Integer-Complete Parameter Synthesis
for Bounded Parametric Timed Automata

Étienne André\textsuperscript{1,2}, Didier Lime\textsuperscript{2}, Olivier H. Roux\textsuperscript{2}

\textsuperscript{1}LIPN, Université Paris 13, Sorbonne Paris Cité, CNRS, France
\textsuperscript{2}École Centrale de Nantes, IRCCyN, CNRS, UMR 6597, France
Context: Formal verification of timed systems

- Model checking

A model of the system

A property to be satisfied

\( \text{is unreachable} \)
**Context: Formal verification of timed systems**

- **Model checking**

![Diagram of a model of a system](image)

A **model** of the system

A **property** to be satisfied

Question: does the model of the system **satisfy** the property?
Context: Formal verification of timed systems

- Model checking

A model of the system

A property to be satisfied

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

Yes

No

Counterexample
Beyond model checking: parameter synthesis

- Timed systems are characterized by a set of timing constants
  - “The packet transmission lasts for 50 ms”
  - “The sensor reads the value every 10 s”

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

- Challenges
  - Numerous verifications: is the system correct for any value within [40; 60]?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?
Beyond model checking: parameter synthesis

- Timed systems are characterized by a set of timing constants
  - “The packet transmission lasts for 50 ms”
  - “The sensor reads the value every 10 s”

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

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?

- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)
  - Find good values for the parameters
Outline

1 Preliminaries

2 Previous Works on Parameter Synthesis

3 Integer-Complete Dense Synthesis

4 Implementation in ROMÉO

5 Conclusion and Perspectives
Outline

1 Preliminaries

2 Previous Works on Parameter Synthesis

3 Integer-Complete Dense Synthesis

4 Implementation in ROMÉO

5 Conclusion and Perspectives
Timed automaton (TA)

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

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

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
- Real-valued variables evolving linearly at the same rate
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
- Real-valued variables evolving linearly at the same rate

- Features
  - Location invariant: property to be verified to stay at a location

---

### Diagram

- States:
  - Green state
  - Blue state
  - Red state

- Edges:
  - Directed edges labeled with actions:
    - `press?`
    - `coffee!`
    - `cup!`

- Guards:
  - $y \leq 5$
  - $y \leq 8$

- Initial state:
  - Green state

- Transitions:
  - From Green to Blue: $y \leq 5$
  - From Blue to Red: $y \leq 8$
**Timed automaton (TA)**

- Finite state automaton (sets of **locations** and **actions**) augmented with a set $X$ of **clocks** [Alur and Dill, 1994]
  - Real-valued variables evolving linearly at the same rate

- **Features**
  - Location **invariant**: property to be verified to stay at a location
  - Transition **guard**: property to be verified to enable a transition

---

**Example:**

- $y = 8$
- `coffee!`

**Transition:**

- Press? $x \geq 1$
- Press? $y \leq 5$
- Cup! $y = 5$
- $y \leq 8$
Timed automaton (TA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks [Alur and Dill, 1994]
- Real-valued variables evolving linearly at the same rate

**Features**

- Location invariant: property to be verified to stay at a location
- Transition guard: property to be verified to enable a transition
- Clock reset: some of the clocks can be set to 0 at each transition

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

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

\[
\begin{align*}
    x &= 1 \\
    y &= 5 \\
    \text{cup!}
\end{align*}
\]

\[
\begin{align*}
    y &\leq 5 \\
    \text{press?}
\end{align*}
\]

\[
\begin{align*}
    x &:= 0 \\
    y &:= 0
\end{align*}
\]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time
  - Possible concrete runs for the coffee machine
    - Coffee with no sugar
      
      \[
      \begin{array}{c|c}
      x & 0 \\
      y & 0 \\
      \end{array}
      \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - Coffee with no sugar

\[
\begin{array}{ccc}
  x & 0 & 15.4 \\
  y & 0 & 15.4 \\
\end{array}
\]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions
  or elapsing of time

- Possible concrete runs for the coffee machine
  - **Coffee with no sugar**

\[
\begin{array}{ccc}
x & 0 & 15.4 & 0 \\
y & 0 & 15.4 & 0 \\
\end{array}
\]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

  Possible concrete runs for the coffee machine

  - **Coffee with no sugar**

    \[
    \begin{array}{cccccc}
    \text{press?} & \rightarrow & 5 \\
    x & 0 & 15.4 & 0 & 5 \\
    y & 0 & 15.4 & 0 & 5 \\
    \end{array}
    \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

  Possible concrete runs for the coffee machine

  - **Coffee with no sugar**

    \[
    \begin{array}{cccccc}
    x & 0 & 15.4 & 0 & 5 & 5 \\
    y & 0 & 15.4 & 0 & 5 & 5 \\
    \end{array}
    \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine

  - **Coffee with no sugar**
    
    \[
    \begin{array}{ccc}
    x & 0 & 15.4 & 0 & 5 & 5 & 8 \\
    y & 0 & 15.4 & 0 & 5 & 5 & 8 \\
    \end{array}
    \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine

  - Coffee with no sugar

<table>
<thead>
<tr>
<th></th>
<th>(x)</th>
<th>(y)</th>
<th>(15.4)</th>
<th>press?</th>
<th>5</th>
<th>cup!</th>
<th>3</th>
<th>coffee!</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine

  - Coffee with no sugar

    \[
    \begin{array}{ccccccc}
    x & 0 & 15.4 & 0 & 5 & 5 & 8 \\
    y & 0 & 15.4 & 0 & 5 & 5 & 8 \\
    \end{array}
    \]

  - Coffee with 2 doses of sugar

    \[
    \begin{array}{ccccccc}
    x & 0 & & & & \text{press?} & \text{cup!} \\
    y & 0 & & & & \text{press?} & \text{cup!} \\
    \end{array}
    \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine

  - Coffee with no sugar

  ![Graph for coffee with no sugar](image)

  - Coffee with 2 doses of sugar

  ![Graph for coffee with 2 doses of sugar](image)
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - **Coffee with no sugar**
  - **Coffee with 2 doses of sugar**
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
    - | State | X | Y |
    - | --- | --- | --- |
    - | 15.4 | 0 | 0 |
    - | press? | 5 | 5 |
    - | cup! | 3 | 8 |
    - | coffee! | 8 | 8 |
  - Coffee with 2 doses of sugar
    - | State | X | Y |
    - | --- | --- | --- |
    - | press? | 0 | 0 |
    - | 1.5 | 1.5 | 0 |
    - | press? | 1.5 | 1.5 | 1.5 |
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - **Coffee with no sugar**
    - \(x\) \(0\) \(15.4\) \(0\) \(5\) \(5\) \(8\) \(8\)
    - \(y\) \(0\) \(15.4\) \(0\) \(5\) \(5\) \(8\) \(8\)
  - **Coffee with 2 doses of sugar**
    - \(x\) \(0\) \(0\) \(1.5\) \(0\) \(2.7\)
    - \(y\) \(0\) \(0\) \(1.5\) \(1.5\) \(4.2\)
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
    - \(x\) \(y\)
      - 0 15.4 0 5 5 8 8
      - 0 15.4 0 5 5 8 8

  - Coffee with 2 doses of sugar
    - \(x\) \(y\)
      - 0 0 1.5 0 2.7 0
      - 0 0 1.5 1.5 4.2 4.2
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  
  - **Coffee with no sugar**
    
    \[
    \begin{array}{cccccccc}
    x & 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    y & 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    \end{array}
    \]

  - **Coffee with 2 doses of sugar**
    
    \[
    \begin{array}{cccccccc}
    x & 0 & 0 & 1.5 & 0 & 2.7 & 0 & 0.8 \\
    y & 0 & 0 & 1.5 & 1.5 & 4.2 & 4.2 & 5 \\
    \end{array}
    \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - **Coffee with no sugar**
    - States and actions:
      - Press button 15.4
      - 5 seconds
      - Cup of coffee
      - 3 seconds
      - Coffee
    - Clock values:
      - \(x\): 0, 15.4, 0, 5, 5, 8, 8
      - \(y\): 0, 15.4, 0, 5, 5, 8, 8
  
  - **Coffee with 2 doses of sugar**
    - States and actions:
      - Press button 1.5
      - 2.7 seconds
      - Press button 0.8
      - Cup of coffee
    - Clock values:
      - \(x\): 0, 0, 1.5, 0, 2.7, 0, 0.8, 0.8
      - \(y\): 0, 0, 1.5, 1.5, 4.2, 4.2, 5, 5
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine
  - **Coffee with no sugar**
    \[
    \begin{align*}
    x &\quad 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    y &\quad 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    \end{align*}
    \]
  - **Coffee with 2 doses of sugar**
    \[
    \begin{align*}
    x &\quad 0 & 0 & 1.5 & 0 & 2.7 & 0 & 8 & 0.8 & 0.8 & 3.8 \\
    y &\quad 0 & 0 & 1.5 & 1.5 & 4.2 & 4.2 & 5 & 5 & 8 \\
    \end{align*}
    \]
Concrete semantics of timed automata

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

- **Concrete run**: alternating sequence of concrete states and actions or elapsing of time

- Possible concrete runs for the coffee machine

  - **Coffee with no sugar**
    \[
    \begin{array}{ccccccccc}
    & 15.4 & \text{press?} & 5 & \text{cup!} & 3 & \text{coffee!} \\
    x & 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    y & 0 & 15.4 & 0 & 5 & 5 & 8 & 8 \\
    \end{array}
    \]

  - **Coffee with 2 doses of sugar**
    \[
    \begin{array}{ccccccccc}
    & \text{press?} & 1.5 & \text{press?} & 2.7 & \text{press?} & 0.8 & \text{cup!} & 3 & \text{coffee!} \\
    x & 0 & 0 & 1.5 & 0 & 2.7 & 0 & 0.8 & 3.8 & 3.8 \\
    y & 0 & 0 & 1.5 & 1.5 & 4.2 & 4.2 & 5 & 8 & 8 \\
    \end{array}
    \]
Parametric Timed Automaton (PTA)

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

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

```
$y = p_3$
coffee!
```

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

```
$y \leq p_2$
press?
x := 0
```

```
x \geq p_1$
cup!
y = p_2
```

$y \leq 8$
Symbolic semantics of a PTA

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a polyhedron (conjunction of inequalities) over \(X\) and \(P\)
Symbolic semantics of a PTA

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a **polyhedron** (conjunction of inequalities) over \(X\) and \(P\)

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

- **Symbolic state** of a PTA: pair $(l, C)$, where
  - $l$ is a location,
  - $C$ is a polyhedron (conjunction of inequalities) over $X$ and $P$

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

- **Example**

  $x \leq p_1$
  $y := 0$
  $x \leq p_3$

  Possible symbolic run for this PTA

  $x = y$
  $x \leq p_1$
Symbolic semantics of a PTA

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a polyhedron (conjunction of inequalities) over \(X\) and \(P\)

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

**Example**

- \(x \geq p_2\)
- \(a\)
- \(x = y\)
- \(x \leq p_1\)
- \(y := 0\)
- \(x \leq p_3\)
- \(b\)
- \(x := 0\)
- \(y \geq p_4\)
- \(c\)

- Possible symbolic run for this PTA
Symbolic semantics of a PTA

- **Symbolic state** of a PTA: pair \((l, C)\), where
  - \(l\) is a location,
  - \(C\) is a polyhedron (conjunction of inequalities) over \(X\) and \(P\)

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

**Example**

\[\begin{align*}
x &\geq p_2 \\
x &\leq p_1 \\
y &:= 0 \\
x &\leq p_3 \\
b &
\end{align*}\]

**Possible symbolic run for this PTA**

\[\begin{align*}
x &= y \\
x &\leq p_1 \\
x - y &\leq p_1 \\
x - y &\geq p_2 \\
x &\leq p_3 \\
p_1 &\geq p_2 \\
y &\geq x \\
y - x &\leq p_3
\end{align*}\]
Valuation of a PTA

Given a PTA $\mathcal{A}$ and a parameter valuation $\nu$, we denote by $\nu(\mathcal{A})$ the (non-parametric) timed automaton where all parameters are valuated by $\nu$. 
Objective: Computation problems

Definition (reachability synthesis (EF))

Input: a PTA $\mathcal{A}$ and a set of locations $G$
Problem: Synthesize all parameter valuations $\nu$ such that there exists a run of $\nu(\mathcal{A})$ reaching a location $l \in G$
Objective: Computation problems

Definition (reachability synthesis (EF))

Input: a PTA $\mathcal{A}$ and a set of locations $\mathcal{G}$
Problem: Synthesize all parameter valuations $\nu$ such that there exists a run of $\nu(\mathcal{A})$ reaching a location $l \in \mathcal{G}$

Definition (unavoidability synthesis (AF))

Input: a PTA $\mathcal{A}$ and a set of locations $\mathcal{G}$
Problem: Synthesize all parameter valuations $\nu$ such that all runs of $\nu(\mathcal{A})$ eventually reach a location $l \in \mathcal{G}$
Outline

1 Preliminaries

2 Previous Works on Parameter Synthesis

3 Integer-Complete Dense Synthesis

4 Implementation in ROMÉO

5 Conclusion and Perspectives
Decidability results: reachability

Reachability emptiness

Reachability emptiness (“does there exist at least one parameter valuation reaching a given location \( l \)?”) is \textit{undecidable} for \( \text{PTA} \)

- even with a single parametric clock \[ \text{[Miller, 2000]} \]
- even with only strict constraints \[ \text{[Doyen, 2007]} \]
- even with a single integer-valued parameter \[ \text{[Beneš et al., 2015]} \]
Decidability results: unavoidability

Reachability emptiness

Unavoidability emptiness ("does there exist at least one parameter valuation such that all runs reach a given location l?") is undecidable for PTA, even with a single bounded parameter

[Jovanović et al., 2015]
Synthesis of bounded integers

What if parameters are bounded integers...?
Synthesis of bounded integers

What if parameters are bounded integers...?

Bounded integers

Reachability and unavoidability emptiness are decidable (and PSPACE-complete) for PTA with bounded integers [Jovanović et al., 2015]
Synthesis of bounded integers

What if parameters are bounded integers...?

Bounded integers

Reachability and unavoidability emptiness are decidable (and PSPACE-complete) for PTA with bounded integers [Jovanović et al., 2015]

Two algorithms:

- IEF: reachability synthesis
- IAF: unavoidability synthesis
Synthesis of bounded integers: How?

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])

Smarter: symbolic algorithm [Jovanović et al., 2015]

More efficient than exhaustive enumeration with Uppaal

É. André (Paris 13 / Nantes)
Synthesis of bounded integers: How?

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])

Smarter: symbolic algorithm [Jovano vić et al., 2015]

More efficient than exhaustive enumeration with Uppaal
Synthesis of bounded integers: How?
Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])

Smarter: symbolic algorithm [Jovanović et al., 2015]
- More efficient than exhaustive enumeration with UPPAAL
**Synthesis of bounded integers: How?**

Naive idea: enumerate all integers, and check the TA (which is PSPACE-complete [Alur and Dill, 1994])

![Diagram](image1)

Smarter: *symbolic algorithm* [Jovanović et al., 2015]

- More efficient than exhaustive enumeration with UPPAAL

![Diagram](image2)
Integer hull of a polyhedron

Definition (integer hull)

Let \( C \) be a polyhedron.
The integer hull of \( C \) is

\[
\mathrm{IH}(C) = \text{Conv}(\mathrm{IV}(C))
\]

(\( \text{Conv} \): convex hull; \( \mathrm{IV} \) set of vectors with integer coordinates)
Integer hull of a polyhedron

Definition (integer hull)

Let $C$ be a polyhedron. The integer hull of $C$ is

$$\text{IH}(C) = \text{Conv}(\text{IV}(C))$$

(Conv: convex hull; IV set of vectors with integer coordinates)
Integer hull of a polyhedron

Definition (integer hull)

Let $C$ be a polyhedron.
The integer hull of $C$ is

\[ \text{IH}(C) = \text{Conv}(\text{IV}(C)) \]

(Conv: convex hull; IV set of vectors with integer coordinates)
Reachability synthesis

Algorithm \( \text{EF}(A, G) \)

\[ K \leftarrow \bot \]

Add the initial state to the waiting list

while the waiting list is not empty

Pick a symbolic state \((l, C)\) from the waiting list

if \( l \in G \) then \( K \leftarrow K \lor C \uparrow_p \)

else if \((l, C) = (l', C')\), for some \((l', C')\) met before

then do not explore further this branch

else store \((l, C)\) and add its successors to the waiting list

return \( K \)
Reachability synthesis of bounded integers using $\text{IH}$

**Algorithm $\text{IEF}(\mathcal{A}, G)$ [Jovanović et al., 2015]**

$K \leftarrow \bot$

Add the initial state to the waiting list

while the waiting list is not empty

    Pick a symbolic state $(l, C)$ from the waiting list

    if $l \in G$ then $K \leftarrow K \lor \text{IH}(C)_{\downarrow_p}$

    else if $(l, \text{IH}(C)) = (l', \text{IH}(C'))$, for some $(l', C')$ met before

        then do not explore further this branch

    else store $(l, \text{IH}(C))$ and add its successors to the waiting list

return $K$
Outline

1. Preliminaries
2. Previous Works on Parameter Synthesis
3. Integer-Complete Dense Synthesis
4. Implementation in ROMÉO
5. Conclusion and Perspectives
What about the dense result?

IEF and IAF return **symbolic** sets of **integer** valuations.
What about the dense result?

IEF and IAF return **symbolic** sets of **integer** valuations

Can we interpret the result of IEF and IAF over dense parameter valuations?
What about the dense result?

|EF and |AF return symbolic sets of integer valuations

Can we interpret the result of |EF and |AF over dense parameter valuations?

😊 For |EF: yes! ...but it may not terminate

(example in paper)
What about the dense result?

IEF and IAF return *symbolic* sets of *integer* valuations

Can we interpret the result of IEF and IAF over dense parameter valuations?

- For IEF: yes! … but it *may not terminate*  
  (example in paper)

- For IAF: no! May yield *incorrect valuations*  
  (counter-example in paper)
A parametric extrapolation for PTA

**Definition (M-extrapolation)**

Let $M$ be the largest constant in $A$ (including the bounds on the parameters), let $x$ be a clock. The $(M, x)$-extrapolation is

$$\text{Ext}_x^M(C) = \left(C \cap (x \leq M)\right) \cup \text{Cyl}_x \left(C \cap (x > M)\right) \cap (x > M).$$
A parametric extrapolation for PTA

**Definition ($M$-extrapolation)**

Let $M$ be the largest constant in $A$ (including the bounds on the parameters), let $x$ be a clock. The $(M,x)$-extrapolation is

$$\text{Ext}_x^M(C) = (C \cap (x \leq M)) \cup \text{Cyl}_x(C \cap (x > M)) \cap (x > M).$$

Generalized to $(M,X)$-extrapolation by applying to all clocks.
Integer reachability synthesis

Algorithm \( \text{IEF}(\mathcal{A}, \mathcal{G}) \)

\[ K \leftarrow \perp \]

Add the initial state to the waiting list

while the waiting list is not empty

Pick a symbolic state \((l, C)\) from the waiting list

if \( l \in \mathcal{G} \) then \( K \leftarrow K \lor \left\lfloor \text{IF}(C) \right\rfloor_p \)

else if \((l, \left\lfloor \text{IF}(C) \right\rfloor) = (l', \left\lfloor \text{IF}(C') \right\rfloor), \)

then do not explore further this branch

else store \((l, \left\lfloor \text{IF}(C) \right\rfloor)\) and add its successors to the waiting list

return \( K \)
Integer complete reachability synthesis RIEF

Algorithm $\text{RIEF}(\mathcal{A}, G)$

$K \leftarrow \bot$

Add the initial state to the waiting list

while the waiting list is not empty

Pick a symbolic state $(l, C)$ from the waiting list

if $l \in G$ then $K \leftarrow K \lor C \downarrow_P$

else if $(l, \text{IH}(\text{Ext}^M_X(C))) = (l', \text{IH}(\text{Ext}^M_X(C'))),$

then do not explore further this branch

for some $(l', C')$ met before

else store $(l, \text{IH}(\text{Ext}^M_X(C)))$ and add its successors to the waiting list

return $K$
Termination of RIEF

Theorem

For any PTA $\mathcal{A}$ with bounded parameters, the computation of $\text{RIEF}(\mathcal{A}, G)$ terminates.

Proof (hint).

From the finiteness of the number of integer hulls of $(M, X)$-extrapolations of possible states.
**Characterization of RIEF**

**Theorem**

Given a PTA $A$ with bounded parameters, $\text{RIEF}(A, G)$ contains

1. no valuation that is not a solution of $\text{EF}(A, G)$  
   
   ![Diagram of RIEF](image)

   [correctness]
Characterization of RIEF

Theorem

Given a PTA $A$ with bounded parameters, $\text{RIEF}(A, G)$ contains

1. no valuation that is not a solution of $\text{EF}(A, G)$ \[\text{correctness}\]
2. all the integer parameter valuations solution of $\text{EF}$ \[\text{[integer-completeness]}\]
Characterization of RIEF

Theorem

Given a PTA \( A \) with bounded parameters, \( \text{RIEF}(A, G) \) contains

1. no valuation that is not a solution of \( \text{EF}(A, G) \) \hspace{1cm} [correctness]
2. all the integer parameter valuations solution of \( \text{EF} \) \hspace{1cm} [integer-completeness]
3. all the rational valuations in the parametric zones computed by the symbolic exploration \hspace{1cm} [?]
Unavoidability

Algorithm RIAF computing parameter valuations such that a set of locations is unavoidable

Similar principle and similar results (see paper)
Outline

1 Preliminaries

2 Previous Works on Parameter Synthesis

3 Integer-Complete Dense Synthesis

4 Implementation in ROMÉO

5 Conclusion and Perspectives
Roméo

Model checker for parametric time Petri nets and PTA [Lime et al., 2009]

Uses the Parma Polyhedra Library (PPL) for operations on polyhedra [Bagnara et al., 2008]

Available in the open source CeCILL license

www.ROMEO.xxx
Case study: scheduling example

Three tasks $\tau_1, \tau_2, \tau_3$ scheduled using static priorities ($\tau_1 > \tau_2 > \tau_3$) in a non-preemptive manner [Jovanović et al., 2015]

Task $\tau_1$: periodic with period $a$ and a non-deterministic duration in $[10, b]$ 

Task $\tau_2$: minimal activation time of $2a$ and a non-deterministic duration in $[18, 28]$ 

Task $\tau_3$: periodic with period $3a$ and a non-deterministic duration in $[20, 28]$.

Each task: deadline equal to its period
Case study: scheduling example

Three tasks $\tau_1, \tau_2, \tau_3$ scheduled using static priorities ($\tau_1 > \tau_2 > \tau_3$) in a non-preemptive manner [Jovanović et al., 2015]

Task $\tau_1$: periodic with period $a$ and a non-deterministic duration in $[10, b]$

Task $\tau_2$: minimal activation time of $2a$ and a non-deterministic duration in $[18, 28]$

Task $\tau_3$: periodic with period $3a$ and a non-deterministic duration in $[20, 28]$

Each task: deadline equal to its period

Goal: synthesize parameter valuations ensuring that the system does not reach a deadline violation.
Experiments

Bounded parameters: \( a \in [0, 50] \) and \( b \in [0, 50] \)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Result</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEF</td>
<td>discrete</td>
<td>7.4 s</td>
</tr>
<tr>
<td>RIEF</td>
<td>dense</td>
<td>12.7 s</td>
</tr>
</tbody>
</table>
Experiments

Bounded parameters: $a \in [0, 50]$ and $b \in [0, 50]$

Result obtained by $\text{IEF}$: $a \geq 34$, $b \geq 10$, $a - b \geq 24$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Result</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEF</td>
<td>discrete</td>
<td>7.4 s</td>
</tr>
<tr>
<td>RIEF</td>
<td>dense</td>
<td>12.7 s</td>
</tr>
</tbody>
</table>
Experiments

Bounded parameters: \( a \in [0, 50] \) and \( b \in [0, 50] \)

Result obtained by IEF: \( a \geq 34, b \geq 10, a - b \geq 24 \)

Result obtained by RIEF: \( a > \frac{562}{17}, b \geq 10, a - b > \frac{392}{17} \)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Result</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEF</td>
<td>discrete</td>
<td>7.4 s</td>
</tr>
<tr>
<td>RIEF</td>
<td>dense</td>
<td>12.7 s</td>
</tr>
</tbody>
</table>
Experiments

Bounded parameters: \( a \in [0, 50] \) and \( b \in [0, 50] \)

Result obtained by IEF: \( a \geq 34, b \geq 10, a - b \geq 24 \)

Result obtained by RIEF: \( a > \frac{562}{17}, b \geq 10, a - b > \frac{392}{17} \)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Result</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEF</td>
<td>discrete</td>
<td>7.4 s</td>
</tr>
<tr>
<td>RIEF</td>
<td>dense</td>
<td>12.7 s</td>
</tr>
</tbody>
</table>

😊 Slightly better result by RIEF
😊 Longer computation time (IH is expensive)
😊 Most important: RIEF is dense
Outline

1 Preliminaries

2 Previous Works on Parameter Synthesis

3 Integer-Complete Dense Synthesis

4 Implementation in ROMÉO

5 Conclusion and Perspectives
Summary

- Two synthesis algorithms for PTA with guaranteed termination and dense result
  - Dense valuations are important for robustness

- First terminating algorithms over dense valuations with guarantee on the results
Perspectives

- Exact characterization of the result of RIEF and RIAF
- What part of the result may be missing?
Perspectives

- Exact characterization of the result of RIEF and RIAF
  - What part of the result may be missing?

- Extension of this principle to further algorithms
  - Inverse method (trace or language preservation) [A., Chatain, Encrenaz, Fribourg, 2009] and implementation in IMITATOR
**Perspectives**

- Exact characterization of the result of RIEF and RIAF
  - What part of the result may be missing?

- Extension of this principle to further algorithms
  - **Inverse method** (trace or language preservation) [A., Chatain, Encrenaz, Fribourg, 2009] and implementation in IMITATOR

- Use multi-core processors
  - E.g., some cores to compute successor states, and some to check the equality of integer hulls
Bibliography
A theory of timed automata.

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

An inverse method for parametric timed automata.

Language preservation problems in parametric timed automata.

The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.


Licensing
Source of the graphics used I

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

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

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

(LATeX source available on demand)

Author: Étienne André

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