On Timed Languages

Eugene Asarin

based on joint work with Paul Caspi, Oded Maler, Cătălin Dima, Matthieu Moy

VERIMAG

The Problem

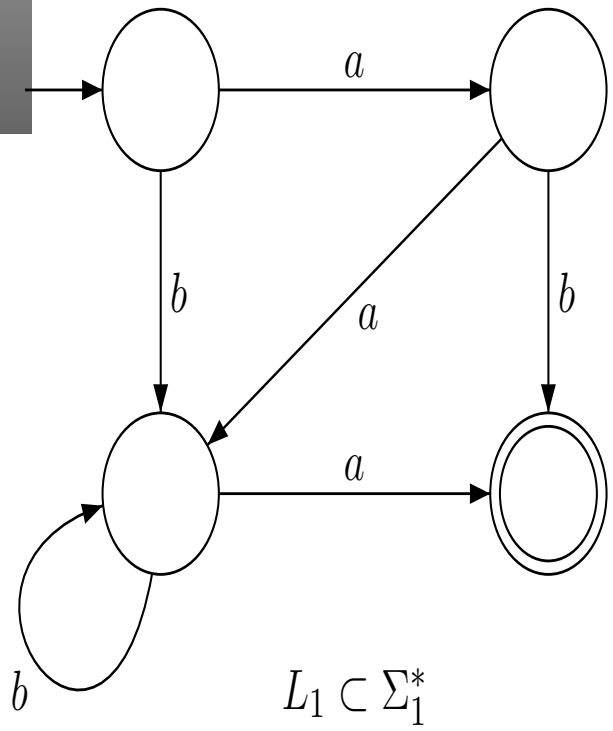
Build a good (natural, simple, and usable) theory of timed languages

Verimag

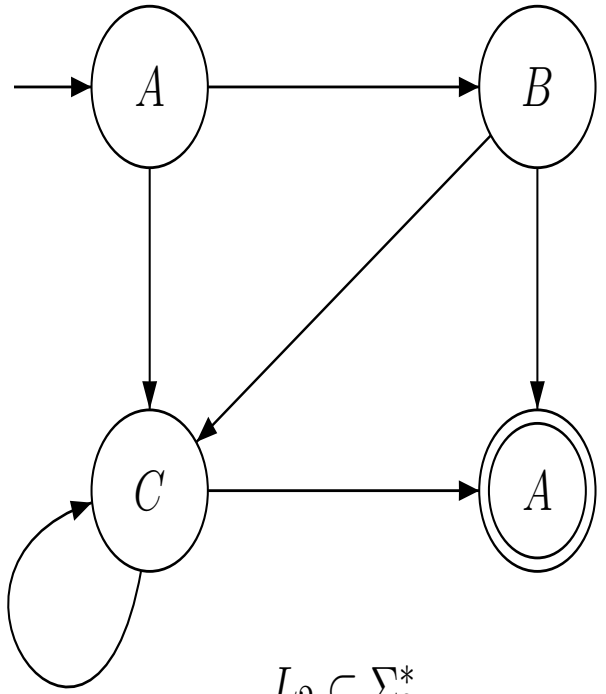
Applying classical recipes

Verimag

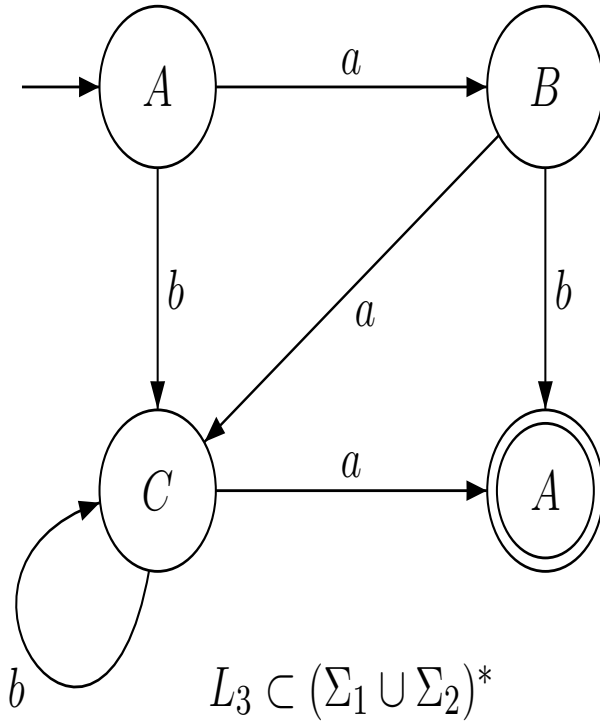
Untimed case: one type of monoids, one class of languages



$L_1 \subset \Sigma_1^*$



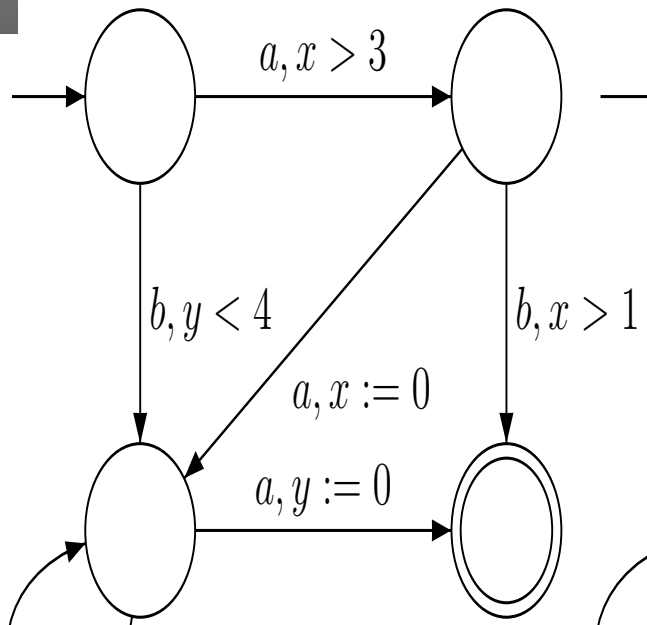
$L_2 \subset \Sigma_2^*$



$L_3 \subset (\Sigma_1 \cup \Sigma_2)^*$

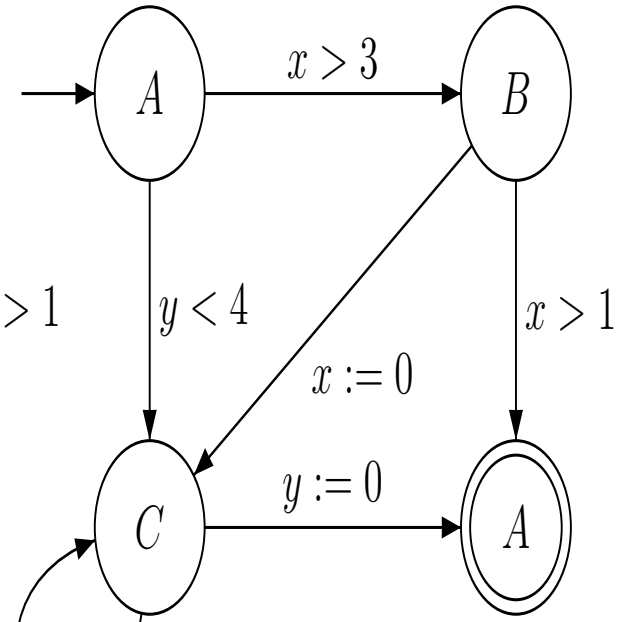
Timed case: 3 types of monoids

Time-event sequences



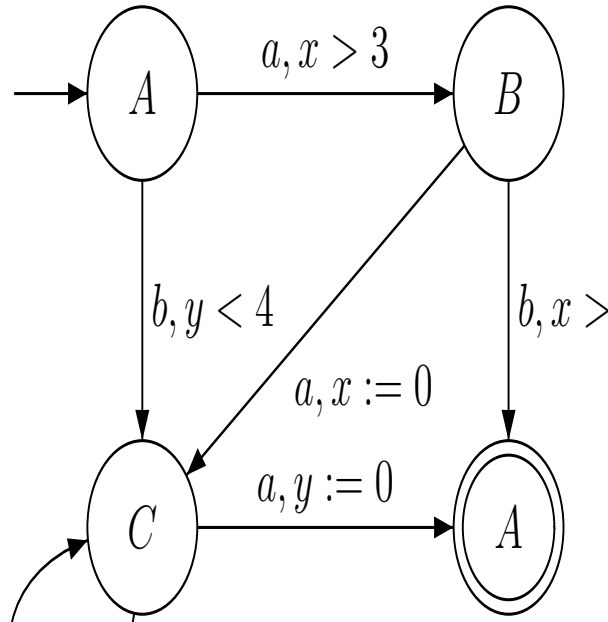
$L_1 \subset \{a, b, c\}^* \oplus \mathbb{R}_{\geq 0}$

Signals



$L_2 \subset \mathbb{R}_{\geq 0 A} \oplus \mathbb{R}_{\geq 0 B} \oplus \mathbb{R}_{\geq 0 C}$

"Sigacts"



$L_3 \subset \{a, b, c\}^* \oplus \mathbb{R}_{\geq 0 A} \oplus \mathbb{R}_{\geq 0 B} \oplus \mathbb{R}_{\geq}$

Properties of timed monoids

- They are not isomorphic. No true Moore-Mealy equivalence

Verimag

Properties of timed monoids

- They are not isomorphic. No true Moore-Mealy equivalence
- None of the monoids above satisfies requirements from Eilenberg, Schutzenberger etc. . .

Verimag

Properties of timed monoids

- They are not isomorphic. No true Moore-Mealy equivalence
- None of the monoids above satisfies requirements from Eilenberg, Schutzenberger etc. . .
- Nevertheless we can try to apply classical recipes.

Verimag

Recognizable languages in $\Sigma^* \oplus \mathbb{R}_{\geq 0}$

L recognizable $\Leftrightarrow$ finitely many $w\backslash L$

- Is $\{5\} \subset \mathbb{R}_{\geq 0}$ recognizable?

Recognizable languages in $\Sigma^* \oplus \mathbb{R}_{\geq 0}$

L recognizable $\Leftrightarrow$ finitely many $w\backslash L$

- Is $\{5\} \subset \mathbb{R}_{\geq 0}$ recognizable? No!

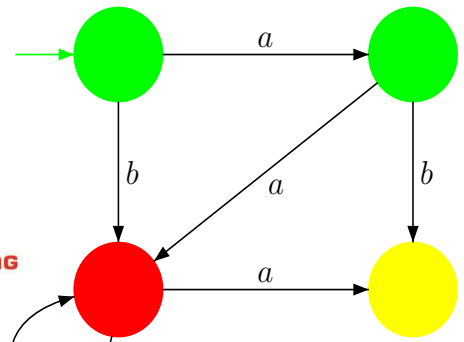
$$\begin{aligned} 1\backslash\{5\} &= \{4\} \\ 2\backslash\{5\} &= \{3\} \\ 2.19\backslash\{5\} &= \{2.81\} \end{aligned}$$

- Only four recognizable subsets of $\mathbb{R}_{\geq 0}$:
  $\emptyset, \{0\}, (0, \infty), \mathbb{R}_{\geq 0}$

Recognizable languages in $\Sigma^* \oplus \mathbb{R}_{\geq 0}$

L recognizable $\Leftrightarrow$ finitely many $w\backslash L$

- Is $\{5\} \subset \mathbb{R}_{\geq 0}$ recognizable? No!
- Only four recognizable subsets of $\mathbb{R}_{\geq 0}$: $\emptyset, \{0\}, (0, \infty), \mathbb{R}_{\geq 0}$
- No quantitative timing! Recognizable timed languages : $(a + b\tau)^*$ etc. . . where τ stands for any positive time.

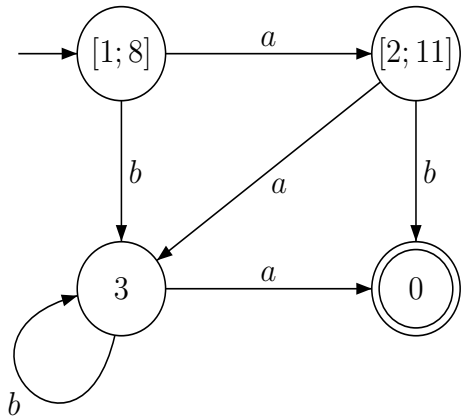


Timed rational languages

- **1st try.** Kleene algebra on $\Sigma^* \oplus \mathbb{R}_{\geq 0}$ generated by singletons: e.g. $(a^*\pi)^* + 2^*$ Known issues:
  - only countable languages. $[2;3]a$ is not in
  - even if we add complementation $[2;3]a$ is not in
  - non-computable numbers allowed

Timed rational languages

- **1st try.** Kleene algebra on $\Sigma^* \oplus \mathbb{R}_{\geq 0}$ generated by singletons: e.g. $(a^*\pi)^* + 2^*$
- **2nd try**(Cătălin's real-time languages.) Kleene algebra on $\Sigma^* \oplus \mathbb{R}_{\geq 0}$ generated by $a, b, c, \ldots, 1, (0, 1)$: e.g $(a^*[2, 3])^* + 2^*$



Conclusion for part 1

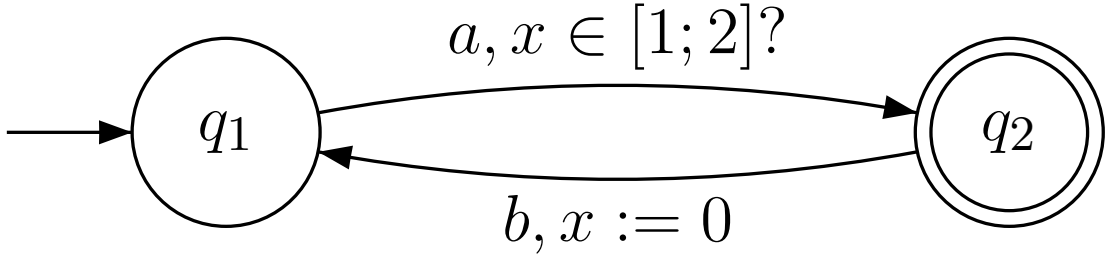
Classical recipes partly apply, but they give classes of languages too restricted to be useful

Verimag

Timed automata

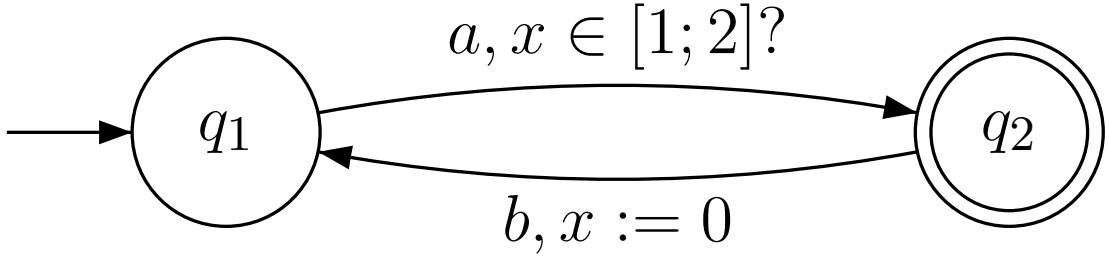
An automaton and its language

- Timed automaton :



An automaton and its language

- Timed automaton :



- Its language : $\{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i.t_i \in [1;2]\}$

Reminder: some old results

- Rich enough for many real-life examples (real-time programs, scheduling, circuits . . .)
- Membership and Empty Language decidable (finite bisimulation)
- Inclusion and Universal Language undecidable
- Languages closed under $\cup, \cap$ and morphism (renaming)
- Languages non-closed under complementation.
- No determinization.
- Model-checking algorithms implemented in KRONOS and UPPAAL

Some technical problems with TA

- They are infinite state : $(q, x_1, \ldots x_k) \in Q \times {\mathbb{R}_{\geq 0}}^k$.
  Unavoidable since recognizable languages are too poor

Some technical problems with TA

- They are infinite state : $(q, x_1, \ldots x_k) \in Q \times {\mathbb{R}_{\geq 0}}^k$.
  Unavoidable since recognizable languages are too poor
- There are long (or infinite) runs that never visit the same state.
  Nevertheless there are complicated pumping lemmata (D. Beauquier)

Verimag

Some technical problems with TA

- They are infinite state : $(q, x_1, \ldots x_k) \in Q \times {\mathbb{R}_{\geq 0}}^k$.
  Unavoidable since recognizable languages are too poor
- There are long (or infinite) runs that never visit the same state.
  Nevertheless there are complicated pumping lemmata (D. Beauquier)
- No direct way to have Kleene Theorem
  Nevertheless there are some ways to have it - see below

Verimag

Conclusions of Part 2

They are useful, they have nice properties, they look like automata, but in the strict sense THEY ARE NOT automata.

Verimag

Regular expressions for timed languages

Verimag

Existing approaches

ACM	TRE - see below
BP99	Expressions with 2^C compositions and stars
BP	Expression with clocks
D	“Multithread” regmino expressions
AD	Balanced expressions

Our old expressions - TRE

$$E ::= 0 \mid \varepsilon \mid \underline{\mathbf{t}} \mid a \mid E+E \mid E \cdot E \mid E^* \mid \langle E \rangle_I \mid E \wedge E \mid [a \mapsto z]E$$

Our old expressions - TRE

$$E ::= 0 \mid \varepsilon \mid \underline{\mathbf{t}} \mid a \mid E+E \mid E \cdot E \mid E^* \mid \langle E \rangle_I \mid E \wedge E \mid [a \mapsto z]E$$

Semantics:

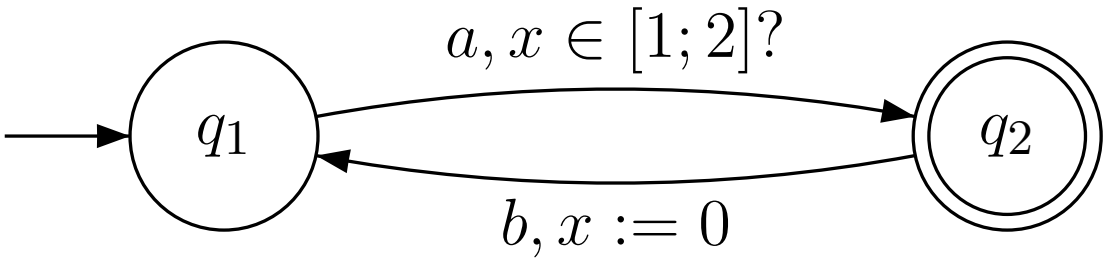
$$\|\underline{\mathbf{t}}\| = \mathbb{R}_{\geq 0} \quad \|a\| = \{a\} \qquad \|0\| = \emptyset \quad \|\varepsilon\| = \{\varepsilon\}$$

$$\|E_1 \cdot E_2\| = \|E_1\| \cdot \|E_2\| \qquad \|E_1 + E_2\| = \|E_1\| \cup \|E_2\|$$

$$\|\langle E \rangle\|_I = \{\sigma \in \|E\| \mid \ell(\sigma) \in I\} \qquad \|E^*\| = \|E\|^*$$

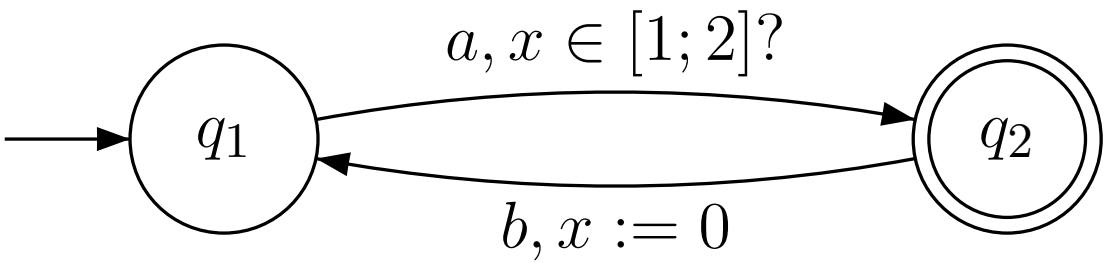
$$\|E_1 \wedge E_2\| = \|E_1\| \cap \|E_2\| \qquad \|[a \mapsto z]E\| = [a \mapsto z]\|E\|$$

A good example and a theorem



$$\{L = \{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i.t_i \in [1;2]\}$$

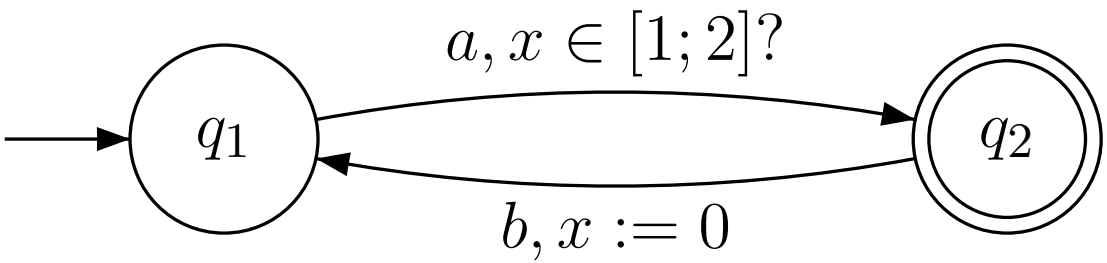
A good example and a theorem



$$\{L = \{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i.t_i \in [1;2]\}$$

An expression for L : $\left(\langle \underline{\mathbf{t}} a \rangle_{[1;2]} \underline{\mathbf{t}} b\right)^*$

A good example and a theorem



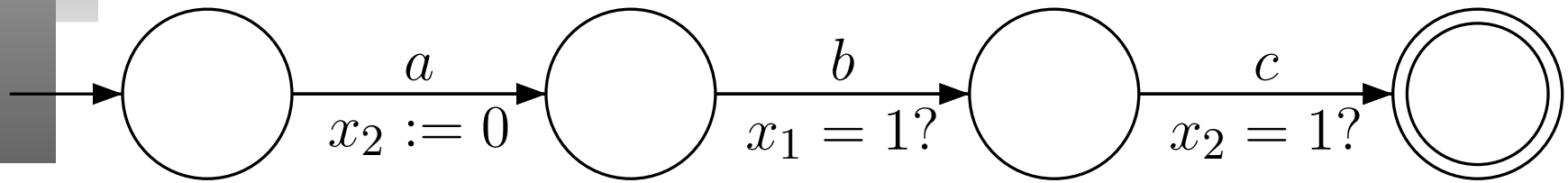
$$\{L = \{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i.t_i \in [1;2]\}$$

An expression for L : $\left(\langle \underline{\mathbf{t}} a \rangle_{[1;2]} \underline{\mathbf{t}} b\right)^*$

"Kleene theorem" [ACM]Timed Automata andTRE (with $\wedge$ and $[a \mapsto z]$) define the same class of timed languages

A nasty example

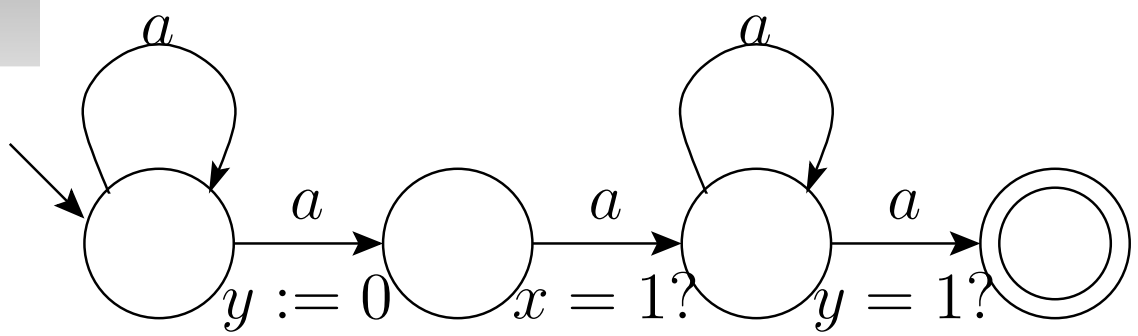
Intersection needed [ACM]



$$\{t_1 a t_2 b t_3 c \mid t_1 + t_2 = 1, t_2 + t_3 = 1\} = \underline{\mathbf{t}} a \langle \underline{\mathbf{t}} b \underline{\mathbf{t}} c \rangle_1 \wedge \langle \underline{\mathbf{t}} a \underline{\mathbf{t}} b \rangle_1 \underline{\mathbf{t}} c$$

Another nasty example

Renaming needed [Herrmann]



$$[b \mapsto a]\big((\underline{\mathbf{t}}a)^*\langle \underline{\mathbf{t}}b(\underline{\mathbf{t}}a)^*\rangle_1 \wedge \langle(\underline{\mathbf{t}}a)^*\underline{\mathbf{t}}b\rangle_1(\underline{\mathbf{t}}a)^*\big).$$

Wanted!

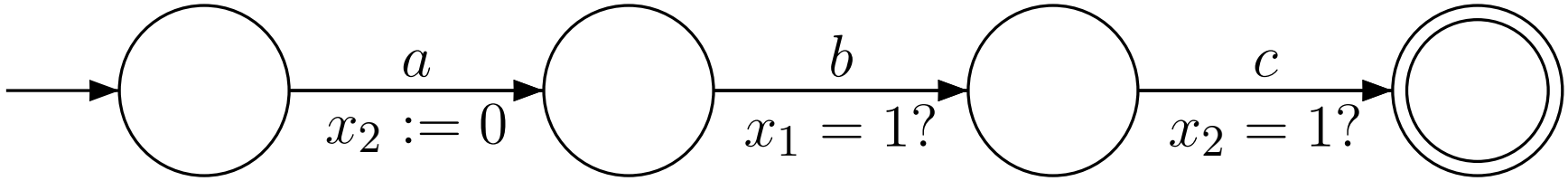
Timed regular expressions

- Equivalent to timed automata
- Automata-free
- Describing directly the timed words (projection-free)
- Without heavy/ugly operations
- With simple and natural semantics
- Elegant
- Ergonomic

Verimag

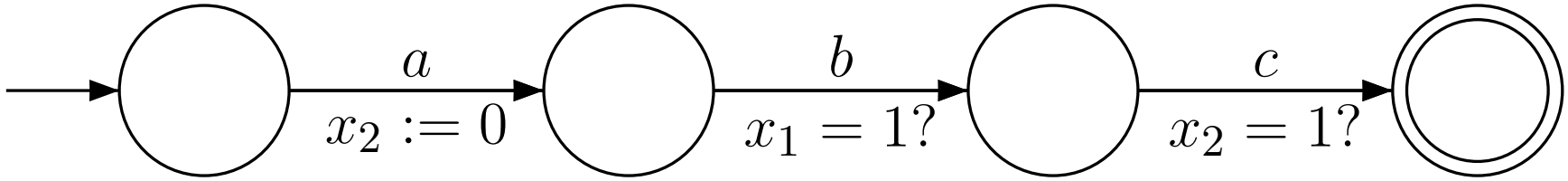
Sometimes Color may help

- Intersection not needed! $\langle \underline{\mathbf{t}} a \langle \underline{\mathbf{t}} b \rangle_1 \underline{\mathbf{t}} c \rangle_1$

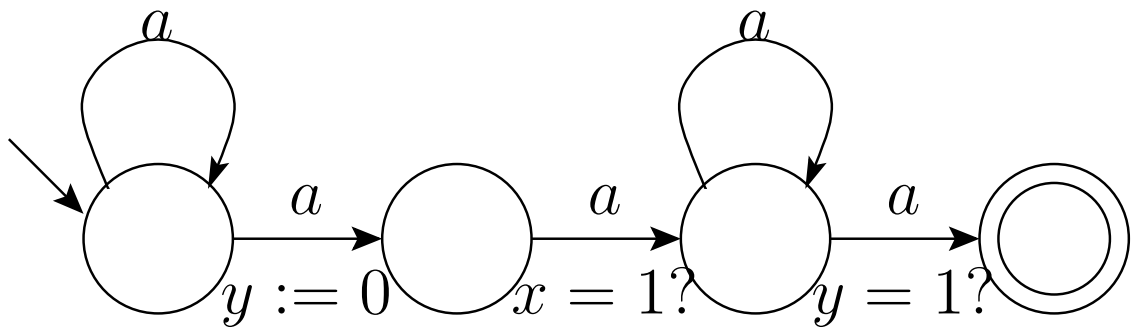


Sometimes Color may help

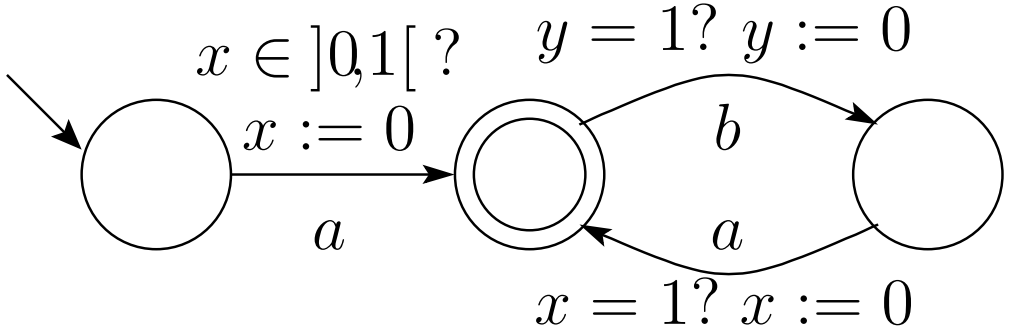
- Intersection not needed! $\langle \underline{\mathtt{t}}a\langle \underline{\mathtt{t}}b\rangle_1 \underline{\mathtt{t}}c\rangle_1$



- Renaming not needed! $\langle (\underline{\mathtt{t}}a)^* \langle \underline{\mathtt{t}}a\rangle_1 (\underline{\mathtt{t}}a)^*\rangle_1$

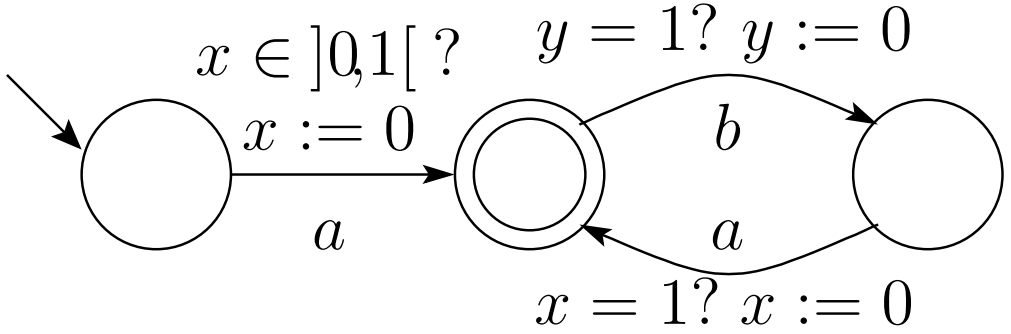


Color is not enough



$$L = \langle \underline{\mathtt{t}}a\, \underbrace{\underline{\mathtt{t}}b\rangle_1 \langle \underline{\mathtt{t}}a}_{1}\, \underbrace{\underline{\mathtt{t}}b\rangle_1 \langle \underline{\mathtt{t}}a}_{1}\, \underline{\mathtt{t}}b\rangle_1 \ldots \langle \underline{\mathtt{t}}a\, \underbrace{\underline{\mathtt{t}}b\rangle_1 \underline{\mathtt{t}}a}_{1}$$

Color is not enough

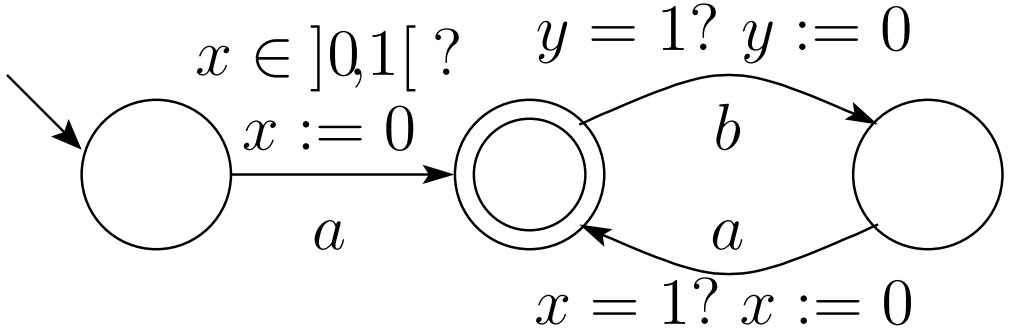


$$L = \langle \underline{\mathbf{t}}a \underbrace{\underline{\mathbf{t}}b\rangle_1 \langle \underline{\mathbf{t}}a}_{1} \underbrace{\underline{\mathbf{t}}b\rangle_1 \langle \underline{\mathbf{t}}a}_{1} \underline{\mathbf{t}}b\rangle_1 \ldots \langle \underline{\mathbf{t}}a \underbrace{\underline{\mathbf{t}}b\rangle_1 \underline{\mathbf{t}}a}_{1}$$

Color does not help, intersection needed:

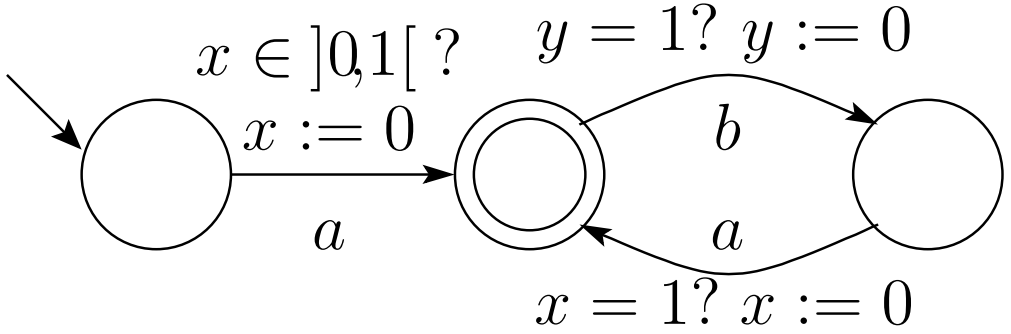
$$\langle \underline{\mathbf{t}}a\rangle_{]0;1[} \langle \underline{\mathbf{t}}b\underline{\mathbf{t}}a\rangle_1^* \wedge \langle \underline{\mathbf{t}}a\underline{\mathbf{t}}b\rangle_1^* \underline{\mathbf{t}}a$$

Let us try again



$$\langle \underline{\mathbf{t}}a\langle \underline{\mathbf{t}}b\rangle_1\langle \underline{\mathbf{t}}a\rangle_1\langle \underline{\mathbf{t}}b\rangle_1\langle \underline{\mathbf{t}}a\rangle_1\langle \underline{\mathbf{t}}b\rangle_1 \ldots \langle \underline{\mathbf{t}}a\rangle_1\langle \underline{\mathbf{t}}b\rangle_1\underline{\mathbf{t}}a\rangle_1$$

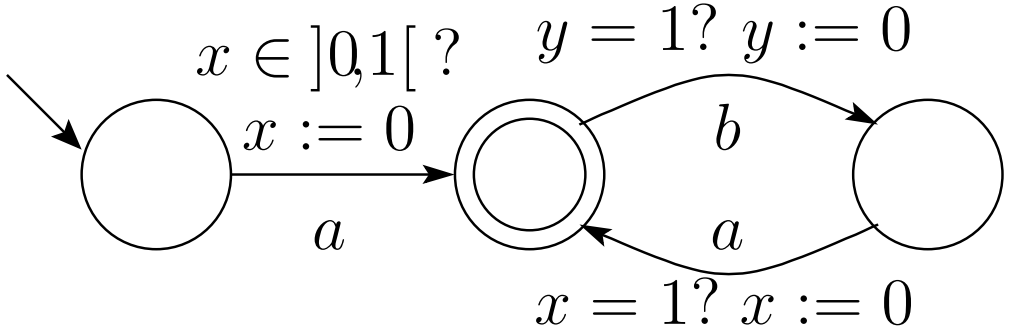
Let us try again



$$\langle \underline{\mathbf{t}}a\langle \underline{\mathbf{t}}b\rangle_1\langle \underline{\mathbf{t}}a\rangle_1\langle \underline{\mathbf{t}}b\rangle_1\langle \underline{\mathbf{t}}a\rangle_1\langle \underline{\mathbf{t}}b\rangle_1 \ldots \langle \underline{\mathbf{t}}a\rangle_1\langle \underline{\mathbf{t}}b\rangle_1\underline{\mathbf{t}}a\rangle_1$$

How to write it shortly?

Let us try again

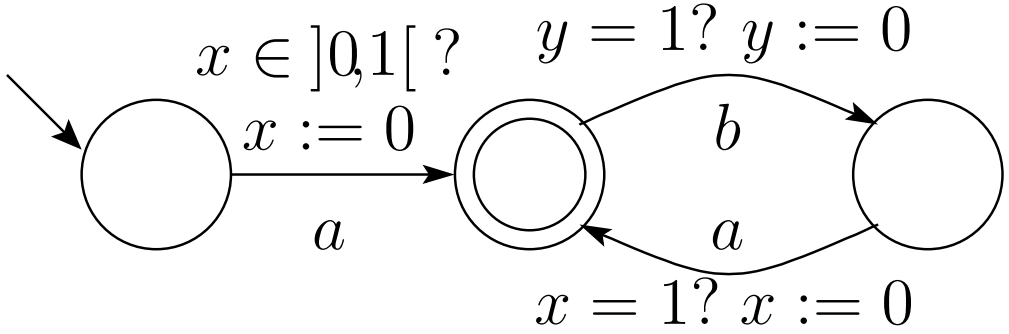


$$\langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \ldots \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1$$

How to write it shortly?

$$\langle \underline{\mathbf{t}}a \Big(\langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \Big)^* \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1$$

Let us try again

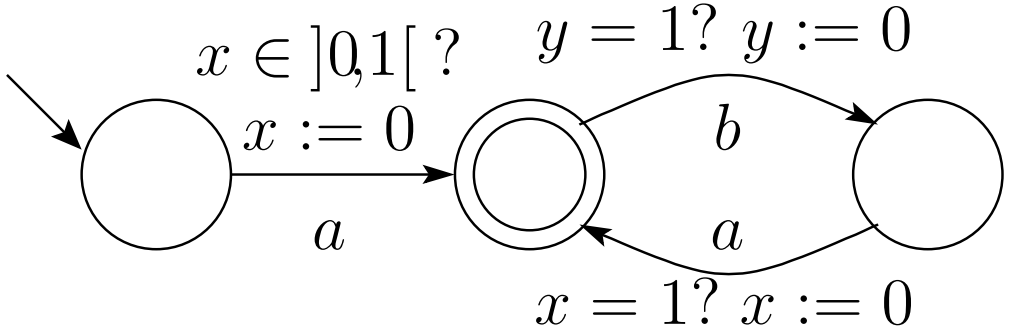


$$\langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \ldots \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1$$

How to write it shortly?

$$\langle \underline{\mathbf{t}}a \Big(\langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \Big)^* \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1 + \langle \underline{\mathbf{t}}a \rangle_{]0,1[}$$

Let us try again



$$\langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \ldots \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1$$

How to write it shortly?

$$\langle \underline{\mathbf{t}}a \left(\langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \right)^* \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1 + \langle \underline{\mathbf{t}}a \rangle_{]0,1[}$$

Eureka!

Balanced Expressions

- **Syntax.** BRE - a regular expression E over $\widetilde{\Sigma} = \Sigma \cup \{\langle, \rangle_I, \langle, \rangle_I, \langle, \rangle_I, \dots\}$

Balanced Expressions

- **Syntax.** BRE - a regular expression E over $\widetilde{\Sigma} = \Sigma \cup \{\langle, \rangle_I, \langle, \rangle_I, \langle, \rangle_I, \dots\}$
- **Two-stage semantics**

Balanced Expressions

- **Syntax.** BRE - a regular expression E over $\widetilde{\Sigma} = \Sigma \cup \{\langle, \rangle_I, \langle, \rangle_I, \langle, \rangle_I, \dots\}$
- **Two-stage semantics**
  1. $|E|$ - untimed language over $\widetilde{\Sigma}$ generated by E. Each $w \in |E|$ can be considered as a (colored) timed regular expression!

Balanced Expressions

- **Syntax.** BRE - a regular expression E over $\widetilde{\Sigma} = \Sigma \cup \{\langle, \rangle_I, \langle, \rangle_I, \langle, \rangle_I, \dots\}$
- **Two-stage semantics**
  1. $|E|$ - untimed language over $\widetilde{\Sigma}$ generated by E. Each $w \in |E|$ can be considered as a (colored) timed regular expression!
  2. Take the union of their timed languages: $\|E\| = \bigcup \{\|w\| \mid w \in |E|\}$

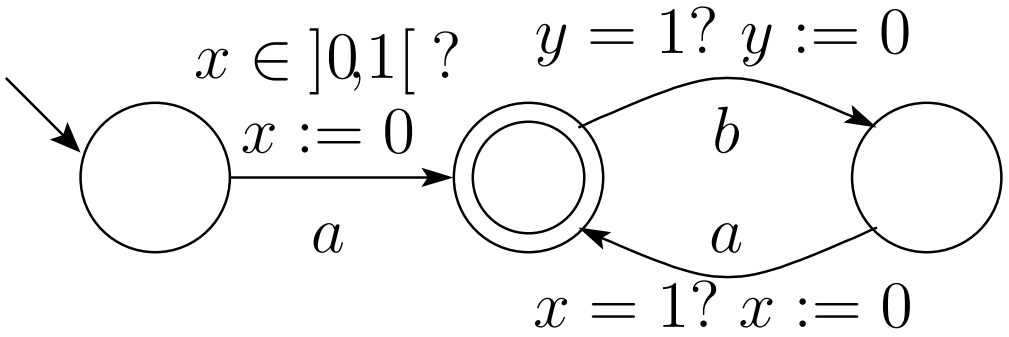
Balanced Expressions

- **Syntax.** BRE - a regular expression E over $\widetilde{\Sigma} = \Sigma \cup \{\langle, \rangle_I, \langle, \rangle_I, \langle, \rangle_I, \dots\}$
- **Two-stage semantics**
  1. $|E|$ - untimed language over $\widetilde{\Sigma}$ generated by E. Each $w \in |E|$ can be considered as a (colored) timed regular expression!
  2. Take the union of their timed languages: $\|E\| = \bigcup \{\|w\| \mid w \in |E|\}$
- **Restriction.** Each $w \in |E|$ should be well parenthesized wrt each color $\boxed{|E| \subset \mathsf{Bal}}$

Balanced Expressions

- **Syntax.** BRE - a regular expression E over $\widetilde{\Sigma} = \Sigma \cup \{\langle, \rangle_I, \langle, \rangle_I, \langle, \rangle_I, \dots\}$
- **Two-stage semantics**
  1. $|E|$ - untimed language over $\widetilde{\Sigma}$ generated by E. Each $w \in |E|$ can be considered as a (colored) timed regular expression!
  2. Take the union of their timed languages: $\|E\| = \bigcup \{\|w\| \mid w \in |E|\}$
- **Restriction.** Each $w \in |E|$ should be well parenthesized wrt each color $\boxed{|E| \subset \mathsf{Bal}}$

Back to the example



- $E = \langle \underline{\mathbf{t}}a \Big(\langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \Big)^* \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1 + \langle \underline{\mathbf{t}}a \rangle_{]0,1[}$
- $|E| = \left\{ \langle \underline{\mathbf{t}}a \rangle_{]0,1[};\ \langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1;\ \langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1; \ldots \right\}$
- $\|E\| = \|\langle \underline{\mathbf{t}}a \rangle_{]0,1[}\| \cup \|\langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1\| \cup \|\langle \underline{\mathbf{t}}a \langle \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rangle_1 \langle \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rangle_1\| \cup \ldots$

Kleene Theorem

Balanced Expressions and Timed Automata define the same class of timed languages

Key lemma

Any TA $\mathcal{A}$ is equivalent to a TA $\overline{\mathcal{A}}$ s.t on each accepting run each clock is checked exactly once after each reset.

Proof idea It is useless to check whether $x \in I$ million times : it should be checked only the first and the last time. This allows to check each clock at most twice...

Checking expression for balance

How to check syntactic correctness: $|E| \subset \mathsf{Bal}$?

Idea: Count unmatched parentheses (structural induction over E).

The Algorithm

$$\mathsf{eff}(a) = \{(\mathbf{0}_n, \mathbf{0}_n)\}$$

$$\mathsf{eff}(\underline{\mathbf{t}}) = \{(\mathbf{0}_n, \mathbf{0}_n)\}$$

$$\mathsf{eff}(\langle^i) = \{(e_i, \mathbf{0}_n)\}$$

$$\mathsf{eff}(\rangle_I^i) = \{(\mathbf{0}_n, e_i)\}$$

$$\mathsf{eff}(E_1 + E_2) = \begin{cases} \perp & \text{iff } \mathsf{eff}(E_1) = \perp \text{ or } \mathsf{eff}(E_2) = \perp \\ \mathsf{eff}(E_1) \cup \mathsf{eff}(E_2) & \text{otherwise} \end{cases}$$

$$\mathsf{eff}(E_1 \cdot E_2) = \begin{cases} \perp & \text{iff } \mathsf{eff}(E_1) = \perp \text{ or } \mathsf{eff}(E_2) = \perp \\ \mathsf{eff}(E_1) \ominus \mathsf{eff}(E_2) & \text{otherwise} \end{cases}$$

$$\mathsf{eff}(E^*) = \begin{cases} \perp & \text{iff } \mathsf{eff}(E) = \perp \\ \perp & \text{iff } \varphi(\mathsf{eff}(E)) \neq \{\mathbf{0}_n\} \\ \bigcup_{k \leq card(\mathsf{eff}(E))} \mathsf{eff}(E)^{k\ominus} & \text{otherwise} \end{cases}$$

Expression E is balanced iff $\mathsf{eff}(E) = \{(\mathbf{0}_n, \mathbf{0}_n)\}$

Comparing formalisms

	BP	TRE	Color	BRE
Equivalent to timed automata	✓	✓	✓	✓
Automata-free	✓	✓	✓	✓
Projection-free	✓	✓	✓	✓
No heavy/ugly operations	✓	✓	✓	✓
Simple and natural semantics	✓	✓	✓	✓
Elegant	?	?	?	?
Ergonomic	✓	✓	✓	✓
Easy syntactic analysis	✓	✓	✓	✓

Verimag

Conclusion of Part 4

We know several solutions, but not the perfect one. . .