A benchmarks library for parametric timed model checking

Étienne André

LIPN, Université Paris 13, CNRS, France
National Institute of Informatics, Japan
JFLI, UMI CNRS, Tokyo, Japan

Supported by the ANR national research program PACS (ANR-14-CE28-0002) and JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603).
Context: Verifying complex timed systems

- Real-time systems are everywhere
  - Hard **timing** constraints and **concurrency**
  - Criticality: risk for huge damages in case of unexpected behavior (**bug**)
  - Bugs discovered when final testing: **expensive**

→ Need for a thorough specification and verification phase
Outline

1. Parametric timed automata
2. A benchmark library for parametric timed model checking
3. Perspectives
Outline

1. Parametric timed automata
2. A benchmark library for parametric timed model checking
3. Perspectives
Model checking timed concurrent systems

- Use formal methods

\[
y = \text{delay} \quad x := 0 \quad x < \text{period}
\]

A model of the system

[Baier and Katoen, 2008]

A property to be satisfied

is unreachable

Turing award (two.osf/zero.osf/zero.osf/seven.osf) to Edmund M. Clarke, Allen Emerson and Joseph Sifakis
Model checking timed concurrent systems

- Use formal methods

\[ y = \text{delay} \]
\[ x := 0 \]
\[ x < \text{period} \]

A model of the system

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

[Baier and Katoen, 2008]

\[ \square \text{is unreachable} \]

A property to be satisfied
Model checking timed concurrent systems

- Use formal methods

```
y = delay
```
```
x := 0
```
```
x < period
```

A model of the system

A property to be satisfied

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

Yes

No

Counterexample

Turing award (2007) to Edmund M. Clarke, Allen Emerson and Joseph Sifakis

[Baier and Katoen, 2008]
Outline

1 Parametric timed automata
   - Timed automata
     - Parametric timed automata
     - IMITATOR in a nutshell

2 A benchmark library for parametric timed model checking

3 Perspectives
Timed automaton (TA)

- Finite state automaton (sets of locations)

---

idle
adding sugar
delivering coffee
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

- 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
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

[Alur and Dill, 1994]
Timed automaton (TA)

- Finite state automaton (sets of \textit{locations} and \textit{actions}) augmented with a set $X$ of \textit{clocks}
  - Real-valued variables evolving linearly \textbf{at the same rate}
  - Can be compared to integer constants in invariants

- Features
  - Location \textit{invariant}: property to be verified to stay at a location

\begin{itemize}
  \item \texttt{press?}
  \item $y \leq 5$
  \item \texttt{cup!}
  \item $y \leq 8$
\end{itemize}

\texttt{press?}

![Diagram of a timed automaton with states and transitions]
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

---

$y \leq 5$

$y = 8$

coffee!

$y \leq 8$

$y = 5$

cup!

$x \geq 1$

press?

press?

idle

adding sugar

delivering coffee

Étienne André

A benchmarks suite for parametric timed verification

16th November 2018
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

\[
y = 8 \\
\text{coffee!}
\]

\[
y \leq 5 \\
\text{press?}
\]

\[
x := 0 \\
y := 0
\]

\[
x \geq 1
\]

\[
x := 0
\]

\[
y = 5
\]

\[
\text{cup!}
\]

\[
y \leq 8
\]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with two.osf doses of sugar

Idle
Adding sugar
Delivering coffee

\[ y = 8 \quad \text{coffee!} \]
\[ y = 5 \quad \text{cup!} \]
\[ x := 0 \]
\[ y := 0 \]
\[ x \geq 1 \]
\[ y \leq 5 \]
\[ x := 0 \]
\[ y \leq 8 \]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

- idle
- adding sugar
- delivering coffee

$\begin{align*}
x &= 0 \\
y &= 0
\end{align*}$
The most critical system: The coffee machine

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

```
x = 0 0
y = 0 0
```
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>press?</th>
<th>1.5</th>
<th>press?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

- idle
- adding sugar
- delivering coffee

$x := 0$
y := 0

$y = 8$

coffee!

$y \leq 5$

$x := 0$
y := 0

$y = 5$

cup!

$x \geq 1$

press?

$y \leq 8$
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

$x = 0 \quad 0 \quad 1.5 \quad 0 \quad 2.7$

$y = 0 \quad 0 \quad 1.5 \quad 1.5 \quad 4.2$
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
The most critical system: The coffee machine

- Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

- Coffee with /two.osf doses of sugar
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar
Timed automata: A success story

- An expressive formalism
  - Dense time
  - Concurrency

- A tractable verification in theory
  - Reachability is PSPACE-complete

- A very efficient verification in practice
  - Symbolic verification: relatively insensitive to constants
  - Several model checkers, notably UPPAAL
  - Long list of successful case studies
Outline

1. Parametric timed automata
   - Timed automata
   - Parametric timed automata
   - IMITATOR in a nutshell

2. A benchmark library for parametric timed model checking

3. Perspectives
Beyond timed model checking: parameter synthesis

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Bouyer et al., 2013]: What happens if 50 is implemented with 49.99?
  - System incompletely specified: Can I verify my system even if I don’t know the period value with full certainty?
Beyond timed model checking: parameter synthesis

- Verification for one set of constants does not usually guarantee the correctness for other values

- Challenges
  - Numerous verifications: is the system correct for any value within [40; 60]?
  - Optimization: until what value can we increase 10?
  - Robustness [Bouyer et al., 2013]: What happens if 50 is implemented with 49.99?
  - System incompletely specified: Can I verify my system even if I don’t know the period value with full certainty?

- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)
timed model checking

A model of the system

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

Yes

No

Counterexample
Parametric timed model checking

A model of the system

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

Yes if...

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

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

![Diagram of a parametric timed automaton with transitions and invariants.]

- \( y \leq 5 \)
- \( y = 8 \)
- \( y \leq 8 \)
- \( x \geq 1 \)
- \( x := 0 \)
- \( y := 0 \)
- \( x := 0 \)

- press?
- coffee!
- cup!
## 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

---

### Example

- **Initial State:**
  - $x := 0$
  - $y := 0$

- **Transition:**
  - $y = p_3$
  - `coffee!`

- **Guard:**
  - $y \leq p_2$

- **Transition:**
  - $x \geq p_1$
  - `press?`
  - $x := 0$

- **Guard:**
  - $y = p_2$

- **Transition:**
  - `cup!`

- **Final State:**
  - $y \leq 8$
1990s: undecidability

- **EF-emptiness problem**
  
  “Is the set of parameter valuations for which a given location $l$ is reachable empty?”

  *undecidable*  

  [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]
1990s: undecidability

- **EF-emptiness problem**
  
  “Is the set of parameter valuations for which a given location $l$ is reachable empty?”
  
  **undecidable**  
  
  [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]

- **AF-emptiness problem**
  
  “Is the set of parameter valuations for which all runs eventually reach a given location $l$ empty?”
  
  **undecidable**  
  
  [Jovanović et al., 2015]
1990s: undecidability

- **EF-emptiness problem**
  “Is the set of parameter valuations for which a given location $l$ is reachable empty?”
  undecidable  
  [Alur et al., 1993, Miller, 2000, Doyen, 2007, Beneš et al., 2015]

- **AF-emptiness problem**
  “Is the set of parameter valuations for which all runs eventually reach a given location $l$ empty?”
  undecidable  
  [Jovanović et al., 2015]

- **Preservation of the untimed language**
  “Given a parameter valuation, does there exist another valuations with the same untimed language?”
  undecidable  
  [André and Markey, 2015]

**Bad news**
All interesting problems are undecidable for (general) parametric timed automata.

[ÉA, STTT 2018]
2010s: decidability

New decidability results
EF-emptiness problem with a limited number of clocks:

✓ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters

[Beneš et al., 2015]
2010s: decidability

New decidability results
EF-emptiness problem with a limited number of clocks:

√ 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters  
[Beneš et al., 2015]

√ 1 parametric clock and arbitrarily many rational-valued parameters  
[Miller, 2000]
2010s: decidability

New decidability results

EF-emptiness problem with a limited number of clocks:

- 1 parametric clock and arbitrarily many non-parametric clocks and integer-valued parameters  
  [Beneš et al., 2015]

- 1 parametric clock and arbitrarily many rational-valued parameters  
  [Miller, 2000]

- 2 parametric clocks and 1 integer-valued parameter  
  [Bundala and Ouaknine, 2014]

A subclass with mild decidability results: L/U-PTA

- Partition of parameters into lower-bound and upper-bound parameters

- Positive results [Hune et al., 2002, Bozzelli and La Torre, 2009, André and Lime, 2017]

- Negative results [Jovanović et al., 2015, André and Lime, 2017, André et al., 2018b]
2010s: pragmatism and applications

New algorithms (often independent of the decidability)

- Bounded model checking [Knapik and Penczek, 2012]
- Symbolic integer parameter synthesis [Jovanović et al., 2015]
- Distributed verification [André et al., 2015]
- LTL synthesis [Bezděk et al., 2016, Bezděk et al., 2018]
- Exploration orders [André et al., 2017, Nguyen et al., 2018]
- etc.

Concrete application domains

- Scheduling real-time systems
- Hardware verification
- Software product lines
- etc.
Outline

1. Parametric timed automata
   - Timed automata
   - Parametric timed automata
   - IMITATOR in a nutshell

2. A benchmark library for parametric timed model checking

3. 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

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 [Fribourg et al., 2012]

- Verification of software product lines [Luthmann et al., 2017]

- Offline monitoring [ÉA, Hasuo, Waga @ ICECCS’18]

- Formal timing analysis of music scores [Fanchon and Jacquemard, 2013]

- Solution to a challenge related to a distributed video processing system by Thales
Problem

Many recent papers propose new optimizations / new algorithms for parametric timed model checking

- Often in parametric timed automata

Problem

How to ensure a fair evaluation of these new techniques?

Related problem:

- what is the bottleneck for parametric timed model checking?
- what syntactic features do we actually need?
  - is the L/U subclass useful in practice?
  - do we need stopwatches?

[Hune et al., 2002]
[Cassez and Larsen, 2000]
Outline

1 Parametric timed automata

2 A benchmark library for parametric timed model checking

3 Perspectives
A benchmarks library

Accumulated from various sources

- Classical academic models (Fischer)
- Industrial protocols (CSMA/CD, BRP…)
- Industrial collaborations (ST-Microelectronics, Thales, ArianeGroup…)
- Also: education models (useful for teaching)
A benchmarks library

Accumulated from various sources

- Classical academic models (Fischer)
- Industrial protocols (CSMA/CD, BRP...)
- Industrial collaborations (ST-Microelectronics, Thales, ArianeGroup...)
- Also: education models (useful for teaching)

Domains of applications

- Hardware (asynchronous circuits)
- Communication protocols
- Real-time systems (jobshop, scheduling)
- Monitoring automotive systems
- And some more: Wireless fire alarm, producer-consumer...
Organization

A benchmark can contain different models

- Different scales (e.g., Fischer with 2, 3, 4… processes)
- Varying the number of parameters

A model can contain different properties

- For a given instance of Fischer, one can either synthesize correct valuations, or evaluate the system robustness
Organization

A benchmark can contain different models

- Different scales (e.g., Fischer with 2, 3, 4… processes)
- Varying the number of parameters

A model can contain different properties

- For a given instance of Fischer, one can either synthesize correct valuations, or evaluate the system robustness

In the current version:

- 34 benchmarks
- 80 models
- 122 properties
Classifying benchmarks

Number of variables

- Numbers of clocks, parameters...

Syntactic features

- Use of global discrete variables
- Use of stopwatches
- L/U property
- etc.

Type of properties

- Reachability/safety
- Unavoidability
- Robustness
- Optimization
- etc.
An insight in the library (with execution times)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Ref</th>
<th>Domain</th>
<th>Scal.</th>
<th>Sc</th>
<th>Ac</th>
<th>Xi</th>
<th>Pi</th>
<th>Vi</th>
<th>L/U</th>
<th>Inv</th>
<th>SW</th>
<th>Prop.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>And-Or</td>
<td>[Clarísó and Cortadella, 2005]</td>
<td>Circuit</td>
<td>X</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td>Misc.</td>
<td>3.01</td>
</tr>
<tr>
<td>CSMA/CD</td>
<td>[Kwiatkowska et al., 2007]</td>
<td>Protocol</td>
<td>✓</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
<td>✓</td>
<td>X</td>
<td></td>
<td></td>
<td>0.04</td>
</tr>
<tr>
<td>Fischer-AHV93</td>
<td>[Alur et al., 1993]</td>
<td>Protocol</td>
<td>✓</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>L/U</td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>HS</td>
<td></td>
</tr>
<tr>
<td>Fischer-HRSV02:3</td>
<td>[Hune et al., 2002]</td>
<td>Protocol</td>
<td>✓</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>L/U</td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>1.49</td>
<td></td>
</tr>
<tr>
<td>Flip-flop:2</td>
<td>[Clarísó and Cortadella, 2007]</td>
<td>Circuit</td>
<td>X</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>U</td>
<td>✓</td>
<td>X</td>
<td>Misc.</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>Flip-flop:12</td>
<td>[Clarísó and Cortadella, 2007]</td>
<td>Circuit</td>
<td>X</td>
<td>5</td>
<td>5</td>
<td>12</td>
<td>0</td>
<td>U</td>
<td>✓</td>
<td>X</td>
<td>Misc.</td>
<td>23.07</td>
<td></td>
</tr>
<tr>
<td>idle-time-sched:3</td>
<td>[Lipari et al., 2014]</td>
<td>RTS</td>
<td>✓</td>
<td>12</td>
<td>21</td>
<td>2</td>
<td>0</td>
<td>U</td>
<td>✓</td>
<td>✓</td>
<td>Safety</td>
<td>14.61</td>
<td></td>
</tr>
<tr>
<td>idle-time-sched:5</td>
<td>[Lipari et al., 2014]</td>
<td>RTS</td>
<td>✓</td>
<td>12</td>
<td>21</td>
<td>2</td>
<td>0</td>
<td>U</td>
<td>✓</td>
<td>✓</td>
<td>Safety</td>
<td>16.41</td>
<td></td>
</tr>
<tr>
<td>Jobshop:3-4</td>
<td>[Abdeddaim and Maier, 2001]</td>
<td>Sched.</td>
<td>✓</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td>Opt. reach.</td>
<td>5.58</td>
<td></td>
</tr>
<tr>
<td>NP-FPS-3tasks:50-0</td>
<td>[Jovanović et al., 2013]</td>
<td>RTS</td>
<td>X</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>1.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP-FPS-3tasks:100-2</td>
<td>[Jovanović et al., 2013]</td>
<td>RTS</td>
<td>X</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>65.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSLAF14-1</td>
<td>[Sun et al., 2013]</td>
<td>RTS</td>
<td>X</td>
<td>7</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>Safety</td>
<td>0.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSLAF14-2</td>
<td>[Wandeler et al., 2006, Sun et al., 2013]</td>
<td>RTS</td>
<td>X</td>
<td>6</td>
<td>14</td>
<td>2</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>Safety</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProdCons:2-3</td>
<td>[Knapik and Penczek, 2012]</td>
<td>Prod.-cons.</td>
<td>✓</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>L/U</td>
<td>✓</td>
<td>X</td>
<td>Reach.</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>train-AHV93</td>
<td>[Alur et al., 1993]</td>
<td>TGC</td>
<td>X</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>L/U</td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>WFAS</td>
<td>[Beneš et al., 2015]</td>
<td>Protocol</td>
<td>X</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td></td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>accel:1</td>
<td>[Hoxha et al., 2014, André et al., 2018a]</td>
<td>PTPM</td>
<td>✓</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td></td>
<td>✓</td>
<td>X</td>
<td>PTPM</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>accel:10</td>
<td>[Hoxha et al., 2014, André et al., 2018a]</td>
<td>PTPM</td>
<td>✓</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td></td>
<td>✓</td>
<td>X</td>
<td>PTPM</td>
<td>12.67</td>
<td></td>
</tr>
<tr>
<td>BRP</td>
<td>[D'Argenio et al., 1997]</td>
<td>Protocol</td>
<td>X</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>12</td>
<td>✓</td>
<td>X</td>
<td>Safety</td>
<td>248.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMTV:1A1</td>
<td>[Sun et al., 2015]</td>
<td>RTS</td>
<td>X</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>✓</td>
<td>X</td>
<td>Opt. reach.</td>
<td>6.97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMTV:1A3</td>
<td>[Sun et al., 2015]</td>
<td>RTS</td>
<td>X</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>✓</td>
<td>X</td>
<td>Opt. reach.</td>
<td>87.39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMTV:2</td>
<td>[Sun et al., 2015]</td>
<td>RTS</td>
<td>X</td>
<td>6</td>
<td>9</td>
<td>2</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>Opt. reach.</td>
<td>1.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gear:1</td>
<td>[Hoxha et al., 2014, André et al., 2018a]</td>
<td>PTPM</td>
<td>✓</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>PTPM</td>
<td>0.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gear:10</td>
<td>[Hoxha et al., 2014, André et al., 2018a]</td>
<td>PTPM</td>
<td>✓</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>PTPM</td>
<td>7.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCP</td>
<td>[Collomb-Annichini and Sighireanu, 2001]</td>
<td>Protocol</td>
<td>X</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>L/U</td>
<td>✓</td>
<td>X</td>
<td>Reach.</td>
<td>1.07</td>
<td></td>
</tr>
<tr>
<td>SIMOP:3</td>
<td>[André et al., 2009]</td>
<td>Automation</td>
<td>X</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>Reach.</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPSMALL:2</td>
<td>[Chevallier et al., 2009]</td>
<td>Circuit</td>
<td>X</td>
<td>11</td>
<td>11</td>
<td>2</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>Reach.</td>
<td>0.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPSMALL:26</td>
<td>[Chevallier et al., 2009]</td>
<td>Circuit</td>
<td>X</td>
<td>11</td>
<td>11</td>
<td>26</td>
<td>0</td>
<td>✓</td>
<td>X</td>
<td>Reach.</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
License

The entire library is available under the **GNU-GPL license**

- required sending some emails to authors of papers written 25 years ago ;-) 

Library with models, data and (some) results available at

[www.imitator.fr/library.html](http://www.imitator.fr/library.html)
Outline

1. Parametric timed automata
2. A benchmark library for parametric timed model checking
3. Perspectives
What’s next?

- **Naming** benchmarks in a unique manner?
  - `benchmark:model:property` (e.g., `Fischer:2:safety`)

- A **versioning system** will be important to track future changes and ensure fair comparisons
  - “We evaluated our new technique against the parametric timed model checking library v. 1.2”

- Syntax and translation
  - So far, only IMITATOR format
  - At the very least, UPPAAL translation (for compatible models) would be highly welcome

- **Contributions**
  - So far, users can propose models to the library on a free basis, but an automated method could be welcome
Bibliography


References IV

Robustness in timed automata.
Invited paper.

Decision problems for lower/upper bound parametric timed automata.

Advances in parametric real-time reasoning.

The impressive power of stopwatches.

Timed verification of the generic architecture of a memory circuit using parametric timed automata.
References V


Symbolic model checking for probabilistic timed automata.

UPPAAL in a nutshell.

Toward parametric timed interfaces for real-time components.

Modeling and testing product lines with unbounded parametric real-time constraints.

Decidability and complexity results for timed automata and semi-linear hybrid automata.


Additional explanation
Explanation for the 4 pictures in the beginning

Allusion to the Northeast blackout (USA, 2003)  
Computer bug  
Consequences: 11 fatalities, huge cost  
(Picture actually from the Sandy Hurricane, 2012)

Error screen on the earliest versions of Macintosh

Allusion to the sinking of the Sleipner A offshore platform (Norway, 1991)  
No fatalities  
Computer bug: inaccurate finite element analysis modeling  
(Picture actually from the Deepwater Horizon Offshore Drilling Platform)

Allusion to the MIM-104 Patriot Missile Failure (Iraq, 1991)  
28 fatalities, hundreds of injured  
Computer bug: software error (clock drift)  
(Picture of an actual MIM-104 Patriot Missile, though not the one of 1991)
Licensing
Source of the graphics used

Title: Hurricane Sandy Blackout New York Skyline
Author: David Shankbone
Source: https://commons.wikimedia.org/wiki/File:Hurricane_Sandy_Blackout_New_York_Skyline.JPG
License: CC BY 3.0

Title: Sad mac
Author: Przemub
Source: https://commons.wikimedia.org/wiki/File:Sad_mac.png
License: Public domain

Title: Deepwater Horizon Offshore Drilling Platform on Fire
Author: ideum
Source: https://secure.flickr.com/photos/ideum/4711481781/
License: CC BY-SA 2.0

Title: DA-SC-88-01663
Author: imcomkorea
Source: https://secure.flickr.com/photos/imcomkorea/3017886760/
License: CC BY-NC-ND 2.0
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

https://creativecommons.org/licenses/by-sa/4.0/