Learning Specifications for Labelled Patterns*

Nicolas Basset[1], Thao Dang[1], Akshay Mambakam[1], and José Ignacio Requeno Jarabo[2]

[1] VERIMAG/CNRS, University Grenoble Alpes, Grenoble, France
nicolas.basset1|thao.dang|Akshay.Mambakam@univ-grenoble-alpes.fr
[2] Department of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
jirj@hvl.no

Abstract. In this work, we introduce a supervised learning framework for inferring temporal logic specifications from labelled patterns in signals, so that the formulae can then be used to correctly detect the same patterns in unlabelled samples. The input patterns that are fed to the training process are labelled by a Boolean signal that captures their occurrences. To express the patterns with quantitative features, we use parametric specifications that are increasing, which we call Increasing Parametric Pattern Predictor (IPPP). This means that augmenting the value of the parameters makes the predicted pattern true on a larger set. A particular class of parametric specification formalisms that we use is Parametric Signal Temporal Logic (PSTL). One of the main contributions of this paper is the definition of a new measure, called ϵ-count, to assess the quality of the learned formula. This measure enables us to compare two Boolean signals and, hence, quantifies how much the labelling signal induced by the formula differs from the true labelling signal (e.g. given by an expert). Therefore, the ϵ-count can measure the number of mismatches (either false positives or false negatives) up to some error tolerance ϵ. Our supervised learning framework can be expressed by a multicriteria optimization problem with two objective functions: the minimization of false positives and false negatives given by the parametric formula on a signal. We provide an algorithm to solve this multi-criteria optimization problem. Our approach is demonstrated on two case studies involving characterization and classification of labeled ECG (electrocardiogram) data.

Keywords: Signal Pattern Matching · Monotonic Specification Learning · Pareto Multi-criteria Optimization · Signal Temporal Logic

1 Introduction

Complex systems consist of various inter-connected components for which rigorous modelling is difficult. Due to technological advances a large amount of data

* Supported by the UGA Project SYMER.

from such systems is available. However, to ensure that systems behave correctly, formal specifications defining the intended behaviour are needed. Data-driven modelling involves the process of learning models and specifications of systems from the traces they generate. Once learnt they can be used for analysing and monitoring these systems. This is particularly useful when rigorous mathematical models based on first principles are difficult to obtain.

In this context, supervised learning involves designing a specification from a given set of labelled signals, so that the specification is later used to label signals via monitoring. One approach is to start from nothing but a sample of labelled signals and learn a logical specification, essentially by enumerating formulae of increasing size (using suitable heuristics) to come up with one that is good enough w.r.t. the sample. A more suitable approach is to exploit prior knowledge made available in the form of a parametric specification. For instance, an engineer observing the behaviours of a concrete collection of cars would ask for the parameter valuations p_1 and p_2 for the following emergency brake pattern: "the car can pass from speed 30 m/s to p_1 m/s within less than p_2 seconds."

Our work is of the second kind, following the trend initiated by [6] with parametric specifications written in Parametric Signal Temporal Logic (PSTL). We are inspired by several works on PSTL [6,10] whose aim was to compute the validity domain of a parametric formula, i.e., the set of parameter valuations that makes the formula true on a (or a set of) signal. Though in our experiments we use PSTL with the extended semantics of [7], our framework is not specific to it and can be applied to other specification formalisms. To provide a generic approach which is not tied to a specific specification formalism, we introduce the notion of parametric pattern predictors (PPP). A PPP is a parametric operator Ψ_p that transforms unlabelled signal s to a labelling Boolean signal $\Psi_p(s)$ that is true on time points where the pattern is predicted. We focus our attention on PPPs that are increasing: when the value of p increases for any given signal s, the set of time points where $\Psi_p(s)$ is true expands.

In our framework, we allow the learned specification Ψ_p to produce some false positives and false negatives on parts of the training signals, i.e., there are time points where Ψ_p predicts a pattern while there is none, or misses it. We are interested in computing several sets of parameter valuations (called *solution sets*[3]) which ensure that the "quantities" of false positive and/or false negatives are lower than given bounds. To define such quantities, we can use neither counts of time points or of intervals nor the Lebesgue measure since, as we will see later, these measures are not suitable. Instead we adapt the notion of ϵ-separated set from information theory [17] to propose a new measure, called ϵ-count, with suitable properties (Prop. 1). Our method for computing solution sets is similar to the method for approximating monotonic validity domains, proposed in [6]. The main difference is that the constraints on false positives and false negatives

[3] Since a learned specification is allowed not to be valid everywhere on a signal, we prefer to use the new term of solution set rather than validity domain.

involve two sets monotonic in opposite directions. To this end, we develop an algorithm that computes the intersection of an upset and a downset in $\mathbb{R}^n$.

The main contributions of the paper can be summarized as follows:

- A generic framework of learning parameter valuations for increasing parametric pattern predictors with quantitative constraints on false positives and false negatives.
- A measure called ϵ-count for expressing "how often" a Boolean signal is true and its application to extend the quantitative notions of false positives and false negatives to Boolean labelling signals.
- An algorithm to compute the intersection of an upset and a downset that are queried from a membership oracle.

Section 2 presents our specification learning framework. Section 3 describes the algorithm that computes the intersection of an upset and a downset in $\mathbb{R}^n$. Section 4 demonstrates our approach on two case studies involving ECG signals.

Related work on PSTL. Parameters in PSTL can be used to express constraints both on values and time bounds. They are called space and timing parameters respectively in [10]. In [6] two different methods for computing validity domains for PSTL formulae are presented. The first method demonstrates how exact validity domains can be computed using quantifier elimination, in principle. Though complete and exact, the main drawback of this approach is the exponential worst case complexity in nested depth of formulae. The second method computes approximations of monotonic validity domains using query functions. This method forms the foundation of our contributions regarding monotonic validity domains. Another method which computes validity domains recursively is proposed in [10]. This method deals only with space parameters and leaves handling timing parameters for future work.

Other works which utilize PSTL for the tasks of clustering and classification are as follows. They concentrate on extracting features and computing a single solution rather than complete validity domains. In [27], template PSTL formulae are used to extract features. These features are then used in an unsupervised learning context to cluster traces. In [28], Hausdorff distance based on monotonic validity domains boundaries [21] is used as a distance metric for traces. Clustering was used to generate labelling and then construct specifications from monotonic PSTL templates. In [23], monotonic PSTL formulae are enumerated using formula signatures. Computation of validity domain boundaries [21] is combined with checks for misclassification rate for parameter estimation. The resulting algorithm is used to search for an STL formula to classify traces. Another enumeration based method for classifying traces using robustness value based decision trees is proposed in [22]. Grid sampling is used to estimate timing parameters. Both the aforementioned enumeration based methods deal with learning classifiers from example labelling (i.e. supervised learning). In [16], parameter estimation for PSTL is formulated as multi-objective optimization with

respect to robustness. For inferring the structure of STL formulae in the absence of templates, they propose an incremental construction approach.

It is to be noted that we explicitly capture certain features using the quantitative semantics of extended STL [7]. This simplifies the task by avoiding their encoding as unknown parameters.

Other related work. Temporal logic and timed automata provide a framework to describe and reason about occurrence of events and their correlations in time. Unsupervised learning of hybrid timed automata from real-valued signals was investigated in [29]. In [13] and [20] Timed Regular Expressions (TRE) and LTL specifications respectively are mined from system traces using formula templates and event binding. Quantitative Regular Expressions (QRE) have been used to express specifications for arrhythmia-detection algorithms [4]. Recently, shape expressions have been proposed for learning specifications and features from signals [26]. The problem of learning Linear Temporal Logic (LTL) formulae without any requirements of a priori information in the form of formula templates has been recently explored in [25]. Learning STL specifications using different restrictions on the syntax has been studied in a series of papers by others. A subclass of STL called reactive STL is investigated in [18]. The formulae in this subclass are enumerated by defining a partial order and simulate annealing is used for parameter estimation. Another sub-class named inference PSTL is proposed in [19] for learning formulae that detect anomalies. A decision tree approach combined with a restricted set of PSTL primitives using impurity measures is proposed in [11].

2 Specification Learning Framework

Before introducing our specification learning framework we need few preliminaries on signals and partial order on $\mathbb{R}^n$.

Signals. A *signal* s is a function from $\mathbb{R}$ to $\mathbb{R}$. A *Boolean signal* w is a signal that takes its values in $\mathbb{B} = \{0, 1\}$, with the common interpretation of 1 and 0 as true and false. The *support* of a signal w denoted by $\mathtt{supp}(w)$ is the smallest closed set that contains the set $\{t \mid w(t) \neq 0\}$. We consider only signals with bounded support (aka. compact support[4]). The signal $t \mapsto 0$ which is always false is denoted by $\mathbf{0}$.

Partial order on $\mathbb{R}^n$. Given two vectors $p, q \in \mathbb{R}^n$, we say that p is lower than q, denoted by $p \leq q$, if $\forall i, p_i \leq q_i$. A set $\overline{X}$ is an *upset* if for all $p, q \in \mathbb{R}^n$ such that $p \leq q$ if $p \in \overline{X}$ then $q \in \overline{X}$. A set $\underline{X}$ is a *downset* if for all $p, q \in \mathbb{R}^n$ such that

[4] A subset of $\mathbb{R}$ is compact if and only if it is closed and bounded.

$q \leq p$ if $p \in \underline{X}$ then $q \in \underline{X}$. The boundary consisting of all the minimal elements of an upset (or all the maximal elements of a downset) is called a *Pareto front* in the field of multi-criteria optimization. The *box* between two vectors $\underline{x}$ and $\overline{x}$ with $\underline{x} \leq \overline{x}$ is $\lfloor \underline{x}, \overline{x} \rceil = \{y \mid \underline{x} \leq y \leq \overline{x}\}$.

2.1 Parametric Pattern Predictor

The labels of patterns in our problem are modelled using Boolean signals that we call *labelling signals*. A labelling signal λ_s for a signal s being 1 or 0 at a time point indicates respectively the occurrence or absence of a pattern in s at this time point. Particular cases of labelling signals are those whose support is a list of time points where patterns occur. In these cases, a pattern is a discrete event, and several labelling signals can be merged together to form what is called a timed word in timed automata theory [5]. We prefer using Boolean signals in continuous time for two main reasons. We want to allow patterns to have duration, that is their occurrence lasts continuously throughout a time interval (composed thus of uncountable number of points). They can be considered both as input or output signals for monitoring tools for temporal properties in dense time, such as StlEval [7] which we will use for our experiments.

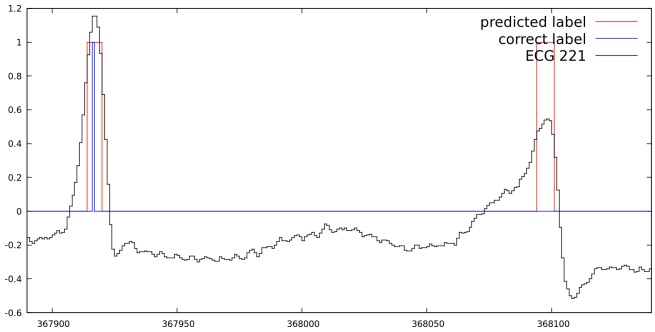


Fig. 1: Showing the single false positive of $\Psi^{ch}_{(8.20,0.64,-0.44)}$ for ECG 221

Example 1. *Consider electro-cardiograms (ECG) from the MIT-BIH Arrhythmia Database of Physionet [15,24]. They are provided with annotations of time-stamps where normal or abnormal peaks occur. The annotations for the normal peaks can be modelled into a labelling signal that is* 1 *when a normal peak occurs and* 0 *everywhere else. A portion of ECG 221 is depicted as in Fig. 1 where the blue labelling signal comes from the database.*

Our aim is to develop a pattern predictor, a tool that generates a labelling signal for a given signal. For ECG signals, it is used to annotate them with normal peaks, such as in Fig. 1 the red signal is predicted by our tool.

Definition 1 ((Increasing) Parametric Pattern Predictor (IPPP)). *A* parametric pattern predictor *(PPP) is a function that maps a vector p of reals to an operator Ψ_p that maps real-valued signals to Boolean-valued signals. Ψ is said* increasing *if for all $p \leq p'$, for all signal s, $\forall t \in [0, l)$ with l the length of s, $\Psi_p(s)(t) \leq \Psi_{p'}(s)(t)$.*

Example 2. *Formula (1) specified in the extended STL[5] [7] gives a simple and rough characterization of a normal ECG peak. $\Psi^{ch}_{(p_1,p_2,p_3)}(s)(t) = 1$ if the maximum of s on $[t - p_1, t + p_1]$ is above $-p_3$, and its variation is within the bound p_2 on $[t - c, t - p_1]$ and on $[t + p_1, t + c]$. The parameter domains are $p_1 \in [0, 70]$, $p_2 \in [0, 1]$ and $p_3 \in [-1, 0]$. Here, $c = 70$ is a constant representing an upper limit on p_1. Note that if one increases (p_1, p_2, p_3), the property is easier to achieve.*

$$\Psi^{ch}_{(p_1,p_2,p_3)} := ((\mathrm{Max}_{[-\mathrm{c},-\mathrm{p_1}]}\, s - \mathrm{Min}_{[-\mathrm{c},-\mathrm{p_1}]}\, s) \leq p_2) \wedge ((\mathrm{Max}_{[-\mathrm{p_1},\mathrm{p_1}]}\, s) \geq -p_3) \wedge ((\mathrm{Max}_{[\mathrm{p_1},\mathrm{c}]}\, s - \mathrm{Min}_{[\mathrm{p_1},\mathrm{c}]}\, s) \leq p_2) \quad (1)$$

We remark again that although our work uses the extended STL [7], our framework can be applied to other specification formalisms. Indeed, many matching problems can be cast into an IPPP, for instance matching as closely as possible a signal for the longest time possible. More formally, we can define an IPPP Ψ^{π} such that $\Psi^{\pi}_{(p_1,p_2)}(s)$ is 1 at time t if the signal s restricted on the interval $[t, t + T - p_1]$ is point-wise p_2-close to a given signal π (representing a shape of interest), that is, $\forall t' \in [0, T - p_1], |s(t + t') - \pi(t')| \leq p_2$. The idea of matching such predefined shapes is inspired by the work on shape expression [26].

2.2 Quantifying Mismatches via ϵ-count

A labelled signal (s, λ_s) is a pair of signal s and labelling signal λ_s. We aim at learning parameters p for an IPPP Ψ_p so that for every given labelled signal (s, λ_s), the labelling signals $\Psi_p(s)$ and λ_s should match together as much as possible. We measure two kind of mismatches by measuring "how often" the two following signals are true. The *false positive signal* $\neg\lambda_s \wedge \Psi_p(s)$ indicates when the predictor predicts an occurrence when there is none. The *false negative signal* $\lambda_s \wedge \neg\Psi_p(s)$ indicates when the predictor misses an actual occurrence.

The phrase "how often" may make one think of counting events like occurrences of a peak. However we cannot count the points where a Boolean formula is true since they are in general uncountable. Counting the intervals where a Boolean signal is true is also problematic since it is not always increasing with the support of the signal. For example, a Boolean signal defined as $b(t) := s(t) < p$ has support that increases with p, but such interval counting is not monotonically increasing with p.Also there can be infinitely many intervals. Last but not

[5] Here and in the rest of the paper we slightly simplified the syntax of [7] by replacing $(\mathrm{On}_{[a,b]}\, \mathrm{Max}\, s)$ by $(\mathrm{Max}_{[\mathrm{a},\mathrm{b}]}\, s)$ whose value in t is $\max_{t' \in [t+a, t+b]} s(t')$.

least, the most standard measure of subsets of $\mathbb{R}$ is the Lebesgue's measure. This is not convenient for our purpose because a signal whose support is the disjoint union of many intervals of almost-null measure which are quite far apart will entails a small measure while for such a signal we want instead a big "count" because it can represent the number of mismatches. In this work we introduce the notion of ϵ-count, inspired by the notions of ϵ-separated sets and ϵ-capacity proposed in [17].

Definition 2 (ϵ-separated set and ϵ-count). *Given a boolean signal w, a set S of reals is ϵ-separated w.r.t. w if $S \subseteq \mathit{supp}(w)$ and for every $t, t' \in S$ with $t \neq t'$, it holds that $|t - t'| \geq \epsilon$. The ϵ-count of a signal w is $c_\epsilon(w) = \max\{|S| \mid S \text{ is } \epsilon\text{-separated w.r.t. } w\}$.*

Proposition 1. *The ϵ-count of a signal w is determined in a greedy manner with the following recursive equations: $c_\epsilon(\mathbf{0}) = 0$ and $c_\epsilon(w) = 1 + c_\epsilon(w')$ where $w'(t) = 0$ if $t < \epsilon + \min(\mathit{supp}(w))$ and $w'(t) = w(t)$ otherwise.*

Proof. Intuitively, the support of the signal w' is obtained from that of the signal w by removing an interval of length not greater than ϵ, meaning that in the difference between the two supports we cannot put two points that are ϵ-separated. More formally, let S be a maximal ϵ-separated set w.r.t. w and $t_{\min}$ be the minimum of S, we first show that the set $S' = S \setminus \{t_{\min}\}$ is ϵ-separated w.r.t. w'. Every point t of this set satisfies $|t - t_{\min}| \geq \epsilon$ and thus $t > t_{\min} + \epsilon$. This means that $S' \subseteq \mathtt{supp}(w')$. Hence $|S'| \leq c_\epsilon(w')$. Since $|S'| = |S| - 1$ and S is maximal we have that $c_\epsilon(w) - 1 \leq c_\epsilon(w')$. It remains to show the inequality $c_\epsilon(w) - 1 \geq c_\epsilon(w')$ to establish the equality. Let now S' be a maximal ϵ-separated w.r.t. w' and let $S = S' \cup \{\min(\mathtt{supp}(w))\}$. Then S is ϵ-separated w.r.t. w, from which we know that $|S| \leq c_\epsilon(w)$. Since $|S| = |S'| + 1 = c_\epsilon(w') + 1$, we obtain the suited inequality. □

The following proposition states useful properties of the ϵ-count. The first two statements are not hard to see. We prove only the third statement.

Proposition 2. *1. The ϵ-count is null iff it is applied to the constant signal $\mathbf{0}$.*
2. The ϵ-count is increasing: if $w \leq w'$ then $c_\epsilon(w) \leq c_\epsilon(w')$.
3. The ϵ-count satisfies a triangular inequality: $c_\epsilon(w \vee w') \leq c_\epsilon(w) + c_\epsilon(w')$.

Proof. Let S'' be a maximal ϵ-separated set w.r.t. $w \vee w'$ so that $|S''| = c_\epsilon(w \vee w')$. Let $S = S'' \cap \mathtt{supp}(w)$. The set S is hence ϵ-separated w.r.t. w and we thus have $|S| \leq c_\epsilon(w)$. Let $S' = S'' \setminus S$ so that in particular $|S''| = |S| + |S'|$. S' is included in $\mathtt{supp}(w \vee w') \setminus \mathtt{supp}(w)$ so it is included in $\mathtt{supp}(w')$. The set S' is hence ϵ-separated w.r.t. w' and we thus have $|S'| \leq c_\epsilon(w')$. We can conclude $c_\epsilon(w \vee w') = |S''| = |S| + |S'| \leq c_\epsilon(w) + c_\epsilon(w')$. □

2.3 Parameter Identification Problems

Given bounds $\mathtt{f}_+, \mathtt{f}_-$ on the allowed ϵ-count of false-positives and false-negatives, we are interested in the following three sets:

$$\mathtt{Dom}{+}(\Psi, \mathcal{S}, \mathtt{f}_+) = \{p \mid \forall (s, \lambda_s) \in \mathcal{S}, c_\epsilon(\Psi_p(s) \wedge \neg\lambda_s) \leq \mathtt{f}_+\}, \tag{2}$$

$$\mathtt{Dom}{-}(\Psi, \mathcal{S}, \mathtt{f}_-) = \{p \mid \forall (s, \lambda_s) \in \mathcal{S}, c_\epsilon(\neg\Psi_p(s) \wedge \lambda_s) \leq \mathtt{f}_-\}, \tag{3}$$

$$\mathtt{DomInter}(\Psi, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-) = \mathtt{Dom}{+}(\Psi, \mathcal{S}, \mathtt{f}_+) \cap \mathtt{Dom}{-}(\Psi, \mathcal{S}, \mathtt{f}_-). \tag{4}$$

For convenience, we call them respectively the *positive*, *negative* and *intersection* solution sets. It is also of great interest to compute the set of couples $(\mathtt{f}_+, \mathtt{f}_-)$, called *set of feasible error bounds*, for which a solution exists:

$$\mathcal{P}(\Psi, \mathcal{S}) = \{(\mathtt{f}_+, \mathtt{f}_-) \mid \mathtt{DomInter}(\Psi, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-) \neq \emptyset\}. \tag{5}$$

In addition, we are interested in a relaxed version of the identification problem for false positive bounding, by tolerating a difference of σ time units in matching the labels. This can be done by replacing λ_s with the signal[6] $\mathrm{F}_{[-\sigma,\sigma]}\,\lambda_s$ in (2). More concretely, the solution set of the corresponding σ-relaxed problem is:

$$\mathtt{Dom}{+}^\sigma(\Psi, \mathcal{S}, \mathtt{f}_+) = \{p \mid \forall (s, \lambda_s) \in \mathcal{S}, c_\epsilon(\Psi_p(s) \wedge \neg\,\mathrm{F}_{[-\sigma,\sigma]}\,\lambda_s) \leq \mathtt{f}_+\}.$$

Hence, the corresponding relaxed version of the intersection solution set (4) is

$$\mathtt{DomInter}^\sigma(\Psi, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-) = \mathtt{Dom}{+}^\sigma(\Psi, \mathcal{S}, \mathtt{f}_+) \cap \mathtt{Dom}{-}(\Psi, \mathcal{S}, \mathtt{f}_-). \tag{6}$$

Note that $\mathtt{Dom}{+}(\Psi, \mathcal{S}, \mathtt{f}_+)$ is a downset and $\mathtt{Dom}{-}(\Psi, \mathcal{S}, \mathtt{f}_-)$ is an upset (see the beginning of Section 2) because Ψ is increasing. Sets of this kind can be learned from membership queries as proposed in [8,21]. The set $\mathtt{DomInter}(\Psi, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-)$ is the intersection of an upset and a downset, we thus face a new problem that we address in Section 3. The set $\mathcal{P}(\Psi, \mathcal{S})$ is an upset and its minimal elements form a Pareto front. We compute it via membership-queries for couples $(\mathtt{f}_+, \mathtt{f}_-)$. They are done via non-emptiness checking of $\mathtt{DomInter}(\Psi, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-)$ which is an easier problem than computing the whole set.

3 Intersecting an Upset and a Downset in $\mathbb{R}^n$

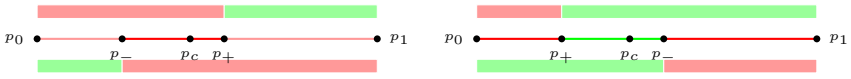
In this section, we describe our algorithm for estimating the intersection of an upset and a downset in $\mathbb{R}^n$ which is required to compute $\mathtt{DomInter}(\Psi, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-)$. The upset and downset are accessed via membership oracles, that is, two Boolean-valued functions $\rho_+ : \mathbb{R}^n \to \mathbb{B}$ and $\rho_- : \mathbb{R}^n \to \mathbb{B}$ which are respectively monotonically increasing and decreasing with respect to the input.

A point where ρ_+ and ρ_- are both 1 (resp. 0) is called a positive (resp. negative) intersection point. Our approach involves intersection search on the diagonal of a hyper-rectangular parameter space.

[6] where $(\mathrm{F}_{[-\sigma,\sigma]}\,\lambda_s)(t) = 1$ iff $\exists t' \in [t-\delta, t+\delta], s(t') = 1$.

Algorithm 1 builds on linear intersection search to compute the positive intersection of an upset and a downset for the multi-dimensional case. An alternative approach is to compute separately the two sets and then their intersection. Computing directly the intersection has the advantage of quickly eliminating the regions that surely do not contain a solution to focus on examining the rest. We can also modify Algorithm 1 to make queries about emptiness of the intersection without computing it exhaustively.

Intersection on a line and expansion. The procedure *boundary* finds the Pareto boundary of a monotonically increasing function on a given line using the classical idea of binary search. The procedure *intersect* finds the intersection of two monotonic Boolean functions ρ_+ (increasing) and ρ_- (decreasing) on a line $\langle \underline{x}, \overline{x} \rangle$. Before starting intersection search on a line, by simple queries on the endpoints we can sometimes altogether discard (o_c=*discard*) or fully accept (o_c=*accept*) the bounding hyper-rectangle. This happens when the hyper-rectangle is wholly contained in a negative or a positive intersection. When this is not the case, we query for the values of ρ_+ and ρ_- at the midpoint. If a point in the intersection is found we return with the result on whether it is positive (o_c=*splitpos*) or negative (o_c=*splitneg*). Otherwise, we continue the search recursively by discarding the half segment not containing an intersection. This is possible because ρ_+ and ρ_- are monotonically increasing and decreasing respectively. In this way we end up either finding an intersection or returning a line segment of length equal to an error bound ε containing the intersection (o_c=*notfound*).On a line (p_0, p_1), we can have three outcomes of the search. The first two outcomes are when a point p_c in the positive intersection (Fig. 2b) or the negative intersection (Fig. 2a) is found. For these cases, we can divide the line into two segments (p_0, p_c) and (p_c, p_1). On these segments we can apply the classical binary search to find the Pareto fronts corresponding to the monotonically decreasing and monotonically increasing functions. We call this operation an *expansion*. In Fig. 2a,2b, the points p_+, p_- represent the points where the Pareto fronts for the monotonically increasing and decreasing functions respectively intersect with the line (p_0, p_1). The third and last case is when no intersection has been found.



(a) Negative intersection. (b) Positive intersection.

Fig. 2: Intersection on a line.

Decomposing the box and continuing the search. Algorithm 1 uses the result of the binary intersection search on the diagonal of a box to deduce which

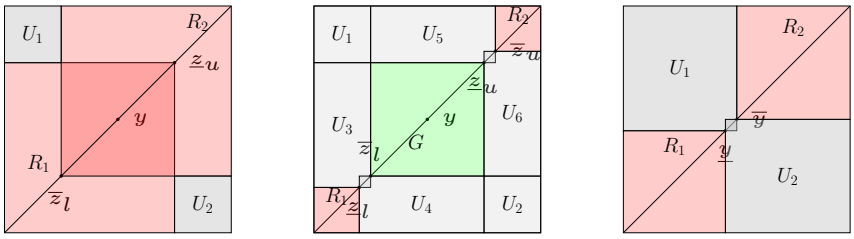
regions (inside the box) do or do not contain a solution and which are undecided. Then, it decomposes the undecided region into sub-boxes and recursively processes the resulting sub-boxes (see Fig. 3)There are three cases:

- *No intersection* has been found (see Fig. 3c). As a result of monotonicity, we know that the sub-boxes R_1 and R_2 do not contain a solution, and proceed with the remaining region which is decomposed into two overlapping sub-boxes U_1 and U_2. This decomposition is formulated in Section 3.1.
- A *negative intersection* has been found (see Fig. 3a). We can identify a line segment on the diagonal where a solution can not exist and deduce that the regions R_1 and R_2 do not contain a solution. The decomposition of the undecided region leads to two sub-boxes U_1 and U_2 (see Section 3.2).
- A *positive intersection* has been found (see Fig. 3b). We obtain the sub-boxes U_1 and U_2 as in the previous cases but use the procedure in Section 3.1 twice to obtain overlapping sub-boxes U_3, U_4, U_5 and U_6.

The decompositions into non-overlapping and overlapping sub-boxes are denoted by I_{nov} and I_{ov} in Algorithm 1 and explained in detail in Sections 3.1 and 3.2.

Algorithm 1 Pareto front intersection algorithm

Input: A box $X = \lfloor \mathbf{0}, \mathbf{1} \rceil$; ρ_+ and ρ_- are respectively monotonically increasing and decreasing Boolean-valued functions; error bounds δ and ε.
Output: A set of boxes S containing the positive intersection of ρ_+ and ρ_- and a set L representing the undecided region such that $|L| \leq \delta$. All sets are represented by unions of boxes.
$L = \{X\}$; $S = \emptyset$ ▷ initialization
repeat
  pop first $\lfloor \underline{x}, \overline{x} \rceil \in L$ ▷ take the largest box
  $\{\langle \underline{y}, \overline{y} \rangle, o_c\} = intersect(\langle \underline{x}, \overline{x} \rangle, \rho_+, \rho_-, \varepsilon)$ ▷ intersect search on the diagonal
  if $o_c == splitpos$ **then** ▷ found a positive intersection.
    $\langle \underline{z}_l, \overline{z}_l \rangle = boundary(\langle \underline{x}, \underline{y} \rangle, \rho_+, \varepsilon)$
    $\langle \underline{z}_u, \overline{z}_u \rangle = boundary(\langle \overline{y}, \overline{x} \rangle, \neg\rho_-, \varepsilon)$
    $S = S \cup \lfloor \overline{z}_l, \underline{z}_u \rceil$
    $L = L \cup I_{nov}(\underline{x}, \overline{x}, \overline{z}_l, \underline{z}_u) \cup I_{ov}(\underline{x}, \underline{z}_u, \underline{z}_l, \overline{z}_l) \cup I_{ov}(\overline{z}_l, \overline{x}, \underline{z}_u, \overline{z}_u)$ ▷ see Fig. 3b
  else if $o_c == splitneg$ **then** ▷ found a negative intersection.
    $\langle \underline{z}_l, \overline{z}_l \rangle = boundary(\langle \underline{x}, \underline{y} \rangle, \neg\rho_-, \varepsilon)$
    $\langle \underline{z}_u, \overline{z}_u \rangle = boundary(\langle \overline{y}, \overline{x} \rangle, \rho_+, \varepsilon)$
    $L = L \cup I_{nov}(\underline{x}, \overline{x}, \underline{z}_l, \underline{z}_u)$ ▷ see Fig. 3a
  else if $o_c == accept$ **then**
    $S = S \cup \lfloor \underline{x}, \overline{x} \rceil$
  else if $o_c \neq discard$ **then** ▷ no intersection found
    $L = L \cup I_{ov}(\underline{x}, \overline{x}, \underline{y}, \overline{y})$ ▷ see Fig 3c.
  $\texttt{Vol}(L) = \texttt{Vol}(X) - \texttt{Vol}(Z)$
until $\texttt{Vol}(L) \leq \delta$
return S, L



(a) Negative intersection (b) Positive intersection (c) No intersection found

Fig. 3: Illustration of sub-boxes.

Before continuing, we need a formal definition of sub-boxes resulting from subdivision based on points $\underline{y} < \overline{y}$ on the diagonal of a box $\lfloor \underline{x}, \overline{x} \rceil$. These sub-boxes are products of sub-intervals I_{α_i} where for each dimension, their bounds are taken among the following sequence of coordinates $\underline{x}_i < \underline{y}_i < \overline{y}_i < \overline{x}_i$.

Definition 3 (Sub-interval encoding). *A sub-interval $I_{\alpha_i} \subseteq [\underline{x}, \overline{x}]$ is encoded by its subscript $\alpha_i \in \{l, m, u, \mathrm{U}, \mathrm{L}, \mathrm{T}\}$ which is a letter such that $\alpha_i = l$ for the lower interval $[\underline{x}_i, \underline{y}_i]$; $\alpha_i = \mathrm{U}$ for its complement[7] $[\underline{y}_i, \overline{x}_i]$; $\alpha_i = u$ for the upper interval $[\overline{y}_i, \overline{x}_i]$; $\alpha_i = \mathrm{L}$ for its complement $[\underline{x}_i, \overline{y}_i]$; $\alpha_i = m$ for the middle interval $[\overline{y}_i, \underline{x}_i]$; and $\alpha_i = \mathrm{T}$ for the whole interval $[\underline{x}_i, \overline{x}_i]$.*

Definition 4 (Sub-boxes). *Given $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{l, m, u, \mathrm{U}, \mathrm{L}, \mathrm{T}\}^n$ and four n-dimensional points $\underline{x} = (\underline{x}_1, \ldots, \underline{x}_n)$, $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$, $\underline{y} = (\underline{y}_1, \ldots, \underline{y}_n)$, $\overline{y} = (\overline{y}_1, \ldots, \overline{y}_n)$ such that $\underline{x} \leq \underline{y} \leq \overline{y} \leq \overline{x}$, the sub-box of $\lfloor \underline{x}, \overline{x} \rceil$ induced by α and $\lfloor \underline{y}, \overline{y} \rceil$ is $B_\alpha = \prod_{i=1}^{n} I_{\alpha_i}$ with I_{α_i} defined in Definition 3.*

3.1 Decomposition Into Overlapping Sub-boxes

This decomposition, proposed in [8], is useful when we have to remove from a box $\lfloor \underline{x}, \overline{x} \rceil$ the downward closure of $\underline{y}$ (*i.e.* $B_{(l,\ldots,l)} = \lfloor \underline{x}, \overline{y} \rceil$) and the upward-closure of $\overline{y}$ (*i.e.* $B_{(u,\ldots,u)} = \lfloor \underline{y}, \overline{x} \rceil$). The resulting sub-boxes can overlap but this decomposition is only used with points that are ε-close. At least in one dimension i the overlap is restricted to the middle interval $[\underline{y}_i, \overline{y}_i]$ whose length is at most ε leading to a negligible volume when ε is small compared to the length of $[\underline{x}_i, \overline{y}_i]$ and $[\underline{y}_i, \overline{x}_i]$.

Definition 5 (Overlapping sub-boxes). *Let $\underline{x}, \underline{y}, \overline{y}, \overline{x}$ be 4 n-dimensional points with $\underline{x} < \underline{y} < \overline{y} < \overline{x}$. We define*

$$I_{ov}(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \{B_\alpha \mid \alpha \in \mathbb{D}_n\}$$

where $(\mathbb{D}_n)_{n \in \mathbb{N}}$ is a sequence of set of words defined inductively as follows:

$\mathbb{D}_2 = \{\mathrm{LU}, \mathrm{UL}\}, \mathbb{D}_3 = \{\mathrm{LUT}, \mathrm{TLU}, \mathrm{UTL}\}$, *and for $n \geq 4$* $\mathbb{D}_{n+1} = \mathrm{T}\mathbb{D}_n \cup \mathrm{LU}^n \cup \mathrm{UL}^n$.

[7] Strictly speaking it is the topological closure of the complement. We avoid introducing open intervals so as to avoid dealing with boxes with partially open boundary. The overlap it causes is harmless since being a null-volume set.

As an example $\mathbb{D}_4 = \{\text{TLUT}, \text{TTLU}, \text{TUTL}, \text{LUUU}, \text{ULLL}\}$

Proposition 3. *The set $I_{nov}(\underline{x}, \overline{x}, \underline{y}, \overline{y})$ contains $(2n-3)$ boxes whose union is the complement in $\lfloor \underline{x}, \overline{x} \rceil$ of $B_{(l,\ldots,l)} \cup B_{(u,\ldots,u)}$.*

3.2 Decomposition Into Non-overlapping Sub-boxes

In case a negative intersection has been found the set of points in the upward-closure of $\underline{y}$ should be removed. This is the sub-box $B_{(\text{U},\ldots,\text{U})} = \lfloor \underline{y}, \overline{x} \rceil$. The same reasoning holds for the downward-closure of $\overline{y}$ which is $B_{(\text{L},\ldots,\text{L})} = \lfloor \underline{x}, \overline{y} \rceil$.

Definition 6 (Non-overlapping sub-boxes). *We define $\mathbb{A}_n$, $\mathbb{C}_n$, $\mathbb{E}_n$ recursively as follows*

$$\mathbb{E}_0 = \mathbb{A}_0 = \mathbb{C}_0 = \emptyset \text{ and for } n \geq 1$$

$$\mathbb{E}_{n+1} = m\mathbb{E}_n \cup u\mathbb{A}_n \cup l\mathbb{C}_n, \quad \mathbb{A}_{n+1} = l\text{T}^n \cup \text{U}\mathbb{A}_n, \quad \mathbb{C}_{n+1} = u\text{T}^n \cup \text{L}\mathbb{C}_n.$$

Let $\underline{x}, \underline{y}, \overline{y}, \overline{x}$ be 4 n-dimensional points with $\underline{x} < \underline{y} < \overline{y} < \overline{x}$. We define

$$I_{nov}(\underline{x}, \overline{x}, \underline{y}, \overline{y}) = \{B_\alpha \mid \alpha \in \mathbb{E}_n\}.$$

Proposition 4. *The set $I_{nov}(\underline{x}, \overline{x}, \underline{y}, \overline{y})$ contains $(n^2 - n)$ boxes whose union is the complement in $\lfloor \underline{x}, \overline{x} \rceil$ of $B_{(\text{U},\ldots,\text{U})} \cup B_{(\text{L},\ldots,\text{L})}$.*

As an illustration we give $\mathbb{A}_n, \mathbb{C}_n, \mathbb{E}_n$ for the dimensions 1, 2 and 3:

n	1	2	3
$\mathbb{A}_n$	l	$l\text{T} \cup \text{U}l$	$l\text{TT} \cup \text{U}l\text{T} \cup \text{UU}l$
$\mathbb{C}_n$	u	$u\text{T} \cup \text{L}u$	$u\text{TT} \cup \text{L}u\text{T} \cup \text{LL}u$
$\mathbb{E}_n$	$\emptyset$	$lu \cup ul$	$mlu \cup mul \cup ul\text{T} \cup u\text{U}l \cup lu\text{T} \cup l\text{L}u$

4 Experiments

We have implemented the algorithms proposed in this work and incorporated them as additions to ParetoLib [9,2] (a Python library for Pareto-Front learning) and the tool StlEval [7,3]. Implementations in the ParetoLib library only deal with upsets (or downsets). To satisfy this, one can easily replace some of the parameters with their opposite.

We now present some experimental results obtained by using our supervised learning framework to analyse labelled electrocardiogram (ECG). ECG signals capture information about electrical activity of the heart and can help detect anomalies in its functioning. We characterize several features (e.g. peaks and ditches) as parametric specification, and provide the intersection solution set for the involved parameters with the best possible trade-off between false positives and false negatives.

4.1 Learning STL Specifications for Labelled ECGs

We present our results on three ECGs (100, 123, 221) each containing between 1500 to 2500 labelled pulses taken from the MIT-BIH Arrhythmia Database of Physionet [15,24]. We use the formula (1) from Section 2 to characterize a parameter predictor Ψ^{ch} for a normal heart pulse. Given a set $\mathcal{S}$ of labelled ECG signals, the upper bounds $\mathtt{f}_-$ and $\mathtt{f}_+$ on the numbers of false negatives and positives respectively and a matching tolerance value σ, we use the intersection algorithm to find the relaxed intersection solution set $\mathtt{DomInter}^{\sigma}(\Psi^{ch}, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-)$ as defined in (6) (that is the set of parameter values p such that the predictor Ψ^{ch}_p can predict normal heart pulses on the labelled signal set $\mathcal{S}$ with the numbers of false positives and false negatives bounded by $\mathtt{f}_+$ and $\mathtt{f}_-$).

Trade off between false negatives and false positives. Using a modification of Algorithm 1, we query about emptiness of $\mathtt{DomInter}^{\sigma}(\Psi^{ch}, \mathcal{S}, \mathtt{f}_+, \mathtt{f}_-)$ and compute the set $\mathcal{P}(\Psi^{ch}, \mathcal{S})$ of feasible error bounds as defined in (5). We recall that Algorithm 1 explores the parameter space up to a given bound δ on volume. In our experiments we search until we have reduced the volume of the undecided region to V_δ percentage of the total volume of the parameter space. The Pareto front that we obtain asymptotically becomes exact as the value of V_δ tends to zero. In Fig. 4 we show two Pareto front approximations for ECG-100; the front separating the brown and the green regions corresponds to $V_\delta = 1\%$. The Pareto front separating the red and the brown regions corresponds to $V_\delta = 0.1\%$ and is more accurate owing to better exploration.

Looking at Fig. 4, it is pertinent to ask why the predictor (1) cannot match the labelling with better accuracy for ECG-100. Actually, our formula only takes the shape of the heart pulses into account but not their time period. Some heart pulses in ECG-100 are not labelled as normal because they violate the natural rhythm of the heart and arrive too soon or too late. It is thus not possible to distinguish them by considering only the shape. For ECG-221, the predictor (1) can match the labelling with no false negatives and only a single false positive (shown in Fig. 1). For ECG-123, it can match with a single false negative and no false positives.

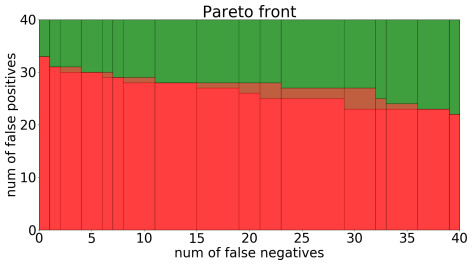


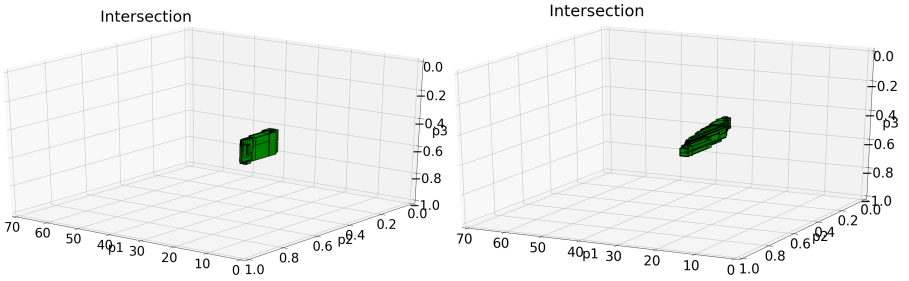
Fig. 4: ECG 100, $V_\delta = 1\%$ vs $V_\delta = 0.1\%$

3D intersections. Once we have computed $\mathcal{P}(\Psi^{ch}, \mathcal{S})$ with adequate accuracy, we can use Algorithm 1 to further explore the parameter space for different values of V_δ, $\mathtt{f}_-$ and $\mathtt{f}_+$. Some exploration results are depicted in Fig. 5 and the associated computation time in Table 1.

Table 1: Computation time for 3D intersection solution sets.

ECG n^0	$\mathtt{f}_-$	$\mathtt{f}_+$	$V_\delta = 1\%$	$V_\delta = 0.1\%$	$V_\delta = 0.01\%$	τ^1
221	0	1	62s	262s	1279s	19s
123	1	0	103s	592s	3189s	36s
100	0	33	758s	5273s	18670s	12s

[1] τ represents the time taken to find the first point in the solution set.



(a) ECG 221,
$V_\delta = 0.01\%$, $\mathtt{f}_- = 0$, $\mathtt{f}_+ = 1$

(b) ECG 123,
$V_\delta = 0.01\%$, $\mathtt{f}_- = 1$, $\mathtt{f}_+ = 0$

Fig. 5: Case study 1: intersection solution sets in 3D

4.2 Classification of ECGs

We now demonstrate the application of our approach to binary classification of signals, using the ECGFiveDays dataset from the UCR Time Series Classification Archive [14]. More concretely, we consider two classes of ECGs taken 5 days apart from the same person and want to find a classifier that can correctly predict given an ECG, on which day it was observed ($\mathtt{day}_1$ or $\mathtt{day}_5$). In [22], an enumerative method over STL is applied to solve this problem in the absence of a priori information. By defining expressive and meaningful features, we show how more informative formulae can be obtained with less training data (23 traces as compared to 300 in [22]). The features are based on the well known theoretical modelling of ECG signal as P and T waves combined with a QRS complex (see e.g. [12]). For each ECG in the dataset, we observe two prominent peaks with a ditch in between. We define STL formulae to quantify some features of the

signal around the ditch and the peaks as shown in Table 2. The range of feature values (D_1,D_5) for $\texttt{day}_1$ and $\texttt{day}_5$ observed in the ECGs are shown in the last two columns of Table 2. They can be computed using a slight modification of classical binary search.

Computation of feature range. We describe briefly the technique we use to compute the range of values taken by a given feature (expressed as an interval). To express the magnitude and timing features listed in Table 2, we use the PSTL formulae given in (7a, 7b) respectively, where the first and the second formulae become monotonically more and less true respectively as parameter p increases. Then using the intersection algorithm we can find the interval containing the feasible values of p (thereby feature range). The parameter c in (7b) is a user-defined upper bound on the feature value; λ is the labelling signal.

$$\mathrm{F}(\lambda \rightarrow \varphi \leq p);\ \mathrm{F}(\lambda \rightarrow \varphi \geq p) \tag{7a}$$

$$\mathrm{F}(\lambda \rightarrow \mathrm{F}_{[0,p]}\, \varphi);\ \mathrm{F}(\lambda \rightarrow \mathrm{F}_{[p,c]}\, \varphi) \tag{7b}$$

Enumeration of formulae and finding a classifier. We rank the features using the measure[8] $m = |D_1 \triangle D_5|/|D_1 \cup D_5|$. We then enumerate all the distinct pairs (φ_i, φ_j) of features using lexicographical ordering over rank. For each pair we use the intersection algorithm to learn the parameters which make the disjunction formula ($\Psi := \varphi_i \vee \varphi_j$) classify correctly the ECGs given in the training data. Let S_1 and S_2 be the labelled signals corresponding to the classes $\texttt{day}_1$ and $\texttt{day}_5$ of ECGs respectively. We compute the intersection of $\texttt{Dom}{-}(\Psi, \mathcal{S}_1, \texttt{f}_-)$ and $\texttt{Dom}{+}(\Psi, \mathcal{S}_2, \texttt{f}_+)$.

Table 2: Features and formulae.

Feature	Formula	D_1 for $\texttt{day}_1$	D_5 for $\texttt{day}_5$
Def. of `peak`	$(s \geq (\mathrm{Max}_{[-10,10]}\ s)) \wedge s \geq 1$	NA	NA
Def. of `ditch` (`dh`)	$(s \leq (\mathrm{Min}_{[-10,10]}\ s)) \wedge s \leq -1$	NA	NA
Depth of the ditch	$(\mathrm{Min}_{[0,136]}\ s) \leq p\ \{or \geq p\}$	(-6.12, -4.767)	(-6.51, -5.71)
Location of the ditch	$\mathrm{F}_{[\theta_1,\theta_2]}\ \texttt{dh}$	(51.00, 58.99)	(51.00, 59.99)
Height of peak 1	$(\mathrm{Max}\ s\ \mathrm{U}\ \texttt{dh}) \leq p\ \{or \geq p\}$	(1.01, 5.42)	(0.77, 3.81)
Location of peak 1	$\mathrm{F}_{[\theta_1,\theta_2]}\ \texttt{peak}$	(48.00, 56.99)	(0.00, 55.99)
Height of peak 2	$\texttt{dh} \wedge ((\mathrm{Max}_{[0,60]}\ s) \leq p)\ \{or \geq p\}$	(1.25, 3.296)	(1.43, 2.58)
Location of peak 2	$\texttt{dh} \wedge \mathrm{F}_{[\theta_1,\theta_2]}\ \texttt{peak}$	(25.00, 30.99)	(23.00, 26.99)

Results for ECG5days classify. Table 3 summarizes our results for five different training and testing configurations. For the first configuration, we use the

[8] $\triangle$ is the symmetric difference of two sets.

original 23 training traces and 861 testing traces from [1] without any changes. For the other configurations we split the training set of size 861 and use one portion for training and the other for testing. The number of traces used for training and testing are mentioned within brackets in the first column of Table 3. For example, in the second row we indicate that for configuration 2 we use 100 traces for training and the remaining 761 for testing. Note that for configuration 4 we use all the 861 traces for training and have no traces for testing. For this configuration, the reason we did not find a solution could be because we required 100% training accuracy. The 0/0 in the column for testing error is because the test set is empty. Note that, for a PSTL formula each parameter valuation in the solution set produces a classifier. Two such classifiers we found, $\Psi^{cl_1}_{(28.3,11.0,4.0)}$ and $\Psi^{cl_2}_{(27.5,1.0,-1.3)}$ have error values 2/861 and 17/861 respectively on the original testing set.

$$\begin{aligned} \Psi^{cl_1}_{(p_1,p_2,p_3)} &:= (\texttt{ditch} \wedge \mathrm{F}_{[p_1,c_2-p_2]}\,\texttt{peak}) \vee (\ (\mathrm{Max}\ \ s\ U\ \texttt{ditch}) \geq p_3) \\ \Psi^{cl_2}_{(p_1,p_2,p_3)} &:= (\texttt{ditch} \wedge \mathrm{F}_{[p_1,c_2-p_2]}\,\texttt{peak}) \vee (\texttt{ditch} \wedge (\mathrm{Max}_{[0,c_2]}\ \ s) \leq -p_3) \end{aligned} \quad (8)$$

Table 3: Results for learning (case study 2). See Formula (8) for Ψ^{cl_1}, Ψ^{cl_2}

Configuration	time (s)			Testing error		Training error	
	$\delta = 10^{-1}$	$\delta = 10^{-2}$	$\delta = 5.10^{-3}$	Ψ^{cl_1}	Ψ^{cl_2}	Ψ^{cl_1}	Ψ^{cl_2}
Confg. 1 (23, 861)	2	184	787	2/861	17/861	0/23	0/23
Confg. 2 (100, 761)	1.5	6	10	2/761	17/761	0/100	0/100
Confg. 3 (300, 561)	2	3	5	2/561	NA[1]	0/300	NA
Confg. 4 (861, 0)	13	79	153	NA	NA	NA	NA
Confg. 5 (861, 0)	5	8.5	12	0/0	NA	2/861	NA

[1] NA: Not Applicable. Parameter search is unsuccessful.

5 Conclusion and Future Work

In this paper, we presented a new method for extracting knowledge from labelled signals based on monotonic parametric specifications. To this end, we introduced the ϵ-count, to measure the amount of mismatch between two Boolean signals (e.g., the Boolean signal induced by the labelled input sample, and the one defined by our learned specification). We then formulated the learning process as a multi-criteria optimization problem with constraints on the ϵ-counts of false positives and false negatives. Finally, we proposed an algorithm to solve this problem based on the intersection of an upset and a downset, and then applied it in particular for learning monotonic PSTL specifications. We demonstrated the performance of our approach on two case studies involving ECG signals.

As future work, we will investigate the computation of the exact or approximate solution sets for non-monotonic parametric specification. To partially solve

this, we can find the minimal set of parameters according to heuristic multi-criteria optimization. However there exist trade-offs among parameters and also between tightness and robustness. Finding tightest parameters for the given training examples might not generalize well. Methods that intelligently explore the parameter space uncovering these trade-offs are needed. Second, we would like to investigate efficient representations for solution sets. We found the need for this when dealing with timing parameters. The formula $\mathrm{F}_{[\tau_1,\tau_2]}\,\varphi$ is monotonic with respect to τ_1 and τ_2, but τ_1 and τ_2 are related by an implicit constraint, $\tau_1 \leq \tau_2$. Replacing multiple occurrences of a parameter with distinct symbols as suggested in [27] might not be straightforward for timing parameters. Consequently, it becomes more difficult to use boxes to represent the solution set. The problem of selecting optimal parameter assignments from the solution set in order to maximize average classification accuracy can also be studied.

References

1. ECGFiveDays data set. http://www.timeseriesclassification.com/description.php?Dataset=ECGFiveDays.
2. Implementation of Pareto front intersection algorithm. Available at https://gricad-gitlab.univ-grenoble-alpes.fr/verimag/tempo/multidimensional_search/-/tree/intersectionAkshay.
3. Implementation of the StlEval ϵ-count operator. Available at https://gricad-gitlab.univ-grenoble-alpes.fr/verimag/tempo/StlEval/-/tree/akshayTest.
4. Houssam Abbas, Alena Rodionova, Konstantinos Mamouras, Ezio Bartocci, Scott A. Smolka, and Radu Grosu. Quantitative regular expressions for arrhythmia detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1586–1597, 2019.
5. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.
6. Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *Proceedings of the 12th International Conference on Runtime Verification*, volume 7687 of *Programming and Software Engineering*, pages 147–160, Berlin, Heidelberg, 2012. Springer.
7. Alexey Bakhirkin and Nicolas Basset. Specification and efficient monitoring beyond STL. In *Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'19, pages 79–97, Cham, 2019. Springer.
8. Alexey Bakhirkin, Nicolas Basset, Oded Maler, and José Ignacio Requeno. Learning pareto front from membership queries. Working paper or preprint, 2019. URL: https://hal.archives-ouvertes.fr/hal-02125140.
9. Alexey Bakhirkin, Nicolas Basset, Oded Maler, and José Ignacio Requeno. ParetoLib: A python library for parameter synthesis. In *Proceedings of the 17th International Conference on Formal Modeling and Analysis of Timed Systems*, volume 11750 of *Theoretical Computer Science and General Issues*, pages 114–120, Cham, 2019. Springer.
10. Alexey Bakhirkin, Thomas Ferrére, and Oded Maler. Efficient parametric identification for STL. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control*, HSCC'18, pages 177–186, New York, NY, USA, 2018. ACM.

11. Giuseppe Bombara, Cristian Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC'16, pages 1–10, New York, NY, USA, 2016. ACM.
12. Taolue Chen, Marco Diciolla, Marta Kwiatkowska, and Alexandru Mereacre. A Simulink hybrid heart model for quantitative verification of cardiac pacemakers. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, HSCC'13, pages 131–136, New York, NY, USA, 2013. ACM.
13. Greta Cutulenco, Yogi Joshi, Apurva Narayan, and Sebastian Fischmeister. Mining timed regular expressions from system traces. In *Proceedings of the 5th International Workshop on Software Mining*, SoftwareMining'16, pages 3–10, New York, NY, USA, 2016. ACM.
14. Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The UCR time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.
15. Ary L. Goldberger, Luis A.N. 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.
16. Susmit Jha, Ashish Tiwari, Sanjit A. Seshia, Tuhin Sahai, and Natarajan Shankar. TeLEx: learning signal temporal logic from positive examples using tightness metric. *Formal Methods in System Design*, 54(3):364–387, 2019.
17. Andrey N. Kolmogorov and Vladimir M. Tikhomirov. ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Matematicheskikh Nauk*, 14(2(86)):386, 1959.
18. Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, HSCC'14, pages 273–282, New York, NY, USA, 2014. ACM.
19. Zhaodan Kong, Austin Jones, and Calin Belta. Temporal logics for learning and detection of anomalous behavior. *IEEE Transactions on Automatic Control*, 62(3):1210–1222, 2017.
20. Caroline Lemieux, Dennis Park, and Ivan Beschastnikh. General LTL specification mining (T). In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, ASE'15, pages 81–92. IEEE, 2015.
21. Oded Maler. Learning monotone partitions of partially-ordered domains (work in progress). Working paper or preprint, 2017. URL: https://hal.archives-ouvertes.fr/hal-01556243.
22. Sara Mohammadinejad, Jyotirmoy V. Deshmukh, and Aniruddh G. Puranic. Mining environment assumptions for cyber-physical system models. In *Proceedings of the 11th ACM/IEEE International Conference on Cyber-Physical Systems (to appear)*, ICCPS'20. IEEE, 2020.
23. Sara Mohammadinejad, Jyotirmoy V. Deshmukh, Aniruddh G. Puranic, Marcell Vazquez-Chanlatte, and Alexandre Donzé. Interpretable classification of time-series data using efficient enumerative techniques. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control (to appear)*, HSCC'20, New York, NY, USA, 2020. ACM.

24. George B Moody and Roger G Mark. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.
25. Daniel Neider and Ivan Gavran. Learning linear temporal properties. In *Proceedings of the 18th International Conference on Formal Methods in Computer Aided Design*, FMCAD'11, pages 1–10, Austin, TX, USA, 2018. ACM.
26. Dejan Nickovic, Xin Qin, Thomas Ferrère, Cristinel Mateis, and Jyotirmoy V. Deshmukh. Shape expressions for specifying and extracting signal features. In *Proceedings of the 19th International Conference on Runtime Verification*, volume 11757 of *Lecture Notes in Computer Science*, pages 292–309, Cham, 2019. Springer.
27. Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, and Sanjit A. Seshia. Logical clustering and learning for time-series data. In *Proceedings of the 29th International Conference on Computer Aided Verification*, volume 10426 of *Theoretical Computer Science and General Issues*, pages 305–325, Cham, 2017. Springer.
28. Marcell Vazquez-Chanlatte, Shromona Ghosh, Jyotirmoy V. Deshmukh, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. Time-series learning using monotonic logical properties. In *Proceedings of the 18th International Conference on Runtime Verification*, volume 11237 of *Lecture Notes in Computer Science*, pages 389–405, Cham, 2018. Springer.
29. Alexander von Birgelen and Oliver Niggemann. Using self-organizing maps to learn hybrid timed automata in absence of discrete events. In *Proceedings of the 22nd IEEE International Conference on Emerging Technologies and Factory Automation*, ETFA'17, pages 1–8. IEEE, 2017.

A One Dimensional Search and Intersection

We first recall the classical binary search algorithm (aka. dichotomic search) and then we give our intersection search algorithm.

Given a Boolean-valued function $\rho : \mathbb{R} \to \mathbb{B}$ which is monotonically increasing with the input, Algorithm 2 computes an interval of any specified length ϵ containing the point on the boundary between the subset of the domain where the function is false and the subset where it is true. See Figure 6 for an illustration of various steps of the algorithm. Algorithm 2 can be extended to the general n-dimensional case [8]. Algorithm 3 computes the intersection of ρ_+ and ρ_- which are monotonically increasing and decreasing Boolean functions respectively.

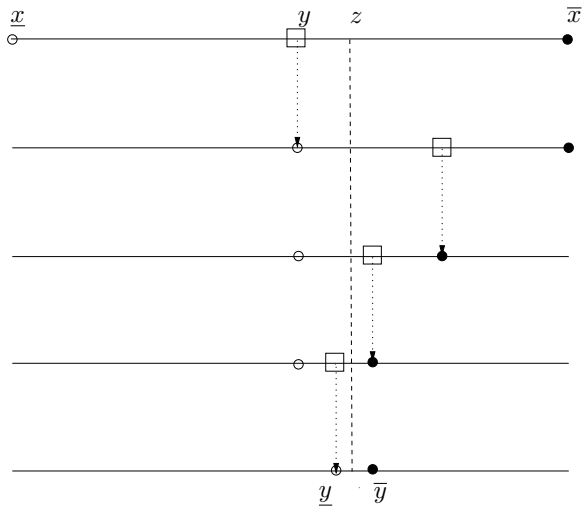


Fig. 6: Binary search and the successive reduction of the uncertainty interval.

Algorithm 2 One dimensional binary search: $boundary(\langle \underline{x}, \overline{x} \rangle, \rho, \varepsilon)$

Input: A line segment $\ell = \langle \underline{x}, \overline{x} \rangle$, ρ a monotonically increasing boolean-valued function and an error bound $\varepsilon \geq 0$.
Output: A line segment $\langle \overline{y}, \underline{y} \rangle$ containing the Pareto boundary of ρ such that $\overline{y} - \underline{y} \leq \varepsilon$.
$\langle \underline{y}, \overline{y} \rangle = \langle \underline{x}, \overline{x} \rangle$
while $\overline{y} - \underline{y} \geq \varepsilon$ **do**
    $y = (\underline{y} + \overline{y})/2$
    if $member(y)$ **then**
        $\langle \underline{y}, \overline{y} \rangle = \langle \underline{y}, y \rangle$ ▷ left sub-interval
    else
        $\langle \underline{y}, \overline{y} \rangle = \langle y, \overline{y} \rangle$ ▷ right sub-interval
return $\langle \underline{y}, \overline{y} \rangle$

Algorithm 3 1D intersection search: $intersect(\langle \underline{x}, \overline{x} \rangle, \rho_+, \rho_-, \varepsilon)$

Input: $\ell = \langle \underline{x}, \overline{x} \rangle$ is a line segment, ρ_+ and ρ_- are monotonically increasing and decreasing Boolean-valued functions; and $\varepsilon \geq 0$ is an error bound.
Output: A line segment $\langle \underline{y}, \overline{y} \rangle$, an enum variable o_c which can take values $\{$*notfound, accept, discard, splitpos, splitneg*$\}$.
$\langle \underline{y}, \overline{y} \rangle = \langle \underline{x}, \overline{x} \rangle$, $o_c =$ *notfound*
if $\rho_+(\overline{y})$ **and** $\rho_-(\underline{y})$ **then**
    return $\langle \underline{y}, \overline{y} \rangle$, $o_c =$ *accept*
if (**not** $\rho_+(\underline{y})$) **and** (**not** $\rho_-(\overline{y})$) **then**
    return $\langle \underline{y}, \overline{y} \rangle$, $o_c =$ *discard*
while $\overline{y} - \underline{y} \geq \varepsilon$ **do**
    $y = (\underline{y} + \overline{y})/2$
    if $\rho_+(y)$ **and** $\rho_-(y)$ **then**
        return $\langle y, y \rangle$, *splitpos* ▷ positive intersection
    else if (**not** $\rho_+(y)$) **and** (**not** $\rho_-(y)$) **then**
        return $\langle y, y \rangle$, *splitneg* ▷ negative intersection
    else if $\rho_+(y)$ **and** (**not** $\rho_-(y)$) **then**
        $\langle \underline{y}, \overline{y} \rangle = \langle \underline{y}, y \rangle$ ▷ left sub-interval
    else if (**not** $\rho_+(y)$) **and** $\rho_-(y)$ **then**
        $\langle \underline{y}, \overline{y} \rangle = \langle y, \overline{y} \rangle$ ▷ right sub-interval
return $\langle \underline{y}, \overline{y} \rangle$, o_c

B Monotonicity, Solution sets and Pareto Fronts.

B.1 Interval count for $b(t) := s(t) < p$ is not monotonic with p.

The Boolean signal $s(t) < 3$ is true on two intervals, while $s(t) < 2$ and $s(t) < 6$ are true on one interval.

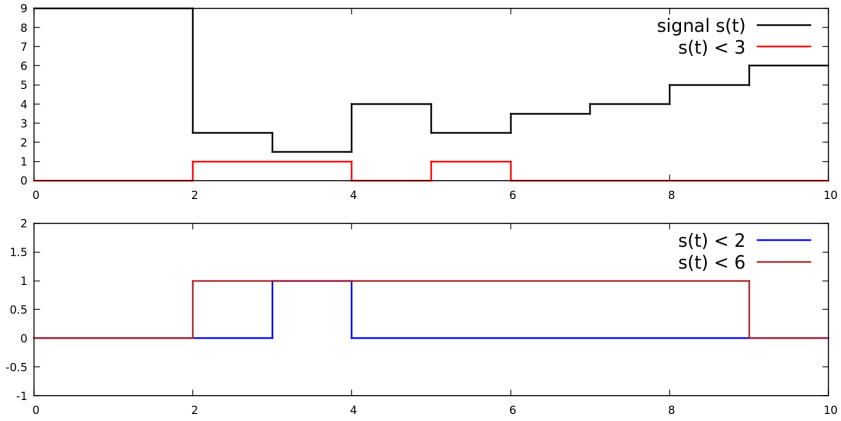


Fig. 7: Non-monotonicity of interval count.

B.2 Solution set for ECG 100 When $\mathtt{f}_- = 0$ and $\mathtt{f}_+ = 33$.

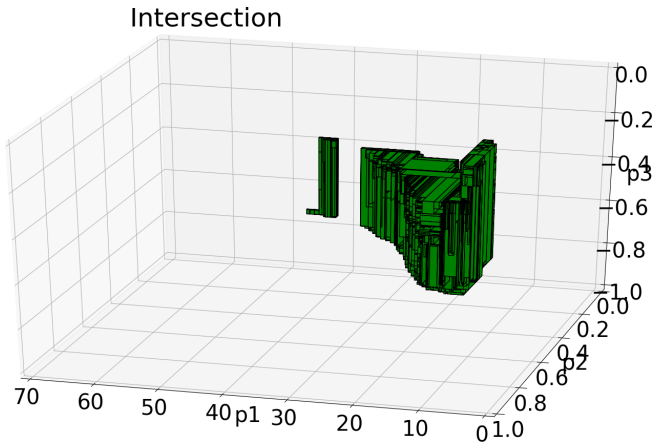
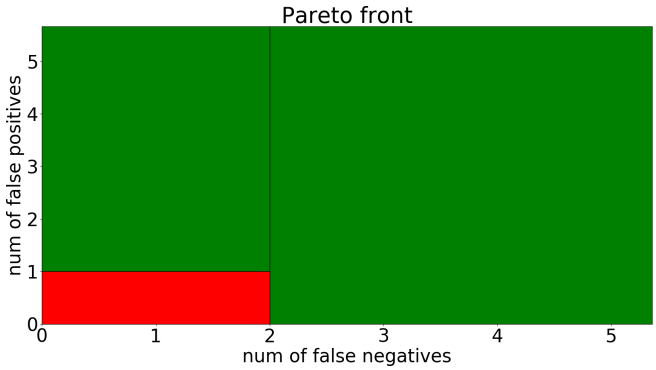
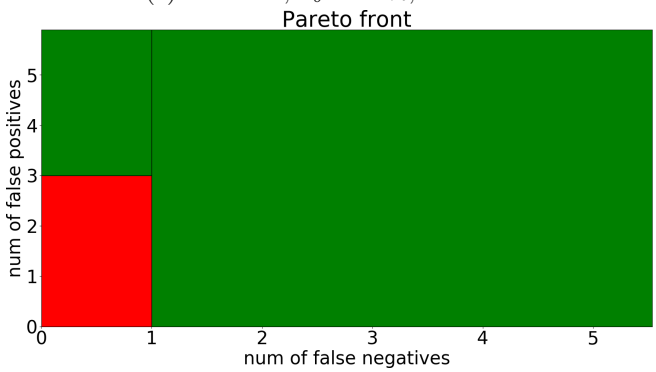


Fig. 8: ECG 100, $V_\delta = 0.01\%$, $\mathtt{f}_- = 0$, $\mathtt{f}_+ = 33$

B.3 Pareto Fronts Between f_+ and f_-.

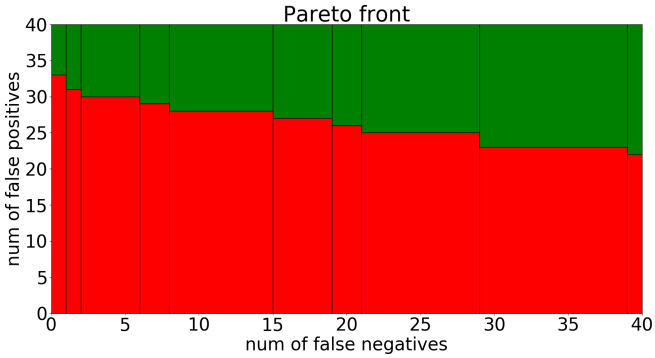


(a) ECG 221, $V_\delta = 0.1\%$, $t \approx 252s$

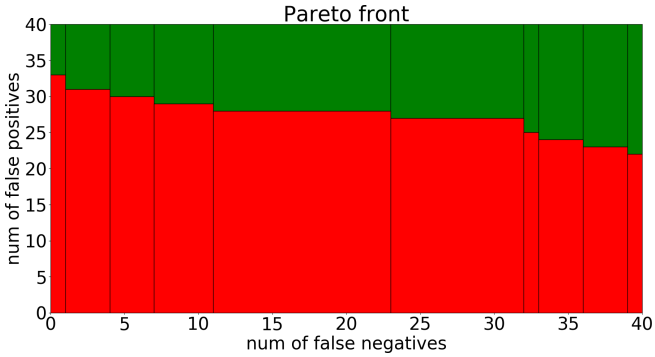


(b) ECG 123, $V_\delta = 0.1\%$, $t \approx 303s$

Fig. 9: Pareto fronts for ECGs 221 and 123



(a) ECG 100, $V_\delta = 0.1\%$, $t \approx 11919s$



(b) ECG 100, $V_\delta = 1\%$, $t \approx 7903s$

Fig. 10: ECG 100: Decreasing V_δ gives a more accurate Pareto front