Thesis Defense

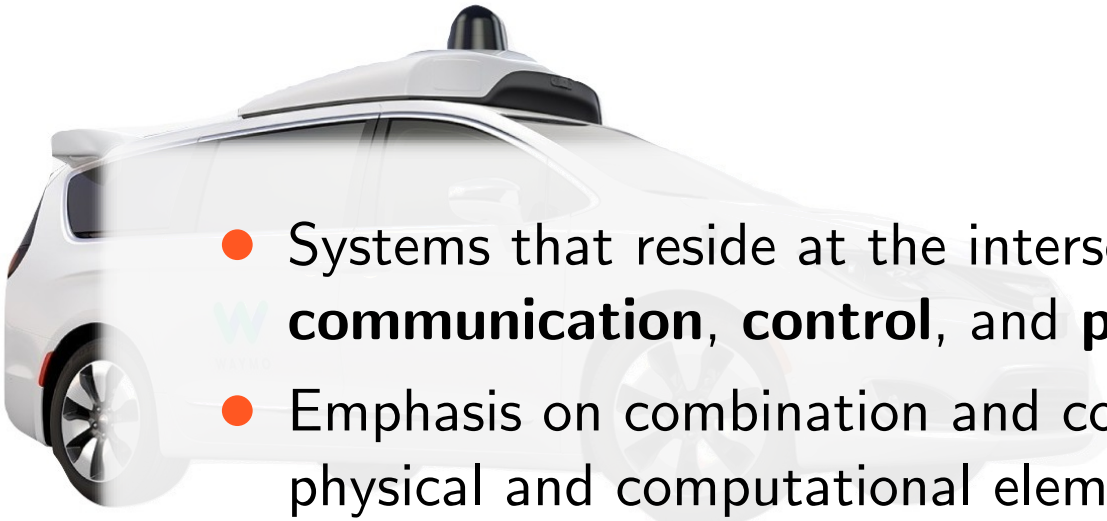
Pattern Matching with Time — Theory and Applications

Doğan Ulus

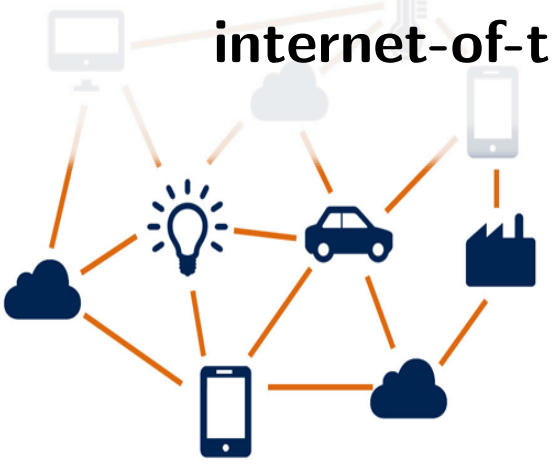
January 15, 2018
Grenoble, France

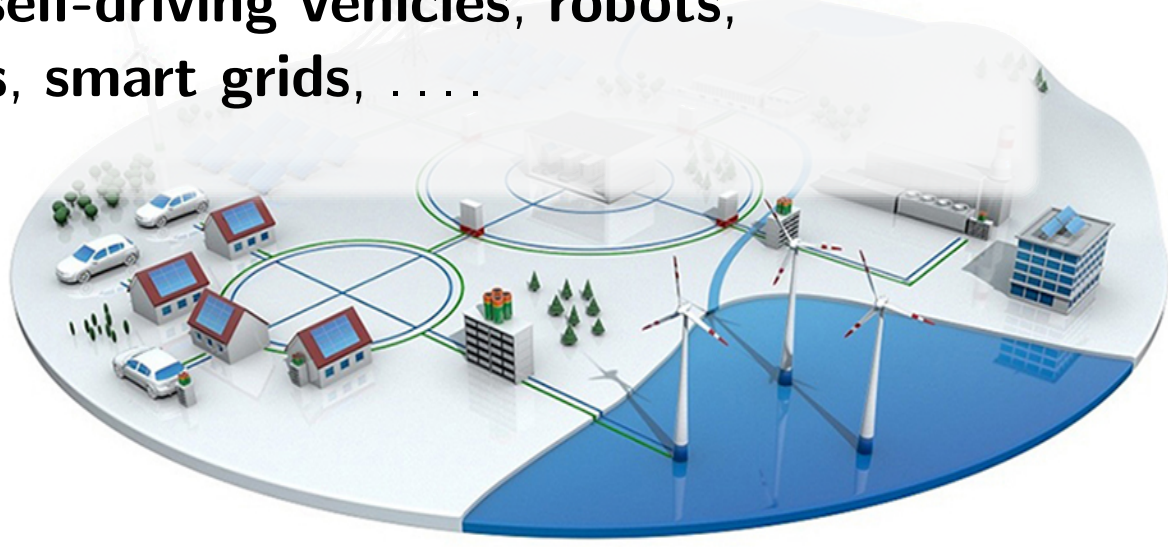
Supervisor	Reviewers	Examiners	Invited
Oded Maler	**Rajeev Alur**	**Saddek Bensalem**	**Eugene Asarin**
	Radu Grosu	**Ahmed Bouajjani**	**Dejan Nickovic**
	Kim G. Larsen	**Patricia Bouyer**	

Cyber-Physical Systems

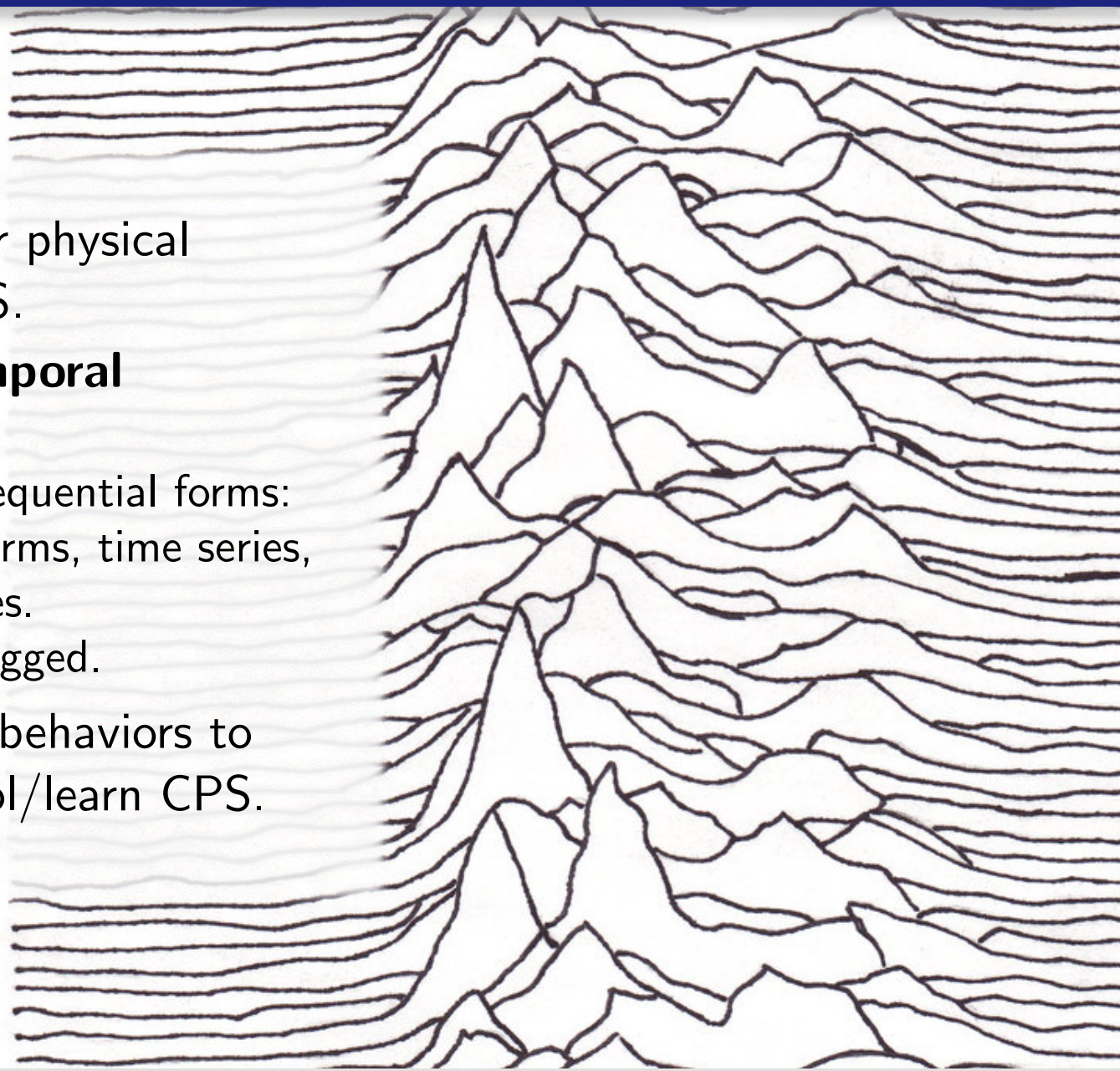


- Systems that reside at the intersection of **communication**, **control**, and **physical processes**.
- Emphasis on combination and coordination between physical and computational elements.
- Examples include **self-driving vehicles**, **robots**, **internet-of-things**, **smart grids**, ...





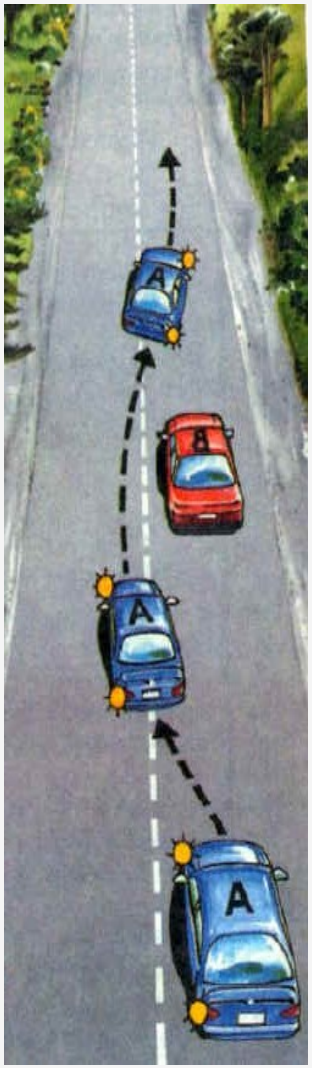
Temporal Behaviors



- **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

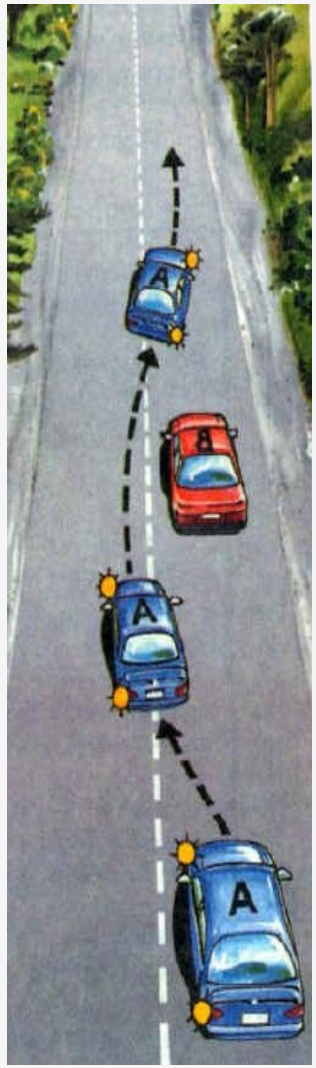
- Specific shapes on waveforms:
  - Rise and falls, various pulses, decays, ...
- Specific arrangements of physical observations.
  - High speed period after high acceleration, ...
- Sequences of actions, simultaneous occurrences, repetitions.
  - Overtaking a car.
  - Speeding-up while overtaken. (illegal)



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.



Pattern Matching over Text — First Example

- 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?

Pattern Matching over Text — First Example

- 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"

Find: patterns

Pattern Matching over Text — First Example

- 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"

Find: patterns

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]|[12][0-9]|3[01]))*
    ```
  - URLs:

    ```
    (((http|https|ftp):)?([[a-zA-Z0-9]])+()([[a-zA-Z0-9]])2,4([[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

    ```
    ((19|20)?[0-9]2[- /.](0?[1-9]|1[012])[- /.](0?[1-9]|[12][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: $11 + 6$, $(3 + 45) \times 2$, ...
  - Time phrases: 15 January 2015, August 5, last week, ...
- Lexers, parsers, phrase taggers, named entity recognizers, ...
  - Rely on **regular expression matching** over text.

Pattern Matching over Text — Advanced

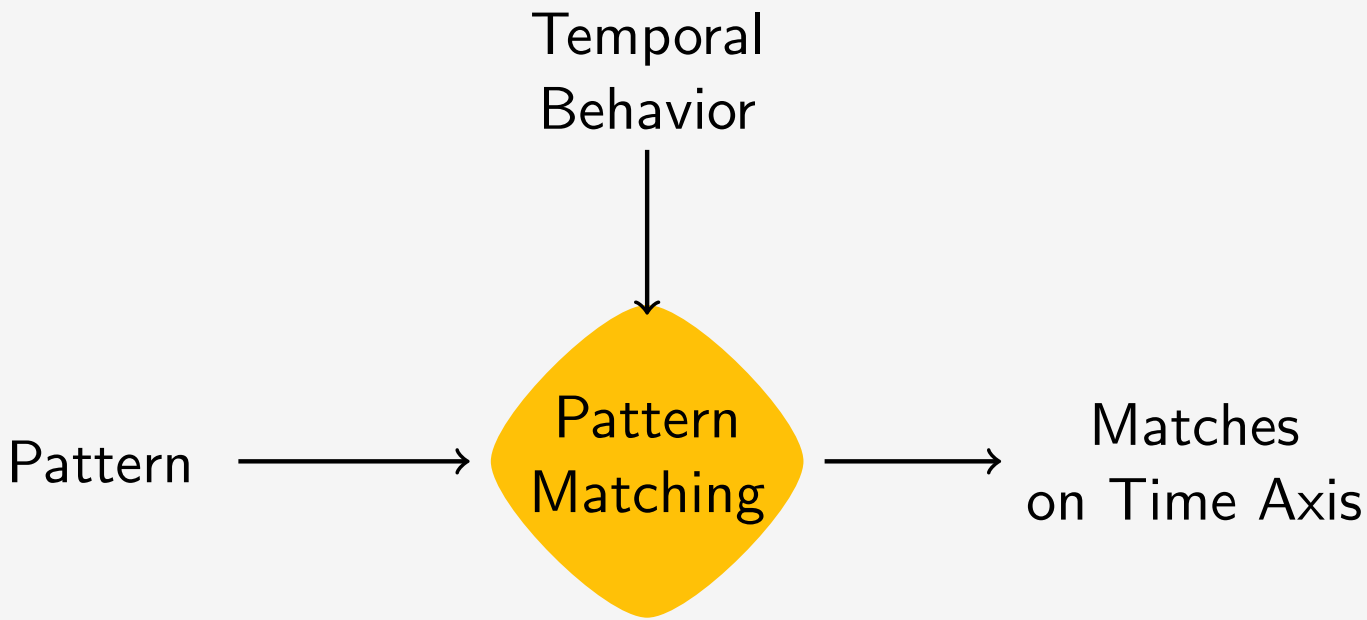
- **Regular expressions** for more complex patterns.
  - Natural numbers: `0|[1-9][0-9]*`
  - Dates: YYYY/MM/DD

    `((19|20)?[0-9]2[- /.](0?[1-9]|1[012])[- /.](0?[1-9]|[12][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: $11 + 6$, $(3 + 45) \times 2$, ...
  - 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**.

Problem Overview

- 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

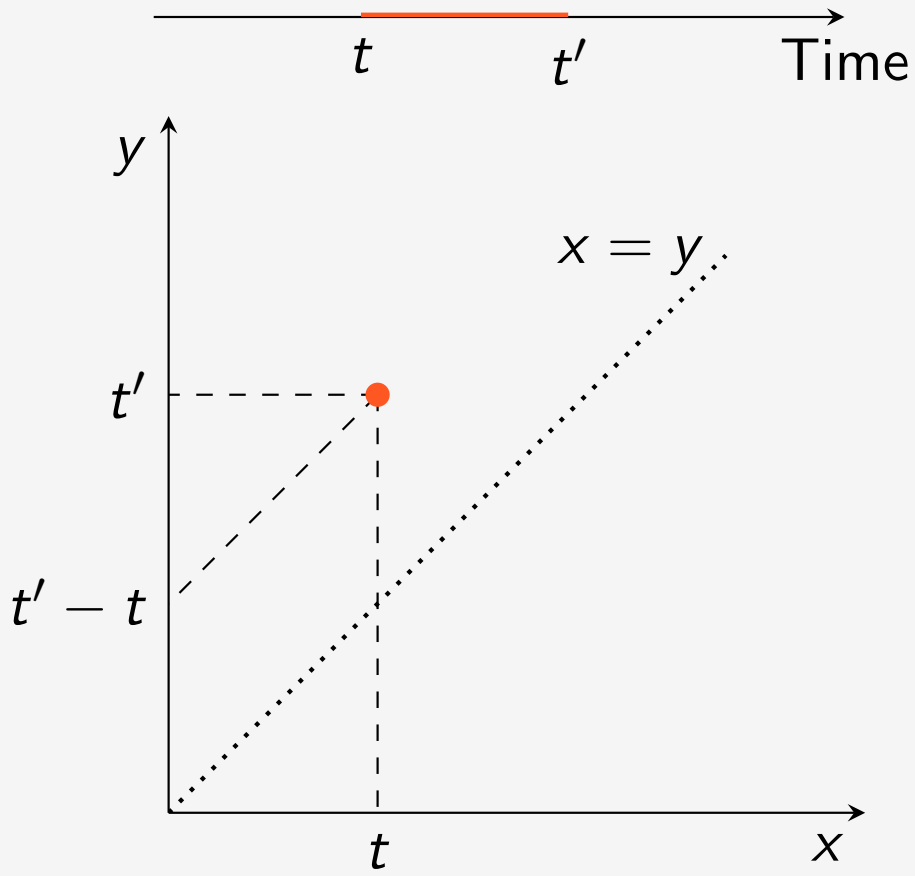
Going Online for Expressions

Montre and Case Studies

Conclusions

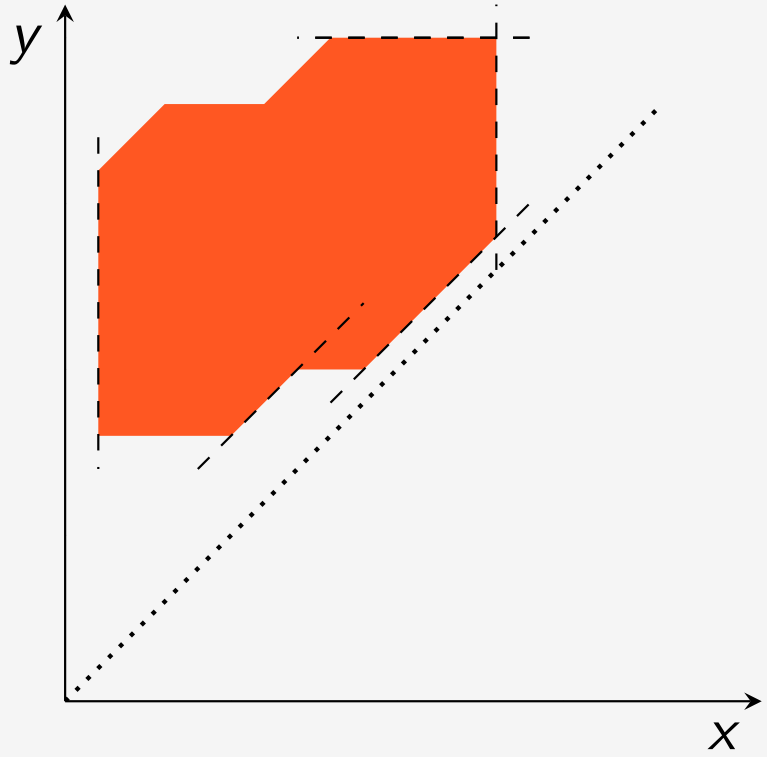
Time Periods

- 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.



Timed Relations

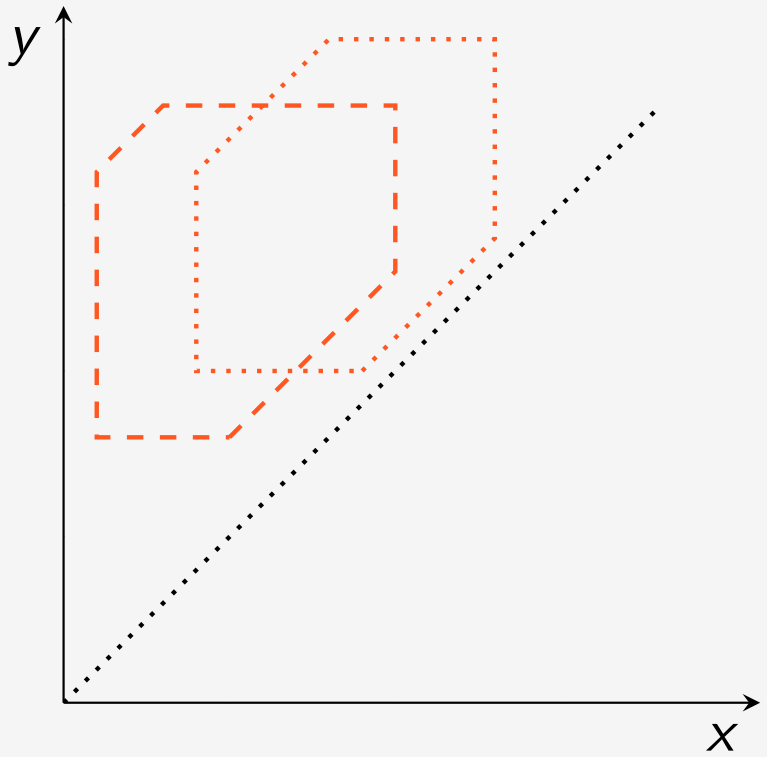
- A **timed relation** is a finitely representable set of time periods.
- Boolean combinations of half-planes of six types:
  1. $x < c$
  2. $c < x$
  3. $y < c$
  4. $c < y$
  5. $y - x < c$
  6. $c < y - x$



- Constraints 1 and 2 on beginnings
  3 and 4 on ends
  5 and 6 on durations

Representing Timed Relations

- A **zone** is a convex timed relation, formed only by intersections of ❶ to ❻.
- 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 ($\overline{\ }$)
- Sequential operations
  - Concatenation ($\cdot$) and Repetition ($^+$)
- 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)

Introduction

Algebra of Timed Relations

Timed Pattern Matching

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

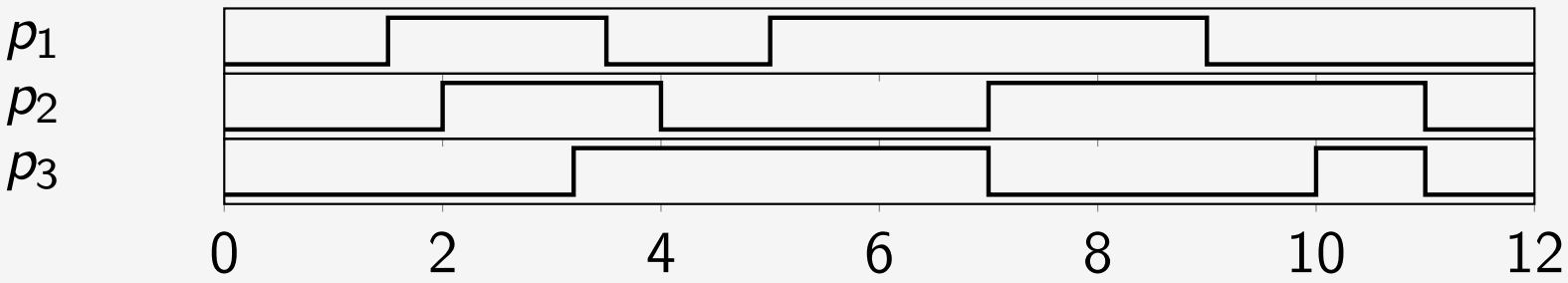
Going Online for Expressions

Montre and Case Studies

Conclusions

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



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

- 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).

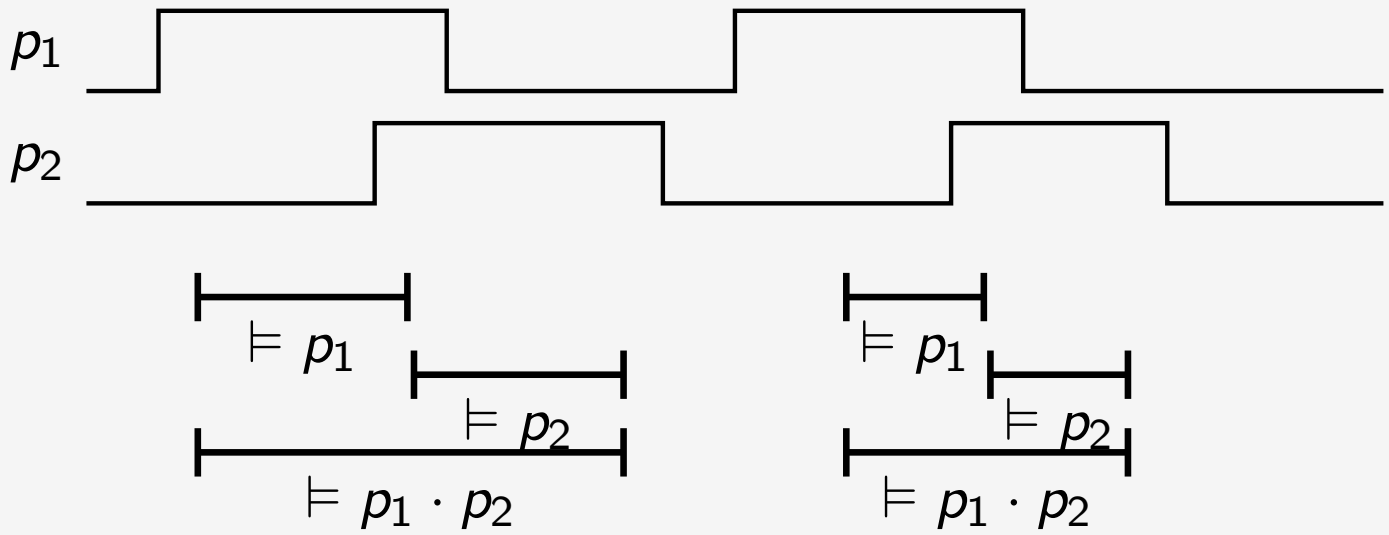
Timed Regular Expressions (TRE)

- 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 φ_1 and φ_2 are TREs, then $\varphi_1 \cup \varphi_2$, $\varphi_1 \cdot \varphi_2$, and φ_1^+ are TREs.
  - If φ_1 and φ_2 are TREs, then $\varphi_1 \cap \varphi_2$ and $\langle \varphi_1 \rangle_I$ are TREs.

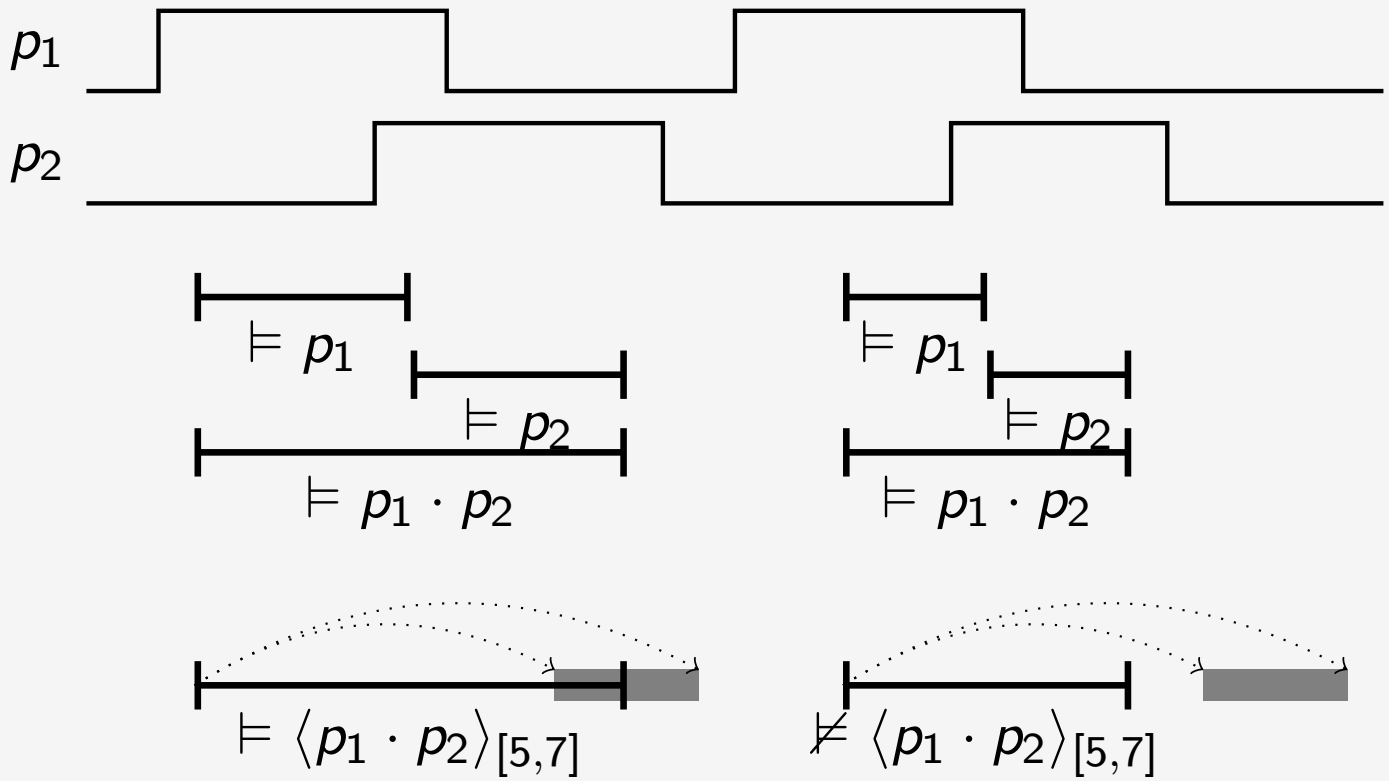
Timed Regular Expressions (TRE)

- 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 φ_1 and φ_2 are TREs, then $\varphi_1 \cup \varphi_2$, $\varphi_1 \cdot \varphi_2$, and φ_1^+ are TREs.
  - If φ_1 and φ_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 φ_2 occurs **right after** φ_1.
  - $\varphi_1 \cap \varphi_2$ specifies that φ_1 and φ_2 occurs **simultaneously**.
  - $\langle \varphi \rangle_I$ specifies that **duration** of φ is in the interval I.

Duration Constraints Explained



Duration Constraints Explained



(satisfies the constraint) (falls short)

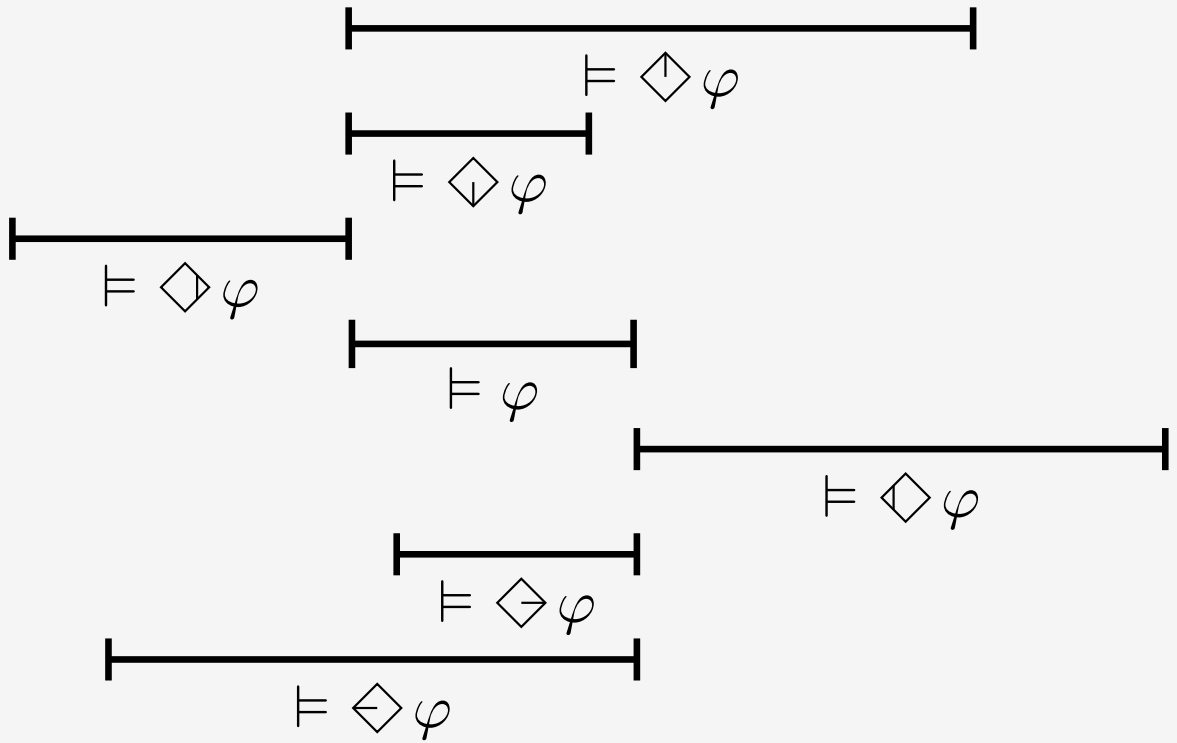
Metric Compass Logic (MCL)

- Extends modal logic of time periods with **timing constraints**.
- Originally proposed for AI. We use for matching.

Metric Compass Logic (MCL)

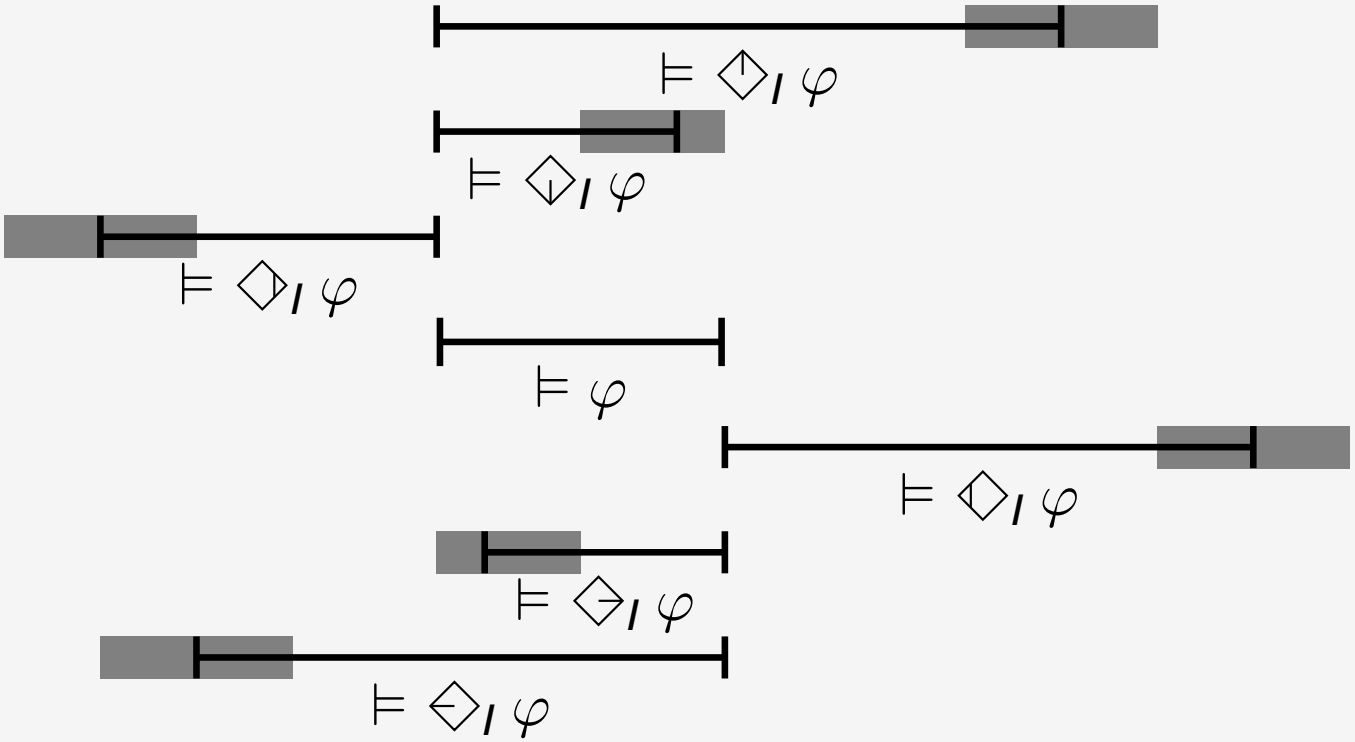
- Extends modal logic of time periods with **timing constraints**.
- Originally proposed for AI. We use for matching.
- Defined inductively over a set P of atomic propositions:
  - An atomic proposition $p \in P$ is a MCL formula.
  - If φ_1 and φ_2 are formulas, then $\varphi_1 \cup \varphi_2$, $\varphi_1 \cap \varphi_2$, and $\overline{\varphi_1}$ are formulas.
  - If φ is a formula, then $\diamondsuit_I \varphi$, $\diamondsuit_I \varphi$, $\diamondsuit_I \varphi$, $\diamondsuit_I \varphi$, $\diamondsuit_I \varphi$, and $\diamondsuit_I \varphi$ are formulas.

Diamonds Explained



- Express all possible situations between time periods.

Diamonds Explained



- 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.

Timed Pattern Matching

- 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 φ over a timed behavior w.

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

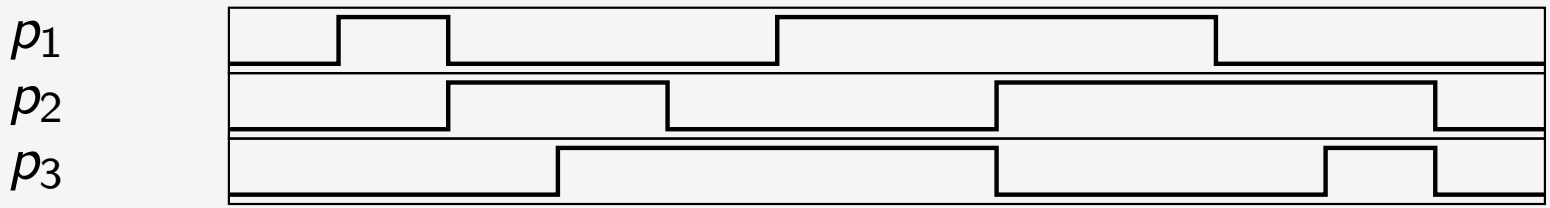
Timed Pattern Matching

- 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 φ over a timed behavior w.

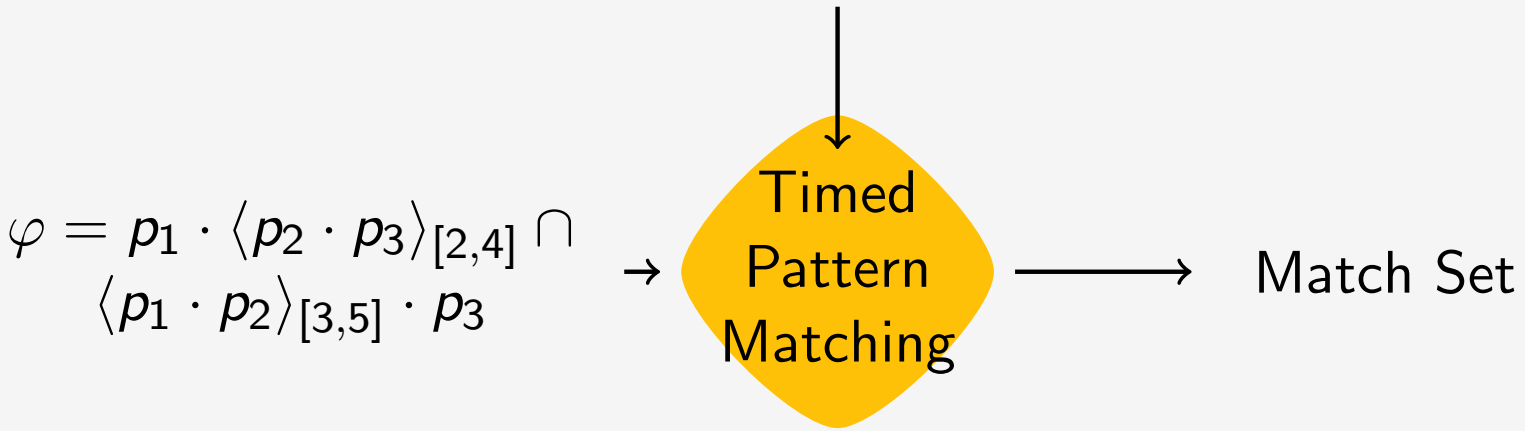
$$\mathcal{M}_w(\varphi) = \{(t, t') \mid w[t, t'] \text{ satisfies } \varphi\}$$

Computing match set = Evaluating φ to a timed relation

Timed Pattern Matching — Example

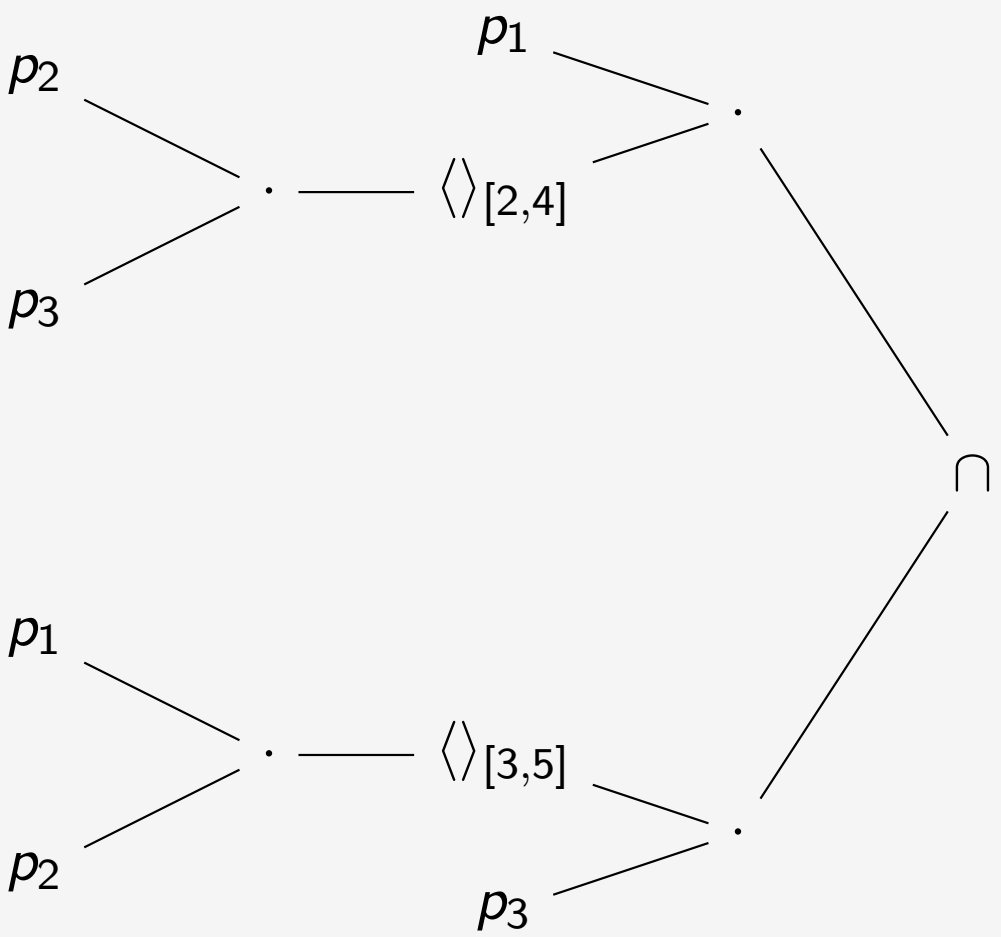


$$\varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3$$



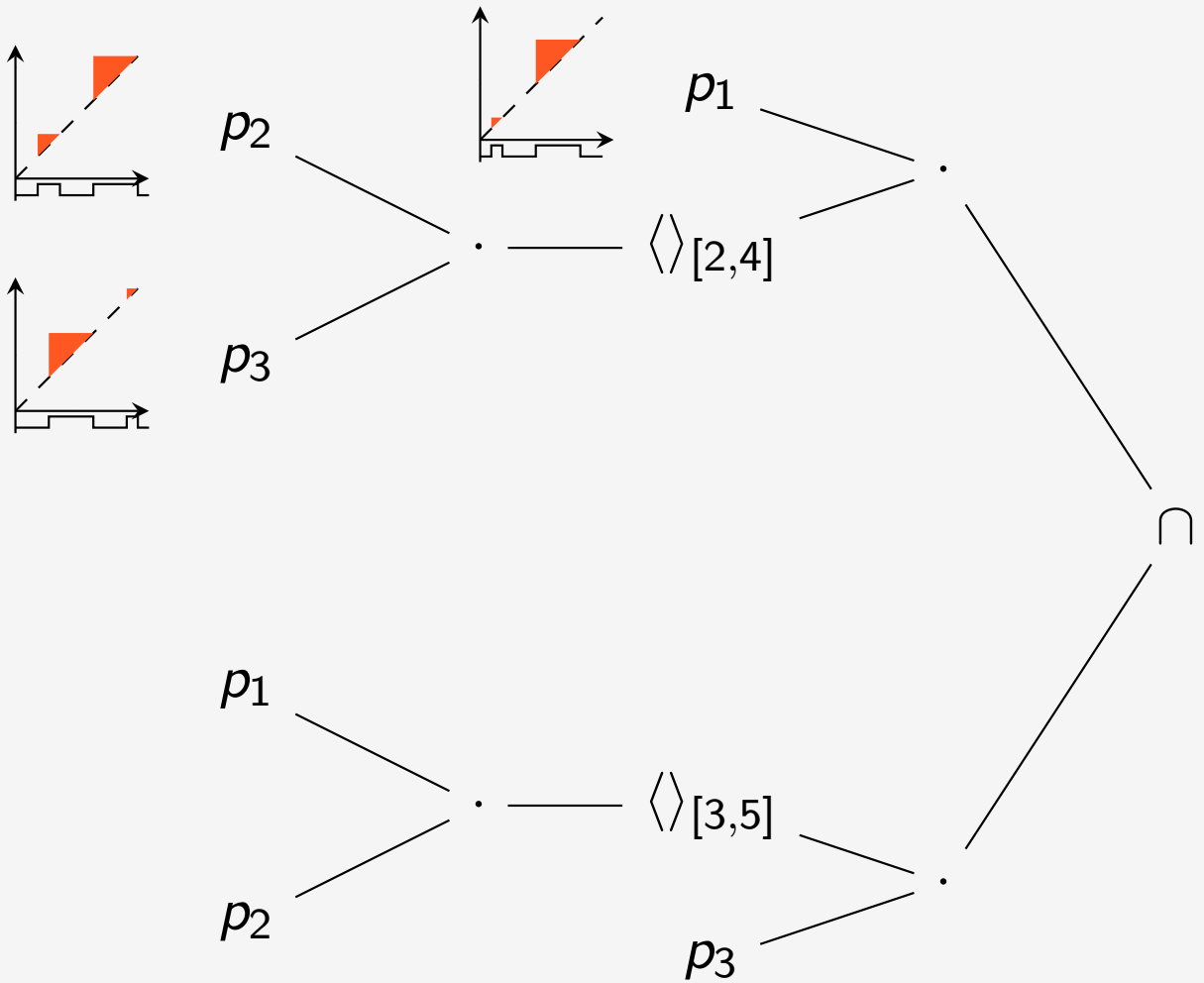
Computing Matches Offline

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



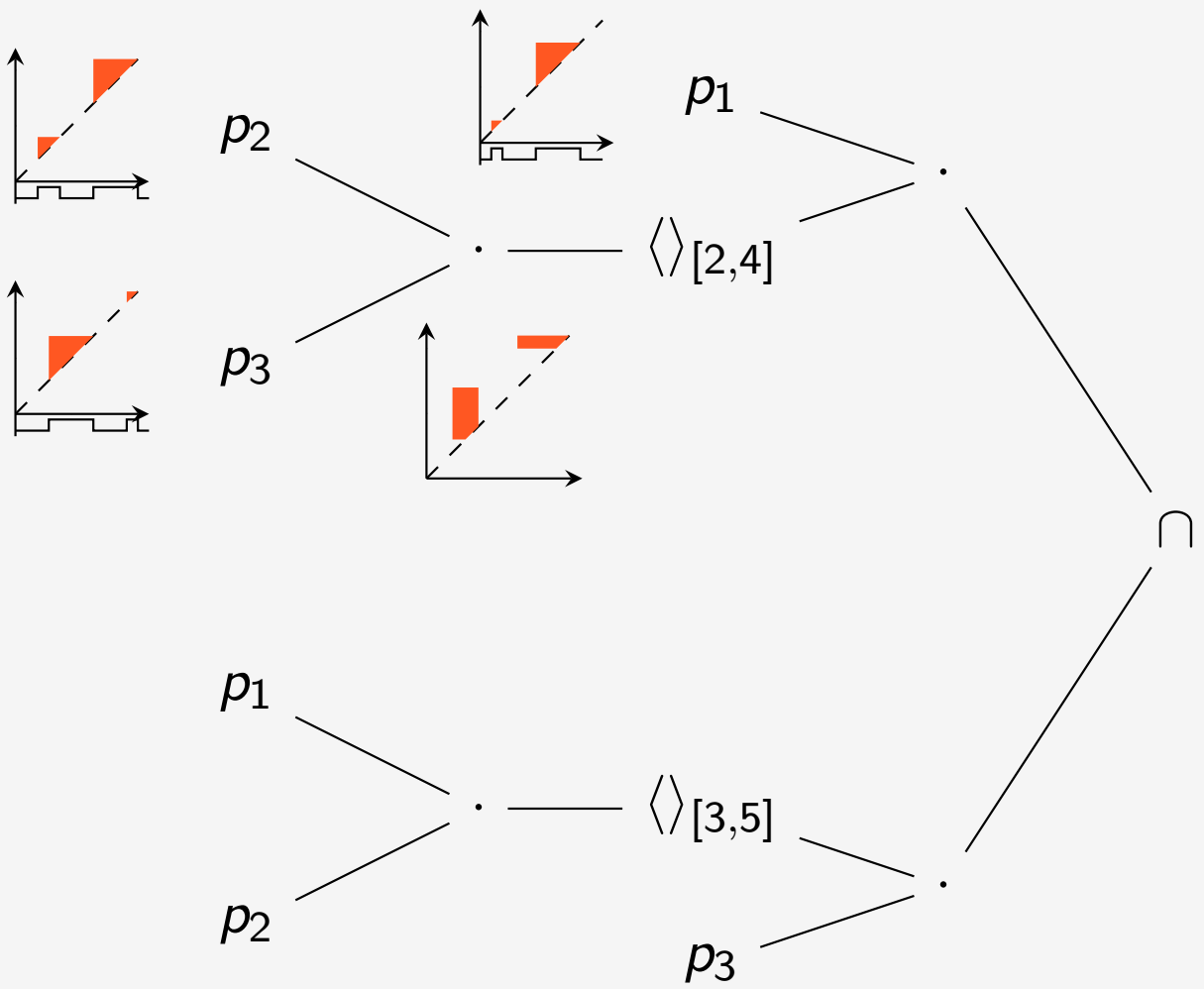
Computing Matches Offline

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



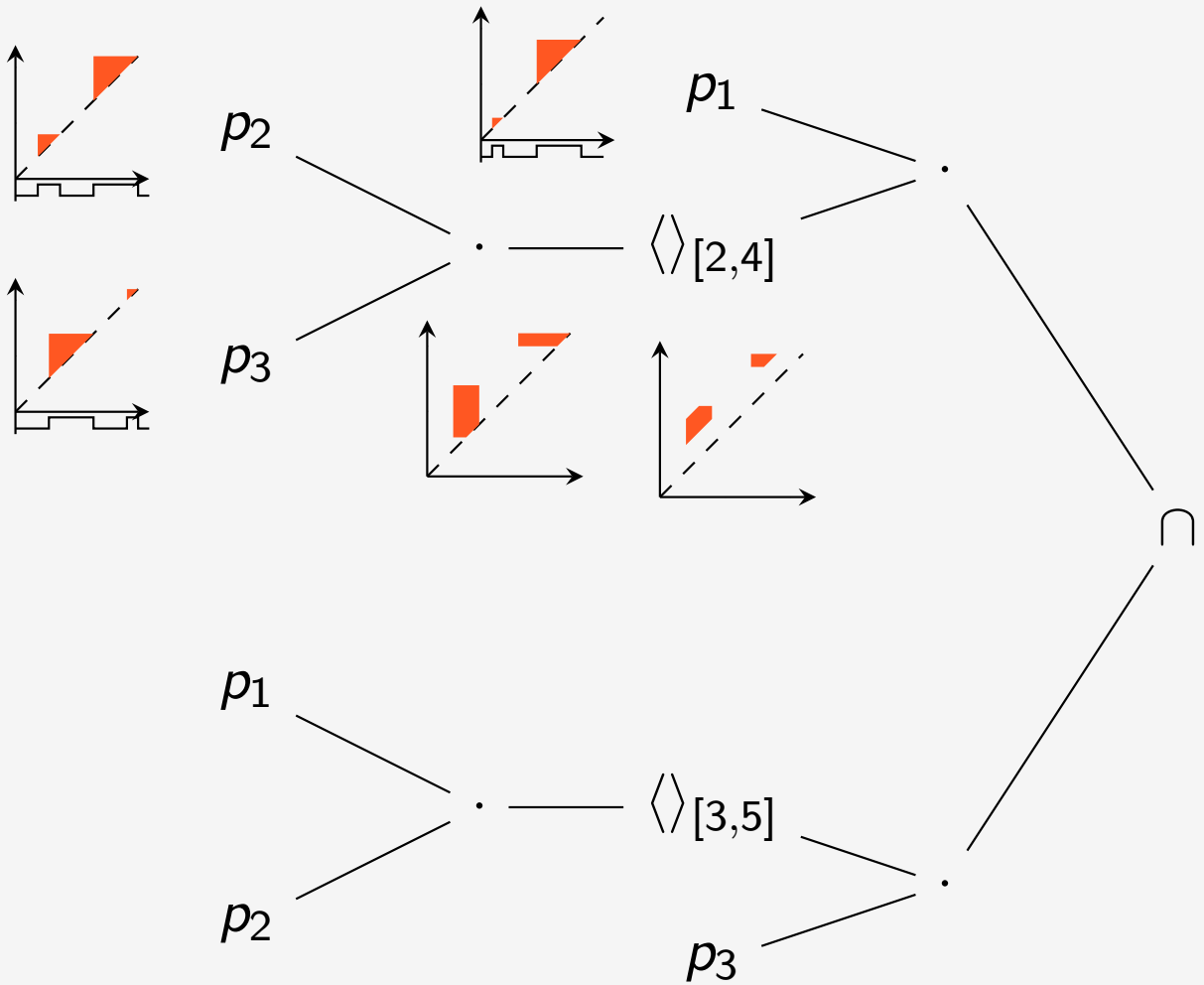
Computing Matches Offline

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



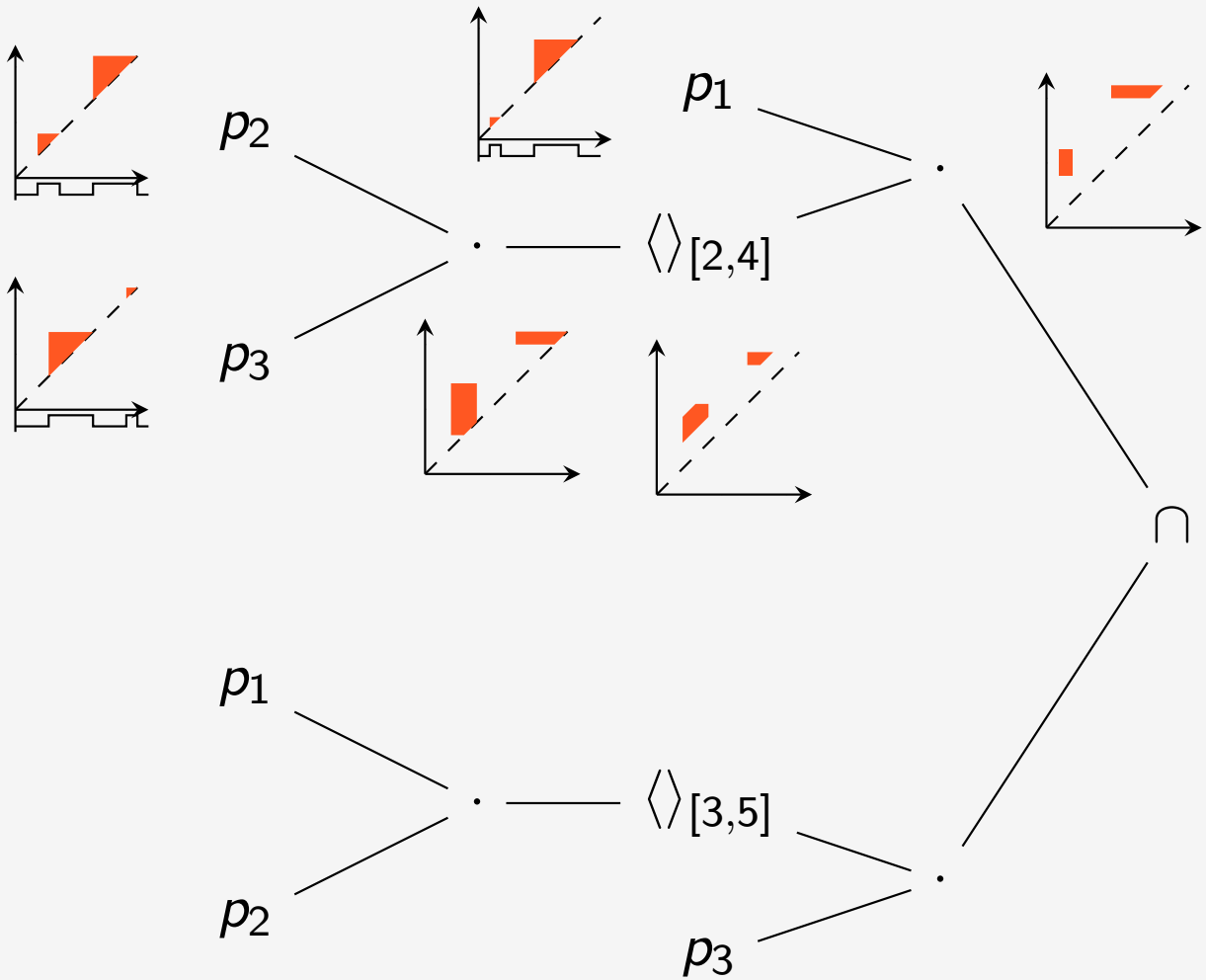
Computing Matches Offline

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



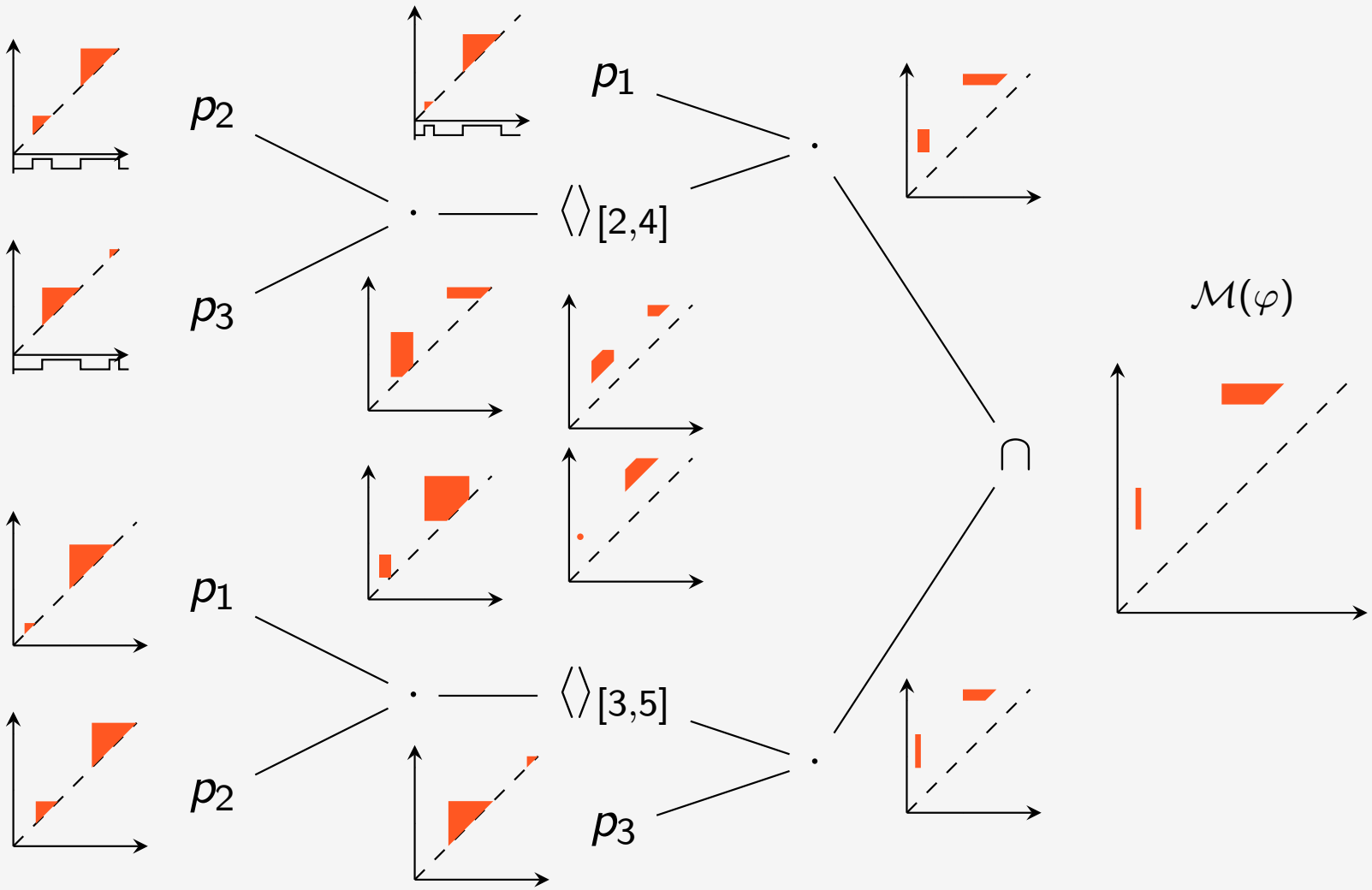
Computing Matches Offline

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

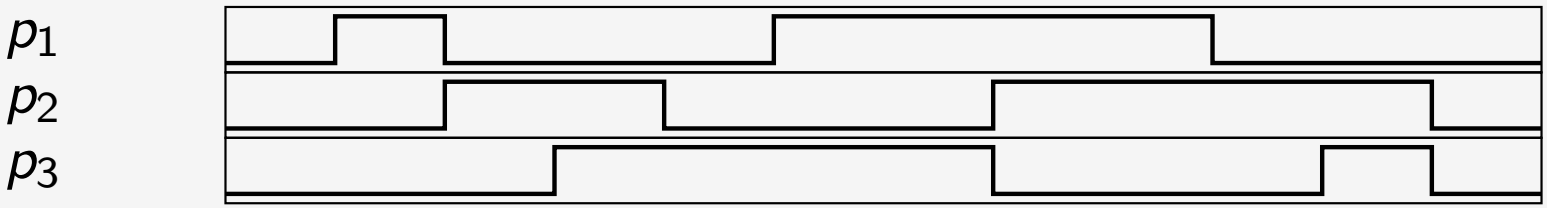


Computing Matches Offline

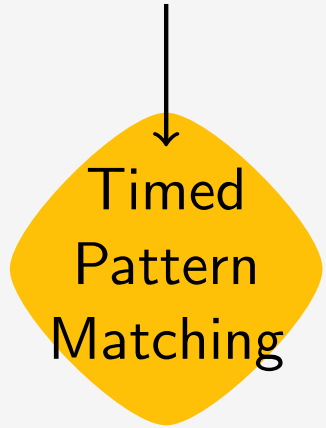
$$\varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3$$

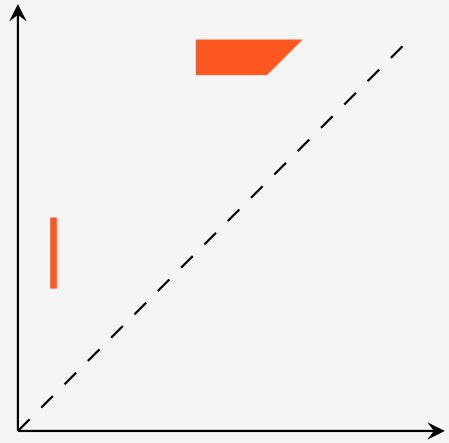


Interpreting Matches

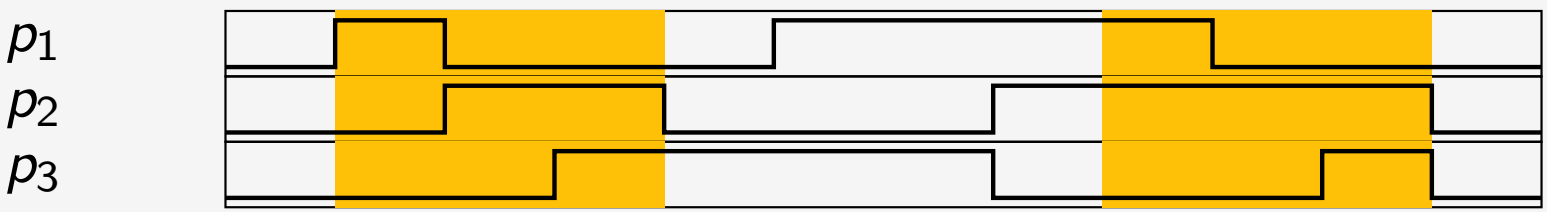


$$\varphi = p_1 \cdot \langle p_2 \cdot p_3 \rangle_{[2,4]} \cap \langle p_1 \cdot p_2 \rangle_{[3,5]} \cdot p_3$$



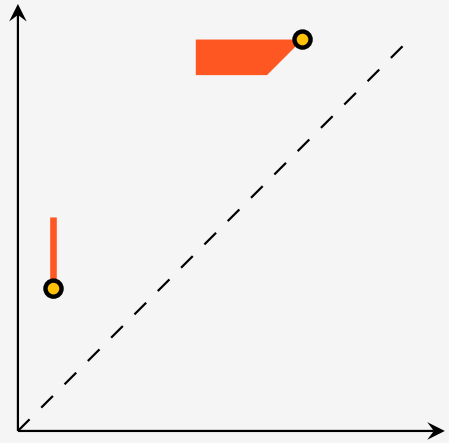


Interpreting Matches



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

→ Timed Pattern Matching →



Going Online for Expressions

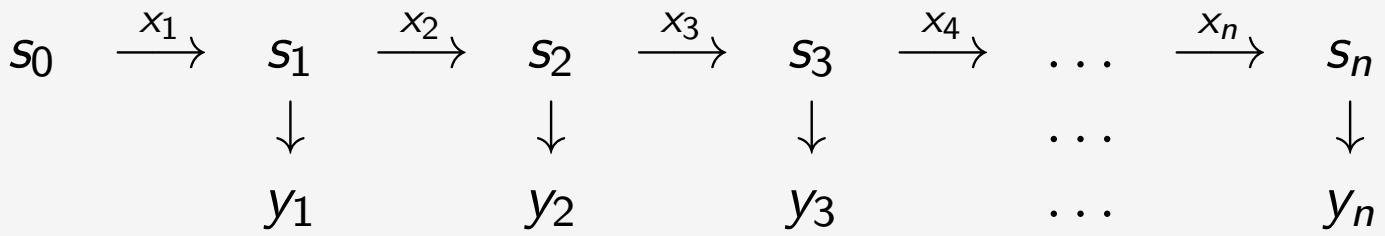
- Goal: Develop online TPM
  - Define derivatives of TRE
  - Compute matches using derivatives

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 $x_1 x_2 x_3 \ldots x_n \to y_1 y_2 y_3 \ldots y_n$, as follows.



- 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:

$$\begin{array}{ccccccccccc} \varphi_0 & \xrightarrow{D_{a_1}} & \varphi_1 & \xrightarrow{D_{a_2}} & \varphi_2 & \xrightarrow{D_{a_3}} & \varphi_3 & \xrightarrow{D_{a_4}} & \varphi_4 & \xrightarrow{D_{a_5}} & \varphi_5 \\ & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ & & 0 & & 1 & & 0 & & 0 & & 1 \end{array}$$

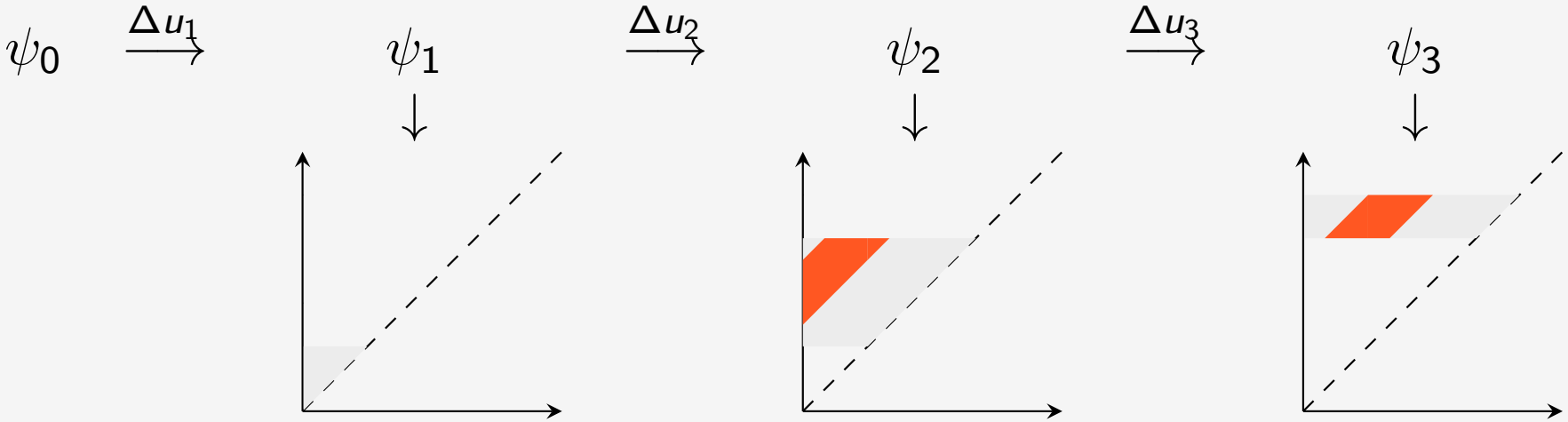
- Output at each step: Acceptance of the word read so far.

Derivatives of Timed Regular Expressions

- 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 $\longrightarrow$ 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.



- 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

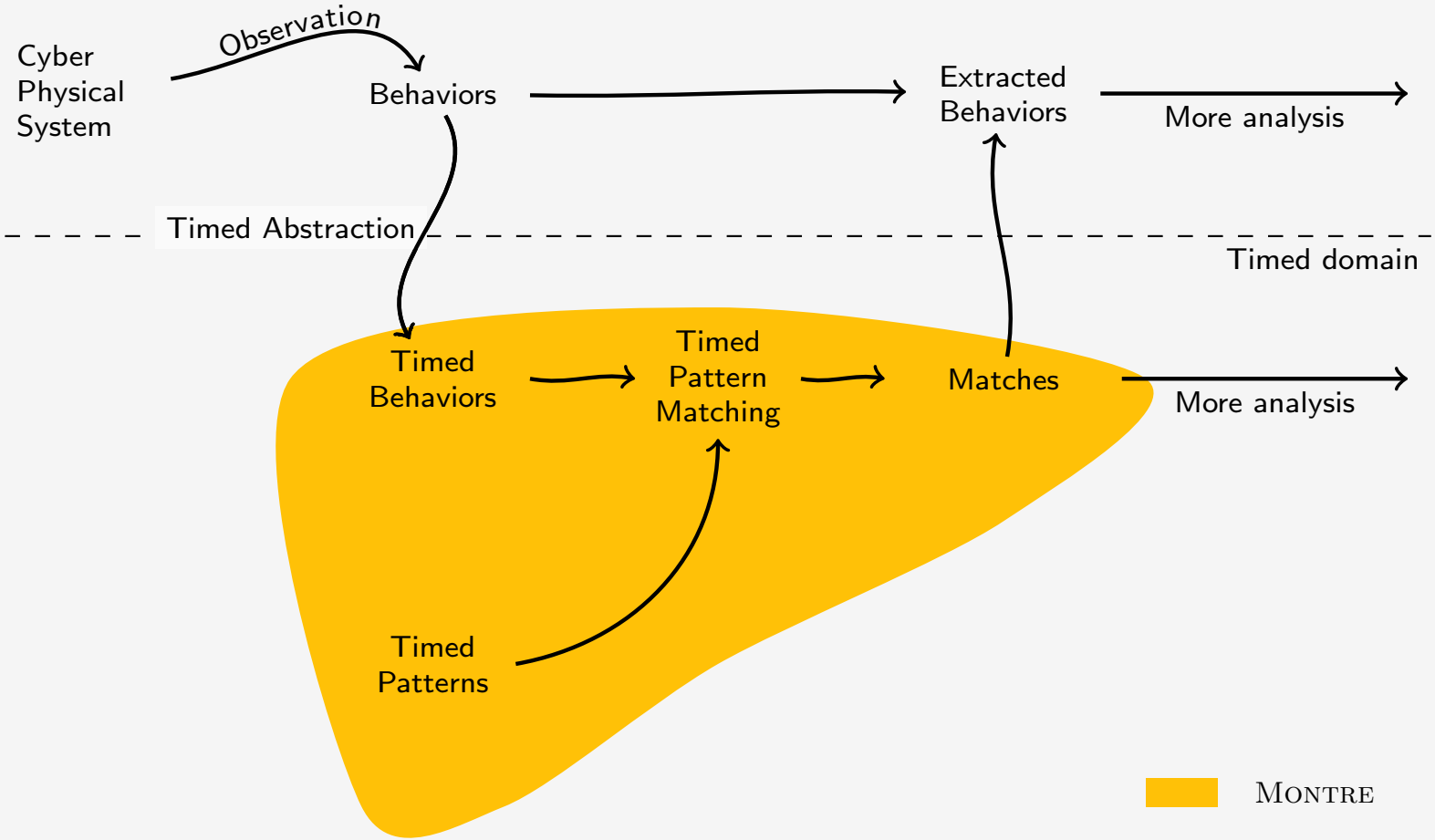
Montre: Tool Description

- 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.



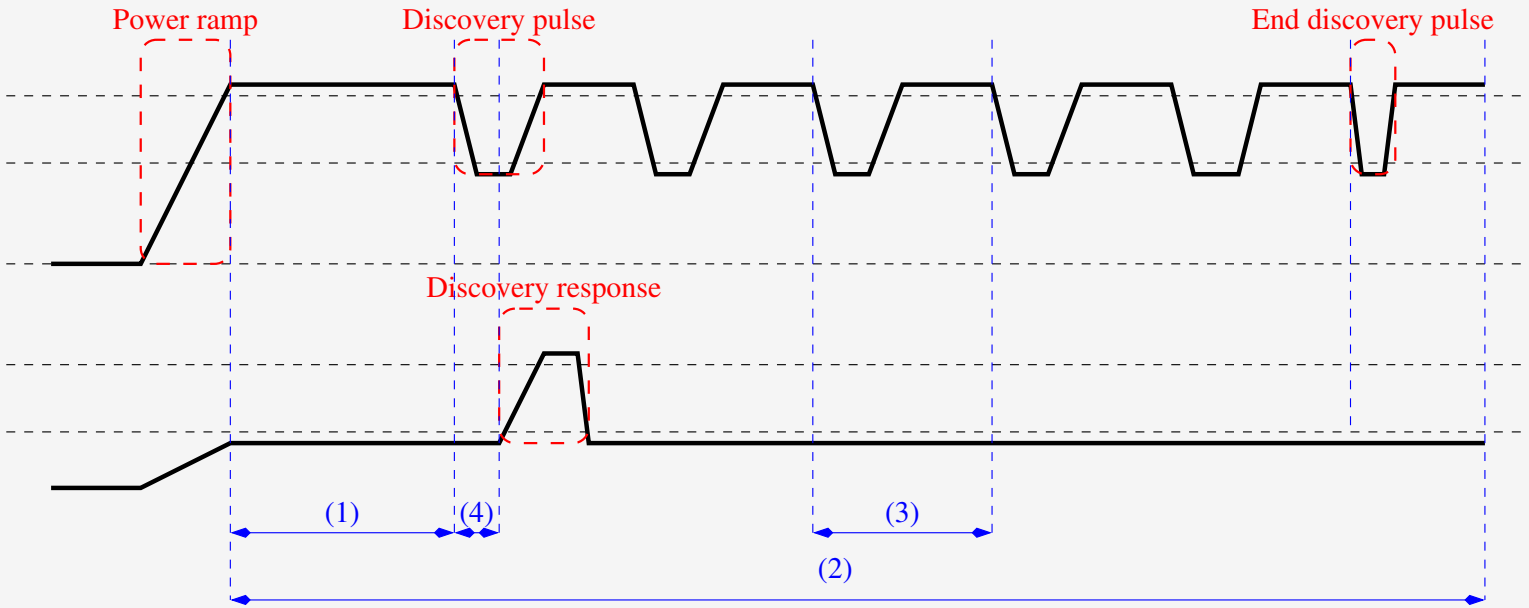
Montre: Performance

Test Patterns	Offline Algorithm Input Size			Online Algorithm Input Size		
	100K	500K	1M	100K	500K	1M
p	0.06/17	0.27/24	0.51/33	6.74/14	29.16/14	57.87/14
$p \cdot q$	0.08/21	0.42/46	0.74/77	8.74/14	42.55/14	81.67/14
$\langle p \cdot q \cdot \langle p \cdot q \cdot p \rangle_I \cdot q \cdot p \rangle_J$	0.23/28	1.09/77	2.14/140	28.07/14	130.96/14	270.45/14
$\langle p \cdot q \rangle_I \cdot r \cap p \cdot \langle q \cdot r \rangle_J$	0.13/23	0.50/51	1.00/86	15.09/15	75.19/15	148.18/15
$p \cdot (q \cdot r)^+$	0.11/20	0.49/37	0.96/60	11.53/15	52.87/15	110.58/15
$\overline{p}$	0.18/12	0.95/45	1.88/92			
$\Diamond_I p$	0.07/16	0.29/65	0.66/163			
$\boxminus_I p$	0.49/23	1.98/100	3.92/163			
$\Diamond_I \Diamond_J p$	0.08/20	0.32/37	0.96/60			
$\Diamond \Diamond (\boxminus p \cdot q)$	0.40/31	1.98/143	3.93/268			
$\Diamond \Diamond (\boxminus p \cdot q) \cap \Diamond_I q$	0.43/38	2.17/179	4.30/304			

- A few seconds for offline, a few hundred seconds for online. (size 1M)
- Linear execution time for typical inputs.
- Constant memory consumption for online algorithm.

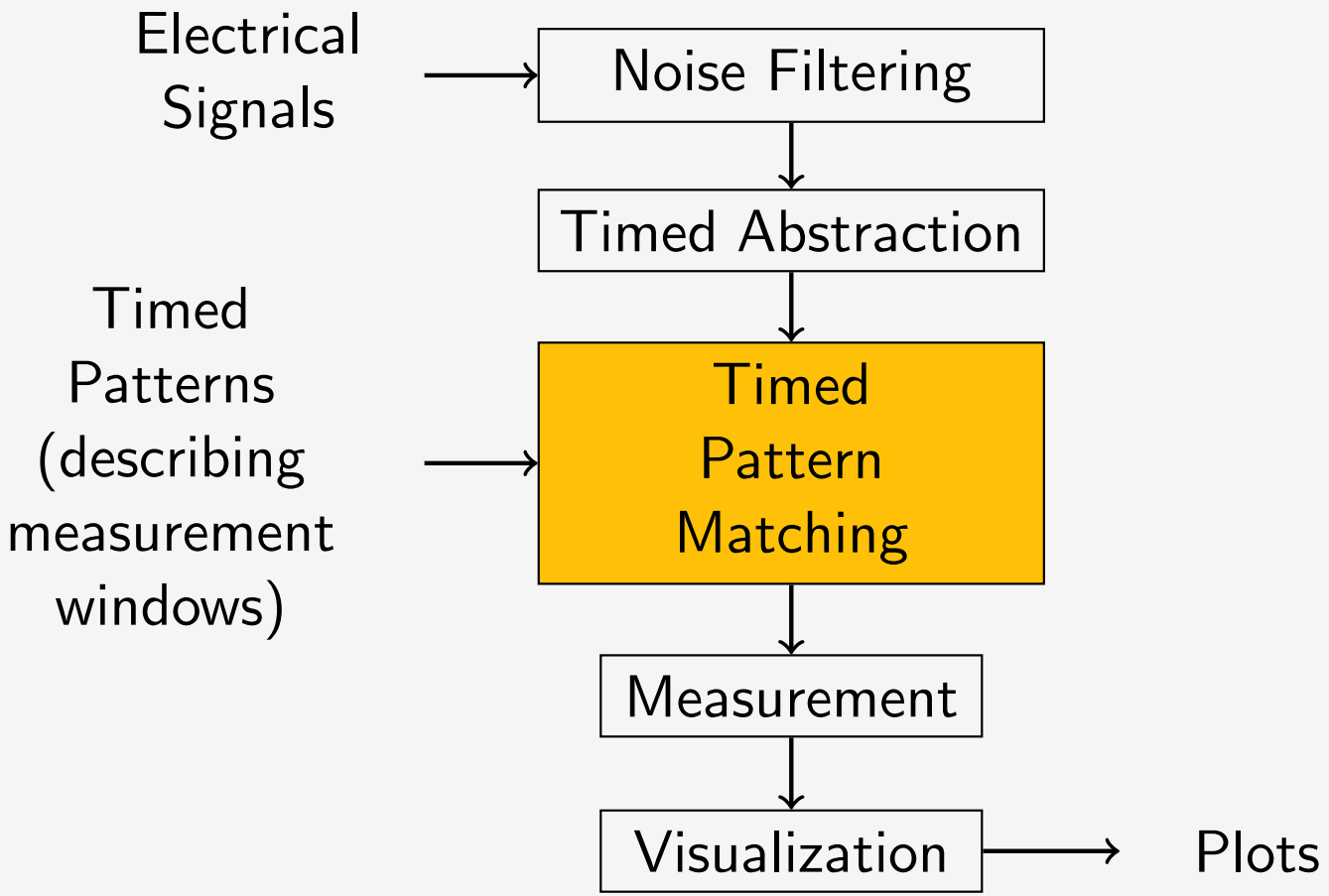
Case Study: DSI3 Protocol

- 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:



Case Study: Matching and Measuring

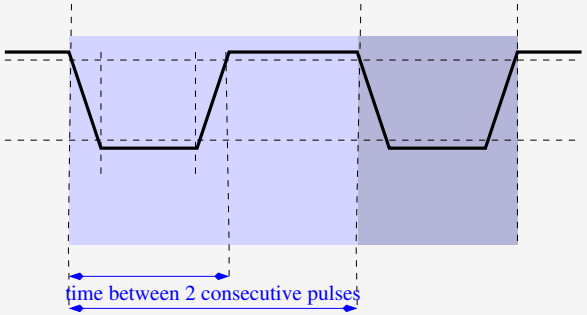
- Specify the pulse pattern:

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

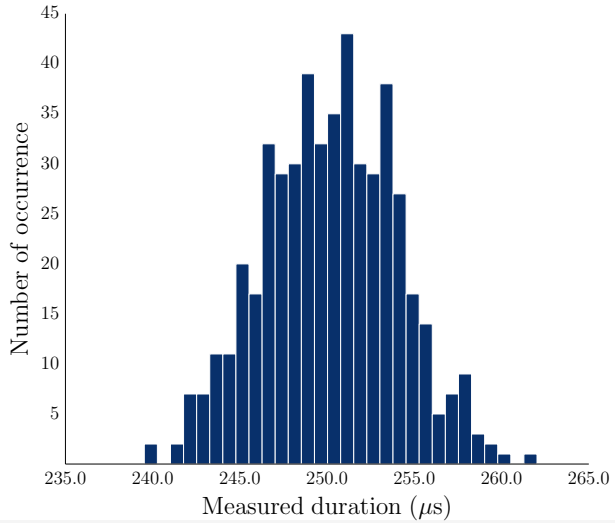
- Specify the tail portion:

$$\varphi_{distpp} = \varphi_{dp} \cdot \texttt{VHigh} \cap \Diamond \varphi_{dp}$$

- Compute the match set $\mathcal{M}(\varphi_{distpp})$ over behaviors of the protocol.
- Measure the duration of each match and plot a histogram.



Label	Voltage
`VHigh`	> 7.8
`VMiddle`	4.92 - 7.8
`VLow`	< 4.92



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

Publications

[1] Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. "Timed Pattern Matching". In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. 2014.

[2] Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. "Online Timed Pattern Matching using Derivatives". In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2016.

[3] Dogan Ulus. "Montre: A Tool for Monitoring Timed Regular Expressions". In: *Computer Aided Verification (CAV)*. 2017.

[4] Dogan Ulus and Oded Maler. "Specifying Timed Patterns using Temporal Logic". In: *Hybrid Systems: Computation and Control (HSCC)*. 2018.

[5] Thomas Ferrère, Oded Maler, Dejan Nickovic, and Dogan Ulus. "Measuring with Timed Patterns". In: *Computer Aided Verification (CAV)*. 2015.

Publications

[6] Alexey Bakhirkin, Thomas Ferrère, Oded Maler, and Dogan Ulus. "On the Quantitative Semantics of Regular Expressions over Real-Valued Signals". In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. 2017.

[7] Eugene Asarin, Oded Maler, Dejan Nickovic, and Dogan Ulus. "Combining the Temporal and Epistemic Dimensions for MTL Monitoring". In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. 2017.

[8] Rajeev Alur, Konstantinos Mamouras, and Dogan Ulus. "Derivatives of Quantitative Regular Expressions". In: *Models, Algorithms, Logics and Tools*. 2017.

[9] Klaus Havelund, Doron Peled, and Dogan Ulus. "First-order temporal logic monitoring with BDDs". In: *Formal Methods in Computer-Aided Design (FMCAD)*. 2017.

[10] Dejan Nickovic, Olivier Lebeltel, Oded Maler, Thomas Ferrère, and Dogan Ulus. "AMT2.0: Qualitative and Quantitative Trace Analysis with Extended Signal Temporal Logic". In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2018.

Thank you for your attention!

Relational Semantics of Timed Regular Expressions

The satisfaction relation $\models$ of a timed regular expression φ 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:

$$
\begin{array}{lll}
(W,t,t') \models p & \leftrightarrow & (t,t') \in V(p) \\
(W,t,t') \models \varphi_1 \cap \varphi_2 & \leftrightarrow & (W,t,t') \models \varphi_1 \text{ and } (W,t,t') \models \varphi_2 \\
(W,t,t') \models \varphi_1 \cdot \varphi_2 & \leftrightarrow & \exists t''.\, (W,t,t'') \models \varphi \text{ and } (W,t'',t') \models \psi \\
(W,t,t') \models \varphi_1 \cup \varphi_2 & \leftrightarrow & (W,t,t') \models \varphi_1 \text{ or } (W,t,t') \models \varphi_2 \\
(W,t,t') \models \varphi^+ & \leftrightarrow & \exists k \geq 1.\, (W,t,t') \models \varphi^k \\
(W,t,t') \models \langle \varphi \rangle_I & \leftrightarrow & (W,t,t') \models \varphi \text{ and } t' - t \in I
\end{array}
$$

Relational Semantics of Metric Compass Logic

The satisfaction relation $\models$ of a metric compass logic formula φ 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:

$$
\begin{array}{lll}
(\mathcal{W}, t, t') \models p & \leftrightarrow & (t, t') \in V(p) \\
(\mathcal{W}, t, t') \models \overline{\varphi} & \leftrightarrow & (\mathcal{W}, t, t') \not\models \varphi \\
(\mathcal{W}, t, t') \models \varphi_1 \cup \varphi_2 & \leftrightarrow & (\mathcal{W}, t, t') \models \varphi_1 \text{ or } (\mathcal{W}, t, t') \models \varphi_2 \\
(\mathcal{W}, t, t') \models \diamondsuit\!\!\downarrow_I \varphi & \leftrightarrow & \exists t'' \in (t, t').\ t' - t'' \in I \text{ and } (\mathcal{W}, t, t'') \models \varphi \\
(\mathcal{W}, t, t') \models \diamondsuit\!\!\uparrow_I \varphi & \leftrightarrow & \exists t'' \in (t', \infty).\ t'' - t' \in I \text{ and } (\mathcal{W}, t, t'') \models \varphi \quad (\\
(\mathcal{W}, t, t') \models \diamondsuit\!\!\rightarrow_I \varphi & \leftrightarrow & \exists t'' \in (t, t').\ t'' - t \in I \text{ and } (\mathcal{W}, t'', t') \models \varphi \\
(\mathcal{W}, t, t') \models \diamondsuit\!\!\leftarrow_I \varphi & \leftrightarrow & \exists t'' \in (-\infty, t).\ t - t'' \in I \text{ and } (\mathcal{W}, t'', t') \models \varphi \quad (\\
(\mathcal{W}, t, t') \models \diamondsuit\!\!\upharpoonright_I \varphi & \leftrightarrow & \exists t'' \in (t', \infty).\ t'' - t' \in I \text{ and } (\mathcal{W}, t', t'') \models \varphi \\
(\mathcal{W}, t, t') \models \diamondsuit\!\!\downharpoonright_I \varphi & \leftrightarrow & \exists t'' \in (-\infty, t).\ t - t'' \in I \text{ and } (\mathcal{W}, t'', t) \models \varphi \quad (
\end{array}
$$

Derivatives of Timed Regular Expressions

Given a (left-reduced) timed regular expression φ and a uniform timed behavior $v : (t, t', a)$, applying the following rules yields an expression ψ such that $[\![\psi]\!] = \Delta_v([\![\varphi]\!])$.

$$
\begin{aligned}
\Delta_v(Z) &= \emptyset \\
\Delta_v(p) &= \begin{cases} Z \cup Z \cdot p & \text{if } a(p) = 1 \text{ where} \\ & \quad Z = \{(r, r') \mid t \le r < r' \le t'\} \\ \emptyset & \text{otherwise} \end{cases} \\
\Delta_v(\psi_1 \cdot \psi_2) &= \Delta_v(\psi_1) \cdot \psi_2 \ \cup \ \mathrm{xt}\big(\psi_1 \cup \Delta_v(\psi_1)\big) \cdot \Delta_v(\psi_2) \\
\Delta_v(\psi_1 \cup \psi_2) &= \Delta_v(\psi_1) \cup \Delta_v(\psi_2) \\
\Delta_v(\psi_1 \cap \psi_2) &= \Delta_v(\psi_1) \cap \Delta_v(\psi_2) \\
\Delta_v(\langle \psi \rangle_I) &= \langle \Delta_v(\psi) \rangle_I \\
\Delta_v(\psi^+) &= \mathrm{xt}(\Delta_v(\psi))^* \cdot \Delta_v(\psi) \cdot \psi^*
\end{aligned}
$$