<table>
<thead>
<tr>
<th>Supervisor</th>
<th>Reviewers</th>
<th>Examiners</th>
<th>Invited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oded Maler</td>
<td>Rajeev Alur</td>
<td>Saddek Bensalem</td>
<td>Eugene Asarin</td>
</tr>
<tr>
<td></td>
<td>Radu Grosu</td>
<td>Ahmed Bouajjani</td>
<td>Dejan Nickovic</td>
</tr>
<tr>
<td></td>
<td>Kim G. Larsen</td>
<td>Patricia Bouyer</td>
<td></td>
</tr>
</tbody>
</table>

Thesis Defense

Pattern Matching with Time — Theory and Applications

Doğan Ulus

January 15, 2018
Grenoble, France
Cyber-Physical Systems

- Systems that reside at the intersection of communication, control, and physical processes.
- Emphasis on combination and coordination between physical and computational elements.
- Examples include self-driving vehicles, robots, internet-of-things, smart grids, . . . .
Temporal Behaviors

- **Time** is crucial for physical processes and CPS.
- CPS generate **temporal behaviors**
  - Expressed in sequential forms: signals, waveforms, time series, event sequences.
  - Streamed or logged.
- Analyze temporal behaviors to understand/control/learn CPS.
Detecting Patterns in Temporal Behaviors

- Specific shapes on waveforms:
  - Rise and falls, various pulses, decays, . . .
- Specific arrangements of physical observations.
  - High speed period after high acceleration, . . .
- Sequences of actions, simultaneous occurrences, repetitions.
  - Overtaking a car.
  - Speeding-up while overtaken. (illegal)
Detecting Patterns in Temporal Behaviors

- Specific shapes on waveforms:
  - Rise and falls, various pulses, decays, . . .
- Specific arrangements of physical observations.
  - High speed period after high acceleration, . . .
- Sequences of actions, simultaneous occurrences, repetitions.
  - Overtaking a car.
  --Speeding-up while overtaken. (illegal)

- Detecting such patterns is an important problem.
Consider some text (from my manuscript) such as

"This thesis concerns patterns and how to detect them in temporal behaviors generated by various systems. These can be known patterns, good or bad patterns, correct or incorrect patterns, simple or complex patterns, but, more precisely, these are timed patterns expressed in some intuitive and well-defined formalisms, namely timed regular expressions and metric compass logic."

How to find patterns in this text?
Consider some text (from my manuscript) such as

"This thesis concerns patterns and how to detect them in temporal behaviors generated by various systems. These can be known patterns, good or bad patterns, correct or incorrect patterns, simple or complex patterns, but, more precisely, these are timed patterns expressed in some intuitive and well-defined formalisms, namely timed regular expressions and metric compass logic."

How to find patterns in this text?

Press $\text{ctrl} + F$, then type "patterns"

Find: patterns
Consider some text (from my manuscript) such as

“This thesis concerns patterns and how to detect them in temporal behaviors generated by various systems. These can be known patterns, good or bad patterns, correct or incorrect patterns, simple or complex patterns, but, more precisely, these are timed patterns expressed in some intuitive and well-defined formalisms, namely timed regular expressions and metric compass logic.”

How to find patterns in this text?
- Press `ctrl + F`, then type “patterns”
• **Regular expressions** for more complex patterns.
  - Natural numbers: 0 | [1-9] [0-9]*
  - Dates: YYYY/MM/DD
    $((19|20)?[0-9][2][-/.](0?[1-9]|1[012])[-/.](0?[1-9]|1[2][0-9]|3[01]))*$
  - URLs:
    $(((http|https|ftp):)?([a-zA-Z0-9]+=%&__ ?]*)*$
Pattern Matching over Text — Advanced

- **Regular expressions** for more complex patterns.
  - Natural numbers: $0|\ [1-9] \ [0-9]*$
  - Dates: YYYY/MM/DD
    ```plaintext
    (((19|20)?[0-9]2[- /.
    (0?[1-9]|1[012])][- /.
    (0?[1-9]|1[2][0-9]|3[01]))]*
    ```
  - URLs:
    ```plaintext
    (((http|https|ftp):)?(\[
    [a-zA-Z0-9-]*\])+(_)(\[
    [a-zA-Z0-9-]*)2,4([[a-zA-Z0-9]+=%
    _ ?\]*))
    ```
- **Even more complex:**
  - Math expressions: $11 + 6, (3 + 45) \times 2, \ldots$
  - Time phrases: 15 January 2015, August 5, last week, \ldots
- Lexers, parsers, phrase taggers, named entity recognizers, \ldots
  - Rely on **regular expression matching** over text.
Pattern Matching over Text — Advanced

- **Regular expressions** for more complex patterns.
  - Natural numbers: 0| [1-9] [0-9] *
  - Dates: YYYY/MM/DD
    
    (((19|20)? [0-9]2[- /.] (0?[1-9]| 1[012]) [- /.] (0?[1-9]| 1[2][0-9]| 3[01])))*
  - URLs:
    
    (((http|https|ftp):)? ([a-zA-Z0-9_]+(_)([a-zA-Z0-9])2,4\([\[a-zA-Z0-9]+=%&\_ ?]*\))

- Even more complex:
  - Math expressions: 11 + 6, (3 + 45) × 2, ...
  - Time phrases: 15 January 2015, August 5, last week, ...

  - Lexers, parsers, phrase taggers, named entity recognizers, ...
    - Rely on **regular expression matching** over text.

- The moral: Pattern matching is a means of **extracting information** and **abstracting data**.
Inspired from textual pattern matching.
Find all matches of a pattern over a temporal behavior.
Challenges

From texts to temporal behaviors:

- Time is dense. Naive discretization does not scale.
- Lengths (durations) are more important for temporal behaviors than texts.
- Temporal behaviors are multi-dimensional unlike text.
- Temporal patterns talk about different dimensions.
Challenges

From texts to temporal behaviors:

- Time is dense. Naive discretization does not scale.
- Lengths (durations) are more important for temporal behaviors than texts.
- Temporal behaviors are multi-dimensional unlike text.
- Temporal patterns talk about different dimensions.

In this thesis, we address these challenges by

- Treating time in a symbolic and quantitative way.
- Making intersection (parallel composition) a first-class operation.
Contributions

- Introduce the **algebra of timed relations** that serves as a computational framework for the rest.
- Propose **timed regular expressions** as a concise, intuitive, and highly-expressive timed pattern specification language.
- Introduce **metric compass logic** and propose it as another timed pattern specification language.
- Provide **offline matching** algorithms for timed regular expressions and metric compass logic.
- Introduce **derivatives of timed regular expressions** and provide an **online matching** algorithm based on derivatives.
- Implement timed pattern matching algorithms and provide the tool **Montre** with some examples and case studies.
# Introduction

## Algebra of Timed Relations

- Goal: Develop a computational framework
  - Define basic objects
  - Define basic operations

## Timed Pattern Matching

## Going Online for Expressions

## Montre and Case Studies

## Conclusions
A time period \((t, t')\) is a pair such that \(t < t'\).

It begins at \(t\), ends at \(t'\), and has a duration of \(t' - t\).

Illustrated on the \(xy\)-plane.
A **timed relation** is a finitely representable set of time periods.

Boolean combinations of half-planes of six types:

1. $x < c$
2. $c < x$
3. $y < c$
4. $c < y$
5. $y - x < c$
6. $c < y - x$

Constraints 1 and 2 on beginnings
3 and 4 on ends
5 and 6 on durations
A **zone** is a convex timed relation, formed only by intersections of 1 to 6.

Represent a timed relation $Z$ by a finite union $R_Z$ of zones.

Ensure: Tightest bounds for zones.

Ensure: No zone in the representation covers another.
Operations on Timed Relations

- Boolean (set) operations
  - Union (\(\cup\)), Intersection (\(\cap\)), and Complementation (\(\bar{\cdot}\))
- Sequential operations
  - Concatenation (\(\cdot\)) and Repetition (\(\dagger\))
- Compass operations
  - Based on binary relations between time periods (so called Allen’s relations).

Timed relations are closed under Boolean, sequential, and compass operations.

Efficient algorithms to compute over finite unions of zones. (e.g. several adaptations of the plane sweep algorithm)
Timed Pattern Matching

- Goal: Introduce TPM
  - Define timed behaviors
  - Define timed pattern specification languages
  - Compute matches offline
Timed Behaviors

- Continuous observations of some propositions denoting some states and activities of systems.
- Modeled as a sequence of uniform segments.
- For example, a behavior $w$ over $p_1$, $p_2$, and $p_3$

$$w = (0, 1.5, \begin{bmatrix} 0 \\ 0 \end{bmatrix}), (1.5, 2, \begin{bmatrix} 1 \\ 0 \end{bmatrix}), (2, 3.2, \begin{bmatrix} 1 \\ 1 \end{bmatrix}), \ldots$$

- Argue that continuous-time functions complicate needlessly.
Timed Regular Expressions (TRE)

- Extend REs with the notion of **time** and **timing constraints**.
- Describe timed patterns (sets of timed behaviors).
Timed Regular Expressions (TRE)

- Extend REs with the notion of **time** and **timing constraints**.
- Describe timed patterns (sets of timed behaviors).
- Defined inductively over a set $P$ of atomic propositions:
  - An atomic proposition $p \in P$ is a TRE.
  - If $\varphi_1$ and $\varphi_2$ are TREs, then $\varphi_1 \cup \varphi_2$, $\varphi_1 \cdot \varphi_2$, and $\varphi_1^+$ are TREs.
  - If $\varphi_1$ and $\varphi_2$ are TREs, then $\varphi_1 \cap \varphi_2$ and $\langle \varphi_1 \rangle_I$ are TREs.
Timed Regular Expressions (TRE)

- Extend REs with the notion of *time* and *timing constraints*.
- Describe timed patterns (sets of timed behaviors).
- Defined inductively over a set $P$ of atomic propositions:
  - An atomic proposition $p \in P$ is a TRE.
  - If $\varphi_1$ and $\varphi_2$ are TREs, then $\varphi_1 \cup \varphi_2$, $\varphi_1 \cdot \varphi_2$, and $\varphi_1^+$ are TREs.
  - If $\varphi_1$ and $\varphi_2$ are TREs, then $\varphi_1 \cap \varphi_2$ and $\langle \varphi_1 \rangle I$ are TREs.
- Intuitively,
  - $p$ specifies that $p$ holds continuously for an arbitrary *duration*.
  - $\varphi_1 \cdot \varphi_2$ specifies that $\varphi_2$ occurs *right after* $\varphi_1$.
  - $\varphi_1 \cap \varphi_2$ specifies that $\varphi_1$ and $\varphi_2$ occurs *simultaneously*.
  - $\langle \varphi \rangle I$ specifies that *duration* of $\varphi$ is in the interval $I$. 
Duration Constraints Explained

\[ p_1 \vdash p_1 \cdot p_2 \]

\[ p_2 \vdash p_1 \cdot p_2 \]
Duration Constraints Explained

$p_1$  

$p_2$

$p_1 \models p_1 \cdot p_2$

$p_1 \models p_1 \cdot p_2$

$p_1 \models \langle p_1 \cdot p_2 \rangle_{[5,7]}$

(p satisfies the constraint)

$p_1 \models \langle p_1 \cdot p_2 \rangle_{[5,7]}$

(falls short)
Metric Compass Logic (MCL)

- Extends modal logic of time periods with **timing constraints**.
- Originally proposed for AI. We use for matching.
Metric Compass Logic (MCL)

- Extends modal logic of time periods with **timing constraints**.
- Originally proposed for AI. We use for matching.
- Defined inductively over a set $P$ of atomic propositions:
  - An atomic proposition $p \in P$ is a MCL formula.
  - If $\varphi_1$ and $\varphi_2$ are formulas, then $\varphi_1 \cup \varphi_2$, $\varphi_1 \cap \varphi_2$, and $\overline{\varphi_1}$ are formulas.
  - If $\varphi$ is a formula, then $\Diamond_I \varphi$, $\Diamond_I \varphi$, $\Diamond_I \varphi$, $\Diamond_I \varphi$, $\Diamond_I \varphi$, and $\Diamond_I \varphi$ are formulas.
Express all possible situations between time periods.
• Express all possible situations between time periods.
• Extended with timing constraints.
  (Restrict range of quantification at one side)
Timed Pattern Matching

- A computation for identifying **all segments** of a timed behavior that satisfy a timed pattern.
A computation for identifying all segments of a timed behavior that satisfy a timed pattern.

Patterns specified in TRE and MCL.
Timed Pattern Matching

- A computation for identifying **all segments** of a timed behavior that satisfy a timed pattern.
- Patterns specified in TRE and MCL.
- The set of all satisfying segments is called the **match set** of the pattern $\varphi$ over a timed behavior $w$.

$$\mathcal{M}_w(\varphi) = \{(t, t') \mid w[t, t'] \text{ satisfies } \varphi\}$$
Timed Pattern Matching

- A computation for identifying all segments of a timed behavior that satisfy a timed pattern.
- Patterns specified in TRE and MCL.
- The set of all satisfying segments is called the match set of the pattern \( \varphi \) over a timed behavior \( w \).

\[
\mathcal{M}_w(\varphi) = \{(t, t') \mid w[t, t'] \text{ satisfies } \varphi\}
\]

Computing match set = Evaluating \( \varphi \) to a timed relation
Timed Pattern Matching — Example

\[ \varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3 \]
$\varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3$
$\varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3$
$\varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3$
Computing Matches Offline

\[ \varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3 \]
\[ \varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3 \]
\[ \varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3 \]
Interpreting Matches

\[ \varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3 \]

Timed Pattern Matching
Interpreting Matches

\[ \varphi = p_1 \cdot (p_2 \cdot p_3)[2,4] \cap (p_1 \cdot p_2)[3,5] \cdot p_3 \]
### Going Online for Expressions

- **Goal:** Develop online TPM
  - Define derivatives of TRE
  - Compute matches using derivatives

### Montre and Case Studies

### Conclusions
Why Going Online?

- Offline matching requires the whole behavior beforehand.
- Reacting to events and patterns as soon as they occur.
  - Alerting humans (for bad patterns)
  - Taking automatic actions
- No need to store temporal behaviors.
Sequential Model of Computation (Recap)

- Formalizes online computations.
- Defines a sequence-to-sequence transformation $x_1x_2x_3\ldots x_n\rightarrow y_1y_2y_3\ldots y_n$, as follows.

$$
\begin{align*}
S_0 & \xrightarrow{x_1} S_1 & \xrightarrow{x_2} S_2 & \xrightarrow{x_3} S_3 & \xrightarrow{x_4} \ldots & \xrightarrow{x_n} S_n \\
\Downarrow & \Downarrow & \Downarrow & \Downarrow & \ldots & \Downarrow \\
Y_1 & Y_2 & Y_3 & \ldots & \Downarrow & Y_n
\end{align*}
$$

- The output $y_i$ at the step $i$ is completely determined by the initial state and the input $x_1\ldots x_i$.

- Closely related to automata. (Focus on input/output)
Derivatives of Regular Expressions

- A technique to compute word acceptance for REs.
- A set of rewrite rules $D_x(\varphi)$ that yields the residual expression w.r.t. a letter $x$.
- Empty word check for the output.
- A sequential computation using derivatives:

\[\varphi_0 \xrightarrow{D^a_1} \varphi_1 \xrightarrow{D^a_2} \varphi_2 \xrightarrow{D^a_3} \varphi_3 \xrightarrow{D^a_4} \varphi_4 \xrightarrow{D^a_5} \varphi_5\]

\[\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow\]

\[0 \quad 1 \quad 0 \quad 0 \quad 1\]

- Output at each step: Acceptance of the word read so far.
• Adapting derivatives for dense time.
• This involves a derivation w.r.t. all factors of a segment.
  • Necessitates a more symbolic approach.
  • Solved by considering timed relations to be constant values in timed regular expressions.
    (Booleans $\rightarrow$ Timed Relations)
• The resulting set of rules
  • Maintains elegance of classical derivatives.
  • Easy to show its correctness.
Online Timed Pattern Matching

- A sequential computation for matches.
- The state of the online procedure is a single expression.
- For each segment $u$ of the input:
  - Derive the current expression with respect to $u = (t, t', a)$.
  - Extract matches from the derived expression.

\[
\psi_0 \xrightarrow{\Delta u_1} \psi_1 \xrightarrow{\Delta u_2} \psi_2 \xrightarrow{\Delta u_3} \psi_3
\]

- Output at each step: Matches ending in the current segment.
Introduction

Algebra of Timed Relations

Timed Pattern Matching

Going Online for Expressions

Montre and Case Studies

- Describe our timed pattern matcher tool
- Demonstrate some case studies:
  - Measuring performance of a communication protocol
  - Detecting sprints of soccer players
    (Not here)
  - Detecting daily activities over a video dataset
    (Not here)

Conclusions
Montre: Tool Description

- Montre is a command-line program that implements timed pattern matching algorithms.
- In C++ and PureLang (term rewriting language).
- Provides a standard text-based input/output interface.
- Enhanced by Anchors and Boolean Layer.
- Available at github.com/doganulus/montre
Montre: A tool in the pipeline

- For most applications, many different tasks have to be done.
- Timed pattern matching is not an isolated task.
Montre: Performance

<table>
<thead>
<tr>
<th>Test Patterns</th>
<th>Offline Algorithm</th>
<th></th>
<th></th>
<th></th>
<th>Online Algorithm</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100K</td>
<td>500K</td>
<td>1M</td>
<td>100K</td>
<td>500K</td>
<td>1M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p )</td>
<td>0.06/17</td>
<td>0.27/24</td>
<td>0.51/33</td>
<td>6.74/14</td>
<td>29.16/14</td>
<td>57.87/14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p \times q )</td>
<td>0.08/21</td>
<td>0.42/46</td>
<td>0.74/77</td>
<td>8.74/14</td>
<td>42.55/14</td>
<td>81.67/14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \langle p \times q \times \langle p \times q \times p \rangle \times q \times p \rangle \times p \rangle \times q )</td>
<td>0.23/28</td>
<td>1.09/77</td>
<td>2.14/140</td>
<td>28.07/14</td>
<td>130.96/14</td>
<td>270.45/14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \langle p \times q \rangle \times r \times p \times \langle q \times r \rangle \times p )</td>
<td>0.13/23</td>
<td>0.50/51</td>
<td>1.00/86</td>
<td>15.09/15</td>
<td>75.19/15</td>
<td>148.18/15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p \times (q \times r) )</td>
<td>0.11/20</td>
<td>0.49/37</td>
<td>0.96/60</td>
<td>11.53/15</td>
<td>52.87/15</td>
<td>110.58/15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{size 1M} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p \times q \times r \times p )</td>
<td>0.40/31</td>
<td>1.98/143</td>
<td>3.93/268</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{size 1M} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- A few seconds for offline, a few hundred seconds for online.
- Linear execution time for typical inputs.
- Constant memory consumption for online algorithm.
Distributed Systems Interface (DSI3) is a bus protocol developed for the automotive industry. The communication is realized by voltage and current signals. The performance is evaluated by measuring some quantitative properties such as **time between consecutive discovery pulses** (#3).
Case Study: Work Flow

- Our workflow for the DSI3 case study:

  Electrical Signals -> Noise Filtering -> Timed Abstraction -> Timed Pattern Matching -> Measurement -> Visualization -> Plots

  Timed Patterns (describing measurement windows)
Case Study: Matching and Measuring

- Specify the pulse pattern:

\[ \varphi_{dp} = \langle \text{VMid} \cdot \text{VLow} \cdot \text{VMid} \rangle_{[12,20]} \cap \Diamond_{[0,5]} \text{VHigh} \cap \Diamond_{[0,5]} \text{VHigh} \]

- Specify the tail portion:

\[ \varphi_{distpp} = \varphi_{dp} \cdot \text{VHigh} \cap \Diamond \varphi_{dp} \]

- Compute the match set \( M(\varphi_{distpp}) \) over behaviors of the protocol.

- Measure the duration of each match and plot a histogram.

<table>
<thead>
<tr>
<th>Label</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHigh</td>
<td>&gt; 7.8</td>
</tr>
<tr>
<td>VMiddle</td>
<td>4.92 - 7.8</td>
</tr>
<tr>
<td>VLow</td>
<td>&lt; 4.92</td>
</tr>
</tbody>
</table>
Conclusions

- Adapted pattern matching for temporal behaviors.
- Developed theoretical foundations and algorithms for timed pattern matching.
- Implemented the first timed pattern matching tool.
- Explored several application areas via case studies.
Future Work

● **Improve the efficiency:**
  ● Automata-based timed pattern matching
  ● Representing timed relations with decision diagrams

● **Increase the expressiveness:**
  ● Conjunctive grammars for timed patterns

● **Specialize for some application domains:**
  ● Runtime verification
  ● Temporal data analysis
  ● Robotics


Thank you for your attention!
The satisfaction relation $\models$ of a timed regular expression $\varphi$ in a temporal structure $W = (\Omega(T), V)$ induced by a timed behavior $w$ with a time domain $T$, relative to a time period $(t, t') \in \Omega(T)$ is defined as follows:

- $(W, t, t') \models p \iff (t, t') \in V(p)$
- $(W, t, t') \models \varphi_1 \cap \varphi_2 \iff (W, t, t') \models \varphi_1$ and $(W, t, t') \models \varphi_2$
- $(W, t, t') \models \varphi_1 \cdot \varphi_2 \iff \exists t''. (W, t, t'') \models \varphi$ and $(W, t'', t') \models \psi$
- $(W, t, t') \models \varphi_1 \cup \varphi_2 \iff (W, t, t') \models \varphi_1$ or $(W, t, t') \models \varphi_2$
- $(W, t, t') \models \varphi^+ \iff \exists k \geq 1. (W, t, t') \models \varphi^k$
- $(W, t, t') \models \langle \varphi \rangle_I \iff (W, t, t') \models \varphi$ and $t' - t \in I$
The satisfaction relation \( \models \) of a metric compass logic formula \( \varphi \) in a temporal structure \( W = (\Omega(T), V) \) induced by a timed behavior \( w \) with a time domain \( T \), relative to a time period \( (t, t') \in \Omega(T) \) is defined as follows:

\[
\begin{align*}
(W, t, t') & \models p \quad \iff \quad (t, t') \in V(p) \\
(W, t, t') & \models \overline{\varphi} \quad \iff \quad (W, t, t') \not\models \varphi \\
(W, t, t') & \models \varphi_1 \lor \varphi_2 \quad \iff \quad (W, t, t') \models \varphi_1 \text{ or } (W, t, t') \models \varphi_2 \\
(W, t, t') & \models \Diamond_{1} \varphi \quad \iff \quad \exists t'' \in (t, t'). \ t' - t'' \in I \text{ and } (W, t, t'') \models \varphi \\
(W, t, t') & \models \Diamond_{1} \varphi \quad \iff \quad \exists t'' \in (t', \infty). \ t'' - t \in I \text{ and } (W, t', t'') \models \varphi \\
(W, t, t') & \models \Diamond_{1} \varphi \quad \iff \quad \exists t'' \in (-\infty, t). \ t - t'' \in I \text{ and } (W, t'', t) \models \varphi \\
(W, t, t') & \models \Diamond_{1} \varphi \quad \iff \quad \exists t'' \in (t', \infty). \ t'' - t' \in I \text{ and } (W, t', t'') \models \varphi \\
(W, t, t') & \models \Diamond_{1} \varphi \quad \iff \quad \exists t'' \in (-\infty, t). \ t - t'' \in I \text{ and } (W, t'', t) \models \varphi
\end{align*}
\]
Given a (left-reduced) timed regular expression $\varphi$ and a uniform timed behavior $v : (t, t', a)$, applying the following rules yields an expression $\psi$ such that $[\psi] = \Delta_v([\varphi])$.

\[
\begin{align*}
\Delta_v(Z) &= \emptyset \\
\Delta_v(p) &= \begin{cases} 
Z \cup Z \cdot p & \text{if } a(p) = 1 \text{ where } \\
\emptyset & \text{otherwise}
\end{cases} \\
\Delta_v(\psi_1 \cdot \psi_2) &= \Delta_v(\psi_1) \cdot \psi_2 \cup xt(\psi_1 \cup \Delta_v(\psi_1)) \cdot \Delta_v(\psi_2) \\
\Delta_v(\psi_1 \cup \psi_2) &= \Delta_v(\psi_1) \cup \Delta_v(\psi_2) \\
\Delta_v(\psi_1 \cap \psi_2) &= \Delta_v(\psi_1) \cap \Delta_v(\psi_2) \\
\Delta_v(\langle \psi \rangle_I) &= \langle \Delta_v(\psi) \rangle_I \\
\Delta_v(\psi^+) &= xt(\Delta_v(\psi))^* \cdot \Delta_v(\psi) \cdot \psi^*
\end{align*}
\]