Timed automata
and parametric timed automata

Étienne André
LIPN, Université Paris 13, CNRS, France
Feat. joint work with Giuseppe Lipari and Sun Youcheng
Context: Verifying complex timed systems

- Need for early bug detection
  - Bugs discovered when final testing: expensive
  - Need for a thorough specification and verification phase
The Therac-25 radiation therapy machine (1/2)

- Radiation therapy machine used in the 1980s
- Involved in accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
  - Approximately 100 times the intended dose!
  - Numerous causes, including race condition
The Therac-25 radiation therapy machine (1/2)

- Radiation therapy machine used in the 1980s
- Involved in accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
  - Approximately 100 times the intended dose!
  - Numerous causes, including race condition

“The failure only occurred when a particular nonstandard sequence of keystrokes was entered on the VT-100 terminal which controlled the PDP-11 computer: an x to (erroneously) select 25MV photon mode followed by ↑, E to (correctly) select 25 MeV Electron mode, then Enter, all within eight seconds.”
The Therac-25 radiation therapy machine (2/2)

The testing engineers could obviously not detect this strange (and quick!) sequence leading to the failure.
The testing engineers could obviously not detect this strange (and quick!) sequence leading to the failure.

Limits of testing
This case illustrates the difficulty of bug detection without formal methods.
Bugs can be difficult to find

...and can have dramatic consequences for critical systems:

- health-related devices
- aeronautics and aerospace transportation
- smart homes and smart cities
- military devices
- etc.
Bugs can be difficult to find

...and can have dramatic consequences for critical systems:

- health-related devices
- aeronautics and aerospace transportation
- smart homes and smart cities
- military devices
- etc.

Hence, high need for formal verification
Outline

1. Finite-state automata

2. Timed automata

3. Parametric timed automata

4. Modeling and verifying real-time systems with parameters

5. A case study: Verifying a real-time system under uncertainty

6. Conclusion and perspectives
Outline: Finite-state automata

1. Finite-state automata
2. Timed automata
3. Parametric timed automata
4. Modeling and verifying real-time systems with parameters
5. A case study: Verifying a real-time system under uncertainty
6. Conclusion and perspectives
Outline: Finite-state automata

1 Finite-state automata

2 Timed automata

3 Parametric timed automata

4 Modeling and verifying real-time systems with parameters

5 A case study: Verifying a real-time system under uncertainty

6 Conclusion and perspectives
Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]

A model of the system

A property to be satisfied

\( Q_1 \rightarrow Q_2 \rightarrow Q_3 \)

is unreachable
Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]

A model of the system

A property to be satisfied

Question: does the model of the system satisfy the property?
Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]

A model of the system

A property to be satisfied

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

Yes

No

Counterexample
Model checking concurrent systems

- Use formal methods [Baier and Katoen, 2008]

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
Finite state automata: A coffee machine $A_C$

- Waiting
- Adding sugar
- Delivering coffee

Example of runs
- Coffee with no sugar
Finite state automata: A coffee machine $A_C$

Example of runs

- Coffee with no sugar
  - press?
  - cup!
  - coffee!

- Coffee with 2 doses of sugar
Finite state automata: A coffee machine $A_C$

- **Example of runs**
  - **Coffee with no sugar**
  - **Coffee with 2 doses of sugar**
  - And so on
Outline: Timed automata

1 Finite-state automata

2 Timed automata
   - Introduction
   - Syntax
   - Semantics
   - Decision problems
   - Software

3 Parametric timed automata

4 Modeling and verifying real-time systems with parameters

5 A case study: Verifying a real-time system under uncertainty
Beyond finite state automata

Finite State Automata give a powerful syntax and semantics to model qualitative aspects of systems

- Executions, sequence of actions
- Modular definitions (parallelism)
- Powerful checking (reachability, safety, liveness...)

Time (the airbag always eventually inflates, but may be 10 seconds after the crash)

Temperature (the alarm always eventually rings, but may be when the temperature is above 75 degrees)
Beyond finite state automata

Finite State Automata give a powerful syntax and semantics to model qualitative aspects of systems

- Executions, sequence of actions
- Modular definitions (parallelism)
- Powerful checking (reachability, safety, liveness...)

But what about quantitative aspects:

- Time ("the airbag always eventually inflates, but maybe 10 seconds after the crash")
- Temperature ("the alarm always eventually ring, but maybe when the temperature is above 75 degrees")
Timed automaton (TA)

- Finite state automaton (sets of locations)
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

```
x := 0
y := 0
y = 5
```
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

```plaintext
press? : x := 0
y := 0

y := 5

cup!

press?

coffee!

cup!
```
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

\[
\begin{align*}
  y &\leq 5 \\
  y &\leq 8
\end{align*}
\]
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

\[
\begin{align*}
x &\geq 1 \\
y &\leq 5 \\
\text{press?} \\
y &\leq 8 \\
x &\geq 1 \\
y &\leq 8 \\
\text{cup!} \\
y &\leq 8 \\
\text{press?} \\
y &\leq 5 \\
\end{align*}
\]
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 at each transition

$$y = 8$$

coffee!

```
x := 0
y := 0
```

```
x ≥ 1
press?
```

```
x := 0
press?
```

```
y = 5
coffee!
```
Concrete semantics of timed automata

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

  Example: $(\text{blue}$, $(x=1.2\, y=3.7))$

- **Concrete run**: alternating sequence of concrete states and actions or time elapse
Example of concrete runs

- Possible concrete runs for the coffee machine
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

x = 0
y = 0
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

\[
\begin{align*}
x & = 0 \\
y & = 0
\end{align*}
\]
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

\[
\begin{align*}
    x &= 0 & y &= 0 & \text{press?} & x &:= 0 \\
    x &= 0 & y &= 5 & \text{cup!} & x &:= 0 \\
    x &= 0 & y &= 5 & \text{press?} & x &:= 0 \\
    x &= 0 & y &= 5 & \text{cup!} & x &:= 0 \\
\end{align*}
\]
Example of concrete runs

Possible concrete runs for the coffee machine

Coffee with no sugar
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

\[
\begin{align*}
x &= 0 \\
y &= 0
\end{align*}
\]

\[
\begin{align*}
x &= 0 \\
y &= 0 \\
x &= 5 \\
y &= 5 \\
x &= 3 \\
y &= 8
\end{align*}
\]
Example of concrete runs

Possible concrete runs for the coffee machine

Coffee with no sugar

\[ x = 0 \quad 0 \quad 0 \quad 5 \quad 5 \quad 8 \quad 8 \]
\[ y = 0 \quad 0 \quad 5 \quad 5 \quad 8 \quad 8 \]
**Example of concrete runs**

Possible concrete runs for the coffee machine

- **Coffee with no sugar**

- **Coffee with 2 doses of sugar**
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no 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>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- 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>0</td>
</tr>
</tbody>
</table>
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no 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>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- 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>1.5</td>
</tr>
<tr>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>2.7</td>
<td>0</td>
</tr>
</tbody>
</table>
Example of concrete runs

- **Possble concrete runs for the coffee machine**

  - **Coffee with no sugar**
    ```
    x = 0 0 5 5 8 8
    y = 0 0 5 5 8 8
    ```

  - **Coffee with 2 doses of sugar**
    ```
    x = 0 0 1.5 0 2.7 0 0.8
    y = 0 0 1.5 1.5 4.2 4.2 5
    ```

---

\[ y \leq 5 \]

\[ y \leq 8 \]
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Example of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Example of concrete runs

Possibile concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Dense time

- Time is dense: transitions can be taken anytime
  - Infinite number of timed runs
  - Model checking needs a finite structure!

- Some runs are equivalent
  - Taking the press? action at $t = 1.5$ or $t = 1.57$ is equivalent w.r.t. the possible actions

- Idea: reason with abstractions
  - Region automaton [Alur and Dill, 1994], and zone automaton
  - Example: in location $\square$, all clock values in the following zone are equivalent
    \[ y \leq 5 \land y - x \geq 4 \]
  - This abstraction is finite
Symbolic states for timed automata

- **Objective**: group all concrete states reachable by the same sequence of discrete actions
- **Symbolic state**: a location \( l \) and a (infinite) set of states \( Z \)
- For timed automata, \( Z \) can be represented by a **convex polyhedron** with a special form called **zone**, with constraints

\[
-d_{0i} \leq x_i \leq d_{i0} \text{ and } x_i - x_j \leq d_{ij}
\]

- Computation of successive reachable symbolic states can be performed **symbolically** with polyhedral operations: for edge \( e = (l, a, g, R, l') \):

\[
\text{Succ}((l, Z), e) = (l', [(Z \cap g)]_R \cap I(l')) \cap I(l')
\]

- With an additional technicality, there is a **finite number** of reachable zones in a TA.
Symbolic states for timed automata: Example

\[ y \leq 4 \quad \text{and} \quad x \geq 2 \]

\[ y := 0 \]

\[ Z_0 = \{(0,0)\} \cup \uparrow \cap I \]

\[ \{(0,0)\} \]

\[ \chi \]

\[ y \]

\[ \chi \geq 2 \]
Symbolic states for timed automata: Example

\[ y \leq 4 \quad \Rightarrow \quad x \geq 2 \quad \Rightarrow \quad y := 0 \]

\[ Z_0 = \{(0, 0)\} \cup I \]

\[ y \]
Symbolic states for timed automata: Example

\[ y \leq 4 \quad \text{and} \quad x \geq 2 \]

\[ y := 0 \]

\[ Z_0 = \{(0, 0)\} \cap I \]

\[ Z_0 \]
Symbolic states for timed automata: Example

\[ y \leq 4 \quad \text{and} \quad x \geq 2 \]

\[ y := 0 \]

\[ Z_0 = \{(0,0)\} \uparrow \cap I \]

\[ Z_0 \cap (x \geq 2) \]
Symbolic states for timed automata: Example

\[ y \leq 4 \quad \Rightarrow \quad \chi \geq 2 \quad \Rightarrow \quad y := 0 \]

\[ Z_0 = \{(0, 0)\} \cap I \]

\[ \{(Z_0 \cap (x \geq 2))\}_{y} \]
Symbolic states for timed automata: Example

\[ y \leq 4 \quad \text{and} \quad x \geq 2 \]

\[ y := 0 \]

\[ Z_0 = \{(0, 0)\} \cap I \]

\[ Z_1 = [(Z_0 \cap (x \geq 2)) \downarrow_y] \]
Abstract semantics of timed automata

- Abstract state of a TA: pair \((l, C)\), where
  - \(l\) is a location, and \(C\) is a constraint on the clocks ("zone")
Abstract semantics of timed automata

- **Abstract state** of a TA: pair \((l, C)\), where
  - \(l\) is a location, and \(C\) is a constraint on the clocks ("zone")
- **Abstract run**: alternating sequence of abstract states and actions
Abstract semantics of timed automata

- Abstract state of a TA: pair \((l, C)\), where
  - \(l\) is a location, and \(C\) is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions
- Example

\[
\begin{align*}
x & \geq 1 \\
y & := 0
\end{align*}
\]

\[
\begin{align*}
& \quad x \leq 3 \\
& \quad a \\
& \quad y := 0
\end{align*}
\]

\[
\begin{align*}
x & \leq 4 \\
& \quad b \\
x & := 0
\end{align*}
\]

\[
\begin{align*}
y & \geq 3 \\
c
\end{align*}
\]

Possible abstract run for this TA

\[
0 \leq x \leq 3 \\
\land \quad x = y
\]
Abstract semantics of timed automata

- **Abstract state** of a TA: pair \((l, C)\), where
  - \(l\) is a location, and \(C\) is a constraint on the clocks ("zone")
- **Abstract run**: alternating sequence of abstract states and actions

**Example**

\(x \geq 1\)

\(y := 0\)

\(x \leq 3\)

\(x \leq 4\)

\(b\)

\(x := 0\)

\(y \geq 3\)

\(c\)

Possible abstract run for this TA

\(0 \leq x \leq 3\)

\(\land x = y\)

\(1 \leq x \leq 4\)

\(\land 1 \leq x - y \leq 3\)
Abstract semantics of timed automata

- **Abstract state** of a TA: pair \((l, C)\), where
  - \(l\) is a location, and \(C\) is a constraint on the clocks (“zone”)
- **Abstract run**: alternating sequence of abstract states and actions

**Example**

\[
\begin{align*}
  &x \geq 1 \\
  &y := 0
\end{align*}
\]

\[
\begin{align*}
  &x \leq 3 \\
  &a
\end{align*}
\]

\[
\begin{align*}
  &x \leq 4 \\
  &b \\
  &x := 0 \\
  &y \geq 3 \\
  &c
\end{align*}
\]

**Possible abstract run for this TA**

\[
\begin{align*}
  &0 \leq x \leq 3 \\
  \land \ &x = y \\
  &1 \leq x \leq 4 \\
  \land \ &1 \leq x - y \leq 3 \\
  &y \geq 0 \\
  \land \ &1 \leq y - x \leq 4
\end{align*}
\]
Abstract semantics of timed automata

- Abstract state of a TA: pair \((l, C)\), where
  - \(l\) is a location, and \(C\) is a constraint on the clocks ("zone")
- Abstract run: alternating sequence of abstract states and actions

Example

Possible abstract run for this TA
What is decidability?

**Definition**

A decision problem is **decidable** if one can design an algorithm that, for any input of the problem, can answer **yes** or **no** (in a finite time, with a finite memory).
What is decidability?

Definition

A decision problem is **decidable** if one can design an algorithm that, for any input of the problem, can answer **yes** or **no** (in a finite time, with a finite memory).

“given three integers, is one of them the product of the other two?”

“given a timed automaton, does there exist a run from the initial state to a given location \( l \)?”

“given a context-free grammar, does it generate all strings?”

“given a Turing machine, will it eventually halt?”
What is decidability?

**Definition**

A decision problem is **decidable** if one can design an algorithm that, for any input of the problem, can answer *yes* or *no* (in a finite time, with a finite memory).

√ “given three integers, is one of them the product of the other two?”

“given a timed automaton, does there exist a run from the initial state to a given location \( l \)?”

“given a context-free grammar, does it generate all strings?”

“given a Turing machine, will it eventually halt?”
What is decidability?

Definition

A decision problem is **decidable** if one can design an algorithm that, for any input of the problem, can answer *yes* or *no* (in a finite time, with a finite memory).

- "given three integers, is one of them the product of the other two?"
- "given a timed automaton, does there exist a run from the initial state to a given location \( l \)?"
- "given a context-free grammar, does it generate all strings?"
- "given a Turing machine, will it eventually halt?"
What is decidability?

**Definition**

A decision problem is **decidable** if one can design an algorithm that, for any input of the problem, can answer **yes** or **no** (in a finite time, with a finite memory).

√ “given three integers, is one of them the product of the other two?”

√ “given a timed automaton, does there exist a run from the initial state to a given location \( l \)?”

× “given a context-free grammar, does it generate all strings?”

“given a Turing machine, will it eventually halt?”
What is decidability?

**Definition**

A decision problem is **decidable** if one can design an algorithm that, for any input of the problem, can answer **yes** or **no** (in a finite time, with a finite memory).

- √ “given three integers, is one of them the product of the other two?”
- √ “given a timed automaton, does there exist a run from the initial state to a given location l?”
- × “given a context-free grammar, does it generate all strings?”
- × “given a Turing machine, will it eventually halt?”
Why studying decidability?

If a decision problem is **undecidable**, it is hopeless to look for algorithms yielding exact solutions (because that is **impossible**).
Why studying decidability?

If a decision problem is undecidable, it is hopeless to look for algorithms yielding exact solutions (because that is impossible)

However, one can:
- design semi-algorithms: if the algorithm halts, then its result is correct
- design algorithms yielding over- or under-approximations
Decision problems for timed automata

The finiteness of the region automaton allows us to check properties

- **Smiley**: Reachability of a location  
  **Alur and Dill, 1994**

- **Smiley**: Liveness (Büchi conditions)  
  **Alur and Dill, 1994**

Some problems impossible to check using the region automaton (but still **decidable**)

- **Smiley**: non-Zenoness emptiness check  
  **Wang et al., 2015**

Some **undecidable** problems (and hence impossible to check in general)

- **Sad**: universality of the timed language  
  **Alur and Dill, 1994**

- **Sad**: timed language inclusion  
  **Alur and Dill, 1994**
Software supporting timed automata

Timed automata have been successfully used since the 1990s

Tools for modeling and verifying models specified using TA

- HyTech (also hybrid, parametric timed automata) [Henzinger et al., 1997]
- Kronos [Yovine, 1997]
- TReX (also parametric timed automata) [Annicchini et al., 2001]
- UPPAAL [Larsen et al., 1997]
- Roméo (parametric time Petri nets) [Lime et al., 2009]
- PAT (also other formalisms) [Sun et al., 2009]
- IMITATOR (also parametric timed automata) [André et al., 2012]
Outline: Parametric timed automata

1. Finite-state automata

2. Timed automata

3. Parametric timed automata
   - Syntax
   - Semantics
   - Decidability results
   - L/U-PTAs

4. Modeling and verifying real-time systems with parameters

5. A case study: Verifying a real-time system under uncertainty
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** [Markey, 2011]: 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 [Markey, 2011]: 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

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...

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

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

```
press?
\[ x := 0 \]
\[ y := 0 \]

\[ y = 8 \]
coffee!

\[ y = 5 \]
cup!
press?
\[ x := 0 \]
```

\[ y \leq 5 \]

\[ y \leq 8 \]
Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set $P$ of parameters [Alur et al., 1993b]
  - Unknown constants compared to a clock in guards and invariants

\[
\begin{align*}
  y & = p_3 \\
  \text{coffee!} \\
  x & \geq p_1 \\
  \text{press?} \\
  x & := 0 \\
  y & := 0 \\
  y & \leq p_2 \\
  \text{press?} \\
  x & := 0 \\
  y & = p_2 \\
  \text{cup!} \\
  y & \leq 8
\end{align*}
\]
Valuation of a PTA

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

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

\[
\nu \left( \begin{array}{c}
\text{press?} \\
x := 0 \\
y := 0 \\
x := 0
\end{array} \right) \rightarrow \begin{array}{c}
y = p_2 \\
\text{cup!}
\end{array} \rightarrow \begin{array}{c}
y \leq p_2 \\
\text{coffee!}
\end{array} \rightarrow \begin{array}{c}
y = p_3 \\
\text{coffee!}
\end{array}
\]

\[
\nu \left( \begin{array}{c}
\text{press?} \\
x := 0 \\
y := 0 \\
x := 0
\end{array} \right) \rightarrow \begin{array}{c}
y = 5 \\
\text{cup!}
\end{array} \rightarrow \begin{array}{c}
y \leq 5 \\
\text{coffee!}
\end{array} \rightarrow \begin{array}{c}
y = 8 \\
\text{coffee!}
\end{array}
\]

with $\nu : \begin{cases} p_1 & \rightarrow & 1 \\ p_2 & \rightarrow & 5 \\ p_3 & \rightarrow & 8 \end{cases}$
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called **parametric zone** [Hune et al., 2002]
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

- **Symbolic run**: alternating sequence of symbolic states and actions

\[x \leq p_1, x \leq p_3, x \geq p_2, a = 0, b = 0, y \geq p_4, c\]

Example possible symbolic run for this PTA

\[x = y, x \leq p_1\]
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

- **Symbolic run**: alternating sequence of symbolic states and actions

- **Example**
  - \(x \geq p_2\)
  - \(a\)
  - \(y := 0\)
  - \(x \leq p_3\)
  - \(b\)
  - \(x := 0\)
  - \(y \geq p_4\)
  - \(c\)

- Possible symbolic run for this PTA
Symbolic semantics of parametric timed automata

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone [Hune et al., 2002]

- **Symbolic run**: alternating sequence of symbolic states and actions

**Example**

- Possible symbolic run for this PTA

\[
\begin{align*}
  x & \geq p_2 \\
  x & \leq p_1 \\
  a & \quad y := 0 \\
  x & \leq p_3 \\
  b & \quad x := 0 \\
  y & \geq p_4 \\
  c &
\end{align*}
\]
**Symbolic semantics of parametric timed automata**

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a convex polyhedron over \(X\) and \(P\) with a special form, called parametric zone

- **Symbolic run**: alternating sequence of symbolic states and actions

- **Example**

- Possible symbolic run for this PTA

\[ \begin{align*}
  x \geq p_2 \\
  a \\
  y := 0 \\
  \text{a} \\
  x \leq p_1 \\
  \text{b} \\
  x := 0 \\
  x \leq p_3 \\
  y \geq p_4 \\
  c \\
  y := 0 \\
  x \geq p_2 \\
  y \geq x \\
  x \leq p_3 \\
  y - x \leq p_3
\]
Symbolic semantics of PTA: Illustration

\[ C' = \left[ (C \cap g) \right]_R \cap I(l') \uparrow \cap I(l') \]
Symbolic semantics of PTA: Illustration

\[ C' = [(C \cap g)]_R \cap I(l') \cap I(l') \]
Symbolic semantics of PTA: Illustration

\[ C' = [(C \cap g)]_R \cap I(l') \uparrow \cap I(l') \]
Symbolic semantics of PTA: Illustration

\[ C' = [(C \cap g)]_R \cap I(l') \cap I(l') \]
Symbolic semantics of PTA: Illustration

\[ C' = [(C \cap g)]_R \cap I(l') \cap I(l') \]
Symbolic semantics of PTA: Illustration

\[ C' = \left[ (C \cap g) \right]_{R} \cap I(l') \cap I(l') \]
Symbolic exploration: Coffee machine

\[ y = p_3 \]

press?
\[ x := 0 \]
\[ y := 0 \]
cup!
\[ x := 0 \]
\[ y \leq p_2 \]

press?
\[ x \geq p_1 \]

\[ x = y \]
Symbolic exploration: Coffee machine

\[ y = p_3 \]
\[ \text{coffee!} \]

\[ y \leq p_2 \]

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

\[ x \geq p_1 \]
\[ \text{press?} \]
\[ x := 0 \]

\[ y = p_2 \]
\[ \text{cup!} \]

\[ 0 \leq y \leq p_2 \]

\[ x = y \]
Symbolic exploration: Coffee machine

\[ y = p_3 \]
\[ \text{coffee!} \]

\[ y \leq p_2 \]

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

\[ x \geq p_1 \]
\[ \text{press?} \]
\[ x := 0 \]

\[ y = p_2 \]
\[ \text{cup!} \]

\[ x = y \]
\[ 0 \leq y \leq p_2 \]

\[ x = y \]
\[ p_2 \leq y \leq 8 \]
Symbolic exploration: Coffee machine

\[ y = p_3 \]
\[ \text{coffee!} \]

\[ y \leq p_2 \]

\[ \text{press?} \]
\[ x := 0 \]
\[ y := 0 \]

\[ y = p_2 \]
\[ \text{cup!} \]

\[ x \geq p_1 \]
\[ \text{press?} \]
\[ x := 0 \]

\[ x = y \]

\[ 0 \leq y \leq p_2 \]

\[ p_2 \leq y \leq 8 \]

\[ p_2 \leq p_3 \leq 8 \]
\[ y = x + p_3 \]
Symbolic exploration: Coffee machine

\[ y = p_3 \]
\[ \text{coffee!} \]

\[ y \leq p_2 \]

\[ x \geq p_1 \]
\[ \text{cup!} \]

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

\[ x = y \]
\[ 0 \leq y \leq p_2 \]

\[ x = y \]
\[ p_2 \leq y \leq 8 \]

\[ x = y \]
\[ p_2 \leq p_3 \leq 8 \]
\[ y = x + p_3 \]

\[ \text{press?} \]

\[ \text{cup!} \]
\[ \text{coffee!} \]
Symbolic exploration: Coffee machine

\[ y = p_3 \]

\[ \text{coffee!} \]

\[ y \leq p_2 \]

press?

\[ x := 0 \]

\[ y := 0 \]

\[ x \geq p_1 \]

\[ \text{cup!} \]

\[ x := 0 \]

\[ x = y \]

\[ 0 \leq y \leq p_2 \]

\[ x = y \]

\[ 0 \leq y \leq p_2 \]

\[ p_2 \leq y \leq 8 \]

\[ y = x + p_3 \]

\[ p_2 \leq p_3 \leq 8 \]

press?

\[ y - x \geq p_1 \]

\[ 0 \leq y \leq p_2 \]

\[ \text{cup!} \]

\[ \cdot \cdot \cdot \]
Symbolic exploration: Coffee machine

\[
\begin{align*}
\text{press?} & \quad x := 0 \\
0 \leq y & \leq p_2 \\
\text{cup!} & \quad y = p_3 \\
\text{coffee!} & \quad x \geq p_1 \\
0 \leq y & \leq p_2 \\
\end{align*}
\]
Symbolic exploration: Coffee machine

\[ y = p_3 \]
\[ \text{coffee!} \]
Decision and computation problems for PTA

- **EF-Emptiness** “Is the set of parameter valuations for which a given location \( l \) is reachable empty?”
  
  **Example:** “Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?”

- **EF-Universality** “Do all parameter valuations allow to reach a given location \( l \)?”
  
  **Example:** “Are all parameter valuations such that I may eventually get a coffee?”

- **Preservation of the untimed language** “Given a parameter valuation, does there exist another valuation with the same untimed language?”
  
  **Example:** “Given the valuation \( p_1 = 1, p_2 = 5, p_3 = 8 \), do there exist other valuations with the same possible untimed behaviours?”
Decision and computation problems for PTA

- **EF-Emptiness** “Is the set of parameter valuations for which a given location \( l \) is reachable empty?”
  
  **Example:** “Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?”
  
  \( \sqrt{\text{yes}}, \text{ e.g., } p_1 = 1, p_2 = 5, p_3 = 8 \)

- **EF-Universality** “Do all parameter valuations allow to reach a given location \( l \)?”
  
  **Example:** “Are all parameter valuations such that I may eventually get a coffee?”

- **Preservation of the untimed language** “Given a parameter valuation, does there exist another valuation with the same untimed language?”
  
  **Example:** “Given the valuation \( p_1 = 1, p_2 = 5, p_3 = 8 \), do there exist other valuations with the same possible untimed behaviours?”
Decision and computation problems for PTA

- **EF-Emptiness** “Is the set of parameter valuations for which a given location $l$ is reachable empty?”
  
  Example: “Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?”
  
  \[ \sqrt{\quad}, \text{ e.g., } p_1 = 1, p_2 = 5, p_3 = 8 \]

- **EF-Universality** “Do all parameter valuations allow to reach a given location $l$?”
  
  Example: “Are all parameter valuations such that I may eventually get a coffee?”
  
  \[ \times, \text{ e.g., } p_1 = 1, p_2 = 5, p_3 = 2 \]

- **Preservation of the untimed language** “Given a parameter valuation, does there exist another valuation with the same untimed language?”
  
  Example: “Given the valuation $p_1 = 1, p_2 = 5, p_3 = 8$, do there exist other valuations with the same possible untimed behaviours?”
Decision and computation problems for PTA

- **EF-Emptiness** “Is the set of parameter valuations for which a given location \( l \) is reachable empty?”
  
  **Example:** “Does there exist at least one parameter valuation for which I can get a coffee with 2 sugars?”
  
  \( \sqrt{\text{, e.g., } p_1 = 1, p_2 = 5, p_3 = 8} \)

- **EF-Universality** “Do all parameter valuations allow to reach a given location \( l \)?”
  
  **Example:** “Are all parameter valuations such that I may eventually get a coffee?”
  
  \( \times, \text{ e.g., } p_1 = 1, p_2 = 5, p_3 = 2 \)

- **Preservation of the untimed language** “Given a parameter valuation, does there exist another valuation with the same untimed language?”
  
  **Example:** “Given the valuation \( p_1 = 1, p_2 = 5, p_3 = 8 \), do there exist other valuations with the same possible untimed behaviours?”
  
  \( \sqrt{\text{ }} \)
Computation problems for PTA

- **EF-Synthesis** “Find all parameter valuations for which a given location \( l \) is reachable”

  **Example:** “What are all parameter valuations such that one may eventually get a coffee?”

- and so on
Computation problems for PTA

- **EF-Synthesis** “Find all parameter valuations for which a given location \( l \) is reachable”
  
  **Example:** “What are all parameter valuations such that one may eventually get a coffee?”

\[
0 \leq p_2 \leq p_3 \leq 8
\]

- and so on
Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata
Undecidability

- The symbolic state space is infinite in general
- No finite abstraction exists like for timed automata

Bad news
All interesting problems are undecidable for (general) parametric timed automata.
Decidability results for PTA (1/2)

- **EF-emptiness problem**
  “Is the set of parameter valuations for which a given location $l$ is reachable empty?”
  **undecidable**  
  [Alur et al., 1993b]
Decidability results for PTA (1/2)

- **EF-emptiness** problem
  “Is the set of parameter valuations for which a given location $l$ is reachable empty?”
  undecidable \[\text{[Alur et al., 1993b]}\]

- **EF-universality** problem
  “Do all parameter valuations allow to reach a given location $l$?”
  undecidable \[\text{[André et al., 2016]}\]
Decidability results for PTA (1/2)

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

- **EF-universality** problem
  “Do all parameter valuations allow to reach a given location $l$?”
  undecidable
  [André et al., 2016]

- **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]
Decidability results for PTA (2/2)

- **AF-universality** problem

  “Do all parameter valuations allow to reach a given location $l$ for all runs?”

  undecidable  
  
  [André et al., 2016]
Decidability results for PTA (2/2)

- **AF-universality problem**
  “Do all parameter valuations allow to reach a given location $l$ for all runs?”
  
  *undecidable*  
  [André et al., 2016]

- **Preservation of the untimed language**
  “Given a parameter valuation, does there exist another valuations with the same untimed language?”
  
  *undecidable*  
  [André and Markey, 2015]
Decidability results for PTA (2/2)

- **AF-universality** problem
  “Do all parameter valuations allow to reach a given location \( l \) for all runs?”
  undecidable  
  [André et al., 2016]

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

In fact most interesting problems for PTAs are **undecidable**  
[André, 2017]
Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:
Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

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

Undecidability of EF-emptiness is achieved for a single parameter [Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

- ✓ 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]
### Limiting the number of clocks

Undecidability of EF-emptiness is achieved for a single parameter

[Miller, 2000, Beneš et al., 2015]

However, reducing the number of clocks yields decidability of the EF-emptiness problem:

- √ 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 lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter $p$ is always compared with clocks as an upper bound or always as a lower bound.

Lower-bound parameters:

Upped-bound parameters:
L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter $p$ is always compared with clocks as an upper bound or always as a lower bound.

Lower-bound parameters: $p_1, p_3$

Upped-bound parameters:
L/U-PTAs

Definition

A lower/upper bound PTA (L/U-PTA) is a PTA in which each parameter $p$ is always compared with clocks as an upper bound or always as a lower bound.

![Diagram of a PTA with parameters and conditions]

Lower-bound parameters: $p_1, p_3$

Upped-bound parameters: $p_2, p_4$
Decidable problems for L/U-PTA

- **EF-emptiness problem**
  
  “Does there exist a parameter valuation for which a given location \( l \) is reachable?”

  *decidable*  

  [Hune et al., 2002]
Decidable problems for L/U-PTA

- **EF-emptiness** problem
  “Does there exist a parameter valuation for which a given location \( l \) is reachable?”
  decidable [Hune et al., 2002]

- **EF-universality** problem
  “Do all parameter valuations allow to reach a given location \( l \)?”
  decidable [Bozzelli and La Torre, 2009]
Decidable problems for L/U-PTA

- **EF-emptiness** problem
  “Does there exist a parameter valuation for which a given location $l$ is reachable?”
  decidable
  
  [Hune et al., 2002]

- **EF-universality** problem
  “Do all parameter valuations allow to reach a given location $l$?”
  decidable

  [Bozzelli and La Torre, 2009]

- **EF-finiteness** problem
  “Is the set of parameter valuations allowing to reach a given location $l$ finite?”
  decidable (for integer valuations)

  [Bozzelli and La Torre, 2009]
Undecidable problems for L/U-PTA

- **AF-emptiness problem**
  “Does there exist a parameter valuation for which a given location $l$ is always eventually reachable?”
  undecidable  
  [Jovanović et al., 2015]
Undecidable problems for L/U-PTA

- **AF-emptiness** problem
  “Does there exist a parameter valuation for which a given location \( l \) is always eventually reachable?”
  \textit{undecidable} \hfill [Jovanović et al., 2015]

- **AF-universality** problem
  “Are all valuations such that a given location \( l \) is always eventually reachable?”
  \textit{undecidable (but…)} \hfill [André and Lime, 2017]
Undecidable problems for L/U-PTA

- **AF-emptiness** problem
  "Does there exist a parameter valuation for which a given location $l$ is always eventually reachable?"
  *undecidable*  
  [Jovanović et al., 2015]

- **AF-universality** problem
  "Are all valuations such that a given location $l$ is always eventually reachable?"
  *undecidable (but...)*  
  [André and Lime, 2017]

- **language preservation** emptiness problem
  "Given a parameter valuation $v$, can we find another valuation with the same untimed language?"
  *undecidable*  
  [André and Markey, 2015]
Outline: Modeling and verifying real-time systems with parameters

1. Finite-state automata
2. Timed automata
3. Parametric timed automata
4. Modeling and verifying real-time systems with parameters
   - Real-time systems
   - Modeling real-time systems under uncertainty
   - Verification
   - IMITATOR in a nutshell
5. A case study: Verifying a real-time system under uncertainty
Context: Hard real-time embedded systems

- Modern hard real-time embedded systems are distributed in nature
- Many of them have critical timing requirements:
  - automotive systems (modern cars have 10-20 embedded boards connected by one or more CAN bus)
  - avionics systems (several distributed control boards connected by one or more dedicated networks)

- To analyze the schedulability of such systems, it is very important to estimate the (worst-case) computation times of the tasks
- Estimating WCET is very difficult in modern architectures
Real-time pipelines

- Many real-time applications can be modeled as pipelines (also called transactions) of tasks

- Executed on a distributed (or multicore) system
- Activated cyclically (periodic or sporadic)
- Using preemption
  - Lower-priority tasks can be temporarily interrupted by a higher-priority task
Model

- A set of pipelines $\{P^{(1)}, \ldots, P^{(p)}\}$ distributed over $m$ nodes
- Each pipeline $P^{(i)}$ is a chain of $n_i$ tasks $\{\tau_{i,1}, \ldots, \tau_{i,n_i}\}$
- Pipeline $P^{(i)}$ has an end-to-end (E2E) deadline $D_i$ and period $T_i$

Scheduling Problem: Guarantee that all pipelines complete before their E2E deadlines
Activations, jitter, deadline

An example

<table>
<thead>
<tr>
<th>Task</th>
<th>Per.</th>
<th>E2E</th>
<th>Comp.</th>
<th>Resp.</th>
<th>Jitter</th>
<th>prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁,₁</td>
<td>15</td>
<td></td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>HI</td>
</tr>
<tr>
<td>τ₁,₂</td>
<td>-</td>
<td></td>
<td>3</td>
<td>8</td>
<td>0-2</td>
<td>LO</td>
</tr>
<tr>
<td>τ₁,₃</td>
<td>-</td>
<td>15</td>
<td>2</td>
<td>13</td>
<td>3</td>
<td>LO</td>
</tr>
<tr>
<td>τ₂,₁</td>
<td>12</td>
<td></td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>HI</td>
</tr>
<tr>
<td>τ₂,₂</td>
<td>-</td>
<td>12</td>
<td>6</td>
<td>?</td>
<td>0-1</td>
<td>ME</td>
</tr>
</tbody>
</table>

Node A
- τ₁,₁
- τ₁,₃
- τ₂,₂

Node B
- τ₁,₂
- τ₁,₃
- τ₂,₁
Activations, jitter, deadline

- An example

<table>
<thead>
<tr>
<th>Task</th>
<th>Per.</th>
<th>E2E</th>
<th>Comp.</th>
<th>Resp.</th>
<th>Jitter</th>
<th>prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_{1,1})</td>
<td>15</td>
<td></td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>HI</td>
</tr>
<tr>
<td>(\tau_{1,2})</td>
<td>-</td>
<td></td>
<td>3</td>
<td>8</td>
<td>0-2</td>
<td>LO</td>
</tr>
<tr>
<td>(\tau_{1,3})</td>
<td>-</td>
<td>15</td>
<td>2</td>
<td>13</td>
<td>3</td>
<td>LO</td>
</tr>
<tr>
<td>(\tau_{2,1})</td>
<td>12</td>
<td></td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>HI</td>
</tr>
<tr>
<td>(\tau_{2,2})</td>
<td>-</td>
<td>12</td>
<td>6</td>
<td>?</td>
<td>0-1</td>
<td>ME</td>
</tr>
</tbody>
</table>

Node A

- \(\tau_{1,1}\)
- \(\tau_{1,3}\)

Node B

- \(\tau_{1,2}\)
- \(\tau_{1,3}\)
- \(\tau_{2,1}\)
- \(\tau_{2,2}\)
Verifying real-time systems with parameters

Real-time systems

Activations, jitter, deadline

- An example

<table>
<thead>
<tr>
<th>Task</th>
<th>Per.</th>
<th>E2E</th>
<th>Comp.</th>
<th>Resp.</th>
<th>Jitter</th>
<th>prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_{1,1} )</td>
<td>15</td>
<td></td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>HI</td>
</tr>
<tr>
<td>( \tau_{1,2} )</td>
<td>-</td>
<td>15</td>
<td>3</td>
<td>8</td>
<td>0-2</td>
<td>LO</td>
</tr>
<tr>
<td>( \tau_{1,3} )</td>
<td>-</td>
<td></td>
<td>2</td>
<td>13</td>
<td>3</td>
<td>LO</td>
</tr>
<tr>
<td>( \tau_{2,1} )</td>
<td>12</td>
<td></td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>HI</td>
</tr>
<tr>
<td>( \tau_{2,2} )</td>
<td>-</td>
<td>12</td>
<td>6</td>
<td>14</td>
<td>0-1</td>
<td>ME</td>
</tr>
</tbody>
</table>

Node A

\( \tau_{1,1} \)

\( \tau_{1,2} \)

\( \tau_{1,3} \)

\( \tau_{2,2} \)

Node B

\( \tau_{2,1} \)

\( \tau_{1,1} \)

\( \tau_{2,2} \)

\( \tau_{1,3} \)
Enriching the model with parameters

- A task is identified by three parameters:
  - $C_i$ is the worst-case computation time (or worst-case transmission time, in case it models a message)
  - $R_i$ is the task worst-case response time, i.e., the worst case finishing time of any task instance relative to the activation of its pipeline.
  - $J_i$ is the task worst-case activation jitter, i.e., the greatest time since its activation that a task must wait for all preceding tasks to complete their execution

- A parameter of major interest is the computation time
Modeling a task / pipeline

\[ \tau_1 \text{ waiting urgent} \rightarrow \tau_1 \text{ release} \rightarrow \tau_1 \text{ released} \]

\[ \begin{align*}
\mathcal{P}_1 \text{ complete} \\
\chi_{\mathcal{P}_1} &\leq T_1 \\
\mathcal{P}_1 &\text{ restart} \\
\chi_{\mathcal{P}_1} &:= 0
\end{align*} \]

\[ \begin{align*}
\tau_1 \text{ completed} &\rightarrow \tau_2 \text{ waiting urgent} \\
\tau_2 \text{ released} &\rightarrow \tau_2 \text{ completed}
\end{align*} \]
Modeling the fixed priority scheduler (preemptive)

Actually a PTA extended with stopwatches [Sun et al., 2013]
an extension of stopwatch automata [Adbeddaïm and Maler, 2002]
Parametric verification of real-time systems

Many problems can be reduced to parametric reachability (EFsynth):
find parameter valuations for which a given state is (un)reachable

- This problem is undecidable for PTAs and many subclasses

- But we can still compute part (and often all) of the solution

Interesting problems:

- Find parameter valuations for which no deadline violation occurs
  (i.e., for which the system is schedulable)

- Compute the worst-case computation time

- Find the parametric WCET when the jitter is unknown
IMITATOR

- A tool for modeling and verifying real-time 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

- Verification
  - Computation of the symbolic state space
  - (non-Zeno) parametric model checking (using a subset of TCTL)
  - Language and trace preservation, and robustness analysis
  - Parametric deadlock-freeness checking
  - Behavioral cartography
IMITATOR

Under continuous development since 2008  [André et al., 2012]

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
  - Project ANR Valmem

- 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]

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

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

1 Finite-state automata

2 Timed automata

3 Parametric timed automata

4 Modeling and verifying real-time systems with parameters

5 A case study: Verifying a real-time system under uncertainty

6 Conclusion and perspectives
The FMTV 2015 Challenge (1/2)

Challenge by Thales proposed during the WATERS 2014 workshop
Solutions presented at WATERS 2015

System: an unmanned aerial video system with uncertain periods
- Period constant but with a small uncertainty (typically 0.01 %)
- Not a jitter!
The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$
The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of \( n = 1 \) and \( n = 3 \)

😊 Not a typical parameter synthesis problem?

- No parameters in the specification
The FMTV 2015 Challenge (2/2)

Goal

Compute the end-to-end BCET and WCET times for a buffer size of $n = 1$ and $n = 3$.

😊 Not a typical parameter synthesis problem?
- No parameters in the specification

😊 A typical parameter synthesis problem
- The end-to-end time can be set as a parameter... to be synthesized
- The uncertain period is typically a parameter (with some constraint, e.g., $P_1 \in [40 - 0.004, 40 + 0.004]$)
Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time.
Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time

2. Add a specific location corresponding to the correct transmission of the frame
Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.
Methodology

1. Propose a PTA model with parameters for uncertain periods and the end-to-end time
2. Add a specific location corresponding to the correct transmission of the frame
3. Run the reachability synthesis algorithm EFsynth (implemented in IMITATOR) w.r.t. that location
4. Gather all constraints (in as many dimensions as uncertain periods + the end-to-end time)
5. Eliminate all parameters but the end-to-end time
6. Exhibit the minimum and the maximum

Note: not eliminating parameters allows one to know for which values of the periods the best / worst case execution times are obtained.
To build the PTA model

- Uncertainties in the system:
  - P1 $\in [40 - 0.004, 40 + 0.004]$
  - P3 $\in \left[ \frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150} \right]$
  - P4 $\in [40 - 0.004, 40 + 0.004]$
To build the PTA model

- **Uncertainties in the system:**
  - \( P_1 \in [40 - 0.004, 40 + 0.004] \)
  - \( P_3 \in \left[ \frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150} \right] \)
  - \( P_4 \in [40 - 0.004, 40 + 0.004] \)

- **Parameters:**
  - \( P_{1\text{ uncertain}} \)
  - \( P_{3\text{ uncertain}} \)
  - \( P_{4\text{ uncertain}} \)
To build the PTA model

- **Uncertainties in the system:**
  - \( P_1 \in [40 - 0.004, 40 + 0.004] \)
  - \( P_3 \in \left[ \frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150} \right] \)
  - \( P_4 \in [40 - 0.004, 40 + 0.004] \)

- **Parameters:**
  - \( P_1\_\text{uncertain} \)
  - \( P_3\_\text{uncertain} \)
  - \( P_4\_\text{uncertain} \)

- The end-to-end latency (another parameter): \( \text{E2E} \)
To build the PTA model

- Uncertainties in the system:
  - $P_1 \in [40 - 0.004, 40 + 0.004]$  
  - $P_3 \in \left[\frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150}\right]$  
  - $P_4 \in [40 - 0.004, 40 + 0.004]$  

- Parameters:
  - $P_{1\text{._uncertain}}$  
  - $P_{3\text{._uncertain}}$  
  - $P_{4\text{._uncertain}}$

- The end-to-end latency (another parameter): $E_{2E}$

- Others:
  - the register between task 2 and task 3: discrete variable $\text{reg}_{2,3}$  
  - the buffer between task 3 and task 4: $n = 1$ or $n = 3$
Simplification

- T1 and T2 are synchronised; T1, T3 and T4 are asynchronised
  - (exact modeling of the system behaviour is too heavy)
Simplification

- T1 and T2 are synchronised; T1, T3 and T4 are asynchronised
  - (exact modeling of the system behaviour is too heavy)

- We choose a single arbitrary frame, called the target one

- We assume the system is initially in an arbitrary status
  - This is our only uncertain assumption (in other words, can the periods deviate from each other so as to yield any arbitrary deviation?)
The initialization automaton

\[ \text{ckT1T2} = \text{WCET}_1 \]
The initialization automaton

\[ ckT1T2 = WCET_1 \]

\[
\begin{align*}
\text{camera0} & \quad \text{camera1} \\
\text{buffer}_{3,4} & := 0 \\
\text{highest}_{3,4} & := 0 \\
\text{buffer}_{3,4} & := 1 \\
\text{highest}_{3,4} & := 1
\end{align*}
\]

Verifying a real-time system under uncertainty
The initialization automaton

\[ \text{ckT1T2} = \text{WCET}_1 \]

\[ \begin{align*}
\text{buffer}_{3,4} & := 0 \\
\text{highest}_{3,4} & := 0
\end{align*} \]

\[ \begin{align*}
\text{buffer}_{3,4} & := 1 \\
\text{highest}_{3,4} & := 1
\end{align*} \]

\[ \text{frame_in}_{3} := 0 \quad \text{frame_in}_{3} := 2 \]
The initialization automaton

\[ \text{camera0} \quad \text{camera1} \quad \text{camera2} \quad \text{camera3} \]

\[ \text{buffer}_{3,4} := 0 \]
\[ \text{highest}_{3,4} := 0 \]

\[ \text{buffer}_{3,4} := 1 \]
\[ \text{highest}_{3,4} := 1 \]

\[ \text{frame\_in\_3} := 0 \quad \text{frame\_in\_3} := 2 \]

\[ \text{reg}_{2,3} := 0 \quad \text{reg}_{2,3} := 3 \]
The initialization automaton

\[ \text{ckT1T2} = \text{WCET}_1 \]

- camera0
  - buffer\(_{3,4}\) := 0
  - highest\(_{3,4}\) := 0

- camera1
  - buffer\(_{3,4}\) := 1
  - highest\(_{3,4}\) := 1
  - frame\(_{in\_3}\) := 0
  - frame\(_{in\_3}\) := 2

- camera2
  - reg\(_{2,3}\) := 0
  - reg\(_{2,3}\) := 3

- camera3

- start

- \text{T1T2: WCET}_1 + WCL_2 \geq \text{ckT1T2} \]
The initialization automaton

\[ ckT1T2 = WCET_1 \]

\[ \text{camera0} \]

\[ \text{camera1} \]

\[ \text{camera2} \]

\[ \text{camera3} \]

\[ \text{start} \]

\[ \text{T1T2done} \]

\[ \text{T1T2} \]

\[ \text{reg}_{2,3} := 0 \]

\[ \text{buffer}_{3,4} := 0 \]

\[ \text{highest}_{3,4} := 0 \]

\[ \text{frame}_\text{in}_3 := 0 \]

\[ \text{frame}_\text{in}_3 := 2 \]

\[ \text{reg}_{2,3} := 3 \]

\[ \text{reg}_{2,3} := \text{target} \]

\[ \text{ckT1T2} \geq WCET_1 + BCL_2 \]

\[ T2\text{done} \]

\[ \text{ckT1T2} \geq WCET_1 + WCL_2 \geq \text{ckT1T2} \]
Task T3
Task T3

- **T3preinit**
  - \( WCET_3 \geq ckT_3 \) or **start**

- **T3process**
  - \( WCET_3 \geq ckT_3 \)
Task T3

\[ WCET_3 \geq ckT_3 \]

\[ WCET_3 \geq ckT_3 \]

\[ P_{3 \text{ uncertain}} \geq ckT_3 \]
Task T3

\[ \text{WCET}_3 \geq \text{ckT}_3 \]

\[ \text{P}_3\_\text{uncertain} = \text{ckT}_3 \]

\[ \text{T}_3 \text{ start} \]

\[ \text{ckT}_3 := 0 \]

\[ \text{frame\_in\_3} := \text{reg}_{2,3} \]

\[ \text{P}_3\_\text{uncertain} \geq \text{ckT}_3 \]
Task T3

\[ T_3 \text{preinit} \]

\[ WCET_3 \geq ckT_3 \text{start} \]

\[ T_3 \text{process} \]

\[ WCET_3 \geq ckT_3 : \]

\[ P_{3\_uncertain} = ckT_3 \]
\[ T_3 \text{ start} \]
\[ ckT_3 := 0 \]
\[ \text{frame\_in\_3} := \text{reg}_{2,3} \]

\[ WCET_3 = ckT_3 \]
\[ \land \text{buffer}_{3,4} = 0 \]
\[ \land \text{frame\_in\_3} > \]
\[ \text{highest}_{3,4} \]
\[ T_3 \text{ done} \]
\[ \text{write\_by\_T3}() \]

\[ T_3 \text{wait} \]

\[ P_{3\_uncertain} \geq ckT_3 : \]
Task T3

\[ WCET_3 \geq \text{ckT}_3 \]

\[ \text{start} \]

\[ \text{T3preinit} \]

\[ \text{T3process} \]

\[ WCET_3 \geq \text{ckT}_3 \]

\[ P_3\_\text{uncertain} = \text{ckT}_3 \]
\[ T_3\_\text{start} \]
\[ \text{ckT}_3 := 0 \]
\[ \text{frame\_in\_3} := \text{reg}_{2,3} \]

\[ \text{write\_by\_T3()} \]

\[ \text{T3\_wait} \]

\[ P_3\_\text{uncertain} \geq \text{ckT}_3 \]

\[ \text{start} \]
Task T3

\[ \text{WCET}_3 \geq \text{ckT}_3 \]

\[ \text{P3\_uncertain} = \begin{cases} \text{ckT}_3 & \text{buffer}_{3,4} = 0 \\ \text{ckT}_3 & \text{highest}_{3,4} \geq \text{frame\_in\_3} \end{cases} \]

\[ \text{WCET}_3 = \begin{cases} \text{ckT}_3 & \text{buffer}_{3,4} = 0 \\ \text{frame\_in\_3} > \text{highest}_{3,4} & \text{T3\_done} \end{cases} \]

\[ \text{write\_by\_T3}() \]

\[ \text{T3\_done} \]

\[ \text{P3\_uncertain} \geq \text{ckT}_3 \]

\[ \text{WCET}_3 \geq \text{ckT}_3 \]

\[ \text{T3\_preinit} \]

\[ \text{T3\_process} \]

\[ \text{T3\_wait} \]
Task T4

P_{4\_uncertain} \geq c k_{T4}
Task T4

\[ P_{4\_uncertain} = ckT4 \]
\[ \land buffer_{3,4} > 0 \]
\[ ckT4 := 0 \]
\[ \text{read\_by\_T4()} \]

\[ 10 \geq ckT4 \]
Task T4

\[ P_{4\_uncertain} = \text{ckT4} \]
\[ \land \text{buffer}_{3,4} = 0 \]
\[ \text{ckT4} := 0 \]

\[ P_{4\_uncertain} = \text{ckT4} \]
\[ \land \text{buffer}_{3,4} > 0 \]
\[ \text{ckT4} := 0 \]
\[ \text{read\_by\_T4()} \]

\[ P_{4\_uncertain} = \text{ckT4} \]
\[ \land \text{frame\_in\_4} \neq \text{target} \]

\[ \text{T4end\_ok} \]
\[ \text{ckT4} := 0 \]

\[ 10 \geq \text{ckT4} \]
Verifying a real-time system under uncertainty

Task T4

The PTA model for $n = 1$

\[
P_4_{\text{uncertain}} = \text{c}kT_4 \\
\land \text{buffer}_{3,4} = 0 \\
\text{ckT}_4 := 0
\]

\[
P_4_{\text{uncertain}} = \text{c}kT_4 \\
\land \text{buffer}_{3,4} > 0 \\
\text{ckT}_4 := 0 \\
\text{read}_\text{by}_T_4() \\
\]

\[
P_4_{\text{uncertain}} \geq \text{ckT}_4 \\
10 = \text{ckT}_4 \\
\land \\
\text{frame}_\text{in}_4 \neq \text{target}
\]

\[
\text{ckT}_4 := 0 \\
\text{T4wait} \xrightarrow{10 \geq \text{ckT}_4} \text{T4process}_\text{nonempty} \\
\]
Verifying a real-time system under uncertainty

The PTA model for $n = 1$

**Task T4**

- $P_{4\_uncertain} = ckt_{4}$
  - $\wedge buffer_{3,4} = 0$
  - $ckt_{4} := 0$

- $P_{4\_uncertain} = ckt_{4}$
  - $\wedge buffer_{3,4} > 0$
  - $ckt_{4} := 0$
  - $\text{read\_by\_T4()}$

- $P_{4\_uncertain} \geq ckt_{4}$
  - $\wedge \text{frame\_in\_4} \neq \text{target}$

- $10 = ckt_{4}$
  - $\wedge \text{frame\_in\_4} = \text{target}$
  - $ckt_{1}T_{2} = \text{E2E}$
  - $ckt_{4} := 0$

- $10 \geq ckt_{4}$
  - $ckt_{4} = 0$
Results

E2E latency results for $n = 1$ and $n = 3$

<table>
<thead>
<tr>
<th></th>
<th>$n = 1$</th>
<th>$n = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>min E2E</td>
<td>63 ms</td>
<td>63 ms</td>
</tr>
<tr>
<td>max E2E</td>
<td>145.008 ms</td>
<td>225.016 ms</td>
</tr>
</tbody>
</table>

Results obtained using IMITATOR in a few seconds
Outline

1. Finite-state automata
2. Timed automata
3. Parametric timed automata
4. Modeling and verifying real-time systems with parameters
5. A case study: Verifying a real-time system under uncertainty
6. Conclusion and perspectives
Summary

- **Finite-state automata**
  - 😊 Mostly decidable results
  - 😊 Efficient model checking algorithms
  - 😞 Miss the quantitative aspects
  - 😊 Many powerful tools

- **Timed automata**
  - 😊 Finite abstract semantics
  - 😊 Some decidable results
  - 😞 Some undecidable results
  - 😊 Several powerful tools

- **Parametric timed automata**
  - 😊 Very expressive
  - 😞 No finite abstract semantics
  - 😞 Almost only undecidability results
  - 😊 Some powerful tools
Perspectives

Address harder problems

- Thales challenge: what is the minimum time between two lost frames? (due to the uncertain periods)
  - Requires to model check thousands of frame processings

Improve the efficiency of parameter synthesis techniques

- Promising heuristics: approximations using the integer hull
  [Jovanović et al., 2015, André et al., 2015]

- Distributed parameter synthesis
  - Multi-core synthesis
  - Distributed synthesis based on locations
  [Laarman et al., 2013]
  [Zhang et al., 2016]
Beyond (parametric) timed automata

Beyond time...

- Cost, temperature, energy
  - Hybrid automata [Alur et al., 1993a, Alur et al., 1995]
    - Very expressive, but often undecidable
    - Some interesting software (including SpaceEx [Frehse et al., 2011])

Probabilities

- Useful when a property cannot be proved with full certainty
- Security
- Another way to model systems known with limited precision
Bibliography
References I

Preemptive job-shop scheduling using stopwatch automata.
In TACAS, volume 2280 of LNCS, pages 113–126. Springer-Verlag.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X.,
The algorithmic analysis of hybrid systems.

Hybrid automata: An algorithmic approach to the specification and verification of
hybrid systems.
In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, Hybrid Systems

A theory of timed automata.

Parametric real-time reasoning.
In STOC, pages 592–601. ACM.
References II

André, É. (2017).
What’s decidable about parametric timed automata?
*International Journal on Software Tools for Technology Transfer.*
To appear.

IMITATOR 2.5: A tool for analyzing robustness in scheduling problems.

Liveness in L/U-parametric timed automata.
In Legay, A. and Schneider, K., editors, *ACSD*, pages 9–18. IEEE.
To appear.

Integer-complete synthesis for bounded parametric timed automata.
In *RP*, volume 9058 of *LNCS*. Springer.

Decision problems for parametric timed automata.
Language preservation problems in parametric timed automata.
In *FORMATS*, volume 9268 of *LNCS*, pages 27–43. Springer.

TReX: A tool for reachability analysis of complex systems.

*Principles of Model Checking*.
MIT Press.

Language emptiness of continuous-time parametric timed automata.

Decision problems for lower/upper bound parametric timed automata.
References IV

Advances in parametric real-time reasoning.
In MFCS, volume 8634 of LNCS, pages 123–134. Springer.

Formal timing analysis of mixed music scores.
In ICMC (International Computer Music Conference).

Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R.,
SpaceEx: Scalable verification of hybrid systems.
In Gopalakrishnan, G. and Qadeer, S., editors, CAV, volume 6806 of LNCS, pages

Robustness analysis for scheduling problems using the inverse method.

HyTech: A model checker for hybrid systems.


References VI

Robustness in real-time systems.

Decidability and complexity results for timed automata and semi-linear hybrid automata.
In HSCC, volume 1790 of LNCS, pages 296–309. Springer.

PAT: Towards flexible verification under fairness.
In CAV, volume 5643 of LNCS, pages 709–714. Springer.

Parametric schedulability analysis of fixed priority real-time distributed systems.
In FTSCS, volume 419 of CCIS, pages 212–228. Springer.

A systematic study on explicit-state non-zenoness checking for timed automata.
References VII

Kronos: A verification tool for real-time systems.

Distributed algorithms for time optimal reachability analysis.
Additional explanation
Explanation for the 4 pictures in the beginning

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

2. Error screen on the earliest versions of Macintosh

3. 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)

4. 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 I

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
Source of the graphics used II

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: Renault Twizy
Author: Citron
Source: https://commons.wikimedia.org/wiki/File:Renault_Twizy.jpg
License: CC BY-SA 3.0

Title: Airbus A380
Author: Axwel
Source: https://commons.wikimedia.org/wiki/File:Airbus_A380_blue_sky.jpg
License: CC BY 2.0
Source of the graphics used III
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/