# SYNTHESIS OF PAST TIME SIGNAL TEMPORAL LOGIC FORMULAS USING MONOTONICITY PROPERTIES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERT ERGÜRTUNA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2020

Approval of the thesis:

**SYNTHESIS OF PAST TIME SIGNAL TEMPORAL LOGIC FORMULAS USING MONOTONICITY PROPERTIES**

submitted by **MERT ERGÜRTUNA** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Ebru Aydın Göl
Supervisor, **Computer Engineering, METU** _____

**Examining Committee Members:**

Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU _____

Assist. Prof. Dr. Ebru Aydın Göl
Computer Engineering, METU _____

Assoc. Prof. Dr. Hüsnü Yenigün
Computer Science & Engineering, Sabancı University _____

Date: 11.06.2020

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Surname: Mert Ergürtuna

Signature :

# ABSTRACT

## SYNTHESIS OF PAST TIME SIGNAL TEMPORAL LOGIC FORMULAS USING MONOTONICITY PROPERTIES

Ergürtuna, Mert
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Ebru Aydın Göl

June 2020, 63 pages

Due to its expressivity and efficient algorithms, Signal Temporal Logic (STL) is widely used in runtime verification, formal control and analysis of time series data. While it is relatively easy to define an STL formula, simulate the system and mark the unexpected behaviors according to the formula as in the testing process, finding an STL formula that would detect the underlying cause of the errors is a complicated process. The main motivation of this thesis is to find a method that would explain the events that lead to the erroneous behavior in the system in an automated, efficient and human-readable way. Since the aim is to find the events the lead to the error in the system, past time signal temporal logic (ptSTL) which deals with the past events is used in the study.

This thesis presents a novel method to find temporal properties that lead to unexpected behaviors from a dataset. The dataset consists of system traces that are labeled at each time point. First, monotonicity properties for ptSTL over the considered dataset is developed. Next using these monotonicity properties, an efficient parameter synthesis method for ptSTL is proposed. Finally, the parameter synthesis method is used in an iterative unguided formula synthesis framework that combines optimized formulas.

In addition, the extension of the proposed approach to a dataset with a single label for each trace is developed. In this part, it is shown that the previously defined monotonicity properties hold and the framework is adapted.

Keywords: Signal Temporal Logic, Formal Methods, Formula Synthesis, Monotonicity, Classification

# ÖZ

## MONOTONLUK ÖZELLİKLERİ KULLANILARAK GEÇMİŞ ZAMANLI SİNYAL ZAMANSAL MANTIK FORMÜLLERİ SENTEZLENMESİ

Ergürtuna, Mert
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ebru Aydın Göl

Haziran 2020 , 63 sayfa

Sinyal Zamansal Mantık(Signal Temporal Logic - STL) verilmli algoritmaları ve anlatım gücü sayesinde çalışma zamanında doğrulama, formel kontrol ve zaman serisi verilerinin analizinde yaygın olarak kullanılmaktadır. Bir STL formülü tanımlayıp, sistemi simüle edip, test süreçlerindeki gibi beklenmeyen davranışları bu formüle göre işaretlemek görece kolayken, hataların altında yatan sebebi bulacak bir STL formülü bulmak karmaşık bir süreçtir. Bu tezin ana motivasyonu, bir sistemde meydana gelen hatalı davranışa sebep olan olayları otomatikleştirilmiş, verimli ve insan tarafından okunabilir şekilde açıklayacak bir yöntem bulmaktır. Bu çalışmada amaç sistemde hatalara yol açan durumları bulmak olduğundan, geçmiş olayları tanımlamakta kullanılan geçmiş zamanlı sinyal sayısal mantık (past time Signal Temporal Logic - ptSTL) kullanılmıştır.

Bu tez bir veri kümesinden, beklenmeyen davranışlara sebep olan zamansal özellikleri bulacak yeni bir yöntem sunmaktadır. Belirtilen veri kümesi her zamanda etiketlenmiş sistem izlerinden oluşmaktadır. İlk olarak, ptSTL formüllerinin bu veri kümesi üzerindeki monotonluk özellikleri tanımlanmıştır. Ardından monotonluk özellikleri

kullanılarak, ptSTL için verimli bir parametre sentezi yöntemi önerilmiştir. Son olarak, parametre sentezi yöntemi ile bulunan optimize formülleri birleştiren yinelemeli ve rehbersiz bir formül sentezleme yapısı geliştirilmiştir.

Ek olarak, önerilen yaklaşımın her iz için tekil etiketi olan veri kümeleri için genişletilmesi sağlanmıştır. Bu kısımda, önceden tanımlanan monotonluk özelliklerinin korunduğu gösterilip, yapı uyarlanmıştır.

Anahtar Kelimeler: Sinyal Zamansal Mantık, Formel Metotlar, Formül Sentezleme, Monotonluk, Sınıflandırma

To my family...

## ACKNOWLEDGMENTS

First of all, I would like to thank to my advisor Assist. Prof. Dr. Ebru Aydın Göl for her endless support. I was able to complete this study with her knowledge and guidance.

I would also like to thank my thesis committee members Prof. Dr. Halit Oğuztüzün and Assoc. Prof. Dr. Hüsnü Yenigün not only for accepting my invitation but also for their helpful and encouraging comments.

I want to also express my gratitude to my parents and my sister for their endless support and for standing by me on every aspect of my life. I would like to thank to my cousin Buse Aslan for always being with me my whole life. Lastly, I would like to express my profound appreciation to my girlfriend Fatmanur Beşli for her encouragement, patience and support throughout this study and my life.

## TABLE OF CONTENTS

# LIST OF TABLES

TABLES

## LIST OF FIGURES

FIGURES

## LIST OF ABBREVIATIONS

ABBREVIATIONS

| | |
|---|---|
| Alg | Algorithm |
| Fig | Figure |
| Eq | Equation |
| Prob | Problem |
| LTL | Linear Temporal Logic |
| STL | Signal Temporal Logic |
| ptSTL | Past Time Signal Temporal Logic |
| CPU | Central Processing Unit |
| DAG | Directed Acyclic Graph |
| CPS | Cyber-Physical System |
| SVM | Support Vector Machine |
| ROI | Region of Interest |

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation and Problem Definition

Design and verification of cyber-physical systems that have to perform complex tasks are challenging processes. The final system is usually, complex and it contains sub-models such as Simulink models [1]. After the development of the model, in the verification phase, usually, the system's outputs/traces that are generated by simulating the model are used for checking whether the system satisfies it's requirements. Using the requirements of the developed system and traces, it is a relatively easy task to mark the traces where an unexpected behavior is observed. On the other hand, one of the challenging problems in the field is to identify the errors in the system which cause unexpected behaviors. It is important to note that as the system gets more complex, it gets harder for a human to understand the events that caused an erroneous behavior.

Our purpose in this work is to develop a method that identifies the root causes of a failure in an automated and efficient way. We also would like our method to be system independent. We do not require any implementation detail of the system. Only a set of system traces is required. Our method will help the system engineer to locate the modeling errors by explaining the events that lead to an erroneous behavior in the system. Thus, it is also important to represent these root causes in a human-readable way.

The problem that we concentrate on in this thesis is that given a set of labeled traces of a cyber-physical system, where the label indicates whether there is an unexpected behavior or not, express the root causes of the erroneous behavior of the system in an

automated, human-readable and efficient way.

### 1.2 Proposed Methods and Models

In this work, we propose a novel method to find temporal properties that lead to unexpected behaviors in the system by using labeled system traces in an automated way. The properties generated by our approach can give an insight into the underlying cause and help the system engineer to identify the corresponding modeling errors. To achieve this goal, we synthesize a formula that contains temporal properties such that evaluation of the synthesized formula mimics the labels in the given traces. In our work temporal properties are expressed with Signal Temporal Logic (STL) for it's expressive power and human-readable structure. STL is a rich specification language which is developed for describing properties of real valued signals [2] and it is an extension to Linear Temporal Logic [3]. Because of it's expressiveness and efficient calculation methods, STL is used in many areas such as: formal control [4], analysis of time series data [5] and run-time verification [6].

STL formula synthesis problem has been studied in the literature in different forms such as: differentiation of the good and the bad behavior of a system by finding an STL formula [7, 8, 9], finding an STL formula which is satisfied by all traces of the system [10, 11] or finding a formula which identifies the bad behavior at the time they occur [12] which is very similar to our case. Since our aim is to find the root causes of the erroneous behavior, we use past time signal temporal (ptSTL) logic where only past time temporal operators are used.

A ptSTL formula is constructed by connecting inequalities over the system variables with temporal and Boolean operators. For example, ptSTL formula $\mathbf{P}_{[0,4]}x > 5$ requires $x$ to exceed $5$ within the last $4$ seconds at least once. By combining boolean and temporal operators, more complex STL formulas can be generated. To give an example, consider the following specification "within the last 5 seconds, $x$ goes below $10$, and since then, within every 4 seconds, $y$ goes above $2$". The specification is expressed as ptSTL formula $\mathbf{P}_{[0,4]}\ y > 2\ \mathbf{S}_{[0,5]}\ x < 10$.

For signals that has a label at each time point where the label specifies whether the

system meets it's requirements at the corresponding time point, we propose an iterative method to synthesize a ptSTL formula in an efficient manner. The proposed method generates a set of ptSTL formulas up to a predefined formula length and iteratively combines them to synthesize a ptSTL formula to describe the label in the given dataset in an unguided way. For generating the set of ptSTL formulas, we define a novel approach that extends the monotonicity properties of STL and using these monotonicity properties we employ an efficient search algorithm.

## 1.3 Contributions and Novelties

Monotonicity properties for parameters in an STL formula were first defined in [10]. In their study, they required an ordering between the parameters in the formula. As a consequence of this ordering, the parameters are optimized in the given order, which causes the parameters with the lower orders to be optimized according to the results of the previously optimized parameters. In our work, there is no such ordering thus a global optimum can be found. Also in [8], an iterative formula generation method is defined as in our case but [8] is valid for only a sub-class of STL formulas whereas our method includes all STL formulas. In addition, our iterative formula construction method allows us to generate complex formulas in an efficient manner which was not achieved in [12].

To conclude, our contributions are as follows:

- We extend the monotonicity definition of STL
- We present an efficient parameter synthesis algorithm
- We develop an iterative STL formula synthesis method

## 1.4 The Outline of the Thesis

In the introduction chapter, first we state our motivation for this study. Then we informally present the problem studied in this thesis. Furthermore, we introduce our formula synthesis method for solving the problem. Lastly we state our contributions

in this area by briefly comparing our work with previous works. In Chapter 2 we will continue by providing the preliminary information on signal temporal logic and it's extensions such as past time signal temporal logic and parametric signal temporal logic. Then, we will introduce the related studies. Since we do parameter optimization using monotonicity properties of STL in our work, we will also introduce the works that uses monotonicity of STL. In Chapter 3, we will formally define our formula synthesis problem. In Chapter 4, we will introduce proposed method in three parts by providing running examples in each part. In Chapter 5 we present results on two case studies and compare them with similar works. In Chapter 6, we will extend the formula synthesis problem for single labeled signals and approach to this problem as a classification problem, propose a solution to the problem and provide a case study. Lastly in Chapter 7 we will summarize our work and conclude the thesis.

# CHAPTER 2

# RELATED WORK AND BACKGROUND

## 2.1 Background Information

### 2.1.1 Linear Temporal Logic

Linear Temporal Logic formulas are generated by using atomic propositions, boolean operators and temporal operators. Boolean conjunction, negation and true are denoted with $\vee$, $\neg$ and $\mathbf{T}$, respectively. Other operators such as disjunction ($\wedge$) or implication ($\rightarrow$) can be obtained by combining the three basic operators. There are two temporal operators in LTL which are Next($\mathbf{X}$) and Until ($\mathbf{U}$). An LTL formula ($\phi$) over a set of atomic propositions ($O$) can be formally defined recursively using the following syntax where $\phi$,$\phi_1$ and $\phi_2$ are LTL formulas and $o \in O$ is an atomic proposition.

$$\phi = o \mid \mathbf{T} \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \mathbf{X}\phi \mid \phi_1 \mathbf{U} \phi_2 \tag{2.1}$$

Two temporal operators which are commonly used Globally($\mathbf{G}$) and Eventually($\mathbf{F}$) are defined by using 2.1 as follows:

$$\mathbf{F}\phi = \mathbf{T}\mathbf{U}\phi \tag{2.2}$$

$$\mathbf{G}\phi = \neg\mathbf{F}\neg\phi \tag{2.3}$$

For a word $w = w(1)w(2)w(3)\ldots \in (2^O)^w$, satisfaction of formula $\phi$ at position $k \in N^+$, denoted by $w(k) \models \phi$ over a set of propositions $O$ is defined as follows:

- $w(k) \models \mathbf{T}$
- $w(k) \models o$ *for some* $o$ *if* $o \in w(k)$
- $w(k) \models \neg\phi$ *if* $w(k) \not\models \phi$

- $w(k) \models \phi_1 \wedge \phi_2$ *if* $w(k) \models \phi_1$ *and* $w(k) \models \phi_2$
- $w(k) \models \mathbf{X}\phi$ *if* $w(k+1) \models \phi$
- $w(k) \models \phi_1 \mathbf{U} \phi_2$ *if there exists* $j \geq k$ *such that* $w(j) \models \phi_2$ *and for all* $k \leq i < j$*, we have* $w(i) \models \phi_1$

To make the formal definition understood better, following informal definitions can be observed:

- $\mathbf{X}\phi$ is satisfied at the current step if $\phi$ is satisfied in the next step
- $\phi_1 \mathbf{U} \phi_2$ is satisfied if $\phi_1$ is satisfied until a $\phi_2$ is satisfied
- $\mathbf{G}\phi$ is satisfied if $\phi$ is satisfied at all steps
- $\mathbf{F}\phi$ is satisfied if $\phi$ is satisfied at some time in the future

### 2.1.2 Signal Temporal Logic

Signal Temporal Logic is an extension to LTL. In STL real valued signals and real valued constraints are used. A Signal Temporal Logic (STL) formula is recursively defined as follows:

$$\phi = \mathbf{T} | x^i \sim c | \neg\phi | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \phi_1 \mathbf{U}_{[a,b]} \phi_2 | \mathbf{F}_{[a,b]} \phi | \mathbf{G}_{[a,b]} \phi \tag{2.4}$$

where $\mathbf{T}$ is boolean $true$, $x^i$ corresponds to the $i^{th}$ signal variable, $\sim \in \{>, <\}$, and $c$ is a real valued constant. $\mathbf{U}_{[a,b]}$ $(until)$, $\mathbf{F}_{[a,b]}$ $(eventually)$, and $\mathbf{G}_{[a,b]}$ $(globally)$ are bounded temporal operators with time bound $[a, b]$.

In our scope an n-dimensional discrete signal $\mathbf{x}$ is a mapping from time domain $\mathbb{N}^+$ to the real numbers $\mathbb{R}^n$. A signal with finite length $K+1$ can be seen as a sequence such as: $\mathbf{x} = x_0, x_1, ..., x_K$. We use notation $x_t^i$ to specify the $i$th dimension of the signal (i.e. $i^{th}$ signal variable) at time $t$.

Informally, for signal $\mathbf{x}$, at time $t$ semantics are as follows:

- $\mathbf{F}_{[a,b]}\phi$ is satisfied if $\phi$ is satisfied in time interval $[t+a, t+b]$ at least once

- $\mathbf{G}_{[a,b]}\phi$ is satisfied if $\phi$ is satisfied in time interval $[t+a, t+b]$ at all points
- $\phi_1 \mathbf{U}_{[a,b]} \phi_2$ is satisfied if $\phi_2$ is satisfied at some time $t' \in [t+a, t+b]$ at least once and $\phi_1$ is satisfied at each point between $[t, t+t^{'})$

$(\mathbf{x}, t) \models \phi$ notation means that formula $\phi$ is satisfied at time $t$ for signal $\mathbf{x}$. Formal semantics for satisfaction of an STL formula is defined as:

$$
\begin{aligned}
&(\mathbf{x}, t) \models \mathbf{T} && \\
&(\mathbf{x}, t) \models x^i \sim c && \text{iff } x_t^i \sim c, \sim \in \{>, <\} \\
&(\mathbf{x}, t) \models \phi_1 \wedge \phi_2 && \text{iff } (\mathbf{x}, t) \models \phi_1 \text{ and } (\mathbf{x}, t) \models \phi_2 \\
&(\mathbf{x}, t) \models \phi_1 \vee \phi_2 && \text{iff } (\mathbf{x}, t) \models \phi_1 \text{ or } (\mathbf{x}, t) \models \phi_2 \\
&(\mathbf{x}, t) \models \mathbf{F}_{[a,b]}\phi && \text{iff } \exists t' \in [t+a, t+b], (\mathbf{x}, t') \models \phi \\
&(\mathbf{x}, t) \models \mathbf{G}_{[a,b]}\phi && \text{iff } \forall t' \in [t+a, t+b], (\mathbf{x}, t') \models \phi \\
&(\mathbf{x}, t) \models \phi_1 \mathbf{U}_{[a,b]} \phi_2 && \text{iff } \exists t' \in [t+a, t+b], (\mathbf{x}, t') \models \phi_2, \\
& && \quad \forall t'' \in [t', t](\mathbf{x}, t'') \models \phi_1,
\end{aligned}
\tag{2.5}
$$

Note that the Eventually ($\mathbf{F}$) and Globally ($\mathbf{G}$) operators are the special cases of the until operator where:

$$\mathbf{F}_{[a,b]}\phi := \mathbf{T}\ \mathbf{U}_{[a,b]}\phi \tag{2.6}$$

$$\mathbf{G}_{[a,b]}\phi := \neg \mathbf{F}_{[a,b]} \neg \phi. \tag{2.7}$$

### 2.1.3 Past Time Signal Temporal Logic

Past Time Signal Temporal Logic (ptSTL) is an extension to STL. While STL focuses on future events, ptSTL deals with past events. A Past Time Signal Temporal Logic (ptSTL) formula is recursively defined as follows:

$$\phi = \mathbf{T} | x^i \sim c | \neg\phi | \phi_1 \wedge \phi_2 | \phi_1 \vee \phi_2 | \phi_1 \mathbf{S}_{[a,b]} \phi_2 | \mathbf{P}_{[a,b]}\phi | \mathbf{A}_{[a,b]}\phi \tag{2.8}$$

where $\mathbf{T}$, $x^i$, $\sim \in \{>, <\}$, $c$, $\wedge$,$\vee$ and $\neg$ are defined as in (2.4). $\mathbf{S}_{[a,b]}$ $(since)$, $\mathbf{P}_{[a,b]}$ $(previously)$ and $\mathbf{A}_{[a,b]}$ $(always\ in\ the\ past)$ are bounded past temporal operators with time bound $[a, b]$.

Informally, for signal $\mathbf{x}$, at time $t$ semantics are as follows:

- $\mathbf{P}_{[a,b]}\phi$ is satisfied if $\phi$ is satisfied in time interval $[t-b, t-a]$ at least once
- $\mathbf{A}_{[a,b]}\phi$ is satisfied if $\phi$ is satisfied in time interval $[t-b, t-a]$ at all points
- $\phi_1 \mathbf{S}_{[a,b]} \phi_2$ is satisfied if $\phi_2$ holds at some time $t' \in [t-b, t-a]$ and $\phi_1$ holds since then.

$(\mathbf{x}, t) \models \phi$ notation means that formula $\phi$ is satisfied at time $t$ for signal $\mathbf{x}$. Formal semantics for satisfaction of a ptSTL formula is defined as:

$$
\begin{aligned}
&(\mathbf{x}, t) \models \mathbf{T} && \\
&(\mathbf{x}, t) \models x^i \sim c && \text{iff } x_t^i \sim c, \sim \in \{>, <\} \\
&(\mathbf{x}, t) \models \phi_1 \wedge \phi_2 && \text{iff } (\mathbf{x}, t) \models \phi_1 \text{ and } (\mathbf{x}, t) \models \phi_2 \\
&(\mathbf{x}, t) \models \phi_1 \vee \phi_2 && \text{iff } (\mathbf{x}, t) \models \phi_1 \text{ or } (\mathbf{x}, t) \models \phi_2 \\
&(\mathbf{x}, t) \models \mathbf{P}_{[a,b]}\phi && \text{iff } \exists t' \in I(t, [a, b]), (\mathbf{x}, t') \models \phi \\
&(\mathbf{x}, t) \models \mathbf{A}_{[a,b]}\phi && \text{iff } \forall t' \in I(t, [a, b]), (\mathbf{x}, t') \models \phi \\
&(\mathbf{x}, t) \models \phi_1 \mathbf{S}_{[a,b]} \phi_2 && \text{iff } \exists t' \in I(t, [a, b]), (\mathbf{x}, t') \models \phi_2, \\
& && \quad \forall t'' \in [t', t] (\mathbf{x}, t'') \models \phi_1,
\end{aligned}
\tag{2.9}
$$

$$\text{where } I(t, [a, b]) = [t-b, t-a] \cap [0, t]$$

Note that the Previously ($\mathbf{P}$) and Always ($\mathbf{A}$) operators are the special cases of the since operator where:

$$\mathbf{P}_{[a,b]}\phi := \mathbf{T}\ \mathbf{S}_{[a,b]}\phi \tag{2.10}$$

$$\mathbf{A}_{[a,b]}\phi := \neg \mathbf{P}_{[a,b]} \neg \phi \tag{2.11}$$

*Parametric Signal Temporal Logic* is an extension of STL[13]. In parametric signal temporal logic, parameters are used instead of numeric constants in predicates and time intervals. An STL formula can be generated by assigning values to all parameters in an parametric STL formula. As an example, consider the parametric ptSTL formula $\phi(p_1, p_2, p_3) = \mathbf{F}_{[p_1,p_2]} x < p_3$. Assigning valuation $v = (p_1 \mapsto 3, p_2 \mapsto 5, p_3 \mapsto 10.2)$ to parametric STL formula $\phi(p_1, p_2, p_3)$ STL formula $\phi(v) = \mathbf{F}_{[3,5]} x < 10.2$ is obtained. Parametric ptSTL formulas are defined in a similar way.

### 2.1.4 Monotonicity of Parametric Signal Temporal Logic

Monotonicity properties for parametric signal temporal logic was first defined in [10]. Parametric STL formula $\phi(p_1, p_2, \ldots p_n)$ is *monotonically increasing* with parameter $p_i$ if (2.12) is satisfied along any signal $\mathbf{x}$. Similarly parametric STL formula $\phi(p_1, p_2, \ldots p_n)$ is *monotonically decreasing* with parameter $p_i$ if (2.13) is satisfied along any signal $\mathbf{x}$ where $v(p)$ denotes the valuation of parameter $p$ in formula $\phi$

$$\begin{aligned} &\text{for all } v, v' \text{ with } v(p_i) < v'(p_i) \textit{ and } v'(p_j) = v(p_j) \text{ for each } i \neq j, \\ &\qquad (\mathbf{x}, t) \models \phi(v) \implies (\mathbf{x}, t) \models \phi(v') \qquad (2.12) \\ &\text{for all } v, v' \text{ with } v(p_i) > v'(p_i) \textit{ and } v'(p_j) = v(p_j) \text{ for each } i \neq j, \\ &\qquad (\mathbf{x}, t) \models \phi(v) \implies (\mathbf{x}, t) \models \phi(v') \qquad (2.13) \end{aligned}$$

Informally, for a parametric STL formula $\phi$ with parameters $p_1, p_2, \ldots p_n$, $\phi$ is said to be monotonically increasing with $p_i$ if the valuation does not change from satisfying to not satisfying when the value of $p_i$ is increased and the rest of the parameters are kept the same. Similarly, if a parametric STL formula $\phi$ is monotonically decreasing with parameter $p_i$, valuation can not change from satisfying to not satisfying when the value of $p_i$ is decreased and the other parameters are kept the same. As an example consider the parametric STL formula $\phi = \mathbf{P}_{[0,5]}x > p_x$. It can be clearly seen that each time point that $\mathbf{P}_{[0,5]}x > 3$ is satisfied will also be satisfied by formula $\mathbf{P}_{[0,5]}x > 1$ on any signal. Since decreasing the value of the $p_x$ can not cause this formula's valuation to change from satisfying to not satisfying, $\phi$ is said to be monotonically decreasing with parameter $p_x$

## 2.2 Related Work

Temporal logic is widely used in formal verification [14] and system monitoring [15] thanks to its expressive power and effective algorithms. These algorithms allows for real time monitoring (or verification) of a system against a given temporal logic formula. In recent years, synthesis of such temporal logic formulas from a dataset of system traces is studied quite extensively. Following the pioneering study in the area [10], many studies has been done in the literature in different forms such as finding a

formula that would be satisfied by all system traces [10, 11, 7] or finding a formula that would categorize the time series data which is labeled as positive or negative [16, 8, 9, 12, 17]. Studies in the literature also differ by the dataset labeling process, while [17, 12] assume a label is assigned to each time point for each signal, [16, 9] assume that a single label is assigned to each signal. Another aspect is the employed temporal logic, while [10, 13, 9] use Signal Temporal Logic (STL),[17, 18, 19, 20, 21] use past time STL (ptSTL).

Synthesis of a formula from a given dataset requires defining patterns observed in the system traces as a temporal logic formula. For metric logics, such as Signal Temporal Logic, formula synthesis involves identifying formula structure and finding time bounds and signal thresholds. The introduction of Parametric Signal Temporal Logic [13] formalized this distinction. In particular, in a parametric STL formula, parameters can be used instead of numerical constants for time bounds and metric thresholds. Parametric STL allows researchers to define common signal behaviors as template formulas, which simplifies requirement writing process [22]. Some of the signal patterns defined in [22] are Spike, Overshoot, Settling Time, Rise Time and Steady State Error.

For example authors defines the steady state error and overshoot behavior in parametric STL form as in (2.14) and (2.15) respectively.

$$\mathbf{F}_{[T,T]} \mid \mathbf{x} - \mathbf{x_{ref}} \mid > a \tag{2.14}$$

$$\mathbf{F}_{[0,T]}(step(\mathbf{x_{ref}}, \mathbf{r}) \wedge \mathbf{F}(\mathbf{x} - \mathbf{x_{ref}} > c)) \tag{2.15}$$
$$\textit{where } step(y, r) = y(t + \epsilon) - y(t) > r$$

While (2.14) states that if the difference between the followed signal $\mathbf{x}$ and the reference ($\mathbf{x_{ref}}$) signal is greater then a predefined error margin $a$ at time horizon $T$, a steady state error exists, (2.15) states, for a step input (with amplitude at least $r$) at time t, if $\mathbf{x}$ exceeds the given reference $\mathbf{x_{ref}}$ by a quantity greater than $c$ at any time in the future, an overshoot scenario is encountered.

With the ability to explain certain signal patterns as parametric STL formulas, the early works on formula synthesis focused on finding valuations for parameters, i.e.,

optimizing these parameters, for a given parametric formula.In [23, 24] authors state that in model-based development, not only verifying/falsifying formal system specifications but also exploring properties that the system satisfies is a desirable action. In [23], authors proposed a method to optimize a parametric STL formula which includes one unknown parameter. They defined robustness functions to optimize the parameter and showed that these robustness functions are monotonic with respect to search parameter. Using stochastic optimization methods [25, 26, 27] provided in S-TaLiRo [28] tool, they iteratively optimize the unknown parameter by analyzing the trace of the given system. In [24], the method given in [23] is extended for multiple parameters. It is stated that optimizing multiple parameters is a problem in the form of Pareto Front. Their method can optimize multiple parameters using S-TaLiRo [28] tool with the condition that the robustness function of the optimized parameter has the same monotonicity with respect to all parameters in the formula. Thus the method is valid for a sub-class of STL formulas. The STL parameter synthesis problem is also studied in [10], where the authors utilize the monotonicity properties of parameters over the quantitative valuation semantics. Their method requires an ordering over the unknown parameters. In particular, the parameters are synthesized in the given order using monotonicity properties.

The above mentioned approaches require an expert to write a parametric formula for the considered system. On the other hand, recent works consider the synthesis of both formula structure, i,e, a parametric formula, and its parameters. These studies are conducted with the assumption that the structure of the solution is not known. In [29, 8], approaches based on directed acyclic graphs (DAG) are developed. In [29] authors fist convert set of STL formulas to Directed Acyclic Graph (DAG). Then using simulated annealing [30] they estimate the parameters. Then they iteratively grow the nodes while pruning the nodes with high cost where high cost means this formula does not fit the observed data. In [8] authors define the problem as an anomaly detection problem and proposed an unsupervised learning algorithm. They use the one class support vector machine (SVM) optimization to lift the input data to a higher dimension and separate the normal data from the anomalous data [31] by adapting the objective function of one class SVM to an optimization function. Using this optimization function they follow the approach given in [29] for optimization of the

parameters. They also added a heuristic tightness function while estimating the parameters which basically penalizes the time parameters($\tau$) since they will use the method for monitoring purposes.

Another approach that is used to solve the formula synthesis problem in the literature is using decision tree based approaches. In [9], authors approach the problem as a binary classification problem. The dataset they used consists of a set of labeled signals where label states which class the signal belongs to (whether desired or undesired behavior). They define misclassification rate as the ratio of false positives and false negatives to total number of signals. They first build a binary decision tree that would classify the signals by minimizing the misclassification rate. Then they built a mapping between a fragment of STL and decision trees. Lastly, using this mapping they converted this decision tree to an STL formula.

In [16] a binary classification problem as in [9] is considered. First, they found regions of interest (ROI) in the form of spatial predicates that basically represents the desired and undesired behaviors. Then using these ROIs they obtain a representative set of signal formulas for the dataset. Using these representative set, they run a decision tree based logical clustering algorithm which generates an STL formula that minimizes the misclassification rate.

Another study [21] follows the decision tree based approaches. The dataset used in this study contains continuously labeled signals while [9, 16] uses a single label for each signal. The labels in the dataset mark the moments of the undesired behavior at the time they occur. In [21], the authors first transform the dataset into another dataset using windowing and applying min-max filters of different lengths. Then using Quinlan's C4.5 decision tree algorithm [32] in WEKA [33] they form a decision tree from the transformed dataset. Lastly they generate the ptSTL formula from the decision tree.

In [12], a grid search based method to find the monitoring rules for a system is proposed. In the system, dataset contains continuously labeled signals. First all possible parametric ptSTL formulas for given system variables up to a predefined number of operators are generated using the recursive definition of the ptSTL formulas. Then using these generated formulas, all possible values of parameters are evaluated using

a greedy search on the dataset and the result that fits the dataset best is returned as the optimal formula.

In this thesis, monotonicity properties of STL formulas are employed as in [10]. However, there is no ordering assumption between the parameters, as a consequence a global optimum can be found. Also different from [24], this method is valid for all STL formulas. Lastly, in [8], an iterative formula synthesis approach is followed as in this work but this iterative approach can be used for a sub-class of STL formulas (iPSTL). Different from [8], our method includes all STL formulas. Another benefit of the iterative construction is to be able to generate complex formulas efficiently that could not be achieved in previous studies [12].

# CHAPTER 3

# PROBLEM FORMULATION

As stated in Chapter 1, our aim in this work is to find the events that lead to erroneous behavior in a system. In addition, to identify these events, we aim at developing a system independent method. In particular, we do not require any information on the system such as implementation of the system, on what platform the system runs or the model of the system. In our work, we only need a labeled set of sample traces (runs) that can be acquired by running or simulating the system. A sample trace is a time series data over the system variables and the label. Essentially, the label is a binary variable which is the indicator of whether the system satisfies the specifications or not. If the specifications are satisfied, the label should be 0. If there is an erroneous behavior, the label should be 1. For a signal with continuous label, we formally define our dataset as follows:

$$\begin{aligned} \mathcal{D} = \{(\mathbf{x}, \mathbf{l}) \mid \mathbf{x} =& x_0, x_1, \ldots, x_K, \\ \mathbf{l} =& l_0, l_1, \ldots, l_K, \text{ and} \\ & x_t \in \mathbb{R}^n, l_t \in \{0, 1\}, t = 0, \ldots, K\}, \end{aligned} \tag{3.1}$$

where $l_t = 1$ means that, an erroneous behavior is occurred on signal $\mathbf{x}$ at time $t$.

Informally, our goal is to synthesize a ptSTL formula $\phi$ such that, the label ($\mathbf{l}^\phi = l_0^\phi, l_1^\phi, \ldots, l_N^\phi$) that will be acquired by evaluating $\phi$ at each time point on signal $\mathbf{x}$ matches the label $\mathbf{l}$ in the dataset (3.1).

By evaluating the synthesized ptSTL formula $\phi$ on signal $\mathbf{x} = x_0, \ldots, x_K$, $\mathbf{l}^\phi$ is gen-

erated as follows:

$$l_t^\phi = \begin{cases} 1 \text{ if } (\mathbf{x}, t) \models \phi \\ 0 \text{ otherwise} \end{cases} \tag{3.2}$$

After providing the information on our dataset $\mathcal{D}$ and the generation of $\mathbf{l}^\phi$, we can define our problem formally as follows:

**Problem 3.0.1** *Given a dataset $\mathcal{D}$ as in* (3.1)*, find a ptSTL formula $\phi$ such that for any $(\mathbf{x}, \mathbf{l}) \in \mathcal{D}$ and $t \in [0, K]$, $l_t = l_t^\phi$, where $l_t^\phi$ is generated as defined in* (3.2)*.*

To clarify Problem 3.0.1, an example dataset and its solution is given in Example 3.0.1.

**Example 3.0.1** *Consider a monitoring system where we monitor the CPU usage rate and measure the temperature of the CPU. If the processor temperature goes above a certain threshold (which is considered as an unwanted event), the data is labeled with 1. A sample run of the system is shown in Figure 3.1.*

*According to our problem definition (3.0.1), our goal is to find a ptSTL formula $\phi$ such that, by evaluating this formula on signal $\mathbf{x}$, we would like to generate a label $\mathbf{l}^\phi$ as in (3.2) that matches the label $\mathbf{l}$.*

*A perfect match of $\mathbf{l}$ and $\mathbf{l}^\phi$ is shown in Figure 3.2 where $\phi = \mathbf{A}_{[0,2]}(x > 30)$ which simply tells us, if the CPU usage rate exceeds 30 % for the last three time steps, an unwanted event (in this case, CPU temperature increases above the critical threshold) occurs.*

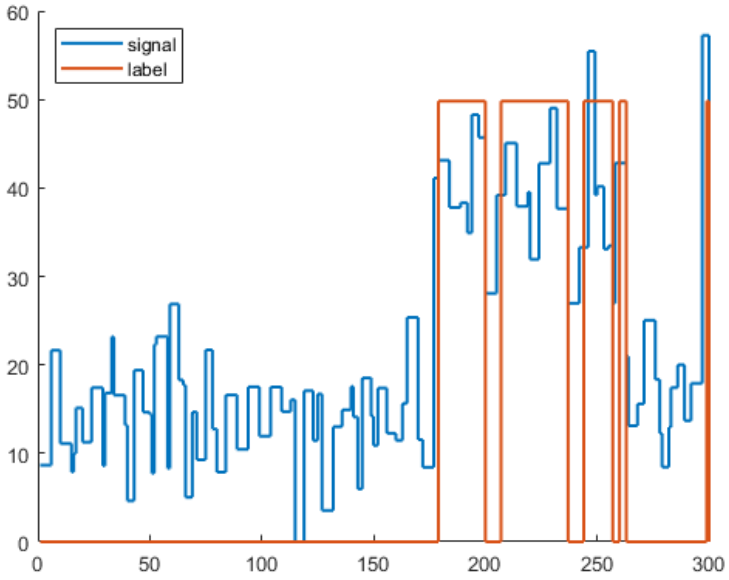



Figure 3.1: An example from CPU usage dataset which consists CPU usage rate signal and label. Label is scaled for clear view.

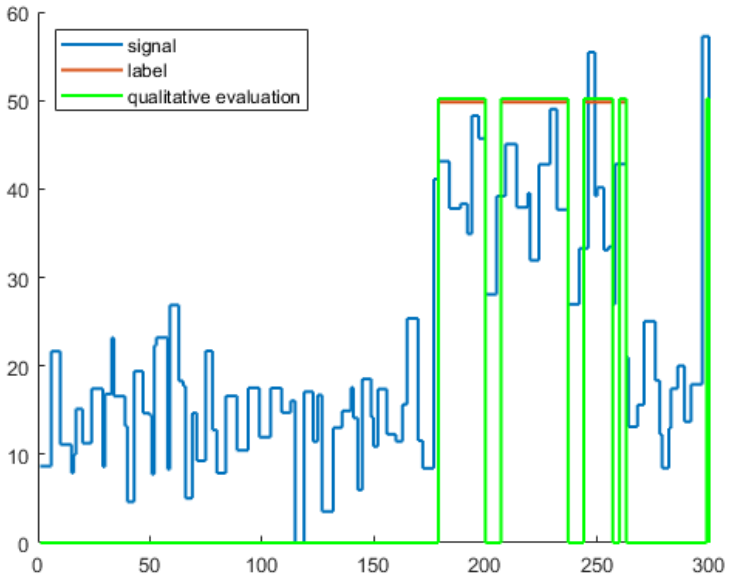



Figure 3.2: A sample signal $\mathbf{x}$, it's label $\mathbf{l}$, and the labels $\mathbf{l}^{\phi}$ obtained by evaluating $\phi$ along $\mathbf{x}$. Both label $\mathbf{l}$ and evaluation result $\mathbf{l}^{\phi}$ are scaled for better view. In this example, the evaluation matches the signal label at each time point.

# CHAPTER 4

# PROPOSED METHOD

In this chapter, the proposed method for solving Problem 3.0.1 will be explained in detail. The overall procedure for the proposed method is given in Figure 4.1. There are 2 main inputs in the method which are, a dataset $\mathcal{D}$ as defined in (3.1) and a set of parametric ptSTL formulas which is denoted with $\Phi$. $\Phi$ can be generated either recursively by using the semantics of ptSTL formulas or can be provided by a field expert. Using the dataset $\mathcal{D}$ and $\Phi$, parameter optimization method (Diagonal Search) optimizes the parameters for each parametric formula given in $\Phi$ and synthesizes optimized formulas. Then an iterative algorithm generates the final formula which is in the following form:

$$\phi = \phi_1 \vee \phi_2 \vee \dots \phi_p \tag{4.1}$$

Our method assumes that the erroneous behavior in the system can be caused by a number of different events. Our aim in the parameter optimization part is to find these events that led to the erroneous behavior efficiently and then in iterative construction method, final formula is constructed via disjunction operator. Parameter optimization method uses the monotonicity properties of the parameters in a parametric ptSTL formula and synthesizes a ptSTL formula in an efficient way.

In Chapter 4.1 extension of the monotonicity properties for STL will be discussed. Then using these monotonicity properties, a novel parameter optimization method will be explained in Chapter 4.2. Lastly in Chapter 4.3, the iterative formula construction method which combines STL formulas by disjuction will be explained.

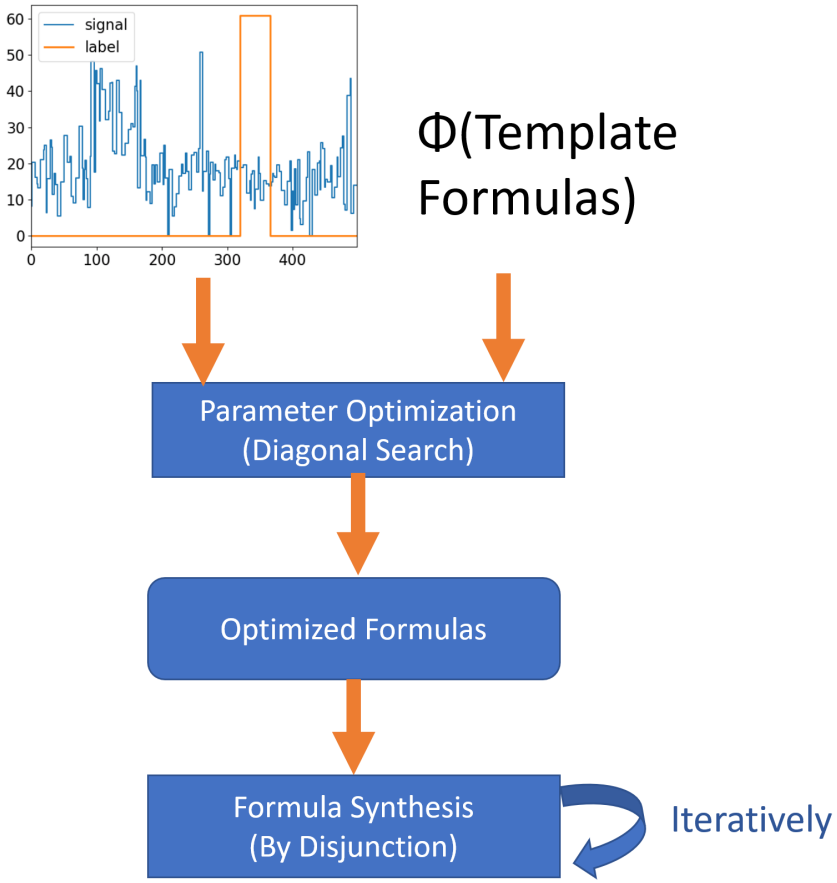



Figure 4.1: Proposed Method

## 4.1 Monotonicity of Parametric Signal Temporal Logic

In this study, we extend the work in [10]. In [10], monotonicity was defined between a parameter value in an STL formula and the satisfaction of this STL formula. We extend this monotonicity property by defining the monotonicity between a parameter value in a ptSTL formula and the number of positive/negative labels [34], which is obtained by evaluating the ptSTL formula on a dataset.

For a signal $\mathbf{x}$ with length $K+1$, the number of positive labels $P^{\#}(\phi, \mathbf{x})$ (4.2) and the number of negative labels $N^{\#}(\phi, \mathbf{x})$ (4.3) are defined as follows:

$$P^{\#}(\phi, \mathbf{x}) = \sum_{i=0}^{K} l_i^{\phi} \tag{4.2}$$

$$N^{\#}(\phi, \mathbf{x}) = \sum_{i=0}^{K} \neg l_i^{\phi} \tag{4.3}$$

where $\mathrm{l}^{\phi}$ is generated as defined in (3.2).

Also note that

$$P^{\#}(\phi, \mathbf{x}) + N^{\#}(\phi, \mathbf{x}) = K + 1 \tag{4.4}$$

For a parametric ptSTL formula $\phi$ with parameters $[p_1, p_1, \ldots p_m]$, we define the monotonicity between a parameter $p_i$ and $P^{\#}(\phi, \cdot)$ or $N^{\#}(\phi, \cdot)$.

For $P^{\#}(\phi, \cdot)$ to be monotonically increasing with $p_i$, the satisfaction value of $\phi$ should also be monotonically increasing with $p_i$. i.e., for any signal $\mathbf{x}$, if (2.12) holds, then (4.5) holds too.

$$\begin{aligned} &\text{for all } v, v' \text{ with } v(p_i) < v'(p_i), v'(p_j) = v(p_j) \text{ for each } i \neq j, \\ &\quad P^{\#}(\phi(v), \mathbf{x}) \leq P^{\#}(\phi(v'), \mathbf{x}) \end{aligned} \tag{4.5}$$

Similarly, for $P^{\#}(\phi, \cdot)$ to be monotonically decreasing with $p_i$, the satisfaction value of $\phi$ should also be monotonically decreasing with $p_i$. Specifically, for any signal $\mathbf{x}$, (4.6) holds when (2.13) holds:

$$\begin{aligned} &\text{for all } v, v' \text{ with } v(p_i) > v'(p_i), v'(p_j) = v(p_j) \text{ for each } i \neq j, \\ &\quad P^{\#}(\phi(v), \mathbf{x}) \leq P^{\#}(\phi(v'), \mathbf{x}) \end{aligned} \tag{4.6}$$

It is clear to see that, by using the information in (4.4), $P^{\#}(\phi, \cdot)$ and $N^{\#}(\phi, \cdot)$ should have the opposite monotonicity property, which means that if $P^{\#}(\phi, \cdot)$ is monotonically increasing with $p_i$ then $N^{\#}(\phi, \cdot)$ have to be monotonically decreasing with $p_i$.

The number of positive labels $P^{\#}(\phi, \mathcal{D})$ of a ptSTL formula $\phi$ for a dataset $\mathcal{D}$ is simply defined as the sum of positive labels for each signal,label pair in the dataset which is derived from (4.2) as follows:

$$P^{\#}(\phi, \mathcal{D}) = \sum_{(\mathbf{x}, \mathbf{l}) \in \mathcal{D}}^{K} P^{\#}(\phi, \mathbf{x}) \tag{4.7}$$

The number of negative labels denoted with $N^{\#}(\phi, \mathcal{D})$ is defined similarly as follows:

$$N^{\#}(\phi, \mathcal{D}) = \sum_{(\mathbf{x},\mathbf{l})\in\mathcal{D}}^{K} N^{\#}(\phi, \mathbf{x}) \tag{4.8}$$

As either a positive (1) or a negative (0) label is assigned to each data point, the equality $|\mathcal{D}| \times (K+1) = P^{\#}(\phi, \mathcal{D}) + N^{\#}(\phi, \mathcal{D})$ trivially holds.The number of correctly identified positive instances (*True Positives*), incorrect positive instances (*False Positives*), correctly identified negative instances (*True Negatives*) and incorrect negative instances (*False Negatives*) with respect to the labels generated by the formula $\phi$ using (3.2) and the dataset labels are defined respectively as follows:

$$TP^{\#}(\phi, \mathcal{D}) = \sum_{(\mathbf{x},\mathbf{l})\in\mathcal{D}} \sum_{i=0}^{K} l_i \wedge l_i^{\phi} \tag{4.9}$$

$$FP^{\#}(\phi, \mathcal{D}) = \sum_{(\mathbf{x},\mathbf{l})\in\mathcal{D}} \sum_{i=1}^{K} \neg l_i \wedge l_i^{\phi} \tag{4.10}$$

$$TN^{\#}(\phi, \mathcal{D}) = \sum_{(\mathbf{x},\mathbf{l})\in\mathcal{D}} \sum_{i=0}^{K} \neg l_i \wedge \neg l_i^{\phi} \tag{4.11}$$

$$FN^{\#}(\phi, \mathcal{D}) = \sum_{(\mathbf{x},\mathbf{l})\in\mathcal{D}} \sum_{i=1}^{K} l_i \wedge \neg l_i^{\phi} \tag{4.12}$$

The derivations of $TP^{\#}(\cdot,\cdot)$, $FP^{\#}(\cdot,\cdot)$, $TN^{\#}(\cdot,\cdot)$ and $FN^{\#}(\cdot,\cdot)$ preserves monotonicity properties (4.5) and (4.6). As a result, if $P^{\#}(\cdot,\cdot)$ is monotonically increasing (or decreasing) with a parameter $p_i$, then both $TP^{\#}(\cdot,\cdot)$ and $FP^{\#}(\cdot,\cdot)$ are increasing (or decreasing) with $p_i$. Same rule holds for negatives too, i.e. if $N^{\#}(\cdot,\cdot)$ is monotonically increasing (or decreasing) with a parameter $p_i$, then both $TN^{\#}(\cdot,\cdot)$ and $FN^{\#}(\cdot,\cdot)$ are increasing (or decreasing) with $p_i$

Notation $\mathcal{M}(p, \phi, P)$ is used to denote the monotonicity (increasing or decreasing) of parameter $p$ in parametric ptSTL formula $\phi$ for the number of positives ( $P^{\#}(\cdot,\cdot)$,

$TP^{\#}(\cdot,\cdot)$ or $FP^{\#}(\cdot,\cdot)$:

$$\mathcal{M}(p,\phi,P) = \begin{cases} \mathbf{I} \text{ if } p \text{ is monotonically increasing in } \phi \\ \mathbf{D} \text{ if } p \text{ is monotonically decreasing in } \phi \end{cases} \tag{4.13}$$

Similarly notation $\mathcal{M}(p,\phi,N)$ is used to denote the monotonicity (increasing or decreasing) of parameter $p$ in parametric ptSTL formula $\phi$ for the number of negatives ( $N^{\#}(\cdot,\cdot)$, $TN^{\#}(\cdot,\cdot)$ or $FN^{\#}(\cdot,\cdot)$

Monotonicity properties for each parameter that can appear in a basic parametric ptSTL formula for positive and negative labels are shown in Table 4.1 and in Table 4.2 respectively.

Table 4.1: Monotonicity relations for Positive Labels

| $\phi$ | $\mathcal{M}(p,\phi,P)$ | $\phi$ | $\mathcal{M}(p,\phi,P)$ |
|---|---|---|---|
| $x > p$ | $\mathbf{D}$ | $x < p$ | $\mathbf{I}$ |
| $\mathbf{A}_{[c,p]}\varphi$ | $\mathbf{D}$ | $\mathbf{A}_{[p,c]}\varphi$ | $\mathbf{I}$ |
| $\mathbf{P}_{[c,p]}\varphi$ | $\mathbf{I}$ | $\mathbf{P}_{[p,c]}\varphi$ | $\mathbf{D}$ |
| $\varphi_1 \mathbf{S}_{[c,p]} \varphi_2$ | $\mathbf{I}$ | $\varphi_1 \mathbf{S}_{[p,c]} \varphi_2$ | $\mathbf{D}$ |

Table 4.2: Monotonicity relations for Negative Labels

| $\phi$ | $\mathcal{M}(p,\phi,N)$ | $\phi$ | $\mathcal{M}(p,\phi,N)$ |
|---|---|---|---|
| $x > p$ | $\mathbf{I}$ | $x < p$ | $\mathbf{D}$ |
| $\mathbf{A}_{[c,p]}\varphi$ | $\mathbf{I}$ | $\mathbf{A}_{[p,c]}\varphi$ | $\mathbf{D}$ |
| $\mathbf{P}_{[c,p]}\varphi$ | $\mathbf{D}$ | $\mathbf{P}_{[p,c]}\varphi$ | $\mathbf{I}$ |
| $\varphi_1 \mathbf{S}_{[c,p]} \varphi_2$ | $\mathbf{D}$ | $\varphi_1 \mathbf{S}_{[p,c]} \varphi_2$ | $\mathbf{I}$ |

Note that these derivations are based on simple parametric ptSTL formulas. For complex formulas, a parameter's monotonicity is decided by checking the syntax tree of the formula. For finding the monotonicity property of a parameter in a formula, one should follow the path from the root node to corresponding parameter node.Each

negation operator ($\neg$) that occurs in the syntax tree inverts the monotonicity property. For example, while $\mathcal{M}(p, \mathbf{A}_{[a,b]}x < p, P)$ is $\mathbf{I}$, $\mathcal{M}(p, \neg(\mathbf{A}_{[a,b]}x < p), P)$ is $\mathbf{D}$.

An example which shows monotonicity properties for a parameter in a simple parametric ptSTL formula is given in Example 4.1.1.

**Example 4.1.1** *Consider parametric ptSTL formula*

$$\phi = \mathbf{A}_{[p_1,p_2]}(x > p_3) \tag{4.14}$$

*Monotonicity properties of $p_1, p_2$ and $p_3$ according to Table 4.1 and Table 4.2 are:*
$\mathcal{M}(p_1, \phi, P) = \mathbf{I}$, $\mathcal{M}(p_2, \phi, P) = \mathbf{D}$, $\mathcal{M}(p_3, \phi, P) = \mathbf{D}$,
$\mathcal{M}(p_1, \phi, N) = \mathbf{D}$, $\mathcal{M}(p_2, \phi, N) = \mathbf{I}$, $\mathcal{M}(p_3, \phi, N) = \mathbf{I}$.

*Using the dataset in Example 3.0.1, monotonicity properties of parameter $p_3$ will be examined. For observing the monotonicity of $p_3$, $p_1$ and $p_2$ are set to 0 and 3 respectively in formula 4.14. Then $P^{\#}(\phi, \mathcal{D})$, $TP^{\#}(\phi, \mathcal{D})$, $FP^{\#}(\phi, \mathcal{D})$, $N^{\#}(\phi, \mathcal{D})$, $TN^{\#}(\phi, \mathcal{D})$ and $FN^{\#}(\phi, \mathcal{D})$ are computed for each $p_3 \in [0, 1, \ldots 40]$. The number of positives and negatives for different values of $p_3$ are given in Figures 4.2 and 4.3 respectively.*

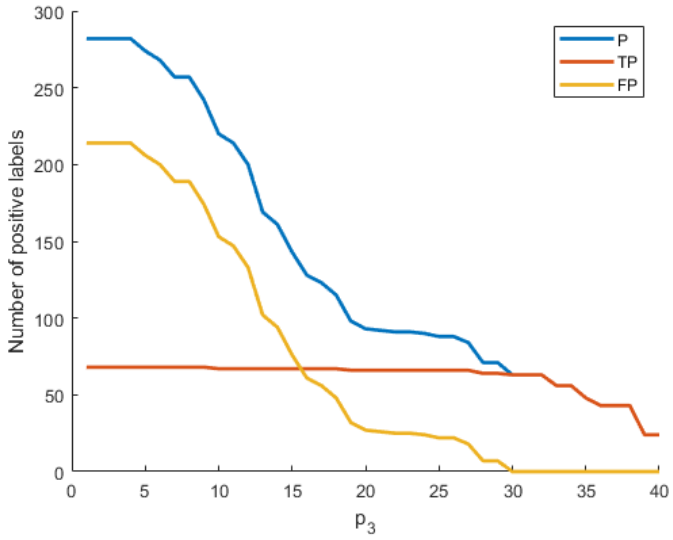



Figure 4.2: Number of positives with changing values of $p_3$

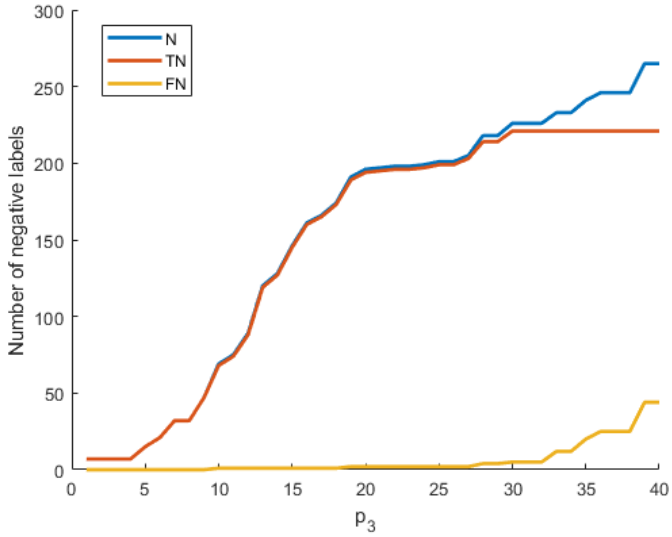



Figure 4.3: Number of negatives with changing values of $p_3$

*These results show that, as expected, the number of negative labels increases with the increasing value of* $p_3$ *since* $\mathcal{M}(p_3, \phi, N) = \mathbf{I}$ *and the number of positive labels decreases with the increasing value of* $p_3$ *since* $\mathcal{M}(p_3, \phi, P) = \mathbf{D}$.

## 4.2 Parameter Optimization Method

In this section, parameter optimization method called diagonal search will be presented. Aim of the diagonal search is to synthesize a ptSTL formula $\phi$ from a parametric ptSTL formula such that evaluation of the formula $\phi$ on a dataset $\mathcal{D}$ gives a high number of $TP^{\#}(\phi, \mathcal{D})$. It should be noted that these synthesized formulas will be combined with each other via disjunction operator in the following phase. The disjunction operator carries error to the resulting formula (error is $FP^{\#}(\phi, \mathcal{D})$ in this case). Therefore while synthesizing a ptSTL formula $\phi$:

- $TP^{\#}(\phi, \mathcal{D})$ should be maximized
- $FP^{\#}(\phi, \mathcal{D})$ should be bounded

Formal definition of this optimization problem is given in Prob. 4.2.1.

**Problem 4.2.1** *Given a labeled dataset $\mathcal{D}$ (3.1), a parametric ptSTL formula $\phi$ with $n$ parameters $p_1, p_2, \ldots, p_n$, lower and upper bounds $l_i, u_i$ for each parameter $p_i$, an error bound $B \in \mathbb{N}$, find the valuation $v$ within the given limits that maximizes $TP^{\#}(\phi(v), \mathcal{D})$ while guaranteeing that $FP^{\#}(\phi(v), \mathcal{D}) \leq B$.*

To solve Problem 4.2.1, first diagonal search algorithm which takes a parametric pt-STL formula with exactly two parameters will be explained. Then, this algorithm will be extended for parametric ptSTL formulas with more than two parameters.

Diagonal search algorithm adapts search problem of the product of an m element chain and an n element chain[35] for ptSTL parameter optimization. The diagonal search algorithm requires

- $\phi$: Parametric ptSTL formula with parameters $p_1$ and $p_2$
- $B$: Bound on $FP^{\#}(\phi(v), \mathcal{D})$
- $\mathcal{D}$: Dataset as defined in (3.1)
- $l_i, u_i, \delta_i$: lower bound, upper bound and step size for parameter $p_i$, $i \in \{1, 2\}$

In this method, an error constraint is defined:

$$FP^{\#}(\phi(v), \mathcal{D}) < B \tag{4.15}$$

Diagonal search algorithm starts with a initial valuation $v$ with $v(p_1)$ is the bound on $p_1$ that maximizes $TP^{\#}(\phi, \mathcal{D})$ (i.e. either $l_1$ or $u_1$ according to the monotonicity of the parameter $p_1$ in formula $\phi$) and $v(p_2)$ is the bound on $p_2$ that minimizes $TP^{\#}(\phi, \mathcal{D})$. Then, the algorithm iteratively changes the value of a parameter with following rules:

- $v(p_2)$ is changed by $\delta_2$ in the direction which increases $P^{\#}(\phi, \mathcal{D})$ if error constraint holds at $v$. This step simply increases both $TP^{\#}(\phi, \mathcal{D})$ and $FP^{\#}(\phi, \mathcal{D})$. Since error constraint is satisfied in the current valuation, algorithm searches for a higher number of $TP^{\#}(\phi, \mathcal{D})$.
- Otherwise $v(p_1)$ is changed by $\delta_1$ in the direction which decreases $P^{\#}(\phi, \mathcal{D})$. If error constraint is not satisfied, it means that number of maximum $FP^{\#}(\phi, \mathcal{D})$ bound is reached. Both $TP^{\#}(\phi, \mathcal{D})$ and $FP^{\#}(\phi, \mathcal{D})$ is decreased in this step.

This algorithm simply changes the values of parameters $p_1$ and $p_2$ according to monotonicity properties of the parameters and moves along the diagonal of the product of the discretized parameter domains while trying to find an evaluation where the $TP^{\#}(\phi, \mathcal{D})$ is maximized while $FP^{\#}(\phi, \mathcal{D})$ is less then the bound $B$. A summary of this method is given in Alg. 1

**Algorithm 1** $DiagonalSearch(\phi, B, \mathcal{D}, l_1, u_1, \delta_1, l_2, u_2, \delta_2)$

**Ensure:** $v_{best} = \arg\max_v \{TP^{\#}(\phi(v), \mathcal{D}) \mid FP^{\#}(\phi(v), \mathcal{D}) < B\}$

1: **if** $\mathcal{M}(p_1, \phi) ==$ **I then**
2: $\quad v(p_1) = u_1, \bar{\delta}_1 = -\delta_1$
3: **else**
4: $\quad v(p_1) = l_1, \bar{\delta}_1 = \delta_1$
5: **end if**
6: **if** $\mathcal{M}(p_2, \phi) ==$ **I then**
7: $\quad v(p_2) = l_1, \bar{\delta}_2 = \delta_2$
8: **else**
9: $\quad v(p_2) = u_1, \bar{\delta}_2 = -\delta_2$
10: **end if**
11: $v_{best} = [], TP_{best} = 0$
12: **while** $l_1 \leq v(p_1) \leq u_1 \wedge l_2 \leq v(p_2) \leq u_2$ **do**
13: $\quad$ **if** $B < FP^{\#}(\phi(v), \mathcal{D})$ **then**
14: $\quad\quad v(p_1) = v(p_1) + \bar{\delta}_1$
15: $\quad$ **else**
16: $\quad\quad$ **if** $TP^{\#}(\phi(v), \mathcal{D}) \geq TP_{best}$ **then**
17: $\quad\quad\quad TP_{best} = TP^{\#}(\phi(v), \mathcal{D}), v_{best} = v$
18: $\quad\quad$ **end if**
19: $\quad\quad v(p_2) = v(p_2) + \bar{\delta}_2$
20: $\quad$ **end if**
21: **end while**
22: **return** $v_{best}$

In lines 1-10 of Alg. 1, the initial values for the parameters $p_1$ and $p_2$ are selected together with their update direction according to the monotonicity properties of the

parameters. Then main loop of the algorithm starts (lines 12-21). At each iteration one parameter value is updated. If the error constraint (line 13) is not satisfied, parameter $p_1$ which is initialized with the purpose of maximizing the $TP^{\#}(\phi, \mathcal{D})$ is changed by $\bar{\delta}$ to reduce $FP^{\#}(\phi(v), \mathcal{D})$. If the error constraint is satisfied, the current parameter assignment becomes a candidate solution. This candidate solution is checked against the currently best known solution (line 16). If candidate solution is better than the current best known solution, best solution is updated as the candidate solution. Then, parameter $p_2$ is changed by $\bar{\delta}_2$ to increase $TP^{\#}(\phi(v), \mathcal{D})$. This iteration continues while both of the parameters are in the given bounds. When iteration ends, $v_{best}$ holds the evaluation which has the highest $TP^{\#}(\phi, \mathcal{D})$ with the constraint $FP^{\#}(\phi(v), \mathcal{D}) < B$. As a result, $O(m_1 + m_2)$ formula evaluations are done, where $m_1 = \frac{u_1 - l_1}{\delta_1}, m_2 = \frac{u_2 - l_2}{\delta_2}$.

In our work, a parameter appears only once in a parametric ptSTL formula. Therefore, the considered formulas are monotonic in each parameter, i.e., either monotonically increasing or monotonically decreasing.

An example that illustrates the Alg. 1 is provided in Example 4.2.1

**Example 4.2.1** *Consider the dataset $\mathcal{D}$ from Ex. 3.0.1 and a template ptSTL formula $\phi = \mathbf{A}_{[0,p2]} x > p1$. According to Table 4.1, monotonicities of $p_1$ and $p_2$ are as follows:*

$$\mathcal{M}(p_1, \phi, P) = \mathbf{D}$$
$$\mathcal{M}(p_2, \phi, P) = \mathbf{D}$$

*According to Prob. 4.2.1, the aim is to find an evaluation $v$ such that, $TP^{\#}(\phi(v), \mathcal{D})$ is maximized while keeping $FP^{\#}(\phi(v), \mathcal{D})$ below a bound. Since both $p_1$ and $p_2$ are monotonically decreasing, $TP^{\#}(\phi, \mathcal{D})$ and $FP^{\#}(\phi(v), \mathcal{D})$ are expected to decrease with increasing values of $p_1$ and $p_2$. $TP^{\#}(\phi(v), \mathcal{D})$ and $FP^{\#}(\phi(v), \mathcal{D})$ with respect to each possible combination of $p_1$ and $p_2$ are given in Fig.4.4 and Fig.4.5 respectively. Note that possible solutions where $FP^{\#}(\phi(v), \mathcal{D}) = 0$ that satisfies the error constraint are marked with green in Fig. 4.5 where*

- $B = 1$

- $l_1 = 0, \delta_1 = 10, u_1 = 40$
- $l_2 = 0, \delta_2 = 1, u_2 = 4$

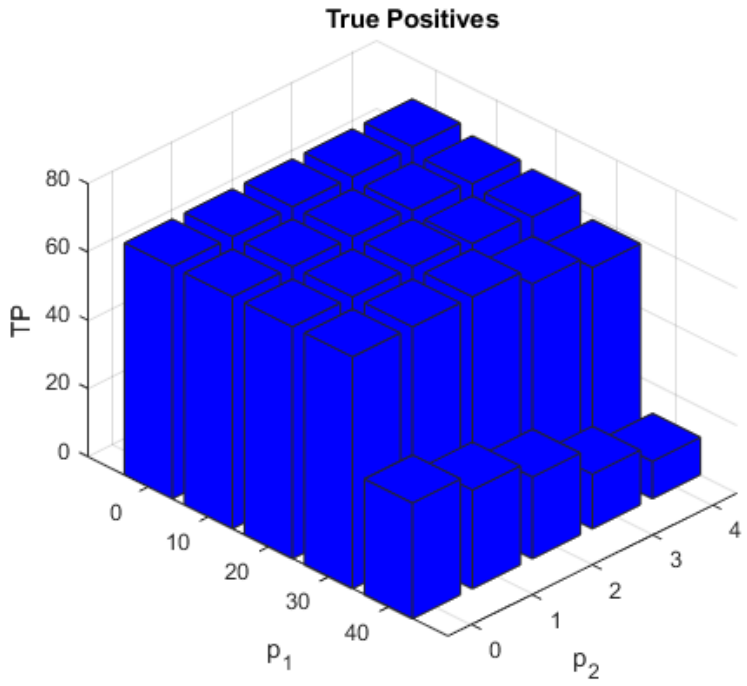



Figure 4.4: Number of true positives for each possible combination of $p_1$ and $p_2$

*According to Alg. 1, starting point is selected as* $v(p_1) = 0$ *and* $v(p_2) = 4$*. Then diagonal search starts by checking whether the current evaluation satisfies the error constraint or not and continues to iteratively update* $v(p_1)$ *or* $v(p_2)$*. This way instead of trying all possible evaluations, only the evaluations that lies in the diagonal of the parameter space are evaluated. In Fig. 4.6, starting point, the evaluations that was done while Alg. 1 is run and candidate solutions for the problem are shown.*

*When Alg. 1 is run, resulting formula that has the highest* $TP^{\#}(\phi(v), \mathcal{D})$ *is shown in* (4.16).

$$\phi = (\mathbf{A}_{[0,2]}(x_0 > 30)) \tag{4.16}$$

Lastly in this chapter, a method to solve Prob. 4.2.1 will be explained. For a parametric ptSTL formula $\phi$ with n parameters $p_1, \ldots, p_n$,

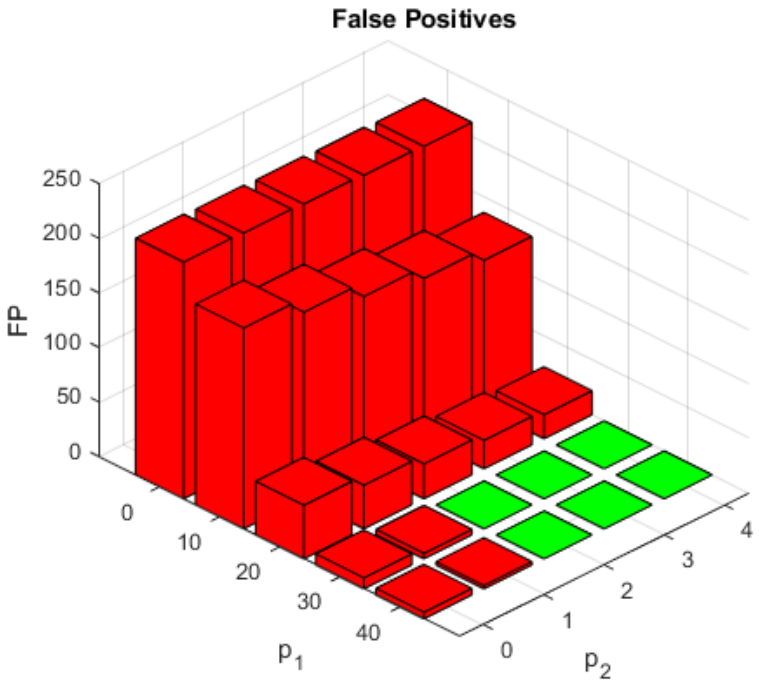



Figure 4.5: Number of false positives for each possible combination of $p_1$ and $p_2$. Instances $FP^{\#}(\phi(v), \mathcal{D}) \geq B$ are marked with red and rest is marked with green.

- If $n = 1$, optimal value is found with binary search
- If $n = 2$, optimal value is found by directly using $DiagonalSearch$ method described in Alg. 1
- If $n > 2$, $DiagonalSearch$ is run for $p_1$ and $p_2$ for all possible combinations of the remaining $n - 2$ parameters, and the optimal parameters are returned.

This whole algorithm is referred as $ParameterSynthesis(\phi, B, \mathcal{D})$.

In literature, the earlier works that finds the optimal parameter valuation without additional constraints (e.g. parameter ordering as in [10]) enumerate the parameter space and perform a grid search [12]. That results in a complexity of $f(\mathcal{D}) = k^n$ for $|p_i| = k$ and a formula with $n$ parameters on the dataset $\mathcal{D}$.
With $ParameterSynthesis(\phi, B, \mathcal{D})$ method, complexity analysis is as in (4.17).

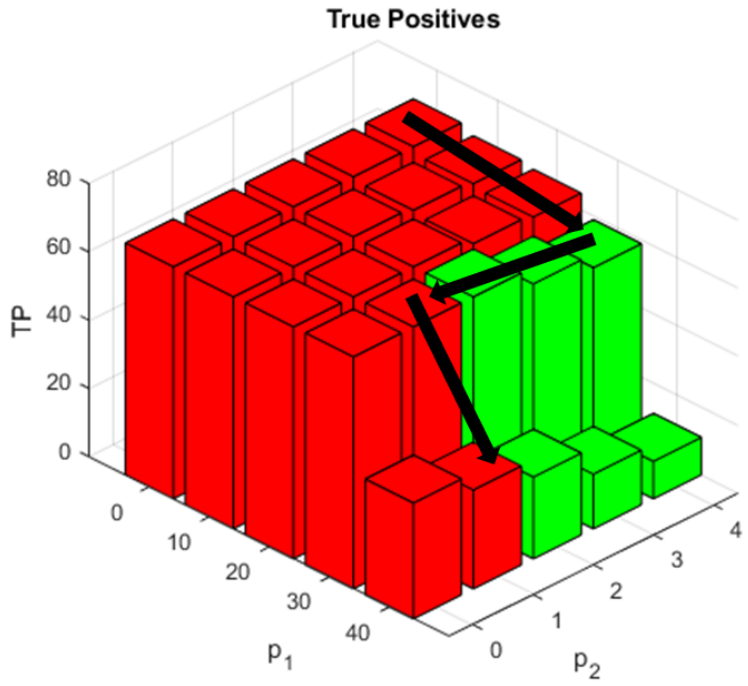



Figure 4.6: Number of true positives for each possible combination of $p_1$ and $p_2$. Instances $FP^{\#}(\phi(v), \mathcal{D}) \geq B$ are marked with red and rest (candidate solutions) is marked with green. Also evaluated combinations of $p_1$ and $p_2$ by the diagonal search are marked with arrows.

$$f(\mathcal{D}) = \begin{cases} k^{n-1} \text{ if } n \geq 2 \\ \log k \text{ if } n = 1 \end{cases} \tag{4.17}$$

## 4.3 Iterative Formula Construction Method

Final step of the solution to the main problem (Prob. 3.0.1) i.e., a method to find a ptSTL formula that explains all of the labeled events in the dataset, is explained in this section. Label 1 is considered to be marking an unexpected event. Generally, an unexpected event can occur due to a number of different reasons in a system. The method aims to find each distinct cause and then by iteratively combining these reasons, explains the events that causes the erroneous behavior as a ptSTL formula. In this method, each distinct cause is represented as a ptSTL formula and these causes

are combined using disjunction ($\vee$) operator. The main purpose at each operation is to add a ptSTL formula to the final formula where added ptSTL formula classifies a subset of the unwanted events while limiting the incorrectly labeled instances (False Positives in this case) since incorrectly labeled instances propogates with the disjunction operator. First, the set of all parametric formulas are defined as in [12], and parameter optimization is performed on each of them using $ParameterSynthesis$ method. Then these formulas are iteratively combined until there is no increase on True Positive numbers or a predefined formula length is reached or all labeled events are explained in the resulting formula (no False Negative).

Given the set of system variables, $\{x^1, \ldots, x^n\}$, and a bound on the number of operators $N$, the set of all parametric ptSTL formulas with up to $N$ operators $\mathcal{F}^{\leq N}$ is recursively defined as:

$$
\begin{aligned}
\mathcal{F}^0 &= \{x^i \sim p_i \mid \sim \in \{<, >\}, i = 1, \ldots, n\} \cup \{\mathbf{T}\} \\
\mathcal{F}^N &= \{\neg\phi, \mathbf{P}_{[a,b]}\phi, \mathbf{A}_{[a,b]}\phi \mid \phi \in \mathcal{F}^{N-1}\} \cup \\
&\quad \bigcup_{i=1}^{n-1} \{\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2, \phi_1 \mathbf{S}_{[a,b]} \phi_2 \mid \\
&\qquad\qquad \phi_1 \in \mathcal{F}^i, \phi_2 \in \mathcal{F}^{N-i-1}\} \\
\mathcal{F}^{\leq N} &= \cup_{i=0}^{N} \mathcal{F}^i
\end{aligned}
\tag{4.18}
$$

The proposed formula synthesis approach is summarized in Alg. 2. This synthesis method takes:

- Labeled dataset $\mathcal{D}$ as defined in (3.1)
- Bound on the number of ptSTL formulas $p$
- Bound on the number of false positives $B$
- Set of parametric formulas $\mathcal{F}$

Method generates a ptSTL formula $\phi^\star$ in the form of (4.1) as a result. Note that generated ptSTL formula contains at most $p$ sub-formulas and $TP^{\#}(\phi^\star, \mathcal{D})$ is maximized while $FP^{\#}(\phi^\star, \mathcal{D}) < B * p$ error constraint is satisfied. Bound $B$ is the maximum

**Algorithm 2** $FormulaSynthesis(\mathcal{F}, B, \mathcal{D}, p)$

**Require:** $\mathcal{F}$: a set of parametric ptSTL formulas, $B$: bound on the number of false positives, $\mathcal{D}$: a dataset as in (3.1), $p$: upper bound on the number of formulas concatenated with disjunction.

1: $\mathcal{F}^v = \{\phi(v) = ParameterSynthesis(\phi, B, \mathcal{D}) \mid \phi \in \mathcal{F}\}$
2: $i = 0, TP_{prev} = 0, TP = 1, \Phi = false$
3: **while** $TP > TP_{prev}$ and $i < p$ **do**
4: $\quad \phi(v)^* = \arg\max_{\phi(v)\in\mathcal{F}^v} TP^{\#}(\Phi \vee \phi(v), \mathcal{D})$
5: $\quad \Phi = \Phi \vee \phi(v)^*$
6: $\quad i = i + 1$
7: $\quad TP_{prev} = TP, TP = TP^{\#}(\Phi, \mathcal{D})$
8: **end while**
9: **return** $\Phi$

number of False Positives that is allowed for each sub-formula. Since at most $p$ formulas are allowed to be concatenated, resulting formula can have at most $B * p$ False Positives. The set of parametric ptSTL formulas that will be used in the algorithm can be autonomously provided by recursively defining them as in (4.18), or, they can be provided by an expert of the considered system. Firstly, in the algorithm, each parametric ptSTL formula $\phi \in \mathcal{F}$ are optimized on given dataset $\mathcal{D}$ with error constraint $B$ with the purpose of maximizing $TP^{\#}(\phi, \mathcal{D})$ (line 1). Then starting with $\Phi = false$, the formula $\phi(v)^\star$ maximizing the valuation of the combined formula $\Phi \vee \phi(v)^\star$ is selected from the set of ptSTL formulas $\mathcal{F}^v$ iteratively until maximum number of sub-formulas allowed ($p$) is reached or the number of True Positives does not increase (i.e. adding a new formula to the result does not improve the result)(lines 3-8). Note that as proposed in (4.1), at each iteration a new formula is added to $\Phi$ using disjunction ($\vee$) operator.

$ParameterSynthesis(\phi, B, \mathcal{D})$ method is run only once for each parametric ptSTL formula $\phi \in \mathcal{F}$ in Alg. 2. Then only $TP^{\#}(\Phi \vee \phi(v), \mathcal{D})$ is computed for each $\phi(v) \in \mathcal{F}^v$ to pick the formula $\phi^\star$ which gives the highest number of True Positives at each iteration. Thus it should be noted that the selected formula $\phi^\star$ using the iterative synthesis approach might not be optimal formula. Basically, highest fitness of the formula can be obtained by optimizing each parameter of the formula in the form of

(4.1). But this computation would not be feasible for large formulas because of the complexity of the parameter synthesis algorithm.

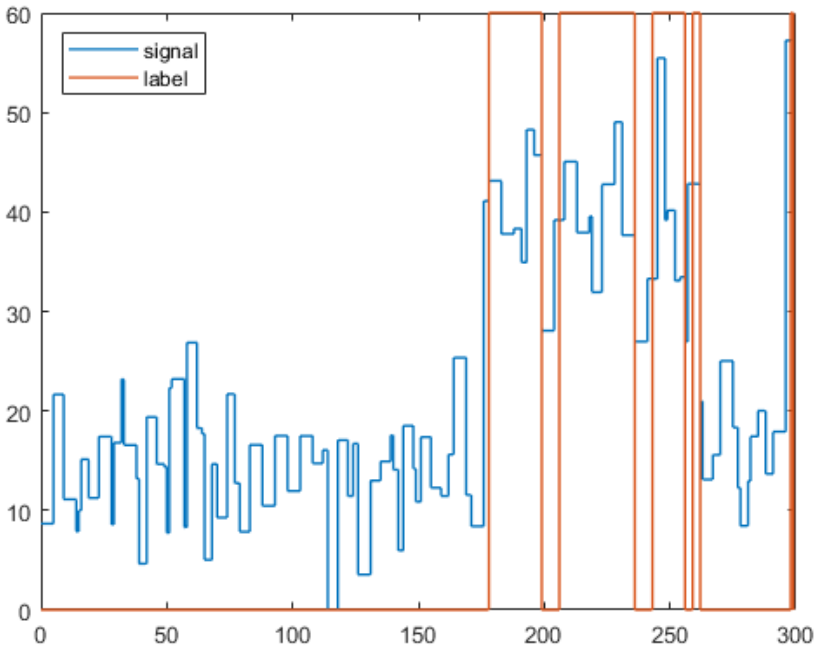
**Example 4.3.1** *Consider the CPU monitoring example from Ex. 3.0.1 where if the processor temperature goes above a certain threshold, the data is labeled with 1.In this example, dataset $\mathcal{D}$ that was used in Ex. 3.0.1 is modified by changing some of the labels from 0 to 1. Parametric formulas over the system variables $\{x_0\}$ with at most $1$ parameters $\mathcal{F}^{\leq 1}$ are generated using* (4.18)*. Parameter domains of each parameter are defined as follows:*
$p_a, p_b \in \{i \mid i = 0, \ldots, 5\}$ *for* $\mathbf{A}_{[p_a,p_b]}$,$\mathbf{P}_{[p_a,p_b]}$,
$p_{x_0} \in \{10i \mid i = 0, \ldots, 5\}$,
*Bound is set to $B = 1$ and sub-formula limit is set to $p = 2$. Then Alg. 2 is run with the dataset $\mathcal{D}$ defined.*

*Resulting formula by running $FormulaSynthesis(\mathcal{F}, B, \mathcal{D}, p)$ is shown in* (4.19).

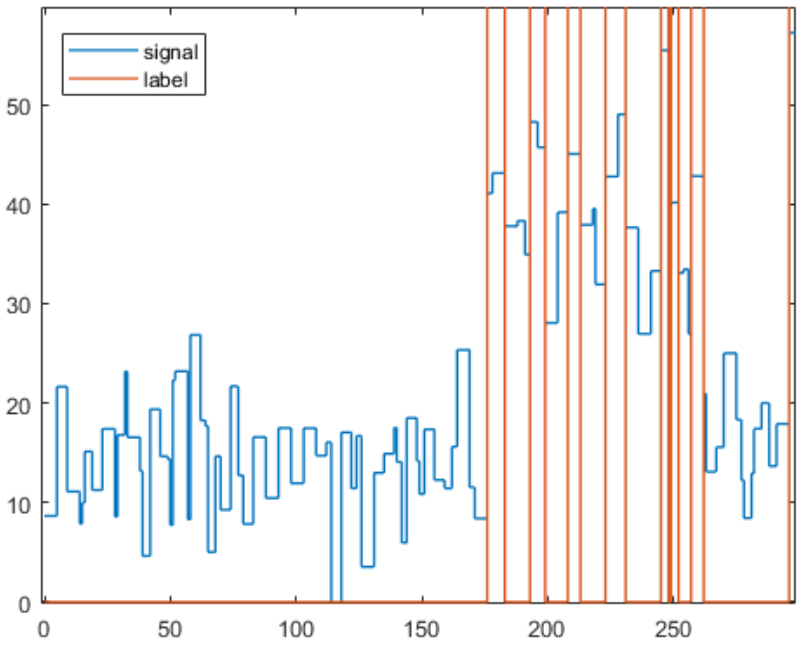
$$\begin{aligned} \phi &= \phi_1 \vee \phi_2 \\ \phi_1 &= (\mathbf{A}_{[0,2]}(x_0 > 30)) \\ \phi_2 &= (x_0 > 40) \end{aligned} \tag{4.19}$$

*Each sub-formula found at (4.19) gives information about the events that caused temperature to go above a certain level. Formula $\phi_1$ states that if cpu usage ($x_0$) was higher than $30$ for last 3 time steps, the processor temperature goes above the certain threshold. Formula $\phi_2$ states if cpu usage is higher than $40$ at the current time step, processor temperature goes above the certain threshold. For each sub-formula $\phi_1$ and $\phi_2$, generated labels on the dataset is shown in Fig. 4.7. In this example all the parametric ptSTL formulas generated according to (4.18) are optimized using $FormulaSynthesis(\mathcal{F}, B, \mathcal{D}, p)$ method. Then among these optimized formulas, 2 formulas that mimics the label in the given dataset are combined so that the combination of them gives the highest True Positive rate. In Fig. 4.1, it can be seen that evaluations of $\phi_1$ and $\phi_2$ generates different labels which results $TP^{\#}(\phi_1, \mathcal{D}) = 40$, $FP^{\#}(\phi_1, \mathcal{D}) = 0$, $TP^{\#}(\phi_2, \mathcal{D}) = 68$, $FP^{\#}(\phi_2, \mathcal{D}) = 0$.*

*Total mismatch count of 300 points is computed as 0 which leads to an accuracy of 100%. A sample trace $\mathbf{x}$ and the generated label $l^{\phi}$ using formula $\phi$ is given in Fig.*




(a) Label generated by evaluating $\phi_1$ on dataset $\mathcal{D}$




(b) Label generated by evaluating $\phi_2$ on dataset $\mathcal{D}$

Figure 4.7: (a) (b)

*4.8*

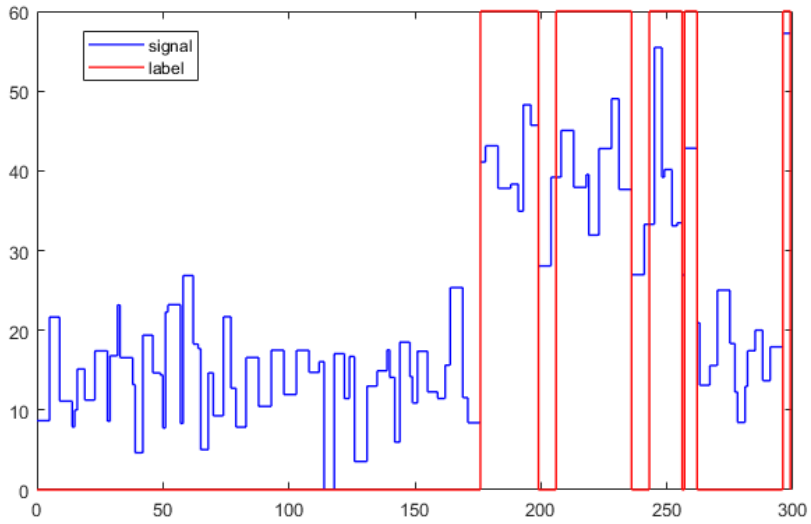



Figure 4.8: Signal and the label generated by evaluating the formula $\phi$ on the signal

*This example shows that if an unwanted event occurs due to a number of different reasons, the proposed formula synthesis method finds these different reasons and combines them to find a ptSTL formula such that this formula mimics the label in the dataset.*

# CHAPTER 5

# EXPERIMENTS AND RESULTS

In this chapter, results obtained by using the proposed method to solve Prob. 3.0.1 are presented on different case studies. For each case study, the system model, the dataset generation process and results obtained by running the propsed method will be explained in detail.

## 5.1 Case Studies

### 5.1.1 Aircraft Dataset

In this case study, Aircraft Longitudinal Flight Control model from Simulink [1] is used. Pilot's command ($pilot$) given from stick is the input and this input is set as the target point of the aircraft. Controller tries to reach the target point by generating internal commands according to the current pitch angle ($alpha$) of the aircraft. Additionally, this system can be perturbed by a simplified Dryden wind gust model that generates the gust values which are denoted with $qGust$ and $wGust$ in the model. A sample trace of the system generated by simulation which contains $pilot$, $alpha$, $qGust$ and $wGust$ system variables is given in Fig 5.1.

#### 5.1.1.1 Dataset Generation

In this case study, the aim is to find the causes which would disturb the aircraft's longitudinal motion. If the difference between the $alpha$ angles which are obtained by simulating the not perturbed and perturbed system is above a given threshold, these

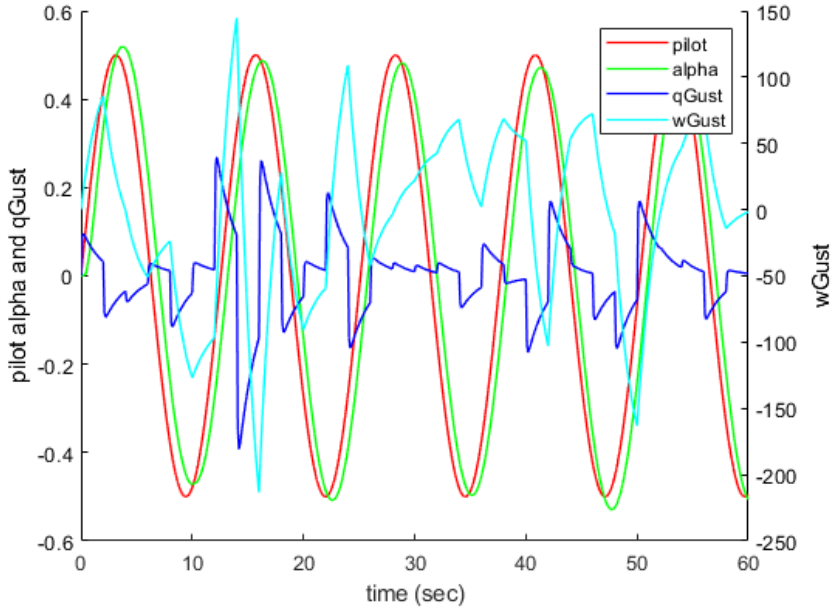



Figure 5.1: Aircraft Longitudinal Flight Control Example.

instances are labeled with 1. For this purpose, two scenarios are created. In the first one, normal operation (in which the environment does not contain wind) is simulated by setting the noise power of the white noise generator ($noise$) to 0 and supplied a sinosoidal input from $pilot$ input. The longitudinal angle $alpha$ of the aircraft is collected (denoted as $alpha_0$). In the second scenario, noise power and sample time of the white noise generator are set to 1000 and 2 seconds respectively. Longitudinal angle of the aircraft is again collected and it is denoted as $alpha_1$. The difference between $alpha_1$ and $alpha_0$ is computed for each time point where this difference can be interpreted as how much the aircraft is disturbed. If this difference is above a threshold at a time $t$ (which is controlled by whether 5.1 is satisfied or not), a label with value 1 is assigned to the time point $t$. Label 0 is assigned for the remaining time points.

$$|alpha^0 - alpha^1| > 0.06 \tag{5.1}$$

Generation of dataset $\mathcal{D}$ is done by simulating the model 5 times for each trace with a length of 600 by setting the seed parameter of the Band-Limited White Noise generator to $k$ for $k^{th}$ trace. Signal $\mathbf{x}$ in dataset $\mathcal{D}$ contains the pilot stick command ($pilot$), aircraft's pitch angle ($alpha$) and the output signals of the Dryden Wind Gust Model

($wGust$ and $qGust$) as shown in (5.2):

$$x_i = \{pilot_i, alpha_i^1, wGust_i, qGust_i\} \tag{5.2}$$

Then dataset is labeled according to 5.1, and in the end, out of 3000, 477 of the data points are labeled with 1 and rest is labeled with 0.

An example that shows the difference $(alpha^0 - alpha^1)$ and the label assigned according to (5.1) is shown in Fig. 5.2.

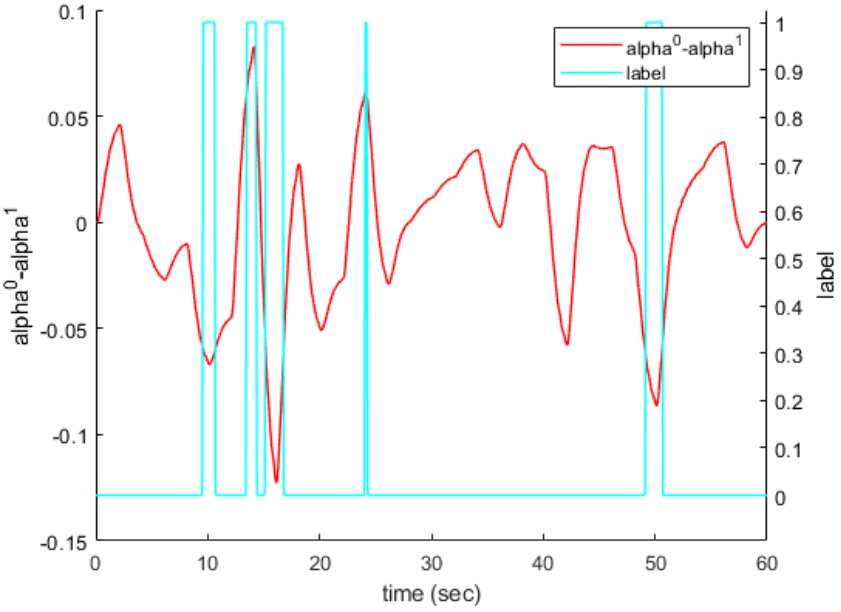



Figure 5.2: $alpha^0 - alpha^1$ and generated label $l$

#### 5.1.1.2 Algorithm Inputs

Parametric formulas over the system variables $\{alpha, pilot, wGust, qGust\}$ with at most 2 parameters $\mathcal{F}^{\leq 2}$ are generated using (4.18). Parameter domains of each parameter are defined as:

$p_a, p_b \in \{2i \mid i = 0, \ldots, 15\}$ for $\mathbf{A}_{[p_a,p_b]}$,$\mathbf{P}_{[p_a,p_b]}$,
$p_{alpha} \in \{-0.5 + 0.05i \mid i = 0, \ldots, 20\}$,
$p_{pilot} \in \{-0.5 + 05i \mid i = 0, \ldots, 20\}$,

$p_{wGust} \in \{-240 + 30i \mid i = 0, \ldots, 15\}$,
$p_{qGust} \in \{-0.4 - 0.05i \mid i = 0, \ldots, 14\}$.

Bound is set to $B = 5$ and sub-formula limit is set to $p = 4$.

#### 5.1.1.3 Results

Resulting formula is given in (5.3) when Alg. 2 is run with the dataset defined.

$$\begin{aligned}
\phi &= \phi_1 \vee \phi_2 \vee \phi_3 \vee \phi_4 \\
\phi_1 &= (\mathbf{P}_{[4,10]}(qGust < 0)) \wedge (wGust < -120) \\
\phi_2 &= (wGust > 120) \wedge (\mathbf{A}_{[14,14]}(pilot > -0.4)) \\
\phi_3 &= \mathbf{P}_{[2,2]}((alpha < 0.3) \wedge (wGust < -120)) \\
\phi_4 &= (\mathbf{A}_{[4,16]}(qGust > 0.1)) \wedge (pilot < -0.4)
\end{aligned} \tag{5.3}$$

Each sub-formula found at (5.3) gives information about the events that caused disturbance in this system. Resulting formula states that a disturbance will occur if any of these following events is encountered:

- $\phi_1$: $qGust$ was lower than $0$ for some time between the last 10 time steps to last 4 time steps at least once and $wGust$ is less than $-120$ in current time step
- $\phi_2$: $wGust$ is greater than $120$ and $14$ time steps ago $pilot$ was greater then $-0.4$
- $\phi_3$: $alpha$ was less than $0.3$ and two steps ago $wGust$ was less than -120
- $\phi_4$: $qGust$ was higher than $0.1$ for each step between last 4 and 16 steps and $pilot$ is less than $-0.4$ in the current step

According to the formula $\phi$ found as 5.3, out of $3000$ data points in $\mathcal{D}$, $437$ points are labeled with 1. TP, FP, TN and FN evaluations are as follows:

- $TP^{\#}(\phi, \mathcal{D}) = 419$

- $FP^{\#}(\phi, \mathcal{D}) = 18$
- $TN^{\#}(\phi, \mathcal{D}) = 2505$
- $FN^{\#}(\phi, \mathcal{D}) = 58$

Total mismatch count of 3000 points is computed as 76 which leads to an accuracy of 97.46%. Resulting formula $\phi$ is computed in 3350 seconds on a PowerEdge T430 machine with Intel Xeon E5-2650 12C/24T processor. It it also important to point out that resulting formula is defined over $4$ system variables, includes $11$ operators and $15$ parameters. Existing formula synthesis methods in the field are validated on simpler formulas because of the computational complexity. This example is an evidence that this method can synthesize complex formulas from labeled dataset efficiently.

### 5.1.2 Traffic System Dataset

This case study is a traffic system which consists 6 links (roads) and 2 traffic signals (traffic lights) and modeled as a piecewise affine system. The traffic system is shown in Fig. 5.3. The state vector of this model consists the number of vehicles (denoted with $x^i$) on link $i$, configuration of traffic signals (denoted with $s^j$) on link $j$. Details of this system is given in [36].

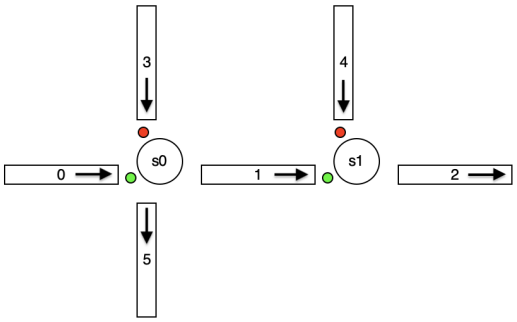



Figure 5.3: Traffic network containing 2 signals and 6 links.

In this system capacity of the links which states the maximum number of vehicles that can be on link $i$ are defined as 40 and 20 for horizontal links ( denoted with $0$, $1$ and

2 on Fig. 5.3 ) and vertical links (denoted with 3, 4 and 5 on Fig. 5.3 ), respectively. Thus the number of vehicles on link $i$ denoted with $x^i$ can take values as follows:

- $x^i \in [0, 40]$ for $i \in \{0, 1, 2\}$
- $x^i \in [0, 20]$ for $i \in \{2, 3, 4\}$

Configuration of traffic signals $s^j$ can be 0 or 1. $s^j$ being 0 and 1 means that traffic is allowed in horizontal and vertical direction, respectively. The aim in this case study is to find the reasons that causes congestion on link $x^1$.

#### 5.1.2.1 Dataset Generation

In this case study, dataset $\mathcal{D}$ is generated as follows:

- System is simulated 20 times with random initial conditions for 100 steps.
- Each simulation trace is labeled according to (5.4)

$$l_t = \begin{cases} 1 \text{ if } (x_t^1 > 30) \\ 0 \text{ otherwise} \end{cases} \tag{5.4}$$

Which means that at time $t$, if number of vehicles on link 1 is more than 30 ($75\%$ of the link capacity), this time point is labeled with 1. The generated dataset consists 2000 data points where 456 time instances are labeled with 1. Fig. 5.4 shows a sample trace of the system. The label assigned to the signal according to (5.4) is given in Fig. 5.5.

#### 5.1.2.2 Algorithm Inputs

Parametric formulas over the system variables $\{x^i \mid i = 0, \ldots, 5\} \cup \{s^0, s^1\}$ with at most 2 operators $\mathcal{F}^{\leq 2}$ are generated using (4.18). The parameter domains are defined as:

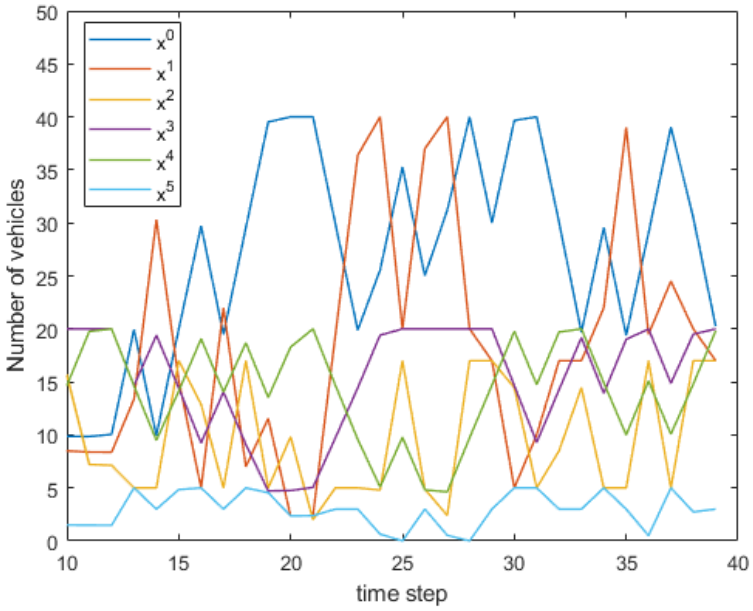



Figure 5.4: Number of vehicles on each link.

$p_a, p_b \in \{i \mid i = 0, \ldots, 5\}$ for$\mathbf{A}_{[p_a,p_b]}$,$\mathbf{P}_{[p_a,p_b]}$,
$p_x \in \{5i + 10 \mid i = 0, \ldots, 6\}$ for $x \in \{0, 1, 2\}$,
$p_x \in \{5i + 5 \mid i = 0, \ldots, 4\}$ for $x \in \{3, 4, 5\}$.

Since the aim in this case study is to find the reasons that lead to congestion, parametric formulas generated ( $\mathcal{F}^{\leq 2}$ ) are modified as in (5.5).

$$\mathcal{F}^{\leq 2,s} = \{\mathbf{P}_{[1,1]}\phi \mid \phi \in \mathcal{F}^{\leq 2}\} \tag{5.5}$$

By adding $\mathbf{P}_{[1,1]}$ to the root node of the syntax tree of each parametric formula $\phi \in \mathcal{F}^{\leq 2}$, evaluation of the new formula on the current step is shifted by 1 time step to the past. This way, each optimized formula will detect the congestion on the link 1 time step before it happens.

Alg. 2 is run with the traffic system dataset $\mathcal{D}$, the parametric formula set $\mathcal{F}^{\leq 2,s}$, error bound $B = 20$, and formula bound $p = 3$.

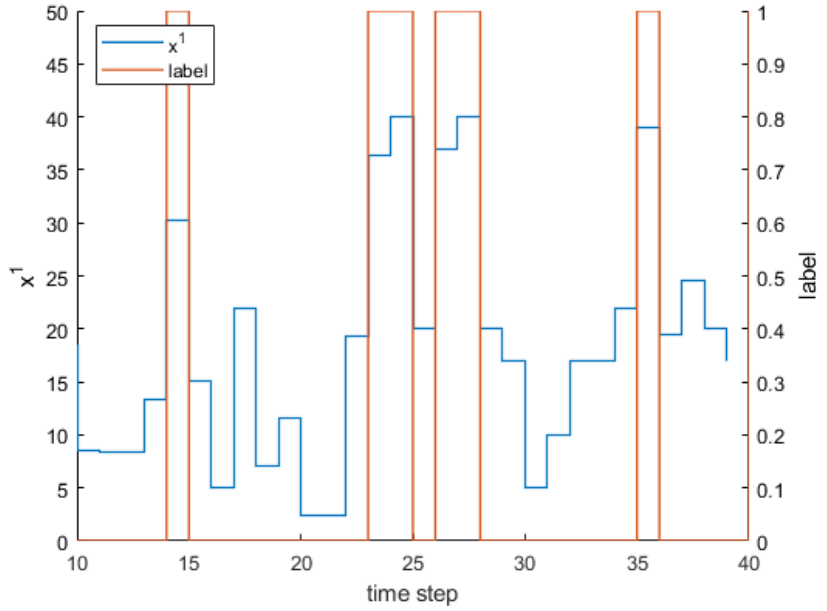



Figure 5.5: Number of vehicles on link $x^1$ and assigned label.

#### 5.1.2.3 Results

The optimal formula $\phi$ that the algorithm returns [1] is given in (5.6) when Alg. 2 is run with the dataset defined.

$$\begin{aligned}
&\phi = \phi_1 \vee \phi_2 \vee \phi_3 \\
&\phi_1 = \mathbf{P}_{[1,1]}((x^1 > 15) \wedge (s^1 = 1) \wedge (s^0 = 0)) \\
&\phi_2 = \mathbf{P}_{[1,1]}((x^1 > 25) \wedge (s^1 = 1)) \\
&\phi_3 = \mathbf{P}_{[1,1]}((x^4 < 10) \wedge (s^1 = 1) \wedge (s^0 = 0))
\end{aligned} \tag{5.6}$$

Each sub-formula from (5.6) gives information about the events that caused congestion on link 1. Each sub-formula states that a congestion will occur in the next time step if any of these events is encountered in the system.

- $\phi_1$: if $s^1$ blocks link 1 while $s^0$ allows flow of vehicles from link 0 to link 1

[1] The inequalities over the signals are written as equalities to simplify the presentation. $s^i > 0$ is equivalent to $s^i = 1$ since $s^i \in \{0, 1\}$.

when there are more than 15 vehicles on link 1

- $\phi_2$: if $s^1$ blocks link 1 when there are more than 25 vehicles on it
- $\phi_3$: if $s^1$ blocks link 1 while $s^0$ allows flow of vehicles from link 0 to link 1 when there are less than 10 vehicles on link 0

According to formula $\phi$ given in 5.6, out of $2000$ data points in $\mathcal{D}$, $484$ points are labeled with 1. TP, FP, TN and FN evaluations are as follows:

- $TP^{\#}(\phi, \mathcal{D}) = 454$
- $FP^{\#}(\phi, \mathcal{D}) = 30$
- $TN^{\#}(\phi, \mathcal{D}) = 1514$
- $FN^{\#}(\phi, \mathcal{D}) = 2$

Total mismatch count of 2000 points is computed as 32 which leads to an accuracy of 98.4%. Formula $\phi$ is computed in 1205 seconds on a PowerEdge T430 machine with Intel Xeon E5-2650 12C/24T processor.

# CHAPTER 6

## EXTENSION OF THE METHOD FOR SINGLE LABEL DATASET

The dataset $\mathcal{D}$ (3.1) that was used in Prob. 3.0.1 had continuously labeled signals. For a signal $\mathbf{x}$ in continuously labeled dataset, at each time step a label is assigned as defined in (3.1). In some cases, it might not be possible to perform an evaluation and assign a label at each time point. Rather a single label can be assigned to the signal according to the satisfaction of a property or occurence of an event along the signal. In this part, the problem given in Prob. 3.0.1 will be extended for datasets that has a single label for each signal. This problem can be thought as a standard binary classification problem since there is only one label (either $0$ or $1$) for each signal. The label specifies whether the signal $\mathbf{x}$ is in the class $0$ or $1$. The dataset for signals that has a single label is defined as in (6.1).

$$\begin{aligned}\mathcal{D} = \{(\mathbf{x}, \mathbf{l}) \mid \mathbf{x} =& x_0, x_1, \ldots, x_K, \mathbf{l} \in \{0, 1\} \text{ and} \\ & x_t \in \mathbb{R}^n, t = 0, \ldots, K\},\end{aligned} \tag{6.1}$$

From a classification point of view, signals $\mathbf{x}$ with label $\mathbf{l} = 1$ belongs to class $1$ and signals $\mathbf{x}$ with label $\mathbf{l} = 0$ belongs to class $0$.

Another difference of the single labeled dataset from the continuous labeled dataset is the generation of a label ($\mathbf{l}^\phi$) by evaluation of a ptSTL formula $\phi$ on the signal $\mathbf{x}$. Generation of the label ($l^\phi$) by evaluating a formula $\phi$ on a signal $\mathbf{x}$ is given in (6.2)

$$l^\phi = \begin{cases} 1 \text{ if } \exists t' \in [0, 1, \ldots K] : (\mathbf{x}, t') \models \phi \\ 0 \text{ otherwise} \end{cases} \tag{6.2}$$

The difference in the generation of the label for single labeled signals (6.2) from continuous labeled signals (3.2) is that, in (3.2) at each time step, formula $\phi$ is being

evaluated and a label is assigned each time step. But as shown in (6.2), for single labeled signals if there is a single time step that the formula is satisfied on signal $\mathbf{x}$, label $1$ is assigned to signal $\mathbf{x}$. Thus as soon as the formula is satisfied at any time on signal $\mathbf{x}$ no further computation is required.

Informally, the goal in this part is to synthesize a ptSTL formula $\phi$ such that, the label ($\mathrm{l}^\phi$) that will be acquired by evaluating $\phi$ on signal $\mathbf{x}$ matches the label $\mathrm{l}$ in the dataset (3.1) which simply means the formula $\phi$ classifies the signals given in dataset $\mathcal{D}$ correctly.

After providing the information for single labeled dataset and the generation of the label ($\mathrm{l}^\phi$), the problem in this chapter is formally defined as in Prob. 6.0.1.

**Problem 6.0.1** *Given a dataset $\mathcal{D}$ as in* (6.1)*, find a ptSTL formula $\phi$ such that for any $(\mathbf{x}, \mathrm{l}) \in \mathcal{D}$, $\mathrm{l} = \mathrm{l}^\phi$, where $\mathrm{l}^\phi$ is generated as defined in* (6.2)*.*

## 6.1 Monotonicity Properties

In this part, monotonicity properties that were defined for continuous labeled dataset will be extended for single labeled dataset. Firstly, Positive($P^\#(\phi, \mathcal{D})$), Negative ($N^\#(\phi, \mathcal{D})$) True Positive ($TP^\#(\phi, \mathcal{D})$), False Positive ($FP^\#(\phi, \mathcal{D})$), True Negative ($TN^\#(\phi, \mathcal{D})$) and False Negative ($FN^\#(\phi, \mathcal{D})$) numbers that are computed by evaluation of the formula $\phi$ on dataset $\mathcal{D}$ are defined as in (6.3).

$$
\begin{aligned}
P^\#(\phi, \mathcal{D}) &= \sum_{(\mathbf{x},\mathrm{l})\in\mathcal{D}} l^\phi \\
N^\#(\phi, \mathcal{D}) &= \sum_{(\mathbf{x},\mathrm{l})\in\mathcal{D}} \neg l^\phi \\
TP^\#(\phi, \mathcal{D}) &= \sum_{(\mathbf{x},\mathrm{l})\in\mathcal{D}} l \wedge l^\phi \\
FP^\#(\phi, \mathcal{D}) &= \sum_{(\mathbf{x},\mathrm{l})\in\mathcal{D}} \neg l \wedge l^\phi \\
TN^\#(\phi, \mathcal{D}) &= \sum_{(\mathbf{x},\mathrm{l})\in\mathcal{D}} \neg l \wedge \neg l^\phi \\
FN^\#(\phi, \mathcal{D}) &= \sum_{(\mathbf{x},\mathrm{l})\in\mathcal{D}} l \wedge \neg l^\phi
\end{aligned}
\tag{6.3}
$$

The existential operator ($\exists$) used in label generation (6.2) preserves the monotonicity properties.Thus monotonicity properties for parameters that can appear in a parametric ptSTL formula given in Table 4.1 and Table 4.2 for continuous labeled signals are also applicable for single labeled signals. Thus, total number of positives ($P^{\#}(\phi, \mathcal{D})$) and total number of negatives ($N^{\#}(\phi, \mathcal{D})$) that is computed by evaluating a formula $\phi$ on dataset $\mathcal{D}$ increases with parameters that are monotonically increasing and decreases with parameters that monotonically decreasing. Also, since $TP^{\#}(\phi, \mathcal{D})$, $FP^{\#}(\phi, \mathcal{D})$, $TN^{\#}(\phi, \mathcal{D})$ and $FN^{\#}(\phi, \mathcal{D})$ defined in Eq. (6.3) are the masked versions of $P^{\#}(\phi, \mathcal{D})$ and $N^{\#}(\phi, \mathcal{D})$, they also preserves the monotonicity properties [37].

## 6.2 Proposed Method

The proposed method to solve Prob. 6.0.1 is the same method that was used for solving Prob. 3.0.1 with the exception of positive and negative label calculation for a formula $\phi$ on the dataset $\mathcal{D}$ as given in (6.2). Proposed method for single labeled dataset also assumes that classification of signals might depend on more than one rule, thus the iterative construction approach (by disjunction of formulas given as in (4.1)) remains the same and $FormulaSynthesis(\mathcal{F}, B, \mathcal{D}, p)$ (Alg. 2) is used for iterative construction. Note that only the calculation of the True Positives in Line 4 of Alg. 2 is changed by definition for single labeled datasets. For finding the sub-formulas ($\phi_i$) again set of parametric ptSTL formulas is defined as in (4.18) and optimization of the parameters in these formulas is performed. Since monotonicity properties that was defined for continuous labeled dataset is preserved for single labeled dataset, parameter optimization methods $ParameterSynthesis(\phi, B, \mathcal{D})$ and $DiagonalSearch(\phi, B, \mathcal{D}, l_1, u_1, \delta_1, l_2, u_2, \delta_2)$ (Alg. 1) remains the same.Note that only computation of $TP^{\#}(\phi, \mathcal{D})$ (Line 16 of Alg. 1) and $FP^{\#}(\phi, \mathcal{D})$ (Line 13 of Alg. 1) is changed by definition for single labeled dataset.

### 6.3 Case Study

In this case study, a dataset that contains the information and vessel positions of the transport vessels that cruises in the Mediterranean Sea [38] is used. The raw dataset contains:

- trip id that is defined uniquely for each trip
- speed, longitude and latitude position, course and heading informations of the vessels with timestamp
- departure and arrival ports of the each trip
- ship id and ship type
- calculated arrival time to destination

There are 755 trips in this dataset and the purpose in this case study is to correctly classify the vessels whose destination port is Tuzla Port. For better understanding of the dataset, most commonly arrived ports and number of trips that has ended in these ports are given in Table 6.1

Table 6.1: Most Commonly Arrived Ports and Number of Trips that Ended in the Corresponding Port

| Destination Port | Total Number of Trips |
|---|---|
| BARCELONA | 126 |
| PALMA DE MALLORCA | 103 |
| VALENCIA | 99 |
| LIVORNO | 51 |
| GENOVA | 47 |
| TUZLA | 32 |
| VALLETTA | 30 |

While preparing the dataset that will be used in this case study, signals ($\mathbf{x}$) are generated by using the longitude (denoted with $lon$) and latitude (denoted with $lat$) information of the vessels with timestamp. Then label $\mathbf{l}$ of the signal is set to 1 if the

corresponding trip destination is Tuzla Port and it is set to 0 otherwise. Then 32 trips whose final destination is Tuzla Port and 32 trips (chosen randomly) whose final destination is not Tuzla Port are selected from these 755 trips and dataset $\mathcal{D}$ is composed using these 64 trips. The routes of the vessels that appears in the generated dataset $\mathcal{D}$ after the preprocessing is shown in Fig. 6.1.

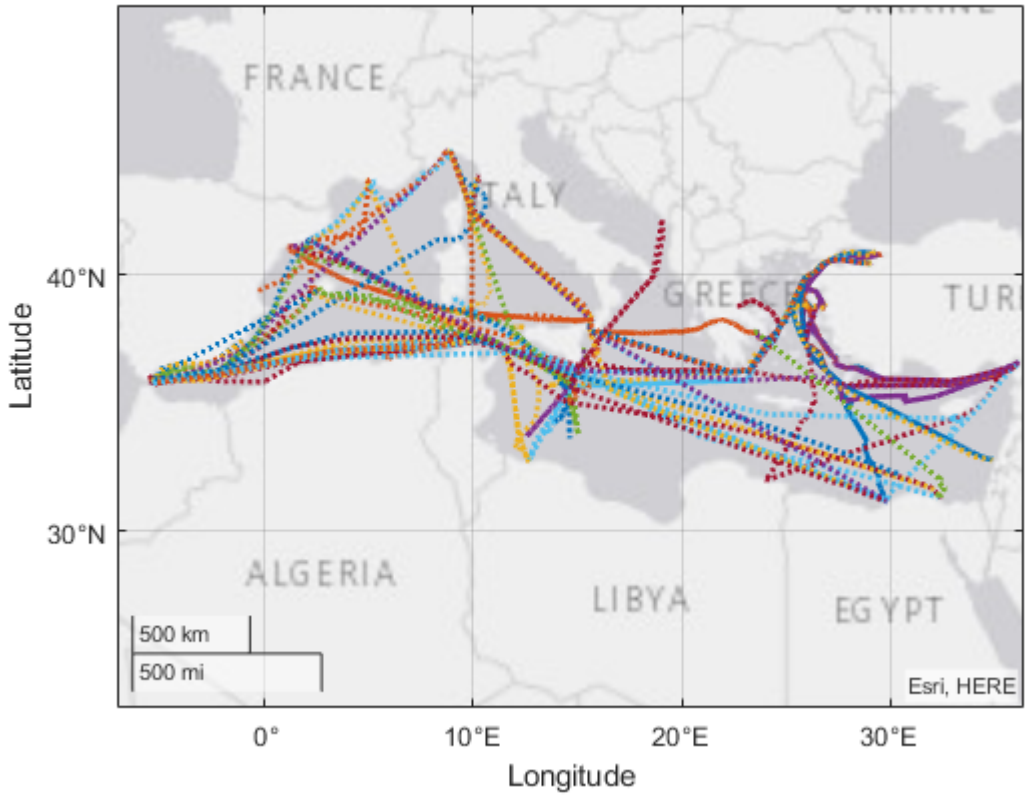



Figure 6.1: Routes of the vessels that cruises in Mediterranean Sea.

Parametric formulas over the system variables $\{lon, lat\}$ with at most 3 parameters $\mathcal{F}^{\leq 3}$ are generated using (4.18). Parameter domains for each parameter are defined as:

$p_a, p_b \in \{200i \mid i = 0, \ldots, 5\}$ for $\mathbf{A}_{[p_a,p_b]}$,$\mathbf{P}_{[p_a,p_b]}$,

$p_{lon} \in \{29^{\circ} + 3' \times i \mid i = 0, \ldots, 12\}$,

$p_{lat} \in \{40^{\circ} + 30' \times i \mid i = 0, \ldots, 4\}$,

Bound is set to $B = 0$ and sub-formula limit is set to $p = 2$.

Resulting formula is given in (6.4) when Alg. 2 is run with the dataset ($\mathcal{D}$).

$$\phi = \phi_1 \vee \phi_2 \tag{6.4}$$
$$\phi_1 = (lat < 41°30' \wedge lat > 40° \wedge lon > 29°15')$$
$$\phi_2 = (lon > 29°6') \wedge (A_{[0,200]}(lon < 29°9'))$$

Each sub-formula found at (6.4) gives information about the classification rules. According to result, if any of the following conditions is satisfied at any time during the trip, this trip will is labeled with $1$ and it is classified as it's final destination is Tuzla Port.

- $\phi_1$: If a vessel's latitude is between 40° and $41°30'$ and it's longitude is greater than $29°15'$
- $\phi_2$: If a vessel's longitude is greater than $29°6'$ and if this vessel's longitude was less than $29°9'$ for last 200 minutes

Location of Tuzla port, destination points of vessels that are classified by only $\phi_1$, by only $\phi_2$ and both by $\phi_1$ and $\phi_2$ is shown in Fig. 6.2.

According to the formula $\phi$ found in (5.3), out of 64 signals in $\mathcal{D}$, $32$ points are labeled with 1. TP, FP, TN and FN evaluations are as follows:

- $TP^{\#}(\phi, \mathcal{D}) = 32$
- $FP^{\#}(\phi, \mathcal{D}) = 0$
- $TN^{\#}(\phi, \mathcal{D}) = 32$
- $FN^{\#}(\phi, \mathcal{D}) = 0$

Total mismatch count of 64 points is computed as 0 which leads to an accuracy of $100\%$.

To comment on the results, sub-formula $\phi_1$ simply defines an area and states that, if a vessel appears appears between the stated latitudes and comes further east then the $29°15'$, it will arrive to Tuzla Port. Sub-formula $\phi_2$ explains the situation if a vessel's

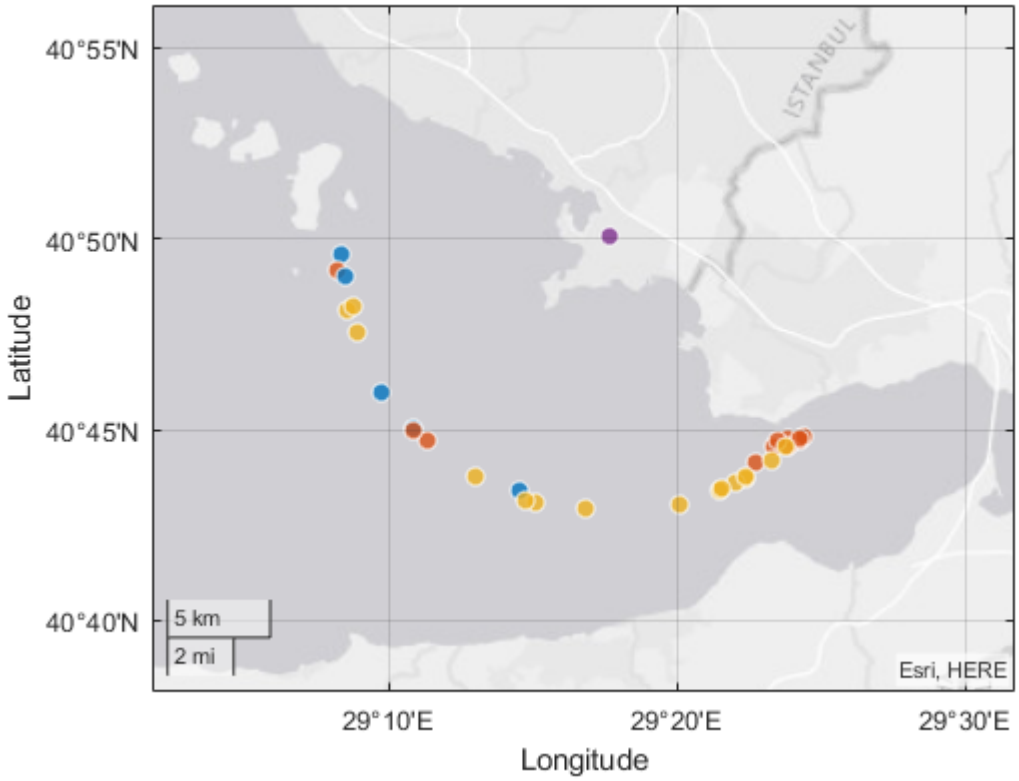



Figure 6.2: Location of Tuzla Port (purple) and destination points of vessels that are classified using only $\phi_1$ (red), using only $\phi_2$ (blue) and classified by both $\phi_1$ and $\phi_2$ (yellow).

longitude further east from $29°6'$ at current time step and it's longitude was further west from $29°9'$ for last 200 time steps it will arrive to Tuzla Port. Sub-formula $\phi_2$ simply classifies the vessels that goes from west to easy direction in the area. Vessels that started their cruise from Tuzla Port and goes to west are not considered as going to Tuzla Port with the $(A_{[0,200]}(lon < 29°9'))$ formula. Lastly, when the resulting formula given in (6.4) is evaluated on the whole dataset which contains 755 trips, total mismatch is computed as 13.

In this chapter, classification of single labeled time series data using tempral logic is studied. Monotonicity properties that was defined for datasets with signals that are continious labeled is extended for datasets with signals that have a single label. And classification is achieved using efficient algorithms. Lastly proposed method is shown on a case study.

# CHAPTER 7

# CONCLUSION

Finding the cause of the erroneous behaviors in cyber-physical systems that performs complex tasks is a challenging process. As the system gets more complex, it becomes harder to locate the errors manually. In this study, the main aim was to find a method that would explain the root causes of the erroneous behaviour in a system. One of the important features of the method is that it should provide a human-readable and intuitive answer with the purpose of helping the system engineer to understand the underlying cause of the erroneous behaviour. Another feature that the method should have is that the method should be efficient in computation. In addition, the method should be system independent. i.e., it should be straight-forward to apply the method to different systems.

Since one of the main concerns of the method is to be intuitive and human readable, temporal logic is used because of its expressiveness and its resemblance to natural language. Also since aim of the method is to find the root causes of the erroneous behaviour, a past time extension of the temporal logic, Past Time Signal Temporal Logic (ptSTL) is used. In Chapter 2 preliminary information about temporal logic (Linear Temporal Logic, Signal Temporal Logic and Past Time Signal Temporal Logic) is provided. In order to attain a system independent approach, a data-driven method is developed. In particular, a set of labeled system traces is used to analyze the cause of errors. Thus, the main problem is defined as finding a ptSTL formula that explains the labeled events (errors) in the dataset.

In Chapter 3 both the dataset, which is basically trace of the considered system, and problem is formally defined. In Chapter 4, the proposed method to solve the main problem is explained in detail. Firstly, previously defined monotonicity properties be-

tween a parameter value and satisfaction of the STL formula are extended by defining the monotonicity between the parameter and the number of positive/negative labels that is obtained by evaluation of the formula on a dataset. Then, success measures are defined for the method and using these success measures and extended monotonicity properties, a parameter optimization method is proposed. This parameter optimization method basically optimizes the parameters in a parametric ptSTL formula such that it maximizes the number of true positives while keeping the error (false positives) under a predefined bound which allows the method to produce an output in controllable error margins. In the parameter optimization method, thanks to the monotonicity properties, computational complexity of the optimization is decreased drastically. Then using an assumption such as an erroneous behaviour might be caused by many reasons, an iterative formula construction method is provided. With the iterative construction method, instead of optimizing a long and complex formula, sub-formulas could be optimized and combined with each other, which also decreases the computational complexity of the computation. Iterative construction approach can generate quite complex and long formulas such that the formula synthesis methods from literature were not able to generate that complex/long formulas due to the computational complexity. Each main issue (monotonicity properties, parameter optimization method and iterative formula construction method) are supported by simple and intuitive examples with the purpose of clarifying the concepts.

In Chapter 5, two case studies (Aircraft Longitudinal Flight Control Model and Traffic System) are provided. On these case studies, the efficiency of the method and ability to generate complex ptSTL formulas with high accuracy are shown.

Lastly in Chapter 6, the main problem given in the thesis is modified. Up until this part, the problem was defined for continuously labeled signals. In Chapter 6, label of the signal is changed from continuous to single. Preservation of the monotonicity properties that was defined for continuously labeled datasets on single labeled datasets is shown. In the end, with the change of the labeling in the dataset, evaluation of a formula on a signal is defined and the approach to solve the problem is kept the same. Extension of the proposed method for this version is shown on a case study which again showed the ability of the proposed method to classify the signals with high accuracy, intuitive and human-readable way.

To summarize, main contributions of this study are:

- Monotonicity properties in the literature is extended.
- With the extended monotonicity properties, an efficient parameter synthesis algorithm is implemented.
- An iterative formula construction method which decreases the computational complexity and controls the error that can appear in the final result is presented.

## 7.1 Future Studies

In the current study, the iterative construction method constructs the final formula by iteratively combining sub-formulas with each other using disjunction operator by maximizing the correctly identified positive instances (True Positives) and limiting the incorrectly identified positive instances (False Positives). A future research direction is to handle the complementary problem. Instead of disjunction, cunjunction of sub-formulas will be carried by maximizing the correctly identified negative instances (True Negatives) and limiting the incorrectly identified negative instances(False Negatives). Another future research plan is to improve the classification method presented in Chapter 6. It is planned to add a cost function to the method with the purpose of early detection of the classification of signal.

# REFERENCES


[1] MATLAB, *version (R2016b)*. Natick, Massachusetts: The MathWorks Inc., 2016.

[2] A. Donze, "On signal temporal logic," in *RV 2013, LNCS 8174*. (A. Legay and S. Bensalem, eds.), pp. 382–383, Springer Berlin Heidelberg, 2013.

[3] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. 2008.

[4] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, (New York, NY, USA), pp. 239–248, ACM, 2015.

[5] M. Vazquez-Chanlatte, J. V. Deshmukh, X. Jin, and S. A. Seshia, "Logical clustering and learning for time-series data," in *Computer Aided Verification* (R. Majumdar and V. Kunčak, eds.), (Cham), pp. 305–325, Springer International Publishing, 2017.

[6] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*, pp. 135–175. Cham: Springer International Publishing, 2018.

[7] E. Bartocci, L. Bortolussi, and G. Sanguinetti, "Data-driven statistical learning of temporal logic properties," in *FORMATS 2014, LNCS, vol 8711*, pp. 23–37, Springer International Publishing, 2014.

[8] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta, "Temporal logic inference for classification and prediction from data," HSCC '14, (New York, NY, USA), pp. 273–282, ACM, 2014.

[9] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. 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, (New York, NY, USA), pp. 1–10, ACM, 2016.

[10] X. Jin, A. Donze, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1704–1717, Nov 2015.

[11] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, "Telex: learning signal temporal logic from positive examples using tightness metric," *Formal Methods in System Design*, Jan 2019.

[12] E. Aydin Gol, "Efficient online monitoring and formula synthesis with past stl," in *5th IEEE International Conference on Control, Decision and Information Technologies (Codit)*, 2018.

[13] E. Asarin, A. Donze, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Proceedings of the Second International Conference on Runtime Verification*, RV'11, (Berlin, Heidelberg), pp. 147–160, Springer-Verlag, 2012.

[14] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.

[15] A. Donzé, T. Ferrère, and O. Maler, "Efficient robust monitoring for stl," in *Computer Aided Verification* (N. Sharygina and H. Veith, eds.), pp. 264–279, Springer Berlin Heidelberg, 2013.

[16] C. Yoo and C. Belta, "Rich time series classification using temporal logic," in *Proceedings of Robotics: Science and Systems*, (Cambridge, Massachusetts), July 2017.

[17] S. K. Aydin and E. Aydin Gol, "Synthesis of monitoring rules with stl," *Journal of Circuits, Systems, and Computers.*, vol. Early Access, 2020.

[18] I. Saglam and E. A. Gol, "Cause mining and controller synthesis with stl," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 4589–4594, 2019.

[19] S. Kağan Aydın and E. A. Göl, "Optimizing parameters of signal temporal logic formulas with local search," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2019.

[20] S. K. Aydin and E. A. Gol, "On the use of genetic algorithms for synthesis of signal temporal logic formulas," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2018.

[21] A. Ketenci and E. A. Gol, "Synthesis of monitoring rules via data mining," in *2019 American Control Conference (ACC)*, pp. 1684–1689, 2019.

[22] J. Kapinski, X. Jin, J. Deshmukh, A. Donze, T. Yamaguchi, H. Ito, T. Kaga, S. Kobuna, and S. Seshia, "St-lib: A library for specifying and classifying model behaviors," in *SAE Technical Paper*, SAE International, 04 2016.

[23] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *Testing Software and Systems* (B. Nielsen and C. Weise, eds.), (Berlin, Heidelberg), pp. 136–151, Springer Berlin Heidelberg, 2012.

[24] B. Hoxha, A. Dokhanchi, and G. Fainekos, "Mining parametric temporal logic properties in model-based design for cyber-physical systems," *International Journal on Software Tools for Technology Transfer*, vol. 20, pp. 1–15, 02 2017.

[25] S. Sankaranarayanan and G. Fainekos, "Falsification of temporal properties of hybrid systems using the cross-entropy method," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, (New York, NY, USA), p. 125–134, Association for Computing Machinery, 2012.

[26] Y. S. R. Annapureddy and G. E. Fainekos, "Ant colonies for temporal logic falsification of hybrid systems," in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, pp. 91–96, 2010.

[27] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proceedings of the 13th ACM International Confer-*

*ence on Hybrid Systems: Computation and Control*, HSCC '10, (New York, NY, USA), p. 211–220, Association for Computing Machinery, 2010.

[28] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems* (P. A. Abdulla and K. R. M. Leino, eds.), (Berlin, Heidelberg), pp. 254–257, Springer Berlin Heidelberg, 2011.

[29] A. Jones, Z. Kong, and C. Belta, "Anomaly detection in cyber-physical systems: A formal methods approach," vol. 2015, 12 2014.

[30] P. J. M. Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. USA: Kluwer Academic Publishers, 1987.

[31] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395 – 408, 2005.

[32] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, p. 10–18, Nov. 2009.

[34] M. Ergurtuna and E. A. Gol, "An efficient formula synthesis method with past signal temporal logic," *IFAC-PapersOnLine*, vol. 52, no. 11, pp. 43 – 48, 2019. 5th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2019.

[35] N. Linial and M. Saks, "Searching ordered structures," *Journal of algorithms*, vol. 6, no. 1, pp. 86–103, 1985.

[36] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 162–172, 2016.

[37] M. Ergürtuna and E. A. Göl, "Classification of time-series data using ptstl,"

in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2020.

[38] *MarineTraffic-Global Ship Tracking Intelligence*, 2018 (accessed 2018-09-24). www.marinetraffic.com.