PhD defense

October 3rd, 2023
Nancy, France

Theoretical and algorithmic contributions to the analysis of safety and security properties in timed systems under uncertainty

Dylan Marinho

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

Reviewers:	Patricia Bouyer-Decitre Thierry Jéron		
Examiners:	Véronique Cortier Thao Dang Frédéric Herbreteau Swen Jacobs	Supervisors:	Étienne André Stephan Merz

Motivation

- Real-time systems:
  - Not only the functional correctness but also the time to answer is important

Motivation

- Critical Real-time systems:
  - Not only the functional correctness but also the time to answer is important
  - Failures (in correctness or timing) may result in dramatic consequences

Motivation

- Critical Real-time systems:
  - Not only the functional correctness but also the time to answer is important
  - Failures (in correctness or timing) may result in dramatic consequences

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attacks
  - Electromagnetic attacks
  - Power attacks
  - Acoustic attacks
  - Timing attacks
  - Temperature attacks
  - etc.

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attacks
  - Electromagnetic attacks
  - Power attacks
  - Acoustic attacks
  - Timing attacks
  - Temperature attacks
  - etc.
- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] http://home.xnet.com/~warinner/pizzacites.html

General context: side-channel attacks

- Threats to a system using non-algorithmic weaknesses
  - Cache attacks
  - Electromagnetic attacks
  - Power attacks
  - Acoustic attacks
  - Timing attacks
  - Temperature attacks
  - etc.
- Example
  - Number of pizzas (and order time) ordered by the white house prior to major war announcements [1]

[1] http://home.xnet.com/~warinner/pizzacites.html

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time:

A simple example of timing attack

```
# input pwd      : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: ϵ

A simple example of timing attack

```
# input pwd     : Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: $\epsilon + \epsilon$

A simple example of timing attack

```
# input pwd     :  Real  password
# input attempt:  Tentative  password
for  i  = 0  to  min(len(pwd),  len(attempt)) − 1  do
     if  pwd[i]  ≠  attempt[i]  then
          return  false
done
return  true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: $\epsilon + \epsilon + \epsilon$

A simple example of timing attack

```
# input pwd      :  Real password
# input attempt: Tentative password
for i = 0 to min(len(pwd), len(attempt)) − 1 do
    if pwd[i] ≠ attempt[i] then
        return false
done
return true
```

pwd	c	h	i	c	k	e	n
attempt	c	h	e	e	s	e	

Execution time: $\epsilon + \epsilon + \epsilon$

- Problem: The execution time is proportional to the number of consecutive correct characters from the beginning of `attempt`

Timing attacks

- Principle: deduce private information from timing data (execution time)

Issues:

- May depend on the implementation (or, even worse, be introduced by the compiler)
- A relatively trivial solution: make the program last always its maximum execution time
  Drawback: loss of efficiency

$\rightsquigarrow$ Non-trivial problem

Detection

Need to detect timing-leak vulnerabilities

Detection

Need to detect timing-leak vulnerabilities

We want formal guarantees → formal methods

- Various methods:
  - Abstract interpretation
  - Static analysis
  - Model checking
  - Theorem proving

Detection

Need to detect timing-leak vulnerabilities

We want formal guarantees $\rightarrow$ formal methods

- Various methods:
  - Abstract interpretation
  - Static analysis
  - Model checking
  - Theorem proving

Methodology

A program

A specification

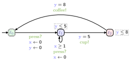
“The program must be secure”

Methodology

A program

→

A model



A specification

"The program must be secure"

→

A property

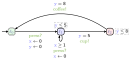
"Is the program secure?"

Methodology

A program

→

A model



A specification

“The program must be secure”

→

A property

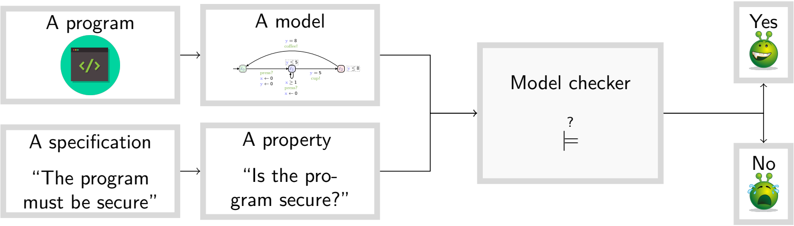
“Is the program secure?”

→

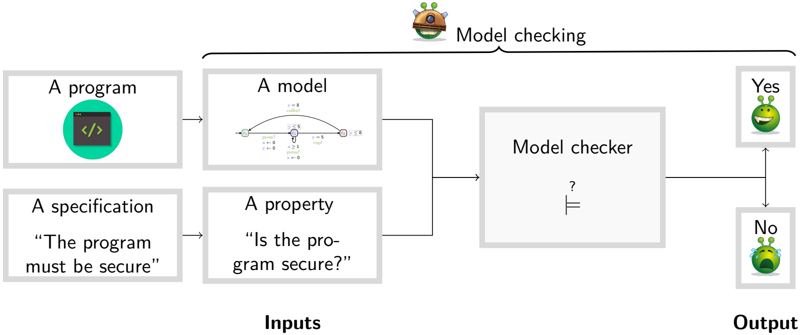
Model checker

$\stackrel{?}{\models}$

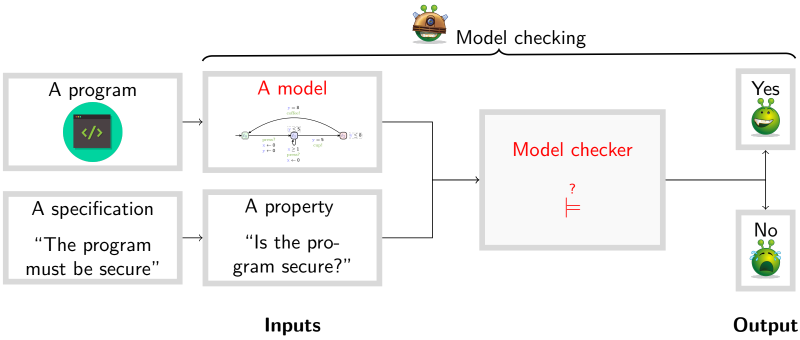
Methodology



Methodology



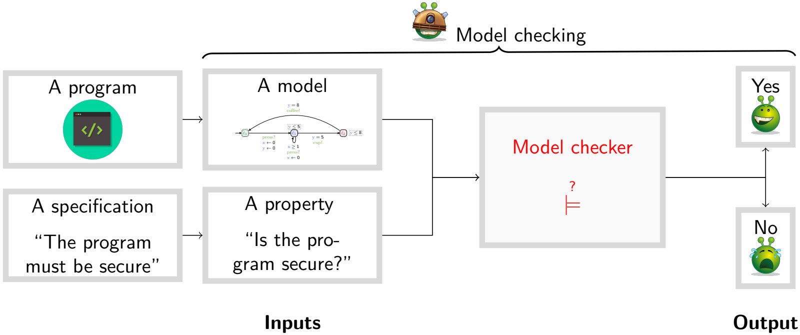
Outline



Outline

1. Preliminaries: Timed model checking

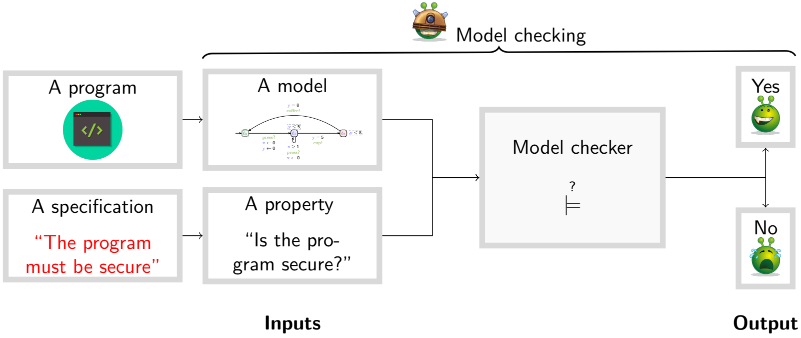
Outline



Outline

1. Preliminaries: Timed model checking
2. Contribution: Efficient verification (Manuscript, Part I)

Outline



Outline

1. Preliminaries: Timed model checking
2. Contribution: Efficient verification (Manuscript, Part I)
3. Contribution: Execution-time opacity (Manuscript, Part II)

Outline

Preliminaries: (Parametric) Timed model checking

Contribution: Efficient verification in Parametric Timed Automata

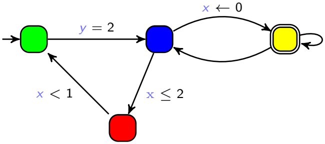
Contribution: Execution-time opacity

Conclusion & Perspectives

Outline

- Preliminaries: (Parametric) Timed model checking
  - Timed model checking and Timed automata
  - Parametric timed model checking and Parametric timed automata
- Contribution: Efficient verification in Parametric Timed Automata
- Contribution: Execution-time opacity
- Conclusion & Perspectives

Timed model checking

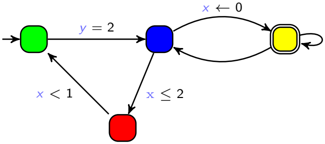


A model of the system

 is unreachable

A property to be satisfied

Timed model checking



A model of the system

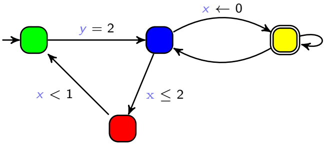
$\overset{?}{\models}$

 is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

Timed model checking



A model of the system

$\overset{?}{\models}$

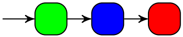
 is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

Yes

No

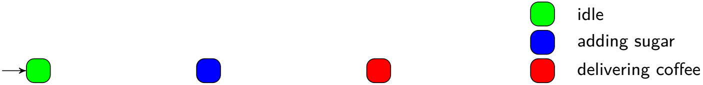


Counterexample

Timed automaton (TA)

[AD94]

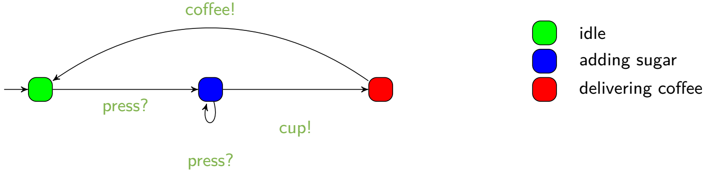
- Finite state automaton (sets of locations)



Timed automaton (TA)

[AD94]

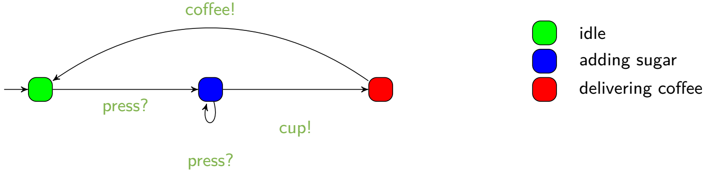
- Finite state automaton (sets of locations and actions)



Timed automaton (TA)

[AD94]

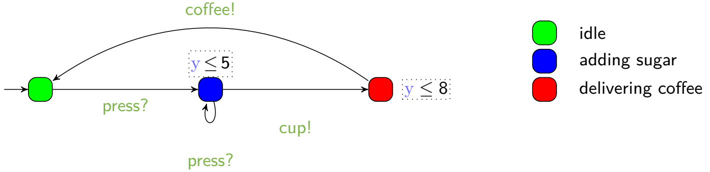
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate



Timed automaton (TA)

[AD94]

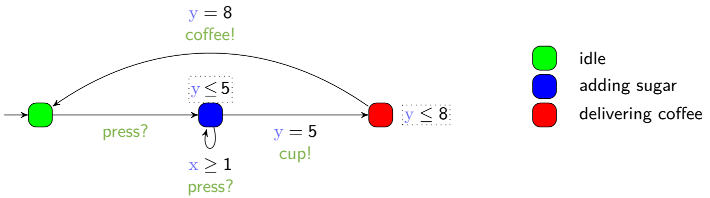
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants
- Features
  - Location invariant: property to be verified to stay at a location



Timed automaton (TA)

[AD94]

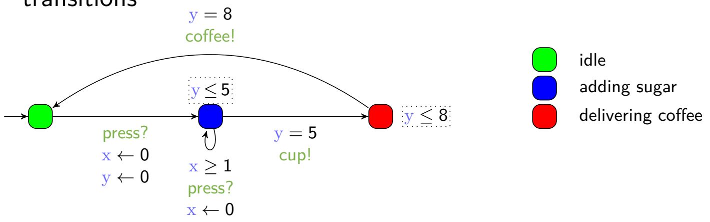
- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



Timed automaton (TA)

[AD94]

- Finite state automaton (sets of locations and actions) augmented with a set X of clocks
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards
- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions



Outline

Preliminaries: (Parametric) Timed model checking

Timed model checking and Timed automata

Parametric timed model checking and Parametric timed automata

Contribution: Efficient verification in Parametric Timed Automata

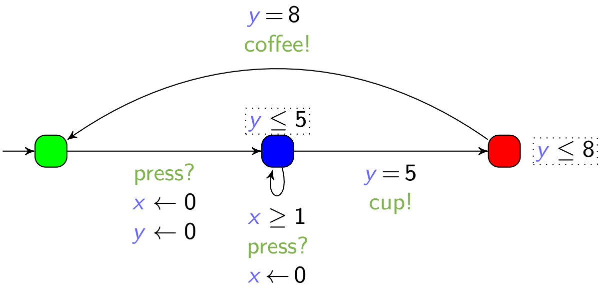
Contribution: Execution-time opacity

Conclusion & Perspectives

Timed Automaton (PTA)

[AHV93]

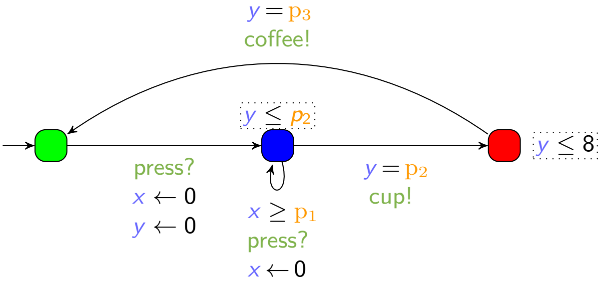
- Timed automaton (sets of locations, actions and clocks)



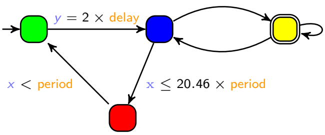
Parametric Timed Automaton (PTA)

[AHV93]

- Timed automaton (sets of locations, actions and clocks) augmented with a set P of parameters
  - Unknown constants compared to a clock in guards and invariants



timed model checking



A model of the system

$\overset{?}{\models}$

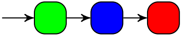
 is unreachable

A property to be satisfied

- Question: does the model of the system satisfy the property?

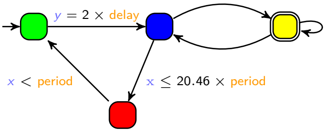
Yes

No



Counterexample

Parametric timed model checking



A model of the system

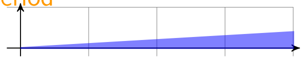
$\overset{?}{\models}$

 is unreachable

A property to be satisfied

- Question: for what values of the parameters does the model of the system satisfy the property?

Yes if...



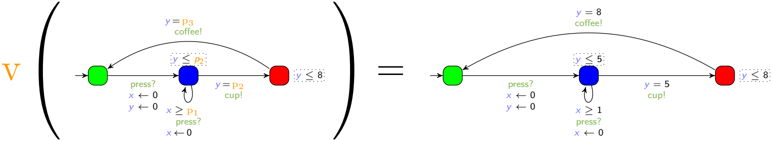
$2 \times \text{delay} > 20.46 \times \text{period}$

Valuation of a PTA = TA

- Given a PTA $\mathcal{P}$ and a parameter valuation v,
  $\mathrm{v}(\mathcal{P})$ is the TA where each parameter p is valuated by $\mathrm{v}(\mathrm{p})$

Valuation of a PTA = TA

- Given a PTA $\mathcal{P}$ and a parameter valuation v,
  $v(\mathcal{P})$ is the TA where each parameter p is valuated by $v(p)$



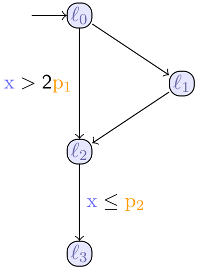
$$\text{with } v : \begin{cases} p_1 & \rightarrow & 1 \\ p_2 & \rightarrow & 5 \\ p_3 & \rightarrow & 8 \end{cases}$$

Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters

Parametric Zone Graph (PZG)

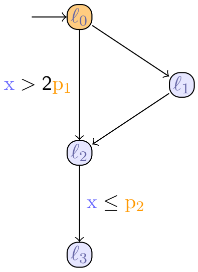
- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters



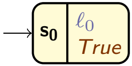
[HT15]

Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters

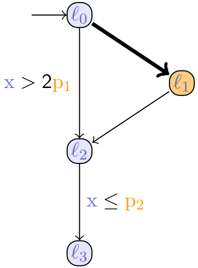


[HT15]

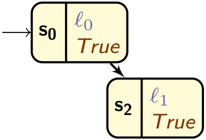


Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters

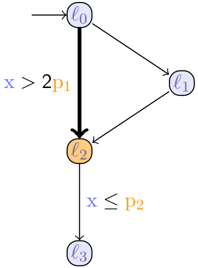


[HT15]

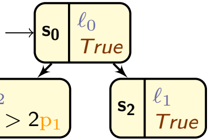


Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters

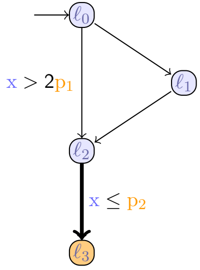


[HT15]

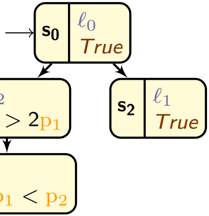


Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters

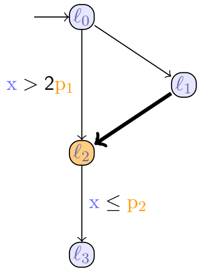


[HT15]

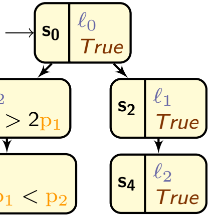


Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters

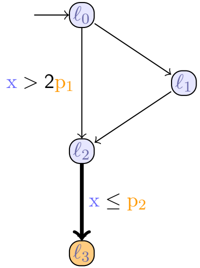


[HT15]

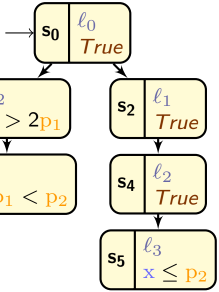


Parametric Zone Graph (PZG)

- Parametric extension zone graph of TAs
- **Symbolic state**: a pair with
  - a location
  - an attached parametric zone: a set of valuations defined by conjunctions of constraints over clocks and parameters



[HT15]



Outline

Contribution: Efficient verification of PTA models

- The verification of systems modeled by PTAs is difficult (undecidability, state-space explosion, ...)

Contribution: Efficient verification of PTA models

- The verification of systems modeled by PTAs is difficult (undecidability, state-space explosion, …)

Goal

- Efficient verification
- Reducing computation time
- Larger/more realistic case studies

$\Rightarrow$ Can we exhibit a more efficient algorithm?

Contribution: Efficient verification of PTA models

- The verification of systems modeled by PTAs is difficult (undecidability, state-space explosion, …)

Goal

- Efficient verification
- Reducing computation time
- Larger/more realistic case studies

$\Rightarrow$ Can we exhibit a more efficient algorithm?

Contributions

- Benchmark library [TAP21]
- Zone merging algorithm [FORMATS22]

Outline

Merging states

Definition

Two states (, $\mathbf{C}_1$) and (, $\mathbf{C}_2$) are mergeable if:

-  = 
- $\mathbf{C}_1 \cup \mathbf{C}_2$ is convex

Their merging is defined by (, $\mathbf{C}_1 \cup \mathbf{C}_2$)

Merging states

Definition

Two states (, $\mathbf{C}_1$) and (, $\mathbf{C}_2$) are mergeable if:

-  = 
- $\mathbf{C}_1 \cup \mathbf{C}_2$ is convex

Their merging is defined by (, $\mathbf{C}_1 \cup \mathbf{C}_2$)

State merging techniques were introduced:

- in TA [Dav05]
- in PTA for the Inverse Method only [AFS13]

Merging states

Definition

Two states (, $\mathbf{C}_1$) and (, $\mathbf{C}_2$) are mergeable if:

- $=$
- $\mathbf{C}_1 \cup \mathbf{C}_2$ is convex

Their merging is defined by (, $\mathbf{C}_1 \cup \mathbf{C}_2$)

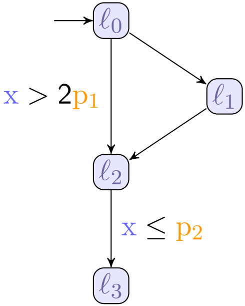
State merging techniques were introduced:

- in TA [Dav05]
- in PTA for the Inverse Method only [AFS13]

$\rightarrow$ Contribution: Extension to reachability properties

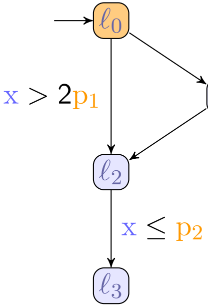
Constructing the PZG with merging algorithm

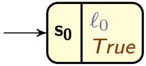
When constructing a new state, check if it can be merged



Constructing the PZG with merging algorithm

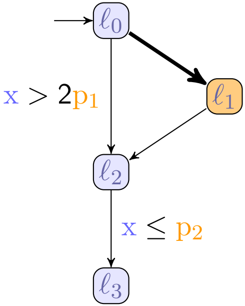
When constructing a new state, check if it can be merged

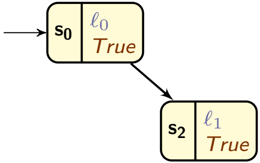




Constructing the PZG with merging algorithm

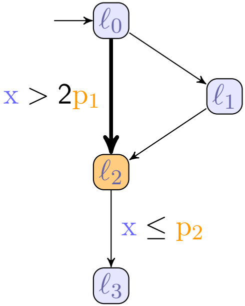
When constructing a new state, check if it can be merged

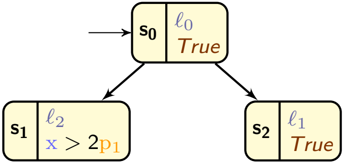




Constructing the PZG with merging algorithm

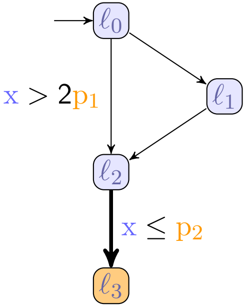
When constructing a new state, check if it can be merged

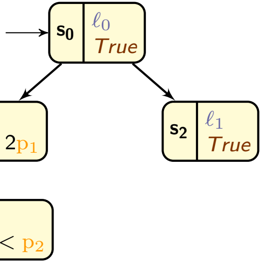




Constructing the PZG with merging algorithm

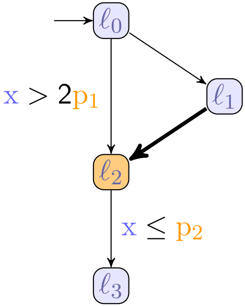
When constructing a new state, check if it can be merged

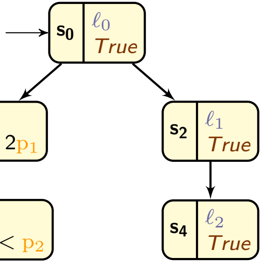




Constructing the PZG with merging algorithm

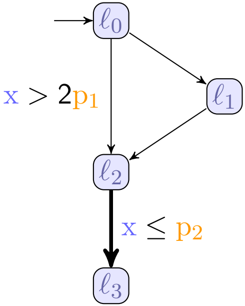
When constructing a new state, check if it can be merged

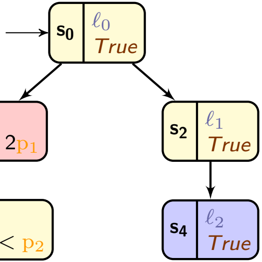




Constructing the PZG with merging algorithm

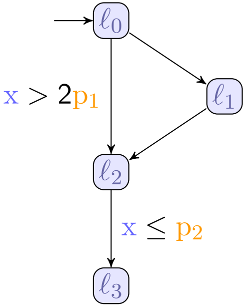
When constructing a new state, check if it can be merged

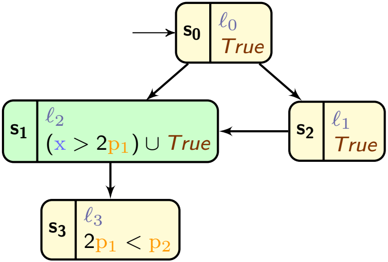




Constructing the PZG with merging algorithm

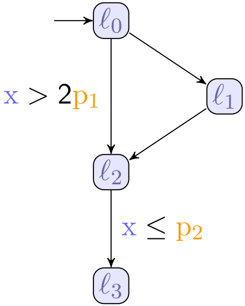
When constructing a new state, check if it can be merged

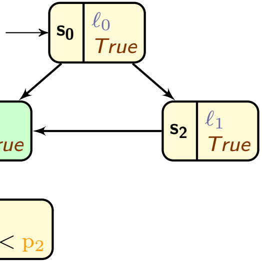




Constructing the PZG with merging algorithm

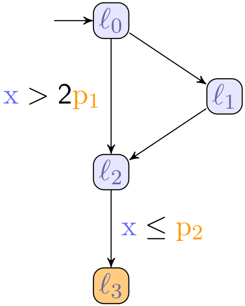
When constructing a new state, check if it can be merged

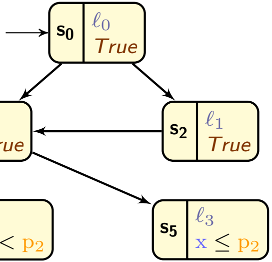




Constructing the PZG with merging algorithm

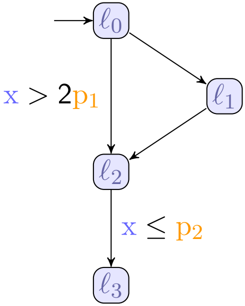
When constructing a new state, check if it can be merged

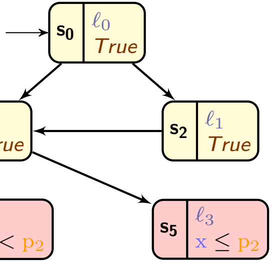




Constructing the PZG with merging algorithm

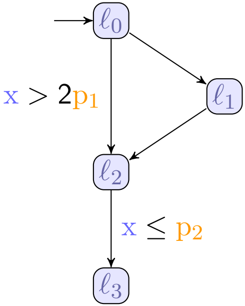
When constructing a new state, check if it can be merged

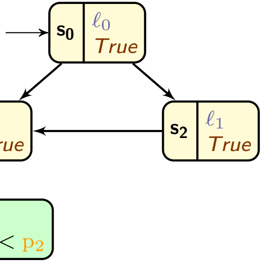




Constructing the PZG with merging algorithm

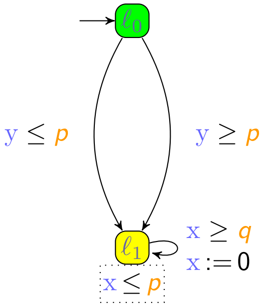
When constructing a new state, check if it can be merged

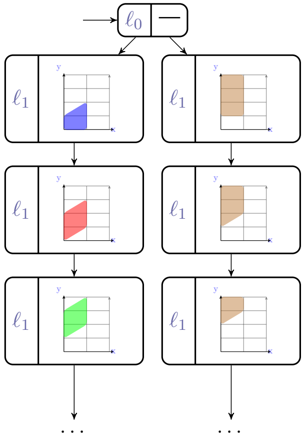




Merging can make difference for termination!

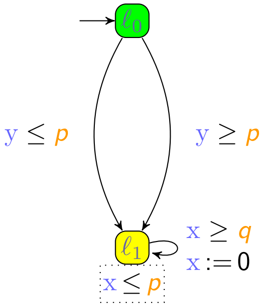
PZG without any heuristic

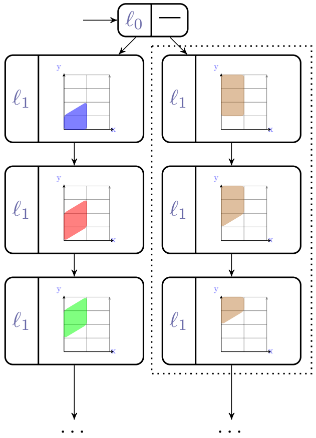




Merging can make difference for termination!

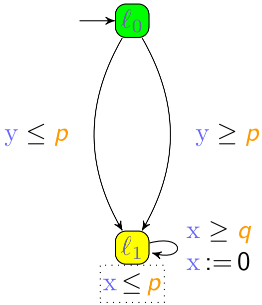
PZG with inclusion

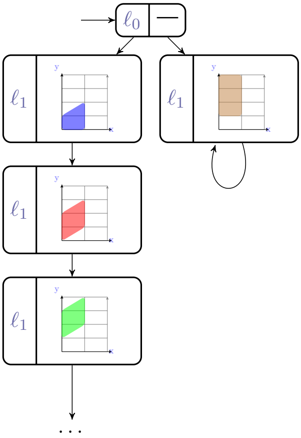




Merging can make difference for termination!

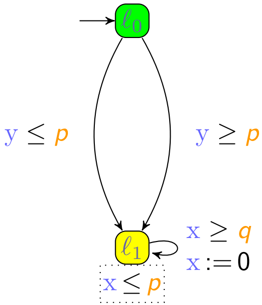
PZG with inclusion

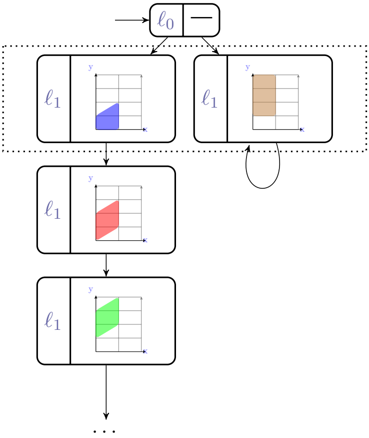




Merging can make difference for termination!

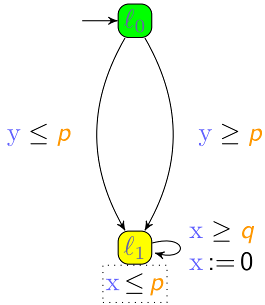
PZG with merging

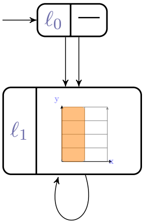




Merging can make difference for termination!

PZG with merging





Preservation of properties

Theorem

Merging states while computing the PZG is correct for reachability properties

Preservation of properties

Theorem

Merging states while computing the PZG is correct for reachability properties

But

- The test of convexity is (very) expensive
- The merge order can have consequences on the efficiency
- Performing an exhaustive zone merging is not efficient

Preservation of properties

Theorem

Merging states while computing the PZG is correct for reachability properties

But

- The test of convexity is (very) expensive
- The merge order can have consequences on the efficiency
- Performing an exhaustive zone merging is not efficient

Implementation

In a model-checker with heuristics

Heuristics for merging

What to merge with what? Queue, Visited, Ordered

Restart after merge?

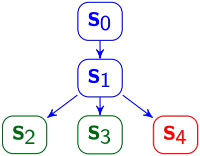
When to update the state space?

- After each merge
- After exploring the candidates list
- After exploring a level

How to update the state space?

- Reconstruction of the state-space
- *In situ*

Illustration of the merging options



- visited
- in the queue
- being processed
- after merge

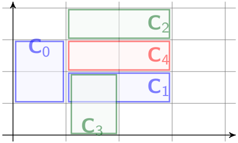
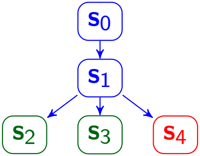
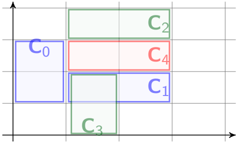


Illustration of the merging options



- visited
- in the queue
- being processed
- after merge



Merge with Queue

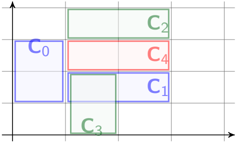
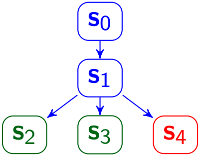
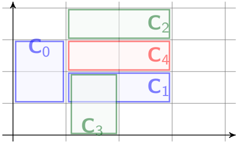


Illustration of the merging options



- □ visited
- □ in the queue
- □ being processed
- □ after merge



Merge with Queue

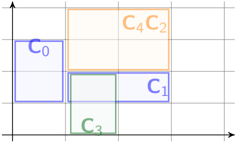
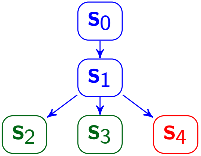
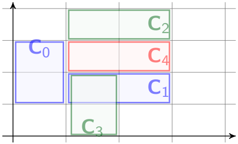


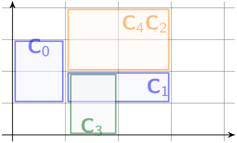
Illustration of the merging options



- visited
- in the queue
- being processed
- after merge



Merge with Queue



Merge with Visited

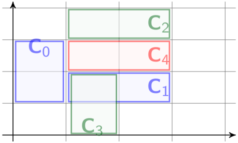
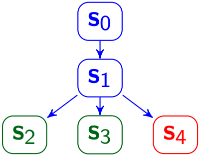
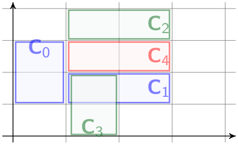


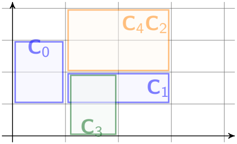
Illustration of the merging options



- visited
- in the queue
- being processed
- after merge



Merge with Queue



Merge with Visited

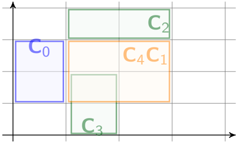


Illustration of the merging options

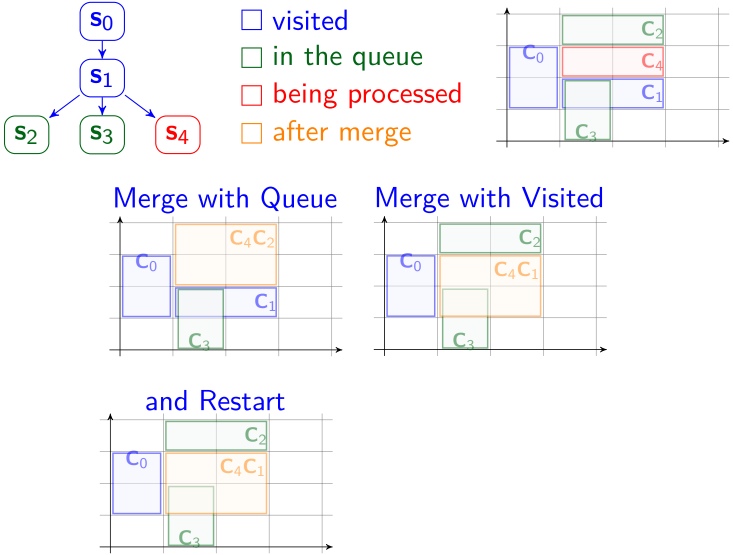
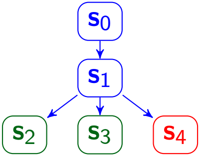
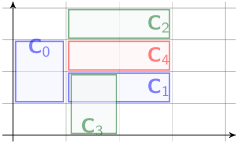


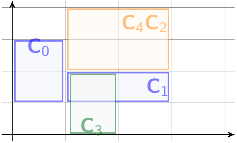
Illustration of the merging options



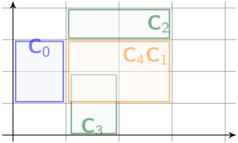
- □ visited
- □ in the queue
- □ being processed
- □ after merge



Merge with Queue



Merge with Visited



and Restart

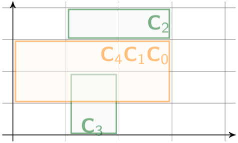
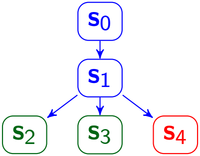
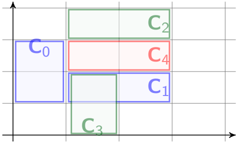


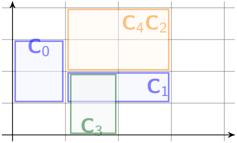
Illustration of the merging options



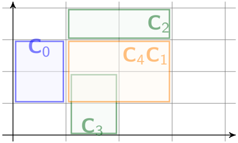
- ☐ visited
- ☐ in the queue
- ☐ being processed
- ☐ after merge



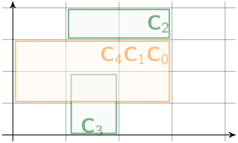
Merge with Queue



Merge with Visited



and Restart



Queue ; Visited

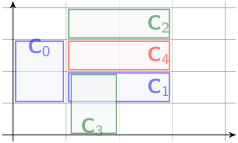
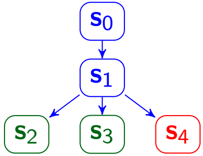
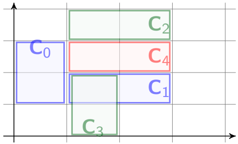


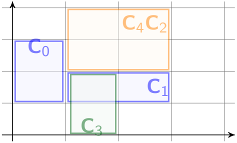
Illustration of the merging options



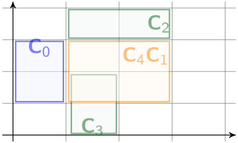
- visited
- in the queue
- being processed
- after merge



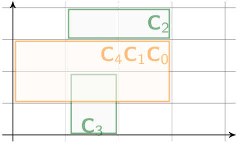
Merge with Queue



Merge with Visited



and Restart



Queue ; Visited

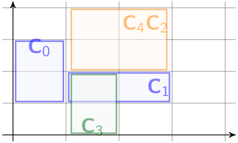
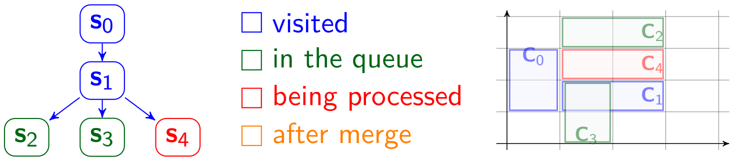
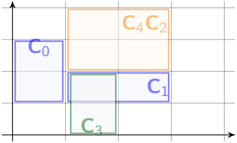


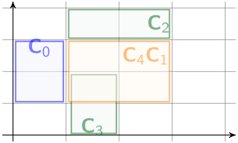
Illustration of the merging options



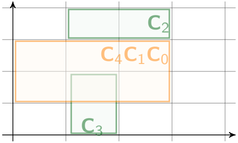
Merge with Queue



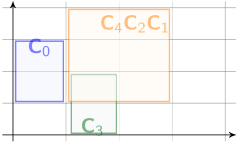
Merge with Visited



and Restart



Queue ; Visited



Outline

Preliminaries: (Parametric) Timed model checking

Contribution: Efficient verification in Parametric Timed Automata

- Merging Zones
- Experiments & Results

Contribution: Execution-time opacity

Conclusion & Perspectives

Experiments

- Comparison of all the combinations of heuristics
- Use of the IMITATOR library, restricted to reachability-based properties [TAP21]:
  - 124 executions (model, reachability property)
  - 42 executions perform at least one merge

[TAP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Parametric Timed Automata". In: *TAP* (2021). LNCS. Springer, 2021

The IMITATOR benchmark library

Overview of Chapter 4 of the manuscript

Library	Size		
Vers.	Bench.	Models	Prop.
1.0	34	80	122
2.0	56	119	216

Contribution

- More benchmarks (publications, industrial collaborations, …)
- Inclusion of liveness properties, unsolvable benchmarks
- Export to JANI
- Semantic information (computation time, expected result…)
- Published as long-term access

imitator.fr/library2
DOI 10.5281/zenodo.4730980

[TAP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Parametric Timed Automata". In: *TAP* (2021). LNCS. Springer, 2021

Comparing merging heuristics: Results

		Nomerge	RVMr	OQM
Time	# wins	24	22	**42**
	Avg (s)	10.0	4.56	**3.77**
	Avg (merge) (s)	18.8	5.57	**3.63**
	Avg (no merge) (s)	3.83	3.85	3.88
	Median (s)	1.39	1.14	**1.12**
States	# wins	0	**37**	16
	Avg	11443.08	**11064.37**	11120.79
	Avg (merge)	1512.02	**592.31**	729.33
	Median	2389.5	**604.5**	905.0

Gain of 62% of the average computation time

Outline

Contribution: Execution-time opacity

- How to detect timing-leak vulnerabilities?

Contribution: Execution-time opacity

- How to detect timing-leak vulnerabilities?

Goal

- Propose a formalization of the private information and attacker model
- Check whether a model is secure or not

Contribution: Execution-time opacity

- How to detect timing-leak vulnerabilities?

Goal

- Propose a formalization of the private information and attacker model
- Check whether a model is secure or not

Contributions

- ET-opacity definition, decidability results and experiments [TOSEM22]
- Expiring ET-opacity definition and decidability results [ICECCS23]
- Untimed control [FTSCS22]

Our attacker model

Attacker capabilities

- Has access to the model (white box)
- Can only observe the total execution time

Our attacker model

Attacker capabilities

- Has access to the model (white box)
- Can only observe the total execution time

Attacker goal

- Wants to deduce some private information based on these observations
  $\rightarrow$ visit of a private location

Outline

Preliminaries: (Parametric) Timed model checking

Contribution: Efficient verification in Parametric Timed Automata

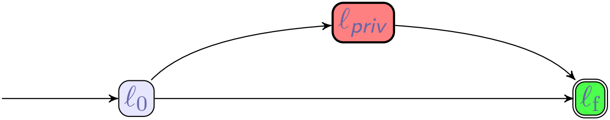
Contribution: Execution-time opacity
- ET-opacity problems in TAs
- ET-opacity problems in PTAs
- Expiring ET-opacity problems
- Untimed control

Conclusion & Perspectives

Formalization

Hypotheses: [AS19][TOSEM22]

- A start location ℓ_0 and an end location ℓ_f
- A special private location ℓ_{priv}

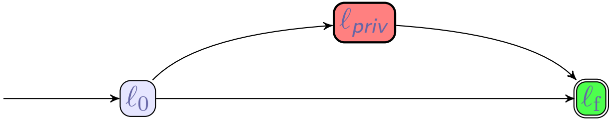


[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

Formalization

Hypotheses: [AS19][TOSEM22]

- A start location ℓ_0 and an end location ℓ_f
- A special private location ℓ_{priv}



Definition (execution-time opacity)

The system is ET-opaque for a duration d if there exist two runs to ℓ_f of duration d

1. one visiting ℓ_{priv}
2. one *not* visiting ℓ_{priv}

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

Three levels of ET-opacity

Existential (∃)

There exist a duration d and two runs of duration d,
one visiting ℓ_{priv},
one not visiting ℓ_{priv}

Three levels of ET-opacity

Existential (∃)

private durations $\cap$ public durations $\neq \emptyset$

Three levels of ET-opacity

Existential (∃)

private durations $\cap$ public durations $\neq \emptyset$

Weak

For all durations d,
There exists a run of duration d visiting ℓ_{priv}
$\Rightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Three levels of ET-opacity

Existential (∃)

private durations ∩ public durations $\neq \emptyset$

Weak

For all durations d,
There exists a run of duration d visiting ℓ_{priv}
$\Rightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Full

For all durations d,
There exists a run of duration d visiting ℓ_{priv}
$\Leftrightarrow$
There exists a run of duration d not visiting ℓ_{priv}

Three levels of ET-opacity

Existential ($\exists$)

private durations $\cap$ public durations $\neq \emptyset$

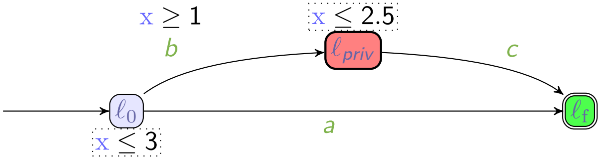
Weak

private durations $\subseteq$ public durations

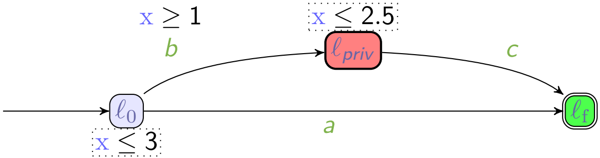
Full

private durations $=$ public durations

Example

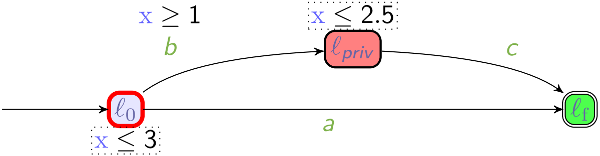


Example



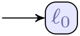
- There exist *(at least)* two runs of duration $d = 2$:

Example

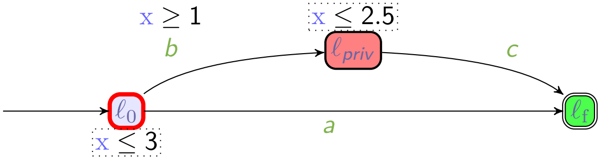


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

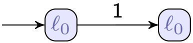


Example

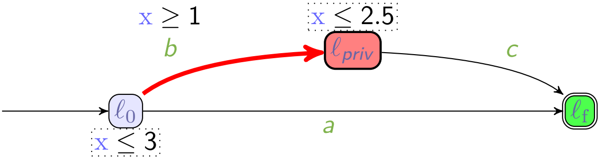


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

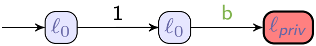


Example

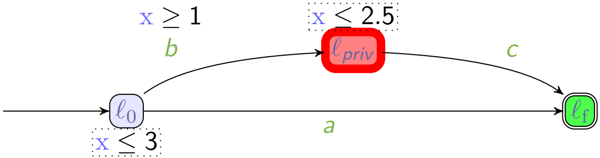


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

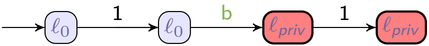


Example

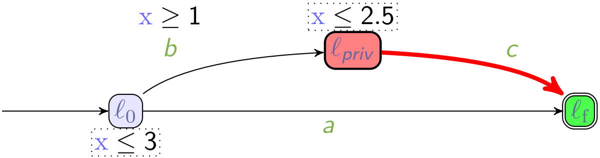


- There exist *(at least)* two runs of duration $d = 2$:

visiting ℓ_{priv}

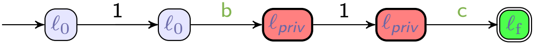


Example

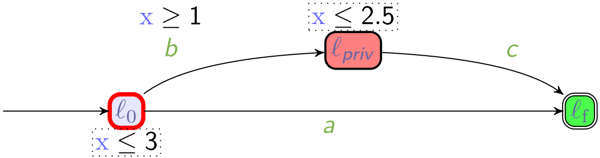


- There exist *(at least)* two runs of duration $\mathrm{d} = 2$:

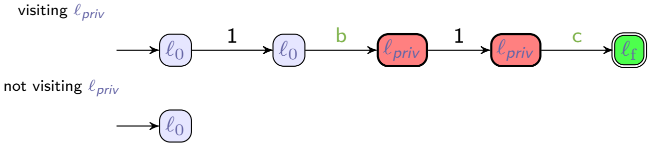
visiting ℓ_{priv}



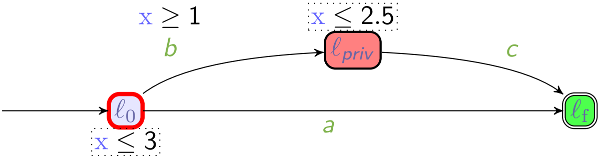
Example



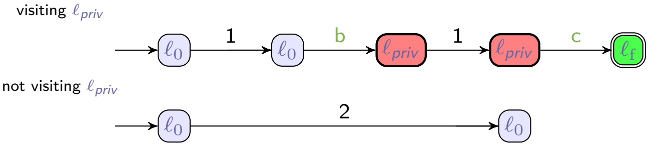
- There exist *(at least)* two runs of duration $d = 2$:



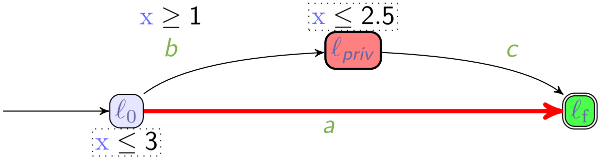
Example



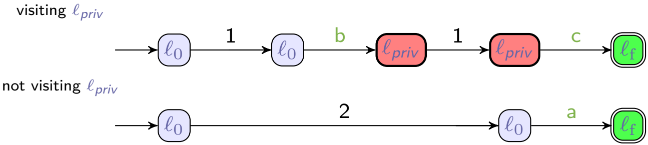
- There exist *(at least)* two runs of duration $d = 2$:



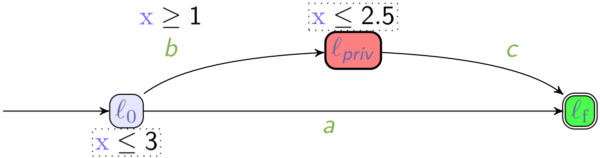
Example



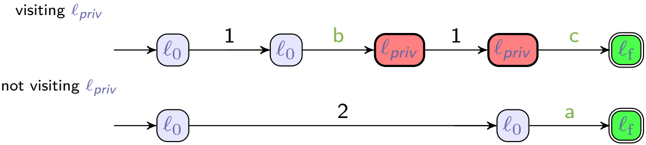
- There exist *(at least)* two runs of duration $d = 2$:



Example



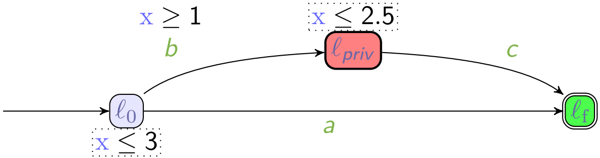
- There exist *(at least)* two runs of duration $\mathrm{d} = 2$:



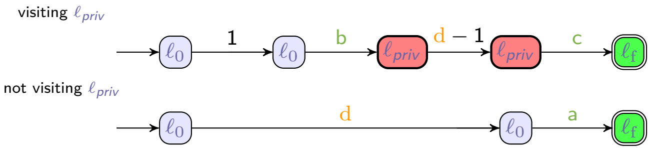
The system is ET-opaque for a duration $\mathrm{d} = 2$

The system is ∃-ET-opaque

Example



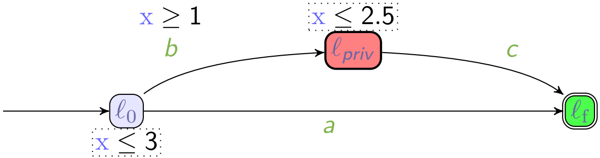
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$:



The system is ET-opaque for all durations in $[1, 2.5]$

The system is ∃-ET-opaque

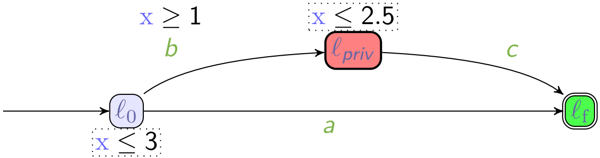
Example



- There exist *(at least)* two runs of duration d for all durations $\mathrm{d} \in [1, 2.5]$

The system is ∃-ET-opaque

Example

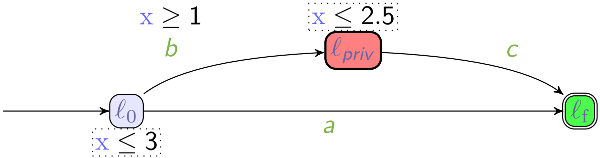


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$

Example

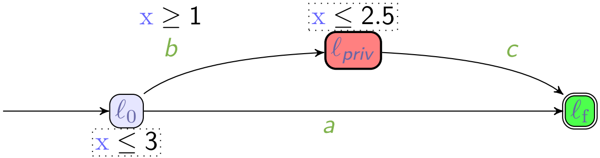


- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

Example



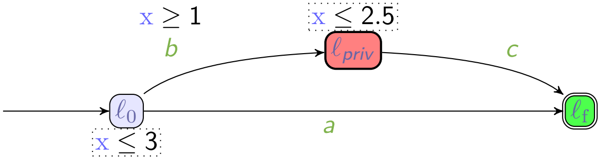
- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

The system is weakly ET-opaque

Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

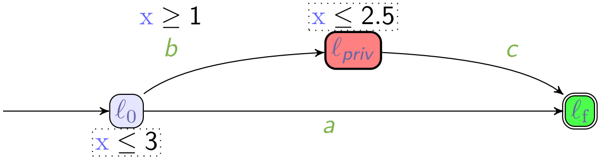
The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

The system is weakly ET-opaque

- private durations $\neq$ public durations

Example



- There exist *(at least)* two runs of duration d for all durations $d \in [1, 2.5]$

The system is ∃-ET-opaque

- private durations are $[1, 2.5]$
  public durations are $[0, 3]$
- private durations $\subseteq$ public durations

The system is weakly ET-opaque

- private durations $\neq$ public durations

The system is *not* fully ET-opaque

Outline

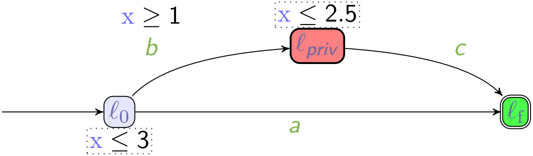
Preliminaries: (Parametric) Timed model checking

Contribution: Efficient verification in Parametric Timed Automata

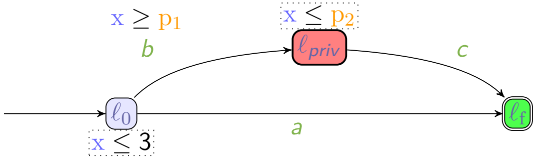
Contribution: Execution-time opacity
- ET-opacity problems in TAs
- ET-opacity problems in PTAs
- Expiring ET-opacity problems
- Untimed control

Conclusion & Perspectives

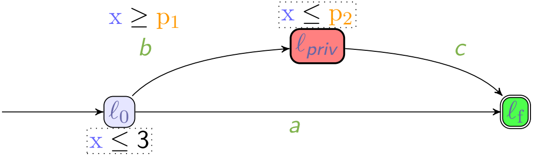
Example



Example

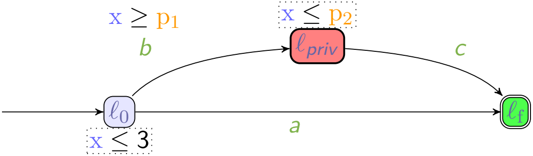


Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

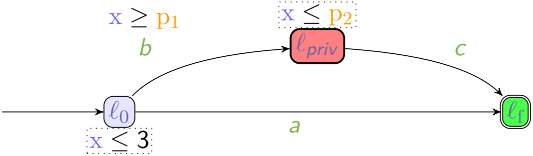
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	Private	Public	Answer
$p_1 = 1 \wedge p_2 = 2.5$			
∃			√
weak	$[1, 2.5]$	$[0, 3]$	√
full			×

Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	Private	Public	Answer
$p_1 = 1 \wedge p_2 = 2.5$			
$\exists$			$\surd$
weak	$[1, 2.5]$	$[0, 3]$	$\surd$
full			$\times$
$p_1 = 0 \wedge p_2 = 3$			
$\exists$			$\surd$
weak	$[0, 3]$	$[0, 3]$	$\surd$
full			$\surd$

Two classes of parametric problems

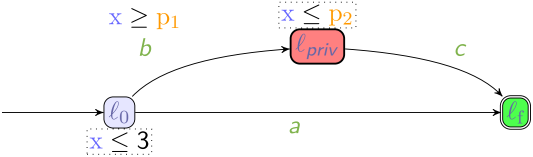
p-Emptiness problem

Decide the emptiness of the set of parameter valuations v s. t. $\mathrm{v}(\mathcal{P})$ is ET-opaque

p-Synthesis problem

Synthesize the set of parameter valuations v s. t. $\mathrm{v}(\mathcal{P})$ is ET-opaque

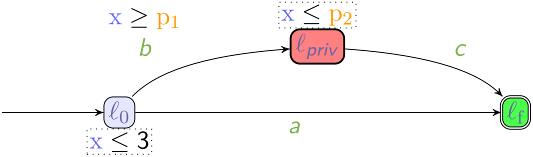
Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	∃	Weak	Full
p-Emptiness			
p-Synthesis			

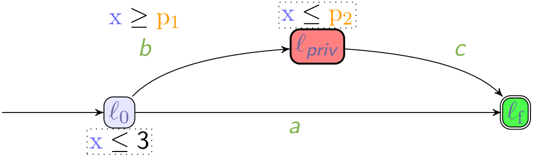
Example



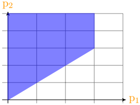
Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	∃	Weak	Full
p-Emptiness	$\times (\exists v)$	$\times (\exists v)$	$\times (\exists v)$
p-Synthesis			

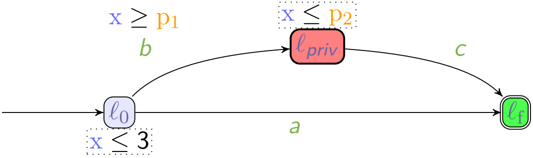
Example



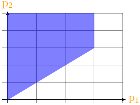
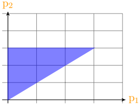
Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	∃	Weak	Full
p-Emptiness	×(∃v)	×(∃v)	×(∃v)
p-Synthesis	$0 \leq p_1 \leq 3$ $\land\ p_1 \leq p_2$ 		

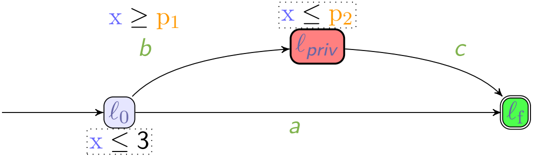
Example



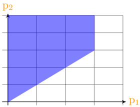
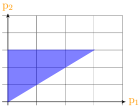
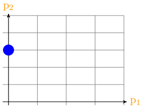
Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	$\exists$	**Weak**	**Full**
p-Emptiness	$\times(\exists v)$	$\times(\exists v)$	$\times(\exists v)$
p-Synthesis	$0 \leq p_1 \leq 3$ $\wedge\ p_1 \leq p_2$ 	$0 \leq p_1 \wedge p_2 \leq 3$ $\wedge\ p_1 \leq p_2$ 	

Example



Private	$[p_1, p_2]$
Public	$[0, 3]$

ET-opacity notion	$\exists$	Weak	Full
p-Emptiness	$\times (\exists v)$	$\times (\exists v)$	$\times (\exists v)$
p-Synthesis	$0 \leq p_1 \leq 3 \wedge p_1 \leq p_2$ 	$0 \leq p_1 \wedge p_2 \leq 3 \wedge p_1 \leq p_2$ 	$p_1 = 0 \wedge p_2 = 3$ 

Decidability results for ET-opacity

		∃-ET-opaque	weakly ET-opaque	fully ET-opaque
Decision	TA	√	√	√
p-emptiness	L/U-PTA	√	×	×
	PTA	×	×	×
p-synthesis	L/U-PTA	×	×	×
	PTA	×	×	×

- L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [Hun+02]
- *Proofs are based on the region automaton (for TAs) and by reduction from EF-emptiness (for PTAs). (see formal proofs in Manuscript, Chapter 7)*

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

Decidability results for ET-opacity

		∃-ET-opaque	weakly ET-opaque	fully ET-opaque
Decision	TA	√	√	√
p-emptiness	L/U-PTA	√	×	×
	PTA	×	×	×
p-synthesis	L/U-PTA	×	×	×
	PTA	×	×	×

- L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [Hun+02]
- *Proofs are based on the region automaton (for TAs) and by reduction from EF-emptiness (for PTAs). (see formal proofs in Manuscript, Chapter 7)*

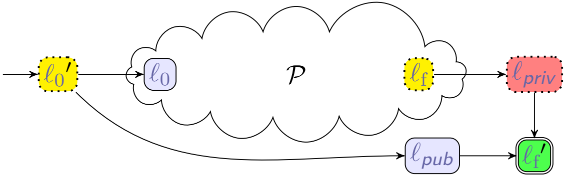
[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* (2022)

ET-opacity synthesis is (very) difficult

Theorem (Undecidability of ∃-ET-opacity p-emptiness)

Given $\mathcal{P}$, the mere existence of a parameter valuation v s. t. $\mathrm{v}(\mathcal{P})$ ∃-ET-opacity is undecidable.

Proof idea: reduction from reachability-emptiness for PTAs



Remark: L/U-PTA is a decidable subclass

Outline

Preliminaries: (Parametric) Timed model checking

Contribution: Efficient verification in Parametric Timed Automata

Contribution: Execution-time opacity

- ET-opacity problems in TAs
- ET-opacity problems in PTAs
- Expiring ET-opacity problems
- Untimed control

Conclusion & Perspectives

Expiring ET-opacity

- How to deal with outdated secrets?
  e. g., cache values, status of the memory, ...

Idea

The secret can expire: beyond a certain duration, knowing the secret is useless to the attacker (e. g., a cache value) [Amm+21]

Expiring ET-opacity

Assumption

Knowing an expired secret is equivalent to not knowing a secret

	Secret runs	**Non-secret runs**
ET-opacity	Runs visiting the private location (= private runs)	Runs not visiting the private location (= public runs)
expiring-ET-opacity	Private runs with ℓ_{priv} visit $\leq \Delta$ before the system completion	(i) Public runs and (ii) Private runs with ℓ_{priv} visit $> \Delta$ before the system completion

[ICECCS23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (2023). To appear. Springer, 2023

Three levels of ET-opacity

Existential (∃)

private durations $\cap$ public durations $\neq \emptyset$

Weak

private durations $\subseteq$ public durations

Full

private durations $=$ public durations

Three levels of expiring ET-opacity

Existential (∃) expiring

secret durations $\cap$ non-secret durations $\neq \emptyset$

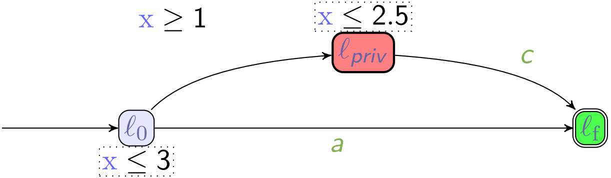
Weak expiring

secret durations $\subseteq$ non-secret durations

Full expiring

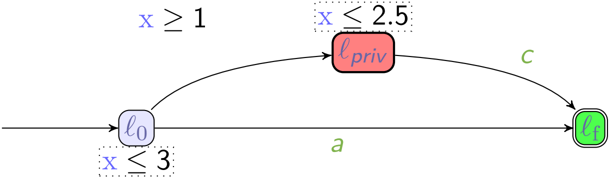
secret durations $=$ non-secret durations

Example



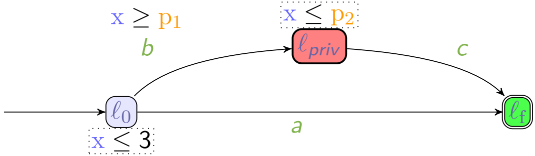
ET-opacity notion		Secret	Non-secret	Answer
	∃			√
	weak	$[1, 2.5]$	$[0, 3]$	√
	full			×
	∃-exp.			√
$\Delta = 1$	weak-exp.	$[1, 2.5]$	$(2, 2.5] \cup [0, 3]$	√
	full-exp.			×

Example



ET-opacity notion		Secret	Non-secret	Answer
	∃	$[1, 2.5]$	$[0, 3]$	√
	weak			√
	full			×
$\Delta = 1$	∃-exp.	$[1, 2.5]$	$(2, 2.5] \cup [0, 3]$	√
	weak-exp.			√
	full-exp.			×
$\Delta = 1.25$	∃-exp.	$[1, 2.5]$	$(2.25, 2.5] \cup [0, 3]$	√
	weak-exp.			√
	full-exp.			×

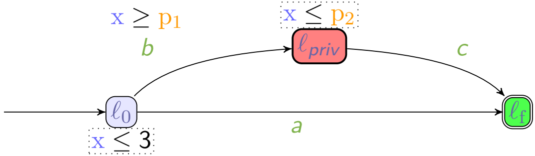
Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	Weak	Full
(p+Δ)-Emptiness		
(p+Δ)-Synthesis		

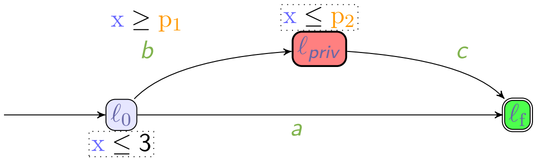
Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**Weak**	**Full**
(p+Δ)-Emptiness	$\times (\exists v)$	$\times (\exists v)$
(p+Δ)-Synthesis		

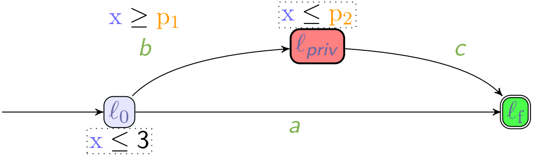
Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**Weak**	**Full**
(p+Δ)-Emptiness	$\times(\exists v)$	$\times(\exists v)$
(p+Δ)-Synthesis	$p_1 > 3 \lor \Delta = 0 \lor p_2 \leq 3 \lor p_1 + \Delta <= 3$	

Example



	if $p_1 \leq 3$	otherwise
Secret	$[p_1, \min(\Delta + 3, p_2)]$	$\emptyset$
Non-secret	$(p_1 + \Delta, p_2] \cup [0, 3]$	$\emptyset \cup [0, 3]$

ET-opacity notion	**Weak**	**Full**
(p+Δ)-Emptiness	$\times(\exists v)$	$\times(\exists v)$
(p+Δ)-Synthesis	$p_1 > 3 \vee \Delta = 0 \vee p_2 \leq 3 \vee p_1 + \Delta <= 3$	$p_1 = 0 \wedge ((\Delta \leq 3 \wedge 3 \leq p_2 \leq \Delta + 3) \vee (p_2 = 3))$

Decidability results for expiring-ET-opacity

		weakly expiring-ET-opaque	fully expiring-ET-opaque
Δ-emptiness	TA	√	√
Δ-synthesis	TA	√	?
$(p + \Delta)$-emptiness	L/U-PTA	×	×
$(p + \Delta)$-emptiness	PTA	×	×
$(p + \Delta)$-synthesis	L/U-PTA	×	×
$(p + \Delta)$-synthesis	PTA	×	×

- ∃-expiring ET-opacity was left as a future work.
- L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [Hun+02]

[ICECCS23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (2023). To appear. Springer, 2023

Decidability results for expiring-ET-opacity

		weakly expiring-ET-opaque	fully expiring-ET-opaque
Δ-emptiness	TA	√	√
Δ-synthesis	TA	√	?
$(p + \Delta)$-emptiness	L/U-PTA	×	×
	PTA	×	×
$(p + \Delta)$-synthesis	L/U-PTA	×	×
	PTA	×	×

- ∃-expiring ET-opacity was left as a future work.
- L/U-PTA (*Lower/Upper-PTA*): subclass of PTA where the parameters are partitioned into two sets (either compared to clocks as upperbound, or as lower bound) [Hun+02]
- *Proofs are based on the region automaton (for TAs) and by reduction from EF-emptiness (for PTAs). (see formal proofs in Manuscript, Chapter 8)*

[ICECCS23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (2023). To appear. Springer, 2023

Outline

Preliminaries: (Parametric) Timed model checking

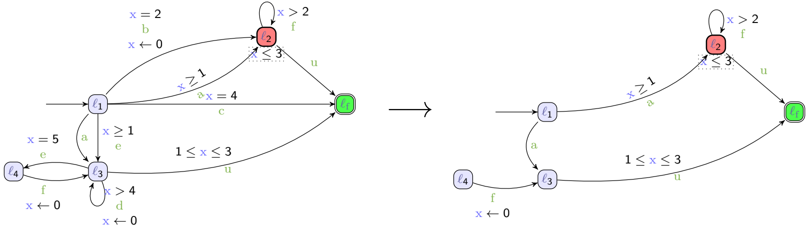
Contribution: Efficient verification in Parametric Timed Automata

Contribution: Execution-time opacity
- ET-opacity problems in TAs
- ET-opacity problems in PTAs
- Expiring ET-opacity problems
- Untimed control

Conclusion & Perspectives

Untimed control

Overview of Chapter 9 of the manuscript



- Restrict the behavior of the system to ensure ET-opacity
- Development of an open-source tool strategFTO (≈ 1200 lines of code, Java)
  - Enumeration of transition sets

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022

Outline

Preliminaries: (Parametric) Timed model checking

Contribution: Efficient verification in Parametric Timed Automata

Contribution: Execution-time opacity

Conclusion & Perspectives

Conclusion

Efficient verification

- A new benchmark library (119 models, 216 properties) [TAP21]
- Zone merging algorithm for PTA verification [FORMATS22]

Execution-time opacity

- Formalization and decidability results of ET-opacity [TOSEM22]
- Extension with secrets with expiration date [ICECCS23]
- Untimed control, implementation of strategFTO [FTSCS22]

Perspectives

Merging states in PZG

- Merge more than 2 states, “best” merge
- Compatibility of merging and liveness properties

Execution-time opacity

- Extension of the definition to another formalism
- Automatic translation of programs to PTAs

  → Preliminary translation with Petri nets including cache system

Other kind of parameters

- Parametric systems: probabilities, costs
- Parameterized systems: number of components

Publications

[FORMATS22] Étienne André, Dylan Marinho, Laure Petrucci, and Jaco van de Pol. "Efficient Convex Zone Merging in Parametric Timed Automata". In: *FORMATS* (2022). LNCS. Springer, 2022.

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022.

[ICECCS23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (2023). To appear. Springer, 2023.

[TAP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Parametric Timed Automata". In: *TAP* (2021). LNCS. Springer, 2021.

[TICSA23] Étienne André, Engel Lefaucheux, Didier Lime, Dylan Marinho, and Jun Sun. "Configuring Timing Parameters to Ensure Execution-Time Opacity in Timed Automata". In: *TICSA*. 2023.

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* 31 (2022).

References I

[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: *TCS* 126 (Apr. 1994).

[AFS13] Étienne André, Laurent Fribourg, and Romain Soulat. "Merge and Conquer: State Merging in Parametric Timed Automata". In: *ATVA* (2013). LNCS. Springer, Oct. 2013.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: *STOC* (1993). ACM, 1993.

[Amm+21] Ikhlass Ammar, Yamen El Touati, Moez Yeddes, and John Mullins. "Bounded opacity for timed systems". In: *Journal of Information Security and Applications* 61 (Sept. 2021).

References II

[AS19] Étienne André and Jun Sun. "Parametric Timed Model Checking for Guaranteeing Timed Opacity". In: *ATVA* (2019). LNCS. Springer, 2019.

[Dav05] Alexandre David. "Merging DBMs Efficiently". In: *NWPT* (2005). DIKU, University of Copenhagen, 2005.

[FORMATS22] Étienne André, Dylan Marinho, Laure Petrucci, and Jaco van de Pol. "Efficient Convex Zone Merging in Parametric Timed Automata". In: *FORMATS* (2022). LNCS. Springer, 2022.

References III

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022.

[HT15] Frédéric Herbreteau and Thanh-Tung Tran. "Improving Search Order for Reachability Testing in Timed Automata". In: *FORMATS* (2015). LNCS. Springer, 2015.

[Hun+02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. "Linear parametric model checking of timed automata". In: *Journal of Logic and Algebraic Programming* 52-53 (2002).

References IV

[ICECCS23] Étienne André, Engel Lefaucheux, and Dylan Marinho. "Expiring opacity problems in parametric timed automata". In: *ICECCS* (2023). To appear. Springer, 2023.

[TAP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Parametric Timed Automata". In: *TAP* (2021). LNCS. Springer, 2021.

[TOSEM22] Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. "Guaranteeing Timed Opacity using Parametric Timed Model Checking". In: *ACM TOSEM* 31 (2022).

Licensing

Source of the graphics used I

Author: Fidsor
Source: https://pixabay.com/fr/illustrations/fraude-pirate-hame%C3%A7onnage-escroquer-7065
License: Pixabay Content License

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien exterminate
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_exterminate.svg
License: public domain

Title: Piratey, vector version

Source of the graphics used II

Author: Gustavb
Source: https://commons.wikimedia.org/wiki/File:Piratey,_vector_version.svg
License: CC by-sa

Title: Expired
Author: RRZEicons
Source: https://commons.wikimedia.org/wiki/File:Expired.svg
License: CC by-sa

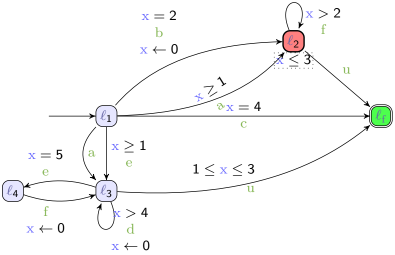
Results

		Nomerge	M2.12	RVMr	OQM
Time	# wins	24	20	22	**42**
	Avg (s)	10.0	5.47	4.56	**3.77**
	Avg (merge) (s)	18.8	7.83	5.57	**3.63**
	Avg (no merge) (s)	3.83	**3.82**	3.85	3.88
	Median (s)	1.39	1.2	1.14	**1.12**
	Norm. avg	1.0	0.91	0.91	**0.87**
	Norm. avg (merge)	1.0	0.75	0.74	**0.64**
	Norm. avg (no merge)	**1.0**	1.02	1.03	1.03
States	# wins	0	19	**37**	16
	Avg	11443.08	11096.54	**11064.37**	11120.79
	Avg (merge)	1512.02	670.43	**592.31**	729.33
	Median	2389.5	703.5	**604.5**	905.0
	Norm. avg	1.0	0.86	**0.84**	0.88

Gain of 62% of the average computation time

An example

[FTSCS22]



Uncontrollable u

Controllable a, b, c, d, e, f

Is the system fully ET-opaque?

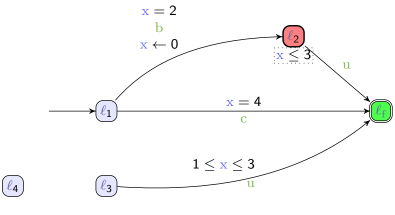
- Visiting ℓ_2: $[1, 5]$
- Not visiting ℓ_2: $[1, 3] \cup [4, 4] \cup [5, +\inf)$

$\Rightarrow$ Not fully ET-opaque

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. “strategFTO: Untimed control for timed opacity”. In: *FTSCS* (2022). ACM, 2022

An example

[FTSCS22]



Uncontrollable u

Controllable a, b, c, d, e, f

Allowed u + b, c

Disabled a, d, e, f

Is the system fully ET-opaque?

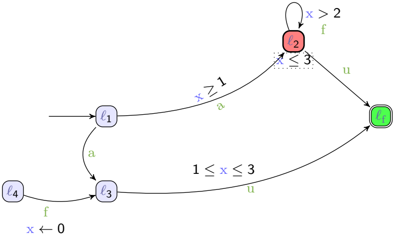
- Visiting ℓ_2: $[2, 5]$
- Not visiting ℓ_2: $[4, 4]$

$\Rightarrow$ Not fully ET-opaque

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022

An example

[FTSCS22]



Uncontrollable u

Controllable a, b, c, d, e, f

Allowed u + a, f

Disabled b, c, d, e

Is the system fully ET-opaque?

- Visiting ℓ_2: $[1, 3]$
- Not visiting ℓ_2: $[1, 3]$

$\Rightarrow$ Fully ET-opaque

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022

Untimed control

Overview of Chapter 9 of the manuscript

- Restrict the behavior of the system to ensure ET-opacity
- Development of an open-source tool strategFTO ($\approx$ 1200 lines of code, Java)
  - Enumeration of transition sets

Experiments

- Proof of concept
- Scalability of the tool

Publication

https://github.com/DylanMarinho/Controlling-TA

DOI 10.5281/zenodo.7181848

[FTSCS22] Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. "strategFTO: Untimed control for timed opacity". In: *FTSCS* (2022). ACM, 2022