PARAMETER SYNTHESIS FOR TIMED AUTOMATA VIA PATH ANALYSIS

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

BY

BURKAY SUCU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

SEPTEMBER 2023

Approval of the thesis:

PARAMETER SYNTHESIS FOR TIMED AUTOMATA VIA PATH ANALYSIS

submitted by **BURKAY SUCU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy 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** _____

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

Examining Committee Members:

Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU _____

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

Prof. Dr. Uluç Saranlı
Computer Engineering, METU _____

Assoc. Prof. Dr. Hürevren Kılıç
Computer Engineering, Atılım University _____

Assoc. Prof. Dr. Çiğdem Turhan
Software Engineering, Atılım University _____

Date:07.09.2023

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: Burkay Sucu

Signature :

ABSTRACT

PARAMETER SYNTHESIS FOR TIMED AUTOMATA VIA PATH ANALYSIS

Sucu, Burkay
Ph.D., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Ebru Aydın Göl

September 2023, 65 pages

Parametric timed automata (PTA) extends timed automata (TA) with parameters instead of fixed timing constraints, providing the flexibility to accommodate uncertainties during the design phase. Once the parametric model is obtained, parameter synthesis methods are applied to find the optimal parameter values such that the resulting timed automaton satisfies the specifications.

In this thesis, parameter synthesis strategies for parametric timed automata are developed. The proposed approaches involve accumulating parameter constraints while exploring the automata graph in a depth-first manner. Realizability problems of encountered paths during the graph search are encoded as Mixed Integer Linear Programming (MILP) or Non-Linear Programming (NLP) problems; then, those problems are evaluated in different ways depending on the given specifications. For specifications where a single witness is enough, e.g., reachability, the depth-first search returns the first parameter constraint that makes a path ending in an accepting state realizable. For specifications where no witness must exist, e.g., safety, parameter constraints are found by applying quantifier elimination on encoded problems and accumulated along the depth-first traversal. Depth-first traversal is optimized via pruning

unrealizable branches and guaranteed to terminate via the proposed novel cycle analysis methods. The collected constraints through the depth-first search guarantee that a parameter valuation satisfying these constraints solves the synthesis problem. The developed algorithms are extended to generate the optimal parameter values for a given criteria. The correctness and completeness of the proposed algorithms are presented. All the results are depicted via examples.

Keywords: Parametric timed automata, Parameter synthesis, Safety, Existential liveness, Existential reachability, MILP, NLP

ÖZ

ZAMANLI OTOMAT İÇİN YOL ANALİZİ İLE PARAMETRE SENTEZİ

Sucu, Burkay
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ebru Aydın Göl

Eylül 2023 , 65 sayfa

Zamanlı otomatların zamanlı kısıtlamalarda sabit değerler yerine parametreler kullanılarak genişletilmesiyle elde edilen parametrik zamanlı otomatlar, belirsizliklerin tasarım aşamasında uygun biçimde ele alınabilmesi için esneklik sağlar. Parametrik model elde edildikten sonra, ortaya çıkacak zamanlı otomatın verilen tanımlamaları sağlayacağı en uygun parametre değerlerini bulmak için parametre sentez yöntemleri bu modele uygulanır.

Bu tezde, parametrik zamanlı otomatlar için parametre sentezi stratejileri geliştirilmiştir. Sunulan yaklaşımlar, otomat grafiğinin derin öncelikli aranması sırasında parametre kısıtlamalarının toplanmasını içerir. Derin öncelikli arama sırasında karşılaşılan yolların gerçekleştirilebilirlik problemleri, Karma Tamsayılı Doğrusal Programlama (MILP) veya Doğrusal Olmayan Programlama (NLP) problemleri olarak kodlanır; daha sonra, bu problemler verilen tanımlamalara bağlı olarak farklı şekillerde değerlendirilir. Ulaşılabilirlik gibi tek bir tanığın yeterli olduğu tanımlamalar için, derin öncelikli arama, kabul eden bir konumda biten bir yolu gerçekleştirilebilir kılan ilk parametre kısıtlamasını döndürür. Güvenlik gibi hiçbir tanığın olmaması

gereken tanımlamalar için, parametre kısıtlamaları, kodlanmış problemlere niceleme eleme uygulanarak bulunur ve derin öncelikli arama boyunca toplanır. Derin öncelikli arama, gerçekleştirilemez dalların kesilmesiyle iyileştirilir ve döngüsel yolların analizi için sunulan yeni yöntemlerle bu aramanın sona erdirilmesi garanti edilir. Derin öncelikli arama aracılığıyla toplanan kısıtlamalar, bu kısıtlamaları karşılayan bir parametre değerlemesinin sentez problemini çözeceğini garanti eder. Daha sonra, geliştirilen algoritmalar, belirli bir kriter için en iyi parametre değerlerini bulmak üzere genişletilmiştir. Önerilen algoritmaların doğruluğu ve eksiksizliği gösterilmiştir. Tüm sonuçlar örneklerle açıklanmıştır.

Anahtar Kelimeler: Parametrik zamanlı otomat, Parametre sentezi, Güvenlik, Varoluşsal canlılık, Varoluşsal ulaşılabilirlik, MILP, NLP

To My Dear Family

ACKNOWLEDGMENTS

Many people have kindly helped and generously supported me in many different ways during this challenging PhD journey and contributed to finalizing my thesis.

Among those, first and foremost, I would like to express my deepest gratitude to my wonderful advisor, Assoc. Prof. Dr. Ebru Aydın Göl. Her wisdom and technical lead lighted the way throughout my research and guided me through. Her encouragement supported me when I felt like I failed. Her enthusiasm helped me overcome the difficulties I faced. Her mentorship and expertise have been invaluable all the time. Her friendly and positive attitude is exceptional. I feel very privileged to have the opportunity to work with her. I want to thank her again for all the time she spared for my research discussions, even on her vacation.

I am sincerely grateful to Prof. Dr. Halit Oğuztüzün and Assoc. Prof. Dr. Hürevren Kılıç, who participated in both my Thesis Monitoring Committee and thesis jury. Discussing my research with them during those meetings had always been a joy. I also want to express special thanks to the other two members of my thesis jury, Prof. Dr. Uluç Saranlı and Assoc. Prof. Dr. Çiğdem Turhan. All five jury members have been kind, supportive, positive and friendly. Those manners of the jury members let me enjoy the defense presentation rather than stress about it. I am genuinely grateful to all of them for all the insightful feedback and constructive criticism.

I want to thank my previous advisor, Prof. Dr. Yusuf Sahillioğlu, for his always helpful, supportive and friendly attitude. I also thank my previous committee members, Prof. Dr. Veysi İşler, Prof. Dr. Tolga Can, Prof. Dr. Uğur Güdükbay and Prof. Dr. Ahmet Oğuz Akyüz, who also had been very tolerant and supportive.

I am grateful to my friends who have supported me in different ways throughout this demanding journey. In particular, I would like to express my gratitude and appreciation to Merve Asiler and Elif Koç for their support, tremendous help, encouragement and camaraderie. I had always told them they each deserved half of the diploma if I got it one day. Nothing will be left to me now. I am thankful to my dear friend Ferayi Öztürk, one of the most sincere and inspirational people I know. I am also thankful to Andaç Akarsu, Damla Yazar, Aynur Uluç Keçik, Serhat İnan, İlknur Güneşli, Aysun Asena Aslan, Damla Dokuzoğlu, Seyfullah Demir and Ahmet Yavuz Karacabey, who also pursue or pursued an academic path. Sometimes, we worked together; sometimes, we just talked and shared the pain. So, talking to them has always been relieving and motivating during desperate times. I am thankful to Suna Tarıyan for being an immensely supportive classmate. I am grateful to Ahmet Rıdvan Potur, Burcu Aykan, Eren Aktuğ, Furkan Kara and Kıvanç Kamay, my close friends and teammates at work, for their sarcastic jokes. Those jokes aside, they bought me tons of time when I needed it the most. Ahmet deserves a separate gratitude as he let me stay overnight to run benchmarks on his computer.

Finally, I would like to extend my heartfelt appreciation to my family for their unwavering love. Everyone in my extended family has always supported and encouraged me, and I am grateful to each of them. I hope now I make them proud. Besides, I must give immense credit to my mom, Pervin, my dad, Yaşar, and my brother, Çağatay, for anything I achieved in my life. I am grateful to them for the love, joy and support they provided.

Unfortunately, I cannot list all the people here. Nevertheless, I will always remember the friendship, support, encouragement and inspiration of everyone who even made the slightest touch or just well-wishes.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

TA	Timed Automata
PTA	Parametric Timed Automata
TTS	Timed Transition System
MILP	Mixed Integer Linear Programming
LP	Linear Programming
NLP	Non-Linear Programming

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Real-time systems are in every part of life: flight control systems, medical devices, industrial or home automation, manufacturing assembly lines, communication, anti-missile systems, self-driving cars, traffic lights, etc. When those systems fail to respond to events within specified time limits, the consequences may be costly or catastrophic. For example, a malfunctioning video conferencing service or online trading system can cause user dissatisfaction, poor service quality or economic loss. Problems in online reservation or auction systems can result in missed opportunities or competitive disadvantages. On the more severe side, a crashing plane or a malfunctioning MRI machine may result in the loss of human lives. Therefore, the correctness and completeness of real-time systems have crucial importance.

Before implementing a real-time system, designing one guaranteed to function correctly is vital in reducing business costs of money and time. Verifying those timed systems against formal specifications is a challenging problem and requires a complete model description. Timed automata (TA), an extension of regular finite state automata by timing properties on states and transitions, is an effective formalism to model real-time systems and can be used for such purposes [1–3]. Timing rules, i.e., clock constraints, are defined over the clocks that measure the time since their last reset and they can be reset at transitions. Via timing rules, states and transitions can be access-restricted to specific time intervals. Timed automata are widely used to model real-time systems and analyze the correctness and completeness of modeled systems at design time via verification. Hence, this formalism avoids possibly huge costs and

risks of making or implementing erroneous systems from the very beginning.

During the design phase, some uncertainties may arise due to different trade-offs or incomplete information about the modeled system. When there are such uncertainties tied to unfixed timed properties, timed automata can be extended to parametric timed automata (PTA) by using parameters instead of fixed values on clock constraints of invariants and guards. So, parameters in a PTA help to express the uncertainties due to timing properties that are not fixed a priori. Thus, when the complete description is not available or there are tunable design choices, modeling a system as a parametric timed automaton provides the much-needed flexibility to accommodate the uncertainties during the design phase. Having at least one constraint with a parameter is enough to make a TA a parametric timed automaton (PTA). Once a parametric model is obtained as a PTA, the problem of finding the proper parameter valuation to satisfy the specification emerges.

In this thesis, the parameter synthesis problem for safety, reachability and repeated reachability specifications is studied. Specifically, given a PTA and a specification, our goal is to find a parameter valuation such that

1. the unsafe locations are not reachable (safety),
2. the target locations are reachable (reachability),
3. the target locations are reachable infinitely many times (repeated reachability)

on the resulting timed automaton.

To solve this synthesis problem, a depth-first analysis of the underlying graph structure of the timed automaton is proposed. For a safety specification, there must be no witness, so the graph traversal algorithm evaluates all the paths in terms of realizability and accumulates the parameter constraints that render analyzed unsafe paths unrealizable. For a reachability specification, since a single witness is enough, the algorithm configuration enables returning the first parameter valuation found that makes a path ending in a target location realizable. Moreover, to get the optimal parameter valuation according to a cost function, the configuration also enables returning the best parameter valuation that satisfies the reachability specification after

the algorithm evaluates all the paths and accumulates the parameter constraints similar to the safety specification case. Repeated reachability specification is similar to a reachability specification, i.e., requiring a single witness. Depending on the configuration, the algorithm returns the first or the best parameter valuation for a repeated reachability specification. In any case, the returned parameter valuation makes a target location reachable infinitely many times. In the best valuation case, the algorithm evaluates all the paths. In particular, a Mixed Integer Linear Programming (MILP) encoding for simple paths and a Non-Linear Programming (NLP) encoding for cyclic paths with no nested cycles are presented to synthesize parameter and delay values. The realizability of the paths is checked via MILP and NLP formulations and, where appropriate, the search is pruned to improve the computation time. When the delay values are not controllable, as in a safety problem, parameter constraints are generated from the MILP and NLP formulations by eliminating quantified delay variables. A parameter valuation that satisfies the generated constraint ensures the considered path satisfies the given specification on the resulting timed automaton, i.e., it is unrealizable for a safety specification, realizable for a reachability specification and realizable infinitely many times for a repeated reachability specification. At the end of the depth-first traversal, the accumulated constraints are *and*ed for safety and *or*ed for reachability specifications. Then, the optimal parameter valuation satisfying this overall expression is calculated.

Any non-trivial parameter synthesis problem on a PTA is undecidable [4]. Here, a decidable variant of this problem is focused on, where the parameters are constrained to integer intervals [5]. The proposed algorithms are evaluated using the PTAs from the benchmark library [6].

1.2 Literature Review

Timed automata models are used in real-time systems [7, 8], multiprocessors [9], pacemakers [10, 11], rail-road crossing systems [12], function block diagrams for programmable logic controllers based on IEC 61131-3 [13], adaptive cruise control systems [14] and medical device systems for the closed-loop control of drug infusion [15].

A timed automaton can be checked whether or not it satisfies a specification by using a model checker tool and can be adjusted accordingly until the desired result is obtained. UPPAAL [16], Imitator [17] and HyTech [18] are some of the model-checking tools also applied to industrial use cases [16, 18, 19].

The parameter synthesis problem for timed automata has been studied in various works since its introduction in [20], where the first undecidability results for the synthesis problems and a semi-algorithm are presented.

In [21], an algorithm is proposed that takes a reference parameter valuation satisfying a safety property and generates a parameter constraint with a more extensive set of possible parameter valuations such that any of them satisfies the initial safety property. In [22], the authors define robustness as the amount by which the clock constraints and the clock values can drift without violating the given safety property and develop parametric approaches to find the robustness. Along with controller synthesis, parameter synthesis for a safety specification is studied in [23], as the authors extend a symbolic parameter synthesis method to encode symbolic constraints over the TA inputs. In [24], the authors solve a parameter synthesis problem for a safety specification to achieve non-interference. In [25], a parametric extension to UPPAAL is implemented to synthesize parameters for different specifications, including safety. [14] synthesizes parameters for an adaptive cruise control system while guaranteeing safety.

[26] develops methods to relax the constraints of a TA model during the design phase for reachability specifications and identifies constraints enforcing the satisfaction of the safety specifications. The parameter synthesis problem is also studied for reachability [5], [25] and unavoidability [27] specifications. In [28], a new subclass of PTA is defined as parametric event-recording automata and verification results for reachability specifications are obtained. The authors of [29] propose solutions to the reachability problem by parameter synthesis while minimizing the time-to-reach or some particular parameter values and show the results on a benchmark set. [30] synthesizes integer parameters for both reachability and unavoidability specifications. In [25], the authors define a subset of parametric timed automata called lower bound/upper bound (L/U) automata and study its reachability problems. [31] extends the work on

L/U PTA to repeated reachability specifications. [32] investigates a symbolic zone-based method to solve the reachability specification where parameters are restricted to bounded integers to make the problem decidable.

1.3 Contributions of the Thesis

The main contribution of this thesis is the complete solution for the parameter synthesis problem of a PTA from safety, reachability and repeated reachability perspectives under some structural limitations, i.e., no nested cycle. In addition, a novel approach based on NLP to assess the realizability of cyclic paths is introduced. Furthermore, the quantifier elimination method is presented to find parameter constraints from MILP/NLP constraints. The parameters satisfying these constraints guarantee the unrealizability, realizability and repeated realizability of the considered path for safety, reachability and repeated reachability specifications, respectively. In consequence, the proposed methods advance the parameter synthesis techniques.

The proposed methods are implemented in a tool called PaSTA (Parameter Synthesizer for Timed Automata, available at https://github.com/burkaysucu/PaSTA). The results on several benchmarks are reported.

1.4 The Outline of the Thesis

This thesis is organized as follows. Chapter 2 gives preliminary notation about parametric timed automata. Chapter 3 explains the core of our proposed algorithms to analyze a single path in detail, how to make it realizable or unrealizable and proves their correctness. Chapter 4 formally defines the parameter synthesis problem for a PTA under a safety specification, generalizes the methods discussed in Chapter 3 for a single path to the safety of a whole PTA. Chapter 5 formally defines the parameter synthesis problem for a PTA under a reachability or repeated reachability specification, generalizes the methods discussed in Chapter 3 for a single path to the reachability or repeated reachability of a whole PTA . Chapter 6 presents case studies and reports the benchmark results. Finally, Chapter 7 draws conclusions.

CHAPTER 2

PRELIMINARIES

2.1 (Parametric) Timed Automata

A timed automaton TA [1] is a finite state automaton with a finite set of real-valued clocks C. Each clock measures the time that has passed since its last reset. A clock constraint is expressed with the grammar $\phi := x \sim b \mid \phi \wedge \phi$ where x is a clock in C, $b \in \mathbb{N}$ and $\sim$ is an operator from $\{<, \leq, >, \geq\}$. $\Phi(C)$ denotes the set of clock constraints over C. A clock valuation $\nu : C \rightarrow \mathbb{R}_{\geq 0}$ is a function that assigns a non-negative real value to each clock. The satisfy relation $\nu \models \phi$ denotes that when each clock in ϕ is replaced with its valuation in ν, ϕ evaluates to true.

A delay operation for a clock valuation is shown by $\nu + d$ where $d \in \mathbb{R}_{\geq 0}$ and it means delaying each clock by an amount of d units, i.e., $(\nu + d)(x) = \nu(x) + d$. A reset operation is shown by $\nu[\lambda]$ where $\lambda \subseteq C$ and it means resetting each clock in λ to zero, i.e., $\nu[\lambda](x) = 0$ for $x \in \lambda$ and $\nu[\lambda](x) = \nu(x)$ for $x \in C \backslash \lambda$. $\mathbf{0}$ is a special valuation where all clock values are zero.

Definition 2.1.1 (Timed Automata) *A timed automaton $\mathcal{A} = (L, l_0, C, \Delta, Inv)$ is a five-tuple where L is a finite set of locations, $l_0 \in L$ is the initial location, C is a finite set of clocks, $\Delta \subseteq L \times 2^C \times \Phi(C) \times L$ is a finite transition relation and $Inv : L \rightarrow \Phi(C)$ is an invariant function.*

A transition $e = (l_s, \lambda, \phi, l_t) \in \Delta$ consists of a source location l_s, a subset of clocks λ to be reset, a clock constraint ϕ (*guard*) and a target location l_t.

A clock constraint is called parametric if it includes a parameter value instead of a

numeric constant. A TA is called parametric (PTA) if it contains at least one parametric clock constraint. A PTA extends a TA by a set of parameters P and lower and upper bound relations $I_{lower} : P \rightarrow \mathbb{N}$ and $I_{upper} : P \rightarrow \mathbb{N}$. A parameter valuation $\gamma : P \rightarrow \mathbb{N}$ is a function that assigns a natural number value to each parameter. A valid parameter valuation γ satisfies the boundary conditions for each parameter, i.e., $I_{lower}(p) \leq \gamma_p$ and $\gamma_p \leq I_{upper}(p)\ \forall p \in P$. The interval notation $I(p) = [I_{lower}(p), I_{upper}(p)]$ and the boundary notation are used interchangeably throughout the thesis.

A parametric constraint ϕ is shown as $\phi(\gamma)$ when parameters of ϕ are replaced with values from the valuation in γ. A PTA with a parameter valuation γ yields a regular TA, denoted by $\mathcal{PA}(\gamma)$, when all parameters in all clock constraints are replaced with values in γ.

Definition 2.1.2 (Timed Transition System) *A timed transition system (TTS) is a four-tuple $\mathcal{T} = (S, s_0, \Gamma, \rightarrow)$ where S is a set of states, $s_0 \in S$ is the initial state, Γ is a set of symbols and $\rightarrow \subseteq S \times \Gamma \times S$ is a transition relation. Elements of the transition relation are also shown as $s \rightarrow s'$.*

Definition 2.1.3 (TTS Semantics for TA) *A TTS $\mathcal{T}(\mathcal{A}) = (S, s_0, \Gamma, \rightarrow)$ describes the semantics of a TA $\mathcal{A} = (L, l_0, C, \Delta, Inv)$ as follows:*

- *Set of states S is the pairs of locations and clock valuations, i.e., $S = \{(l, \nu) | l \in L, \nu \in \mathbb{R}^{|C|}_{\geq 0}, \nu \models Inv(l)\}$*
- *Initial state s_0 is the pair of initial location and clock valuation of zero, i.e., $s_0 = (l_0, \boldsymbol{0})$*
- *Set of symbols is the union of non-negative real values for delay transitions and act for discrete transitions, i.e., $\Gamma = \mathbb{R}_{\geq 0} \cup \{act\}$*
- *There are two types of transitions:*
  - *$(l, \nu) \xrightarrow{d} (l, \nu + d)$ is a delay transition if $(\nu + d) \models Inv(l)$,*
  - *$(l, \nu) \xrightarrow{act} (l', \nu')$ is a discrete transition if there exists $(l, \lambda, \phi, l') \in \Delta$ where $\nu \models \phi$, $\nu' = \nu[\lambda]$ and $\nu' \models Inv(l')$.*

A run ρ of a TA is shown as an alternating sequence of delay and discrete transitions of the corresponding TTS:

$$\rho = (l_0, \nu_0) \xrightarrow{d_0} (l_1, \nu_1) \xrightarrow{d_1} (l_2, \nu_2)... \tag{2.1}$$

where ν_0 is $\mathbf{0}$ and $(l_i, \nu_i) \xrightarrow{d_i} (l_{i+1}, \nu_{i+1})$ is used to denote a delay transition followed by a discrete transition. The set of all possible runs of $\mathcal{A}$ is shown as $[\![\mathcal{A}]\!]$. A path $\pi = l_0, e_0, l_1, e_1, l_2, ..., l_n$ is said to be *realizable* if there exists a delay sequence $d_0, ..., d_{n-1}$ and run $\rho = (l_0, \mathbf{0}) \xrightarrow{d_0} (l_1, v_1) \xrightarrow{d_1} ...(l_n, v_n) \in [\![\mathcal{A}]\!]$ such that $e_i = (l_i, \lambda_i, \phi_i, l_{i+1})$ for each $i = 0, ..., n-1$.

CHAPTER 3

PATH ANALYSIS

In this chapter, methods to analyze simple and cyclic paths are presented in the following sections.

3.1 Simple Path Analysis

In this section, methods to make a simple path realizable and unrealizable are introduced. First, the necessary notation to represent the clock valuations along a simple path is presented.

The value of the clock x at a point on the path $\pi = l_0, e_0, l_1, e_1, l_2, ...$ is expressed as the sum of time spent on the previous locations since its last reset. In particular, the value of x when entering l_i is $\Gamma_A(x, \pi, i)$ (3.1) and when leaving l_i is $\Gamma_D(x, \pi, i)$ (3.2):

$$\Gamma_A(x, \pi, i) = \sum_{j=k}^{i-1} d_j, \tag{3.1}$$

$$\Gamma_D(x, \pi, i) = \sum_{j=k}^{i} d_j \quad \text{where} \tag{3.2}$$

$$k = max(\{r + 1 | x \in \lambda_r, r < i\} \cup \{0\}).$$

The difference between the value of x at the arrival of l_i, i.e., $\Gamma_A(x, \pi, i)$, and the value of x when leaving from l_i, i.e., $\Gamma_D(x, \pi, i)$, is d_i, which represents the time spent on location l_i.

Once the calculation method of clock values on a simple path is established, the method to encode the constraints on such paths is presented next.

3.1.1 Make a Simple Path Realizable

The fundamental subproblem in solving the parameter synthesis problem for a PTA is checking the existence of and, if it exists, finding a delay sequence and parameter valuation that render a single path realizable.

In this subsection, the constraint synthesis problem solved to generate constraints such that a given path becomes realizable on the resulting PTA for any parameter valuation satisfying the generated constraints is presented. This synthesis problem is formalized in Problem 3.1.1.

Problem 3.1.1 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ and a path $\pi = l_0, e_0, l_1, e_1, l_2, ..., l_n$, find a set of constraints ω such that there exists a parameter valuation γ that is compatible with I, satisfies ω and π becomes realizable on $\mathcal{PA}(\gamma)$.*

As introduced in [5], the timing constraints on a simple path are encoded as a MILP problem to solve Problem 3.1.1. Proposition 3.1.1 formulizes the translation between the timing constraints and the MILP problem. Any solution to it gives a parameter valuation along with a delay sequence, and together, they render the path realizable.

Proposition 3.1.1 *Let $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ be a PTA, P be a set of parameters, $I : P \to \mathbb{N} \times \mathbb{N}$ be a parameter intervals relation and $\pi = l_0, e_0, l_1, e_1, l_2, ..., l_n$ be a path where $e_i = (l_i, \lambda_i, \phi_i, l_{i+1})$. A parameter valuation γ such that π is realizable on $\mathcal{PA}(\gamma)$ exists if and only if the MILP problem given in* (3.3) *with decision variables $(d_0, d_1, d_2, ..., d_{n-1})$ and $\gamma_p, p \in P$ is feasible.*

Find γ_p for each $p \in P$ and $d_0, d_1, d_2, ..., d_{n-1}$ such that:

$$\gamma_p \in I(p) \qquad \forall p \in P, \tag{3.3a}$$

$$d_i \in \mathbb{R}_{\geq 0} \qquad \forall i \in [0, n), \tag{3.3b}$$

$$\Gamma_A(x, \pi, i) \sim g(b) \quad \forall i \in [0, n], (x \sim b) \in Inv(l_i), \tag{3.3c}$$

$$\Gamma_D(x, \pi, i) \sim g(b) \quad \forall i \in [0, n), (x \sim b) \in \phi_i, \tag{3.3d}$$

$$\Gamma_D(x, \pi, i) \sim g(b) \quad \forall i \in [0, n), (x \sim b) \in Inv(l_i) \tag{3.3e}$$

where $g(b) = b$ *if* $b \in \mathbb{N}$ *and* $g(b) = \gamma_p$ *if* b *is the parameter* $p \in P$*. Furthermore, if the MILP is feasible, then the path* π *of TA* $\mathcal{PA}(\gamma)$ *is realizable through the delay sequence* $d_0, d_1, d_2, ..., d_{n-1}$*, where* γ *and the delay sequence is a solution of* (3.3)*.*

Throughout the thesis, $MILP(\mathcal{PA}, \pi)$ is used to denote the set of the constraints given in (3.3) for path π of automaton $\mathcal{PA}$.

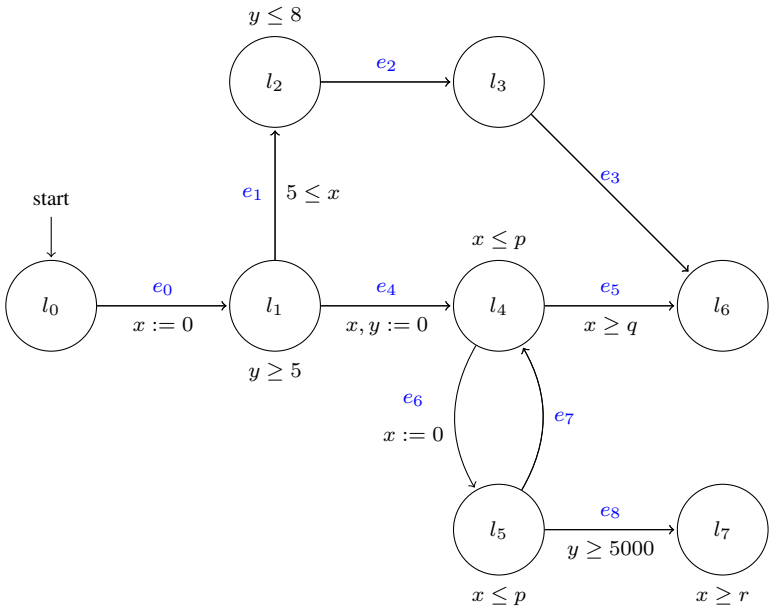

Figure 3.1: An example PTA with parameters p, q, r and different paths to the unsafe locations l_6, l_7. Transition labels are shown in blue.

Example 3.1.1 *Consider the path* $\pi_1 = l_0, e_0, l_1, e_4, l_4, e_5, l_6$ *of the PTA shown in Figure 3.1. The parameter bounds are defined as* $I(p) = [0, 10]$*,* $I(q) = [5, 15]$ *and* $I(r) = [5, 15]$*. The corresponding MILP's decision variables are* γ_p, γ_q *and* $d_{l_0}, d_{l_1}, d_{l_4}$ *for the parameters and the delay valuations, respectively. The constraints of* $MILP(\mathcal{PA}, \pi_1)$ *are*

$$\gamma_p \in I(p) \wedge \gamma_q \in I(q), \tag{3.4a}$$

$$d_{l_i} \in \mathbb{R}_{\geq 0} \quad \forall i \in \{0, 1, 4\}, \tag{3.4b}$$

$$d_{l_0} \geq 5, \quad 0 \leq \gamma_p, \tag{3.4c}$$

$$d_{l_4} \geq \gamma_q, \tag{3.4d}$$

$$d_{l_0} + d_{l_1} \geq 5, \quad d_{l_4} \leq \gamma_p. \tag{3.4e}$$

MILP (3.4) *is feasible. A sample solution is* $\gamma_p = 10$, $\gamma_q = 5$, $d_{l_0} = 5$, $d_{l_1} = 0$ *and* $d_{l_4} = 5$.

Now consider the path $\pi_2 = l_0, e_0, l_1, e_1, l_2$ *from the same automaton. The decision variables of the MILP constructed for* π_2 *in a similar fashion are only the delay valuations* d_{l_0}, d_{l_1} *since this path has no parametric constraints.* $MILP(\mathcal{PA}, \pi_2)$ *is* $d_{l_0} \geq 5 \wedge d_{l_0} + d_{l_1} \leq 8 \wedge 5 \leq d_{l_1} \wedge d_{l_0} + d_{l_1} \geq 5$. *The inequalities* $d_{l_0} \geq 5$ *and* $5 \leq d_{l_1}$ *implies* $d_{l_0} + d_{l_1} \geq 10$, *which contradicts with the MILP constraint* $d_{l_0} + d_{l_1} \leq 8$. *Thus, the path* π_2 *is not realizable since* $MILP(\mathcal{PA}, \pi_2)$ *is infeasible.*

The decision variables of a MILP problem are the set of delay and parameter variables. When all those decision variables are controllable, e.g., making a path realizable through parameter synthesis, $MILP(\mathcal{PA}, \pi)$ becomes the solution constraint set ω for Problem 3.1.1.

3.1.2 Make a Simple Path Unrealizable

In some cases, such as a safety specification, the need to force a realizable path unrealizable arises. The following definition is needed to formalize the approaches to solve this problem.

Definition 3.1.1 (Parameter Constraint) *A parameter constraint is expressed with the grammar* $\omega := \Sigma(a * p) \sim b \mid \omega \wedge \omega \mid \omega \vee \omega$ *where* a *and* b *are constant natural numbers in* $\mathbb{N}$, p *is a parameter in* P *and* $\sim$ *is an operator from* $\{<, \leq, >, \geq\}$. *The MILP of PTA paths may lead to such parameter constraints after some simplification, such as Fourier-Motzkin elimination.*

Example 3.1.2 *Consider the path* $\pi_1 = l_0, e_0, l_1, e_4, l_4, e_5, l_6$ *from Example 3.1.1 and its MILP in* (3.4). *The terms* $d_{l_4} \geq q$ *and* $d_{l_4} \leq p$ *in* $MILP(\mathcal{PA}, \pi)$ *can be deduced to the parameter constraint* $q - p \leq 0$ *by Fourier-Motzkin elimination.*

In this subsection, the constraint synthesis problem solved to generate parameter constraints such that a given path becomes unrealizable on the resulting PTA for any

parameter valuation satisfying the generated constraints is presented. This synthesis problem is formalized in Problem 3.1.2.

Problem 3.1.2 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ for each parameter, a set of parameter constraints Ω and a realizable path $\pi = l_0, e_0, l_1, e_1, l_2, ..., l_n$, find a set of parameter constraints ω such that $MILP(\mathcal{PA}, \pi)$ (3.3) together with $\Omega \cup \omega$ becomes infeasible; however, there exists a parameter valuation γ that is compatible with I and satisfies $\Omega \cup \omega$.*

Recall that the decision variables of $MILP(\mathcal{PA}, \pi)$ (3.3) are the set of delay variables $(d_0, d_1, d_2, ..., d_{n-1})$ and the set of parameter variables $\gamma_p, p \in P$. The goal of Problem 3.1.2 is to find a set of parameter constraints ω such that there exists a parameter valuation γ that is compatible with I and satisfies both Ω (previously collected constraints) and ω; however, no delay sequence $(d_0, d_1, d_2, ..., d_{n-1})$ and parameter valuation γ satisfies $MILP(\mathcal{PA}, \pi)$ (3.3) and $\Omega \cup \omega$. The non-existence of the delay sequence for the MILP and Ω is formalized with the formula given in Equation (3.5), where Ω^C is $\bigwedge_{\phi \in \Omega} \phi$.

$$\forall (d_0, \ldots, d_{n-1}) : \left\{ MILP(\mathcal{PA}, \pi) \wedge \Omega^C \right\} = False. \tag{3.5}$$

A set of parameter constraints ω is obtained by eliminating the quantified delay variables from (3.5). Note that the constraints obtained via the quantifier elimination are over the remaining decision variables, which are the parameters. If the delay variables can not be fully eliminated, the resulting constraint depends on the delay variables. In that case, the path π can not be forced to be unrealizable through parameter synthesis. On the other hand, if the quantified delay variables are eliminated from (3.5), a parameter constraint ω is obtained. In addition to the elimination, it is checked if any parameter valuation γ exists that satisfies interval constraints I and the accumulated set of constraints $\Omega \cup \omega$:

$$\{\gamma \in I \mid \gamma \models \bigwedge_{\phi \in \Omega \cup \omega} \phi\} \neq \emptyset. \tag{3.6}$$

Observe that the quantifier elimination combined with the feasibility check provides a solution to Problem 3.1.2. Elimination of delay variables from the quantified expression is needed when the decision variables of MILP related to delay variables are not controllable, e.g., making a path unrealizable through parameter synthesis.

Example 3.1.3 *As explained in Example 3.1.1, the MILP encoding for the path* $\pi_1 = l_0, e_0, l_1, e_4, l_4, e_5, l_6$ *of the PTA shown in Figure 3.1 is feasible and ends in the unsafe location* l_6*. Problem 3.1.2 is solved for this path in line 9 of the main algorithm. As this is the first encountered realizable unsafe path,* Ω *is empty. The quantified formula generated for this path is given in* (3.7).

$$\begin{aligned} &\forall d_{l_0}, d_{l_1}, d_{l_4} : \\ &\quad \Big\{ \gamma_p \in I(p) \wedge \gamma_q \in I(q) \wedge d_{l_0} \geq 5 \wedge 0 \leq \gamma_p \wedge \\ &\quad\quad d_{l_4} \geq \gamma_q \wedge d_{l_0} + d_{l_1} \geq 5 \wedge d_{l_4} \leq \gamma_p \wedge \\ &\quad\quad (d_{l_i} \in \mathbb{R}_{\geq 0} \forall i \in \{0, 1, 4\}) \Big\} = False. \end{aligned} \tag{3.7}$$

As given in Example 3.1.2, the terms $d_{l_4} \geq q$ *and* $d_{l_4} \leq p$ *can be deduced to the parameter constraint* $q - p \leq 0$*. This constraint must be true for* $MILP(\mathcal{PA}, \pi_1)$ *to be feasible. The negation of the parameter constraint* $q - p \leq 0$ *guarantees that* $MILP(\mathcal{PA}, \pi_1)$ *is infeasible; thus,* (3.7) *is false for any delay sequence and a parameter valuation* γ *satisfying this constraint.*

3.2 Cyclic Path Analysis Methods

In this section, three different methods to analyze cyclic paths are presented. The following formalization of a cyclic path is needed to explain those methods.

A cyclic path is in the following form

$$\pi = l_0 e_0 \dots e_{p-1} (l_p e_p \dots e_{p+q-1})^c l_{p+q} e_{p+q} \dots l_{p+q+r} \tag{3.8}$$

where $l_p = l_{p+q}$ and $c \geq 1$. Here, c is simply a parameter denoting the number of times the cycle is taken. Thus, π is a path of the considered PTA for any $c \geq 0$.

3.2.1 Cyclic Path Analysis by Unrolling

The first method to analyze a cyclic path is determining the cycle count and unrolling the cycle that many times. [5] uses this approach to solve the reachability problem where a limit on the cycle count $lim(c)$ is set. The algorithm produces the simple

paths $\{l_0 e_0 \dots e_{p-1}(l_p e_p \dots e_{p+q-1})^i l_{p+q} e_{p+q} \dots l_{p+q+r} | 0 \leq i \leq lim(c)\}$ for a cyclic path and processes them using the MILP method explained in 3.1.

As discussed in [5], an algorithm using this unrolling method for single paths is complete under some assumptions for reachability, i.e., when an automaton is acyclic (trivial) or there is a deadline that can be used as a proper cycle limit. This also applies to other specifications, i.e., when those assumptions are met, the unrolling method is complete for any specification. In the same paper, the unrolling method is also proven to be complete with an additional processing step, splitting the path into two parts from the location where the cycle limit is hit. If the MILP for the first part is found to be feasible, keep the constraints for that part and start over from the first location of the second part. This way, the parameter search space gets reduced significantly, computation gets faster and completeness is achieved.

When using unrolling, the number of decision variables in a MILP problem for a cyclic path with n locations in the cycle gets increased by n for each cycle. With a cycle limit of lim, it reaches $lim * n$ at the last unrolling. The lim value must be tuned per path to get a performant solution.

3.2.2 Cyclic Path Analysis Through External Tools

The second method to analyze a cyclic path is asking an external tool, such as UPPAAL or Imitator, for a constraint set that contains some parameter valuation satisfying the specification. This method can be applied to simple paths as well. However, it is limited by the capabilities of the external tool used. For example, as shown in Chapter 6, Imitator does not terminate for a safety specification problem of a PTA with a simple cycle.

3.2.3 Cyclic Path Analysis by Parameterizing The Cycle Count

Example 3.2.1 *In order to motivate the need for a special encoding for cyclic paths, consider the cyclic path* $\pi = l_0, e_0, l_1, e_4, l_4, e_6, l_5, e_8, l_7$ *from the PTA shown in Figure 3.1. Even though* $MILP(\mathcal{PA}, \pi)$ *is infeasible, the path*

$p_4 = l_0, e_0, l_1, e_4, (l_4, e_6, l_5, e_7)^{500}, l_4, e_6, l_5, e_8, l_7$ *is realizable.*

Similar to the realizability check performed for the non-cyclic paths as described in Proposition 3.1.1, the realizability check for the cyclic paths is encoded as an optimization problem over a set of delay variables and the PTA parameters. However, in this case, in addition to the delay variables and PTA parameters, a cycle count variable c is also defined. Note that, according to the encoding given in (3.3), the number of delay variables depends on the value of c. Consequently, a straightforward encoding is not possible. Here, a new encoding is proposed via unrolling the cycle twice (before and after taking the cycle) and using average delay representation for taking the cycle more than twice. The average delay durations are multiplied with the cycle count variable to encode different cycle count values, resulting in a non-linear encoding. Next, details of this approach are presented.

A cyclic path is composed of three parts: pre-cycle ($\pi_{pre} = l_0 e_0 \dots e_{p-1}$), cycle ($\pi_{cyc} = l_p e_p \dots e_{p+q-1}$) and post-cycle ($\pi_{post} = l_{p+q} e_{p+q} \dots l_{p+q+r}$), i.e., π (3.8) is $\pi_{pre} \pi_{cyc}{}^{c} \pi_{post}$. A cyclic path is illustrated in Figure 3.2.

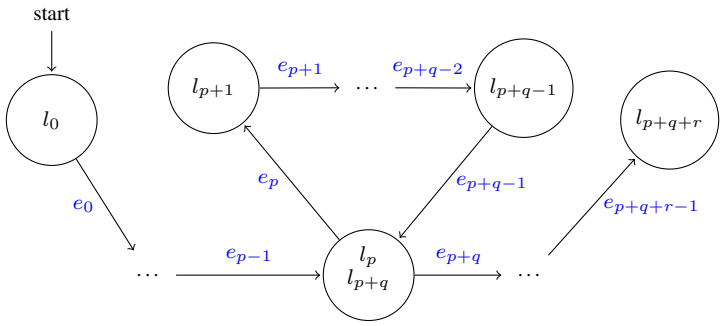

Figure 3.2: The generic form of a path with a simple cycle having p pre-cycle, q cycle and r post-cycle locations.

First, the cycle is unrolled twice to obtain the path given in (3.9).

$$\pi_u = \pi_{pre} \pi_{cyc,f} \pi_{cyc} \pi_{cyc,l} \pi_{post}. \tag{3.9}$$

Following the indices introduced in (3.8), the path π_u consists of a total of $p + q +$

$q+q+r+1$ locations. Within this path, the delay variables d_i are assigned based on their respective indices. For indices $i < p+q$, the delay variable d_i corresponds to a location from the prefix $\pi_{pre}\pi_{cyc,f}$ of the path π_u. On the other hand, delay variables with indices $i \geq p+2q$ are associated with the suffix $\pi_{cyc,l}\pi_{post}$ of the path. The remaining delay variables, specifically d_i with indices between $p+q$ and $p+2q-1$, represent the cycle π_{cyc} part. These delay variables are considered to represent the average time spent at the corresponding locations.

A new clock value function is defined that incorporates the average time notion for π_{cyc}, resembling Γ (3.1). Subsequently, the path constraints based on (3.3) are introduced, ensuring that a path π (3.8) is realizable for some $c \geq 2$ if and only if the introduced constraints are feasible, which is proven afterwards.

Similar to (3.1) and (3.2), the value of a clock is represented as the sum of delay values since its last reset. The major difference here is that if the clock is reset before π_{cyc} and read after π_{cyc}, then the total average duration ($\sum_{i=p+q}^{p+2q-1} d_i$) multiplied by the variable representing the number of times the cycle π_{cyc} is taken, i.e., c, is added to the total duration. For all the other cases, the sum of durations since the last reset of the clock along the path π_u is computed as in (3.1).

$$\Gamma_A^{cyc}(x,\pi_u,i) = \begin{cases} \sum_{j=k}^{p+q-1} d_j + c\sum_{j=p+q}^{p+2q-1} d_j + \sum_{j=p+2q}^{i-1} d_j & i \geq p+2q, k < p \\ \sum_{j=k}^{i-1} d_j & otherwise \end{cases} \tag{3.10}$$

where

$$k = max(\{r+1 | x \in \lambda_r, r < i\} \cup \{0\}).$$

Similar to (3.2), the value of the clock when leaving a location l_i is its value at the arrival plus the time passed in l_i:

$$\Gamma_D^{cyc}(x,\pi_u,i) = d_i + \Gamma_A^{cyc}(x,\pi_u,i). \tag{3.11}$$

Prior to presenting the NLP encoding used to analyze the realizability of a cyclic path, an assumption that must be satisfied is introduced.

Assumption 3.2.1 *Let $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ be a PTA and π_u be an unrolled cyclic path as shown in* (3.9)*. Then either a) or b) holds for each clock $x \in C$:*

a) x is not reset on π_{cyc}, i.e., $x \notin C_R = \{x' | x' \in \lambda_i, p \leq i \leq p+q-1\}$

b) x is reset within the cycle: in this case, it is reset while entering the cycle or before it is read, i.e., if $x \in C_R$ and $x \sim c \in Inv(l_i) \cup \phi_i$ for some $i \in [p, \ldots, p+q-1]$, then

$$x \in \lambda_j \text{ for some } p-1 \leq j < i. \tag{3.12}$$

Assumption 3.2.1 guarantees that a constraint in a cycle over a clock $x \in C_R$ only depends on the delays on the current cycle locations. Note that in many cases, a violation of Assumption 3.2.1 can be avoided by adjusting the start location of the cycle as in Example 3.2.2.

Proposition 3.2.1 *Let $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ be a PTA, P be its set of parameters, $I : P \to \mathbb{N} \times \mathbb{N}$ be a parameter intervals relation and π_u be an unrolled cyclic path* (3.9)*. A parameter valuation γ and $c \in \mathbb{N}$ such that π_u with cycle count c is realizable on $\mathcal{PA}(\gamma)$ exists if and only if the NLP problem given in* (3.13) *with the decision variables $d_0, d_1, d_2, ..., d_{p+3q+r-1}$ (delay variables), c (cycle count) and $\gamma_p, p \in P$ (PTA parameters) is feasible.*

$$\gamma_p \in I(p) \qquad \forall p \in P, \tag{3.13a}$$

$$d_i \in \mathbb{R}_{\geq 0} \qquad \forall i \in [0, K), \tag{3.13b}$$

$$c \in \mathbb{N}, \tag{3.13c}$$

$$\Gamma_A^{cyc}(x, \pi, i) \sim g(b) \quad \forall i \in [0, K], (x \sim b) \in Inv(l_i), \tag{3.13d}$$

$$\Gamma_D^{cyc}(x, \pi, i) \sim g(b) \quad \forall i \in [0, K), (x \sim b) \in \phi_i, \tag{3.13e}$$

$$\Gamma_D^{cyc}(x, \pi, i) \sim g(b) \quad \forall i \in [0, K), (x \sim b) \in Inv(l_i) \tag{3.13f}$$

where $K = p + 3q + r$, $g(b) = b$ if $b \in \mathbb{N}$ and $g(b) = \gamma_p$ if b is the parameter $p \in P$.

Throughout the thesis, $NLP(\mathcal{PA}, \pi)$ is used to denote the set of the constraints given in (3.13) for path π of automaton $\mathcal{PA}$.

Proof 3.2.1 *(IF) Assume that the NLP* (3.13) *is feasible and let* $d_0^\star, \ldots, d_K^\star$, $c^\star$ *and* $\gamma_p^\star, p \in P$ *be a solution. The corresponding actual unrolled path* π^a *of length* $N = p+2q+c^\star q+r$ *with respect to* π_u (3.9) *is defined by mapping the indices representing taking the cycle* $c^\star$ *times to the corresponding locations from* π_u*. First, the index mapping function* (3.14) *and then the path* π^a (3.15) *is defined.*

$$h(i) = p + q + ((i - p) \mod q), \tag{3.14}$$

$$\pi^a = l_0^a e_0^a \ldots e_N^a l_{N+1}^a, \text{ where} \tag{3.15}$$

$$l_i^a = l_i \text{ when } i < p + q, \tag{3.16}$$

$$l_i^a = l_j, j = h(i) \text{ when } p + q \leq i < p + q + c^\star q, \tag{3.17}$$

$$l_i^a = l_{i-(c^\star-1)q} \text{ when } i \geq p + q + c^\star q. \tag{3.18}$$

The mapping for the edge indices follows from the location indices and is omitted for brevity. In path π^a*, the locations from* (3.16) *correspond to* $\pi_{pre}\pi_{cyc,f}$*, the locations from* (3.17) *correspond to taking the cycle for* $c^\star$ *times* $(\pi_{cyc}^{c\star})$*, and the locations from* (3.18) *correspond to* $\pi_{cyc,l}\pi_{post}$*. Next, a delay sequence for path* π^a *from the solution* $d_0^\star, \ldots, d_K^\star$ *is defined:*

$$\mathbf{d}^a = d_0^a, \ldots, d_N^a, \text{ where} \tag{3.19}$$

$$d_i^a = d_i^\star \text{ when } i < p + q, \tag{3.20}$$

$$d_i^a = d_j^\star, j = h(i) \text{ when } p + q \leq i < p + q + c^\star q, \tag{3.21}$$

$$d_i^a = d_{i-(c^\star-1)q}^\star \text{ when } i \geq p + q + c^\star q. \tag{3.22}$$

Also, the clock value sequence $\nu = \nu_0, \nu_1, \ldots, \nu_N$ *is defined with respect to the delay sequence* (3.19) *and path* (3.15) *as in Definition 2.1.3 and* (2.1)*:*

$$\nu_{i+1} = (\nu_i + d_i^a)[\lambda_i] \quad \text{for } i = 1, \ldots, N \text{ and } \nu_0 = \mathbf{0}. \tag{3.23}$$

Observe that the clock value sequence is consistent with $\Gamma_A^{cyc}(x, \pi_u, i)$ *for the prefix* $\pi_{pre}\pi_{cyc,f}$ *and suffix* $\pi_{cyc,l}\pi_{post}$ *of the path* π^a*. In particular*

$$\nu_i(x) = \begin{cases} \Gamma_A^{cyc}(x, \pi_u, i) & \text{for } i < p + q \\ \\ \Gamma_A^{cyc}(x, \pi_u, i - (c^\star - 1)q) & \text{for } i \geq p + q + c^\star q. \end{cases} \tag{3.24}$$

Consequently, it follows from (3.13)*,* (3.19) *and* (3.24) *that the invariants and guards of the prefix* $\pi_{pre}\pi_{cyc,f}$ *and the suffix* $\pi_{cyc,l}\pi_{post}$ *are satisfied by* ν*. Next, the constraints from* π_{cyc} *are focused on. They are analyzed in two groups: 1) the constraints over the clocks* x *that are reset within the cycle, i.e.,* $x \in C_R$ *and 2) the constraints over the clocks* x *that are not reset within the cycle, i.e.,* $x \in C \setminus C_R$*.*

$x \in C_R$ *: Due to Assumption 3.2.1, the clock value mapping function* Γ_A^{cyc} (3.10)*, the delay sequence* $\mathbf{d}^a$ *for path* π^a (3.19)*, given a cycle index* $j \in \{0, \ldots, q-1\}$ *the value of a clock* $x \in C_R$ *when arriving at the* $j-th$ *location of the cycle is*

$$\nu_{p+q+uq+j}(x) = \Gamma_A^{cyc}(x, \pi_u, p+q+j) \quad u = 0, ..., c^\star - 1. \tag{3.25}$$

Consequently, the NLP constraints (3.13)*, together with the relation from* (3.25)*, ensure that all the constraints over the clocks from* C_R *are satisfied by* ν_i *with* $i = p+q, \ldots, p+q+qc^\star-1$ *(i.e.,* $\pi_{cyc}^{c\star}$ *part of the actual path* (3.17)*).*

$x \in C \setminus C_R$*: In this case, the clock* x *is never reset within the cycle. Consequently, for any* i, j *with* $p \leq i < j < p+2q+qc^\star$*, it holds that*

$$\nu_i(x) \leq \nu_j(x). \tag{3.26}$$

Thus, if a lower bound constraint $x \sim b, \sim \in \{>, \geq\}$ *over such a clock* x *is satisfied at* $p+i$ *for some* $i = 0, \ldots, q-1$ *(i.e., within* $\pi_{cyc,f}$*), it is also satisfied at* $p+i+qu$ *for each* $u = 1, \ldots, c^\star$*. Similarly, if an upper bound constraint* $x \sim b, \sim \in \{<, \leq\}$ *over* $x \in C \setminus C_R$ *is satisfied at* $p+q+qc^\star+i$ *for some* $i = 0, \ldots, q-1$ *(i.e., within* $\pi_{cyc,l}$*), it is also satisfied at* $p+i+qu$ *for each* $u = 1, \ldots, c^\star$*. These observations, together with the NLP constraints* (3.13)*, the clock value mapping function* Γ_A^{cyc} (3.10)*, the relation of* ν *and* Γ_A^{cyc} (3.25) *ensure that all the constraints over the clocks from* $C \setminus C_R$ *are satisfied by* ν_i *with* $i = p+q, \ldots, p+q+qc^\star-1$*.*

These analyses lead to the conclusion that the path π^a *of* $\mathcal{PA}(\gamma_p^\star)$ *can be realized via the delay sequence* $\mathbf{d}^a$ (3.19)*.*

(ONLY IF) Assume that the NLP is infeasible. By employing proof by contradiction, also assume that there exists a parameter valuation $\bar{\gamma}_p, p \in P$ *satisfying the parameter boundaries, a cycle count* $\bar{c}$*, a delay sequence* $\bar{\mathbf{d}} = \bar{d}_0, \ldots, \bar{d}_{\bar{N}}$ *with* $\bar{N} = p+2q+\bar{c}q+r$ *such that the path* $\bar{\pi^a}$ (3.27) *obtained from* π_u (3.9) *by unrolling the*

cycle π_{cyc} for $\bar{c}$ times is realizable via the delay sequence $\bar{\mathbf{d}}$ in $\mathcal{PA}(\bar{\gamma})$.

$$\bar{\pi}^a = \bar{l}^a_0 \bar{e}^a_0 \dots \bar{e}^a_{\bar{N}} \bar{l}^a_{\bar{N}+1}, \text{ where} \tag{3.27}$$

$$\bar{l}^a_i = l_i \text{ when } i < p+q, \tag{3.28}$$

$$\bar{l}^a_i = l_j, j = h(i) \text{ when } p+q \leq i < p+q+\bar{c}q, \tag{3.29}$$

$$\bar{l}^a_i = l_{i-(\bar{c}-1)q} \text{ when } i \geq p+q+\bar{c}q. \tag{3.30}$$

The construction of $\bar{\pi}^a$ from π_u is identical to that of π^a (3.15)*. A delay sequence $\mathbf{d}'$ of length $K = p+3q+r$ is defined for π_u from $\bar{\mathbf{d}}$ as follows:*

$$\mathbf{d}' = d'_0, \dots, d'_K, \text{ where} \tag{3.31}$$

$$d'_i = \bar{d}_i \text{ when } i < p+q, \tag{3.32}$$

$$d'_i = \frac{\sum_{j=0}^{\bar{c}-1} \bar{d}_{i+jq}}{\bar{c}} \text{ when } p+q \leq i < p+2q, \tag{3.33}$$

$$d'_i = \bar{d}_{i+(\bar{c}-1)q} \text{ when } i \geq p+2q. \tag{3.34}$$

Let $\bar{\nu} = \bar{\nu}_0, \dots, \bar{\nu}_{\bar{N}}$ and $\nu' = \nu'_0, \dots, \nu'_K$ be the clock value sequences defined with respect to the path and delay value sequence pairs $\bar{\pi}^a$ (3.27)*-$\bar{d}$ and π_u* (3.9)*-$\mathbf{d}'$* (3.31)*, respectively. Observe that, due to Assumption 3.2.1, the path definitions and* (3.10)*, whenever a constraint exists over the clock $x \in C$ at ith position, it holds that:*

$$\nu'_i(x) = \begin{cases} \bar{\nu}_i(x) = \Gamma^{cyc}_A(x, \pi_u, i) & \text{when } i < p+q \\ \\ \bar{\nu}_{i+(\bar{c}-1)q}(x) = \Gamma^{cyc}_A(x, \pi_u, i) & \text{when } p+2q \leq i \leq K. \end{cases} \tag{3.35}$$

Thus, when the constraints from $\bar{\pi}^a$ at positions $i < p+q$ $(\pi_{pre} p_{cyc,f})$ and $i \geq p+q+q\bar{c}$ $(\pi_{cyc,l}\pi_{post})$ are satisfied by $\bar{\nu}$; then, the corresponding constraints are also satisfied by ν' for path π_u. Furthermore, the corresponding NLP constraints (3.13) *with indices $i < p+q$ and $p+2q \leq i \leq K$ are satisfied by $\bar{\gamma}_p$ and $\mathbf{d}'$* (3.31) *due to* (3.35)*.*

Next, the remaining NLP constraints are analyzed, i.e., with indices $p+q \leq i < p+2q$. Since d'_i is defined as the average of the corresponding durations of the unrolled path (3.33)*, the ν'_i values are between the maximum and the minimum of the corresponding values from $\bar{\nu}$, i.e., for each $i = p+q, \dots, p+2q-1$:*

$$\min \mathcal{N}_i \leq \nu'_i \leq \max \mathcal{N}_i \text{ where } \mathcal{N}_i = \{\bar{\nu}_{i+jq} \mid j = 0, \dots, \bar{c}-1\}. \tag{3.36}$$

Consequently, by considering the convexity of the path constraints (3.13)*, it follows that if each clock value from $\mathcal{N}_i$ satisfies a constraint at position i, then ν_i' also satisfies it. Therefore, if there exists a delay sequence $\bar{\mathbf{d}} = \bar{d}_0, \ldots, \bar{d}_{\bar{N}}$, parameter valuations $\bar{\gamma}_p, p \in P$ and a cycle count $\bar{c}$, then the NLP is feasible. However, this leads to a contradiction, which completes the proof.* □

Example 3.2.2 *Consider the cyclic path $\pi_v = l_0, e_0, l_1, e_1, l_2, e_2, l_3, e_3, l_1, e_4, l_4$ of the PTA shown in Figure 3.3. This path violates Assumption 3.2.1 as clock t is reset on e_2 after it is read on e_1 and clock u is reset on e_3 after it is read on e_2. However, the cyclic path obtained by using l_3 as the first location of the cycle satisfies Assumption 3.2.1:*

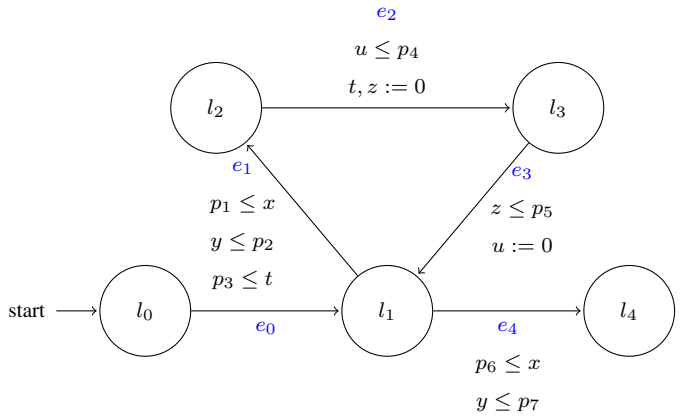

Figure 3.3: An example PTA with parameters $p_1 - p_7$ and clocks x, y, z, t, u.

$$\pi = \underbrace{l_0, e_0, l_1, e_1, l_2, e_2}_{\pi_{pre}}, \underbrace{l_3, e_3, l_1, e_1, l_2, e_2}_{\pi_{cyc}}, \underbrace{l_3, e_3, l_1, e_4, l_4}_{\pi_{post}}. \tag{3.37}$$

Path π does not violate the assumption, i.e., if a clock is reset within the cycle, any constraint in the cycle, which includes that clock, only depends on the time spent on the current cycle locations. For the path π, $p = 3$, $q = 3$ and $r = 2$. Thus, $K = 14$ for the corresponding unrolled path π^u (3.9)*. The decision variables of the NLP* (3.13) *are c, $\gamma_{p_i}, i = 1, \ldots, 7$ and $d_i, i = 0, \ldots, 13$, where d_0, d_1, d_2 are for π_{pre}, d_3, d_4, d_5 are for $\pi_{cyc,f}$, d_6, d_7, d_8 are for π_{cyc}, d_9, d_{10}, d_{11} are for $\pi_{cyc,l}$ and, finally, d_{12}, d_{13} are for π_{post}. The set of clocks reset within the cycle is $C_R = \{z, t, u\}$. As the path does not include any invariant constraints, all the clock constraints of the NLP are of*

type (3.13f) *(guards). The NLP constraints with respect to the index* i *of the transition along the path* π_u *are given in* (3.38).

$$i = 0(e_0) : \textit{no constraint}, \tag{3.38a}$$

$$i = 1, 4, 7(e_1) : p_1 \leq \sum_{j=0}^{j=i} d_j \leq p_2, \quad p_3 \leq d_{i-1} + d_i, \tag{3.38b}$$

$$i = 2(e_2) : d_0 + d_1 + d_2 \leq p_4, \tag{3.38c}$$

$$i = 3, 6, 9, 12(e_3) : d_i \leq p_5, \tag{3.38d}$$

$$i = 5, 8, 11(e_2) : d_{i-1} + d_i \leq p_4, \tag{3.38e}$$

$$i = 10(e_1) : p_1 \leq \left(\sum_{j=0}^{5} d_j + c \sum_{j=6}^{8} d_j + \sum_{j=9}^{10} d_j \right) \leq p_2, \quad p_3 \leq d_9 + d_{10}, \tag{3.38f}$$

$$i = 13(e_4) : p_6 \leq \left(\sum_{j=0}^{5} d_j + c \sum_{j=6}^{8} d_j + \sum_{j=9}^{13} d_j \right) \leq p_7. \tag{3.38g}$$

The clocks x *and* y *are not reset within the cycle. Therefore, for the constraints over these clocks at* $\pi_{cyc,l}$ (3.38f) *and* π_{post} (3.38g)*, the total cycle duration is multiplied by the cycle count to represent the total average duration via* (3.10)*. This example illustrates the key observations regarding the monotonicity of the non-reset clocks, as demonstrated in* (3.26)*. Specifically, the satisfaction of the lower bound constraint* $p_1 \leq x$ *on* e_1 *at* $i = 4$*, which corresponds to the first time the cycle is taken (*$\pi_{cyc,f}$*), ensures its satisfaction in all subsequent iterations. Similarly, the satisfaction of the upper bound constraint* $y \leq p_2$ *on* e_1 *at* $i = 10$ *denoting the last time the cycle is taken (*$\pi_{cyc,l}$*) guarantees its satisfaction in all preceding repetitions of the cycle. Finally, thanks to Assumption 3.2.1, the values of clocks* $u, z, t \in C_R$ *within the cyclic part solely depend on the delay variables from the corresponding cycle* (3.38b), (3.38e), (3.38d).

Remark 3.2.1 *In the proposed methods of this thesis, unrolling the first and the last cycles and using the average duration multiplied by the cycle count allows us to solve both parameter and constraint synthesis problems for a simple (unnested) cycle. When the automaton has a nested cycle, a location appears in two cycles, say cycle A and cycle B. This scenario introduces a complexity arising from the need to account for potential permutations of these cycles through the shared location (e.g., AAA, AAB, ABB, etc.), along with their associated reset actions and constraints. As a result*

of these intricate interplays, a parallel encoding approach akin to the one employed becomes infeasible for nested cycles. Therefore, the proposed methods can only be applied to the cases without nested cycles.

3.3 Cyclic Path Analysis

In Section 3.2.3, the NLP encoding for a cyclic path is defined and its correctness is proved. In this section, the feasibility and parameter synthesis problems specifically for cyclic paths, along with the proposed NLP-based solutions, are presented.

3.3.1 Make a Cyclic Path Realizable

In this section, the realizability problem for cyclic paths is formulated and the method to solve it using a NLP-based approach is presented.

Problem 3.3.1 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ for parameters, a set of parameter constraints Ω and a cyclic path π* (3.8)*, find a cycle count c and a parameter valuation γ that satisfy I and Ω ensuring the realizability of the path π with cycle count c for the $\mathcal{PA}(\gamma)$.*

As the unrolled path π_u is employed in Proposition 3.2.1, the following steps are carried out to solve Problem 3.3.1: 1) solve MILP (3.3) for paths π^0 and π^1, where π^i is obtained by unrolling the cycle π_{cyc} of π (3.8) i times; 2) solve NLP (3.13). If any of the optimization problems is feasible, then the cyclic path is realizable. Conversely, the cyclic path is unrealizable if none of the optimization problems are feasible.

3.3.2 Make a Cyclic Path Unrealizable

The constraint synthesis problem to make a cyclic path unrealizable is formulated as follows:

Problem 3.3.2 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ for each parameter, a set of parameter constraints*

Ω *and a cyclic path* π (3.8)*, find a set of parameter constraints* ω *such that for any* $c \in \mathbb{N}$ *and parameter valuation* γ *satisfying* I *and* $\Omega \cup \omega$*, the path* π^c *obtained by unrolling the cycle* π_{cyc} *within the path* π c *times is unrealizable.*

Similar to the proposed solution for non-cyclic paths in Section 3.1.2, the constraint synthesis problem is encoded as a quantified formula and then the quantified delay variables are eliminated. However, in this case, the non-existence of a cycle count c is enforced in addition to the delay variables. Recall that the cyclic path π is realizable for any $c \geq 2$, if and only if the NLP (3.13) is feasible. Parallel to (3.5), the unrealizability of the cyclic path is encoded as

$$\forall(d_0, \ldots, d_K, c) : \left\{NLP(\mathcal{PA}, \pi) \wedge \Omega^C\right\} = False \tag{3.39}$$

where $K = p + 3q + r$, $NLP(\mathcal{PA}, \pi)$ denotes the set of constraints given in (3.13) for the cyclic path π (3.8) and Ω^C is $\bigwedge_{\phi \in \Omega} \phi$.

A set of parameter constraints ω is obtained by eliminating the quantified delay variables and the cycle count parameter from (3.39). Note that the constraints obtained after the quantifier elimination are over the remaining variables, which are the parameters of the PTA. If the delay variables, together with the cycle count parameter, can not be entirely eliminated, then the cyclic path can not be forced to be unrealizable through parameter synthesis. On the other hand, if the quantified variables are eliminated from (3.39), a parameter constraint ω is obtained. In addition to the elimination, it is checked if any parameter valuation γ exists that satisfies the interval constraints I and the accumulated set of constraints $\Omega \cup \omega$ as in (3.6).

CHAPTER 4

PARAMETER SYNTHESIS UNDER SAFETY SPECIFICATIONS

In this chapter, the safety problem addressed in this thesis is defined and the proposed solution is presented.

4.1 Problem Formulation

Problem 4.1.1 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ and a set of unsafe locations $U \subset L$:*

- *Is there a parameter valuation γ that makes all of the unsafe locations $l_U \in U$ unreachable in the resulting timed automaton $\mathcal{PA}(\gamma)$?*
- *Find a parameter valuation γ if it exists.*

In other words, the goal of this section is to find a parameter valuation γ for a given PTA such that the resulting TA $\mathcal{PA}(\gamma)$ satisfies a safety specification.

4.2 Solution Approach

In this section, the algorithms for finding a parameter valuation for the considered synthesis problem against safety specifications formulated in Problem 4.1.1 are presented.

A solution is proposed to the above problem by performing a depth-first search on the given PTA. Central to the proposed solution is the synthesis of parameter constraints

for the realizable paths that end in an unsafe location such that the MILP generated for such a path, together with the synthesized constraints, becomes infeasible as described in Section 3.1.2. Furthermore, the method described in Section 3.3.2, the NLP encoding with an additional cycle count parameter for a cyclic path, is used to generate constraints such that the path is unrealizable for any cycle count parameter. These constraints are accumulated through the depth-first search and it is shown that a parameter valuation satisfying all of these constraints at the same time solves the given synthesis problem.

The main approach is summarized in Algorithm 1. From a high-level perspective, the algorithm searches for a realizable path that ends in an unsafe location in a depth-first manner. When such a path is found, it generates constraints over the parameters to make the path unrealizable. If it is not possible, there is no solution for the considered parameter synthesis problem. Otherwise, the algorithm keeps the parameter constraints for making the path unrealizable and continues with the search for other unsafe paths. The algorithm is guaranteed to terminate in finite time thanks to the analysis of the cyclic paths as a special case and pruning the search for such directions.

The stack for the depth-first search is initialized with the path $\pi_0 = l_0$ in line 1 and the accumulated constraint set Ω for the parameters is initialized in line 2. At each iteration of the main loop (lines 3 through 28), a path from the exploration stack $pathStack$ is analyzed. There are two types of paths stored in the exploration stack: non-cyclic and cyclic. The first analysis is the realizability check at line 5. Depending on the path's type, a modified version of the MILP (3.3) or NLP (3.13) is solved and a valuation making it realizable is returned by the function *getSolution* in line 5. In particular, in addition to the constraints $MILP(\mathcal{PA}, \pi)$ from (3.3) for a simple path or $NLP(\mathcal{PA}, \pi)$ from (3.13) for a cyclic path, the accumulated set of parameter constraints Ω is added to the problem. If the path (cyclic or non-cyclic) is found to be unrealizable, i.e., the function call in line 5 returns an empty set and the check on the same line holds, the exploration branch is pruned by discarding the path in line 6.

Algorithm 1 Synthesis-Safety

Require: A PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$, a set of parameters P, a parameter intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$, a set of unsafe locations $U \subset L$.

Ensure: Parameter valuation γ such that L_U is unreachable on $\mathcal{PA}(\gamma)$.

1: $pathStack = [l_0]$
2: $\Omega = \{\}$ ▷ restrictions
3: **while** $pathStack$ is not empty **do**
4: $\pi = pathStack.pop()$
5: **if** $getSolution(\pi, \Omega) == \emptyset$ **then**
6: $Continue$ ▷ prune that branch
7: **end if**
8: **if** $\pi.lastLoc \in U$ **then**
9: $\omega = makeUnrealizable(\pi)$
10: **if** $\omega == \emptyset \vee \{\gamma \in I \mid \gamma \models \bigwedge_{\phi \in \Omega \cup \omega} \phi\} == \emptyset$ **then**
11: **return** $\emptyset$ ▷ PTA cannot be made safe
12: **end if**
13: $\Omega = \Omega \cup \omega$
14: $Continue$ ▷ now unrealizable, can prune
15: **end if**
16: **if** $count(\pi, \pi.lastLoc) == 2 \ \wedge \ \pi.lastLoc \notin \pi.cycleLocations$ **then**
17: $\pi.cycleLocations = \pi.cycleLocations \ \cup getNewCycleLocations(\pi)$
18: $cyclicPath = getCyclicPath(\pi)$
19: $pathStack.push(cyclicPath)$
20: **end if**
21: **for each** $e \in \{(l, \lambda, \phi, l') \in \Delta \mid l = \pi.lastLoc\}$ **do**
22: $\pi' = \pi + e + l'$
23: **if** $count(\pi', l') == 3$ **then**
24: $Continue$ ▷ previously visited cycle
25: **end if**
26: $pathStack.push(\pi')$
27: **end for**
28: **end while**
29: **return** $optimizeSafety(P, I, \Omega, costCoefficients)$

Next, whether the path ends in an unsafe location is checked in line 8. For an unsafe path π, the algorithm tries to synthesize parameter constraints ω such that the parameter synthesis problem for the path π together with Ω and ω becomes infeasible in line 9. Here, for the parameter synthesis problem, the $MILP(\mathcal{PA}, \pi)$ encoding from (3.3) and the $NLP(\mathcal{PA}, \pi)$ encoding from (3.13) are used as explained in detail in Sections 3.1.2 and 3.3.2 for simple and cyclic paths, respectively. Suppose that this constraint synthesis attempt fails, meaning that it is not possible to find parameter constraints ω such that there exists a parameter valuation γ that satisfies $\Omega \cup \{\omega\}$ and π of $\mathcal{PA}(\gamma)$ is not realizable. In that case, the algorithm returns an empty set in line 11. In other words, it is not possible to find a parameter valuation to ensure that each unsafe path seen until this point is unrealizable.

On the other hand, if a set of constraints ω making the path unrealizable is found, they are added to the set Ω. Finally, an unsafe path is not further explored (line 14) since the synthesized constraints ω guarantee that this path is unrealizable. Note that the algorithm starts with the path $\pi_0 = l_0$ and iteratively traverses the underlying graph of the input PTA. For a path ending in an unsafe location, the algorithm either returns with a negative result or adds some parameter constraints that make the path unrealizable and does not further explore that path. Therefore, an unsafe location can only be the last location of a path in the stack.

The exploration and further analysis performed for safe and realizable paths are given in lines 16- 28. First, whether the last location creates a cycle or not is checked in line 16. In the same line, it is also checked that if this is the first location of the cycle as well, so the same cycle is not unnecessarily processed from different pivot points. If that check passes, the path is marked as cyclic (line 18) and added back to the exploration stack (line 19). Note that the realizability check (line 5) and the constraint synthesis for simple paths (line 9) are already performed over the considered path (e.g., the analysis is performed when the cycle is rolled out once). Finally, for each transition leaving the last location of the considered path π (line 21), a new path π' is created (line 22). As tested in line 23, if a new path has the last location three times, i.e., it has the same cycle twice, that path is discarded in line 24. Any new path passing this test is added to the stack in line 26.

If the stack is emptied without observing an infeasible problem (line 11), the execution reaches line 29. At this point, the constraints in Ω ensure that any parameter valuation satisfying all of those is a solution to the safety problem. Here, the algorithm returns the optimal parameter valuation, which best conforms to the parameter intervals relation I and the constraints Ω according to a given cost criteria.

The function at line 9 of Algorithm 1 constructs the quantified expression (3.5) for the given path and runs quantifier elimination [33, 34] on it. The failure to eliminate the quantified delay variables is represented with an empty set in line 10. The check for this type of failure and the non-existence of parameter values are performed in the same line. Note that the parameter constraint, retrieved after quantifiers are eliminated, is in the form presented in Definition 3.1.1; thus, it may contain disjunctive inequalities.

Example 4.2.1 *Algorithm 1 is run on the PTA illustrated in Figure 3.1. The depth-first path traversal performed by the algorithm is illustrated in Figure 4.1. Initially, the path* l_0 *is analyzed and the path* l_0, e_0, l_1 *is inserted into the stack. This path is realizable and safe; thus, paths* l_0, e_0, l_1, e_1, l_2 *and* l_0, e_0, l_1, e_4, l_4 *are inserted into the stack after the second iteration of the main loop. As explained in Example 3.1.1, the MILP constructed for the path* $\pi_2 = l_0, e_0, l_1, e_1, l_2$ *is infeasible. Hence, the algorithm cuts the exploration of that branch at line 5, i.e., it does not process the paths* $l_0, e_0, l_1, e_1, l_2, e_2, l_3$ *or* $l_0, e_0, l_1, e_1, l_2, e_2, l_3, e_3, l_6$.

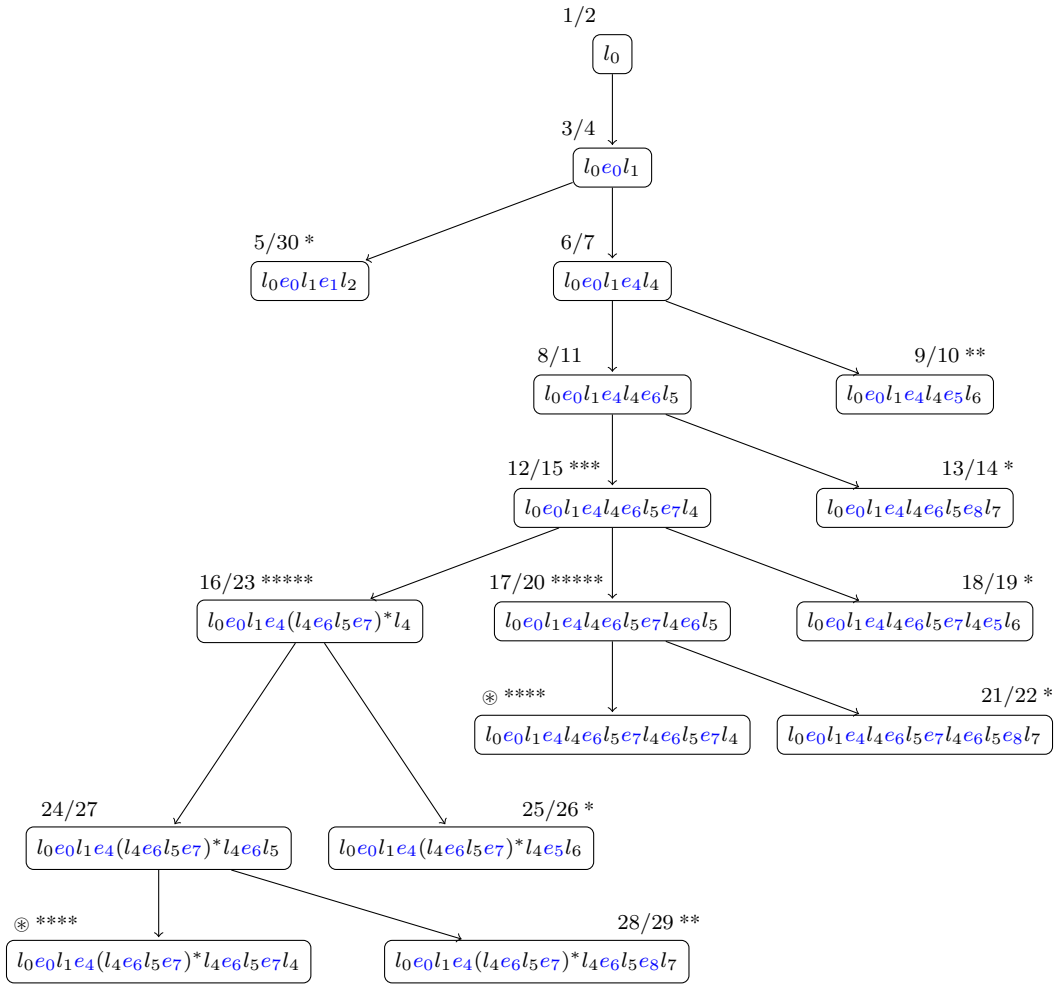

Figure 4.1: Depth-first traversal of the example PTA shown in Figure 3.1. The i and j indices show the order of a path that goes into and out of the exploration stack, respectively. * The path is unrealizable. ** The realizable path ends in an unsafe location. A constraint ω is collected to make it unrealizable. *** The path's last location appears twice and is not part of a previously encountered cycle. **** The path's last location appears three times. ⊛ The path is not pushed to the exploration stack and is discarded on the fly. ***** The last location of the path is in a previously encountered cycle, so a cyclic path is not created from here.

Recall that in Algorithm 1, a path is marked as cyclic when $c = 1$ and $r = 0$ and pushed back to the exploration stack. However, in the subsequent iterations of the main loop, the possible post-cycle paths are considered for the same cyclic path.

Note that while only considering (3.39) within Algorithm 1 (line 9) is sufficient, a complete solution for Problem 3.3.2 is obtained by also generating constraints for paths π^0 and π^1 as explained in Section 3.1.2.

Corollary 4.2.1 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ and a set of unsafe locations $U \subset L$, there exists a parameter valuation γ such that U is unreachable on $\mathcal{PA}(\gamma)$ if and only if Algorithm 1 returns one.*

Corollary 4.2.1 is a direct consequence of Proposition 3.1.1, Proposition 3.2.1 and the constraint accumulation method applied within Algorithm 1 via depth-first analysis.

CHAPTER 5

PARAMETER SYNTHESIS UNDER EXISTENTIAL LIVENESS SPECIFICATIONS

In this chapter, the reachability and repeated reachability problems are defined and the methods adapted from Chapter 4 to solve them are proposed.

5.1 Reachability

5.1.1 Problem Formulation

Problem 5.1.1 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ and a set of target locations $T \subset L$:*

- *Is there a parameter valuation γ that makes any of the target locations $l_T \in T$ reachable in the resulting timed automaton $\mathcal{PA}(\gamma)$?*
- *Find a parameter valuation γ if it exists.*

5.1.2 Solution Approach

Different than a safety property, a reachability property can be satisfied by a single witness. Considering this observation, multiple solutions with different characteristics are proposed for solving Problem 5.1.1.

For a reachability specification, the decision variables of a MILP/NLP problem related to delay variables or cycle counts are controllable. However, eliminating delay

variables and cycle counts from the quantified expression is required to find the most optimum solution according to a cost criteria defined over parameters.

The first solution approach to Problem 5.1.1 is summarized in Algorithm 2. It returns the most optimum parameter valuation according to a given cost criteria. The key differences of it with Algorithm 1 are as follows:

- The last location check is done against the set of target locations, which is passed as an input.
- *getSolution* function does not consider the pre-collected constraints Ω.
- *makeRealizable* function is called in line 9. In that function, the quantification to build the constraints for a path is an existential one and the quantified expression is required to be $True$. It is

$$\exists(d_0, \ldots, d_{n-1}) : \left\{MILP(\mathcal{PA}, \pi) \wedge \Omega^C\right\} = True. \tag{5.1}$$

  for simple paths and it is

$$\exists(d_0, \ldots, d_K, c) : \left\{NLP(\mathcal{PA}, \pi) \wedge \Omega^C\right\} = True \tag{5.2}$$

  for cyclic paths. A set of parameter constraints ω is obtained by eliminating the quantified delay variables from (5.1) or (5.2). The constraints obtained after the quantifier elimination are over the remaining decision variables, which are the parameters. For any parameter valuation satisfying the found parameter constraint ω, there exists a delay valuation for simple paths and delay valuation with cycle count for cyclic paths which together with the parameter valuation makes the path realizable.
- The return value ω of the *makeRealizable* function is not needed to be verified since the path realizability is already checked in line 5.
- Since a parameter valuation satisfying any of the collected constraints is a solution to the problem, after the loop, the collected constraints are combined in a disjunctive way.

Lines 13 through 25 of the main loop in Algorithm 2 generate the paths to be traversed from the current path π exactly as Algorithm 1 (lines 16 through 28) does.

Algorithm 2 Synthesis-Reachability-Optimum

Require: A PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$, a set of parameters P, a parameter intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$, a set of target locations $T \subset L$.

Ensure: Parameter valuation γ such that L_T is reachable on $\mathcal{PA}(\gamma)$.

1: $pathStack = [l_0]$
2: $\Omega = \{\}$ ▷ restrictions
3: **while** $pathStack$ is not empty **do**
4: $\pi = pathStack.pop()$
5: **if** $getSolution(\pi) == \emptyset$ **then**
6: $Continue$ ▷ prune that branch
7: **end if**
8: **if** $\pi.lastLoc \in T$ **then**
9: $\omega = makeRealizable(\pi)$
10: $\Omega = \Omega \cup \omega$
11: $Continue$ ▷ already reachable. comment out for repeated reachability
12: **end if**
13: **if** $count(\pi, \pi.lastLoc) == 2 \ \wedge \ \pi.lastLoc \notin \pi.cycleLocations$ **then**
14: $\pi.cycleLocations = \pi.cycleLocations \cup getNewCycleLocations(\pi)$
15: $cyclicPath = getCyclicPath(\pi)$
16: $pathStack.push(cyclicPath)$
17: **end if**
18: **for each** $e \in \{(l, \lambda, \phi, l') \in \Delta \mid l = \pi.lastLoc\}$ **do**
19: $\pi' = \pi + e + l'$
20: **if** $count(\pi', l') == 3$ **then**
21: $Continue$ ▷ previously visited cycle
22: **end if**
23: $pathStack.push(\pi')$
24: **end for**
25: **end while**
26: **return** $optimize(P, I, Or(...\Omega), costCoefficients)$

Line 5 of Algorithm 2 returns a valuation making the considered path realizable. In

cases where the most optimum solution is not needed, the algorithm can be simplified and the execution times can be improved significantly by removing quantifier elimination.

Algorithm 3 is presented to solve Problem 5.1.1 more efficiently regarding those observations. It is adapted from Algorithm 2 with some modifications. Instead of accumulating the constraints, it accumulates the solutions found during the traversal. Depending on the configuration in line 11, it returns the first or all the realizable paths to target locations.

Algorithm 2 returns the optimum solution for a reachability problem. In cases where a faster solution is preferred over an optimum one, Algorithm 3 can be used to get the first or all the realizable paths to target locations. Consequently, three approaches are presented to solve Problem 5.1.1, which return the best parameter valuation, first path and all paths.

Algorithm 3 Synthesis-Reachability

Require: A PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$, a set of parameters P, a parameter intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$, a set of target locations $T \subset L$.

Ensure: Parameter valuation γ such that L_T is reachable on $\mathcal{PA}(\gamma)$.

$pathStack = [l_0]$
$\Gamma = \{\}$ ▷ valuations
while $pathStack$ is not empty **do**
    $\pi = pathStack.pop()$
    $\gamma = getSolution(\pi)$
    if $\gamma == \emptyset$ **then**
        $Continue$ ▷ prune that branch
    end if
    if $\pi.lastLoc \in T$ **then**
        $\Gamma = \Gamma \cup \gamma$
        $//Break$ ▷ uncomment to return the first path
        $Continue$ ▷ already reachable. comment out for repeated reachability
    end if
    if $count(\pi, \pi.lastLoc) == 2 \ \wedge \ \pi.lastLoc \notin \pi.cycleLocations$ **then**
        $\pi.cycleLocations = \pi.cycleLocations \ \cup getNewCycleLocations(\pi)$
        $cyclicPath = getCyclicPath(\pi)$
        $pathStack.push(cyclicPath)$
    end if
    for each $e \in \{(l, \lambda, \phi, l') \in \Delta \mid l = \pi.lastLoc\}$ **do**
        $\pi' = \pi + e + l'$
        if $count(\pi', l') == 3$ **then**
            $Continue$ ▷ previously visited cycle
        end if
        $pathStack.push(\pi')$
    end for
end while
return Γ

5.2 Repeated Reachability

5.2.1 Problem Formulation

Problem 5.2.1 *Given a PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$ with a set of parameters P, an intervals relation $I : P \to \mathbb{N} \times \mathbb{N}$ and a set of target locations $T \subset L$:*

- *Is there a parameter valuation γ that makes any of the target locations $l_T \in T$ repeatedly reachable in the resulting timed automaton $\mathcal{PA}(\gamma)$?*
- *Find a parameter valuation γ if it exists.*

5.2.2 Solution Approach

When a reachable location is a part of a cycle, that cycle does not have an upper-bound timing constraint and the path satisfies Assumption 3.2.1; that location is repeatedly reachable. The following proposition formalizes the proposed solution to the repeated reachability problem utilizing this observation.

Proposition 5.2.1 *A location l of a PTA is repeatedly reachable if all of the below hold:*

- *l is part of the cycle part π_{cyc} of a cyclic path π,*
- *π_{cyc} does not have any upper bound timing constraints of invariants or guards over a clock that is not reset within the cycle,*
- *Path π satisfies Assumption 3.2.1,*
- *l is reachable.*

Algorithm 2 and Algorithm 3 already run checks for the last two conditions. For a repeated reachability specification, those algorithms can easily be adapted to solve Problem 5.2.1 by adding an extra check to accommodate the first two conditions. Replacing the checks at line 8 of Algorithm 2 and line 9 of Algorithm 3 with the one in below snippet gives the algorithms to solve the repeated reachability problem.

Algorithm 4 Snippet-Repeated-Reachability-Check

1: **if** $\pi.lastLoc \in T \wedge \pi.lastLoc \in \pi.cycleLocations$
$\wedge hasUpperBoundConstraints(\pi.lastCycle) == False$ **then**
2: ...
3: **end if**

Alternatively, instead of running the checks for the first two conditions during the main loop, a preprocessing step to filter the target locations may be used. After applying the filter on the target location set T via static analysis, as shown in Algorithm 5, a repeated reachability problem reduces to a reachability problem and can be solved by running Algorithm 2 or Algorithm 3. Algorithm 5 takes only the PTA as input and returns the list of locations that are on a cycle without an upper bound constraint. Therefore, the locations returned satisfy the first two conditions of Proposition 5.2.1

Algorithm 5 Synthesis-Repeated-Reachability-Possible-Locations

Require: A PTA $\mathcal{PA} = (L, l_0, C, \Delta, Inv)$.

Ensure: Locations $l \in L$ such that l has the potential to be repeatedly reachable.

1: $L' = \{\}$
2: $cycleList = GetCycles(\mathcal{PA})$
3: **while** $cycleList$ is not empty **do**
4: $cycle = cycleList.pop()$
5: **if** $hasUpperBoundConstraints(cycle) == False$ **then**
6: $L'.push(cycle.locations)$
7: **end if**
8: **end while**
9: **return** L'

CHAPTER 6

CASE STUDIES

6.1 Implementation

In this section, we evaluate the proposed methods over several benchmarks from the literature. We implemented the proposed parameter synthesis methods in a tool called PaSTA in Python. The tool uses the Z3 [34] library to solve the MILP, NLP, optimization and quantifier elimination problems. All experiments were run on a PC with AMD Ryzen 9 5900X 12-Core 3.70 GHz processor, 32 GB DDR4 RAM and Windows 11 OS.

We applied the proposed synthesis methods to 13 PTAs selected from the benchmark library [6, 35], along with the running example illustrated in Figure 3.1. Table 6.1 provides the list of these PTAs and their properties, along with a unique id for cross-referencing with tables listing the benchmark results for different specification problems in the following sections. In that table, the one PTA marked with a * is illustrated in Figure 3.1, and the other PTAs are from the benchmark library. Id, Loc., Tr., Cl., Cyc. and Para. columns show the model id, the number of locations, transitions, clocks, cycles and parameters, respectively. Except for *JLR15_Fig6*, the parameters of all PTAs from the benchmark library are missing an upper bound. The missing upper bounds for parameters are set to 10 for all of them. The PTAs from the benchmark library are illustrated in Appendix A.

Table 6.1: The sample models used for the experiments.

Name	Id	Loc.	Tr.	Cl.	Cyc.	Para.
BlT09_fig1	1	4	6	2	3	2
Cycles_2	2	2	2	2	1	1
Cycles_5_6	3	2	2	2	1	1
Cycles_notFiniteDisjunction	4	2	2	2	1	1
JLR15_Fig6	5	2	2	2	1	2
NuclearPlant	6	6	8	2	1	4
Synth_int01	7	2	2	2	1	1
Synth_InvN	8	2	2	2	1	1
Synth_N	9	2	2	2	1	1
Synth_pN	10	2	2	2	1	2
Synth_pNplusq	11	2	2	2	1	3
Train	12	5	6	2	1	3
UntimedLanguage	13	4	6	2	3	1
PTAfromFig1 *	14	8	9	2	1	3

6.2 Safety

In this section, we run Algorithm 1 on the 14 PTAs listed in Table 6.1. The benchmarks with ids 6 and 12 have safety specifications in the original versions. For the rest, the location farthest from the initial location is marked as unsafe for the analysis.

The experimental results are reported in Table 6.2. The average computation times of three runs are reported in milliseconds. In path counts, feasibility and QE columns, "/" separates the simple and cyclic paths' results. The total analyzed paths count, the realizable paths count and the realizable unsafe paths count are reported in the path

counts column in the given order. The rows marked with a star (*) show the results found using real-valued parameters. The computation times for Imitator are shown in the last column, where "-" indicates that a result is not generated within 10 minutes. Additionally, Imitator is not run on the real-valued parameters, which are indicated with "#".

Table 6.2: The experimental results of the benchmarks for safety.

Id	Path Counts	Computation times			
		Feasibility	QE	Z3/Total	Imitator
1	22-15-1/63-35-0	150.33/562.84	6.82/0.00	719.99/740.18	5
2	4-3-1/1-0-0	18.38/5.63	4.51/0.00	28.53/35.89	5
3	4-3-1/2-1-0	17.98/10.18	4.84/0.00	33.00/43.93	-
4	4-4-2/2-2-1	20.98/10.84	8.61/55.17	95.60/319.96	-
5	4-4-2/2-1-0	20.54/15.12	12.29/0.00	47.94/94.35	-
5*	4-4-2/2-2-1	17.96/10.85	12.58/140.58	181.96/909.69	#
6	16-13-3/0-0-0	201.07/0.00	14.99/0.00	216.06/241.22	7
7	4-3-1/2-1-0	21.80/11.53	5.66/0.00	38.99/45.62	-
7*	4-3-1/2-2-1	19.73/10.69	4.73/467.39	502.54/972.47	#
8	4-3-1/1-0-0	19.59/4.14	5.14/0.00	28.87/36.02	-
9	4-4-2/2-2-1	22.74/10.66	8.69/5.12	47.22/61.86	-
10	4-4-2/2-2-1	21.65/34.05	9.12/7.41	72.23/91.34	-
11	4-4-2/2-2-1	22.58/78.61	9.44/6.25	116.88/133.88	-
12	12-10-2/0-0-0	76.03/0.00	8.95/0.00	84.99/99.07	5
13	14-10-2/14-8-1	82.14/119.43	13.24/12.81	227.62/260.12	41
14	11-7-1/4-3-1	67.86/39.78	6.27/1043.18	1157.09/1390.09	-

In all cases, the parameter synthesis problem is solved within a few seconds. The

results reveal that the computation time increases with the number of analyzed paths. Furthermore, as expected, the analysis of cyclic paths (NLP and QE for nonlinear constraints) increases the total computation time. In addition, we observe that the feasibility check procedures effectively prune a significant number of search directions in experiments 1, 12-14, thereby improving the computation time.

We further conducted experiments where we treated the PTA parameters as real-valued instead of integer-valued. In most cases, the computation times remained similar even if the MILP feasibility checks were switched to LP. However, as indicated in Table 6.2, two specific cases exhibited significant changes in computation times. In both instances, the introduction of real-valued parameters rendered an unsafe path realizable, necessitating additional quantifier elimination procedures.

Although the underlying graphs for the PTAs 2-5, 7-11 are the same (Figure A.2, Figure A.3, Figure A.4), the analysis results highlight distinct characteristics influenced by various factors. The number of locations, transitions, parameters, clocks and constraints on the analyzed paths directly affect the number of decision variables of MILP and NLP problems, consequently affecting the computation times. Additionally, the structure of a constraint also affects the computation time. In particular, an equality relation is more restrictive than a less than or a greater than relation. The effect of that on the computation times becomes apparent when comparing the results for PTAs *Cycles_notFiniteDisjunction* (4) and *Synth_N* (9).

We run Imitator on the same set of parameter synthesis problems using the Docker image on the same machine. We experimented with the original and parameter-upper-bound-added versions of PTAs. The computation times for the second case are given in the last column of Table 6.2. For 9 out of 14 cases, Imitator did not generate a result within 10 minutes and was terminated externally. Adding the upper bound for parameters affects only the PTA *UntimedLanguage*, i.e., Imitator returns only after adding the upper bound for the parameter.

Although Imitator solves the synthesis problem faster for 5 PTAs, it cannot solve the remaining 9 cases, whereas our new approach finds a solution to all of the 14 cases. It is also important to note that the proposed algorithm solves all the cases in a reasonable amount of time, i.e., under 1.4 seconds. In addition, note that the proposed

method efficiently handles scenarios in which an unsafe location becomes reachable when a cycle is taken several times, as exemplified in our running example Example 3.2.1, e.g., case *PTAfromFig1*. Another example highlighting the capabilities of the proposed method is shown in Figure A.2. In PTA *Cycles_2*, parameter p limits the total time spent in location l_0. Whereas, in *Cycles_5_6*, there is no upper bound on the time spent in location l_0, for which Imitator failed to generate a result in 10 minutes. Observe that the invariant $p >= y$ and the guard $y >= 6$ must be satisfied to reach the unsafe location. The proposed constraint synthesis method utilizing the NLP encoding generates the constraint $p < 6$, which makes l_1 unreachable.

6.3 Reachability

In this section, we run Algorithm 2 and Algorithm 3 on the 14 PTAs listed in Table 6.1. Similar to the modification in the previous section, the location farthest from the initial location is marked as the target for the benchmarks with ids 6 and 12, which have safety specifications in the original versions.

The experimental results are reported in Table 6.3. The rows with ids *14 - l_6* and *14 - l_7* correspond to two additional cases where the PTA from Figure 3.1 is used with l_6 and l_7 as the single target locations. The average computation times of three runs are reported in milliseconds. In Best / All and First columns, top line shows the execution times to get the best valuation, all paths and the first path, respectively. "/" in the bottom line separates the simple and cyclic paths' results. The total analyzed paths count, the realizable paths count and the count of the realizable paths that end in a target location are reported in the below line in the given order. The computation times of Imitator for the all paths and first path cases are shown in the last column, where "-" indicates that a result is not generated within 10 minutes.

Table 6.3: The experimental results of the benchmarks for reachability.

Id	Best / All	First	Imitator All / First
1	815.77 / 447.58 22 - 22 - 8 / 63 - 63 - 28	16.15 4 - 4 - 1 / 0 - 0 - 0	5 / 6
2	188.93 / 26.2 4 - 3 - 1 / 2 - 2 - 1	15.53 4 - 3 - 1 / 0 - 0 - 0	5 / 6
3	57.35 / 27.06 4 - 4 - 2 / 2 - 2 - 1	9.03 2 - 2 - 1 / 0 - 0 - 0	- / 6
4	276.72 / 26.11 4 - 4 - 2 / 2 - 2 - 1	9.8 2 - 2 - 1 / 0 - 0 - 0	- / 5.67
5	1054.66 / 29.52 4 - 4 - 2 / 2 - 2 - 1	8.5 2 - 2 - 1 / 0 - 0 - 0	- / 5.33
6	175.35 / 110.77 16 - 16 - 6 / 0 - 0 - 0	15.69 3 - 3 - 1 / 0 - 0 - 0	6.33 / 5.67
7	1066.65 / 27.45 4 - 3 - 1 / 2 - 2 - 1	17.93 4 - 3 - 1 / 0 - 0 - 0	- / 5.67
8	33.23 / 24.16 4 - 3 - 1 / 2 - 1 - 0	15.66 4 - 3 - 1 / 0 - 0 - 0	- / 5
9	45.11 / 28.82 4 - 4 - 2 / 2 - 2 - 1	10.53 2 - 2 - 1 / 0 - 0 - 0	- / 5.67
10	60.79 / 37.34 4 - 4 - 2 / 2 - 2 - 1	9.83 2 - 2 - 1 / 0 - 0 - 0	- / 5.67
11	75.53 / 49.72 4 - 4 - 2 / 2 - 2 - 1	8.59 2 - 2 - 1 / 0 - 0 - 0	- / 5.67
12	98.53 / 61.32 12 - 12 - 4 / 0 - 0 - 0	12.7 3 - 3 - 1 / 0 - 0 - 0	5.67 / 5.67

Table 6.3 (continued)

Id	Best / All	First	Imitator All / First
13	3927.07 / 262.19 16 - 12 - 2 / 24 - 13 - 1	18.63 4 - 4 - 1 / 0 - 0 - 0	37.67 / 5.67
14	1623.58 / 86.58 11 - 8 - 2 / 4 - 4 - 2	17.37 4 - 4 - 1 / 0 - 0 - 0	- / 6
14 - l_6	127.02 / 84.97 11 - 8 - 2 / 4 - 4 - 1	17.17 4 - 4 - 1 / 0 - 0 - 0	- / 5.67
14 - l_7	1580.91 / 84.54 11 - 8 - 0 / 4 - 4 - 1	79.35 10 - 8 - 0 / 4 - 4 - 1	- / 662.67

Imitator is very fast to find the first realizable path to a target location in almost all cases. However, as the output it produces shows, it uses unrolling to solve the problems regarding a cyclic path. Hence, even finding the first path that solves the reachability problem for case *14 - l_7* takes 100 times more than its average on the remaining 15 cases. Imitator suffers when analyzing paths that become realizable after taking a cycle several times, as exemplified in Example 3.2.1. This behavior becomes more apparent when Imitator is asked for all the paths, as it cannot return for 11 cases, all of which have cycles that can be taken infinitely many times.

On the other hand, our proposed approach encodes the cycles using the NLP approach and terminates for all 16 cases with a solution. The *best-case* method finds all the paths so that it evaluates the same number of paths as the *all-cases* approach. When there is more than one solution, the *first-case* approach evaluates fewer paths as expected. In fact, the *first-case* approach always evaluates 1 realizable path ending in a target location and terminates. As expected, execution time is the worst for the *best-case* approach since it finds all the paths and runs quantifier elimination when needed. For all the cases, just removing quantifier elimination and returning all the solution paths through the *all-cases* method reduces the execution time to under 500 milliseconds.

CHAPTER 7

CONCLUSIONS

This thesis studies the parameter synthesis problem for a PTA under safety, reachability and repeated reachability specifications. An algorithm that provides a complete solution for PTAs without nested cycles is presented.

The proposed algorithms employ a depth-first search on the PTA, utilizing MILP encoding for non-cyclic paths and a novel NLP encoding for cyclic paths to check path realizability. When a realizable path ending in an unsafe or target location is found, quantifier elimination is applied to MILP/NLP constraints to derive parameter constraints that render the path unrealizable or realizable for a safety or reachability property, respectively. Computationally, the most challenging problems of this approach are checking the realizability and forcing the realizability or unrealizability of a cyclic path. Fourier-Motzkin and Gaussian elimination methods are employed to solve those problems. This novel approach of encoding simple cyclic paths as an NLP problem, together with the solutions to their realizability and forced realizability or unrealizability problems, is the central contribution of this work to get a complete solution to the parameter synthesis problem of a PTA from a safety, reachability and repeated reachability perspective. The TA obtained from the generated parameters is proved to satisfy the given specification when the algorithm produces a solution. Finally, the optimal parameters satisfying the derived constraints are generated, where the cost is defined as the sum of the parameter values multiplied by their given coefficient. The benchmark results clearly indicate the efficacy and efficiency of the proposed methods.

The proposed methods substantially contribute to the safety and reachability analysis of PTAs and the real-time systems they are modeling.

The main limitation of the proposed NLP encoding, hence the algorithms, is that it only works for simple unnested cycles, as stated in Remark 3.2.1.

Future research directions include extending the proposed methods to address these limitations and applying the proposed approach to related problems such as controller synthesis. Investigating new ways to loosen the nested cycle restrictions is a possible direction. One such way is removing the transitions that end in the start location and on which all the clocks get reset to eliminate the nested cycles. However, this approach requires static analysis and careful consideration of the removed transitions. Adding controller synthesis capabilities to the proposed methods is another future research objective that will be a significant improvement. Satisfying a pair of safety and (repeated) reachability specifications simultaneously will contribute dramatically to the proposed methods' usefulness. Dead-lock analysis and getting dead-lock-free solutions will be another important improvement. Also, the proposed methods can be extended for unavoidability specifications.

REFERENCES

[1] R. Alur and D. L. Dill, "A theory of timed automata," Theoretical computer science, vol. 126, no. 2, pp. 183–235, 1994.

[2] R. Alur, "Timed automata," in Computer Aided Verification: 11th International Conference, CAV'99 Trento, Italy, July 6–10, 1999 Proceedings 11, pp. 8–22, Springer, 1999.

[3] K. G. Larsen and W. Yi, "Time abstracted bisimulation: Implicit specifications and decidability," in Mathematical Foundations of Programming Semantics: 9th International Conference New Orleans, LA, USA, April 7–10, 1993 Proceedings 9, pp. 160–176, Springer, 1994.

[4] É. André, "What's decidable about parametric timed automata?," International Journal on Software Tools for Technology Transfer, vol. 21, no. 2, pp. 203–219, 2019.

[5] E. A. Göl, "Control synthesis for parametric timed automata under reachability," Turkish Journal of Electrical Engineering and Computer Sciences, vol. 29, no. 3, pp. 1751–1764, 2021.

[6] T. I. Team, "The imitator benchmarks library (v2.0)."

[7] A. Fehnker, "Scheduling a steel plant with timed automata," in Proceedings Sixth International Conference on Real-Time Computing Systems and Applications. RTCSA'99 (Cat. No. PR00306), pp. 280–286, IEEE, 1999.

[8] A. David, J. Illum, K. G. Larsen, and A. Skou, "Model-based framework for schedulability analysis using uppaal 4.1," in Model-based design for embedded systems, pp. 117–144, CRC Press, 2018.

[9] N. Guan, Z. Gu, Q. Deng, S. Gao, and G. Yu, "Exact schedulability analysis for static-priority global multiprocessor scheduling using model-checking," in IFIP

International Workshop on Software Technolgies for Embedded and Ubiquitous Systems, pp. 263–272, Springer, 2007.

[10] M. Kwiatkowska, A. Mereacre, N. Paoletti, and A. Patanè, "Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques," in Hybrid Systems Biology: Fourth International Workshop, HSB 2015, Madrid, Spain, September 4-5, 2015. Revised Selected Papers 4, pp. 119–140, Springer, 2015.

[11] Z. Jiang, M. Pajic, R. Alur, and R. Mangharam, "Closed-loop verification of medical devices with model abstraction and refinement," International Journal on Software Tools for Technology Transfer, vol. 16, no. 2, pp. 191–213, 2014.

[12] F. Wang, "Formal verification of timed systems: A survey and perspective," Proceedings of the IEEE, vol. 92, no. 8, pp. 1283–1305, 2004.

[13] D. Soliman, K. Thramboulidis, and G. Frey, "Transformation of function block diagrams to uppaal timed automata for the verification of safety applications," Annual Reviews in Control, vol. 36, no. 2, pp. 338–345, 2012.

[14] M. Y. Kara and E. A. Gol, "Adaptive cruise control with timed automata," IFAC-PapersOnLine, vol. 53, no. 2, pp. 1918–1923, 2020.

[15] M. Pajic, R. Mangharam, O. Sokolsky, D. Arney, J. Goldman, and I. Lee, "Model-driven safety analysis of closed-loop medical systems," IEEE Transactions on Industrial Informatics, vol. 10, no. 1, pp. 3–16, 2012.

[16] A. David, K. G. Larsen, G. Behrmann, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks, "Uppaal 4.0," in Third International Conference on the Quantitative Evaluation of SysTems (QEST) 2006, pp. 125–126, IEEE Computer Society Press, 2006.

[17] É. André, L. Fribourg, U. Kühne, and R. Soulat, "Imitator 2.5: A tool for analyzing robustness in scheduling problems," in FM 2012: Formal Methods: 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings 18, pp. 33–36, Springer, 2012.

[18] T. A. Henzinger, J. Preussig, and H. Wong-Toi, "Some lessons from the hytech experience," in Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), vol. 3, pp. 2887–2892, IEEE, 2001.

[19] É. André, L. Fribourg, J.-M. Mota, and R. Soulat, "Verification of an industrial asynchronous leader election algorithm using abstractions and parametric model checking," in Verification, Model Checking, and Abstract Interpretation: 20th International Conference, VMCAI 2019, Cascais, Portugal, January 13–15, 2019, Proceedings 20, pp. 409–424, Springer, 2019.

[20] R. Alur, T. A. Henzinger, and M. Y. Vardi, "Parametric real-time reasoning," in Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, pp. 592–601, 1993.

[21] É. André and R. Soulat, "Synthesis of timing parameters satisfying safety properties," in Reachability Problems: 5th International Workshop, RP 2011, Genoa, Italy, September 28-30, 2011. Proceedings 5, pp. 31–44, Springer, 2011.

[22] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin, "Robust safety of timed automata," Formal Methods in System Design, vol. 33, no. 1-3, pp. 45–84, 2008.

[23] E. André, M. Knapik, W. Penczek, and L. Petrucci, "Controlling actions and time in parametric timed automata," in 2016 16th International Conference on Application of Concurrency to System Design (ACSD), pp. 45–54, IEEE, 2016.

[24] É. André and A. Kryukov, "Parametric non-interference in timed automata," in 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 37–42, IEEE, 2020.

[25] T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager, "Linear parametric model checking of timed automata," The Journal of Logic and Algebraic Programming, vol. 52, pp. 183–220, 2002.

[26] J. Bendík, A. Sencan, E. A. Gol, and I. Černá, "Timed automata robustness analysis via model checking," Logical Methods in Computer Science, vol. 18, 2022.

[27] E. A. Göl, "Control synthesis for parametric timed automata under unavoidability specifications," in 2021 European Control Conference (ECC), pp. 740–745, IEEE, 2021.

[28] É. André and S.-W. Lin, "Learning-based compositional parameter synthesis for event-recording automata," in Formal Techniques for Distributed Objects, Components, and Systems: 37th IFIP WG 6.1 International Conference, FORTE 2017, Held as Part of the 12th International Federated Conference on Distributed Computing Techniques, DisCoTec 2017, Neuchâtel, Switzerland, June 19-22, 2017, Proceedings 37, pp. 17–32, Springer, 2017.

[29] É. André, V. Bloemen, L. Petrucci, and J. Van de Pol, "Minimal-time synthesis for parametric timed automata," in Tools and Algorithms for the Construction and Analysis of Systems: 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part II 25, pp. 211–228, Springer, 2019.

[30] A. Jovanovic, D. Lime, and O. H. Roux, "Integer parameter synthesis for real-time systems," IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 445–461, 2015.

[31] L. Bozzelli and S. La Torre, "Decision problems for lower/upper bound parametric timed automata," Formal Methods in System Design, vol. 35, no. 2, pp. 121–151, 2009.

[32] P. Bezděk, N. Beneš, J. Barnat, and I. Černá, "Ltl parameter synthesis of parametric timed automata," in Software Engineering and Formal Methods: 14th International Conference, SEFM 2016, Held as Part of STAF 2016, Vienna, Austria, July 4-8, 2016, Proceedings 14, pp. 172–187, Springer, 2016.

[33] N. Bjørner, "Linear quantifier elimination as an abstract decision procedure," in Automated Reasoning (J. Giesl and R. Hähnle, eds.), (Berlin, Heidelberg), pp. 316–330, Springer Berlin Heidelberg, 2010.

[34] N. Bjørner and L. Nachmanson, "Navigating the universe of z3 theory solvers,"

in Formal Methods: Foundations and Applications (G. Carvalho and V. Stolz, eds.), (Cham), pp. 8–24, Springer International Publishing, 2020.

[35] É. André, D. Marinho, and J. van de Pol, "A benchmarks library for extended parametric timed automata," in International Conference on Tests and Proofs, pp. 39–50, Springer, 2021.

APPENDIX A

PTA USED IN BENCHMARKS

PTA from the benchmark library [6] used for the case studies in Chapter 6 are illustrated below. In all of them, start location is renamed to l_0 and following location names are found by incrementing the index. All parameter names are renamed to p, q, r and t. All clocks are renamed to x and y.

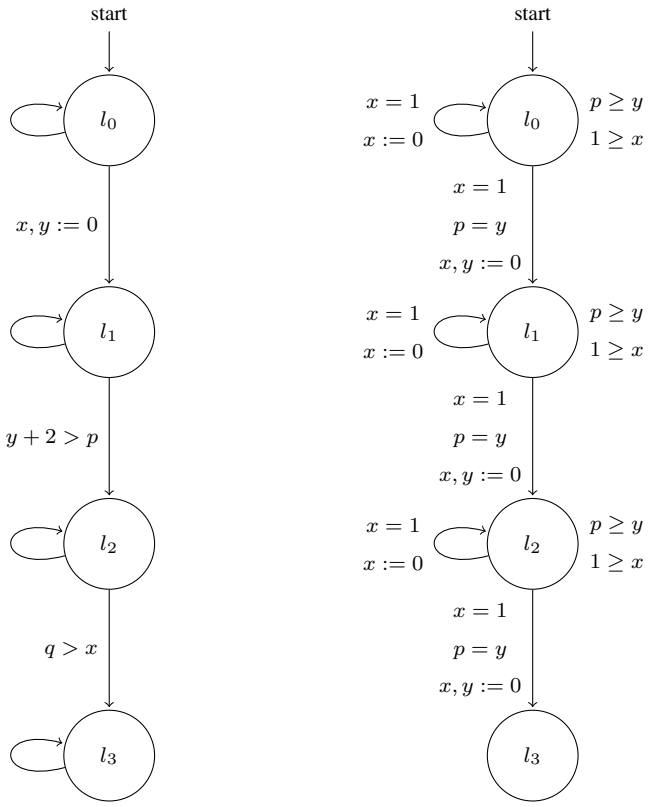

Figure A.1: *BlT09_fig1* (left) and *UntimedLanguage* (right) PTAs from the Imitator benchmark library. In both, l_3 is marked as the unsafe or target location and the upper bound for parameters are set to 10.

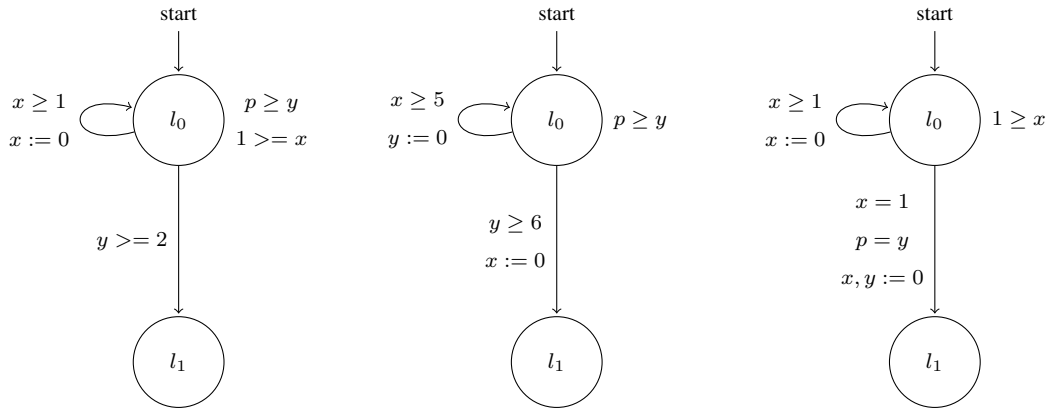

Figure A.2: *Cycles_2* (left), *Cycles_5_6* (middle) and *Cycles_notFiniteDisjunction* (right) PTAs from the Imitator benchmark library. In all, l_1 is marked as the unsafe or target location and the upper bound for parameters are set to 10.

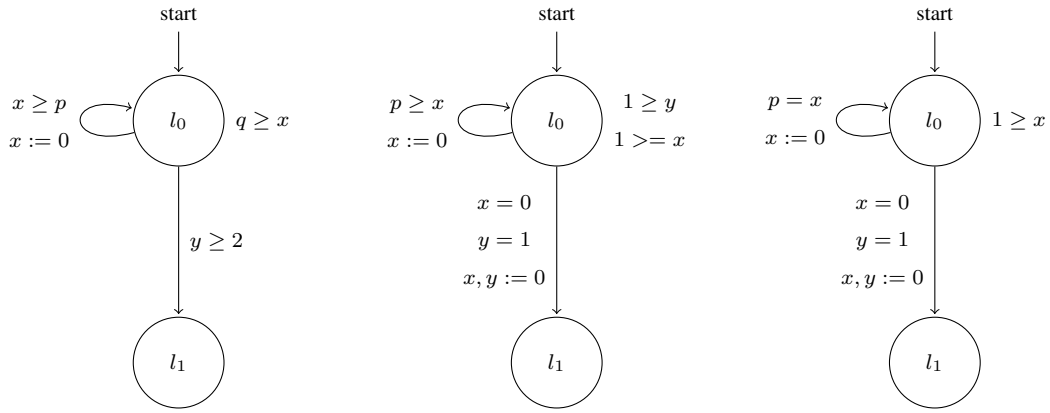

Figure A.3: *JLR15_Fig6* (left), *Synth_int01* (middle) and *Synth_InvN* (right) PTAs from the Imitator benchmark library. In all, l_1 is marked as the unsafe or target location and the upper bound for parameters are set to 10.

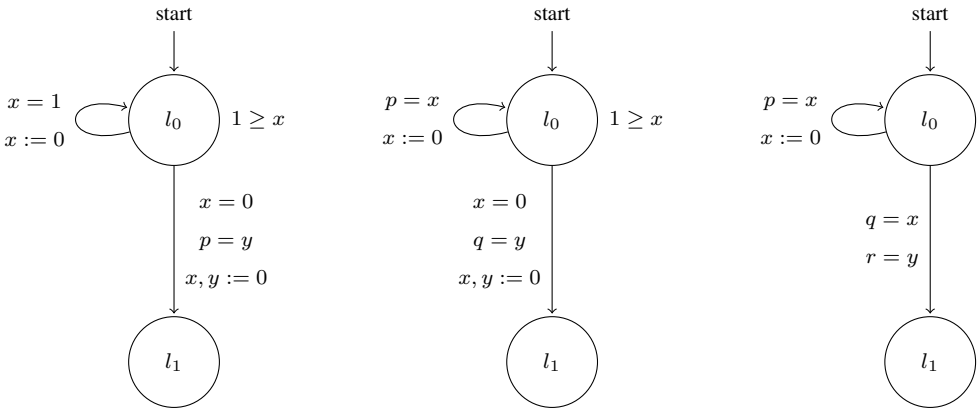

Figure A.4: *Synth_N* (left), *Synth_pN* (middle) and *Synth_pNplusq* (right) PTAs from the Imitator benchmark library. In all, l_1 is marked as the unsafe or target location and the upper bound for parameters are set to 10.

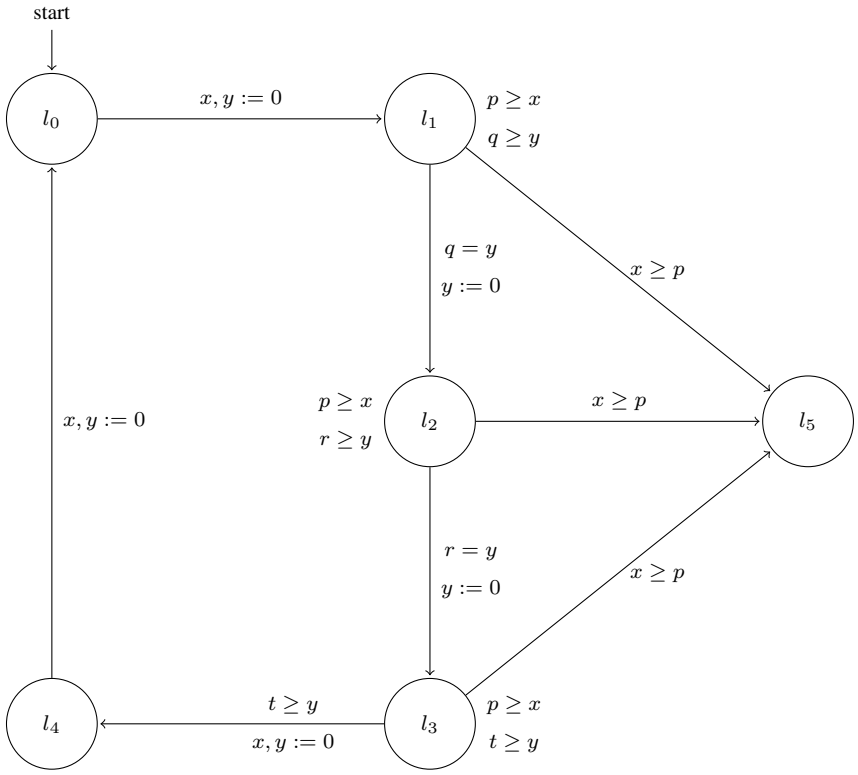

Figure A.5: *NuclearPlant* PTA from the Imitator benchmark library. l_5 is marked as the unsafe or target location and the upper bound for parameters are set to 10.

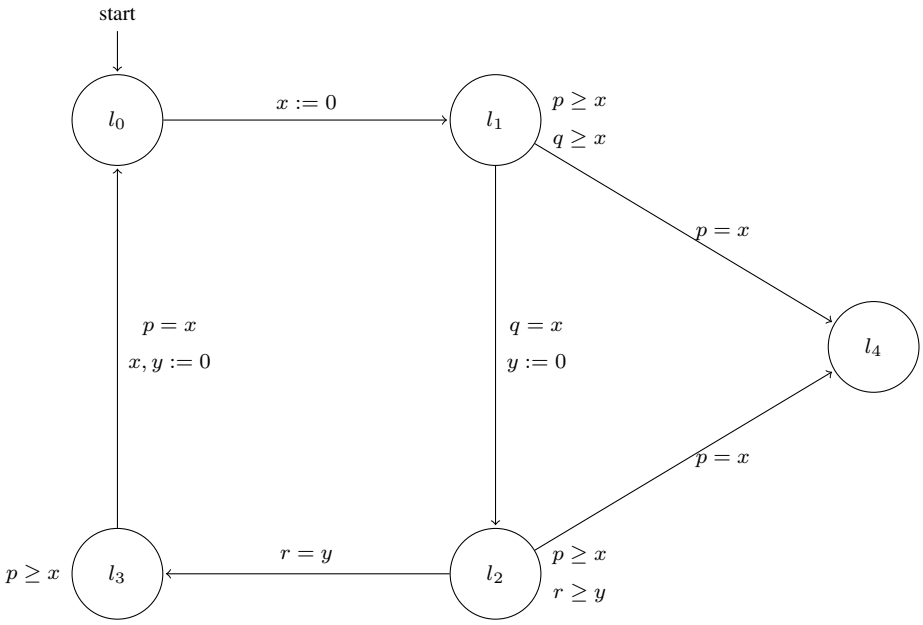

Figure A.6: *Train1PTA* PTA from the Imitator benchmark library. l_4 is marked as the unsafe or target location and the upper bound for parameters are set to 10.

CURRICULUM VITAE

Burkay Sucu

EDUCATION

Degree	Instution	Year of Graduation
PhD	Middle East Technical University Computer Engineering	2023
MS	University of Nevada, Reno Computer Science and Engineering	2013
BS	Middle East Technical University Computer Engineering	2010

PUBLICATIONS

1. B. Sucu and E. Folmer, "The blind driver challenge: steering using haptic cues," in Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility, pp. 3–10, 2014.

2. B. Sucu and E. Folmer, "Haptic interface for non-visual steering," in Proceedings of the 2013 international conference on Intelligent user interfaces, pp. 427–434, 2013.