Timed Regular Expressions Derivatives and Matching

Dogan Ulus[1], **Thomas Ferrère**[1], Eugene Asarin[2], and Oded Maler[1]

[1] VERIMAG, Université Grenoble-Alpes & CNRS, France
[2] IRIF, Université Paris Diderot, France

May 10, 2016

EQINOCS Workshop

Regular Expressions

Formal Languages

- Alphabet: set of actions $\Sigma = \{a, b, c, \ldots, z\}$
- Word: sequence $w = a_1 a_2 a_3 \ldots a_n$ where $a_i \in \Sigma$

Regular Expressions

- Syntax:

$$\varphi := \varnothing \mid \epsilon \mid a \mid \varphi \cdot \varphi \mid \varphi \vee \varphi \mid \varphi^*$$

- Semantics:

$$[\![\varnothing]\!]_w = \emptyset \qquad [\![\epsilon]\!]_w = \{(i,i) : 0 \leq i \leq n\}$$

$$[\![a]\!]_w = \{(i-1,i) : w[i-1,i] = a\}$$

$$[\![\varphi \cdot \psi]\!]_w = \{(i,j) : \exists k,\ (i,k) \in [\![\varphi]\!]_w \wedge (k,j) \in [\![\psi]\!]_w\}$$

$$[\![\varphi \vee \psi]\!]_w = [\![\varphi]\!]_w \cup [\![\psi]\!]_w \qquad [\![\varphi^*]\!]_w = \bigcup_{i \geq 0} [\![\varphi^i]\!]_w$$

Pattern Matching

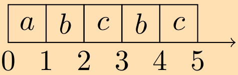
Problem (Matching)

Given $w = a_1 a_2 a_3 \dots a_n$ *and expression* φ*, compute* $[\![\varphi]\!]_w$.

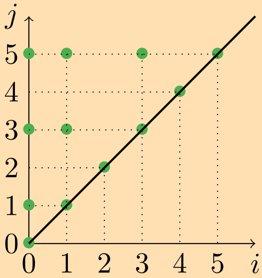
Example

$\varphi = a^*(bc)^*$

$w :$



$[\![\varphi]\!]_w :$



Timed Languages

Timed words $w = (t_1, a_1)(t_2, a_2) \dots (t_n, a_n)$ with $0 \leq t_1 \leq t_2 \leq \dots \leq t_n$.

Two alternative interpretations:

- Event sequences
  - interpret (t, a) as "a occurs at time t"
  - notation: $w = r_1 a_1 r_2 a_2 \dots r_n a_n$ (with $r_i = t_i - t_{i-1}$, $t_0 = 0$)
- Continuous Signals
  - interpret (t, a) as "a holds up to time t"
  - notation: $w = a_1^{r_1} a_2^{r_2} \dots a_n^{r_n}$ (with $r_i = t_i - t_{i-1}$, $t_0 = 0$)
  - **closure under stuttering**: $a^r \cdot a^s = a^{r+s}$ and $a^0 = \epsilon$

Timed Regular Expressions

- New operator $\langle\,\rangle_I$ requiring duration in interval I
- Atomic expressions a now denote $r \cdot a$ (resp. a^r) for abritrary r
- Syntax:

$$\varphi := \varnothing \mid \epsilon \mid a \mid \varphi \cdot \varphi \mid \varphi \vee \varphi \mid \varphi^* \mid \langle\varphi\rangle_I$$

- Semantics:

$$\llbracket \varnothing \rrbracket_w = \emptyset \qquad \llbracket \epsilon \rrbracket_w = \{(t,t) : 0 \leq t \leq |w|\}$$
$$\llbracket a \rrbracket_w = \{(t,t') : w[t,t'] = a\}$$
$$\llbracket \varphi \cdot \psi \rrbracket_w = \{(t,t') : \exists t'',\ (t,t'') \in \llbracket \varphi \rrbracket_w \wedge (t'',t') \in \llbracket \psi \rrbracket_w\}$$
$$\llbracket \varphi \vee \psi \rrbracket_w = \llbracket \varphi \rrbracket_w \cup \llbracket \psi \rrbracket_w \qquad \llbracket \varphi^* \rrbracket_w = \bigcup_{i \geq 0} \llbracket \varphi^i \rrbracket_w$$
$$\llbracket \langle\varphi\rangle_I \rrbracket_w = \{(t,t') \in \llbracket \varphi \rrbracket_w : t' - t \in I\}$$

Timed Regular Expressions

- New operator $\langle\,\rangle_I$ requiring duration in interval I
- Atomic expressions a now denote $r \cdot a$ (resp. a^r) for abritrary r
- Syntax:

$$\varphi := \varnothing \mid \epsilon \mid a \mid \varphi \cdot \varphi \mid \varphi \vee \varphi \mid \varphi^* \mid \langle \varphi \rangle_I$$

- Semantics:

$$\llbracket \varnothing \rrbracket_w = \emptyset \qquad \llbracket \epsilon \rrbracket_w = \{(t,t) : 0 \leq t \leq |w|\}$$
$$\llbracket a \rrbracket_w = \{(t,t') : w[t,t'] = a\}$$
$$\llbracket \varphi \cdot \psi \rrbracket_w = \llbracket \varphi \rrbracket_w \circ \llbracket \psi \rrbracket_w$$
$$\llbracket \varphi \vee \psi \rrbracket_w = \llbracket \varphi \rrbracket_w \cup \llbracket \psi \rrbracket_w \qquad \llbracket \varphi^* \rrbracket_w = \bigcup_{i \geq 0} \llbracket \varphi^i \rrbracket_w$$
$$\llbracket \langle \varphi \rangle_I \rrbracket_w = \{(t,t') \in \llbracket \varphi \rrbracket_w : t' - t \in I\}$$

Atomic Events and Atomic Signals

Let $w = (t_1, a_1)(t_2, a_2) \dots (t_n, a_n)$.

Definition (sub-sequence)

$w[t, t'] = a$ if and only if $\exists i$, $t = t_{i-1} \leq t' = t_i$, and $a_i = a$

Definition (sub-signal)

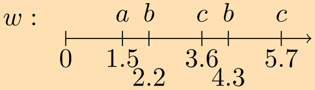
$w[t, t'] = a$ if and only if $\exists i$, $t_{i-1} \leq t < t' \leq t_i$, and $a_i = a$ assuming w stutter-free.

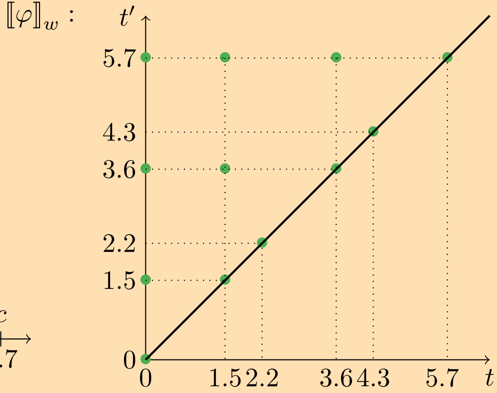
- In event sequences $w[t, t']$ only defined if some events occur at times t and t'
- In continuous signals $w[t, t']$ defined everywhere

Timed Pattern Matching (Event Sequences)

Example

$\varphi = a^*(bc)^*$



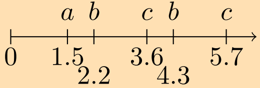


Timed Pattern Matching (Event Sequences)

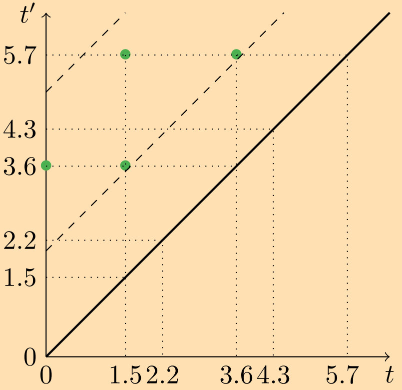
Example

$\varphi = \langle a^*(bc)^* \rangle_{[2,5]}$

$w :$



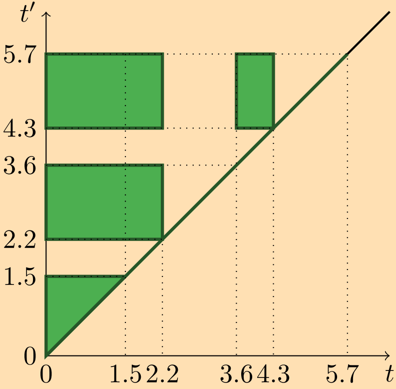
$[\![\varphi]\!]_w :$



Timed Pattern Matching (Continuous Signals)

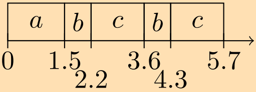
Example

$\llbracket \varphi \rrbracket_w :$



$$\varphi = a^*(bc)^*$$

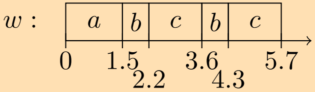
$w :$

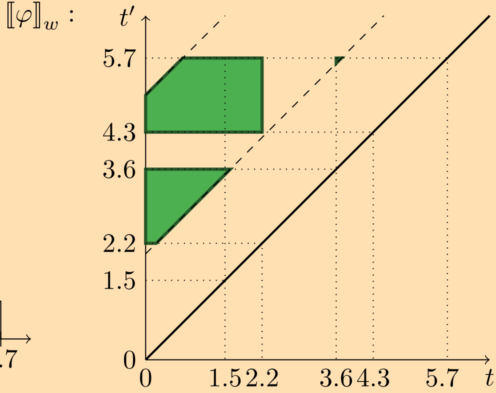


Timed Pattern Matching (Continuous Signals)

Example

$$\varphi = \langle a^*(bc)^* \rangle_{[2,5]}$$





Offline Matching Continuous Signals

Theorem (FORMATS'14)

The set of matches $[\![\varphi]\!]_w$ *is computable as a finite union of 2d zones.*

Proof.

Structural induction over φ:

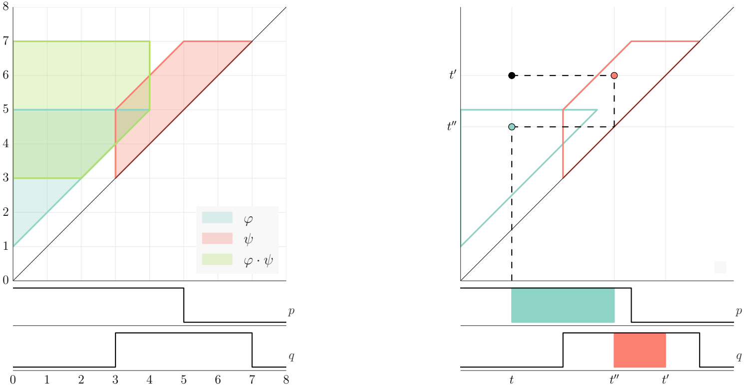
- a: triangular zones for every segment
- $\varphi \vee \psi$: simple union
- $\varphi \cdot \psi$: composition of binary relations
- φ^*: closure is obtained in $2(|w| + \|w\|) + 1$ iterations
- $\langle \varphi \rangle_I$: intersection of zones with diagonal band of $\{(t, t') : t' - t \in I\}$

□

Offline Matching Continuous Signals (Concatenation)

Example

Concatenation of $\varphi = \langle p \rangle_{[1,\infty]}$ with $\psi = \langle q \rangle_{[0,2]}$



Derivatives of Regular Expressions

Nullable check

$$\nu(\varnothing) = \varnothing \qquad \nu(\varphi_1 \cdot \varphi_2) = \nu(\varphi_1) \cdot \nu(\varphi_2)$$
$$\nu(\epsilon) = \epsilon \qquad \nu(\varphi_1 \vee \varphi_2) = \nu(\varphi_1) \vee \nu(\varphi_2)$$
$$\nu(a) = \varnothing \qquad \nu(\varphi^*) = \epsilon$$

Derivatives

$$\mathrm{d}_a(\varnothing) = \varnothing \qquad \mathrm{d}_a(\varphi_1 \cdot \varphi_2) = \mathrm{d}_a(\varphi_1) \cdot \varphi_2 \vee \nu(\varphi_1) \cdot \mathrm{d}_a(\varphi_2)$$
$$\mathrm{d}_a(\epsilon) = \varnothing \qquad \mathrm{d}_a(\varphi_1 \vee \varphi_2) = \mathrm{d}_a(\varphi_1) \vee \mathrm{d}_a(\varphi_2)$$
$$\mathrm{d}_a(a) = \epsilon \qquad \mathrm{d}_a(\varphi^*) = \mathrm{d}_a(\varphi) \cdot \varphi^*$$
$$\mathrm{d}_a(b) = \varnothing$$

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols	a	b	c	b	c
Positions	1	2	3	4	5

$1 \quad \varphi$

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols	a	b	c	b	c
Positions	1	2	3	4	5

1 $\quad \varphi \xrightarrow[\mathrm{d}_a]{} a^*(bc)^*$

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols	a	b	c	b	c
Positions	1	2	3	4	5
1 $\quad \varphi \xrightarrow[\mathrm{d}_a]{}$	$a^*(bc)^*$ $\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$			

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols	a	b	c	b	c
Positions	1	2	3	4	5

1 $\quad \varphi \xrightarrow[d_a]{} a^*(bc)^* \xrightarrow[d_b]{} c(bc)^* \xrightarrow[d_c]{} (bc)^*$

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols	a	b	c	b	c
Positions	1	2	3	4	5

1 $\quad \varphi \xrightarrow[\mathrm{d}_a]{} a^*(bc)^* \xrightarrow[\mathrm{d}_b]{} c(bc)^* \xrightarrow[\mathrm{d}_c]{} (bc)^* \xrightarrow[\mathrm{d}_b]{} c(bc)^*$

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols			a		b		c		b		c
Positions			1		2		3		4		5
1	φ	$\xrightarrow[\mathrm{d}_a]{}$	$a^*(bc)^*$	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$

Matching using Derivatives

Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

Symbols			a		b		c		b		c
Positions			1		2		3		4		5
1	φ	$\xrightarrow[\mathrm{d}_a]{}$	$a^*(bc)^*$	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$
2			φ	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$
3					φ	$\xrightarrow[\mathrm{d}_c]{}$	$\varnothing$	$\xrightarrow[\mathrm{d}_b]{}$	$\varnothing$	$\xrightarrow[\mathrm{d}_c]{}$	$\varnothing$
4							φ	$\xrightarrow[\mathrm{d}_b]{}$	$c(bc)^*$	$\xrightarrow[\mathrm{d}_c]{}$	$(bc)^*$
5									φ	$\xrightarrow[\mathrm{d}_c]{}$	$\varnothing$

Derivatives of Timed Regular Expressions (Event Sequences)

- Nullable check: $\nu(\langle\varphi\rangle_I) = \begin{cases} \epsilon & \text{if } 0 \in I \text{ and } \nu(\varphi) \\ \varnothing & \text{otherwise} \end{cases}$

- Derivative relative to events: $\mathrm{d}_a(\langle\varphi\rangle_I) = \langle \mathrm{d}_a(\varphi)\rangle_I$
- Derivative relative to time: $\mathrm{d}_r(a) = a$ and $\mathrm{d}_r(\langle\varphi\rangle_I) = \langle \mathrm{d}_r(\varphi)\rangle_{I \ominus r}$

- The set of derivatives over some event sequence is finite

Derivatives of Timed Regular Expressions (Continuous Signals)

- Problem: there can be an unbounded number of segments in some atomic sub-signal a^r for $r > 0$
- In general $\mathrm{d}_{a^r}(\varphi_1 \cdot \varphi_2) \neq \mathrm{d}_{a^r}(\varphi_1) \cdot \varphi_2 \vee \nu(\varphi_1) \cdot \mathrm{d}_{a^r}(\varphi_2)$
- We also have terms $\mathrm{d}_{a^{r-s}}(\varphi_2)$ for every $0 < s < r$ such that $a^s \in \varphi_1$

- The set of derivatives over some event sequence is typically uncountable

Partial Derivatives of Timed Regular Expressions (Continuous Signals)

- Pratial derivatives are sets of expressions
- Property: $uv \in \varphi$ if and only if there exists $\varphi' \in \partial_u(\varphi)$ such that $v \in \varphi'$
- Intuitively (total) derivatives $\sim$ states in a DFA acceptor; partial derivatives $\sim$ states in a NFA acceptor

Derivatives

$$\partial_{a^r}(\varnothing) = \varnothing \qquad \partial_{a^r}(\varphi_1 \cdot \varphi_2) = \partial_{a^r}(\varphi_1) \cdot \varphi_2 \cup \bigcup_{\substack{a^{r-s} \in \varphi_1 \\ 0 \leq s \leq r}} \partial_{a^s}(\varphi_2)$$

$$\partial_{a^r}(\epsilon) = \varnothing \qquad \partial_{a^r}(\varphi_1 \vee \varphi_2) = \partial_{a^r}(\varphi_1) \cup \partial_{a^r}(\varphi_2)$$

$$\partial_{a^r}(a) = \{a, \epsilon\} \qquad \partial_{a^r}(\varphi^*) = \partial_{a^r}(\varphi) \cdot \varphi^* \text{ if } r \leq d$$

$$\partial_{a^r}(b) = \varnothing \qquad \partial_{a^r}(\langle\varphi\rangle_I) = \langle\partial_{a^r}(\varphi)\rangle_{I \ominus r}$$

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

	a	b	c	b	c
	1	2	3	4	5
1 φ					

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

	a	b	c	b	c
	1	2	3	4	5
1 $\varphi \xrightarrow[\delta_a]{}$	$[0,1](bc)^*$				

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

	a	b	c	b	c
	1	2	3	4	5
1 φ $\xrightarrow[\delta_a]{}$	$[0,1](bc)^*$ $\xrightarrow[\delta_b]{}$	$[0,2]c(bc)^*$			

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

		a		b		c	b	c
		1		2		3	4	5
1 φ	$\xrightarrow[\delta_a]{}$	$[0,1](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,2]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[0,3](bc)^*$		

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

		a		b		c		b		c
		1		2		3		4		5
1 φ	$\xrightarrow[\delta_a]{}$	$[0,1](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,2]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[0,3](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,4]c(bc)^*$		

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

			a		b		c		b		c
			1		2		3		4		5
1	φ	$\xrightarrow[\delta_a]{}$	$[0,1](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,2]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[0,3](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,4]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[0,5](bc)^*$

Reduction of Regular Expressions

Instead of replacing letter by ϵ we replace them by matched intervals. Example: $\varphi = a^*(bc)^*$ and $w = abcbc$.

			a		b		c		b		c
			1		2		3		4		5
1	φ	$\xrightarrow[\delta_a]{}$	$[0,1](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,2]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[0,3](bc)^*$	$\xrightarrow[\delta_b]{}$	$[0,4]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[0,5](bc)^*$
2			φ	$\xrightarrow[\delta_b]{}$	$[1,2]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[1,3](bc)^*$	$\xrightarrow[\delta_b]{}$	$[1,4]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[1,5](bc)^*$
3					φ	$\xrightarrow[\delta_c]{}$	$\varnothing$	$\xrightarrow[\delta_b]{}$	$\varnothing$	$\xrightarrow[\delta_c]{}$	$\varnothing$
4							φ	$\xrightarrow[\delta_b]{}$	$[3,4]c(bc)^*$	$\xrightarrow[\delta_c]{}$	$[3,5](bc)^*$
5									φ	$\xrightarrow[\delta_c]{}$	$\varnothing$

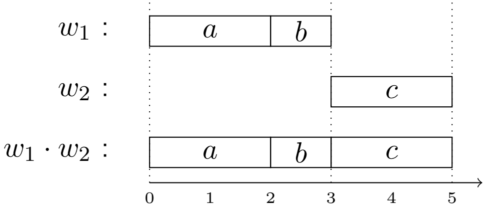
Signals with Absolute Time

- A signal is a piecewise-constant function over Σ:

$$w : [t, t') \to \Sigma$$

with domain $\mathrm{dom}(w) = [t, t')$.

- The concatenation $w_1 \cdot w_2$ is defined only if w_1 meets w_2:



Extended Signals

- The special symbol $\checkmark$ extends the alphabet Σ.
- $\Sigma_{\checkmark} = \Sigma \cup \{\checkmark\}$
- An *extended* signal w if $w \in \Sigma_{\checkmark}^{(*)}$.
- Some classes of extended signals: **pure** signals in $\Sigma^{(*)}$, **reduced** signals in $\checkmark^{(*)}$, and **left-reduced** signals in $\checkmark^{(*)} \cdot \Sigma^{(*)}$

Intuition

- Pure - Original Specification
- Reduced - Fully Checkmarked Specification
- Left-reduced - Partially checkmarked Specification

Extended Timed Regular Expressions

- Syntax:

$$\varphi := \varnothing \mid \epsilon \mid a \mid \checkmark \mid \varphi_1 \cdot \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi^* \mid {}^K_J\langle\varphi\rangle_I$$

where I, J, K are intervals of $\mathbb{R}_{\geq 0}$.

- Semantics:

$$\begin{aligned} &\ldots \\ [\![\checkmark]\!]_w &= \{(t,t') : 0 \leq t \leq t' \leq t_n\} \\ &\ldots \\ [\![{}^K_J\langle\varphi\rangle_I]\!]_w &= \{(t,t') \in [\![\varphi]\!]_w : t \in J,\, t' \in K,\, t' - t \in I\} \end{aligned}$$

Left Reduction

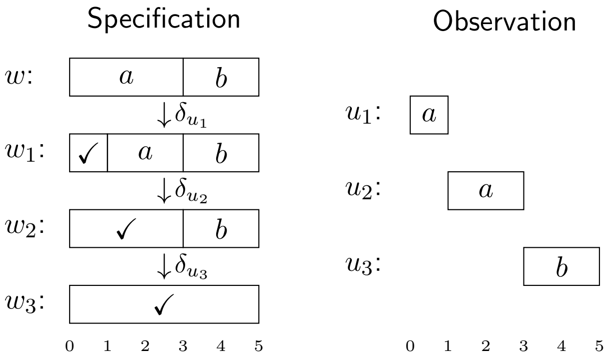


Figure: A left reduction example.

Left Reduction

- We introduce left reduction as a position (and duration) preserving derivative operation.

Definition (Left Reduction)

The left reduction of a language $\mathcal{L}$ with respect to u is:

$$\delta_u(\mathcal{L}) = \left\{ \begin{array}{lr} \sigma\tau v \;:\; & \sigma u v \in \mathcal{L},\ \sigma, \tau \in \checkmark^{(*)},\ v \in \Sigma^{(*)}, \\ & \text{and } \mathrm{dom}(\tau) = \mathrm{dom}(u) \end{array} \right\}$$

Language Membership

$$u \in \mathcal{L} \quad \text{iff} \quad \delta_u(\mathcal{L}) \cap \checkmark^{(*)} \neq \emptyset$$

Timed Regular Expressions Reduction

Theorem (TACAS'16)

$$\delta_v(\varnothing) = \varnothing$$
$$\delta_v(\epsilon) = \varnothing$$
$$\delta_v(\checkmark) = \varnothing$$
$$\delta_v(a) = \begin{cases} {}^{[t,t']}_{[t,t']}\langle \checkmark \rangle_{[0,t'-t]} \cdot (\epsilon \vee a) & \textit{if } v \in a^{(*)} \\ \varnothing & \textit{otherwise} \end{cases}$$
$$\delta_v(\psi_1 \cdot \psi_2) = \delta_v(\psi_1) \cdot \psi_2 \;\vee\; \mu(\psi_1 \vee \delta_v(\psi_1)) \cdot \delta_v(\psi_2)$$
$$\delta_v(\psi_1 \vee \psi_2) = \delta_v(\psi_1) \vee \delta_v(\psi_2)$$
$$\delta_v({}^K_J\langle \psi \rangle_I) = {}^K_J\langle \delta_v(\psi) \rangle_I$$
$$\delta_v(\psi^*) = \mu(\delta_v(\psi))^* \cdot \delta_v(\psi) \cdot \psi^*$$

where $\mu(\varphi)$ is such that $\mu(\varphi) = \varphi \cap \checkmark^{(*)}$

Online Timed Pattern Matching

Inputs/Outputs:

- Input φ a timed regular expression;
- Input $w = u_1 u_2 \ldots u_n$ incrementally;
- Output set of matches ending in j^{th} segment at each step.

Procedure:

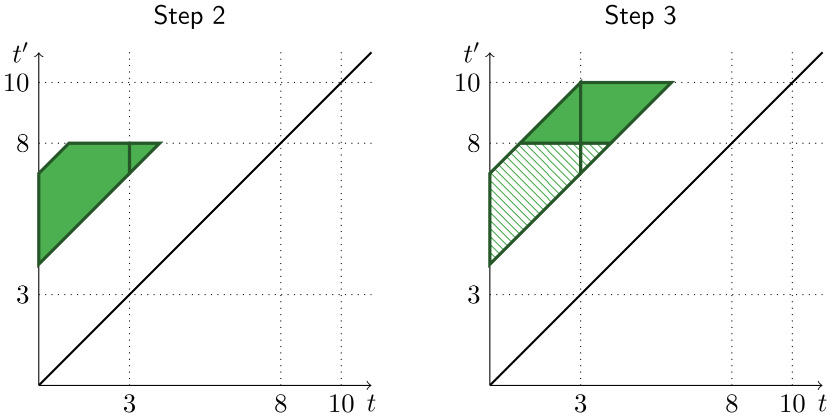
- Extract φ to see if the empty word is a match
- For $1 \leq j \leq n$ repeat:
  - Derive previous state relative to u_j and add new expression $\delta_{u_j}(\varphi)$ to the current state
  - Extract matches ending in j^{th} segment from the state

Example

Symbols Segments		$p\bar{q}$ $[0,3)$	pq $[3,8)$	$\bar{p}q$ $[8,10)$
$[0,3)$	$\langle p \cdot q \rangle_I$	$\xrightarrow[\Delta_{w_1}]{}$ $\langle \Gamma_1 \cdot q \rangle_I \vee \langle \Gamma_1 \cdot p \cdot q \rangle_I$	$\xrightarrow[\Delta_{w_2}]{}$ $\langle \Gamma_1 \cdot \Gamma_2 \rangle_I \vee \langle \Gamma_1 \cdot \Gamma_2 \cdot q \rangle_I \vee \langle \Gamma_1 \cdot \Gamma_2 \cdot p \cdot q \rangle_I$	$\xrightarrow[\Delta_{w_3}]{}$ $\langle \Gamma_1 \cdot \Gamma_2 \cdot \Gamma_3 \rangle_I \vee \langle \Gamma_1 \cdot \Gamma_2 \cdot \Gamma_3 \cdot q \rangle_I$
$[3,8)$		$\langle p \cdot q \rangle_I$	$\xrightarrow[\Delta_{w_2}]{}$ $\langle \Gamma_2 \rangle_I \vee \langle \Gamma_2 \cdot q \rangle_I \vee \langle \Gamma_2 \cdot p \cdot q \rangle_I$	$\xrightarrow[\Delta_{w_3}]{}$ $\langle \Gamma_2 \cdot \Gamma_3 \rangle_I \vee \langle \Gamma_2 \cdot \Gamma_3 \cdot q \rangle_I$
$[8,10)$			$\langle p \cdot q \rangle_I$	$\xrightarrow[\Delta_{w_3}]{}$ $\varnothing$

- $\Gamma_1 = {}^{[0,3]}_{[0,3]}\langle \checkmark \rangle_{[0,3]}$
- $\Gamma_2 = {}^{[3,8]}_{[3,8]}\langle \checkmark \rangle_{[0,5]}$
- $\Gamma_3 = {}^{[8,10]}_{[8,10]}\langle \checkmark \rangle_{[0,2]}$

Example



- At each step, report segments satisfying the expression

Online vs Offline Performance

	Offline Algorithm Input Size			Online Algorithm Input Size		
Test Patterns	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
$p \cdot (q \cdot r)^*$	0.11/20	0.49/37	0.96/60	11.53/15	52.87/15	110.58/15

- Execution times/memory usage (in seconds/megabytes).
- Both are linear for typical inputs.
- Online is 100 times slower but memory usage is constant.
- These numbers are sufficient for many applications.

Discussion

- There exists an inductive characterization of the match set of a timed regular expression in terms of zones
- An offline monitoring procedure has been implemented
- Several approaches to quotient timed regular expressions by timed words (online monitoring)
  1. Derivatives
  2. Partial Derivatives
  3. Rewriting
- An online matching procedure (based on 3.) has been implemented
- The procedure consumes a constant segment from the input signal and reports a set of matches ending in that segment.
- Applications: Runtime verification, robotics, medical monitoring, ...