Contributions to parametric timed model checking:

Theory and algorithms

Étienne André

LIPN, Université Paris 13, CNRS, France
Context: Critical real-time systems

- Real-time systems are everywhere
  - Hard **timing** constraints and **concurrency**
  - Criticality: risk for huge damages in case of unexpected behavior (**bug**)
Context: Critical real-time systems

- Real-time systems are everywhere
  - Hard timing constraints and concurrency
  - Criticality: risk for huge damages in case of unexpected behavior (bug)

- Verification to ensure absence of bugs is required

- Common techniques
  - Testing
  - Abstract interpretation
  - Theorem proving
  - Model checking
Context: Critical real-time systems

■ Real-time systems are everywhere
  ■ Hard **timing** constraints and **concurrency**
  ■ Criticality: risk for huge damages in case of unexpected behavior (**bug**)

■ **Verification** to ensure absence of bugs is required

■ Common techniques
  ■ Testing
  ■ Abstract interpretation
  ■ Theorem proving
  ■ Model checking
Model checking timed concurrent systems

- Use formal methods

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

A model of the system

[Baier and Katoen, 2008]

- is unreachable

A property to be satisfied

Étienne André
Contributions to parametric timed model checking
Model checking timed concurrent systems

- Use formal methods

![Diagram of a system model]

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

[Baier and Katoen, 2008]

A model of the system

A property to be satisfied

? 

is unreachable

[Étienne André]

Contributions to parametric timed model checking
Model checking timed concurrent systems

- Use formal methods

![Diagram of a model of the system]

A model of the system

- 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]
Timed automaton (TA)

- Finite state automaton (sets of locations)

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

idle
adding sugar
delivering coffee
Timed automaton (TA)

- Finite state automaton (sets of locations and actions)

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
Contributions to parametric timed model checking
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 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

---

$$y \leq 5$$

---

Étienne André
Contributions to parametric timed model checking
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
\text{press?}
```

\[
x \geq 1
\]

```
y = 8
coffee!
```

```
y = 5
\text{cup!}
```

```y \leq 8:```

---

```
idle
```

```
adding sugar
```

```
delivering coffee
```
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

---

$y \leq 5$
$y = 8$
$y \leq 8$
$x \geq 1$
$x := 0$
$y := 0$
$y := 0$
$x := 0$
$y := 0$
$y := 0$
$y := 0$
$x := 0$
$press?$
$coffee!$
$press?$
$cup!$

idle
adding sugar
delivering coffee

Étienne André
Contributions to parametric timed model checking
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with \( \frac{x}{2} \) doses of sugar

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

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

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

Example of concrete run for the coffee machine

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

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
</tr>
</tbody>
</table>

- 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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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

\[
\begin{align*}
x &= 0 & 0 & 1.5 & 0 & 2.7 \\
y &= 0 & 0 & 1.5 & 1.5 & 4.2
\end{align*}
\]
The most critical system: The coffee machine

![Diagram of a coffee machine with states and transitions]

- idle
- adding sugar
- delivering coffee

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>State</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1.5</td>
<td>0</td>
<td>1.5</td>
</tr>
<tr>
<td>2.7</td>
<td>0</td>
<td>4.2</td>
</tr>
<tr>
<td>4.2</td>
<td>0</td>
<td>4.2</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></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>2.7</td>
<td>0</td>
<td>0.8</td>
<td>5</td>
</tr>
</tbody>
</table>

x := 0
y := 0

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

\[
\begin{align*}
\text{press?} & \quad x = 0 & y = 0 \\
1.5 & \quad \text{press?} & 1.5 \\
2.7 & \quad \text{press?} & 4.2 \\
0 & \quad \text{press?} & 4.2 \\
0.8 & \quad \text{cup!} & 5 \\
0.8 & \quad \text{cup!} & 5
\end{align*}
\]
The most critical system: The coffee machine

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

\[ y \leq 5 \]

\[ x \geq 1 \gaben \text{cup!} \]

\[ x := 0 \]

\[ y := 0 \]

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[ x = \]
\[ y = \]

\[ x = 0 \]
\[ y = 0 \]
\[ x = 1.5 \]
\[ y = 1.5 \]
\[ x = 2.7 \]
\[ y = 4.2 \]
\[ x = 0 \]
\[ y = 4.2 \]
\[ x = 0.8 \]
\[ y = 5 \]
\[ x = 0.8 \]
\[ y = 5 \]
\[ x = 3 \]
\[ y = 3.8 \]
The most critical system: The coffee machine

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

Étienne André
Contributions to parametric timed model checking
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

[Alur and Dill, 1994]

[Larsen et al., 1997]
Need to allow for abstractions and uncertainty

- Need for abstraction
  - Constants known with limited certainty
  - Unknown constants

Challenging problems

- Existence (dually: emptiness): find one valuation for which a property holds
  - "Can I exhibit a valuation for which I am guaranteed to eventually get a coffee?"

- Synthesis: find some/all parameter valuations for which a property holds
  - "Synthesize all valuations for which I am guaranteed to eventually get a coffee with /two.osf sugars"
Need to allow for abstractions and uncertainty

- Need for abstraction

  - Constants known with limited certainty
  - Unknown constants

- Idea: reason with parameters (unknown constants)

  - Verify the system in presence of uncertain constants
  - Synthesize suitable valuations for unknown parameters
  - Optimize parameter valuations
Need to allow for abstractions and uncertainty

- **Need for abstraction**
  - Constants known with limited certainty
  - Unknown constants

- **Idea:** reason with parameters (unknown constants)
  - Verify the system in presence of uncertain constants
  - Synthesize suitable valuations for unknown parameters
  - Optimize parameter valuations

- **Challenging problems**
  - **Existence (dually: emptiness):** find one valuation for which a property holds
    - “Can I exhibit a valuation for which I am guaranteed to eventually get a coffee?”
  - **Synthesis:** find some/all parameter valuations for which a property holds
    - “Synthesize all valuations for which I am guaranteed to eventually get a coffee with 2 sugars”
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...

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

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

$\begin{align*}
  y &= 8 \\
  \text{coffee!} \\
  y &= 5 \\
  \text{cup!}
\end{align*}$

$\begin{align*}
  x &:= 0 \\
  y &:= 0 \\
  y &\leq 5 \\
  x &\geq 1 \\
  x &:= 0
\end{align*}$
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

\[ y = p_3 \]

coffee!

\[ y \leq p_2 \]

\[ x \geq p_1 \]

press?

\[ x := 0 \]

\[ y := 0 \]

\[ y = p_2 \]

cup!

\[ x := 0 \]

\[ y \leq 8 \]
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 $A$ and a parameter valuation $v$, we denote by $v(A)$ the (non-parametric) timed automaton where each parameter $p$ is valuated by $v(p)$.

\[
\begin{align*}
    v & : \begin{cases}
        p_1 & \rightarrow 1 \\
        p_2 & \rightarrow 5 \\
        p_3 & \rightarrow 8
    \end{cases}
\end{align*}
\]
Objectives

**Main objective**

Perform *efficient parameter synthesis* for parametric timed automata
Objectives

Main objective

Perform efficient parameter synthesis for parametric timed automata

Before designing algorithms, one shall first study theory

- Decidability
- Complexity
- Syntactic restrictions
Outline

1. Decidability
2. Efficient synthesis
3. Applications to schedulability analysis
4. Perspectives
Outline

1. Decidability
2. Efficient synthesis
3. Applications to schedulability analysis
4. Perspectives
25 years of (un)decidability results for PTAs

Key problem considered: EF-emptiness

- “given a PTA $A$ and a location $\bullet$, is the set of parameter valuations $\nu$ such that $\nu(A)$ reaches $\bullet$ empty”?
25 years of (un)decidability results for PTAs

Key problem considered: **EF-emptiness**

- “given a PTA \( \mathcal{A} \) and a location \( \bullet \), is the set of parameter valuations \( \nu \) such that \( \nu(\mathcal{A}) \) reaches \( \bullet \) empty”?

\[
\begin{array}{c}
\mathcal{A} \\
\bullet
\end{array}
\]
25 years of (un)decidability results for PTAs

Key problem considered: \textit{EF-emptiness}

- “given a PTA $A$ and a location $\bullet$, is the set of parameter valuations $\nu$ such that $\nu(A)$ reaches $\blacksquare$ empty”?

\begin{itemize}
  \item [\textcolor{red}{Undecidable}]
    \begin{itemize}
      \item [\textcolor{red}{Alur et al., /one.osf/nine.osf/nine.osf/three.osf}]
      \item [\textcolor{red}{Miller, /two.osf/zero.osf/zero.osf/zero.osf}]
      \item [\textcolor{red}{Doyen, /two.osf/zero.osf/zero.osf/seven.osf}]
      \item [\textcolor{red}{Beneš et al., /two.osf/zero.osf/one.osf/five.osf}]
    \end{itemize}
  \item [\textcolor{green}{Decidable}]
    \begin{itemize}
      \item [\textcolor{green}{Limiting the number of clocks}]
        \begin{itemize}
          \item [\textcolor{green}{Alur et al., /one.osf/nine.osf/nine.osf/three.osf, Bundala and Ouaknine, /two.osf/zero.osf/one.osf/four.osf, Beneš et al., /two.osf/zero.osf/one.osf/five.osf}]
        \end{itemize}
      \item [\textcolor{green}{Bounded integer-valued parameters}]
        \begin{itemize}
          \item [\textcolor{green}{Jovanović et al., /two.osf/zero.osf/one.osf/five.osf}]
        \end{itemize}
      \item [\textcolor{green}{Restricting the use of parameters}]
        \begin{itemize}
          \item [\textcolor{green}{Hune et al., /two.osf/zero.osf/zero.osf/two.osf}]
        \end{itemize}
    \end{itemize}
\end{itemize}
25 years of (un)decidability results for PTAs

Key problem considered: **EF-emptiness**

- "given a PTA $\mathcal{A}$ and a location $\bullet$, is the set of parameter valuations $\nu$ such that $\nu(\mathcal{A})$ reaches $\circ$ empty"?

Large collection of results

Survey: [André, STTT (2017)]
25 years of (un)decidability results for PTAs

Key problem considered: EF-emptiness

- “given a PTA $\mathcal{A}$ and a location $\bullet$, is the set of parameter valuations $\nu$ such that $\nu(\mathcal{A})$ reaches $\bullet$ empty”?

Large collection of results

Survey: [André, STTT (2017)]

- Undecidable
  - Undecidable for only 3 clocks  
    [Alur et al., 1993]
  - Undecidable for only 1 clock compared to parameters  
    [Miller, 2000]
  - Undecidable with only strict constraints ($<$, $>$)  
    [Doyen, 2007]
  - Undecidable for only one parameter  
    [Beneš et al., 2015]
25 years of (un)decidability results for PTAs

Key problem considered: EF-emptiness
- “given a PTA $A$ and a location $\bullet$, is the set of parameter valuations $\nu$ such that $\nu(A)$ reaches $\bullet$ empty”?

Large collection of results

Survey: [André, STTT (2017)]

Undecidable
- Undecidable for only 3 clocks [Alur et al., 1993]
- Undecidable for only 1 clock compared to parameters [Miller, 2000]
- Undecidable with only strict constraints ($<$, $>$) [Doyen, 2007]
- Undecidable for only one parameter [Beneš et al., 2015]

Decidable
- Limiting the number of clocks [Alur et al., 1993, Bundala and Ouaknine, 2014, Beneš et al., 2015]
- Bounded integer-valued parameters [Jovanović et al., 2015]
- Restricting the use of parameters [Hune et al., 2002]
Contributions: new (un)decidability results for PTAs

Investigating further problems for parametric timed automata

- Short version: all non-trivial problems are undecidable for PTAs
Contributions: new (un)decidability results for PTAs

Investigating further problems for parametric timed automata

- Short version: all non-trivial problems are \textbf{undecidable} for PTAs
- Long version: see manuscript (Chapter 3)
Contributions: new (un)decidability results for PTAs

Investigating further problems for parametric timed automata
- Short version: all non-trivial problems are **undecidable** for PTAs
- Long version: see manuscript (Chapter 3)

A new subclass: **integer-point PTAs** (IP-PTAs)
- “Each symbolic state (polyhedron) contains an integer point”

Good news: EF-emptiness is decidable (for bounded parameters)

Bad news: it is not possible to decide whether a PTA is IP

**Syntactic subclass: reset-PTA**
- “Whenever a clock is compared to a parameter, all clocks must be reset”
- \( y \leq 5 \)
- \( y \leq 8 \)
- \( \text{press?} \)
- \( x, y := 0 \)
- \( y = p_2 \cup! \)
- \( x, y := 0 \)
- \( x \geq p_1 \)
- \( \text{press?} \)
- \( x, y := 0 \)
- \( y = 8 \)
- \( \text{coffee!} \)
Contributions: new (un)decidability results for PTAs

Investigating further problems for parametric timed automata

- Short version: all non-trivial problems are **undecidable** for PTAs
- Long version: see manuscript (Chapter 3)

A new subclass: **integer-point PTAs** (IP-PTAs)

- “Each symbolic state (polyhedron) contains an integer point”

Good news: EF-emptiness is **decidable** (for bounded parameters)
Contributions: new (un)decidability results for PTAs

Investigating further problems for parametric timed automata
- Short version: all non-trivial problems are undecidable for PTAs
- Long version: see manuscript (Chapter 3)

A new subclass: integer-point PTAs (IP-PTAs)
- “Each symbolic state (polyhedron) contains an integer point”

   ![Diagram of integer-point PTAs]

- Good news: EF-emptiness is decidable (for bounded parameters)
- Bad news: it is not possible to decide whether a PTA is IP
Contributions: new (un)decidability results for PTAs

Investigating further problems for parametric timed automata

- Short version: all non-trivial problems are undecidable for PTAs
- Long version: see manuscript (Chapter 3)

A new subclass: integer-point PTAs (IP-PTAs)

- “Each symbolic state (polyhedron) contains an integer point”

- Good news: EF-emptiness is decidable (for bounded parameters)
- Bad news: it is not possible to decide whether a PTA is IP
- Good news: syntactic subclass: reset-PTA
  - “Whenever a clock is compared to a parameter, all clocks must be reset”
16 years of (un)decidability results for L/U-PTAs

- Subclass of PTAs partitioning parameters into upper-bound parameters \((x \leq p, x < p)\) and lower-bound parameters \((x \geq p, x > p)\)

```
press?
```

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

```
coffee!
```

\[
y = p_3
\]

```
press?
```

\[
x \geq p_1 \\
y := 0
\]

```
cup!
```

\[
y = p_2
\]

\[
y \leq 8
\]
16 years of (un)decidability results for L/U-PTAs

- Subclass of PTAs partitioning parameters into upper-bound parameters \( (x \leq p, x < p) \) and lower-bound parameters \( (x \geq p, x > p) \)

\[
\begin{align*}
\text{press?} & : x := 0 \\
y & := 0 \quad \text{coffee!} \\
y & \leq p_2 \quad \text{cup!}
\end{align*}
\]

EF-emptiness and -universality are decidable
[Hune et al., two.osf/zero.osf/zero.osf/two.osf]
Büchi-emptiness is decidable
[Bozzelli and La Torre, two.osf/zero.osf/zero.osf/nine.osf]
AF-emptiness is undecidable
[Jovanović et al., two.osf/zero.osf/one.osf/five.osf]

Seems to allow for interesting applications
16 years of (un)decidability results for L/U-PTAs

- Subclass of PTAs partitioning parameters into upper-bound parameters \((x \leq p, x < p)\) and lower-bound parameters \((x \geq p, x > p)\)

- EF-emptiness and -universality are decidable \([\text{Hune et al., 2002}]\)
16 years of (un)decidability results for L/U-PTAs

- Subclass of PTAs partitioning parameters into upper-bound parameters ($x \leq p, x < p$) and lower-bound parameters ($x \geq p, x > p$)

- EF-emptiness and -universality are decidable [Hune et al., 2002]
- Büchi-emptiness is decidable [Bozzelli and La Torre, 2009]
16 years of (un)decidability results for L/U-PTAs

- Subclass of PTAs partitioning parameters into upper-bound parameters \((x \leq p, x < p)\) and lower-bound parameters \((x \geq p, x > p)\)

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

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

- \(x := 0 \)  
  \(y := 0 \)

- \(y \leq p_2\)

- EF-emptiness and -universality are decidable  
  [Hune et al., 2002]

- Büchi-emptiness is decidable  
  [Bozzelli and La Torre, 2009]

- AF-emptiness is undecidable  
  [Jovanović et al., 2015]
16 years of (un)decidability results for L/U-PTAs

- Subclass of PTAs partitioning parameters into upper-bound parameters \((x \leq p, x < p)\) and lower-bound parameters \((x \geq p, x > p)\)

- EF-emptiness and -universality are decidable [Hune et al., 2002]
- Büchi-emptiness is decidable [Bozzelli and La Torre, 2009]
- AF-emptiness is undecidable [Jovanović et al., 2015]
- Seems to allow for interesting applications
Contributions: new (un)decidability results for L/U-PTAs

- Language-preservation problem: undecidable

---

Mathias Ramparison's PhD thesis

Étienne André
Contributions to parametric timed model checking
Contributions: new (un)decidability results for L/U-PTAs

- Language-preservation problem: \textbf{undecidable} [FORMATS’15]
Contributions: new (un)decidability results for L/U-PTAs

- Language-preservation problem: undecidable
  
- Deadlock-existence-emptiness: undecidable

Mathias Ramparison’s PhD thesis
Contributions: new (un)decidability results for L/U-PTAs

- **Language-preservation problem:** undecidable

  ![Language-preservation](attachment:image.png)

- **Deadlock-existence-emptiness:** undecidable

  ![Deadlock-existence](attachment:image.png)

- **Cycle-existence-emptiness:** decidable

  ![Cycle-existence](attachment:image.png)

**Mathias Ramparison’s PhD thesis**

Étienne André  Contributions to parametric timed model checking
Contributions: new (un)decidability results for L/U-PTAs

- Language-preservation problem: **undecidable**

- Deadlock-existence-emptiness: **undecidable**

- Cycle-existence-emptiness: **decidable**

- EG-emptiness: **decidable** only if the parameters are bounded with closed bounds
Contributions: new (un)decidability results for L/U-PTAs

- Language-preservation problem: **undecidable**

- Deadlock-existence-emptiness: **undecidable**

- Cycle-existence-emptiness: **decidable**

- EG-emptiness: **decidable** only if the parameters are bounded with closed bounds

- Full (T)CTL-emptiness: **undecidable** even for U-PTAs

---

**Mathias Ramparison’s PhD thesis**

Étienne André

Contributions to parametric timed model checking
### Summary of theoretical contributions

<table>
<thead>
<tr>
<th>Class</th>
<th>U-PTAs</th>
<th>bL/U-PTAs</th>
<th>L/U-PTAs</th>
<th>bPTAs</th>
<th>PTAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>emptiness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EF</td>
<td>[Hune et al., 2002]</td>
<td></td>
<td>[Hune et al., 2002]</td>
<td>[Miller, 2000]</td>
<td>[Alur et al., 1993]</td>
</tr>
<tr>
<td>AF</td>
<td>open</td>
<td></td>
<td>[Jovanović et al., 2015]</td>
<td>[ICFEM'16]</td>
<td>[Jovanović et al., 2015]</td>
</tr>
<tr>
<td>EG</td>
<td>open</td>
<td></td>
<td>[ACSD'17]</td>
<td></td>
<td>[ACSD'17]</td>
</tr>
<tr>
<td>AG</td>
<td>[ACSD'17]</td>
<td></td>
<td>[ACSD'17]</td>
<td>[ICSFEM'16]</td>
<td>[ICFEM'16]</td>
</tr>
<tr>
<td>TCTL</td>
<td>[FORMATS'18]</td>
<td></td>
<td>[Jovanović et al., 2015]</td>
<td>[Miller, 2000]</td>
<td>[Alur et al., 1993]</td>
</tr>
<tr>
<td>EC</td>
<td>[ACSD'17]</td>
<td></td>
<td>[ACSD'17]</td>
<td>[ACSD'17]</td>
<td>[ACSD'17]</td>
</tr>
<tr>
<td>ED</td>
<td>open</td>
<td></td>
<td>[ACSD'17]</td>
<td></td>
<td>[ICTAC'16]</td>
</tr>
<tr>
<td>Lg-Pres.</td>
<td>open</td>
<td></td>
<td></td>
<td></td>
<td>[FORMATS'15]</td>
</tr>
<tr>
<td>Trace-Pres.</td>
<td>open</td>
<td></td>
<td></td>
<td></td>
<td>[FORMATS'15]</td>
</tr>
</tbody>
</table>

**[FORMATS’15]** É. André and N. Markey  
**[ICTAC’16]** É. André  
**[ICFEM’16]** É. André, D. Lime and O. H. Roux  
**[ACSD’17]** É. André and D. Lime  
**[FORMATS’18]** É. André, D. Lime and M. Ramparison  
**WiP** Work in progress (decidable)

bL/U-PTA: bounded L/U-PTAs
Perspectives

Less expressive classes
- A quite unexplored formalism: U-PTA
  - Still able to model interesting systems

More expressive classes
- Extension to hybrid systems
  - Clocks become variables with arbitrary (and different) rates
Outline

1. Decidability

2. Efficient synthesis

3. Applications to schedulability analysis

4. Perspectives
Efficient synthesis: Motivation

- Parametric timed automata are very expressive
- But most problems are undecidable
- Still, they represent an excellent opportunity for \textit{pragmatic} parametric model checking

[André, Lime, Roux, FORMATS’16]
Efficient synthesis: Motivation

- Parametric timed automata are very expressive
- But most problems are undecidable
- Still, they represent an excellent opportunity for **pragmatic** parametric model checking

**Goal**

Design efficient parameter synthesis algorithms

[André, Lime, Roux, FORMATS’16]
Efficient synthesis: Motivation

- Parametric timed automata are very expressive
  
- But most problems are undecidable
  
- Still, they represent an excellent opportunity for **pragmatic** parametric model checking

**Goal**

Design efficient parameter synthesis algorithms

**Two possible directions:**

1. Achieving termination without guarantee on the completeness
2. Achieving exact synthesis without guarantee on termination
Outline

1. Decidability

2. Efficient synthesis
   - Parametric reachability preservation
   - Compositional parameter synthesis
   - Implementation in IMITATOR

3. Applications to schedulability analysis

4. Perspectives
Parametric reachability preservation

**Input:** a PTA $\mathcal{A}$, a goal location $\blacksquare$, a parameter valuation $v$

**Problem:** synthesize valuations $v'$ such that $v(\mathcal{A})$ reaches $\blacksquare$ iff $v'(\mathcal{A})$ reaches $\blacksquare$
Parametric reachability preservation

**Parametric reachability preservation problem**

**Input:** a PTA $\mathcal{A}$, a goal location $\bullet$, a parameter valuation $v$

**Problem:** synthesize valuations $v'$ such that $v(\mathcal{A})$ reaches $\bullet$ iff $v'(\mathcal{A})$ reaches $\bullet$.
Parametric reachability preservation problem

**Input:** a PTA \( \mathcal{A} \), a goal location \( \bullet \), a parameter valuation \( v \)

**Problem:** synthesize valuations \( v' \) such that \( v(\mathcal{A}) \) reaches \( \bullet \) iff \( v'(\mathcal{A}) \) reaches \( \bullet \)

Reachability-preservation-emptiness problem **undecidable**

[André, Lipari, Nguyen, Sun, NFM'15]
Parametric reachability preservation: An algorithm

A pragmatic procedure: \( PRP(\mathcal{A}, v) \)
Parametric reachability preservation: An algorithm

**A pragmatic procedure:** $\text{PRP}(\mathcal{A}, \nu)$

Built on top of two existing algorithms:
- Reachability synthesis ($\text{EFsynth}$) [Alur et al., Jovanović et al.]
- Trace-preservation-synthesis ($\text{IM}$) [André et al.]

Key heuristics: only explore behaviors “similar” to that of $\nu(\mathcal{A})$;
- Non-necessarily terminating, incomplete on purpose, but fast in practice.
Parametric reachability preservation: An algorithm

A pragmatic procedure: $\text{PRP}(\mathcal{A}, v)$

- Built on top of two existing algorithms
  - Reachability synthesis ($\text{EFsynth}$) 
  - Trace-preservation-synthesis ($\text{IM}$) 

- Key heuristics: only explore behaviors “similar” to that of $v(\mathcal{A})$
  - Non-necessarily terminating, incomplete on purpose, but fast in practice
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC

![Diagram of a bounded parameter domain](image)

Étienne André

Contributions to parametric timed model checking
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC

![Diagram of a 2D parameter space with a point $v_1$.]
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC
Solving reachability synthesis using **PRP**

- Idea: select valuations in a bounded parameter domain, and call **PRP** on these valuations, until a sufficient coverage is reached: algorithm **PRPC**
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm \textit{PRPC}

![Diagram](image_url)

\begin{itemize}
  \item [p_1] \begin{itemize}
    \item [p_2]
    \end{itemize}
  \end{itemize}

**Principle:** "many small analyses rather than one big analysis"; memory gain

**Unexpected:** time gain in several cases too!

Case study

<table>
<thead>
<tr>
<th>Clocks</th>
<th>Points</th>
<th>\textit{EFsynth}</th>
<th>\textit{BC}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>\textit{PRPC}</td>
<td>\textit{toy}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/two.osf</td>
<td>/six.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/zero.osf</td>
<td>/one.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/eight.osf</td>
<td>/one.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/five.osf</td>
<td>/nine.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/three.osf</td>
<td>/two.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/five.osf</td>
<td>/seven.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/four.osf</td>
<td>/five.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/three.osf</td>
<td>/seven.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/five.osf</td>
<td>/one.osf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/four.osf</td>
<td>/five.osf</td>
</tr>
</tbody>
</table>

Étienne André

Contributions to parametric timed model checking
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC.
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC
Solving reachability synthesis using PRP

- Idea: select valuations in a bounded parameter domain, and call PRP on these valuations, until a sufficient coverage is reached: algorithm PRPC

- Principle: “many small analyses rather than one big analysis” \( \sim \) memory gain
Solving reachability synthesis using **PRP**

- **Idea:** select valuations in a bounded parameter domain, and call **PRP** on these valuations, until a sufficient coverage is reached: algorithm **PRPC**

- **Principle:** “many small analyses rather than one big analysis” \( \sim \) memory gain
- **Unexpected:** time gain in several cases too!

### Case study

<table>
<thead>
<tr>
<th>Case study</th>
<th>Clocks</th>
<th>Points</th>
<th>EFsynth</th>
<th>BC</th>
<th>PRPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>toy</td>
<td>2</td>
<td>2 601</td>
<td>0.401</td>
<td>( \infty )</td>
<td>0.078</td>
</tr>
<tr>
<td>Sched1</td>
<td>13</td>
<td>6 561</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>1 595</td>
</tr>
<tr>
<td>Sched2.50.0</td>
<td>6</td>
<td>3 321</td>
<td>9.25</td>
<td>990</td>
<td>14.55</td>
</tr>
<tr>
<td>Sched2.50.2</td>
<td>6</td>
<td>3 321</td>
<td>662</td>
<td>( \infty )</td>
<td>213</td>
</tr>
<tr>
<td>Sched2.100.0</td>
<td>6</td>
<td>972 971</td>
<td>21.4</td>
<td>2 093</td>
<td>116</td>
</tr>
<tr>
<td>Sched2.100.2</td>
<td>6</td>
<td>972 971</td>
<td>3 757</td>
<td>( \infty )</td>
<td>4 557</td>
</tr>
<tr>
<td>Sched5</td>
<td>21</td>
<td>1 681</td>
<td>352</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>SPSMALL</td>
<td>11</td>
<td>3 082</td>
<td>7.49</td>
<td>587</td>
<td>118</td>
</tr>
</tbody>
</table>

**BC:** former algorithm [André and Fribourg, 2010]

---

Étienne André

Contributions to parametric timed model checking
Towards distributed parameter synthesis

- Point-based algorithms (that iterate on parameter valuations) can obviously be distributed
  - Distribution over a cluster: many computers with their own memory (communication through a network)
Towards distributed parameter synthesis

- Point-based algorithms (that iterate on parameter valuations) can obviously be distributed
  - Distribution over a cluster: many computers with their own memory (communication through a network)
- Challenge: How to efficiently distribute these algorithms?
  - “Shape” of the result unknown a priori
  - Risk of redundant computations
Towards distributed parameter synthesis

- Point-based algorithms (that iterate on parameter valuations) can obviously be distributed
  - Distribution over a cluster: many computers with their own memory (communication through a network)
- Challenge: How to efficiently distribute these algorithms?
  - “Shape” of the result unknown a priori
  - Risk of redundant computations
- We proposed several distribution policies
  - Most efficient: dynamic domain decomposition (25 times faster on 128 nodes)

[André, Coti, Evangelista, EuroMPI’14] [André, Coti, Nguyễn, ICFEM’15]

Nguyễn Hoàng Gia’s PhD thesis
Towards distributed parameter synthesis

■ Point-based algorithms (that iterate on parameter valuations) can obviously be distributed
  ■ Distribution over a cluster: many computers with their own memory (communication through a network)
■ Challenge: How to efficiently distribute these algorithms?
  ■ “Shape” of the result unknown a priori
  ■ Risk of redundant computations
■ We proposed several distribution policies
  ■ Most efficient: dynamic domain decomposition (25 times faster on 128 nodes)

[André, Coti, Evangelista, EuroMPI’14] [André, Coti, Nguyễn, ICFEM’15]

Nguyễn Hoàng Gia’s PhD thesis

■ Application to PRPC

<table>
<thead>
<tr>
<th>Case study</th>
<th>Clocks</th>
<th>Points</th>
<th>EFsynth</th>
<th>BC</th>
<th>PRPC</th>
<th>PRPC distr(12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>toy</td>
<td>2</td>
<td>2 601</td>
<td>0.401</td>
<td>∞</td>
<td>0.078</td>
<td>0.050</td>
</tr>
<tr>
<td>Sched1</td>
<td>13</td>
<td>6 561</td>
<td>∞</td>
<td>∞</td>
<td>1 595</td>
<td>219</td>
</tr>
<tr>
<td>Sched2.50.0</td>
<td>6</td>
<td>3 321</td>
<td>9.25</td>
<td>990</td>
<td>14.55</td>
<td>4.77</td>
</tr>
<tr>
<td>Sched2.50.2</td>
<td>6</td>
<td>3 321</td>
<td>662</td>
<td>∞</td>
<td>213</td>
<td>84</td>
</tr>
<tr>
<td>Sched2.100.0</td>
<td>6</td>
<td>972 971</td>
<td>21.4</td>
<td>2 093</td>
<td>116</td>
<td>10.1</td>
</tr>
<tr>
<td>Sched2.100.2</td>
<td>6</td>
<td>972 971</td>
<td>3 757</td>
<td>∞</td>
<td>4 557</td>
<td>1543</td>
</tr>
<tr>
<td>Sched5</td>
<td>21</td>
<td>1 681</td>
<td>352</td>
<td>∞</td>
<td>∞</td>
<td>917</td>
</tr>
<tr>
<td>SPSMALL</td>
<td>11</td>
<td>3 082</td>
<td>7.49</td>
<td>587</td>
<td>118</td>
<td>11.2</td>
</tr>
</tbody>
</table>
Outline

1 Decidability

2 Efficient synthesis
   - Parametric reachability preservation
   - Compositional parameter synthesis
   - Implementation in IMITATOR

3 Applications to schedulability analysis

4 Perspectives
Compositional verification of timed systems

Learning an unknown timed system via interactions with a teacher

- Membership queries
- Candidate queries

Extension $TL^*$ of the $L^*$ algorithm [Angluin, 1987]

- Using a subclass of timed automata [Alur et al., 1999]
- More efficient than [Grinchtein et al., 2010]

[Lin, André et al., ATVA’11] [Lin, André et al., FM’12]
Compositional verification of timed systems

Learning an unknown timed system via interactions with a teacher

- Membership queries
- Candidate queries

Extension $\text{TL}^*$ of the $\text{L}^*$ algorithm [Angluin, 1987]

- Using a subclass of timed automata [Alur et al., 1999]
- More efficient than [Grinchtein et al., 2010]

Use $\text{TL}^*$ to learn an abstraction of a component (assume-guarantee reasoning)
Compositional verification of timed systems

Learning an unknown timed system via interactions with a teacher

- Membership queries
- Candidate queries

Extension $\text{TL}^*$ of the $\mathbb{L}^*$ algorithm [Angluin, 1987]

- Using a subclass of timed automata [Alur et al., 1999]
- More efficient than [Grinchtein et al., 2010]

[Lin, André et al., ATVA’11] [Lin, André et al., FM’12]

Use $\text{TL}^*$ to learn an abstraction of a component (assume-guarantee reasoning)

\[
\parallel \quad \models \varphi
\]

with

\[
\models
\]
Compositional verification of timed systems

Learning an unknown timed system via interactions with a teacher

- Membership queries
- Candidate queries

Extension $\text{TL}^*$ of the $L^*$ algorithm [Angluin, 1987]

- Using a subclass of timed automata [Alur et al., 1999]
- More efficient than [Grinchenko et al., 2010]

Use $\text{TL}^*$ to learn an abstraction of a component (assume-guarantee reasoning)

\[
\begin{align*}
\phi & \quad \models \\
\text{with} & \quad \models
\end{align*}
\]

- Membership and candidate queries are performed by model checking
- Much faster than monolithic verification [Lin, André et al., TSE 2014]
Compositional parameter synthesis

Given a parametric component $A$ and a non-parametric component $B$

\[ v(A) \parallel \tilde{B} \models \varphi \]

Else generalize the counter-example (cheap)

Find another point and restart
Compositional parameter synthesis

Given a parametric component $A$ and a non-parametric component $B$

1. Pick a parameter valuation $v$

\[
\begin{align*}
\mathcal{U} \langle (\text{\small A}) \parallel \text{\small B} \rangle \models \varphi
\end{align*}
\]
Given a parametric component $A$ and a non-parametric component $B$

1. Pick a parameter valuation $\nu$
2. Compute an abstraction $\tilde{B}$ of $B$

![Diagram showing the process of composition and abstraction](image-url)
Compositional parameter synthesis

[André and Lin, FORTE’17]

Given a parametric component $A$ and a non-parametric component $B$

1. Pick a parameter valuation $v$
2. Compute an abstraction $\tilde{B}$ of $B$
3. If $v(A) \parallel \tilde{B} \models \varphi$, synthesize $\text{PRP}(A \parallel \tilde{B}, v)$

---

Étienne André
Contributions to parametric timed model checking
Compositional parameter synthesis

[André and Lin, FORTE’17]

Given a parametric component $A$ and a non-parametric component $B$:

1. Pick a parameter valuation $v$
2. Compute an abstraction $\tilde{B}$ of $B$
3. If $v(A) \parallel \tilde{B} \models \varphi$, synthesize $\text{PRP}(A \parallel \tilde{B}, v)$

4. Find another point and restart

---

Etienne André
Contributions to parametric timed model checking
Given a parametric component $A$ and a non-parametric component $B$

1. Pick a parameter valuation $v$
2. Compute an abstraction $\tilde{B}$ of $B$
3. If $v(A) \parallel \tilde{B} \models \varphi$, synthesize $\text{PRP}(A \parallel \tilde{B}, v)$
   Else generalize the counter-example (cheap)
4. Find another point and restart
Compositional parameter synthesis

[André and Lin, FORTE’17]

Given a parametric component $A$ and a non-parametric component $B$

1. Pick a parameter valuation $v$
2. Compute an abstraction $\tilde{B}$ of $B$
3. If $v(A) \parallel \tilde{B} \models \varphi$, synthesize PRP$(A \parallel \tilde{B}, v)$
   Else generalize the counter-example (cheap)
4. Find another point and restart
Compositional parameter synthesis: experiments

Toolkit made of IMITATOR and CV, interfaced by a Python script

<table>
<thead>
<tr>
<th>Case study</th>
<th>#A</th>
<th>#X</th>
<th>#P</th>
<th>Spec</th>
<th>EFsynth</th>
<th>#abs</th>
<th>#c.-ex.</th>
<th>learning</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMS-1</td>
<td>6</td>
<td>18</td>
<td>2</td>
<td>1</td>
<td>0.299</td>
<td>1</td>
<td>1</td>
<td>0.074</td>
<td>0.136</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>0</td>
<td>0.010</td>
<td>0</td>
<td>1</td>
<td>0.038</td>
<td>0.046</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>0.282</td>
<td>1</td>
<td>0</td>
<td>0.090</td>
<td>0.242</td>
</tr>
<tr>
<td>FMS-2</td>
<td>11</td>
<td>37</td>
<td>2</td>
<td>1</td>
<td>∞</td>
<td>1</td>
<td>1</td>
<td>84.2</td>
<td>88.9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>81.4</td>
<td>85.2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>0.051</td>
<td></td>
<td>0</td>
<td>2</td>
<td>1.10</td>
<td>2.44</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>0.062</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1.42</td>
<td>1.53</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>31.4</td>
<td>40.8</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>37.2</td>
<td>42.4</td>
</tr>
<tr>
<td>AIP</td>
<td>11</td>
<td>46</td>
<td>2</td>
<td>1</td>
<td>0.551</td>
<td>0</td>
<td>1</td>
<td>0.086</td>
<td>0.114</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>2.11</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1.22</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>3.91</td>
<td></td>
<td>0</td>
<td>1</td>
<td>8.50</td>
<td>8.54</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>0.235</td>
<td></td>
<td>1</td>
<td>1</td>
<td>8.39</td>
<td>8.42</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0.394</td>
<td>0.871</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>5.32</td>
<td>9.58</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1.76</td>
<td>3.19</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1.13</td>
<td>4.35</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td>∞</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0.762</td>
<td>1.84</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td>0.022</td>
<td></td>
<td>0</td>
<td>1</td>
<td>0.072</td>
<td>0.094</td>
</tr>
<tr>
<td>Fischer-3</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td></td>
<td>2.76</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fischer-4</td>
<td>6</td>
<td>16</td>
<td>2</td>
<td></td>
<td>∞</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Works well when loosely synchronized and loosely timed
Outline

1 Decidability

2 Efficient synthesis
   - Parametric reachability preservation
   - Compositional parameter synthesis
   - Implementation in IMITATOR

3 Applications to schedulability analysis

4 Perspectives
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
  - 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]

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

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

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

- Beyond distributed verification
  - Multicore verification
  - Swarm verification

- Combine non-parametric and parametric analyses
  - Machine learning
    - “Learn” a constraint by repeated call to a non-parametric model checker (much faster)
    - Preliminary works in [Li, Sun, Gao, André, ICFEM’17]
Outline

1. Decidability
2. Efficient synthesis
3. Applications to schedulability analysis
4. Perspectives

Étienne André
Contributions to parametric timed model checking
Context: schedulability analysis

Real-time system:

- Set of tasks (with a period, a WCET and a deadline)
- One processor (uniprocessor) or more (multiprocessor)
- Scheduling policies: fixed priority (FPS), earliest deadline first (EDF)...

Definition (Schedulability analysis)

Given a real-time system and a scheduling policy, certify that no deadline miss will ever occur
Context: schedulability analysis

Real-time system:
- Set of tasks (with a period, a WCET and a deadline)
- One processor (uniprocessor) or more (multiprocessor)
- Scheduling policies: fixed priority (FPS), earliest deadline first (EDF)...

Definition (Schedulability analysis)

Given a real-time system and a scheduling policy, certify that no deadline miss will ever occur

Solved in [Liu and Layland, 1973]*
Context: schedulability analysis

Real-time system:

- Set of tasks (with a period, a WCET and a deadline)
- One processor (uniprocessor) or more (multiprocessor)
- Scheduling policies: fixed priority (FPS), earliest deadline first (EDF)...

Definition (Schedulability analysis)

Given a real-time system and a scheduling policy, certify that no deadline miss will ever occur

Solved in [Liu and Layland, 1973]*

*for fixed priority, for a single processor, without jitter, without sporadic tasks, without preemption, without precedence constraints, without resource sharing, without uncertainty...
Context: schedulability analysis

Real-time system:

- Set of tasks (with a period, a WCET and a deadline)
- One processor (uniprocessor) or more (multiprocessor)
- Scheduling policies: fixed priority (FPS), earliest deadline first (EDF)…

Definition (Schedulability analysis)

Given a real-time system and a scheduling policy, certify that no deadline miss will ever occur

Solved in [Liu and Layland, 1973]*

*for fixed priority, for a single processor, without jitter, without sporadic tasks, without preemption, without precedence constraints, without resource sharing, without uncertainty…

In general, schedulability analysis is hard
Outline

1. Decidability

2. Efficient synthesis

3. Applications to schedulability analysis
   - Parametric stopwatch automata
   - The Thales challenge

4. Perspectives
Schedulability analysis with parametric model checking

<table>
<thead>
<tr>
<th>Goal: parametric schedulability analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given a real-time system and a scheduling policy, <strong>synthesize</strong> valuations (deadlines, periods...) such that the system is schedulable.</td>
</tr>
</tbody>
</table>

Modeling a real-time system with PTAs
- Each task or chain of task: one PTA
- Each scheduler: one PTA
- Use **stopwatches** to model preemption
Schedulability analysis with parametric model checking

Goal: parametric schedulability analysis

Given a real-time system and a scheduling policy, synthesize valuations (deadlines, periods...) such that the system is schedulable.

Modeling a real-time system with PTAs

- Each task or chain of task: one PTA
- Each scheduler: one PTA
- Use stopwatches to model preemption

Comparison with analytical methods

- Much better in terms of completeness
- And can evaluate robustness

[Roman Soulat’s PhD thesis]

Étienne André
Contributions to parametric timed model checking
Outline

1. Decidability

2. Efficient synthesis

3. Applications to schedulability analysis
   - Parametric stopwatch automata
   - The Thales challenge

4. Perspectives
The Thales challenge (1/2)

“FMTV challenge” by Thales proposed during the WATERS 2014 workshop
Solutions presented at WATERS 2015

System: an unmanned aerial video system

- Architecture: 4 processors, 4 tasks, 2 buffers
- …with uncertain periods
  - Period constant but with a small uncertainty (typically 0.01 %)
  - Not a jitter!

<table>
<thead>
<tr>
<th>Processor</th>
<th>Task</th>
<th>Period</th>
<th>WCET</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>T1 (Pre-processing)</td>
<td>40ms +/- 0.01%</td>
<td>28ms</td>
</tr>
<tr>
<td>GPP1</td>
<td>T2 (Processing)</td>
<td>40/3ms +/- 0.05%</td>
<td>17ms</td>
</tr>
<tr>
<td>GPU</td>
<td>T3 (Filtering)</td>
<td>40ms +/- 0.01%</td>
<td>19ms</td>
</tr>
<tr>
<td>GPP2</td>
<td>T4 (D/A converting)</td>
<td>40ms +/- 0.01%</td>
<td>1ms or 10ms</td>
</tr>
</tbody>
</table>

BCET = WCET = 8ms

CET = 1ms or 10ms

to display
The Thales challenge (2/2)

**Goal**

Compute the end-to-end BCET and WCET times for a buffer size of 1 and 3
The Thales challenge (2/2)

Goal

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

Challenging!

- Distributed system (multiprocessor)
- Buffers
- Dependencies between tasks
- Uncertain periods
The Thales challenge (2/2)

Goal

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

Challenging!

- Distributed system (multiprocessor)
- Buffers
- Dependencies between tasks
- Uncertain periods

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., $P1 \in [40 - 0.004, 40 + 0.004]$)
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

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)
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.
Results obtained by IMITATOR

E2E latency results for the two buffer sizes

<table>
<thead>
<tr>
<th>Buffer size</th>
<th>1</th>
<th>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

[André, Lipari, Sun, WATERS’15]
Perspectives

■ Better scalability
  ■ Design dedicated synthesis algorithms
  ■ Compositional synthesis

■ Better integration
  ■ Parametric task automata
  ■ Support of existing industrial formalisms in IMITATOR

[André, FMICS’17]
Outline

1. Decidability
2. Efficient synthesis
3. Applications to schedulability analysis
4. Perspectives
Summary of contributions

- Theory
  - New decidable subclasses of parametric timed automata

- Efficient synthesis algorithms
  - Implementation in IMITATOR

- Application to real-time systems
  - Application to industrial case studies

Also (not presented)

- Robustness of timed concurrent systems
- Formal specification of (timed) concurrent systems
  - Mahdi Benmoussa’s PhD thesis
General perspectives

- Timing parameters in more complex settings
  - Probabilities:
    - preliminary works in [André and Delahaye, TIME’16]
  - Hybrid systems

- More parameters
  - Probabilistic parameters
  - Discrete parameters (networks of identical processes)

- Beyond parameter synthesis: controller synthesis
  - More abstraction
  - More uncertainty

- More applications
  - Real-time systems
  - Biological systems
  - Cybersecurity
Bibliography
References I

A theory of timed automata.

Event-clock automata: A determinizable class of timed automata.

Parametric real-time reasoning.

André, É. (2016).
Parametric deadlock-freeness checking timed automata.

André, É. (2017a).
A unified formalism for monoprocessor schedulability analysis under uncertainty.
Best paper award.
References II


Etienne André  Contributions to parametric timed model checking
References V

Learning regular sets from queries and counterexamples.

*Principles of Model Checking.*
MIT Press.

Language emptiness of continuous-time parametric timed automata.
In *ICALP, Part II*, volume 9135 of LNCS, pages 69–81. Springer.

Decision problems for lower/upper bound parametric timed automata.

Advances in parametric real-time reasoning.

Robust parametric reachability for timed automata.


References VII

UPPAAL in a nutshell.

Li, J., Sun, J., Gao, B., and André, É. (2017).
Classification based parameter synthesis for parametric timed automata.
To appear.

An efficient algorithm for learning event-recording automata.

Learning assumptions for compositional verification of timed systems.

Automatic compositional verification of timed systems.

Scheduling algorithms for multiprogramming in a hard-real-time environment.


Summary of publications
## Summary of publications

<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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Books</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td><strong>Proceedings</strong></td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td><strong>International journals</strong></td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>11</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>57</td>
</tr>
<tr>
<td><strong>International conferences</strong></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td><strong>Other publications</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>15</td>
<td>14</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>3</td>
<td>80</td>
</tr>
</tbody>
</table>
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)

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)
Varying the coffee machine
The most critical system: the coffee machine

- $y = 8$
  - coffee!
- $y \leq 5$
- $y = 5$
  - cup!
- $y \leq 8$

Example of concrete run for the coffee machine

1. Coffee with no sugar
   - $x = 0$
   - $y = 0$

2. Coffee with two doses of sugar
   - $x = 0$
   - $y = 0$

Colors:
- Green: idle
- Blue: adding sugar
- Red: delivering coffee
The most critical system: the coffee machine

$y = 8$
coffee!

$y \leq 5$

press?
$x := 0$
y := 0

$y = 5$
cup!

$x \geq 1$
press?
$x := 0$

$y \leq 8$

Example of concrete run for the coffee machine

Coffee with no sugar

$x = 0$
y = 0
The most critical system: the coffee machine

Example of concrete run for the coffee machine

- Coffee with no sugar
The most critical system: the coffee machine

Example of concrete run for the coffee machine

- Coffee with no sugar

\[
\begin{array}{c}
x = 0 \\
y = 0 \\
y = 8
\end{array}
\]
The most critical system: the coffee machine

Example of concrete run for the coffee machine

Coffee with no sugar
The most critical system: the coffee machine

Example of concrete run for the coffee machine

- Coffee with no sugar
The most critical system: the coffee machine

Example of concrete run for the coffee machine

Coffee with no sugar

| x  | y  |
|----|----|-------------------------------|------------------|
| 0  | 0  | 5                             | coffee!          |
| 0  | 5  | 5                             | cup!             |
| 5  | 5  | 8                             | y = 5            |
| 5  | 8  | y = 8                         | press?           |
| 0  | 0  | x := 0                        | y ≤ 5            |
| 1  | 0  | x := 0                        | y ≤ 8            |

Étienne André
Contributions to parametric timed model checking
The most critical system: the coffee machine

Example of concrete run for the coffee machine

Coffee with no sugar

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

Example of concrete run for the coffee machine

- Coffee with no sugar
  - \( x = 0 \)
  - \( y = 0 \)

- Coffee with 2 doses of sugar
  - \( x = 0 \)
  - \( y = 0 \)
The most critical system: the coffee machine

\[ y = \begin{cases} 
8 & \text{coffee!} \\
\leq 5 & \text{press?} \\
x := 0 \\
y := 0 \\
\geq 1 & \text{cup!} \\
x := 0 \\
y \leq 8 & \text{idle}, \text{adding sugar}, \text{delivering coffee}
\end{cases} \]

- Example of concrete run for the coffee machine

- **Coffee with no sugar**
  \[
  \begin{array}{ccccc}
  x & y & \text{press?} & \text{cup!} & \text{coffee!} \\
  0 & 0 & 0 & 5 & 5 & 8 & 8 \\
  \end{array}
  \]

- **Coffee with 2 doses of sugar**
  \[
  \begin{array}{ccc}
  x & y & \text{press?} \\
  0 & 0 & 1.5 \\
  \end{array}
  \]

Étienne André
Contributions to parametric timed model checking
The most critical system: the coffee machine

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

\[
y \leq 5
\]

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

\[
x \geq 1 \\
x := 0 \\
y = 5 \\
\text{cup!}
\]

Example of concrete run for the coffee machine

- Coffee with no sugar

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

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

- Coffee with 2 doses of sugar

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

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

Example of concrete run for the coffee machine

■ Coffee with no sugar

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

\[y = 8\]
coffee!

\[y \leq 5\]

\[y = 5\]
cup!

\[x \geq 1\]
press?

\[x = 0\]
press?

\[y = 0\]
press?

\[x = 0\]
press?

\[x = 0\]
press?

\[x = 0\]
press?

\[x = 0\]
press?

\[x = 0\]
press?

Example of concrete run for the coffee machine

- **Coffee with no sugar**

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

- **Coffee with 2 doses of sugar**

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

Étienne André
Contributions to parametric timed model checking
The most critical system: the coffee machine

$y = 8$
coffee!

$y \leq 5$
press?

$x := 0$
$y := 0$

$y = 5$
cup!

$x \geq 1$
press?

$x := 0$

Example of concrete run for the coffee machine

Coffee with no sugar

$x = 0$
$y = 0$

Combine diagrams with

Coffee with two doses of sugar

$x = 0$
$y = 0$
The most critical system: the coffee machine

Example of concrete run for the coffee machine

- **Coffee with no sugar**
  - \( x := 0 \)
  - \( y := 0 \)
  - \( x \geq 1 \)

- **Coffee with 2 doses of sugar**
  - \( x := 0 \)
  - \( y := 0 \)
The most critical system: the coffee machine

\begin{align*}
  x &:= 0 \\
  y &:= 0 \\
  y &\leq 5 \\
  y &= 5 \quad \text{cup!} \\
  x &\geq 1 \\
  x &= 0
\end{align*}

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

- Example of concrete run for the coffee machine

- Coffee with no sugar

\[ x = 0 \quad 0 \quad 5 \quad 5 \quad 8 \quad 8 \]
\[ y = 0 \quad 0 \quad 5 \quad 5 \quad 8 \quad 8 \]

- Coffee with 2 doses of sugar

\[ x = 0 \quad 0 \quad 1.5 \quad 0 \quad 2.7 \quad 0 \quad 0.8 \quad 0.8 \quad 3.8 \]
\[ y = 0 \quad 0 \quad 1.5 \quad 1.5 \quad 4.2 \quad 4.2 \quad 5 \quad 5 \quad 8 \]
The most critical system: the coffee machine

Example of concrete run for the coffee machine

Coffee with no sugar

Coffee with 2 doses of sugar

Étienne André  Contributions to parametric timed model checking
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

\[ y = 8 \]

coffee!

\[ y \leq 5 \]

press?

\[ x := 0 \]

\[ y := 0 \]

\[ x \geq 1 \]

\[ y = 5 \]

cup!

\[ y \leq 8 \]

Example of concrete run for the coffee machine

Coffee with \( /two.osf \) doses of sugar

\[ x = 0 \]

\[ y = 0 \]

\[ x = 0 \]

idle

adding sugar

delivering coffee
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

\[
\begin{align*}
\text{press?} & \quad x := 0 \\
y & := 0 \\
\end{align*}
\]

\[
\begin{align*}
y & \leq 4 \\
x & \geq 1 \\
y & := 4 \\
\end{align*}
\]

\[
\begin{align*}
x & := 0 \\
y & \leq 6 \\
\end{align*}
\]

idle
adding sugar
delivering coffee

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

Étienne André
Contributions to parametric timed model checking
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

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

\[ x \geq 1 \] press?
\[ y = 4 \] cup!
\[ y \leq 6 \]

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 (2/2)

A second (faster) coffee machine

\[ y = 8 \]
coffee!

\[ y \leq 4 \]

\[ y = 4 \]
cup!

\[ y \leq 6 \]

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 (2/2)

A second (faster) coffee machine

\[ y = 8 \]

coffee!

\[ y \leq 4 \]

\[ y = 4 \]
cup!

\[ y \leq 6 \]

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[
\begin{aligned}
x &= 0 & 0 & 1.5 \\
y &= 0 & 0 & 1.5 \\
\end{aligned}
\]
A second (faster) coffee machine

\[ y = 8 \]

\[ \text{coffee!} \]

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[ \begin{align*}
x &= 0 & 0 & 1.5 & 0 \\
y &= 0 & 0 & 1.5 & 1.5
\end{align*} \]
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

\[ y = 8 \]

coffee!

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

press?

\[ y \leq 4 \]

\[ x \geq 1 \]

\[ x := 0 \]

press?

\[ y = 4 \]

cup!

\[ y \leq 6 \]

idle

adding sugar

delivering coffee

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[
\begin{align*}
x &= 0 & 0 & 1.5 & 0 & 1.7 \\
y &= 0 & 0 & 1.5 & 1.5 & 3.2
\end{align*}
\]
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

\[
\begin{align*}
    & x := 0 \\
    & y := 0 \\
    & y \leq 4 \\
    & y = 4 \\
    \text{coffee!}
\end{align*}
\]

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

\[
\begin{array}{cccccc}
    x & 0 & 0 & 1.5 & 0 & 1.7 & 0 \\
    y & 0 & 0 & 1.5 & 1.5 & 3.2 & 3.2
\end{array}
\]
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

\[ y = 8 \]

**coffee!**

\[ y \leq 4 \]

\[ x : = 0 \]

\[ y : = 0 \]

\[ y = 4 \]

**cup!**

\[ x \geq 1 \]

**press?**

\[ x : = 0 \]

\[ y \leq 6 \]

---

Example of concrete run for the coffee machine

- Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>Press</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = )</td>
<td>0</td>
</tr>
<tr>
<td>( y = )</td>
<td>0</td>
</tr>
</tbody>
</table>
The most critical system: the coffee machine \((2/2)\)

A second (faster) coffee machine

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

\[
x := 0 \\
y := 0 \\
y \leq 4
\]

\[
x \geq 1 \\
y = 4 \\
\text{cup!}
\]

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

Example of concrete run for the coffee machine

- **Coffee with 2 doses of sugar**

<table>
<thead>
<tr>
<th>Press</th>
<th>Time</th>
<th>Press</th>
<th>Time</th>
<th>Press</th>
<th>Time</th>
<th>Press</th>
<th>Time</th>
<th>Cup</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>
<td>(y)</td>
<td>(x)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>1.7</td>
<td>0</td>
<td>0.8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
<td>3.2</td>
<td>3.2</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

idle  
adding sugar  
delivering coffee
The most critical system: the coffee machine (2/2)

A second (faster) coffee machine

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

\[ y \leq 4 \]
\[ \text{press?} \]
\[ x := 0 \]
\[ y := 0 \]

\[ y = 4 \]
\[ \text{cup!} \]
\[ x \geq 1 \]
\[ x := 0 \]

\[ y \leq 6 \]

Example of concrete run for the coffee machine

Coffee with 2 doses of sugar

\[ x = \]
\[ y = \]

\[ \begin{array}{cccccccccc}
0 & 0 & 1.5 & 0 & 1.7 & 0 & 0.8 & 0.8 & 0.8 & 2.8 \\
0 & 0 & 1.5 & 1.5 & 3.2 & 3.2 & 4 & 4 & 6 \\
\end{array} \]
Decidability of PTAs
# Surveying EF-emptiness for PTAs

<table>
<thead>
<tr>
<th>T</th>
<th>P</th>
<th>Guards</th>
<th>Invariants</th>
<th>P-clocks</th>
<th>NP-clocks</th>
<th>Params</th>
<th>Decidability</th>
<th>Main ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>( x \gg p \cdot d )</td>
<td>1</td>
<td>0</td>
<td>fixed</td>
<td>(at most) PTIME</td>
<td>[Miller, 2000] (consequence)</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( x \gg p \cdot d )</td>
<td>1</td>
<td>0</td>
<td>any</td>
<td>(at most) NP-complete</td>
<td>[Miller, 2000] (consequence)</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( \leq p \cdot d^+ )</td>
<td>1</td>
<td>any</td>
<td>any</td>
<td>NEXPTIME-complete</td>
<td>[Bundala and Ouaknine, 2014]</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( x \gg p \cdot d ) ( \leq p \cdot d^+ )</td>
<td>1</td>
<td>any</td>
<td>any</td>
<td>(at most) NEXPTIME</td>
<td>[Beneš et al., 2015] (consequence)</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( \leq p \cdot d^+ )</td>
<td>2</td>
<td>any</td>
<td>1</td>
<td>PSPACE(^{-\text{NEXP}-\text{hard}})</td>
<td>[Bundala and Ouaknine, 2014]</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( \geq p \cdot d^+ )</td>
<td>2</td>
<td>any</td>
<td>&gt; 1</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( x \gg p \cdot d ) None</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>undecidable</td>
<td>[Beneš et al., 2015]</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( x = p \cdot d ) None</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>undecidable</td>
<td>[Alur et al., 1993] (consequence)</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>( x \ll p \cdot d )</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>bounded ( x \gg p \cdot d ) ( x \ll p \cdot d \cdot d^+ )</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>(at most) PSPACE-complete</td>
<td>[Jovanović et al., 2015] (consequence)</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>( x \gg p \cdot d )</td>
<td>1</td>
<td>0</td>
<td>fixed</td>
<td>(at most) PTIME</td>
<td>[Miller, 2000] (consequence)</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>( x \gg p \cdot d )</td>
<td>1</td>
<td>0</td>
<td>any</td>
<td>(at most) NP-complete</td>
<td>[Miller, 2000] (consequence)</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>( x \ll p \cdot d )</td>
<td>any</td>
<td>2</td>
<td>any</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>( x = p \cdot d ) None</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>undecidable</td>
<td>[Beneš et al., 2015]</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>( x = p \cdot d ) None</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>undecidable</td>
<td>[Alur et al., 1993] (consequence)</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>( x \ll p \cdot d )</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>N</td>
<td>bounded ( x \gg p \cdot d ) ( x \ll p \cdot d \cdot d^+ )</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>PSPACE-complete</td>
<td>[Jovanović et al., 2015]</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>( x \gg p \cdot d )</td>
<td>1</td>
<td>0</td>
<td>fixed</td>
<td>PTIME</td>
<td>[Miller, 2000]</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>( x \ll p \cdot d )</td>
<td>any</td>
<td>1</td>
<td>1 or 2</td>
<td>any</td>
<td>open</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q [1; 2]</td>
<td>( x \gg p \cdot d )</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>undecidable</td>
<td>[Miller, 2000]</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>( x \ll p \cdot d )</td>
<td>any</td>
<td>2</td>
<td>0 or 1</td>
<td>any</td>
<td>open</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q [1; 2]</td>
<td>( x \gg p \cdot d )</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>undecidable</td>
<td>[Miller, 2000] (consequence)</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>( x = p \cdot d ) None</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>undecidable</td>
<td>[Alur et al., 1993]</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>( x \ll p \cdot d )</td>
<td>&lt; 2</td>
<td>3</td>
<td>2</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>( x \ll p \cdot d )</td>
<td>2</td>
<td>&lt; 3</td>
<td>2</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Q</td>
<td>( x \ll p \cdot d )</td>
<td>2</td>
<td>3</td>
<td>&lt; 2</td>
<td>open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q^+/R^+</td>
<td>Q^+/R^+</td>
<td>( x \ll p \cdot d )</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>undecidable</td>
<td>[Doyen, 2007]</td>
<td></td>
</tr>
</tbody>
</table>
PRP in details
Reachability Preservation

Key idea

“If we know a parameter valuation $v$ that reaches (resp. does not reach) $\blacksquare$, can we find other valuations around $v$ that reach (resp. do not reach) $\blacksquare$?”
Reachability Preservation

Key idea

“If we know a parameter valuation $v$ that reaches (resp. does not reach) $\square$, can we find other valuations around $v$ that reach (resp. do not reach) $\square$?”

Étienne André
Contributions to parametric timed model checking
Reachability Preservation

Key idea

“If we know a parameter valuation $v$ that reaches (resp. does not reach) $\square$, can we find other valuations around $v$ that reach (resp. do not reach) $\square$?”
Reachability Preservation: Undecidability

Problem (PREACH-emptiness)

Let $\mathcal{A}$ be a PTA, and $v$ a parameter valuation. Does there exist $v' \neq v$ such that $v'(\mathcal{A})$ preserves the reachability of $\blacklozenge$ in $v(\mathcal{A})$?

Theorem ([André, Lipari, Nguyen, Sun, NFM'/one.osf/five.osf])
PREACH-emptiness is undecidable.

Proof.
Reachability Preservation: Undecidability

Problem (PREACH-emptiness)

Let $A$ be a PTA, and $v$ a parameter valuation. Does there exist $v' \neq v$ such that $v'(A)$ preserves the reachability of in $v(A)$?

Theorem ([André, Lipari, Nguyen, Sun, NFM’15])

PREACH-emptiness is undecidable.
Reachability Preservation: Undecidability

Problem (PREACH-emptiness)

Let $\mathcal{A}$ be a PTA, and $\nu$ a parameter valuation. Does there exist $\nu' \neq \nu$ such that $\nu'(\mathcal{A})$ preserves the reachability of $\blacksquare$ in $\nu(\mathcal{A})$?

Theorem ([André, Lipari, Nguyen, Sun, NFM’15])

PREACH-emptiness is undecidable.

Proof.

[Diagram showing the states and transitions of the system, with labels $l_0$, $l_1$, $A_{2CM}$, $l_{halt}$, and $p = 0$.]
PRP: Parametric Reachability Preservation

Input: parameter valuation $v$

Output: constraint $K$ such that

1. $v \models K$, and
2. $\forall v' \models K$, $v'(A)$ preserves the reachability of $\bullet$ in $v(A)$

Inspired by EFsynth [Alur et al., 1993, Jovanović et al., 2015] and a variant of IM in [André and Soulat, 2011]
PRP: Parametric Reachability Preservation

Input: parameter valuation $v$
Output: constraint $K$ such that

1. $v \models K$, and
2. $\forall v' \models K, v'(A) \text{ preserves the reachability of } \bullet \text{ in } v(A)$

Inspired by EFsynth [Alur et al., 1993, Jovanović et al., 2015] and a variant of IM in [André and Soulat, 2011]
PRP: Case 1

As long as $\text{☐}$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!

![State space diagram](image-url)
PRP: Case 1

As long as ∟ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!

![Diagram of symbolic state space](image-url)
PRP: Case 1

As long as $\square$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!
PRP: Case 1

As long as \( \square \) is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in \( v(\mathcal{A}) \)!
PRP: Case 1

As long as $\square$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $\nu(\mathcal{A})$!
PRP: Case 1

As long as ☐ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(\mathcal{A})$!

![Diagram]
PRP: Case 1

As long as $\square$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(\mathcal{A})$!

![Symbolic state space diagram]
PRP: Case 1

As long as 😡 is not met…

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(\mathcal{A})$!
PRP: Case 1

As long as is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(\mathcal{A})$!
PRP: Case 1

As long as \(\square\) is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in \(v(\mathcal{A})\)!
As long as \( \lozenge \) is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in \( v(A) \)!

\[
\neg \land \cdots \land \neg
\]
Ensures a subset of the behaviors of \( v(A) \), and hence guarantees the unreachability of...
PRP: Case 1

As long as \( \square \) is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in \( v(A) \)!

\[
\text{Ensures a subset of the behaviors of } v(A), \text{ and hence guarantees the unreachability of...}
\]
PRP: Case 1

As long as is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!

![Symbolic state space diagram]

Ensures a subset of the behaviors of $v(A)$, and hence guarantees the unreachability of...
PRP: Case 1

As long as \( \square \) is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in \( v(A) \)!
PRP: Case 1

As long as ☐ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!

When no successors, and if $\neg \wedge \cdots \neg$ Ensures a subset of the behaviors of $v(A)$, and hence guarantees the unreachability of
As long as ☐ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!
As long as $\square$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $\nu(A)$!
PRP: Case 1

As long as is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!
PRP: Case 1

As long as $\square$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(A)$!
PRP: Case 1

As long as $\square$ is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(\mathcal{A})$!
As long as ° is not met...

- Explore the symbolic state space
- But do not explore the behaviors not present in $v(\mathcal{A})$!

When no successors, and if ° was never met:

- return $\neg ° \land \cdots \land \neg °$
- Ensures a subset of the behaviors of $v(\mathcal{A})$, and hence guarantees the unreachability of °
Questions

How do we know the possible behaviors of $v(A)$?
How do we know that a symbolic state of $A$ corresponds to a behavior of $v(A)$?

We could compute the zone graph of $v(A)$. But this is not necessary. In fact, we do not even need to know whether $v(A)$ reaches or not.

Trick: A symbolic state $(l, C)$ corresponds to a behavior of $v(A)$ iff $v|_C = C$. 
Questions

How do we know the possible behaviors of $v(\mathcal{A})$?
How do we know that a symbolic state of $\mathcal{A}$ corresponds to a behavior of $v(\mathcal{A})$?

We could compute the zone graph of $v(\mathcal{A})$.
But this is not necessary.
In fact, we do not even need to know whether $v(\mathcal{A})$ reaches $\blacksquare$ or not.
Questions

How do we know the possible behaviors of $v(\mathcal{A})$?
How do we know that a symbolic state of $\mathcal{A}$ corresponds to a behavior of $v(\mathcal{A})$?

We could compute the zone graph of $v(\mathcal{A})$.
But this is not necessary.
In fact, we do not even need to know whether $v(\mathcal{A})$ reaches $\square$ or not.

Trick

A symbolic state $(l, C')$ corresponds to a behavior of $v(\mathcal{A})$ iff $v \models C$. 
When \( \square \) is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in \( v(\mathcal{A}) \)
When $\square$ is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $\nu(\mathcal{A})$
When $\square$ is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $v(\mathcal{A})$
When is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in \( \nu(A) \)
When \( \square \) is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in \( v(A) \)
PRP: Case 2

When $\square$ is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $\nu(A)$
When \( \text{\textbullet} \) is met, switch to an \textit{EFsynth}-like algorithm…

- But still without exploring the behaviors not present in \( v(\mathcal{A}) \)
When \(\text{\textbullet} \) is met, switch to an \textsc{EFsynth}-like algorithm…

- But still without exploring the behaviors not present in \(v(A)\).
When $\text{red}$ is met, switch to an EFsynth-like algorithm…

But still without exploring the behaviors not present in $\nu(\mathcal{A})$
PRP: Case 2

When $\square$ is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $\nu(\mathcal{A})$
When is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $\nu(A)$
When is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $v(\mathcal{A})$
When  is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $v(\mathcal{A})$
When \( \square \) is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in \( \nu(\mathcal{A}) \)
When $\Box$ is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in $v(\mathcal{A})$
When \( \square \) is met, switch to an EFsynth-like algorithm…

- But still without exploring the behaviors not present in \( v(\mathcal{A}) \)
When $\circ$ is met, switch to an \textit{EFsynth}-like algorithm…

- But still without exploring the behaviors not present in $\nu(\mathcal{A})$

When no successors, and if $\circ$ was met:

- return $\circ \lor \cdots \lor \circ$
- \textbf{Guarantees the reachability of $\circ$}
Compositional parameter synthesis
Learning an abstraction: \text{LearnAbstr}(B, \nu(A), \text{AG}\neg L^{\otimes})

Input: \nu(A) \parallel B
Output: an abstraction \tilde{B} or a counter-example

\text{TL}^*: \text{learning algorithm} to compute a candidate abstraction \tilde{B} of an ERA B

[Lin et al., 2014]
Learning an abstraction: LearnAbstr\((B, \nu(A), \text{AG}\neg L^\circ)\)

Input: \(\nu(A) \parallel B\)
Output: an abstraction \(\tilde{B}\) or a counter-example

\[
\begin{align*}
\text{TL}^* & \quad \downarrow \tilde{B} \\
\nu(A) \parallel \tilde{B} & \models \varphi?
\end{align*}
\]

\textbf{TL}^*: learning algorithm to compute a candidate abstraction \(\tilde{B}\) of an ERA \(B\)

\cite{Lin et al., 2014}

\(\models\): can be checked using \textit{model checking}
Learning an abstraction: \textsf{LearnAbstr}(\mathcal{B}, \nu(\mathcal{A}), \text{AG}\neg L^\odot)

Input: \(\nu(\mathcal{A}) \parallel \mathcal{B}\)
Output: an abstraction \(\tilde{\mathcal{B}}\) or a counter-example

\[
\begin{align*}
\text{TL}^* & \quad \tilde{\mathcal{B}} \\
\nu(\mathcal{A}) \parallel \tilde{\mathcal{B}} & \models \phi? \\
\text{yes} & \\
\mathcal{B} & \models \tilde{\mathcal{B}}?
\end{align*}
\]

\textbf{TL}^*: learning algorithm to compute a candidate abstraction \(\tilde{\mathcal{B}}\) of an ERA \(\mathcal{B}\)
[Lin et al., 2014]
\models: can be checked using model checking
Learning an abstraction: \textbf{LearnAbstr}(B, \nu(A), AG\neg L^\odot)

Input: \(\nu(A) \parallel B\)
Output: an abstraction \(\tilde{B}\) or a counter-example

\[\text{TL}^*: \text{learning algorithm to compute a candidate abstraction } \tilde{B} \text{ of an ERA } B\]

[Lin et al., 2014]

\(\models: \text{can be checked using model checking}\)
Learning an abstraction: \textbf{LearnAbstr}(B, v(A), AG\neg L^\circ)

Input: \(v(A) \parallel B\)
Output: an abstraction \(\tilde{B}\) or a counter-example

\textbf{TL\*}: \textbf{learning algorithm} to compute a candidate abstraction \(\tilde{B}\) of an ERA \(B\)

\textbf{[Lin et al., 2014]}\
\(\models\): can be checked using \textbf{model checking}
Learning an abstraction: \textbf{LearnAbstr}(B, v(A), AG\neg\neg L^{\circ})

\begin{itemize}
  \item \textbf{Input}: $v(A) \parallel B$
  \item \textbf{Output}: an abstraction $\tilde{B}$ or a counter-example
\end{itemize}

\textbf{TL*}: \textit{learning algorithm} to compute a candidate abstraction $\tilde{B}$ of an ERA $B$

[\textit{Lin et al., 2014}]

$\models$: \textit{can be checked using model checking}
Learning an abstraction: \( \text{LearnAbstr}(B, \nu(A), \text{AG}\neg L^\circ) \)

Input: \( \nu(A) \parallel B \)
Output: an abstraction \( \widetilde{B} \) or a counter-example

\[ \text{TL}^*: \text{learning algorithm to compute a candidate abstraction } \widetilde{B} \text{ of an ERA } B \]

[Lin et al., 2014]

\( \models: \text{can be checked using model checking} \)

Refinement: can be performed using learning
Learning an abstraction: \textbf{LearnAbstr}($B, v(A), AG \neg L \odot$)

Input: $v(A) \parallel B$
Output: an abstraction $\tilde{B}$ or a counter-example

\begin{itemize}
  \item \textbf{Abstraction}: $\tilde{B}$ accepted by $B$
  \item \textbf{Counterex}: $\text{trace}(\rho)$
\end{itemize}

\textbf{TL*: learning algorithm} to compute a candidate abstraction $\tilde{B}$ of an ERA $B$

$\models$: can be checked using \textbf{model checking}

Refinement: can be performed using \textbf{learning}

\textbf{Contributions to parametric timed model checking}

\textbf{Étienne André}
Learning an abstraction: $\text{LearnAbstr}(B, v(A), AG \neg L)$

Input: $v(A) \parallel B$
Output: an abstraction $\tilde{B}$ or a counter-example

TL*: learning algorithm to compute a candidate abstraction $\tilde{B}$ of an ERA $B$

[Lin et al., 2014]

|$\models$: can be checked using model checking

Refinement: can be performed using learning
Learning an abstraction: \textbf{LearnAbstr}(B, v(A), AG\neg L^\odot)

Input: \(v(A) \parallel B\)
Output: an abstraction \(\tilde{B}\) or a counter-example

\textbf{TL}*: learning algorithm to compute a candidate abstraction \(\tilde{B}\) of an ERA \(B\)
[Lin et al., 2014]
\(\models\) can be checked using model checking
Refinement: can be performed using learning
Replaying a trace

Given a finite trace (i.e., a sequence of actions), we can replay it in the parametric framework

- i.e., find all parameter valuations for which this trace is feasible
- Using a symbolic semantics defined for PERAs

😊 Very cheap
Our overall procedure CompSynth

Key ideas:

- Iterate on integer points \( v \)
- Try to compute an abstraction \( \tilde{B} \) of the non-parametric component w.r.t. \( v(A) \) and \( \varphi \)
  - If succeed, synthesize “similar” valuations using PRP on \( A \parallel \tilde{B} \)
  - If fail, synthesize the valuations corresponding to the counterex.
Our overall procedure CompSynth

Key ideas:

- Iterate on integer points \( v \)
- Try to compute an abstraction \( \tilde{B} \) of the non-parametric component w.r.t. \( v(A) \) and \( \varphi \)
  - If succeed, synthesize “similar” valuations using \( \text{PRP} \) on \( A \parallel \tilde{B} \)
  - If fail, synthesize the valuations corresponding to the counterex.

```
K_{bad} \leftarrow \bot; \quad K_{good} \leftarrow \bot

while there is an integer point not covered by \( K_{bad} \) or \( K_{good} \) do

  Pick such a point \( v \)

  switch LearnAbstr(\( B, v(A), AG\neg L^{\diamond} \)) do

  case Abstraction(\( \tilde{B} \)) do
    \( K_{good} \leftarrow K_{good} \cup \text{PRP}(A \parallel \tilde{B}, v, L^{\diamond}) \)
  case Counterex(\( \tau \)) do
    \( K_{bad} \leftarrow K_{bad} \cup \text{ReplayTrace}(A \parallel B, \tau) \)

return \((K_{good}, K_{bad})\)
```
Parametric task automata
A unified formalism: Parametric task automata

Extension of task automata [Norström et al., 1999, Fersman et al., 2007] with parameters

\[ l_0 \quad t_0 \quad x > 10 \quad x := 0 \quad x = 40 \quad x := 0 \]

Priorities
\[ t_0 > t_2 > t_1 > t_3 \]

<table>
<thead>
<tr>
<th>Task</th>
<th>B</th>
<th>W</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 )</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>4</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Étienne André
Contributions to parametric timed model checking
A unified formalism: Parametric task automata

Extension of task automata [Norström et al., 1999, Fersman et al., 2007] with parameters [André, FMICS’17]

\[ \begin{align*}
    x &= 20 \\
    x &= 0 \\
    x &= 0 \\
    x &= 40 \\
\end{align*} \]

Priorities

\[ t_0 > t_2 > t_1 > t_3 \]

<table>
<thead>
<tr>
<th>Task</th>
<th>B</th>
<th>W</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 )</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>4</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>0</td>
<td>1</td>
<td>( p' )</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>
A unified formalism: Parametric task automata

Extension of task automata [Norström et al., 1999, Fersman et al., 2007] with parameters

\[ x > p \]
\[ x := 0 \]

Priorities
\[ t_0 > t_2 > t_1 > t_3 \]

Parametric task automata can model
- Preemption
- Periodic tasks, sporadic tasks, pseudo-periodic tasks...
- Dependencies between tasks
- Offset, jitter
- Uncertainty
- Uniprocessor only
Parametric task automata: theory and practice

Schedulability-emptiness ("is the set of valuations for which the system is schedulable empty?")

- **Undecidable** in general
- **Decidable** under some assumptions

[André, FMICS’17]
Parametric task automata: theory and practice

Schedulability-emptiness ("is the set of valuations for which the system is schedulable empty?")

- **Undecidable** in general
- **Decidable** under some assumptions [André, FMICS’17]

Implementation in IMITATOR

- Translation into a network of parametric stopwatch automata
- Schedulability analysis
- Parametric and/or robust schedulability analysis
The FMTV Challenge in details
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:
  - \( P1 \in [40 - 0.004, 40 + 0.004] \)
  - \( P3 \in \left[ \frac{40}{3} - \frac{1}{150}, \frac{40}{3} + \frac{1}{150} \right] \)
  - \( P4 \in [40 - 0.004, 40 + 0.004] \)

- Parameters:
  - \( P1_{\text{uncertain}} \)
  - \( P3_{\text{uncertain}} \)
  - \( P4_{\text{uncertain}} \)

- The end-to-end latency (another parameter): \( E2E \)
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): $E2E$

- 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?)
Outline

1 Decidability

2 Efficient synthesis

3 Applications to schedulability analysis

4 Perspectives
   ■ The PTA model for $n = 1$
The initialization automaton

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

\[ c_{kT1T2} = WCET_1: \]

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

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

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

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

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

\[ \text{frame}_{in\_3} := 0 \]
\[ \text{frame}_{in\_3} := 2 \]
The initialization automaton

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

- Buffer: buffer\_3,4 := 0, highest\_3,4 := 0
- Camera:
  - camera0
  - camera1
  - camera2
  - camera3

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

\[ \text{reg}_2,3 := 0, \text{reg}_2,3 := 3 \]
The initialization automaton

\[
\begin{align*}
\text{ckT1T2} &= \text{WCET}_1; \\
\text{buffer}_{3,4} &:= 0 \\
\text{highest}_{3,4} &:= 0 \\
\text{buffer}_{3,4} &:= 1 \\
\text{highest}_{3,4} &:= 1 \\
\text{frame}_{\text{in},3} &:= 0 \\
\text{frame}_{\text{in},3} &:= 2 \\
\text{reg}_{2,3} &:= 0 \\
\text{reg}_{2,3} &:= 3 \\
\text{T1T2} : \text{wcet}_1 + \text{wcl}_2 &\geq \text{ckT1T2}; \\
\text{T1T2} &:# \text{start} \\
\end{align*}
\]
The initialization automaton

\[ \text{camera}0 \quad \text{ckT}1\text{T}2 = \text{WCET}_1: \]

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

\[ \text{camera}1 \quad \text{ckT}1\text{T}2 = \text{WCET}_1: \]

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

\[ \text{frame}_{\text{in}} _{3} := 0 \quad \text{frame}_{\text{in}} _{3} := 2 \]

\[ \text{camera}2 \quad \text{ckT}1\text{T}2 = \text{WCET}_1: \]

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

\[ \text{camera}3 \quad \text{ckT}1\text{T}2 = \text{WCET}_1: \]

\[ \text{start} \]

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

\[ \text{T}1\text{T}2\text{done} \quad \text{ckT}1\text{T}2 \geq \text{WCET}_1 + \text{BCL}_2 \]

\[ \text{T}1\text{T}2\text{done} \quad \text{T}2\text{done} \quad \text{reg}_{2,3} := \text{target} \]

\[ \text{T}1\text{T}2\text{done} \quad \text{WCET}_1 + \text{WCL}_2 \geq \text{ckT}1\text{T}2: \]
Task T3

T3preinit

WCET_{T3} \geq \text{ck}_{T3}

\text{P3}_{\text{uncertain}} \geq \text{ck}_{T3}

\text{ck}_{T3} := 0

\text{frame}_{in_{3}} := \text{reg_{2,3}}

WCET_{T3} = \text{ck}_{T3} \land \text{buffer}_{3,4} = 0 \land \text{frame}_{in_{3}} > \text{highest}_{3,4}

T3_{done}\text{write by } T3()

WCET_{T3} = \text{ck}_{T3} \land \text{buffer}_{3,4} > 0 \land \text{highest}_{3,4} \geq \text{frame}_{in_{3}}

Étienne André
Contributions to parametric timed model checking
Task T3

\[ \text{WCET}_3 \geq \text{ckT3} \]

\[ \text{WCET}_3 \geq \text{ckT3} \]
Task T3

- T3preinit
  - WCET₃ ≥ ckT3

- T3process
  - WCET₃ ≥ ckT3

- T3done
  - P3_uncertain ≥ ckT3
Task T3

T3preinit

\[ WCET_3 \geq ckT3 \]

\[ start \]

T3process

\[ WCET_3 \geq ckT3 \]

P3_uncertain

\[ = \]

\[ ckT3 \]

\[ T3\_start \]

\[ ckT3 := 0 \]

\[ frame\_in\_3 \]

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

T3done

\[ P3\_uncertain \geq ckT3 \]
Task T3

\[ WCET_3 \geq ckT3 \]

\[ WCET_3 \geq ckT3 \]

\[ P_3_{\text{uncertain}} = ckT3 \]
\[ T_3_{\text{start}} \]
\[ ckT3 := 0 \]
\[ \text{frame}_{\text{in}}_3 := \text{reg}_{2,3} \]

\[ WCET_3 = ckT3 \land \text{buffer}_{3,4} = 0 \land \text{frame}_{\text{in}}_3 > \text{highest}_{3,4} \land T_3_{\text{done}} \land \text{write}_{\text{by}}_T3() \]

\[ P_3_{\text{uncertain}} \geq ckT3 \]
Task T3

T3preinit

WCET_3 \geq ckt_3

start

T3process

WCET_3 \geq ckt_3

P3_uncertain

= ckt_3
T3_start
ckt_3 := 0
frame_in_3 := reg_{2,3}

WCET_3

= ckt_3
\land buffer_{3,4} = 0
\land frame_in_3 > highest_{3,4}
T3_done
write_by_T3()

T3done

P3_uncertain \geq ckt_3

WCET_3

= ckt_3
\land buffer_{3,4} > 0
T3_done

start
Task T3

\[
\begin{align*}
\text{WCET}_3 & \geq \text{ckT3} \\
\text{start} & \\
\end{align*}
\]

\[
\begin{align*}
\text{P3_uncertain} & = \text{ckT3} \\
\text{T3_start} & \\
\text{ckT3} & := 0 \\
\text{frame_in_3} & := \text{reg2,3} \\
\end{align*}
\]

\[
\begin{align*}
\text{WCET}_3 & = \text{ckT3} \\
\text{buffer}_{3,4} & = 0 \\
\text{frame_in_3} & > \text{highest}_{3,4} \\
\text{T3_done} & \\
\text{write_by_T3}() & \\
\end{align*}
\]

\[
\begin{align*}
\text{WCET}_3 & = \text{ckT3} \\
\text{buffer}_{3,4} & > 0 \\
\text{T3_done} & \\
\end{align*}
\]

\[
\begin{align*}
\text{P3_uncertain} & \geq \text{ckT3} \\
\end{align*}
\]

\[
\begin{align*}
\text{frame_in_3} & = \text{reg2,3} \\
\end{align*}
\]
**Task T4**

P4\_uncertain ≥ ckT4
Task T4

\[
\begin{align*}
P_{4\text{ uncertain}} &= ckt_4 \\
&\land buffer_{3,4} > 0 \\
ckt_4 &:= 0 \\
\text{read\_by\_T4}() \\
\end{align*}
\]

10 \geq ckt_4
Task T4

\[
P_{4\text{ uncertain}} = ck_{T4} \\
\land buffer_{3,4} = 0 \\
ck_{T4} := 0
\]

\[
P_{4\text{ uncertain}} = ck_{T4} \\
\land buffer_{3,4} > 0 \\
ck_{T4} := 0 \\
read_{by\_T4()}
\]

T4wait

\[
P_{4\text{ uncertain}} \geq ck_{T4}
\]

T4process_nonempty

\[
10 \geq ck_{T4}
\]
Task T4

P4_uncertain = ckT4
∧ buffer_{3,4} = 0
ckT4 := 0

T4_wait

P4_uncertain = ckT4
∧ buffer_{3,4} > 0
ckT4 := 0
read_by_T4()

T4_process_nonempty

10 = ckT4
∧
frame_in_4 \neq \text{target}

10 \geq ckT4
Task T4

P4_uncertain = ckT4 ∧ buffer_{3,4} = 0
ckT4 := 0

T4wait

P4_uncertain = ckT4 ∧ buffer_{3,4} > 0
ckT4 := 0
read_by_T4()

T4process_nonempty

P4_uncertain ≥ ckT4
10 = ckT4 ∧ frame_in_{4} ≠ target

T4end_ok

10 = ckT4 ∧ frame_in_{4} = target ∧ ckT1T2 = E2E
ckT4 := 0

ckT4 = 0
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: Deepwater Horizon Offshore Drilling Platform on Fire  
Author: ideum  
Source: https://secure.flickr.com/photos/ideum/471481781/  
License: CC BY-SA 2.0

Title: DA-SC-88-01663  
Author: imcomkorea  
Source: https://secure.flickr.com/photos/imcomkorea/30786760/  
License: CC BY-NC-ND 2.0

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
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)

(TEX source available on demand)

Author: Étienne André

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