January 20th, 2017
Paris, France

Parametric model checking timed automata under non-Zenoness assumption

Hoang Gia NGUYEN

Supervisors: Laure Petrucci and Étienne André

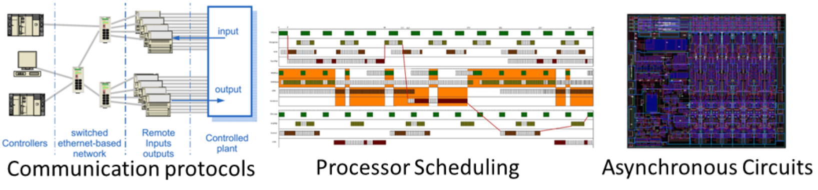
LIPN, Université Paris 13, Sorbonne Paris Cité, CNRS, France

Outline

Outline

Parametric Verification of Real-Time Systems

- Verification techniques used for critical systems, timed systems where changes of time value is vital! such as:

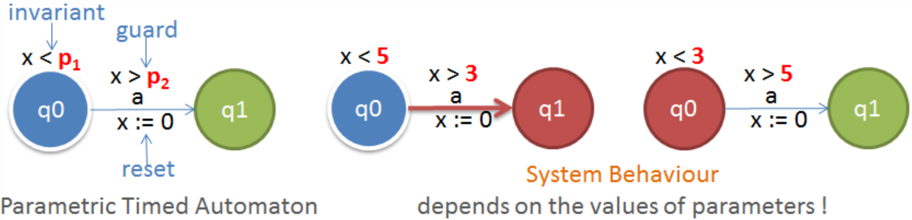


1. Systems incompletely specified, some timing delays may not be known yet, or may change
2. Verifying system for numerous values of constants requires a very long time, or even infinite

$\Rightarrow$ Use parameterised techniques, by using parameters instead of constants, then one can check many values at the same time

Parametric Timed Automata (PTA)

PTA is a formalism to model and verify concurrent real-time systems [Alur et al., 1993]



x: Clock

p: Parameters allow to represent unknown values (e.g. a transmission delay or a timeout)

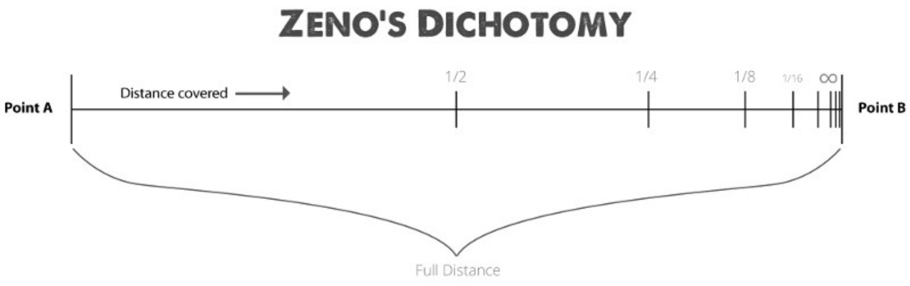
K_0: Initial parameter contraint (e.g. $p1 \leq p2$ or $p1 > p2$)

Outline

Zenoness Introduction

Definition

An infinite number of discrete actions in a finite time



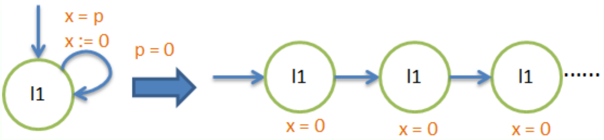
Get a half way of $A \rightarrow B$ ($\frac{1}{2}$), then get half the remaining distance $\frac{1}{2} \rightarrow B$ ($\frac{1}{4}$), then again and again $\rightarrow$ never reach B! (A and B can be the parameters).

$\Rightarrow$ Infeasible in reality!

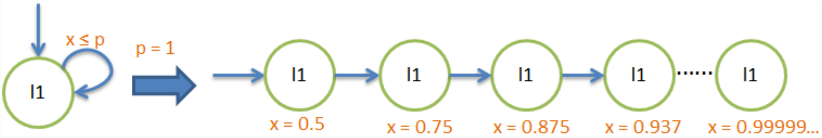
Zenoness in parametric timed model checking

2 types of Zeno run (infinite):

1. Run has a clock such that time cannot elapse



2. Run has a clock bounded by a parameter or a constant



Existing an infinite run in a finite time is not feasible!

Zenoness in parametric timed model checking (cont.)

Problem

1. Existing loop in product of Büchi automata (negated LTL formula), etc. Zeno run in counter-example is spurious
2. Zeno run cannot be checked directly on PTA model or its symbolic semantic!

$\Rightarrow$ Important to find and avoid Zeno loops in checking result!

Outline

CUB-TA Introduction

CUB Introduction

CUB stands for "Clock Upper Bound", an approach derived from the paper [Wang et al., 2015] for solving the non-Zenoness problem on Timed Safety Automata (TA)

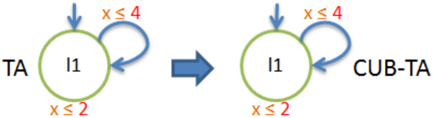
1. Zeno loops can be checked directly on CUB-TA's Zone Graph
2. More efficient than other current existing approaches
3. No need to introduce any new clock

⇒ We define a CUB approach for PTA

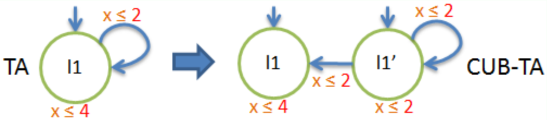
CUB-TA Introduction (cont.)

CUB-TA Definition

- A path is non-decreasing upper bound iff for each edge from location l to l' with guard g, for each clock x, the upper bound l_x is less than or equal to g_x and l'_x (if x is not reset)
- A TA $\mathcal{A}$ is a *CUB-TA*, iff every clock has a non-decreasing upper bound along any path before it is reset



TA containing a non-decreasing upper bound path example



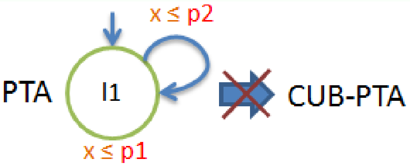
TA containing a decreasing upper bound path example

CUB-TA Transformation examples

CUB-PTA Introduction

CUB-PTA Definition

A PTA $\mathcal{A}$ is a *CUB-PTA*, iff there exists a constraint $\mathcal{A}.K_0$ on parameters that guarantees every clock has a non-decreasing upper bound along any path before it is reset, for all parameter valuations satisfying the initial constraint $\mathcal{A}.K_0$



There are *2 cases*:

$\mathcal{A}.\mathrm{K}_0 = \mathrm{p1} \leq \mathrm{p2}$: non-decreasing upper bound path! $\Rightarrow$ CUB-PTA

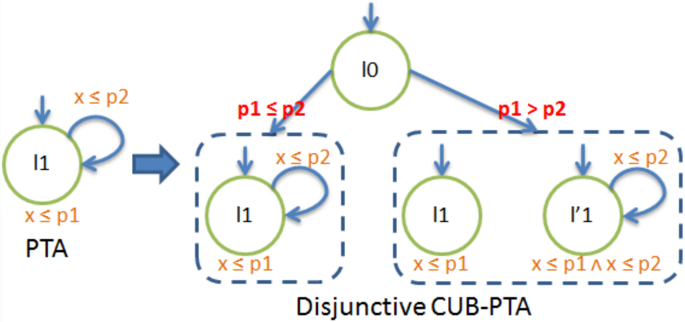
$\mathcal{A}.\mathrm{K}_0 = \mathrm{p1} > \mathrm{p2}$: decreasing upper bound path! $\Rightarrow$ not CUB-PTA

$\Rightarrow$ No transformation exists such that a CUB-PTA can cover all cases! But a list of CUB-PTAs can

CUB-PTA Introduction (cont.)

Disjunctive CUB-PTA Definition

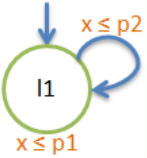
A *disjunctive CUB-PTA* is a list of *CUB-PTA*s



With a *CUB-PTA* or *disjunctive CUB-PTA*, we can synthesize parameter valuations of non-Zeno runs on its symbolic semantic Parametric Zone Graph - PZG (similar to Zone Graph of TA and not always finite).

CUB-PTA Detection

CUB-PTA detection aims at non-Zenoness synthesizing a partial or complete result without modification on the given model



$\mathcal{A}.K_0 = p1 \leq p2 \wedge \ p1 \leq p1$

$\Leftrightarrow$ CUB-PTA with $\mathcal{A}.K_0 = p1 \leq p2$

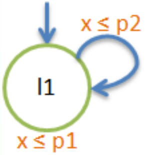
(Partial result)

Missing result: $\mathcal{A}.K_0 = p1 > p2$

Main idea

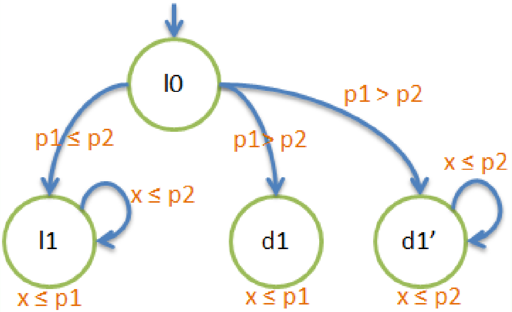
Given PTA $\mathcal{A}$, for each clock x on each edge with guard g from location l to l' we enforce a constraint with upper bound l_x less than or equal to g_x and l'_x (if x is not reset). If a conjunction of all constraints $\mathcal{A}.K_0$ contains some valuations then the given PTA is *CUB-PTA*

CUB-PTA Transformation



PTA

Transforms



Disjunctive CUB-PTA with
$\mathcal{A}.K_0 = p1 \leq p2$ or $p1 > p2$ (Complete result)

> An arbitrary PTA can be transformed into a *disjunctive CUB-PTA* (with a new initial location), while preserving the symbolic runs

CUB-PTA Transformation (cont.)

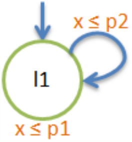
Main idea

Given a PTA $\mathcal{A}$:

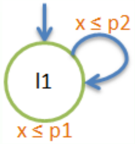
1. Infer all possible parameter relations $\mathcal{A}.K_0$s
2. Each copy of $\mathcal{A}$ will be transformed due to each $\mathcal{A}.K_0$ by:
    1. Splitting the location* into new locations with different upper bounds
    2. Copying all incoming and outgoing edges of old location to the new location
    3. Removing all decreasing upper bound edges
3. Add a new initial location connecting to all initial locations of the copies of $\mathcal{A}$

location*: a location containing an outgoing edge implies a decreasing upper bound

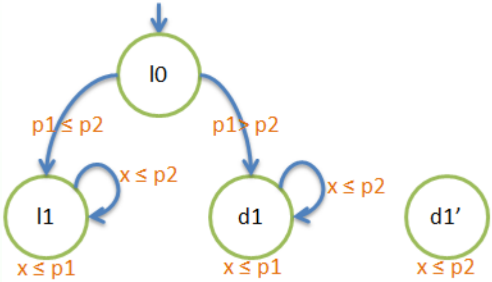
CUB-PTA Transformation Example



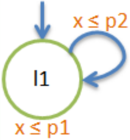
CUB-PTA Transformation Example



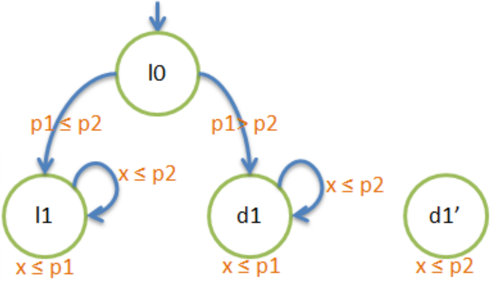
Infer all possible $\mathcal{A}.K_0$s on the fly.
For each $\mathcal{A}.K_0$, split the location* into different upper bounds



CUB-PTA Transformation Example

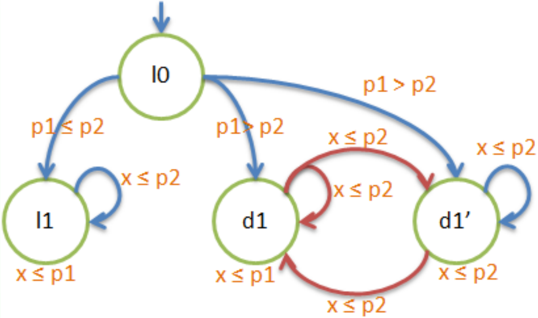


Infer all possible $\mathcal{A}.K_0$s on the fly.
For each $\mathcal{A}.K_0$, split the location* into different upper bounds



Copy all incoming and outgoing edges of old location to new location

Remove all decreasing upper bound edges



Non-Zenoness Parametric Model Checking

With CUB-PTA Parametric Non-Zenoness can be checked directly on the Parametric Zone Graph - PZG

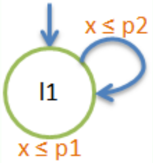
Main idea

A *CUB-PTA* $\mathcal{A}$ contains a non-Zeno run iff:

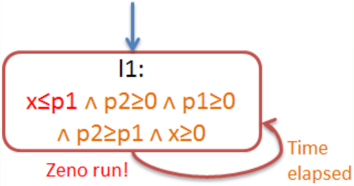
1. There exists parameter valuation such that $\text{PZG}(\mathcal{A})$ has a SCC containing an edge from location l to l' where time can elapse
2. For every clock x in $\mathcal{A}$, if x is bounded by a constant or a parameter for some location in the SCC, there exists an edge in the SCC where x is reset

SCC: Strongly Connected Component

Non-Zenoness Parametric Model Checking (cont.)



CUB-PTA
(Detection)



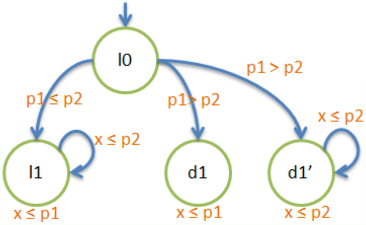
PZG of the CUB-PTA

Emptiness non-Zenoness check: False!

Approximation: Under-approximation

(no result is given for p1>p2)

Non-Zenoness Parametric Model-Checking (cont.)



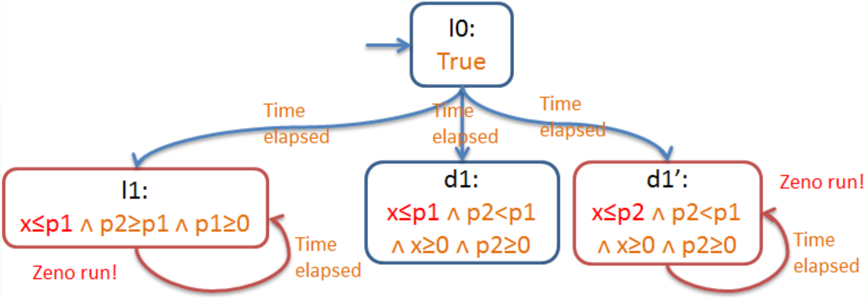
Disjunctive CUB-PTA

Emptiness non-Zeno check:

False!

Approximation:

Exact!



PZG of the disjunctive CUB-PTA

Outline

Implementation

- Implementation in IMITATOR [André, Fribourg, Kühne, Soulat, 2012]
  - About 3,000 lines of new OCaml code for our non-Zenoness parameter synthesis algorithm
  - Thank to the Parma Polyhedra Library (PPL) library for solving linear inequality systems

Experiments

Model				synthCycle			CUBdetect					CUBtrans				
Name	# X	# P	# L	time (s)	Result	Appr.	Detec time (s)	Total time (s)	CUB for	Result	Appr.	Trans time (s)	Total time (s)	#L CUB	Result	Appr.
CSMA/CD	3	3	28	TO	$\top$	invalid	0.013	0.013	$\bot$	-	-	0.300	TO	74	$\top$	exact
Fischer	2	4	13	TO	$\top$	invalid	0.003	0.003	$\bot$	-	-	0.012	TO	20	$\top$	exact
RCP	6	5	48	TO	Some	invalid	0.013	0.013	$\bot$	-	-	0.348	TO	71	$\bot$	under
WFAS	4	2	10	TO	Some 102%	invalid	0.009	0.009	$\bot$	-	-	0.246	1848	40	Some 100%	exact
AndOr	4	4	27	TO	Some 166%	invalid	0.012	0.012	$\bot$	-	-	0.059	TO	34	Some 100%	under
Flip-flop	5	2	52	0.058	$\bot$	exact	0.002	0.086	$\top$	$\bot$	exact	0.010	0.972	58	$\bot$	exact
Sched5	21	2	153	190	$\bot$	exact	0.051	0.051	$\bot$	-	-	1.180	TO	180	$\bot$	under
simop	8	2	46	TO	$\bot$	invalid	0.012	0.012	$\bot$	-	-	0.219	TO	81	$\bot$	under
train-gate	5	9	11	TO	$\bot$	invalid	0.000	TO	Some	$\bot$	under	0.059	TO	23	$\bot$	under
coffee	2	3	4	TO	Some 100%	invalid	0.000	TO	Some	Some 100%	under	0.012	TO	10	Some 100%	under
CUBPTA1	1	3	2	0.006	$\top$ 208%	over	0.000	0.015	Some	Some 69%	under	0.006	0.073	6	Some 100%	exact
JLR13	2	2	2	TO	$\bot$	invalid	0.000	TO	$\top$	$\bot$	under	0.000	TO	3	$\bot$	under

- synthCycle (without non-Zenoness assumption): Synthesizes all parameter valuations of loops
- CUBdetect: Detects a given PTA is CUB-PTA then synthesizes parameter valuations of non-Zeno runs
- CUBtrans: Transforms a given PTA into CUB-PTA then synthesizes parameter valuations of non-Zeno runs

Outline

Conclusions

Contributions:

- Proposed and implemented new Zeno-free parametric model synthesizing approaches in IMITATOR [André, Fribourg, Kühne, Soulat, 2012] tool
- Gave an overall view of our algorithms' performance and complexity, a set of case studies for non-Zenoness parametric model checking study

Paper submitted:

- Étienne André, Hoang Gia Nguyen, Laure Petrucci, Jun Sun Parametric model checking timed automata under non-Zenoness assumption

Future work:

- Implement other techniques such as yet to be defined parametric extensions of strongly non-Zeno TAs [Tripakis et al., 2005] or guessing zone graph [Herbreteau et al., 2012] could turn to be more efficient and should be investigated

Bibliography

References I

Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993).
Parametric real-time reasoning.
In *STOC*, pages 592–601. ACM.

André, É., Fribourg, L., Kühne, U., and Soulat, R. (2012).
IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.
In *FM*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer.

Herbreteau, F., Srivathsan, B., and Walukiewicz, I. (2012).
Efficient emptiness check for timed Büchi automata.
Formal Methods in System Design, 40(2):122–146.

Tripakis, S., Yovine, S., and Bouajjani, A. (2005).
Checking timed Büchi automata emptiness efficiently.
Formal Methods in System Design, 26(3):267–292.

Wang, T., Sun, J., Wang, X., Liu, Y., Si, Y., Dong, J. S., Yang, X., and Li, X. (2015).
A systematic study on explicit-state non-zenoness checking for timed automata.
IEEE Transactions on Software Engineering, 41(1):3–18.

Licensing

Source of the graphics used I

Title: Ocaml logo
Author: Amir Chaudhry
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: CC BY-SA 4.0

Title: IMITATOR logo (Typing Monkey)
Author: Kater Begemot
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: CC BY-SA 3.0

Title: PPL logo
Author: Unknown
Source: http://bugseng.com/files/ext/images/site/ppl_mm_8.png
License: GCC