Moore-Machine Filtering for Timed and Untimed Pattern Matching

Masaki Waga$^1$ and Ichiro Hasuo$^1$

National Institute of Informatics$^1$
2 Oct. 2018, EMSOFT 2018

The authors are supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), and JSPS Grants-in-Aid No. 15KT0012 & 18J22498.
IoT / Embedded Systems
Connected via Network
IoT / Embedded Systems
Connected via Network

Embedded Systems

Network

Server

Monitoring!!
Timed Pattern Matching

[Ulus et al., FORMATS’14]

Input

• **Time-series data** (Logs of a car/ a robot)
  e.g., The gear of a car: **high** at 0.1s, **low** at 0.2s, ...

• **Real-time spec.** (Spec. useful for debugging)
  e.g., Frequent gear change of a car

Output

• The intervals where the **spec.** is satisfied in the **log**
  e.g., Frequent gear change in 0.2s-0.7s
Remote Monitoring

Sensor

Log

high 0.1
low 0.2
low 0.5

Spec.

\( h, x < 1 \)
\( 1/x := 0 \)

feedback

(0.2, 0.5)
(0.23, 0.55)
(0.31, 0.6)
Remote Monitoring

- **Huge** data size
- **Limited** network capacity
  ⇒ Reduce the size!!
Remote Monitoring

- **Huge** data size
- **Limited** network capacity
  ⇒ Reduce the size!!

- **Assumption**: Too Heavy for embedded systems

- Timed Pattern Matching

Sensor

Log

Monitor

Spec.

\[
\begin{align*}
\text{high} & : 0.1 \\
\text{low} & : 0.2 \\
\text{low} & : 0.5 \\
\end{align*}
\]

feedback

(0.2, 0.5)

(0.23, 0.55)

(0.31, 0.6)
Naive Remote Monitor

Sensor

log high 0.1
low 0.2
low 0.5

Monitor

Spec.

\[
\begin{align*}
& h, x < 1 \\
\text{log} & \quad 1/x := 0
\end{align*}
\]

feedback

log (0.2, 0.5)
(0.23, 0.55)
(0.31, 0.6)

Contribution: Filter for Remote Monitor

Sensor

removed!!

Filter

Spec.

\[
\begin{align*}
& h, x < 1 \\
\text{log} & \quad 1/x := 0
\end{align*}
\]

Same Result (soundness)

feedback

(0.2, 0.5)
(0.23, 0.55)
(0.31, 0.6)
Naive Remote Monitor

Sensor → Log → Monitor

Spec.
- high 0.1
- low 0.2
- low 0.5

Sensor Monitor Spec.
- high 0.1
- low 0.2
- low 0.5

Contribution: Filter for Remote Monitor

Sensor → Filter → Log → Monitor

Spec.
- high 0.1
- low 0.2
- low 0.5

Same Result (soundness)
- (0.2, 0.5)
- (0.23, 0.55)
- (0.31, 0.6)
Moore Machine

DFA + state-dependent output

• Efficient (linear time)

• Hardware implementable
Contribution

- Moore machine filter from NFA / TA
- Soundness (the result is the same)
- Implementation & experiment
- Data size $\approx 1/100 - 1/2$
Related Works

**Timed Pattern Matching**
- Offline Algorithm
  - [Ulus+, FORMATS’14]
- Online Algorithm
  - [Ulus+, TACAS’16]
- Optimization (skipping)
  - [Waga+, FORMATS’16]
  - [Waga+, FORMATS’17]

**Filtering for Pattern Matching**
- Multiple string
  - [Salmela+, JEA'06]
  - [Kandhan+, PVLDB'10]
- regular expression
  - (different problem)
  - [Liu+, ACNS’12]

This Work

M. Waga (NII)
Outline

• Moore Machine Filter for **Untimed** Pattern Matching
  
  • Spec.: NFA $\mathcal{A}$
  
  • Log: word ($w \in \Sigma^*$)

• Moore Machine Filter for **Timed** Pattern Matching
  
  • Spec.: TA $\mathcal{A}$
  
  • Log: timed word ($w \in (\Sigma \times \mathbb{R}_{>0})^*$)
  
• Experiments (**timed**)
(Untimed) Pattern Matching

**Input**
- Word $w \in \Sigma^*$
- NFA $\mathcal{A}$

**Output**
$$\text{Match}(w, \mathcal{A}) = \{(i,j) \mid w(i,j) \in L(\mathcal{A})\}$$

**Example**
- $w = \text{dbadcdc}$
- $L(\mathcal{A}) = \text{dc*\{ba|dc\}}$
- $\text{Match}(w, \mathcal{A}) = \{(1,3), (4,7)\}$
- $w(1,3) = \text{dba} \in L(\mathcal{A})$
- $w(4,7) = \text{dcdc} \in L(\mathcal{A})$
Idea: Filtering by Masking

Mask subwords not matching \((ab \mid cd)d^*\)

Sensor \(\rightarrow\) Filter \(\rightarrow\) Monitor

Goal

Construct a sound and efficient filter
\(\Rightarrow\) by a Moore machine

\(w = \text{acdda}\)

\(w' = \text{\textup{⊥}}\text{cdd}\text{\textup{⊥}}\)

DFA + state-dependent output

M. Waga (NII)
Overview of Filtering

Input Word

\[ a_1, a_2, a_3, a_4, a_5, \bot, \bot \]

Queue

Moore machine

Output Word

Size \( N = 2 \)

read char.

mask / pass

M. Waga (NII)
Overview of Filtering

Input Word

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad \bot \quad \bot \]

Queue

Moore machine

Output Word
Overview of Filtering

Input Word

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad \bot \quad \bot \]

Queue

Output Word

\[ \bot \]

\[ w \]

Pop the queue and write
Overview of Filtering

Input Word

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad \bot \quad \bot \]

Queue

Moore machine

\[ \bot \quad a_1 \]

\[ \times \quad \times \]

Output Word

\[ \bot \]

- Pop the queue and write
- Read and push
Overview of Filtering

Input Word

\[ w \]

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad \bot \quad \bot \]

Queue

Moore machine

\[ \bot \quad a_1 \]

\[ \times \quad \times \]

Output Word

\[ \bot \]
Overview of Filtering

Input Word

[Diagram showing a queue with input words]

Output Word

[Diagram showing a Moore machine and an empty queue]

Pop the queue and write
Overview of Filtering

Input Word

Output Word

Queue

Read and push

Pop the queue and write

M. Waga (NII)
Overview of Filtering

Input Word

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad \bot \quad \bot \]

Output Word

\[ \bot \quad \bot \quad \]

Queue

\[ \text{Moore machine} \]

\[ a_1 \quad a_2 \]

\[ \checkmark \quad \times \]
Overview of Filtering

Input Word

\[
\begin{array}{cccccc}
a_1 & a_2 & a_3 & a_4 & a_5 & \bot & \bot \\
\end{array}
\]

Output Word

\[
\begin{array}{cccccc}
\bot & \bot & a_1 & \bot & \bot & \bot & \bot \\
\end{array}
\]

Queue

\[
\begin{array}{cccc}
a_2 & a_3 & x & x \\
\end{array}
\]

\[w\]

✓: Write the char.
Overview of Filtering

Input Word

\[\begin{array}{cccccc}
a_1 & a_2 & a_3 & a_4 & a_5 & \bot & \bot \\
\end{array}\]

Output Word

\[\begin{array}{cccc}
\bot & \bot & a_1 & \\
\end{array}\]

Queue

\[\begin{array}{cccc}
a_2 & a_3 & \times & \times \\
\end{array}\]

Moore machine
Overview of Filtering

Input Word

| a₁ | a₂ | a₃ | a₄ | a₅ | ⊥ | ⊥ |

Queue

| a₃ | a₄ |

Moore machine

| ⊥ | ⊥ |

Output Word

| ⊥ | ⊥ | a₁ | ⊥ |

X: Write ⊥
Overview of Filtering

Input Word

\[
\begin{array}{cccccc}
\text{a}_1 & \text{a}_2 & \text{a}_3 & \text{a}_4 & \text{a}_5 & \bot & \bot \\
\end{array}
\]

- some chars are masked
- masked $\Rightarrow$ removable

Output Word

\[
\begin{array}{cccccc}
\bot & \bot & \text{a}_1 & \bot & \bot & \bot & \text{a}_4 & \text{a}_5 \\
\end{array}
\]
Untimed Filter Construction

1. add a **counter** to each state
   ← Count: distance from the init. state mod $N$

2. **powerset** construction
   ← Moore machine (deterministic)

3. add **queue**
Idea of Mask / Unmask

- Surely Unused ⇒ **Mask** (✗)
- Surely Used ⇒ **Pass** (✓)
- Not Sure ⇒ **Pass** (✓)
Properties of Untimed Moore Machine Filter

Log: $w = a_1 a_2 \ldots a_n$  Pattern: $A$  Queue size: $N$

**Thm. (soundness)**
Our filter Moore machine $\mathcal{M}$ does not change the result of the untimed pattern matching problem.

**Thm. (finite completeness)**
Our filter Moore machine $\mathcal{M}$ masks any unused char. if we have $\max\{|w| \mid w \in L(A)\} \leq N < \infty$.

**Thm. (monotonicity)**
The Moore machine filter with $N$-length queue does not mask more events than the filter with with $mN$-length queue.
Outline

Sensor → Filter → Monitor

• Moore Machine Filter for **Untimed** Pattern Matching
  • Spec.: NFA $A$
  • Log: word ($w \in \Sigma^*$)

• Moore Machine Filter for **Timed** Pattern Matching
  • Spec.: TA $A$
  • Log: timed word ($w \in (\Sigma \times \mathbb{R}_{>0})^*$)
  • Experiments (**timed**)

Main Problem

Theoretical preparation
Timed Pattern Matching

[Ulus et al., FORMATS’14]

**Input**
- Timed Word \( w \in (\Sigma \times \mathbb{R}_{>0})^* \)
- TA \( \mathcal{A} \)

**Output**
- \( \text{Match}(w, \mathcal{A}) = \{(t, t') \mid w|_{(t, t')} \in L(\mathcal{A})\} \)

**Example**

\[
\begin{align*}
|w| & = \text{low} \quad \text{high} \quad \text{low} \quad \text{high} \\
0 & \quad 0.1 \quad 0.2 \quad 0.5 \quad 0.7t \\
\mathcal{A} & = \text{start} \quad s_0 \quad \text{low} \leq \text{x} < 1 \quad s_1 \quad \text{high} \cdot \text{x} < 1 \quad s_2 \quad \text{low} \leq \text{x} < 1 \quad s_3 \quad \text{high} \cdot \text{x} < 1
\end{align*}
\]

\( \text{Match}(w, \mathcal{A}) = \{(t, t') \mid 0.2 \leq t < 0.5, 0.7 < t'\} \)
Timed Filter Construction

1. add a **counter** to each state
   ← Count: distance from the init. state mod $N$

2. **powerset** construction
   ← impossible for timed automata
   (state space will be infinite)
   ⇒ Overapproximation!!

3. add **queue**
One-Clock Determinization

[Krichen & Tripakis, FMSD’09]

Thm.
For any TA $A$, there exists a DRTA $A^{rt}$ s.t. $L(A) \subseteq L(A^{rt})$

DRTA : (deterministic real-time autom.)

- Restricted form of TA
- Guard: the duration at the source state
- Transition function $\Delta : Q \times (\Sigma \times \mathbb{R}_{\geq 0}) \to Q$
Timed Filter Construction

1. add a **counter** to each state
   ← Count: distance from the init. state mod $N$

2. **One-clock determinization**
   ← DRTA: similar to DFA

3. add **queue**
Properties of Timed Moore Machine Filter

**Thm. (soundness)**
Our filter Moore machine $\mathcal{M}$ does not change the result of the timed pattern matching problem.

*Completeness* does not hold $\Rightarrow$ experimental results
Outline

Sensor → Filter → Monitor

- Moore Machine Filter for **Untimed** Pattern Matching
  - Spec.: NFA $\mathcal{A}$
  - Log: word ($w \in \Sigma^*$)

- Moore Machine Filter for **Timed** Pattern Matching
  - Spec.: TA $\mathcal{A}$
  - Log: timed word ($w \in (\Sigma \times \mathbb{R}_{>0})^*$)

- Experiments (**timed**)

Main Problem

Theoretical preparation
Research Questions

• RQ1: Does our filter Moore machine mask many events?
  • Theoretically, only soundness
  • Queue size vs. filtering rate

• RQ2: Is our filter Moore machine online capable?
  • Linear time? Constant memory usage?
Environment of Experiment

• MacBook Pro Early 2013
  • Intel Core i5 2.6 GHz and DDR3 1600MHz 8 GB
• Mac OS 10.13.4
• clang-900.0.39.2 with optimization flag -O3
• Used 3 benchmarks (Accel, Gear, Torque)
RQ1: Filtering Rate

![Diagram showing the filtering process with different queue sizes (N = 1, 2, 3, 4, 10) and the length of the filtered timed word versus the length of the input timed word. The graph indicates that N = 10 looks enough (length: 1/100).]
RQ1: Filtering Rate (Cont’d)

Sensor → Filter → Monitor

Torque

Length of the Filtered Timed Word \([\times 10^5]\)

- \(N = 1\)
- \(N = 2\)
- \(N = 3\)
- \(N = 4\)
- \(N = 5\)
- \(N = 10\)

\(N = 2\) looks enough (length: 1/2)

Gear

Length of the Filtered Timed Word \([\times 10^5]\)

- \(N = 1\)
- \(N = 2\)
- \(N = 3\)
- \(N = 4\)
- \(N = 5\)

\(N = 2\) looks enough (length: 1/3)
RQ2: Execution Time

**Accel**

<table>
<thead>
<tr>
<th>$N$</th>
<th>Execution Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>0.24</td>
</tr>
<tr>
<td>3</td>
<td>0.36</td>
</tr>
<tr>
<td>5</td>
<td>0.70</td>
</tr>
</tbody>
</table>

**Torque**

<table>
<thead>
<tr>
<th>$N$</th>
<th>Execution Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.14</td>
</tr>
<tr>
<td>2</td>
<td>0.28</td>
</tr>
<tr>
<td>3</td>
<td>0.42</td>
</tr>
<tr>
<td>5</td>
<td>0.84</td>
</tr>
</tbody>
</table>

**Gear**

<table>
<thead>
<tr>
<th>$N$</th>
<th>Execution Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.20</td>
</tr>
<tr>
<td>2</td>
<td>0.40</td>
</tr>
<tr>
<td>3</td>
<td>0.60</td>
</tr>
<tr>
<td>5</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Linear Execution Time
RQ2: Memory Usage

**Accel**

- Constant Memory Usage

**Torque**

- Constant Memory Usage

**Gear**

- Constant Memory Usage
Conclusion

- We proposed Moore machine filter for untimed/timed pattern matching.
- Reduces the data size to send.
- Online capable.
- (Omitted in this talk) it also accelerates timed pattern matching.
Future Works

• Case study in an embedded system
• Extension to distributed systems
  • Many sensor node vs one central server
• Hardware Implementation (e.g., FPGA, ASIC)