Dynamic Clock Elimination in Parametric Timed Automata

Étienne André

Laboratoire d’Informatique de Paris Nord
Université Paris 13, Sorbonne Paris Cité
Context: Verifying Complex Timed Systems (1/2)

- Need for early bug detection
  - Bugs discovered when final testing: expensive
  - Need for thorough modeling and verification
Context: Verifying Complex Timed Systems (2/2)

- Use formal methods

A finite model of the system

A formula to be satisfied
Context: Verifying Complex Timed Systems (2/2)

- Use formal methods

A finite model of the system

A formula to be satisfied

Question: does the model of the system satisfy the formula?
Context: Verifying Complex Timed Systems (2/2)

- Use formal methods

\[ \Delta \text{AG} \]

A finite model of the system

A formula to be satisfied

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

Yes

No

Counterexample
Context: 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”
  - etc.

- Verification for one set of constants does not 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: What happens if 50 is implemented with 49.99?
Context: 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”
  - etc.

- Verification for one set of constants does not 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: 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. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion
Outline

1 Parametric Timed Automata
2 Motivation: Clock Reduction
3 Dynamic Elimination
4 Experimental Validation
5 Conclusion
Timed Automaton

- Finite state automaton (sets of locations)
Timed Automaton

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

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

---

**Diagram**

- Locations and actions with transitions labeled `a`, `b`, and `c`.
Timed Automaton

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

**Features**

- Location *invariant*: property to be verified to stay at a location
Timed Automaton

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

Features

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

\[
x \leq 3 \quad a \quad x \geq 1 \\
\]

\[
\begin{align*}
  & b \\
  & y \geq 2.5 \\
  & c
\end{align*}
\]
Parametric Timed Automata

Timed Automaton

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

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
Parametric Timed Automaton (PTA)

- Finite state automaton (sets of locations and actions) augmented with
  - A set $X$ of clocks (i.e., real-valued variables evolving linearly at the same rate [Alur and Dill, 1994])
  - A set $P$ of parameters (i.e., unknown constants), used in guards and invariants [Alur et al., 1993]

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

\[
\begin{align*}
  & x \geq p_2 \\
  & x \leq p_1 \quad y := 0 \\
  & y \geq p_4 \\
  & x \leq p_3 \\
  & x := 0
\end{align*}
\]
Semantics of Parametric Timed Automata

- **State** of a PTA: couple \((q, C)\), where
  - \(q\) is a location,
  - \(C\) is a constraint (conjunction of inequalities) over \(X\) and \(P\)
Semantics of Parametric Timed Automata

- **State** of a PTA: couple \((q, C)\), where
  - \(q\) is a location,
  - \(C\) is a constraint (conjunction of inequalities) over \(X\) and \(P\)
- **Path**: alternating sequence of states and actions
Semantics of Parametric Timed Automata

- **State** of a PTA: couple \((q, C)\), where
  - \(q\) is a location,
  - \(C\) is a constraint (conjunction of inequalities) over \(X\) and \(P\)

- **Path**: alternating sequence of states and actions

- **Example**

  \[
  \begin{align*}
  x &\leq p_1 \\
  y &:= 0 \\
  x &\leq p_2 \\
  y &\geq p_4 \\
  x &\leq p_3 \\
  x &:= 0
  \end{align*}
  \]

- Possible path for this PTA
Semantics of Parametric Timed Automata

- **State** of a PTA: couple \((q, C)\), where
  - \(q\) is a location,
  - \(C\) is a constraint (conjunction of inequalities) over \(X\) and \(P\)

- **Path**: alternating sequence of states and actions

**Example**

\[
\begin{align*}
x &\geq p_2 \\
x &\leq p_1 \\
y &:= 0 \\
x &\leq p_3 \\
x &:= 0 \\
y &\geq p_4
\end{align*}
\]

- Possible path for this PTA

\[
\begin{align*}
x &= y \\
x &\leq p_1 \\
x - y &\leq p_1 \\
x - y &\geq p_2 \\
x &\leq p_3
\end{align*}
\]
Semantics of Parametric Timed Automata

- **State** of a PTA: couple \((q, C)\), where
  - \(q\) is a location,
  - \(C\) is a **constraint** (conjunction of inequalities) over \(X\) and \(P\)

- **Path**: alternating sequence of states and actions

**Example**

\[
\begin{align*}
x &\geq p_2 \\
x &\leq p_1 \\
y &:= 0 \\
x &\leq p_3 \\
x &:= 0 \\
y &\geq p_4 \\
x - y &\leq p_1 \\
x - y &\geq p_2 \\
x &\leq p_3 \\
p_1 &\geq p_2 \\
y &\geq x \\
y - x &\leq p_3
\end{align*}
\]

- Possible path for this PTA
Traces

- **Trace over a PTA: time-abstract path**
  - Finite alternating sequence of locations and actions

![Diagram](image-url)
Traces

- Trace over a PTA: time-abstract path
  - Finite alternating sequence of locations and actions

![Diagram showing a path over a PTA with locations a and b]
Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion
Reducing the number of clocks

- The fewer clocks, the more efficient model checking is
  [Bengtsson and Yi, 2003]

- Consequence: State space reduction
  - Smaller constraints (represented as arrays, matrices, etc.)
  - Less states (due to side-effect merging)

- Clock reduction native in some formalisms
  - Parametric time Petri nets [Traonouez et al., 2009]
  - Parametric stateful timed CSP [Sun et al., 2013]
Reducing the number of clocks

- The fewer clocks, the more efficient model checking is
  [Bengtsson and Yi, 2003]

- **Consequence**: State space reduction
  - Smaller constraints (represented as arrays, matrices, etc.)
  - Less states (due to side-effect merging)

- Clock reduction native in some formalisms
  - Parametric time Petri nets [Traonouez et al., 2009]
  - Parametric stateful timed CSP [Sun et al., 2013]
  - …but not in PTA
Reducing the number of clocks: Some approaches

- Clock elimination in non-parametric timed automata
  [Daws and Yovine, 1996]
  - Detection of (in)active clocks
  - Detection of clocks equal to each other
  - Relatively easy in a non-parametric setting (use of Difference Bound Matrices)

- (Tentative) elimination of the global clocks in a network of timed automata in a distributed setting [Balaguer and Chatain, 2012]

- Native elimination in other formalisms
  [Traonouez et al., 2009, Sun et al., 2013]
  - Translation from timed automata?
Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion
**Principle**

- **Inactive clocks**
  - In the value of $x_2$ is *useless*. It will not be used until its next reset when entering $\bullet$.

![Diagram showing dynamic clock elimination process](image)
**Principle**

- **Inactive clocks**
  - In the value of $x_2$ is **useless**. It will not be used until its next reset when entering.

- **Goal:** detect and eliminate inactive clocks
  - ⇒ Smaller memory
  - ⇒ Less states
  - ⇒ Better termination
Assumptions

Remark
Detecting really useless clocks would require us to know the future, hence to perform the analysis... which we want to avoid.
Assumptions

Remark
Detecting really useless clocks would require us to know the future, hence to perform the analysis... which we want to avoid

- Assumptions
  - Static a priori detection of the useless clocks
  - Local clocks only
  - Dynamic elimination during the analysis

- Consequence: possible under-approximation of the set of eliminated clocks
**Static Detection**

- **Backward marking algorithm for a clock $x$**
  - **Goal:** mark all locations where $x$ is useful
  - **Start by marking the locations where $x$ is used (invariant or outgoing guard)**
  - **Iterate in a backward manner until a reset is found**
  - **Stop when reaching fixpoint**

---

**Example for $x_1$:**

- $x_2 \leq p_2$
- $x_1 = p_1$
- $x_1 := 0$
- $x_2 := 0$
- $x_2 = p_1$
- $x_2 := 0$

Hence, $x_1$ can be eliminated when in...
Static Detection

- Backward marking algorithm for a clock $x$
  - Goal: mark all locations where $x$ is useful
  - Start by marking the locations where $x$ is used (invariant or outgoing guard)
  - Iterate in a backward manner until a reset is found
  - Stop when reaching fixpoint

Example for $x_1$:

- $x_2 \leq p_2$
- $x_1 \leq p_2$
- $x_1 = p_1$
- $x_1 := 0$
- $x_2 := 0$
- $x_2 = p_1$
- $x_2 := 0$

Hence, $x_1$ can be eliminated when in...
Static Detection

- **Backward marking algorithm for a clock** $x$
  - **Goal:** mark all locations where $x$ is useful
  - **Start by marking the locations** where $x$ is used (invariant or outgoing guard)
  - **Iterate in a backward manner** until a reset is found
  - **Stop when reaching fixpoint**

Example for $x_1$:

- $x_2 \leq p_2$
- $x_1 = p_1$
- $x_1 := 0$
- $x_2 := 0$
- $x_2 = p_1$
- $x_2 := 0$
Static Detection

- Backward marking algorithm for a clock $x$
  - Goal: mark all locations where $x$ is useful
  - Start by marking the locations where $x$ is used (invariant or outgoing guard)
  - Iterate in a backward manner until a reset is found
  - Stop when reaching fixpoint

Example for $x_1$:

```
x_2 \leq p_2

x_1 = p_1
x_1 := 0
x_2 := 0

x_2 = p_1
x_2 := 0
```

Hence, $x_1$ can be eliminated when in ·
Static Detection

- Backward marking algorithm for a clock $x$
  - Goal: mark all locations where $x$ is useful
  - Start by marking the locations where $x$ is used (invariant or outgoing guard)
  - Iterate in a backward manner until a reset is found
  - Stop when reaching fixpoint

Example for $x_1$:

- $x_2 \leq p_2$
- $x_1 \leq p_2$
- $x_1 = p_1$
- $x_1 := 0$
- $x_2 := 0$
- $x_2 = p_1$
- $x_2 := 0$

Hence, $x_1$ can be eliminated when in $\bullet$. 
Dynamic Elimination

- Clocks can be eliminated on-the-fly when computing a new state
  - Refer to the static table of the useless clocks in the current location

- Elimination à la Fourier-Motzkin [Schrijver, 1986]
  - So as not to modify the relationship between other clocks and parameters
    - Costly operation

Example: $x_1 \leq x_2 \leq p_2$ becomes $x_1 \leq p_2$ after elimination of $x_2$
Characterization

- Bijection between the sets of traces without and with elimination of the clocks
  - All linear-time properties (LTL) can be checked using this optimization
  - The inverse method can be applied [André and Soulat, 2013]

- Bijection between the sets of parametric paths without and with elimination of the clocks
  - Optimization suitable to perform parametric model checking
Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion
**IMITAT0R**

- **IMITATOR 2.6** [André et al., 2012]
  - “Inverse Method for Inferring Time Abstract Behavior”
  - 10,000 lines of OCaml code
  - Makes use of the PPL library [Bagnara et al., 2008]
  - Available under the GNU-GPL license
  - Now integrated in the CosyVerif platform [André et al., 2013]

- Experimental validation using the inverse method
  [André and Soulat, 2013]
  - Generalizes a reference parameter valuation by synthesizing a constraint
**IMITATOR**

- **IMITATOR 2.6** [André et al., 2012]
  - “Inverse Method for Inferring Time Abstract Behavior”
  - 10,000 lines of OCaml code
  - Makes use of the PPL library [Bagnara et al., 2008]
  - Available under the GNU-GPL license
  - Now integrated in the CosyVerif platform [André et al., 2013]

- Experimental validation using the inverse method
  [André and Soulat, 2013]
  - Generalizes a reference parameter valuation by synthesizing a constraint
## Experiments

<table>
<thead>
<tr>
<th>Example</th>
<th>X</th>
<th>P</th>
<th>S</th>
<th>T</th>
<th>t</th>
<th>IM</th>
<th>IM_{dyn}</th>
<th>t</th>
<th>Comparison</th>
<th>S</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>loop</td>
<td>2</td>
<td>2</td>
<td>0.007</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Figure 3</td>
<td>2</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>loop</td>
<td>6</td>
<td>8</td>
<td>0.006</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>AndOr</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>0.047</td>
<td>11</td>
<td>11</td>
<td>0.050</td>
<td>100</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>SPSMALL</td>
<td>10</td>
<td>26</td>
<td>31</td>
<td>30</td>
<td>0.580</td>
<td>31</td>
<td>30</td>
<td>0.584</td>
<td>100</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>Train</td>
<td>3</td>
<td>6</td>
<td>78</td>
<td>94</td>
<td>0.100</td>
<td>61</td>
<td>76</td>
<td>0.072</td>
<td>78</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>BRP</td>
<td>7</td>
<td>6</td>
<td>429</td>
<td>474</td>
<td>3.50</td>
<td>429</td>
<td>474</td>
<td>3.21</td>
<td>100</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>CSMA/CD₆</td>
<td>3</td>
<td>3</td>
<td>13,365</td>
<td>14,271</td>
<td>19.6</td>
<td>13,365</td>
<td>14,271</td>
<td>19.5</td>
<td>100</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>RCP</td>
<td>5</td>
<td>6</td>
<td>327</td>
<td>518</td>
<td>0.68</td>
<td>181</td>
<td>282</td>
<td>0.41</td>
<td>55</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>AAM06</td>
<td>3</td>
<td>8</td>
<td>1,497</td>
<td>1,844</td>
<td>8.28</td>
<td>768</td>
<td>997</td>
<td>2.92</td>
<td>51</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>AM02</td>
<td>3</td>
<td>4</td>
<td>182</td>
<td>215</td>
<td>0.392</td>
<td>182</td>
<td>215</td>
<td>0.386</td>
<td>100</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>BB04</td>
<td>6</td>
<td>7</td>
<td>806</td>
<td>827</td>
<td>25.4</td>
<td>806</td>
<td>827</td>
<td>27.2</td>
<td>100</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>CTC</td>
<td>15</td>
<td>21</td>
<td>1,364</td>
<td>1,363</td>
<td>83.4</td>
<td>201</td>
<td>291</td>
<td>2.52</td>
<td>15</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>LA02</td>
<td>3</td>
<td>5</td>
<td>6,290</td>
<td>8,023</td>
<td>710</td>
<td>4,932</td>
<td>7,154</td>
<td>473</td>
<td>78</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>LPPRC10</td>
<td>4</td>
<td>7</td>
<td>78</td>
<td>102</td>
<td>0.375</td>
<td>78</td>
<td>102</td>
<td>0.395</td>
<td>100</td>
<td>105</td>
<td></td>
</tr>
</tbody>
</table>

Sources: [http://www.lsv.ens-cachan.fr/Software/imitator/dynamic/](http://www.lsv.ens-cachan.fr/Software/imitator/dynamic/)
Outline

1. Parametric Timed Automata
2. Motivation: Clock Reduction
3. Dynamic Elimination
4. Experimental Validation
5. Conclusion
Conclusion

- Extension of dynamic clock elimination to parametric automata
- Preserves linear-time parametric model checking
- Often leads to state space reduction and memory reduction
- Surprisingly little noticeable overhead, even when the number of clocks remains constant
  ⇒ Optimization could be added as default in IMITATOR
Perspectives

- Integration of further state space reduction techniques
  [André et al., 2013]

- Improvement of the internal representation of constraints
  - Relying on the Parma Polyhedra Library [Bagnara et al., 2008]
  - Future work: remove dimensions when eliminating clocks

- Extension to the multi-core setting [Laarman et al., 2013]


References II

*The Inverse Method.*
FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc.

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

Avoiding shared clocks in networks of timed automata.

Timed automata: Semantics, algorithms and tools.

Reducing the number of clock variables of timed automata.
Multi-core emptiness checking of timed Buchi automata using inclusion abstraction.
In CAV’13, volume 8044 of Lecture Notes in Computer Science. Springer.

Theory of linear and integer programming.
John Wiley & Sons, Inc.

Modeling and verifying hierarchical real-time systems using Stateful Timed CSP.

Parametric model-checking of stopwatch Petri nets.
Additional explanations
Explanations 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 any plane crash
(Picture actually from the happy-ending US Airways Flight 1549, 2009)

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)
Source of the pictures used (1/2)

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: Miracle on the Hudson
Author: Janis Krum (cropped by Étienne André)
Source: https://secure.flickr.com/photos/davidwatts1978/3199405401/
License: CC BY 2.0

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 pictures used (2/2)

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 3.0 Unported (CC BY-SA 3.0)

Author: Étienne André

https://creativecommons.org/licenses/by-sa/3.0/