Online Parametric Timed Pattern Matching with Automata-Based Skipping

Masaki Waga$^{1,2,3}$ and Étienne André$^{4,5,1}$

National Institute of Informatics$^1$, SOKENDAI$^2$, JSPS Research Fellow$^3$, LIPN, Université Paris 13, CNRS$^4$, JFLI, UMI CNRS$^5$

7 May 2019, NFM 2019

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), by JSPS Grants-in-Aid No. 15KT0012 & 18J22498 and by the ANR national research program PACS (ANR-14-CE28-0002).
Online Parametric Timed Pattern Matching with Automata-Based Skipping

Masaki Waga$^{1,2,3}$ and Étienne André$^{4,5,1}$

National Institute of Informatics$^1$, SOKENDAI$^2$, JSPS Research Fellow$^3$, LIPN, Université Paris 13, CNRS$^4$, JFLI, UMI CNRS$^5$

7 May 2019, NFM 2019

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), by JSPS Grants-in-Aid No. 15KT0012 & 18J22498 and by the ANR national research program PACS (ANR-14-CE28-0002).
Why Monitoring?

**Exhaustive formal method**
(e.g. model checking, reachability analysis)
- The system is correct/incorrect for any execution
- We need system model (white box)
- Scalability is a big issue

**Monitoring**
- The system is correct/incorrect for the given execution
- We do not need system model (black box is OK)
- Usually scalable
(Non-Parametric) Timed Pattern Matching

[Ulus+, FORMATS’14, Waga+, FORMATS’16]

Input

• **Time-series data**
  - System *log*
    - e.g., change of engine rotation ($\omega$) / velocity ($v$) of a car
      $v\uparrow$ at 0.1s, $\omega\downarrow$ at 0.2s, …
  
• **Real-time spec.**
  - *Spec.* useful for debugging
    - e.g., unexpected behavior of a car
      $\omega$ gets high and remains $\Rightarrow$ $v$ gets high $>2$ s. later

Output

• The intervals where the *spec.* is satisfied in the *log*
  - e.g., The above behavior occurs in 0.8s-3.4s
Parametric Timed Pattern Matching

[André, Hasuo, & Waga, ICECCS’18]

Input

• Time-series data
  • System log
    • e.g., change of engine rotation ($\omega$) / velocity ($v$) of a car
      $v$↑ at 0.1s, $\omega$↓ at 0.2s, ...

• Parametric Real-time spec.
  • Spec. useful for debugging (with parameters)
    • e.g., unexpected behavior of a car (with parameters)
      $\omega$ gets high and remains $\Rightarrow v$ gets high $p$ s. later

Output

• The intervals + param. valuation, s.t. the spec. is satisfied in the log
  • e.g., The above behavior occurs in 0.8s-3.4s, $p = 2.5$
Parametric Timed Pattern Matching

[André, Hasuo, & Waga, ICECCS’18]

Input

• **Time-series data**
  - System *log*
    - e.g., change of engine rotation ($\omega$) / velocity ($v$) of a car
      $v \uparrow$ at 0.1s, $\omega \downarrow$ at 0.2s, …

• **Parametric** Real-time spec.
  - *Spec.* useful for debugging (with *parameters*)
    - e.g., unexpected behavior of a car (with *parameters*)
      $\omega$ gets high and remains $\Rightarrow v$ gets high $p$ s. later

Output

• The intervals *param. valuation*, s.t. the *spec.* is satisfied in the *log*
  - e.g., The above behavior occurs in 0.8s-3.4s, $p = 2.5$

\[ p > 2 : \text{satisfied (unexpected beh.)} \]
\[ p \leq 2 : \text{violated (expected beh.)} \]
PTPM for Periods Detection

**Imaginary Example**: Electrocardiography (Atrial/Ventricular Spikes)

**Input**

<table>
<thead>
<tr>
<th></th>
<th>AS</th>
<th>VS</th>
<th>AS</th>
<th>VS</th>
<th>AS</th>
<th>VS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td></td>
<td>0.3</td>
<td>0.7</td>
<td>0.8</td>
<td>1.4</td>
<td>1.6</td>
</tr>
</tbody>
</table>

**Output**

\[
\text{Match}(w, A) = \{(t, t', v) \mid t \in [0, 0.1), t' \in (1.4, 1.6], v(p_1) > 0.6, v(p_2) > 0.2, \ldots \} \\
\cup \ldots \cup \{(t, t', v) \mid t \in [1.6, 8.5), t' \in (10.5, \infty), v(p_1) > 1.5, v(p_2) > 0.5\}
\]
Contribution

• Give a specialized alg. for parametric timed pattern matching
  
  • [André, Hasuo, & Waga, ICECCS’18] IMITATOR-based
  
  • Optimized the algorithm with skipping from string matching (FJS)
  
• Implementation + experiment
  
  • we have 3 new Alg. + IMITATOR-based: naive, parametric/non-parametric skipping

• Our algorithms are much faster than IMITATOR-based algorithm
Outline

• Motivation + Introduction

• Technical Part
  • The parametric timed pattern matching problem
  • Naive algorithm for Parametric TPM [Alg. 1]
  • Skipping optimization for Parametric TPM
    • Parametric Skipping Algorithm [Alg. 2]
    • Non-Parametric Skipping Algorithm [Alg. 3]
  • Experiment
Parametric Timed Pattern Matching

**Input**
- Timed word $w \in (\Sigma \times \mathbb{R}_{>0})^*$
  - System log
- PTA $\mathcal{A}$
  - Parameterized spec.

**Output**
- $\text{Match}(w, \mathcal{A}) = \{(t,t',v) \mid w|_{(t,t')} \in \mathcal{L}(v(\mathcal{A}))\}$
- Interval $(t,t') + \text{param. val. } v$ s.t. spec. is satisfied

**Example**
- $w = \omega\uparrow \omega\downarrow v\downarrow \omega\uparrow v\uparrow$
- $\mathcal{A} = \omega\uparrow / x := 0 \xrightarrow{\omega\uparrow} v\uparrow, x = p \xrightarrow{v\uparrow} v\uparrow$
- $\mathcal{A} = \{(t,t',v) \mid 0.7 \leq t < 1.0, \quad 3.5 < t', v(p) = 2.5\}$

[André, Hasuo, & Waga, ICECCS’18]

$p$: Duration between $\omega\uparrow$ and $v\uparrow$
Outline

• Motivation + Introduction

• Technical Part
  • The parametric timed pattern matching problem
  • Naive algorithm for Parametric TPM [Alg. 1]
  • Skipping optimization for Parametric TPM
    • Parametric Skipping Algorithm [Alg. 2]
    • Non-Parametric Skipping Algorithm [Alg. 3]

• Experiment
Idea of our (naive) online algorithm

follow the transitions of PTA
+ abstraction of clock/param. val. by convex polyhedra
Our online (naive) algorithm

0.7 ≤ t < 1.0, p ∈ ℝ,
    x = 0

\[
\begin{align*}
\omega^\uparrow/x & := 0 \\
v^\uparrow, x &= p \\
v^\uparrow & \\
\checkmark
\end{align*}
\]
Our online (naive) algorithm

0.7 ≤ t < 1.0, p ∈ ℝ, 
    x = 0

0.7 ≤ t < 1.0, p ∈ ℝ, 
    x = 0

start → $ω_↑/x := 0$ → $ω_↑$, $x = p$ → $v_↑$ → $\checkmark$

0.7 ≤ t < 1.0, p ∈ ℝ, 
    x = 0

$t$ 0 0.1 0.4 0.7 1.0 3.5$
Our online (naive) algorithm

\[ 0.7 \leq t < 1.0, \; p \in \mathbb{R}, \]
\[ x = 0 \]

\[ 0.7 \leq t < 1.0, \; p \in \mathbb{R}, \]
\[ x = 0 \]

\[ 0.7 \leq t < 1.0, \]
\[ p = 2.5, \; x = 2.5 \]
Our online (naive) algorithm

0.7 ≤ t < 1.0, \( p \in \mathbb{R} \),
\( x = 0 \)

\[
\begin{align*}
\omega_{\uparrow} / x & := 0 \\
v_{\uparrow}, x & = p
\end{align*}
\]

0.7 ≤ t < 1.0, \( p \in \mathbb{R} \),
\( x = 0 \)

\[
0.7 \leq t < 1.0, \quad p = 2.5, \quad x = 2.5
\]

0.7 ≤ t < 1.0,
3.5 ≤ t′
\( p = 2.5 \)

\[\text{FOUND!!}\]

\begin{align*}
0.7 \leq t < 1.0, \\
3.5 \leq t'
\end{align*}

\[
\begin{align*}
\omega_{\uparrow} & \quad \omega_{\downarrow} & \quad v_{\downarrow} & \quad \omega_{\uparrow} \\
0 & \quad 0.1 & \quad 0.4 & \quad 0.7 & \quad 1.0 & \quad 3.5 & \quad t
\end{align*}
\]

\( \text{M. Waga (NII)} \)
Our online (naive) algorithm

\[
0 \leq t < 0.1, \quad p \in \mathbb{R}, \quad x = 0
\]

\[
0 \leq t < 0.1, \quad p \in \mathbb{R}, \quad x = 0
\]
Our online (naive) algorithm

\[ \omega_{\uparrow}/x := 0, \quad v_{\uparrow}, x = p, \quad v_{\uparrow} \]

0.1 \leq t < 0.4, \ p \in \mathbb{R}, \ x = 0

\[ 0.1 \leq t < 0.4, \ p \in \mathbb{R}, \ x = 0 \]

M. Waga (NII)
Our online (naive) algorithm

0.4 ≤ t < 0.7, p ∈ ℝ, x = 0
Our online (naive) algorithm

\[
0.7 \leq t < 1.0, \ p \in \mathbb{R}, \ x = 0
\]

\[
0.7 \leq t < 1.0, \ p \in \mathbb{R}, \ x = 0
\]

\[
0.7 \leq t < 1.0, \ p = 2.5, \ x = 2.5
\]

FOUND!!

\[
0.7 \leq t < 1.0, \ 3.5 < t', \ p = 2.5
\]
Outline

• Motivation + Introduction

• Technical Part
  • The parametric timed pattern matching problem
  • Naive algorithm for Parametric TPM [Alg. 1]
  • Skipping optimization for Parametric TPM
    • Parametric Skipping Algorithm [Alg. 2]
    • Non-Parametric Skipping Algorithm [Alg. 3]

• Experiment
Motivation: Skip Unnecessary Trials

Can we skip some of them? ⇒ (Usually) Yes!! by skipping from string matching
Idea of Skipping from String Matching

• For each length $n \in \mathbb{N}$, check if
  
  Over-approx. of the read timed word $\cap n$-shift + accepted timed words $= \emptyset$

  before the matching trials.

• Empty $\Rightarrow$ no need to try $n$-shift

• Over-approx. using location $l \in L$ (finite)
  
  • In PTA, we also use param. val. $v: \mathbb{P} \rightarrow \mathbb{R}$
  
  • Infinite but represented by convex polyhedra
Skipping from Non-Parametric to Parametric

** Skipping for TA** [Waga+, FORMATS’17]

For each \( l \in L, n \in \mathbb{N} \), if

\[
\mathcal{L}(A_l) \cdot T(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(A) = \emptyset
\]

Over-approx. of the read timed word

n-shift + accepted timed words

---

(Parametric) Skipping for PTA [Contribution, Alg. 2]

For each \( l \in L, v: \mathbb{P} \rightarrow \mathbb{R}, n \in \mathbb{N} \), if there is \( v': \mathbb{P} \rightarrow \mathbb{R} \) s.t.

\[
\mathcal{L}(v(A_l)) \cdot T(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(A)) = \emptyset
\]

Over-approx. of the read timed word for the given param. val. \( v \)

n-shift + accepted timed words for some param. val. \( v' \)
Skipping from Non-Parametric to Parametric

Skipping for TA [Waga+, FORMATS’17]
For each $l \in L$, $n \in \mathbb{N}$, if
\[ \mathcal{L}(A_l) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(A) = \emptyset \]

Over-approx. of the read timed word

n-shift + accepted timed words

(Parametric) Skipping for PTA [Contribution, Alg. 2]
For each $l \in L$, $v: \mathbb{P} \to \mathbb{R}$ $n \in \mathbb{N}$, if there is $v': \mathbb{P} \to \mathbb{R}$ s.t.
\[ \mathcal{L}(v(A_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(A)) = \emptyset \]

Over-approx. of the read timed word for the given param. val. $v$

n-shift + accepted timed words for some param. val. $v'$

Runtime Overhead!!
Skipping with Less Overhead

(Parametric) Skipping for PTA [Contribution, Alg. 2]

For each $l \in L$, $v : \mathbb{P} \rightarrow \mathbb{R}$ $n \in \mathbb{N}$, if there is $v' : \mathbb{P} \rightarrow \mathbb{R}$ s.t.

$\mathcal{L}(v(A_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(A)) = \emptyset$

Over-approx. of the read timed word for the given param. val. $v$

n-shift + accepted timed words for some param. val. $v'$

More overhead better approx.

Trade off!!

(Non-Parametric) Skipping for PTA [Contribution, Alg. 3]

For each $l \in L$, $n \in \mathbb{N}$, if there is $v, v' : \mathbb{P} \rightarrow \mathbb{R}$ s.t.

$\mathcal{L}(v(A_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(A)) = \emptyset$

Over-approx. of the read timed word for some param. val. $v$

n-shift + accepted timed words for some param. val. $v'$

Less overhead worse approx.
Outline

• Motivation + Introduction

• Technical Part
  • The parametric timed pattern matching problem
  • Naive algorithm for Parametric TPM [Alg. 1]
  • Skipping optimization for Parametric TPM
    • Parametric Skipping Algorithm [Alg. 2]
    • Non-Parametric Skipping Algorithm [Alg. 3]

• Experiment
RQ: Which is the fastest algorithm?

- Probably, IMITATOR-based is not very fast
  - it solves more general problem (model checking)
- Param. Skip vs. Non-Param. Skip
  - Better over-approx. vs. Less Overhead

Contributions

[André, Hasuo, & Waga, ICECCS'18]

Algorithms

- Naive
- Parametric Skip
- Non-Parametric Skip
- IMITATOR-based
Environment of Experiment

- Amazon EC2 c4.large instance
  - 2.9 GHz Intel Xeon E5-2666 v3, 2 vCPUs, 3.75 GiB RAM
- Ubuntu 18.04 LTS (64 bit)
- GCC 7.3.0 with optimization flag -O3
- Used 4 benchmarks
  - **Automotive**: Accel, Gear
  - **Toy**: Blowup, OnlyTiming

Performance for extreme inputs
Comparison with IMITATOR

Table 2: Execution time for Gear [s]

<table>
<thead>
<tr>
<th></th>
<th>No Skip</th>
<th>Non-Param. Skip</th>
<th>Param. Skip</th>
<th>IMITATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1467</td>
<td>0.04</td>
<td>0.05</td>
<td>0.05</td>
<td>1.781</td>
</tr>
<tr>
<td>2837</td>
<td>0.0725</td>
<td>0.0805</td>
<td>0.09</td>
<td>3.319</td>
</tr>
<tr>
<td>4595</td>
<td>0.124</td>
<td>0.13</td>
<td>0.1405</td>
<td>5.512</td>
</tr>
<tr>
<td>5839</td>
<td>0.1585</td>
<td>0.156</td>
<td>0.17</td>
<td>7.132</td>
</tr>
<tr>
<td>831</td>
<td>0.201</td>
<td>0.193</td>
<td>0.2115</td>
<td>8.909</td>
</tr>
<tr>
<td>13189</td>
<td>0.241</td>
<td>0.2315</td>
<td>0.2505</td>
<td>10.768</td>
</tr>
<tr>
<td>315</td>
<td>0.2815</td>
<td>0.269</td>
<td>0.2875</td>
<td>12.778</td>
</tr>
<tr>
<td>631</td>
<td>0.322</td>
<td>0.301</td>
<td>0.325</td>
<td>14.724</td>
</tr>
<tr>
<td>13189</td>
<td>0.3795</td>
<td>0.3245</td>
<td>0.353</td>
<td>16.453</td>
</tr>
<tr>
<td>14657</td>
<td>0.392</td>
<td>0.361</td>
<td>0.395</td>
<td>18.319</td>
</tr>
</tbody>
</table>

Table 3: Execution time for Accel [s]

<table>
<thead>
<tr>
<th></th>
<th>No Skip</th>
<th>Non-Param. Skip</th>
<th>Param. Skip</th>
<th>IMITATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2559</td>
<td>0.03</td>
<td>0.0515</td>
<td>0.06</td>
<td>2.332</td>
</tr>
<tr>
<td>4894</td>
<td>0.0605</td>
<td>0.0605</td>
<td>0.0705</td>
<td>4.663</td>
</tr>
<tr>
<td>7799</td>
<td>0.1005</td>
<td>0.071</td>
<td>0.08</td>
<td>7.532</td>
</tr>
<tr>
<td>10045</td>
<td>0.13</td>
<td>0.08</td>
<td>0.09</td>
<td>9.731</td>
</tr>
<tr>
<td>12531</td>
<td>0.161</td>
<td>0.09</td>
<td>0.1</td>
<td>12.503</td>
</tr>
<tr>
<td>15375</td>
<td>0.1985</td>
<td>0.1005</td>
<td>0.113</td>
<td>15.583</td>
</tr>
<tr>
<td>17688</td>
<td>0.2265</td>
<td>0.1095</td>
<td>0.1215</td>
<td>17.754</td>
</tr>
<tr>
<td>20299</td>
<td>0.261</td>
<td>0.1095</td>
<td>0.1325</td>
<td>21.881</td>
</tr>
<tr>
<td>22691</td>
<td>0.288</td>
<td>0.121</td>
<td>0.143</td>
<td>23.044</td>
</tr>
<tr>
<td>25137</td>
<td>0.3205</td>
<td>0.1315</td>
<td>0.159</td>
<td>25.815</td>
</tr>
</tbody>
</table>

Table 4: Execution time for Blowup [s]

<table>
<thead>
<tr>
<th></th>
<th>No Skip</th>
<th>Non-Param. Skip</th>
<th>Param. Skip</th>
<th>IMITATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>66.75</td>
<td>68.0125</td>
<td>67.9735</td>
<td>OutOfMemory</td>
</tr>
<tr>
<td>4000</td>
<td>267.795</td>
<td>271.642</td>
<td>269.084</td>
<td>OutOfMemory</td>
</tr>
<tr>
<td>6000</td>
<td>601.335</td>
<td>611.782</td>
<td>607.58</td>
<td>OutOfMemory</td>
</tr>
<tr>
<td>8000</td>
<td>1081.42</td>
<td>1081.25</td>
<td>1079</td>
<td>OutOfMemory</td>
</tr>
<tr>
<td>10000</td>
<td>1678.15</td>
<td>1688.22</td>
<td>1694.53</td>
<td>OutOfMemory</td>
</tr>
</tbody>
</table>

Table 5: Execution time for OnlyTiming [s]

<table>
<thead>
<tr>
<th></th>
<th>No Skip</th>
<th>Non-Param. Skip</th>
<th>Param. Skip</th>
<th>IMITATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.0995</td>
<td>0.1305</td>
<td>0.11</td>
<td>1.690</td>
</tr>
<tr>
<td>2000</td>
<td>0.191</td>
<td>0.23</td>
<td>0.191</td>
<td>3.518</td>
</tr>
<tr>
<td>3000</td>
<td>0.2905</td>
<td>0.3265</td>
<td>0.273</td>
<td>5.499</td>
</tr>
<tr>
<td>4000</td>
<td>0.3905</td>
<td>0.426</td>
<td>0.3525</td>
<td>7.396</td>
</tr>
<tr>
<td>5000</td>
<td>0.488</td>
<td>0.5225</td>
<td>0.4325</td>
<td>9.123</td>
</tr>
<tr>
<td>6000</td>
<td>0.588</td>
<td>0.6235</td>
<td>0.517</td>
<td>11.005</td>
</tr>
</tbody>
</table>

Fig. 4: Execution time for the benchmarks with parameters which MONAA cannot handle: Gear (above left), Accel (above right), Blowup (below left), and OnlyTiming (below right)
Comparison among ours (Accel & Gear)

- No Skip has the steepest slope ⇒ worst scalability
- Parametric Skip is slower than Non-Parametric Skip due to the overhead
Comparison among ours (Blowup & OnlyTiming)

- **Blowup**: Skipping does not help much
  - Exponential blowup vs. constant speed up by skipping
- **OnlyTiming**: No Skip has the steepest slope ⇒ worst scalability
  
  Parametric Skip is the fastest due to the better over-approx.
Conclusion

• Give a specialized alg. for param. timed pattern matching

• Optimized the algorithm by \texttt{skipping} from string matching

• Our algorithms are much faster than the state-of-the-art (IMITATOR-based algorithm)

• Param. vs. Non-Param. Skipping depends on the Autom.
Future Works

• Hybrid of Parametric/Non-Parametric Skipping
  • Maybe the best trade-off

• More expressive logic (e.g., FOL)

• Case study
  • not only automotive domain but also medical CPS or IoT (security)
Appendix
Why Autom? not TL or RE?

**Cons.**
- Difficult to write (and read?) for the end user

**Pros.**
- More straightforward online monitoring algorithm
- Optimization technique from untimed to timed
- TRE → TA, MITL → TA are possible
- TA as common platform
- In our industrial collaboration, we use TRE → TA

Easy 😊 Not easy…😢

e.g., *Skipping* in this talk