Parametric non-interference in timed automata

Étienne André and Aleksander Kryukov
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

Supported by the ANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015)
Outline

1. Introduction
2. Parametric timed automata
3. Problem
4. Approach
5. Case study
6. Perspectives
Context: security

Security of computer systems

- Threats coming from an intruder or an unsafe medium (Internet)

---


Context: security

Security of computer systems

- Threats coming from an intruder or an unsafe medium (Internet)

Risk: consequence of external actions onto critical internal behaviors: non-interference

---


Context: security

Security of computer systems

- Threats coming from an intruder or an unsafe medium (Internet)

Risk: consequence of external actions onto critical internal behaviors: non-interference

Timed systems: challenging

- Time is a potential attack vector against secure systems [Koc96][Ben+15]
- A non-interferent system can become interferent when timing information is added [GMR07]

---


Étienne André  Parametric non-interference in timed automata 5th March 2021
Context: non-interference

Measure the disturbance of a system.
A system is non-interferent when some disturbance on some high-level actions does not affect the observable behavior (“low-level” actions).
When adding time information: Question of the frequency.
Context: non-interference

Measure the disturbance of a system

A system is non-interferent when some disturbance on some high-level actions does not affect the observable behavior (“low-level” actions)

When adding time information: Question of the frequency

Key point: frequency

Does performing an arbitrary high-level action at a given frequency disturbs the observable behavior?

- [Bar+o2]: observable behavior = timed language
- [BT03]: observable behavior = set of discrete states


[BT03] Roberto Barbuti and Luca Tesei. “A Decidable Notion of Timed Non-Interference”. In: *Fundamenta Informaticae* 54.2-3 (2003), pp. 137–150
Context: non-interference

Measure the disturbance of a system

A system is non-interferent when some disturbance on some high-level actions does not affect the observable behavior ("low-level" actions)

When adding time information: Question of the frequency

Key point: frequency

Does performing an arbitrary high-level action at a given frequency disturbs the observable behavior?

- [Bar+02]: observable behavior = timed language
- [BT03]: observable behavior = set of discrete states

Here, we will address a parametric version of the problem, and synthesize this frequency: at which frequency can we perform high-level actions without disturbing the observable behavior?
Outline

1. Introduction
2. Parametric timed automata
3. Problem
4. Approach
5. Case study
6. Perspectives
Outline

1 Introduction

2 Parametric timed automata
   - Timed automata
   - Parametric timed automata

3 Problem

4 Approach

5 Case study

6 Perspectives
Timed automaton (TA)

- Finite state automaton (sets of locations)

Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

---


Étienne André
Timed automaton (TA)

- 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)**

- 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

---


---

Étienne André

Parametric non-interference in timed automata

5th March 2021
Timed automaton (TA)

- 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)

- 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

---


Étienne André
Parametric non-interference in timed automata

5th March 2021 7 / 35
Outline

1 Introduction

2 Parametric timed automata
   - Timed automata
   - Parametric timed automata

3 Problem

4 Approach

5 Case study

6 Perspectives
timed model checking

A model of the system

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

A property to be satisfied

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

Yes if...

\[ 2 \times \text{delay} > 20.46 \times \text{period} \]
Parametric Timed Automaton (PTA)

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

```
\begin{align*}
y & = 8 \\
& \text{coffee!}
\end{align*}
```

```
\begin{align*}
& x \leftarrow 0 \\
& y \leftarrow 0 \\
& y \leq 5 \\
& \text{press?} \\
& x \geq 1 \\
& x \leftarrow 0 \\
\end{align*}
```

```
\begin{align*}
& y = 5 \\
& \text{cup!}
\end{align*}
```

---

Parametric Timed Automaton (PTA)

- 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

[rhs]

Notation: Valuation of a PTA

- Given a PTA $\mathcal{A}$ and a parameter valuation $v$, we denote by $v(\mathcal{A})$ the (non-parametric) timed automaton where each parameter $p$ is valuated by $v(p)$. 
Notation: Valuation of a PTA

- Given a PTA \( \mathcal{A} \) and a parameter valuation \( v \), we denote by \( v(\mathcal{A}) \) the (non-parametric) timed automaton where each parameter \( p \) is valuated by \( v(p) \).

\[
\begin{align*}
&\begin{array}{l}
\text{press?} \\
x \leftarrow 0 \\
y \leftarrow 0 \\
y = p_3 \\
\text{coffee!}
\end{array} \\
&\begin{array}{l}
x \geq p_1 \\
\text{press?} \\
x \leftarrow 0 \\
y = p_2 \\
\text{cup!}
\end{array} \\
&\begin{array}{l}
y \leq p_2 \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{l}
x \leftarrow 0 \\
y \leftarrow 0 \\
y = 5 \\
\text{cup!}
\end{array} \\
&\begin{array}{l}
x \geq 1 \\
\text{press?} \\
x \leftarrow 0 \\
y = 5 \\
\text{coffee!}
\end{array} \\
&\begin{array}{l}
y \leq 5 \\
\end{array}
\end{align*}
\]

with \( v : \) 
\[
\begin{align*}
p_1 &\rightarrow 1 \\
p_2 &\rightarrow 5 \\
p_3 &\rightarrow 8
\end{align*}
\]
Concrete semantics of timed automata

- **Concrete state** of a TA: pair \((\ell, w)\), where
  - \(\ell\) is a location,
  - \(w\) is a valuation of each clock

Example: \((\square, (x=1.2, y=3.7))\)

- **Concrete run**: alternating sequence of concrete states and actions or time elapse
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with two doses of sugar

idle
adding sugar
delivering coffee
The most critical system: The coffee machine

Example of concrete run for the coffee machine
- Coffee with 2 doses of sugar

\[ x = 0 \\
\] \[ y = 0 \]
The most critical system: The coffee machine

Example of concrete run for the coffee machine
- Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

 idle
 adding sugar
 delivering coffee

Étienne André
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1.5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

x = 0 0 1.5 0 2.7
y = 0 0 1.5 1.5 4.2
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>Press</th>
<th>Time (s)</th>
<th>Press</th>
<th>Time (s)</th>
<th>Press</th>
<th>Time (s)</th>
<th>Press</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>2.7</td>
<td>0</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>1.5</td>
<td>4.2</td>
</tr>
<tr>
<td>2.7</td>
<td>4.2</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>1.5</td>
<td>0</td>
<td></td>
<td>2.7</td>
<td>0</td>
<td></td>
<td>0.8</td>
<td>5</td>
<td></td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td>4.2</td>
<td>4.2</td>
<td></td>
<td>0.8</td>
<td>5</td>
<td></td>
<td>3.8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td>4.2</td>
<td>4.2</td>
<td></td>
<td>0.8</td>
<td>5</td>
<td></td>
<td>3.8</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. Parametric timed automata
3. Problem
4. Approach
5. Case study
6. Perspectives
$n$-location-non-interference: Definition

Let $\Sigma = L \cup H$

- $L$: low-level actions
- $H$: high-level actions

**Definition**

A TA $\mathcal{A}$ is $n$-location-non-interferent if the sets of reachable locations are equal in the following TAs:

1. $\mathcal{A}$ without any high-level action
2. $\mathcal{A}$ with high-level actions separated by at least $n$ time units
**$n$-location-non-interference: Example**

\[ y < 3 \land x = p \quad \text{and} \quad y > 2 \land x = 0 \]

\[ x \leftarrow 0 \]

Fix $p = 1.1$
n-location-non-interference: Example

Fix $p = 1.1$

1. Locations reachable in $\mathcal{A}$ without any high-level action: \{\○, □\}
\( n \)-location-non-interference: Example

\[ y < 3 \land x = p \]
\[ y > 2 \]
\[ x \leftarrow 0 \]

Fix \( p = 1.1 \)

1. Locations reachable in \( \mathcal{A} \) without any high-level action: \{ \[ , \] \}
2. Locations reachable in \( \mathcal{A} \) with high-level actions separated by at least 1 time unit: \{ \[ , \[ , \] \}

\[ \Rightarrow \mathcal{A} \text{ is not 1-location-non-interfering} \]
$n$-location-non-interference: Example

\[ y < 3 \land x = p \]

\[ y > 2 \]

\[ x \leftarrow 0 \]

\[ y > 2 \land x = 0 \]

Fix $p = 1.1$

1. Locations reachable in $\mathcal{A}$ without any high-level action: \{ \[ \text{\color{green} \circlearrowright}, \text{\color{red} \circlearrowleft} \] \}

2. Locations reachable in $\mathcal{A}$ with high-level actions separated by at least 1 time unit: \{ \[ \text{\color{green} \circlearrowright}, \text{\color{red} \circlearrowleft}, \text{\color{blue} \circlearrowright} \] \}

3. Locations reachable in $\mathcal{A}$ with high-level actions separated by at least 2 time units: \{ \[ \text{\color{green} \circlearrowright}, \text{\color{red} \circlearrowleft} \] \}

$\Rightarrow \mathcal{A}$ is not 1-location-non-interfering

$\Rightarrow \mathcal{A}$ is 2-location-non-interfering
Problem

Problem: \( n \)-location-non-interference synthesis

Inputs:
- A parametric TA \( \mathcal{A} \) with parameters \( P \)
- A parameter \( n \)

Goal:
“Synthesize valuations \( v \) of \( P \) and of \( n \) such that \( v(\mathcal{A}) \) is \( n \)-location-non-interfering.”
Outline

1. Introduction
2. Parametric timed automata
3. Problem
4. Approach
5. Case study
6. Perspectives
Our approach in a nutshell: Gadget

We take the parallel product of

- $A$, and
- a special gadget PTA $\mathcal{I}_{\text{Interf}_H^n}$ constraining any high-level action to be separated by at least $n$ time units

\[ L \xrightarrow{L} \ell_0 \xleftarrow{H} x_{\text{interf}} \leftarrow 0 \]
\[ H \xrightarrow{L} \ell_1 \xrightarrow{\ell_1} \]
\[ x_{\text{interf}} \geq n \]
\[ x_{\text{interf}} \leftarrow 0 \]
Our approach in a nutshell: Reachability synthesis

Then, we compute

1. a set of locations $G$ to be reached for some desired property in $\mathcal{A}$
2. the set of parameter valuations for which $G$ is reachable in $\mathcal{A} \parallel \text{Interf}_H^n$


Étienne André
Parametric non-interference in timed automata
Our approach in a nutshell: Reachability synthesis

Then, we compute

1. a set of locations \( G \) to be reached for some desired property in \( \mathcal{A} \)
2. the set of \textbf{parameter valuations} for which \( G \) is reachable in \( \mathcal{A} \| \text{Interf}^n_H \)

Toolkit:

- Semi-algorithm: \textit{reachability synthesis} [JLR15]
  - semi-algorithm: no theoretical guarantee on termination
- implemented in IMITATOR

---

Outline

1. Introduction
2. Parametric timed automata
3. Problem
4. Approach
5. Case study
6. Perspectives
Outline

1. Introduction
2. Parametric timed automata
3. Problem
4. Approach
5. Case study
   - Fischer protocol
   - IMITATOR in a nutshell
   - Results
6. Perspectives
Fischer mutual exclusion protocol

Two processes $P_1$ and $P_2$ running in parallel compete for the critical section. Atomic reads and writes are permitted to a shared variable $v$. Every access to the shared memory containing $v$ takes $acc$ units of time.
Fischer mutual exclusion protocol

Two processes $P_1$ and $P_2$ running in parallel compete for the **critical section**. Atomic reads and writes are permitted to a shared variable $v$
Every access to the shared memory containing $v$ takes $acc$ units of time.
Each process $P_i$ executes the following code:

```java
repeat
    await v = 0
    v := i
    delay b
until v = i
v := 0
(* Critical section* )
```

[BT03] Roberto Barbuti and Luca Tesei. “A Decidable Notion of Timed Non-Interference”. In: *Fundamenta Informaticae* 54.2-3 (2003), pp. 137–150
Fischer mutual exclusion protocol

Two processes $P_1$ and $P_2$ running in parallel compete for the critical section. Atomic reads and writes are permitted to a shared variable $v$ Every access to the shared memory containing $v$ takes $acc$ units of time. Each process $P_i$ executes the following code:

```
repeat
    await v = 0
    v := i
    delay b
until v = i
v := 0
(* Critical section*)
```

An assignment takes (at most) $a$ time units
Maximum time needed to execute the critical section is $ucs$
Fischer mutual exclusion protocol

Two processes $P_1$ and $P_2$ running in parallel compete for the critical section. Atomic reads and writes are permitted to a shared variable $v$. Every access to the shared memory containing $v$ takes $acc$ units of time. Each process $P_i$ executes the following code:

```plaintext
repeat
    await v = 0
    v := i
    delay b
until v = i
v := 0
(* Critical section* )
```

An assignment takes (at most) $a$ time units.
Maximum time needed to execute the critical section is $ucs$.
Crux: $P_i$ is allowed into the critical section only when $v = i$.
Attacker scheme

An **intruder** can take anytime a high-level transition “*att*”, nondeterministically changing \( v \) to 0, 1 or 2
Fischer: objective

Objective

Automatically infer conditions over $n$, $a$, $b$, $acc$ and $ucs$ guaranteeing $n$-location-non-interference.
Fischer: objective

**Objective**

Automatically infer conditions over $n, a, b, \text{acc}$ and $\text{ucs}$ guaranteeing $n$-location-non-interference.

Put it differently: offer guarantees that the Fischer protocol will still be valid even in the situation of an attack on the variable $v$, with a maximum frequency $n$.

In particular, since the reachable locations do not change, the location where both processes are in the critical section at the same time (safety violation) remains unreachable.
Outline

1. Introduction

2. Parametric timed automata

3. Problem

4. Approach

5. Case study
   - Fischer protocol
   - IMITATOR in a nutshell
   - Results

6. Perspectives
IMITATOR

- A tool for modeling and verifying **timed concurrent systems** with unknown constants modeled with **parametric timed automata**
  - Communication through (strong) broadcast synchronization
  - Rational-valued shared discrete variables
  - **Stopwatches**, to model schedulability problems with preemption
  - **Multi-rate** clocks

- **Synthesis algorithms**
  - (non-Zeno) parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking
IMITATOR

Under continuous development since 2008

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- ...and more

Free and open source software: Available under the GNU-GPL license


IMITATOR

Under continuous development since 2008

A library of benchmarks

- Communication protocols
- Schedulability problems
- Asynchronous circuits
- ...and more

Free and open source software: Available under the GNU-GPL license

Try it!

www.imitator.fr


Some success stories

■ Modeled and verified an *asynchronous memory circuit* by ST-Microelectronics

■ Parametric schedulability analysis of a prospective architecture for the flight control system of the *next generation of spacecrafts* designed at ASTRIUM Space Transportation

■ Verification of software product lines

■ Formal timing analysis of *music scores*

■ Solution to a challenge related to a *distributed video processing system* by Thales

■ *Parametric timed pattern matching*

---


Étienne André
Outline

1 Introduction

2 Parametric timed automata

3 Problem

4 Approach

5 Case study
   - Fischer protocol
   - IMITATOR in a nutshell
   - Results

6 Perspectives
A modified model

Modified the model from [BT03]:

- Corrected some (non-trivial) aspects
  - Entirely rewrote the serializer (responsible for synchronizing the processes and the intruder)
- Added parameters, notably $n$
- Added the $n$-non-interference gadget

Target set of locations $G'$:

- All locations except those where the mutual exclusion is violated (both processes in the critical section together)
Preliminary results

😊 Analysis with IMITATOR does not terminate
😊 …but an over-approximation is synthesized

Preliminary results

😊 Analysis with IMITATOR does not terminate
😊 …but an over-approximation is synthesized

Claim: this result might be exact

😊 We “tested” dozens of parameter valuations with UPPAAL [LPY97]
😊 No formal guarantee of soundness!

Preliminary results

😊 Analysis with IMITATOR does not terminate
😊 …but an over-approximation is synthesized

Claim: this result might be exact
😊 We “tested” dozens of parameter valuations with UPPAAL [LPY97]
😊 No formal guarantee of soundness!

One disjunct among the synthesized constraint:

\[
\begin{align*}
    n & \geq 0 \\
    b & \geq acc + n \\
    b & \geq 3 \times acc \\
    a & > 0 \\
    acc & > ucs > 0
\end{align*}
\]

Sources, binaries, models, results available at www.imitator.fr/static/ICECCS20

Outline

1 Introduction
2 Parametric timed automata
3 Problem
4 Approach
5 Case study
6 Perspectives
Conclusion and perspectives

Conclusion

- A first notion of parametric $n$-non-interference
- Helps to quantify the admissible frequency of attacks without any effect on the intended behavior
- Dually: quantify the effect of internal actions (by admins) without observable behavior from the outside
- Approximated constraint for Fischer protocol
  - Toolkit: IMITATOR

Perspectives:

- Theoretical issues: decidable subclasses?
- Non-interference w.r.t. the language
- Extend to control

---


Étienne André

Parametric non-interference in timed automata
Conclusion and perspectives

Conclusion

- A first notion of **parametric** $n$-non-interference
- Helps to **quantify the admissible frequency** of attacks without any effect on the intended behavior
- Dually: quantify the effect of internal actions (by admins) without observable behavior from the outside
- **Approximated constraint** for Fischer protocol
  - Toolkit: IMITATOR

Perspectives:

- **Theoretical issues**: decidable subclasses?
- Non-interference w.r.t. the **language**
- Extend to **control** [Ben+15]

---

We hire!

- **Who:**
  - PhD students
  - Research fellows (post-doc)

- **Project ANR-NRF ProMiS**
  - quantitative formal methods + security (2020-2023)

- **Where:** France (Nancy / Nantes), Singapore

...starting anytime!
Bibliography


Roberto Barbuti and Luca Tesei. “A Decidable Notion of Timed Non-Interference”. In: *Fundamenta Informaticae* 54.2-3 (2003), pp. 137–150.


Licensing
Source of the graphics used I

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: Typing monkey
Author: KaterBegemot
Source: https://commons.wikimedia.org/wiki/File:Typing_monkey.svg
License: CC by-sa 3.0
License of this document

This presentation can be published, reused and modified under the terms of the license Creative Commons Attribution-ShareAlike 4.0 Unported (CC BY-SA 4.0).

(\LaTeX source available on demand)

Author: Étienne André