NTU Seminar

August 15th, 2017
Singapore

Parametric model checking timed automata under non-Zenoness assumption

Étienne André
Joint work with: Nguyên Hoàng Gia, Laure Petrucci, and Jun Sun

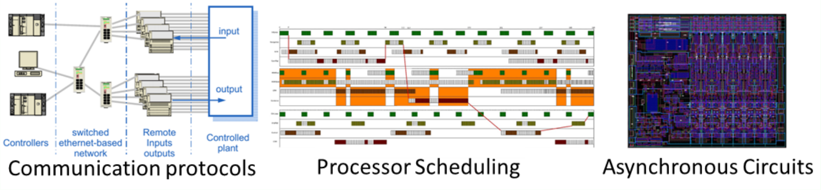
LIPN, Université Paris 13, CNRS, France

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:

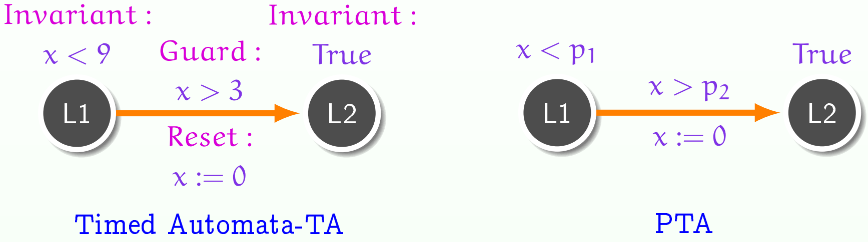


1. Systems incompletely specified, some timing delays may not be known yet, or may change
2. Verifying system for numerous values of constants requires a very long time, or even infinite

$\Rightarrow$ Use parameterised techniques, by using parameters instead of constants, then one can check many values at the same time, but also infer good valuations of these timing constants

Parametric Timed Automata (PTA)

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



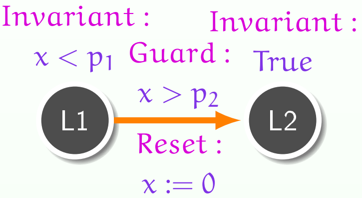
x: Clock

p: Parameters allow to represent unknown values

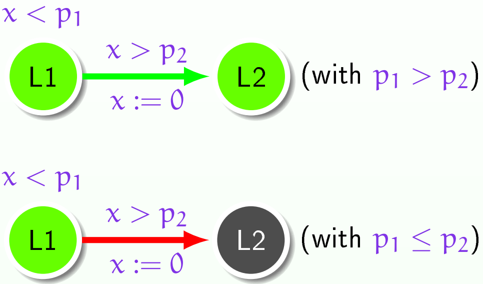
K_0: Initial parameter contraint (e.g. $p_1 \leq p_2$ or $p_2 > p_1$)

Parametric Timed Automata (PTA)

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



PTA



System Behaviour depends on the values of parameters

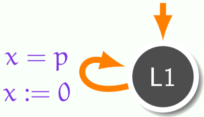
Outline

Zenoness in parametric timed model checking

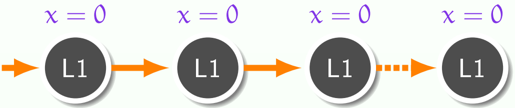
Zeno Run Definition

A Zeno run is a run with an infinite number of actions within a finite time.

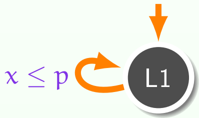
1. Run has a clock such that time cannot elapse



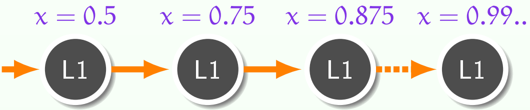
with
$p = 0$



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



with
$p = 1$



In fact, this run is Zeno for any value of p

$\Rightarrow$ Infeasible in practice, and should not be considered as a counter-example!

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)

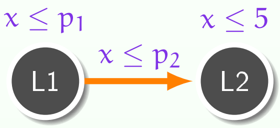
1. Zeno loops can be checked directly on CUB-TA's Zone Graph
2. More efficient than other current existing approaches
3. No need to introduce any new clock to the model

$\Rightarrow$ We define a CUB approach for PTA

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 $\mathcal{A}.K_0$



$\mathcal{A}.K_0 = p_1 \leq p_2 \wedge p_1 \leq 5$: non-decreasing upper bound path! $\Rightarrow$ CUB-PTA

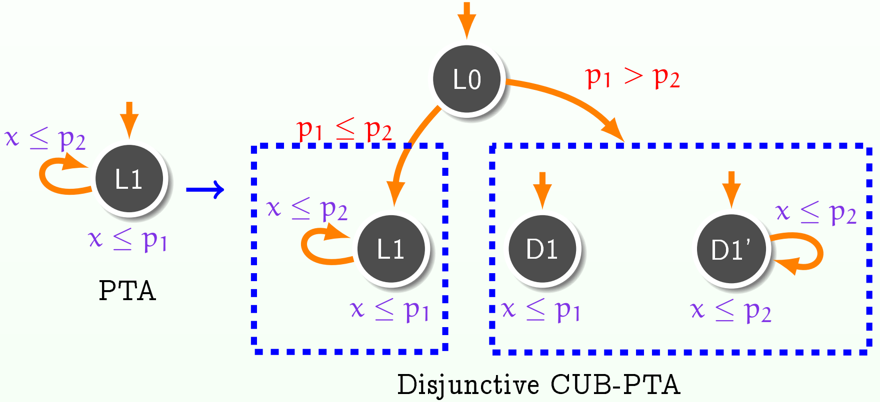
$\mathcal{A}.K_0 = p_1 > p_2 \vee p_1 > 5$: decreasing upper bound path! $\Rightarrow$ not CUB-PTA

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

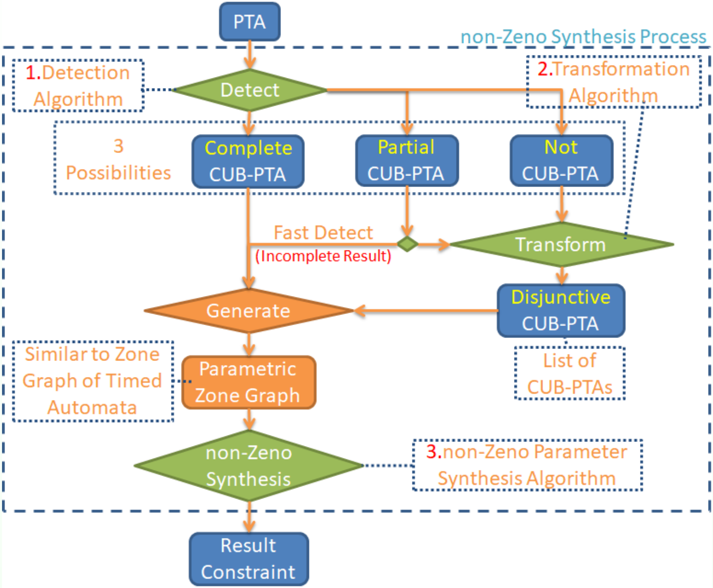
CUB-PTA Introduction (cont.)

Disjunctive CUB-PTA Definition

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



CUB-PTA Introduction (cont.)



CUB-PTA Detection

CUB-PTA detection aims at non-Zenoness synthesis of a partial or even complete result without modification on the given model.

$\mathcal{A}.K_0 =$

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 Detection

CUB-PTA detection aims at non-Zenoness synthesis of a partial or even complete result without modification on the given model.

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

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 Detection

CUB-PTA detection aims at non-Zenoness synthesis of a partial or even complete result without modification on the given model.

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

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 Detection

CUB-PTA detection aims at non-Zenoness synthesis of a partial or even complete result without modification on the given model.

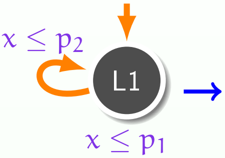
$\mathcal{A}.K_0 = p1 \leq p2 \wedge\ p1 \leq p1 \Leftrightarrow$ CUB-PTA with $\mathcal{A}.K_0 = p1 \leq p2$

Unchecked parameter valuations: $\mathcal{A}.K_0 = p1 > p2 \Rightarrow$ Partial CUB-PTA!

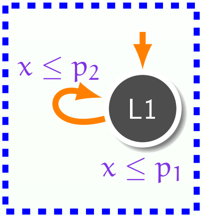
Main idea

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



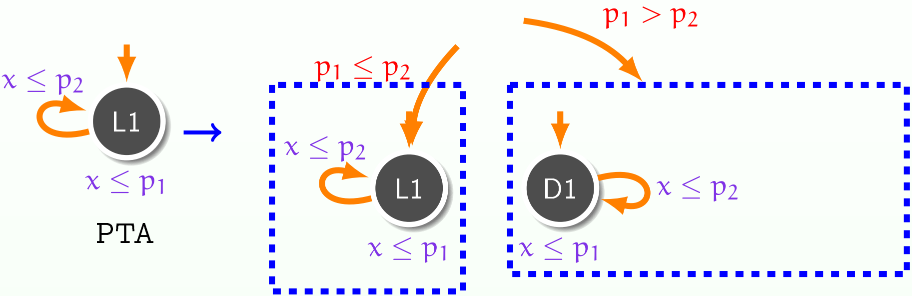
PTA



Main idea

Given a PTA $\mathcal{A}$

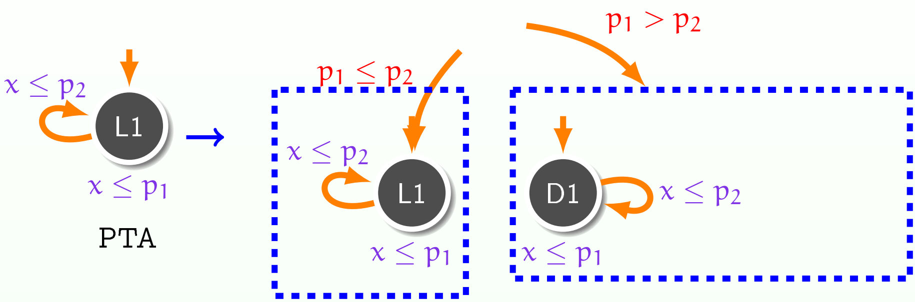
CUB-PTA Transformation



Main idea

Infer all possible parameter relations $\mathcal{A}.K_0$s

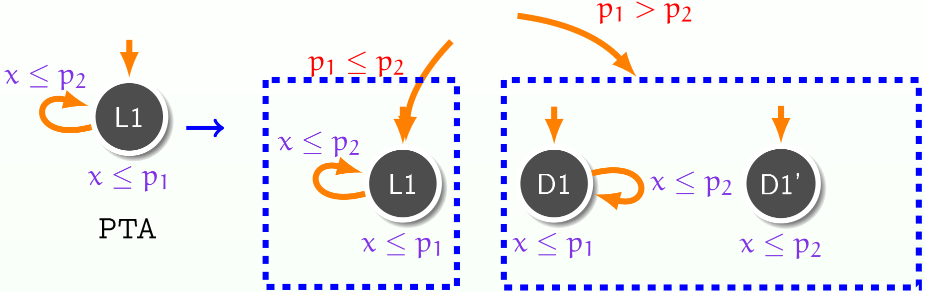
CUB-PTA Transformation



Main idea

Each copy of $\mathcal{A}$ will be transformed for each $\mathcal{A}.K_0$ by:

CUB-PTA Transformation

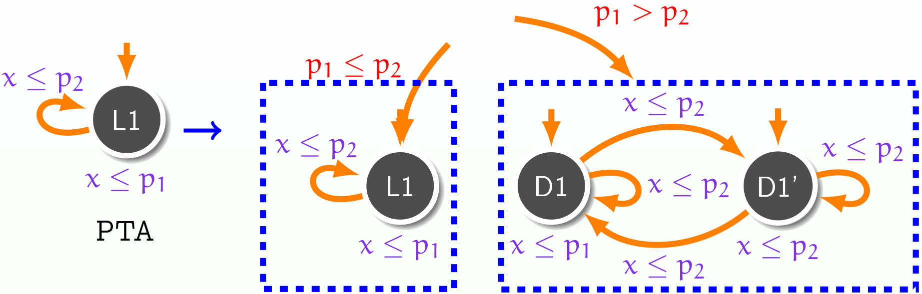


Main idea

Splitting the location* into new locations with different upper bounds

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

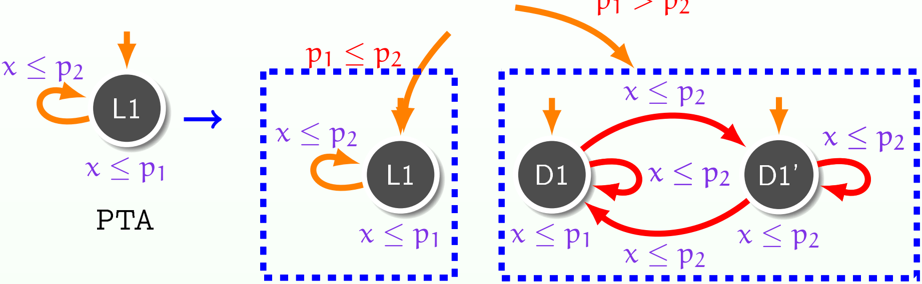
CUB-PTA Transformation



Main idea

Copying all incoming and outgoing edges of of the old location to the new location

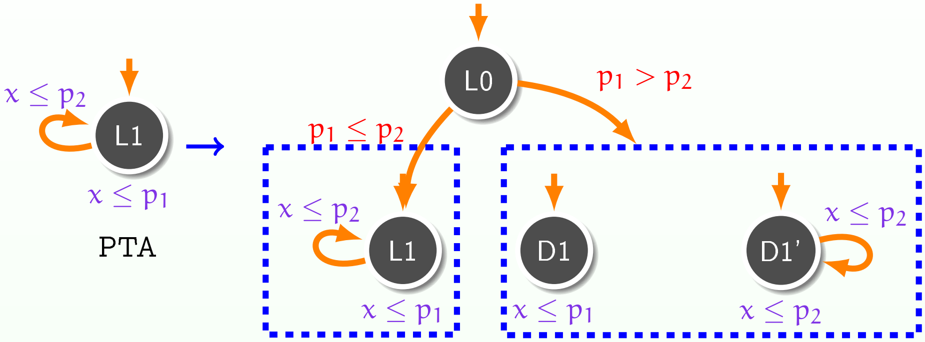
CUB-PTA Transformation



Main idea

Removing all decreasing upper bound edges

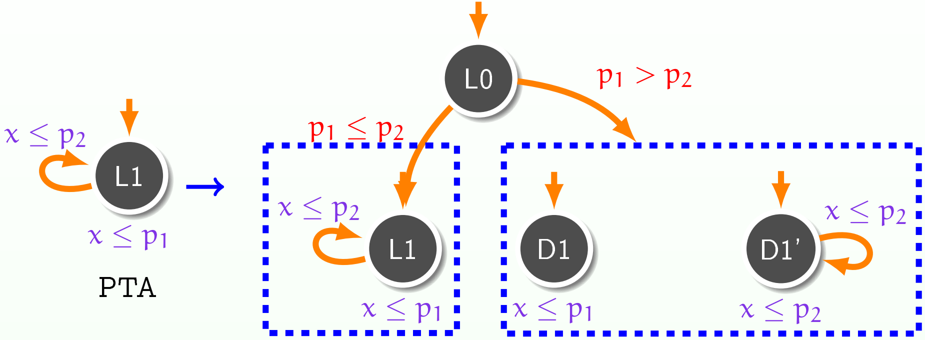
CUB-PTA Transformation



Main idea

Add a new initial location connecting to all initial locations of the copies of $\mathcal{A}$

CUB-PTA Transformation

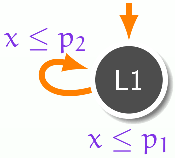


Disjunctive CUB-PTA

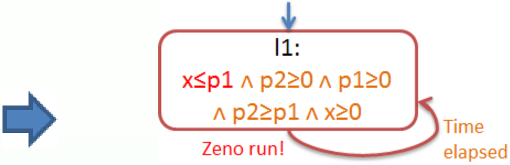
with $\mathcal{A}.K_0 = p1 \leq p2$ or $p1 > p2$ (Complete result)

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

Non-Zenoness Parametric Model Checking



CUB-PTA
(Detection)



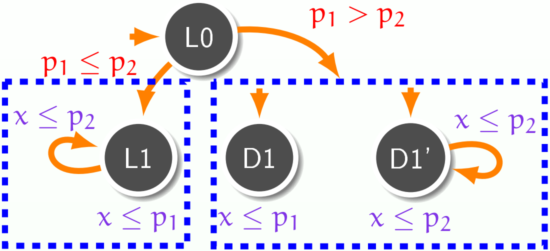
PZG of the CUB-PTA
Emptiness non-Zeno check: False!

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

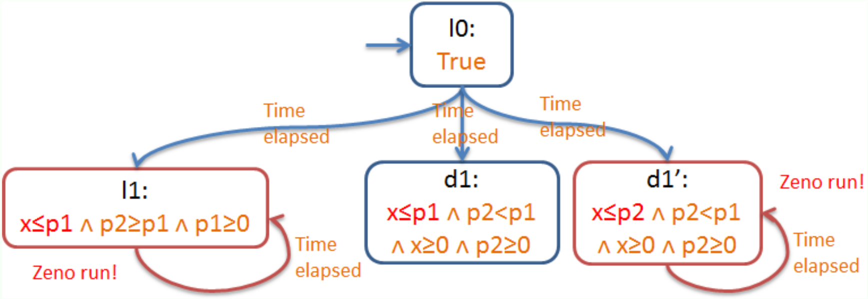
1. There exists parameter valuation such that $PZG(\mathcal{A})$ has a SCC containing an edge from location l to l' where time can elapse
2. For every clock x in $\mathcal{A}$, if x is bounded by a constant or a parameter for some location in the SCC, there exists an edge in the SCC where x is reset

SCC: Strongly Connected Component

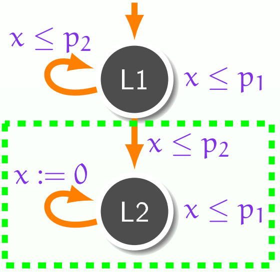
Non-Zenoness Parametric Model-Checking



Emptiness non-Zeno check: False!

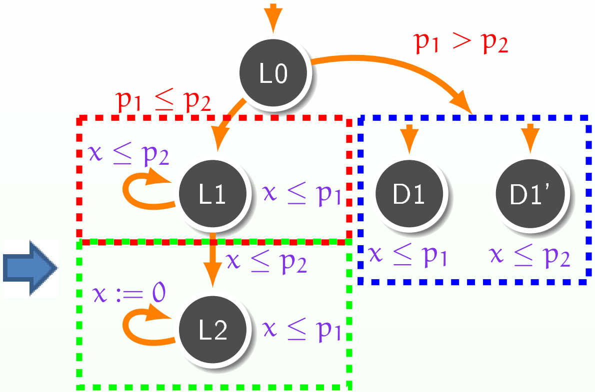


Non-Zenoness Parametric Model Checking



Other PTA example
(with non-Zeno run)

Emptiness non-Zeno check: $p_2 \geq p_1$



Disjuctive CUB-PTA
(Contaning non-Zeno run)

Emptiness non-Zeno check: $p_2 \geq p_1$

Outline

Implementation

- Implementation in IMITATOR [André, Fribourg, Kühne, Soulat, 2012] [1]
  - 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

[1] http://www.imitator.fr/

Experiments

Model				synthCycle		CUBdetect			CUBtransform		
Name	# X	# P	# L	Result	Appr.	CUB for	Result	Appr.	#L CUB	Result	Appr.
CSMA/CD	3	3	28	⊤	invalid?	⊥	-	-	74	⊤	exact
Fischer	2	4	13	⊤	invalid?	⊥	-	-	20	⊤	exact
RCP	6	5	48	Some	invalid?	⊥	-	-	71	⊥	under
WFAS	4	2	10	Some 102%	invalid?	⊥	-	-	40	Some 100%	exact
AndOr	4	4	27	Some 166%	invalid?	⊥	-	-	34	Some 100%	under
Flip-flop	5	2	52	⊥	exact	⊤	⊥	exact	58	⊥	exact
Sched5	21	2	153	⊥	exact	⊥	-	-	180	⊥	under
simop	8	2	46	⊥	invalid?	⊥	-	-	81	⊥	under
train-gate	5	9	11	⊥	invalid?	Some	⊥	under	23	⊥	under
coffee	2	3	4	Some 100%	invalid?	Some	Some 100%	under	10	Some 100%	under
CUBPTA1	1	3	2	⊤ 208%	over	Some	Some 69%	under	6	Some 100%	exact
JLR13	2	2	2	⊥	invalid?	⊤	⊥	under	3	⊥	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

Experiments

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

- synthCycle: almost never terminates, and its result (under-approx of an over-approx) cannot be kept
- CUBdetect: is not very interesting
- CUBtrans: sometimes gives an exact result, sometimes an under-approx result, sometimes nothing

Outline

Conclusions

Contributions:

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

Future work:

- Implement other techniques such as yet to be defined parametric extensions of:
  - Strongly non-Zeno TAs [Tripakis et al., 2005]
  - Guessing zone graph [Herbreteau et al., 2012]

  They 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