Learning-Based Compositional Parameter Synthesis for Event-Recording Automata

Étienne André\textsuperscript{1} and Shang-Wei Lin\textsuperscript{2}

\textsuperscript{1}LIPN, Université Paris 13, CNRS, France
\textsuperscript{2}SCSE, Nanyang Technological University, Singapore
**Context: Critical distributed systems**

- Need for early bug detection
  - Bugs discovered when final testing: **expensive**
  - $\leadsto$ Need for a thorough **modeling and verification** phase
Timed model checking (1/2)

A model of the system

A property to be satisfied

\[ y = \text{delay} \]

\[ x := 0 \]

\[ x < \text{period} \]

\[ \text{is unreachable} \]
Timed model checking (1/2)

A model of the system

A property to be satisfied

Question: does the model of the system satisfy the property?
Timed model checking (1/2)

A model of the system

A property to be satisfied

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

Yes

No

Counterexample
Timed model checking (2/2)

- 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”

- Powerful model checking tools, e.g.:
  - UPPAAL
  - PAT

[Larsen et al., 1997]
[Sun et al., 2009]
Beyond timed model checking: parameter synthesis

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

- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?
  - System incompletely specified: Can I verify my system even if I don’t know the period value with full certainty?
Beyond timed model checking: parameter synthesis

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

- Challenges
  - Numerous verifications: is the system correct for any value within \([40; 60]\)?
  - Optimization: until what value can we increase 10?
  - Robustness [Markey, 2011]: What happens if 50 is implemented with 49.99?
  - System incompletely specified: Can I verify my system even if I don’t know the period value with full certainty?

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

A model of the system

A property to be satisfied

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

Yes

No

Counterexample
**Parametric timed model checking**

A model of the system

A property to be satisfied

- **Question:** for what values of the parameters does the model of the system satisfy the property?
  
  **Yes if...**
  
  $2^{\text{delay}} > \text{period} \\
  \wedge \text{period} < 20.46$
Outline

1. Parametric Event-Recording Automata
2. Learning-based compositional synthesis
3. Experiments
4. Conclusion and perspectives
Outline

1. Parametric Event-Recording Automata
2. Learning-based compositional synthesis
3. Experiments
4. Conclusion and perspectives
Event-recording automata (ERA)

- Finite state automaton (sets of locations)
Event-recording automata (ERA)

- Finite state automaton (sets of locations and actions)

![Diagram of an event-recording automaton]

- coffee!
- press?
- start!
- cup!
- press?
Event-recording automata (ERA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks, one clock for each action [Alur et al., 1999]
  - Clocks: Real-valued variables evolving linearly at the same rate
  - A (strict) subclass of timed automata [Alur and Dill, 1994]

Diagram:

- Press
- Start
- Coffee
- Cup
- Press
Event-recording automata (ERA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks, one clock for each action [Alur et al., 1999]
  - Clocks: Real-valued variables evolving linearly at the same rate
  - A (strict) subclass of timed automata [Alur and Dill, 1994]

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

![Diagram of an event-recording automaton](image)
Event-recording automata (ERA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks, one clock for each action [Alur et al., 1999]
  - Clocks: Real-valued variables evolving linearly at the same rate
  - A (strict) subclass of timed automata [Alur and Dill, 1994]

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

\[
\begin{align*}
x_{\text{start}} &= 8 \\
\text{coffee!}
\end{align*}
\]
Event-recording automata (ERA)

- Finite state automaton (sets of locations and actions) augmented with a set $X$ of clocks, one clock for each action [Alur et al., 1999]
  - Clocks: Real-valued variables evolving linearly at the same rate
  - A (strict) subclass of timed automata [Alur and Dill, 1994]

Features

- Location invariant: constraint to be verified to stay at a location
- Transition guard: constraint to be verified to enable a transition
- For each action $a$, the corresponding clock $x_a$ is (implicitly) reset

\[ x_{\text{start}} = 8 \]

Coffee!

Diagram:

- Press? (green)
  - $x_{\text{press}} = 0$
  - $x_{\text{start}} = 5$
- Start! (blue)
- Cup! (red)
  - $x_{\text{start}} \leq 8$
  - $x_{\text{press}} \geq 1$
Concrete semantics of event-recording automata

- **Concrete state** of an ERA: 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 time elapse
Examples of concrete runs

Possible concrete runs for the coffee machine
Examples of concrete runs

- Possible concrete runs for the coffee machine

  - Coffee with no sugar

\[ x_{\text{press}} = 0 \\
\]

\[ x_{\text{start}} = 0 \]
Examples of concrete runs

- Possible concrete runs for the coffee machine

- Coffee with no sugar

\[
\begin{array}{c|c|c}
\chi_{\text{press}} & 0 & 15.4 \\
\chi_{\text{start}} & 0 & 15.4 \\
\end{array}
\]
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

<table>
<thead>
<tr>
<th>x_{press}</th>
<th>press</th>
<th>x_{start}</th>
<th>start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>15.4</td>
</tr>
<tr>
<td>0</td>
<td>15.4</td>
<td>15.4</td>
<td>15.4</td>
</tr>
</tbody>
</table>
Examples of concrete runs

Possible concrete runs for the coffee machine

Coffee with no sugar

\[
\begin{array}{cccc}
\chi_{\text{press}} & 0 & 15.4 & 0 & 0 \\
\chi_{\text{start}} & 0 & 15.4 & 15.4 & 0 \\
\end{array}
\]
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

<table>
<thead>
<tr>
<th>( x_{\text{press}} )</th>
<th>( x_{\text{start}} )</th>
<th>( x_{\text{start}} )</th>
<th>( x_{\text{press}} )</th>
<th>( x_{\text{start}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15.4</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>15.4</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

| \( x_{\text{press}} \) | 0 | 15.4 | 0 | 0 | 5 | 5 |
| \( x_{\text{start}} \) | 0 | 15.4 | 15.4 | 0 | 5 | 5 |
Examples of concrete runs

Possible concrete runs for the coffee machine

Coffee with no sugar

<table>
<thead>
<tr>
<th>(x_{\text{press}})</th>
<th>0</th>
<th>15.4</th>
<th>0</th>
<th>0</th>
<th>5</th>
<th>5</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_{\text{start}})</td>
<td>0</td>
<td>15.4</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
Examples of concrete runs

- Possible concrete runs for the coffee machine

- **Coffee with no sugar**

<table>
<thead>
<tr>
<th>( \chi_{\text{press}} )</th>
<th>0</th>
<th>15.4</th>
<th>0</th>
<th>0</th>
<th>5</th>
<th>5</th>
<th>8</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \chi_{\text{start}} )</td>
<td>0</td>
<td>15.4</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
    - 15.4
    - 0
    - 0
    - 0
    - 5
    - 5
    - 5
    - 8
    - 8
  - Coffee with 2 doses of sugar
    - 0
    - 0
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - Coffee with no sugar
    - Initial state: $x_{start} = 8$
    - Transition: $x_{start} \leq 5$
    - Transition: $x_{start} = 5$
    - Transition: $x_{start} \leq 8$
  - Initial state: $x_{press} = 0$
    - Transition: $x_{press} \geq 1$
  - States:
    - press?
    - start!
    - cup!
    - coffee!
  - States values:
    - $x_{press}$: 0, 15.4, 0, 0, 5, 5, 5, 8, 8
    - $x_{start}$: 0, 15.4, 15.4, 0, 5, 5, 8, 8

- Coffee with 2 doses of sugar
  - Initial state: $x_{press} = 0$
  - States:
    - press?
    - start!
  - States values:
    - $x_{press}$: 0, 0, 0
    - $x_{start}$: 0, 0, 0
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

Possible concrete runs for the coffee machine

- **Coffee with no sugar**

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

- Possible concrete runs for the coffee machine

- Coffee with no sugar

- Coffee with 2 doses of sugar
Examples of concrete runs

**Possible concrete runs for the coffee machine**

- **Coffee with no sugar**

  - States:
    - Press question: 15.4
    - Start: 0
    - Press: 0
    - Coffee!: 5
    - Cup!: 3
    - Coffee!: 8

  - States table:
    | x_press | x_start |
    |---------|---------|
    | 0       | 15.4    |
    | 0       | 15.4    |
    | 15.4    | 0       |
    | 0       | 5       |
    | 5       | 5       |
    | 5       | 8       |
    | 8       | 8       |

- **Coffee with 2 doses of sugar**

  - States:
    - Press question: 0
    - Start: 0
    - Press: 0
    - Press?: 0
    - Press?: 1.5
    - Press?: 2.7
    - Press?: 0

  - States table:
    | x_press | x_start |
    |---------|---------|
    | 0       | 0       |
    | 0       | 0       |
    | 0       | 1.5     |
    | 1.5     | 1.5     |
    | 2.7     | 4.2     |
    | 4.2     | 4.2     |
Examples of concrete runs

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

  - **Coffee with no sugar**
    
    ```
    \[
    \begin{array}{cccccccc}
    \text{press?} & \text{start!} & 5 & \text{cup!} & 3 & \text{coffee!} \\
    15.4 & 0 & 0 & 0 & 5 & 5 & 8 & 8 \\
    \end{array}
    \]
    
    \[
    \begin{array}{cccccccc}
    \text{press?} & \text{start!} & 1.5 & \text{press?} & 2.7 & \text{press?} & 0.8 \\
    0 & 0 & 0 & 1.5 & 0 & 2.7 & 0 & 0.8 \\
    \end{array}
    \]
    
    - **Coffee with 2 doses of sugar**
      
      ```
      
      \[
      \begin{array}{cccccccc}
      \text{press?} & \text{start!} & 1.5 & \text{press?} & 2.7 & \text{press?} & 0.8 \\
      0 & 0 & 0 & 1.5 & 0 & 2.7 & 0 & 0.8 \\
      \end{array}
      \]
Examples of concrete runs

- Possible concrete runs for the coffee machine
  - **Coffee with no sugar**
    - \( x_{\text{press}} \) values: 0, 0, 0, 1.5, 0, 2.7, 0, 0.8, 0.8
    - \( x_{\text{start}} \) values: 0, 0, 0, 1.5, 1.5, 4.2, 4.2, 5, 5
  - **Coffee with 2 doses of sugar**
    - \( x_{\text{press}} \) values: 0, 0, 0, 1.5, 0, 2.7, 0, 0.8, 0.8
    - \( x_{\text{start}} \) values: 0, 0, 0, 1.5, 1.5, 4.2, 4.2, 5, 5
Examples of concrete runs

Possible concrete runs for the coffee machine

- Coffee with no sugar

<table>
<thead>
<tr>
<th>x_press</th>
<th>x_start</th>
<th>press?</th>
<th>start!</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>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>15.4</td>
<td>15.4</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

- Coffee with 2 doses of sugar

<table>
<thead>
<tr>
<th>x_press</th>
<th>x_start</th>
<th>press?</th>
<th>start!</th>
<th>1.5</th>
<th>press?</th>
<th>2.7</th>
<th>press?</th>
<th>0.8</th>
<th>cup!</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<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>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.5</td>
<td>4.2</td>
<td>4.2</td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
Examples of concrete runs

Possible concrete runs for the coffee machine

- **Coffee with no sugar**

  \[
  \begin{array}{cccccccc}
  \chi_{\text{press}} & 15.4 & press? & \chi_{\text{start}} & 5 & \chi_{\text{cup}} & 3 & \chi_{\text{coee}} \\
  \chi_{\text{start}} & 0 & 15.4 & 0 & 0 & 5 & 8 & 8 \\
  \end{array}
  \]

- **Coffee with 2 doses of sugar**

  \[
  \begin{array}{cccccccc}
  \chi_{\text{press}} & press? & \chi_{\text{start}} & 1.5 & press? & 2.7 & press? & 0.8 \\
  \chi_{\text{start}} & 0 & 0 & 0 & 1.5 & 0 & 4.2 & 5 \\
  \end{array}
  \]
Parametric event-recording automata (PERA)

- Event-recording automata (sets of locations, actions and clocks)
Parametric event-recording automata (PERA)

- Event-recording automata (sets of locations, actions and clocks) augmented with a set $P$ of parameters
  - Parameters: Unknown constants used in guards and invariants
  - Extension in the spirit of parametric timed automata

\[ \text{Alur et al., 1993} \]
Valuation of a PERA

Given a PERA $A$ and a parameter valuation $\nu$, we denote by $\nu(A)$ the (non-parametric) event-recording automaton where all parameters are valuated by $\nu$. 
Valuation of a PERA

Given a PERA $A$ and a parameter valuation $\nu$, we denote by $\nu(A)$ the (non-parametric) event-recording automaton where all parameters are valuated by $\nu$

\[ \nu \cdot \begin{pmatrix} \text{press?} \\ \text{press?} \end{pmatrix} \]

with $\nu : \begin{cases} p_1 &\rightarrow & 1 \\ p_2 &\rightarrow & 5 \\ p_3 &\rightarrow & 8 \end{cases}$
Problems

Reachability-emptiness

Input: a PERA $A$, a location $l$
Question: is the set of valuations $v$ for which $a$ runs reaches $l$ in $v(A)$ empty?

Reachability-synthesis

Input: a PERA $A$, a location $l$
Question: synthesize all valuations $v$ for which $a$ runs reaches $l$ in $v(A)$
An undecidability result

Theorem (Undecidability)

The reachability-emptiness problem is undecidable for PERAs

Proof idea.

By encoding a 2-counter machine into a PERA (using a proof of undecidability for parametric timed automata [ICFEM 2016])

Consequence: no hope for exact synthesis
An undecidability result

Theorem (Undecidability)

The reachability-emptiness problem is undecidable for PERAs

Proof idea.

By encoding a 2-counter machine into a PERA (using a proof of undecidability for parametric timed automata [ICFEM 2016])

Consequence: no hope for exact synthesis... but

- One can design semi-algorithms
- One can design algorithms synthesizing under- or over-approximations
Outline

1. Parametric Event-Recording Automata
2. Learning-based compositional synthesis
3. Experiments
4. Conclusion and perspectives
Goal

- Algorithms for synthesis in parametric timed formalisms are very expensive
  - No efficient data structures (in contrast to BDDs or DBMs)
    - Even though parametric DBMs were proposed [Annichini et al., 2000, Hune et al., 2002]
  - Expensive operations on polyhedra (e.g., using PPL [Bagnara et al., 2008])
Goal

- Algorithms for synthesis in parametric timed formalisms are very expensive
  - No efficient data structures (in contrast to BDDs or DBMs)
    - (Even though parametric DBMs were proposed
      [Annichini et al., 2000, Hune et al., 2002])
  - Expensive operations on polyhedra (e.g., using PPL
    [Bagnara et al., 2008])

Goal

Design an efficient (semi-)algorithm for the parameter synthesis for PERAs
Goal

- Algorithms for synthesis in parametric timed formalisms are very expensive
  - No efficient data structures (in contrast to BDDs or DBMs)
    - (Even though parametric DBMs were proposed [Annichini et al., 2000, Hune et al., 2002])
  - Expensive operations on polyhedra (e.g., using PPL [Bagnara et al., 2008])

Goal

Design an efficient (semi-)algorithm for the parameter synthesis for PERAs

Idea

Use learning-based techniques to compute an abstraction of the non-parametric components, so as to speed up the verification
Assume-Guarantee Reasoning (AGR)

\[ A \parallel \tilde{B} \models \varphi \]
\[ B \models \tilde{B} \]
\[ A \parallel B \models \varphi \]

“If A with an abstraction \(\tilde{B}\) of B satisfy property \(\varphi\) then A with B satisfy \(\varphi\)”
Assumption and partitioning heuristics

- The system is modeled using a network of PERAs
  - Some components are parametric (i.e., contains parameters)
  - Some components are non-parametric
Assumption and partitioning heuristics

- The system is modeled using a network of PERAs
  - Some components are parametric (i.e., contains parameters)
  - Some components are non-parametric

- Partitioning the system into $A \parallel B$
  1. If a component has timing parameters, it is collected in group $A$;
  2. If a component shares common action labels with the property, the component is collected in group $A$.

Other components are collected in group $B$
Point-based parameter synthesis

In order to apply (non-parametric) AGR abstractions, we use a point-based method

- Iterate on points (parameter valuations) in a bounded parameter domain
  - e.g., integer points
- For each not point by a parameter constraint, generalize the point using the algorithm PRP
  - PRP: Parametric reachability preservation [NFM 2015]
  - Synthesizes a dense constraint around a point that preserves the reachability of a location
    - If the location is reachable for $v$, then it is for the constraint and vice-versa
- ... until (at least) a certain set of discrete points is covered
  - e.g., all integers
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Parametric reachability preservation cartography
Learning an abstraction: \texttt{LearnAbstr}(B, \nu(A), AG\neg L^\circ)

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

\texttt{TL}^*:

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

[Lin et al., 2014]
Learning an abstraction: LearnAbstr(B, v(A), AG¬L*)

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

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

**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
Learning an abstraction: \textit{LearnAbstr}(B, \nu(A), AG\neg L\circledast)

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

\begin{tikzpicture}[node distance=1cm, auto]
  	
  	
  	
  

\textbf{TL}^*: learning algorithm to compute a candidate abstraction \tilde{B} of an ERA B
\models: can be checked using model checking

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

**Input:** \( \nu(A) \parallel B \)

**Output:** an abstraction \( \tilde{B} \) or a counter-example

\[
\begin{align*}
\text{TL}^* & \quad \downarrow \tilde{B} \\
\nu(A) \parallel \tilde{B} & \models \varphi? \\
\text{yes} & \quad \downarrow \tilde{B} ? \\
B & \models \tilde{B} ? \\
\text{yes} & \quad \text{yes} \\
\text{(i.e., } \nu(A) \parallel B & \models \varphi) \\
\text{Abstraction: } & \tilde{B}
\end{align*}
\]

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

\( \models \): can be checked using model checking

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

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

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

\[ TL^* \]
\[ \tilde{\mathcal{B}} \]
\[ \nu(\mathcal{A}) \parallel \tilde{\mathcal{B}} \models \varphi? \]
\[ \text{yes} \]
\[ \rho \text{ accepted by } \nu(\mathcal{A})? \]
\[ \text{yes} \]
\[ (i.e., \nu(\mathcal{A}) \parallel \mathcal{B} \not\models \varphi) \]
\[ \text{Counterex: trace}(\rho) \]
\[ \text{no, } \rho \]
\[ \mathcal{B} \models \tilde{\mathcal{B}}? \]
\[ \text{yes} \]
\[ (i.e., \nu(\mathcal{A}) \parallel \mathcal{B} \models \varphi) \]
\[ \text{Abstraction: } \tilde{\mathcal{B}} \]

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

\[ \models: \text{can be checked using model checking} \]
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

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

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

**Input:** \(v(A) \parallel B\)

**Output:** an abstraction \(\tilde{B}\) or a counter-example

\begin{align*}
\text{TL*: learning algorithm to compute a candidate abstraction } & \tilde{B} \text{ of an ERA } B \\
\models: \text{can be checked using model checking} \\
\text{Refinement: can be performed using learning}
\end{align*}

[Lin et al., 2014]
Learning an abstraction: \( \text{LearnAbstr}(B, \nu(A), AG \not\subseteq L^\circ) \)

**Input:** \( \nu(A) \parallel B \)

**Output:** an abstraction \( \tilde{B} \) or a counter-example

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

\[\text{Counterex: } \text{trace}(\rho)\]

\[\text{Abstraction: } \tilde{B}\]

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

\[B \models \tilde{B}?\]

\[\rho \text{ accepted by } B?\]

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

\[\models: \text{can be checked using model checking}\]

\[\text{Refinement: can be performed using learning}\]
Learning an abstraction: \textbf{LearnAbstr}(B, \nu(A), AG \neg L^{\ominus})

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

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

\|\!: can be checked using model checking
Refinement: can be performed using learning

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

\[\equiv: \text{can be checked using model checking}\]
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 (see paper)

😊 Very cheap
Our overall procedure **CompSynth**

Key ideas:

- Iterate on integer points $\nu$
- Try to compute an abstraction $\tilde{B}$ of the non-parametric component w.r.t. $\nu(A)$ 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 PRP on $A \parallel \tilde{B}$
  - If fail, synthesize the valuations corresponding to the counterex.

```
1  K_{bad} \leftarrow \bot; \quad K_{good} \leftarrow \bot
2  \textbf{while} \text{ there is an integer point not covered by } K_{bad} \text{ or } K_{good} \text{ do }
3    \text{Pick such a point } v
4    \textbf{switch} \text{ LearnAbstr}(B, v(A), AG\neg L^\circ) \text{ do }
5      \textbf{case} \text{ Abstraction}(\tilde{B})
6          \hspace{1em} K_{good} \leftarrow K_{good} \cup \text{PRP}(A \parallel \tilde{B}, v, L^\circ)
7      \textbf{case} \text{ Counterex}(\tau)
8          \hspace{1em} K_{bad} \leftarrow K_{bad} \cup \text{ReplayTrace}(A \parallel B, \tau)
9  \textbf{return} (K_{good}, K_{bad})
```
Soundness and termination

Proposition (Soundness)

Assume $\text{CompSynth}(A, B, L^\ominus)$ terminates with result $(K_{\text{good}}, K_{\text{bad}})$. Then, for all $\nu$

1. if $\nu \models K_{\text{good}}$ then $\nu(A \parallel B)$ does not reach $L^\ominus$;
2. if $\nu \models K_{\text{bad}}$ then $\nu(A \parallel B)$ reaches $L^\ominus$. 
Soundness and termination

Proposition (Soundness)

Assume \( \text{CompSynth}(A, B, L^{\ominus}) \) terminates with result \((K_{\text{good}}, K_{\text{bad}})\). Then, for all \( \nu \)

1. if \( \nu \models K_{\text{good}} \) then \( \nu(A \parallel B) \) does not reach \( L^{\ominus} \);
2. if \( \nu \models K_{\text{bad}} \) then \( \nu(A \parallel B) \) reaches \( L^{\ominus} \).

Proposition (Integer-completeness)

Assume \( \text{CompSynth}(A, B, L^{\ominus}) \) terminates with result \((K_{\text{good}}, K_{\text{bad}})\). Then any integer point in the bounded parameter domain is either in \( K_{\text{good}} \) or in \( K_{\text{bad}} \).
Outline

1. Parametric Event-Recording Automata
2. Learning-based compositional synthesis
3. Experiments
4. Conclusion and perspectives
Our toolkit

- **IMITATOR**: state-of-the-art tool for parameter synthesis for parametric timed automata  
  \[\text{[André et al., 2012]}\]

- **CV**: (new) prototype compositional verifier of the compositional verification framework for ERAs  
  \[\text{[Lin et al., 2014]}\]

- interface in Python (about 700 lines)
## Experiments

<table>
<thead>
<tr>
<th>Case study</th>
<th>#A</th>
<th>#X</th>
<th>#P</th>
<th>Spec</th>
<th>EFsynth</th>
<th>PRPC</th>
<th>CompSynth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#iter</td>
<td>total</td>
<td>#abs</td>
</tr>
<tr>
<td>FMS-1</td>
<td>6</td>
<td>18</td>
<td>2</td>
<td>1</td>
<td>0.299</td>
<td>2</td>
<td>0.654</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>0.010</td>
<td>1</td>
<td>0.372</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>0.282</td>
<td>1</td>
<td>0.309</td>
<td>1</td>
</tr>
<tr>
<td>FMS-2</td>
<td>11</td>
<td>37</td>
<td>2</td>
<td>1</td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>0.051</td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>0.062</td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td>AIP</td>
<td>11</td>
<td>46</td>
<td>2</td>
<td>1</td>
<td>0.551</td>
<td>-</td>
<td>T.O.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>2.11</td>
<td>-</td>
<td>T.O.</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>3.91</td>
<td>-</td>
<td>T.O.</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>0.235</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td>0.022</td>
<td>-</td>
<td>T.O.</td>
<td>0</td>
</tr>
<tr>
<td>Fischer-3</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td>2.76</td>
<td>4</td>
<td>14.0</td>
<td>0</td>
</tr>
<tr>
<td>Fischer-4</td>
<td>6</td>
<td>16</td>
<td>2</td>
<td>T.O.</td>
<td>-</td>
<td>T.O.</td>
<td>0</td>
</tr>
</tbody>
</table>

**EFsynth:** monolithic reachability synthesis  
**PRPC:** point-based synthesis without abstraction  

All data (sources, binaries, models, logs) are available at [https://www.imitator.fr/static/FORTE17/](https://www.imitator.fr/static/FORTE17/)
Experiments: scalability

Testing the scalability w.r.t. the size of the parameter domain

<table>
<thead>
<tr>
<th>Case study</th>
<th>#A</th>
<th>#X</th>
<th>#P</th>
<th>Spec</th>
<th>D₀</th>
<th>#abs</th>
<th>#c.-ex.</th>
<th>find next point</th>
<th>learning</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMS-2</td>
<td>11</td>
<td>37</td>
<td>2</td>
<td>1</td>
<td>2,500</td>
<td>1</td>
<td>1</td>
<td>0.0</td>
<td>81.0</td>
<td>85.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10,000</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
<td>82.5</td>
<td>87.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250,000</td>
<td>1</td>
<td>1</td>
<td>2.2</td>
<td>82.0</td>
<td>89.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1,000,000</td>
<td>1</td>
<td>1</td>
<td>8.9</td>
<td>83.1</td>
<td>96.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25,000,000</td>
<td>1</td>
<td>1</td>
<td>221.2</td>
<td>83.1</td>
<td>309.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100,000,000</td>
<td>1</td>
<td>1</td>
<td>888.1</td>
<td>83.5</td>
<td>976.4</td>
</tr>
</tbody>
</table>

Gets slower from 10⁶ points

- Reason: use exhaustive enumeration
Experiments: scalability

Testing the scalability w.r.t. the size of the parameter domain

<table>
<thead>
<tr>
<th>Case study</th>
<th>#A</th>
<th>#X</th>
<th>#P</th>
<th>Spec</th>
<th>D0</th>
<th>CompSynth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#abs</td>
</tr>
<tr>
<td>FMS-2</td>
<td>11</td>
<td>37</td>
<td>2</td>
<td>1</td>
<td>2,500</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1,000,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25,000,000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100,000,000</td>
<td>1</td>
</tr>
</tbody>
</table>

Gets slower from $10^6$ points

- Reason: use exhaustive enumeration

Future work: use, e.g., an SMT solver
Outline

1 Parametric Event-Recording Automata
2 Learning-based compositional synthesis
3 Experiments
4 Conclusion and perspectives
Summary

- New method for parameter synthesis for distributed systems, modeled by PERAs
  - PERAs: Strong assumption! Event-recording automata are needed to perform language inclusion in $\mathcal{TL}^*$ (undecidable for timed automata)

- Despite undecidability in general, proposed an efficient algorithm
  - Experiments show a dramatic improvement for loosely-synchronized PERAs
Perspectives

- Reuse the distributed point-based synthesis on a cluster
  [A., Coti, Nguyen, ICFEM 2015]

- Beyond PERAs: what can we reuse to perform compositional verification of parametric timed automata?

- Combine our approach with that of [Aștefănoaei et al., 2016]
  - Invariant-based compositional synthesis
Bibliography
References I

A theory of timed automata.

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

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

Enhanced distributed behavioral cartography of parametric timed automata.
In *ICFEM*, Lecture Notes in Computer Science. Springer.

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

Decision problems for parametric timed automata.
In ICFEM, volume 10009 of Lecture Notes in Computer Science, pages 400–416. Springer.

Reachability preservation based parameter synthesis for timed automata.
In NFM, volume 9058 of Lecture Notes in Computer Science, pages 50–65. Springer.

Symbolic techniques for parametric reasoning about counter and clock systems.
In CAV’00, pages 419–434. Springer-Verlag.

Compositional parameter synthesis.
In FM, volume 9995 of Lecture Notes in Computer Science, pages 60–68.

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


Additional explanation
Explanation for the 4 pictures in the beginning

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

- Error screen on the earliest versions of Macintosh

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

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

Test various partitioning heuristics

<table>
<thead>
<tr>
<th>Case study</th>
<th>#A</th>
<th>#X</th>
<th>#P</th>
<th>Spec</th>
<th>Partitioning A</th>
<th>CompSynth</th>
<th>Batch Size</th>
<th>Learning Time</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>#abs</td>
<td>#c.-ex.</td>
<td>learning</td>
<td>total</td>
<td></td>
</tr>
<tr>
<td>FMS-1</td>
<td>6</td>
<td>18</td>
<td>2</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1.071</td>
<td>1.137</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CM</td>
<td>R1R2A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CMR1</td>
<td>R2A</td>
<td></td>
<td>0.077</td>
<td>0.148</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CMR2</td>
<td>R1A</td>
<td></td>
<td>5.152</td>
<td>5.406</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CMA</td>
<td>R1R2</td>
<td></td>
<td>5.663</td>
<td>5.980</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CMR1R2</td>
<td>A</td>
<td></td>
<td>0.123</td>
<td>0.290</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CMR1A</td>
<td>R2</td>
<td></td>
<td>0.119</td>
<td>0.360</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>CMR2A</td>
<td>R1</td>
<td></td>
<td>6.150</td>
<td>6.690</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CM</td>
<td>R1R2A</td>
<td>0</td>
<td>0.133</td>
<td>0.149</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CMR1</td>
<td>R2A</td>
<td>0</td>
<td>0.077</td>
<td>0.123</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CMR2</td>
<td>R1A</td>
<td>0</td>
<td>0.040</td>
<td>0.056</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CMA</td>
<td>R1R2</td>
<td>0</td>
<td>0.824</td>
<td>0.842</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CMR1R2</td>
<td>A</td>
<td>0</td>
<td>0.034</td>
<td>0.044</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CMR1A</td>
<td>R2</td>
<td>0</td>
<td>0.096</td>
<td>0.144</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>CMR2A</td>
<td>R1</td>
<td>0</td>
<td>0.042</td>
<td>0.051</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CM</td>
<td>R1R2A</td>
<td>1</td>
<td>0.211</td>
<td>0.270</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CMR1</td>
<td>R2A</td>
<td>1</td>
<td>0.082</td>
<td>0.186</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CMR2</td>
<td>R1A</td>
<td>1</td>
<td>1.094</td>
<td>1.208</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CMA</td>
<td>R1R2</td>
<td>1</td>
<td>0.729</td>
<td>0.881</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CMR1R2</td>
<td>A</td>
<td>1</td>
<td>0.119</td>
<td>0.279</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CMR1A</td>
<td>R2</td>
<td>1</td>
<td>0.314</td>
<td>0.634</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>CMR2A</td>
<td>R1</td>
<td>1</td>
<td>0.104</td>
<td>0.257</td>
<td></td>
</tr>
</tbody>
</table>
Licensing
Source of the graphics used I

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

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

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

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

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

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