Distributed Timed Automata with Independently Evolving Clocks

Paul Gastin

LSV, ENS Cachan, CNRS

Joint work with
S. Akshay, Benedikt Bollig, Madhavan Mukund, K Narayan Kumar

Séminaire LIAFA, 6 April 2009

Motivations

Aim

Study the expressive power of local clocks as a synchronization mechanism in a distributed system.

- Distributed systems with no explicit communication or synchronization.
- Clocks as a synchronization mechanism.
- Clocks on different processes evolve independently according to local times.

Plan

1. Distributed Timed Automata

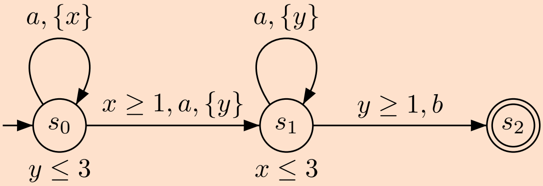
Region abstraction and existential semantics

Universal semantics and undecidability

Reactive (Game) Semantics

Timed automata (Alur & Dill)

Example: TA



$$\begin{array}{ccccccccccc}
s_0 & \xrightarrow[0.7]{a} & s_0 & \xrightarrow[1.5]{a} & s_0 & \xrightarrow[2.6]{a} & s_1 & \xrightarrow[3.3]{a} & s_1 & \xrightarrow[5.0]{b} & s_2 \\
x = 0 & & 0 & & 0 & & 1.1 & & 1.8 & & 2.5 \\
y = 0 & & 0.7 & & 1.5 & & 0 & & 0 & & 1.7
\end{array}$$

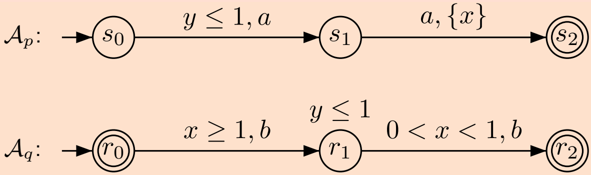
Distributed Timed automata

Definition: DTA

$\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ where

- each $\mathcal{A}_p$ is a classical timed automaton
- $\pi : \mathcal{Z} \longrightarrow Proc$ assigns processes to clocks. If $\pi(x) = p$ then
  - clock x evolves according to local time on process p
  - only process p may reset clock x
  - all processes may read clock x (i.e., use x in guards or invariants)

Example: DTA with $\pi(x) = p$ and $\pi(y) = q$



Local Times

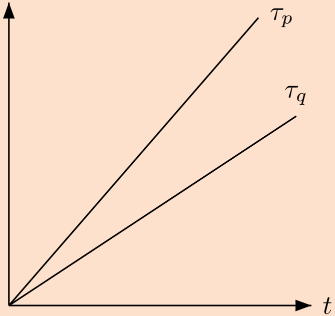
Local Times

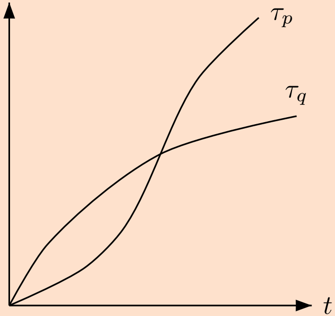
- Processes do not have access to the absolute (global) time.
- Each process has its own local time: $\tau_p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$

$$\tau_p(t)\text{: local time on process } p \text{ at absolute time } t$$

continuous, strictly increasing, diverging, $\tau_p(0) = 0$.

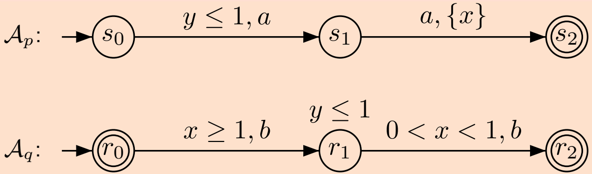
Example: Local Times





Runs of DTA's & Untimed Behaviours

Example: DTA with $\pi(x) = p$ and $\pi(y) = q$

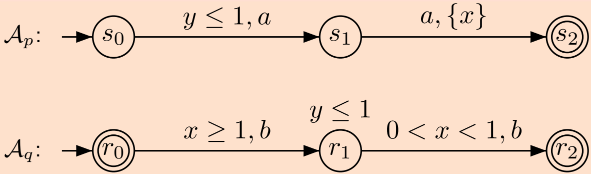


If $\tau_p > \tau_q$ then $abab \in \mathcal{L}(\mathcal{D}, \tau)$ (e.g. $\tau_p(t) = 2t$ and $\tau_q(t) = t$)

$$
\begin{array}{ccccccccc}
s_0 & \xrightarrow{a} & s_1 & \xrightarrow{b} & s_1 & \xrightarrow{a} & s_2 & \xrightarrow{b} & s_2 \\
r_0 & 0.2 & r_0 & 0.6 & r_1 & 0.7 & r_1 & 0.8 & r_2 \\
x = 0 & & 0.4 & & 1.2 & & 0 & & 0.2 \\
y = 0 & & 0.2 & & 0.6 & & 0.7 & & 0.8
\end{array}
$$

Runs of DTA's & Untimed Behaviours

Example: DTA with $\pi(x) = p$ and $\pi(y) = q$



If $\tau_p > \tau_q$ then $abab \in \mathcal{L}(\mathcal{D}, \tau)$ (e.g. $\tau_p(t) = 2t$ and $\tau_q(t) = t$)

s_0	$\xrightarrow{a}$	s_1	$\xrightarrow{b}$	s_1	$\xrightarrow{a}$	s_2	$\xrightarrow{b}$	s_2
r_0	0.2	r_0	0.6	r_1	0.7	r_1	0.8	r_2
$x = 0$		0.4		1.2		0		0.2
$y = 0$		0.2		0.6		0.7		0.8

If $\tau_p = \tau_q$ then $abab \notin \mathcal{L}(\mathcal{D}, \tau)$ (e.g. $\tau_p(t) = \tau_q(t) = 2t$)

s_0	$\xrightarrow{a}$	s_1	$\xrightarrow{b}$	s_1	$\xrightarrow{a}$	s_2
r_0	0.2	r_0	0.5	r_1	0.5	r_1
$x = 0$		0.4		1		0
$y = 0$		0.4		1		1

Formal Semantics of DTA's

Let $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ be an DTA with local times $\tau = (\tau_p)_{p \in Proc}$.

Definition: (Infinite) Transition System $\mathrm{TS}(\mathcal{D}, \tau)$

- Configurations are tuples (s, t, v) where
  - $s = (s_p)_{p \in Proc}$ where s_p is a state of $\mathcal{A}_p$ for each $p \in Proc$
  - $t \in \mathbb{R}_{\geq 0}$ is the absolute time
  - $v : \mathcal{Z} \longrightarrow \mathbb{R}_{\geq 0}$ is the valuation of clocks.

Formal Semantics of DTA's

Let $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ be an DTA with local times $\tau = (\tau_p)_{p \in Proc}$.

Definition: (Infinite) Transition System $\mathrm{TS}(\mathcal{D}, \tau)$

- Configurations are tuples (s, t, v) where
  - $s = (s_p)_{p \in Proc}$ where s_p is a state of $\mathcal{A}_p$ for each $p \in Proc$
  - $t \in \mathbb{R}_{\geq 0}$ is the absolute time
  - $v : \mathcal{Z} \longrightarrow \mathbb{R}_{\geq 0}$ is the valuation of clocks.
- For $t < t'$ we define $v_{t,t'}(x) = v(x) + \tau_{\pi(x)}(t') - \tau_{\pi(x)}(t)$.

Formal Semantics of DTA's

Let $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ be an DTA with local times $\tau = (\tau_p)_{p \in Proc}$.

Definition: (Infinite) Transition System $\mathrm{TS}(\mathcal{D}, \tau)$

- Configurations are tuples (s, t, v) where
  - $s = (s_p)_{p \in Proc}$ where s_p is a state of $\mathcal{A}_p$ for each $p \in Proc$
  - $t \in \mathbb{R}_{\geq 0}$ is the absolute time
  - $v : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ is the valuation of clocks.
- For $t < t'$ we define $v_{t,t'}(x) = v(x) + \tau_{\pi(x)}(t') - \tau_{\pi(x)}(t)$.
- Transitions : $(s, t, v) \xrightarrow{g,a,R} (s', t', v')$ if
  - $s_p \xrightarrow{g,a,R} s'_p$ for some $p \in Proc$ and $s'_q = s_q$ for all $q \neq p$,
  - $v_{t,t''} \models \bigwedge_{q \in Proc} I_q(s_q)$ for all $t \leq t'' \leq t'$,
  - $v_{t,t'} \models g$
  - $v' = v_{t,t'}[R]$ (clocks in R are reset)
  - $v' \models \bigwedge_{q \in Proc} I_q(s'_q)$.

Formal Semantics of DTA's

Let $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ be an DTA with local times $\tau = (\tau_p)_{p \in Proc}$.

Definition: (Infinite) Transition System $\mathrm{TS}(\mathcal{D}, \tau)$

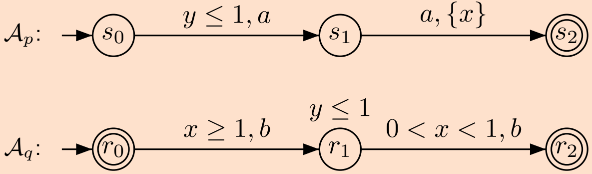
- Configurations are tuples (s, t, v) where
  - $s = (s_p)_{p \in Proc}$ where s_p is a state of $\mathcal{A}_p$ for each $p \in Proc$
  - $t \in \mathbb{R}_{\geq 0}$ is the absolute time
  - $v : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ is the valuation of clocks.
- For $t < t'$ we define $v_{t,t'}(x) = v(x) + \tau_{\pi(x)}(t') - \tau_{\pi(x)}(t)$.
- Transitions : $(s, t, v) \xrightarrow{g,a,R} (s', t', v')$ if
  - $s_p \xrightarrow{g,a,R} s'_p$ for some $p \in Proc$ and $s'_q = s_q$ for all $q \neq p$,
  - $v_{t,t''} \models \bigwedge_{q \in Proc} I_q(s_q)$ for all $t \leq t'' \leq t'$,
  - $v_{t,t'} \models g$
  - $v' = v_{t,t'}[R]$ (clocks in R are reset)
  - $v' \models \bigwedge_{q \in Proc} I_q(s'_q)$.
- $w = a_1 \dots a_n \in \mathcal{L}(\mathcal{D}, \tau)$ (with $a_i \in \Sigma \cup \{\varepsilon\}$) if there is a run in $\mathrm{TS}(\mathcal{D}, \tau)$

$$(s_0, t_0, v_0) \xrightarrow{g_1,a_1,R_1} (s_1, t_1, v_1) \xrightarrow{g_2,a_2,R_2} \cdots \xrightarrow{g_n,a_n,R_n} (s_n, t_n, v_n)$$

with s_0 initial, $t_0 = 0$, $v_0(x) = 0$ for all $x \in \mathcal{Z}$ and s_n final.

Semantics of DTA's

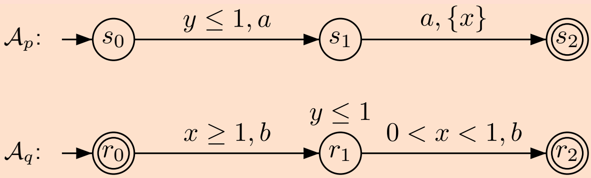
Example: DTA $\mathcal{D}$ with $\pi(x) = p$ and $\pi(y) = q$



- If $\tau_p = \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa\}$.

Semantics of DTA's

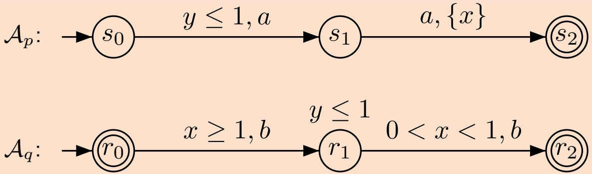
Example: DTA $\mathcal{D}$ with $\pi(x) = p$ and $\pi(y) = q$



- If $\tau_p = \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa\}$.
- If $\tau_p > \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa, abab, baab\}$.

Semantics of DTA's

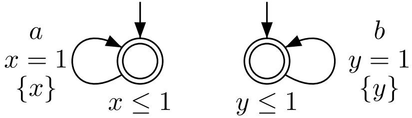
Example: DTA $\mathcal{D}$ with $\pi(x) = p$ and $\pi(y) = q$



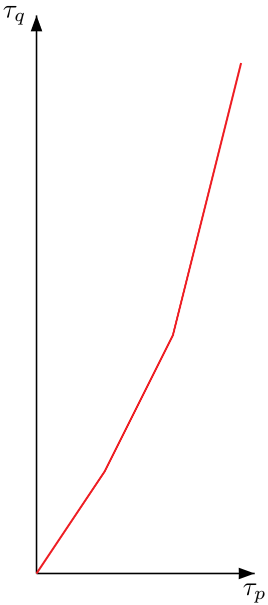
- If $\tau_p = \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa\}$.
- If $\tau_p > \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa, abab, baab\}$.
- For all local times τ, we have $aa \in \mathcal{L}(\mathcal{D}, \tau)$.

Unregular Behaviours

Consider the following DTA $\mathcal{D}$

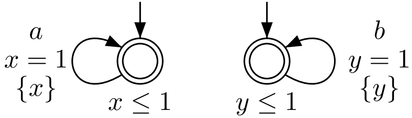


with $\pi(x) = p$ and $\pi(y) = q$
and the local times on the right.



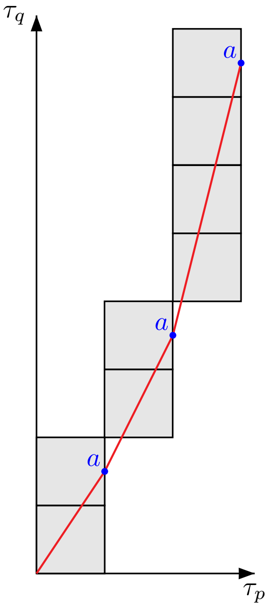
Unregular Behaviours

Consider the following DTA $\mathcal{D}$



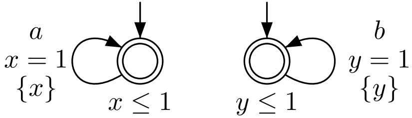
with $\pi(x) = p$ and $\pi(y) = q$
and the local times on the right.

a occurs every local time unit of p.



Unregular Behaviours

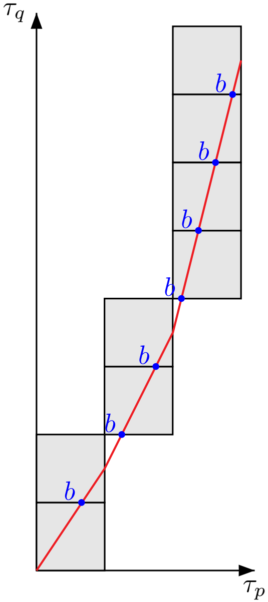
Consider the following DTA $\mathcal{D}$



with $\pi(x) = p$ and $\pi(y) = q$
and the local times on the right.

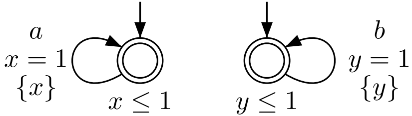
a occurs every local time unit of p.

b occurs every local time unit of q.

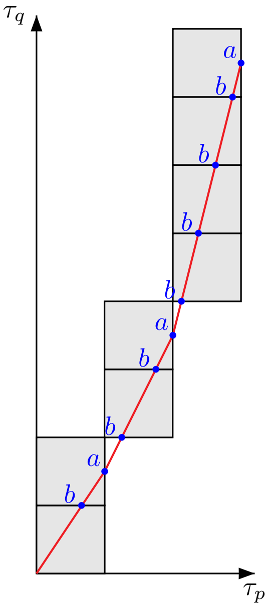


Unregular Behaviours

Consider the following DTA $\mathcal{D}$



with $\pi(x) = p$ and $\pi(y) = q$
and the local times on the right.



a occurs every local time unit of p.

b occurs every local time unit of q.

$\mathcal{L}(\mathcal{D}, \tau)$ are the finite prefixes of $bab^2ab^4ab^8a\cdots$

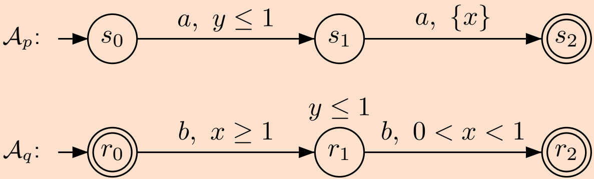
Existential & Universal Semantics

Definition: Existential & Universal Semantics

Let $\mathcal{D}$ be a DTA.

- $\mathcal{L}_\exists(\mathcal{D}) = \bigcup_\tau \mathcal{L}(\mathcal{D}, \tau)$
- $\mathcal{L}_\forall(\mathcal{D}) = \bigcap_\tau \mathcal{L}(\mathcal{D}, \tau)$

Example: $\mathcal{L}_\exists(\mathcal{D}) = \{aa, abab, baab\}$ $\mathcal{L}_\forall(\mathcal{D}) = \{aa\}$



Negative & Positive Specifications

Aim: robustness of a DTA $\mathcal{D}$ against relative local times

Definition: Negative Specifications (Safety)

Given a set Bad of undesired behaviours,

Does a DTA $\mathcal{D}$ robustly avoid Bad

$$\mathcal{L}_{\exists}(\mathcal{D}) \cap \mathrm{Bad} = \emptyset$$

Negative & Positive Specifications

Aim: robustness of a DTA $\mathcal{D}$ against relative local times

Definition: Negative Specifications (Safety)

Given a set Bad of undesired behaviours,

Does a DTA $\mathcal{D}$ robustly avoid Bad

$$\mathcal{L}_{\exists}(\mathcal{D}) \cap \mathrm{Bad} = \emptyset$$

Definition: Positive Specifications (Liveness)

Given a set Good of desired behaviours,

Does a DTA $\mathcal{D}$ robustly exhibit Good

$$\mathrm{Good} \subseteq \mathcal{L}_{\forall}(\mathcal{D})$$

Plan

Distributed Timed Automata

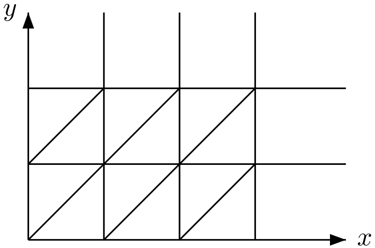
2 Region abstraction and existential semantics

Universal semantics and undecidability

Reactive (Game) Semantics

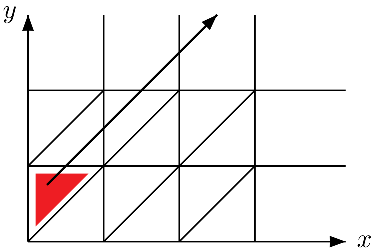
Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$



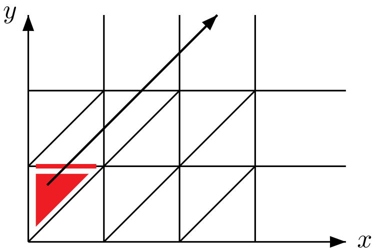
Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$



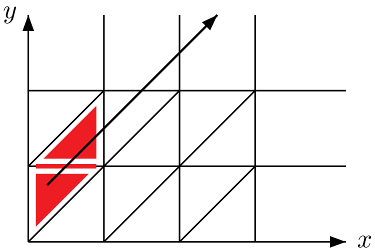
Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$



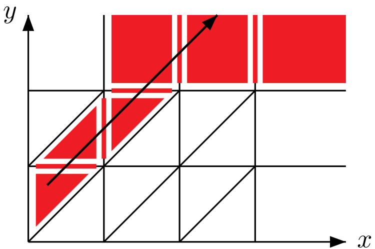
Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$



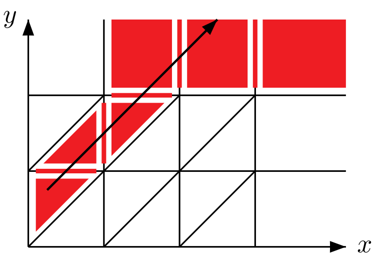
Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$

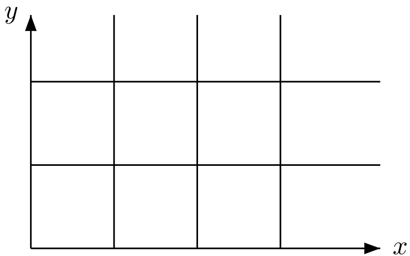


Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$

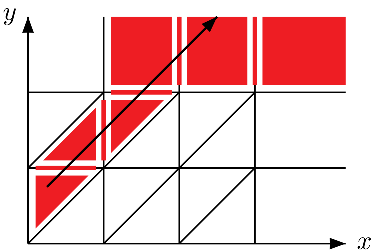


Regions when $\pi(x) \neq \pi(y)$

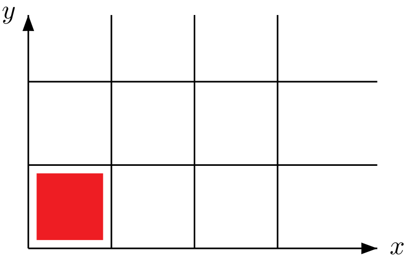


Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$

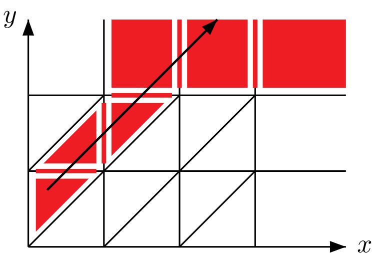


Regions when $\pi(x) \neq \pi(y)$

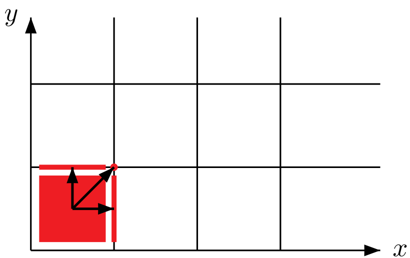


Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$

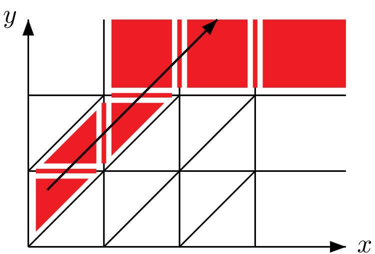


Regions when $\pi(x) \neq \pi(y)$

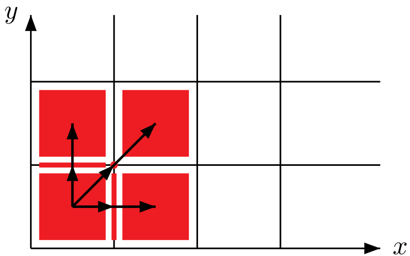


Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$

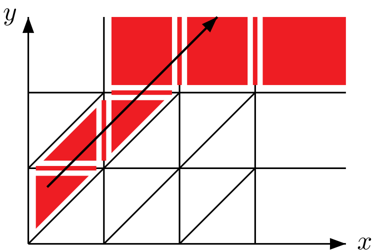


Regions when $\pi(x) \neq \pi(y)$

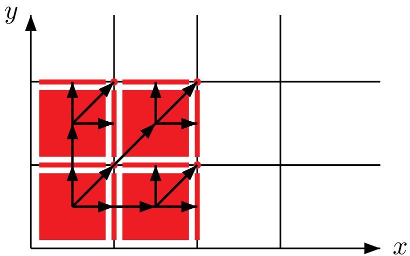


Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$

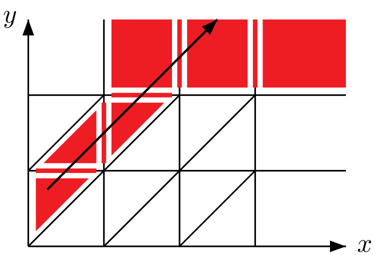


Regions when $\pi(x) \neq \pi(y)$

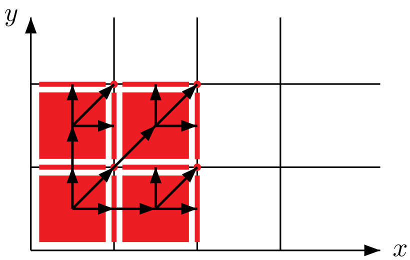


Region abstraction for ∃-semantics

Regions when $\pi(x) = \pi(y)$



Regions when $\pi(x) \neq \pi(y)$



Proposition:

The region equivalence of a DTA is a timed abstract bisimulation for its ∃-semantics.

Region abstraction for $\exists$-semantics

Theorem: Region abstraction

Let $\mathcal{D}$ be a DTA. Let $\mathcal{R}_{\mathcal{D}}$ be its region abstraction.

$$\mathcal{L}_{\exists}(\mathcal{D}) = \mathcal{L}(\mathcal{R}_{\mathcal{D}})$$

and

$$|\mathcal{R}_{\mathcal{D}}| \leq |\mathcal{D}| \cdot (2\,C + 2)^{|\mathcal{Z}|} \cdot |\mathcal{Z}|!$$

Region abstraction for ∃-semantics

Theorem: Region abstraction

Let $\mathcal{D}$ be a DTA. Let $\mathcal{R}_{\mathcal{D}}$ be its region abstraction.

$$\mathcal{L}_{\exists}(\mathcal{D}) = \mathcal{L}(\mathcal{R}_{\mathcal{D}})$$

and

$$|\mathcal{R}_{\mathcal{D}}| \leq |\mathcal{D}| \cdot (2\,C + 2)^{|\mathcal{Z}|} \cdot |\mathcal{Z}|!$$

Corollary: Negative specifications

Model checking regular negative specifications for DTA's is decidable.

$$\mathcal{L}_{\exists}(\mathcal{D}) \cap \mathrm{Bad} = \emptyset$$

Plan

Distributed Timed Automata

Region abstraction and existential semantics

3. Universal semantics and undecidability

Reactive (Game) Semantics

Undecidability of the universal semantics

Theorem: Undecidability Skip proof.

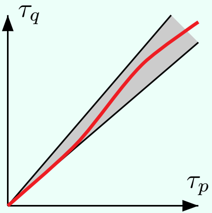
Let $\mathcal{D}$ be a DTA.

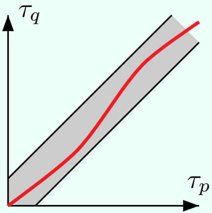
emptiness: $\mathcal{L}_\forall(\mathcal{D}) = \emptyset$ is undecidable.

universality: $\mathcal{L}_\forall(\mathcal{D}) = \Sigma^*$ is undecidable.

Even for 2 processes, 1 clock each and bounded drifts: $\exists\alpha > 0, \forall t > 0,$

$$1 - \alpha \leq \frac{\tau_q(t)}{\tau_p(t)} < 1 + \alpha \qquad \text{or} \qquad |\tau_q(t) - \tau_p(t)| \leq \alpha$$





Corollary: Positive specifications $\quad \mathrm{Good} \subseteq \mathcal{L}_\forall(\mathcal{D})$

Model checking regular positive specifications for DTA's is undecidable.

Undecidability of emptiness

Proof: Reduction from Post Correspondance Problem

- Given two morphisms $f, g : A^+ \to \{0,1\}^+$ with $A = \{a_1, \dots, a_k\}$.
- Does there exist $w \in A^+$ such that $f(w) = g(w)$?

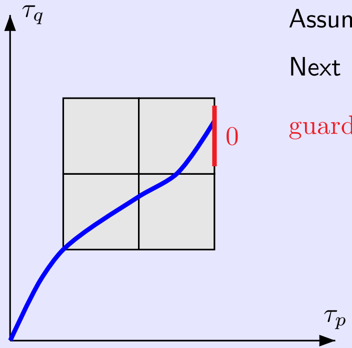
Undecidability of emptiness

Proof: Reduction from Post Correspondance Problem

- Given two morphisms $f, g : A^+ \to \{0,1\}^+$ with $A = \{a_1, \ldots, a_k\}$.
- Does there exist $w \in A^+$ such that $f(w) = g(w)$?

Definition: Words defined by local times

Each pair of local times $\tau = (\tau_p, \tau_q)$ is mapped to a word $\mathrm{dir}(\tau) \in \{0,1,2\}^\omega$.



Assume $x = y = 0$ when entering the 2×2 square.

Next letter of $\mathrm{dir}(\tau)$ is 0

$\mathrm{guard}(0) := x = 2 \wedge 1 < y < 2$

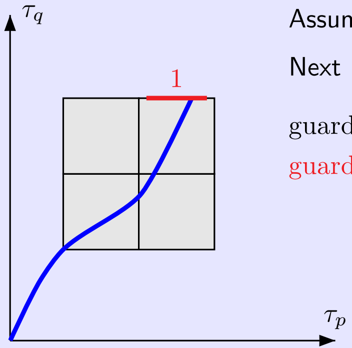
Undecidability of emptiness

Proof: Reduction from Post Correspondance Problem

- Given two morphisms $f, g : A^+ \to \{0,1\}^+$ with $A = \{a_1, \ldots, a_k\}$.
- Does there exist $w \in A^+$ such that $f(w) = g(w)$?

Definition: Words defined by local times

Each pair of local times $\tau = (\tau_p, \tau_q)$ is mapped to a word $\mathrm{dir}(\tau) \in \{0,1,2\}^\omega$.



Assume $x = y = 0$ when entering the 2×2 square.

Next letter of $\mathrm{dir}(\tau)$ is 1

$\mathrm{guard}(0) := x = 2 \wedge 1 < y < 2$

$\mathrm{guard}(1) := 1 < x < 2 \wedge y = 2$

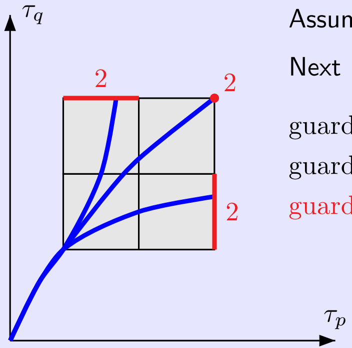
Undecidability of emptiness

Proof: Reduction from Post Correspondance Problem

- Given two morphisms $f, g : A^+ \to \{0,1\}^+$ with $A = \{a_1, \ldots, a_k\}$.
- Does there exist $w \in A^+$ such that $f(w) = g(w)$?

Definition: Words defined by local times

Each pair of local times $\tau = (\tau_p, \tau_q)$ is mapped to a word $\mathrm{dir}(\tau) \in \{0,1,2\}^\omega$.



Assume $x = y = 0$ when entering the 2×2 square.

Next letter of $\mathrm{dir}(\tau)$ is 2

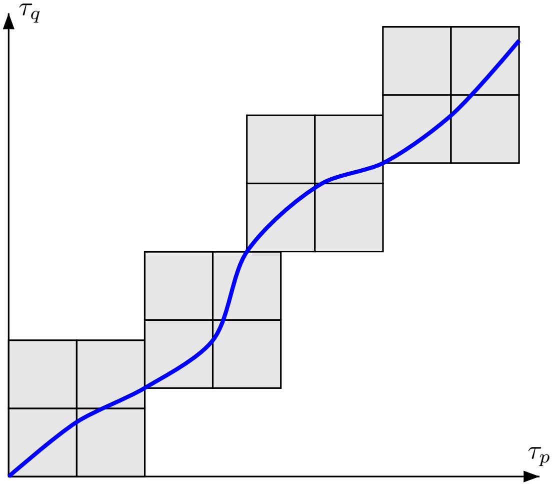
$\mathrm{guard}(0) := x = 2 \wedge 1 < y < 2$

$\mathrm{guard}(1) := 1 < x < 2 \wedge y = 2$

$\mathrm{guard}(2) := (x = 2 \wedge (y \leq 1 \vee y = 2)) \vee (x \leq 1 \wedge y = 2)$

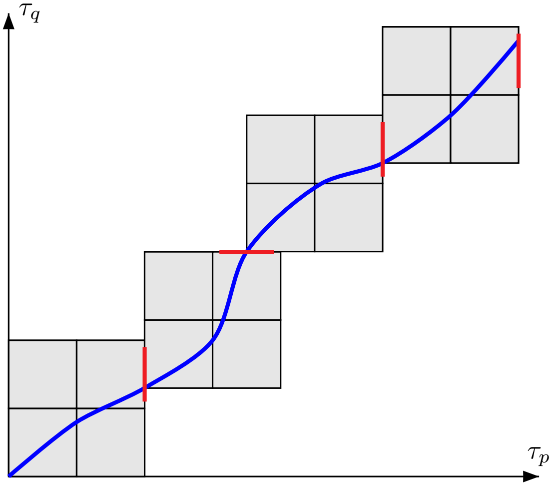
Words defined by local times

Clocks x, y are reset when reaching the 2×2 square boundary



Words defined by local times

Clocks x, y are reset when reaching the 2×2 square boundary



$\mathrm{dir}(\tau) = 0100\cdots$

Undecidability of emptiness

Recall that we are given two morphisms

$$f, g : A^+ \to \{0,1\}^+$$

We want to construct DTA's $\mathcal{D}_f$ and $\mathcal{D}_g$ such that for all local times $\tau = (\tau_p, \tau_q)$

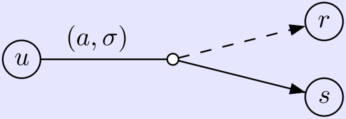
$$\mathcal{L}(\mathcal{D}_f, \tau) = \{wb \in A^+b \mid f(w)2 \not\leq \mathrm{dir}(\tau)\}$$
$$\mathcal{L}(\mathcal{D}_g, \tau) = \{wb \in A^+b \mid g(w)2 \leq \mathrm{dir}(\tau)\}$$

For simplicity, we use a central controls for our automata,
but they can be distributed to get DTA's.

Undecidability of emptiness

Definition: Macro transition

For $a \in A$ and $\sigma = d_1 d_2 \dots d_n \in \{0,1,2\}^+$ we define



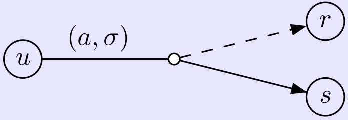
From u with $x = y = 0$, reading input letter a we reach

- s with $x = y = 0$ if local times $\tau = (\tau_p, \tau_q)$ evolve according to σ
- r otherwise

Undecidability of emptiness

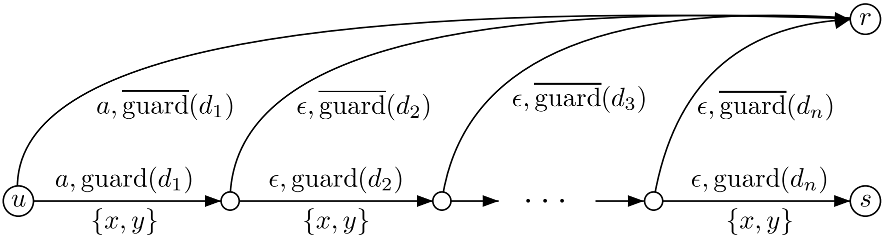
Definition: Macro transition

For $a \in A$ and $\sigma = d_1 d_2 \dots d_n \in \{0, 1, 2\}^+$ we define

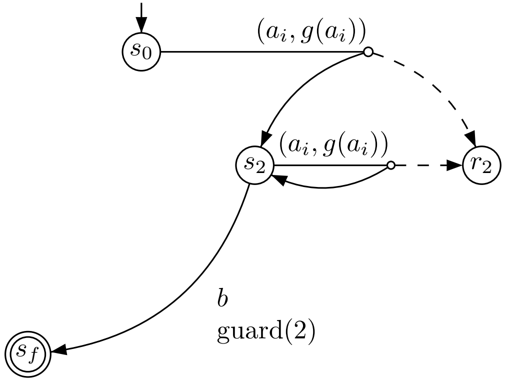


From u with $x = y = 0$, reading input letter a we reach

- s with $x = y = 0$ if local times $\tau = (\tau_p, \tau_q)$ evolve according to σ
- r otherwise



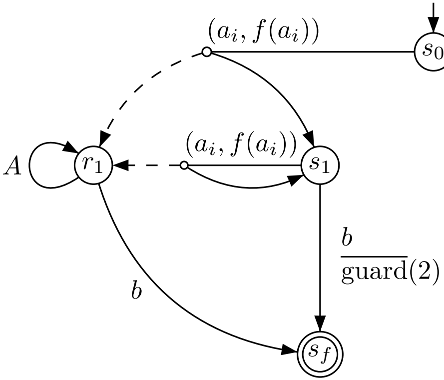
Undecidability of emptiness



Proposition: $\mathcal{L}(\mathcal{D}_g, \tau) = \{wb \in A^+b \mid g(w)2 \leq \mathrm{dir}(\tau)\}$

- $s_0 \xrightarrow{w} s_2$ iff $g(w) \leq \mathrm{dir}(\tau)$
- $s_0 \xrightarrow{w} r_2$ iff $g(w) \not\leq \mathrm{dir}(\tau)$

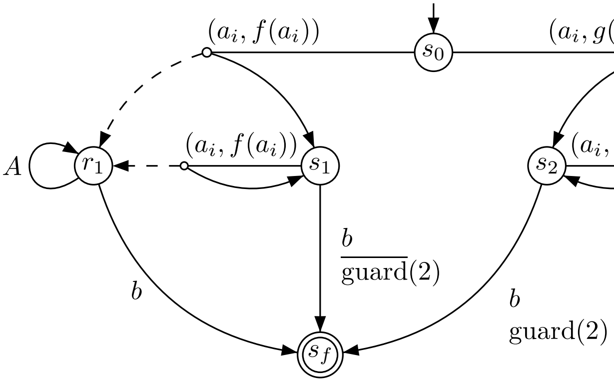
Undecidability of emptiness



Proposition: $\mathcal{L}(\mathcal{D}_f, \tau) = \{wb \in A^+b \mid f(w)2 \not\leq \mathrm{dir}(\tau)\}$

- $s_0 \xrightarrow{w} s_1$ iff $f(w) \leq \mathrm{dir}(\tau)$
- $s_0 \xrightarrow{w} r_1$ iff $f(w) \not\leq \mathrm{dir}(\tau)$

Undecidability of emptiness



Proposition: $\mathcal{L}_\forall(\mathcal{D}) = \{wb \in A^+b \mid f(w) = g(w)\}$

- $s_0 \xrightarrow{w} s_1$ iff $f(w) \leq \mathrm{dir}(\tau)$
- $s_0 \xrightarrow{w} r_1$ iff $f(w) \not\leq \mathrm{dir}(\tau)$
- $s_0 \xrightarrow{w} s_2$ iff $g(w) \leq \mathrm{dir}(\tau)$

Plan

Distributed Timed Automata

Region abstraction and existential semantics

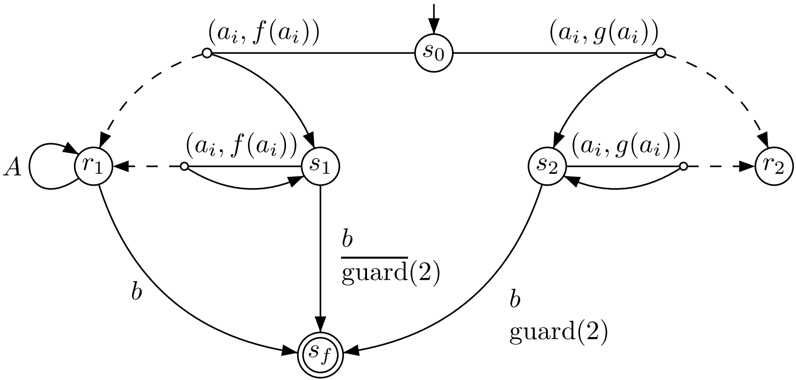
Universal semantics and undecidability

4 Reactive (Game) Semantics

Reactive (Game) Semantics

Remark: Positive Specifications and universal semantics

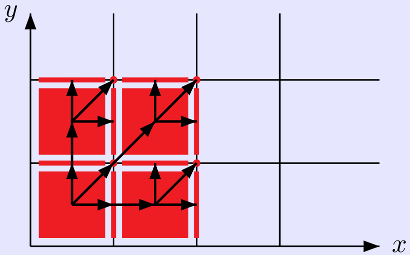
Good $\subseteq \mathcal{L}_\forall(\mathcal{D})$ does not imply that the system can be controlled in order to exhibit all Good behaviours, whatever local times are.



Reactive (Game) Semantics

Definition: Reactive (Game) Semantics

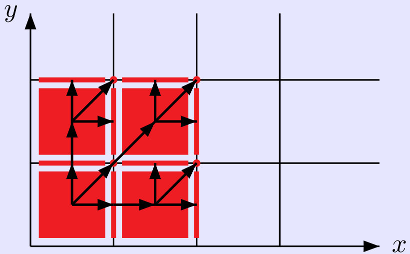
- Environment controls how local times evolve (time-elapse transitions)



Reactive (Game) Semantics

Definition: Reactive (Game) Semantics

- Environment controls how local times evolve (time-elapse transitions)

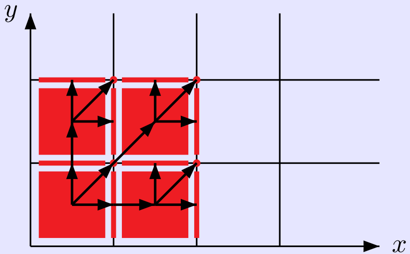


- System observes current region and controls discrete transitions
- Not turn-based: system may execute several discrete transitions

Reactive (Game) Semantics

Definition: Reactive (Game) Semantics

- Environment controls how local times evolve (time-elapse transitions)



- System observes current region and controls discrete transitions
- Not turn-based: system may execute several discrete transitions

$$\mathcal{L}_{\mathrm{react}}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{System has a winning strategy}\}$$

Decidability of the reactive semantics

Theorem: Regularity

Let $\mathcal{D}$ be a DTA. $\mathcal{L}_{\mathrm{react}}(\mathcal{D})$ is regular.

Proof: construct an alternating automaton with ε-transitions accepting $\mathcal{L}_{\mathrm{react}}(\mathcal{D})$.

Decidability of the reactive semantics

Theorem: Regularity

Let $\mathcal{D}$ be a DTA. $\mathcal{L}_{\text{react}}(\mathcal{D})$ is regular.

Proof: construct an alternating automaton with ε-transitions accepting $\mathcal{L}_{\text{react}}(\mathcal{D})$.

Corollary: Positive specifications

Model checking regular positive specifications is decidable for the reactive semantics.

$$\text{Good} \subseteq \mathcal{L}_{\text{react}}(\mathcal{D})$$

Decidability of the reactive semantics

Theorem: Regularity

Let $\mathcal{D}$ be a DTA. $\mathcal{L}_{\text{react}}(\mathcal{D})$ is regular.

Proof: construct an alternating automaton with ε-transitions accepting $\mathcal{L}_{\text{react}}(\mathcal{D})$.

Corollary: Positive specifications

Model checking regular positive specifications is decidable for the reactive semantics.

$$\text{Good} \subseteq \mathcal{L}_{\text{react}}(\mathcal{D})$$

Proposition: Reactive vs. Universal

- $\mathcal{L}_{\text{react}}(\mathcal{D}) \subseteq \mathcal{L}_{\forall}(\mathcal{D})$ for all DTA's $\mathcal{D}$.
- In general, $\mathcal{L}_{\text{react}}(\mathcal{D}) \subsetneq \mathcal{L}_{\forall}(\mathcal{D})$.
  Even for DTA's over 2 processes having 1 clock each.

Conclusion

Summary

- Distributed systems which synchronize using clocks with local times.
- Regular existential semantics suited for negative specifications
- Regular reactive semantics suited for positive specification
- Undecidable universal semantics

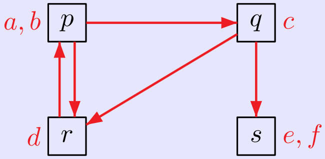
Conclusion

Summary

- Distributed systems which synchronize using clocks with local times.
- Regular existential semantics suited for negative specifications
- Regular reactive semantics suited for positive specification
- Undecidable universal semantics

Further work: Synthesis Problem

Given a regular specification $\mathrm{Spec} \subseteq \Sigma^*$ and an architecture A,
Construct a DTA $\mathcal{D}$ over A such that $\mathcal{L}_{\mathrm{react}}(\mathcal{D}) = \mathrm{Spec} = \mathcal{L}_{\exists}(\mathcal{D})$



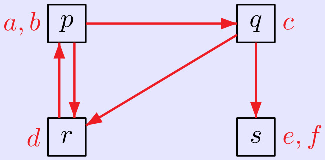
Conclusion

Summary

- Distributed systems which synchronize using clocks with local times.
- Regular existential semantics suited for negative specifications
- Regular reactive semantics suited for positive specification
- Undecidable universal semantics

Further work: Synthesis Problem

Given a regular specification $\mathrm{Spec} \subseteq \Sigma^*$ and an architecture A,
Construct a DTA $\mathcal{D}$ over A such that $\mathcal{L}_{\mathrm{react}}(\mathcal{D}) = \mathrm{Spec} = \mathcal{L}_{\exists}(\mathcal{D})$



If we are given two sets Good and Bad, find a DTA $\mathcal{D}$ such that

$$\mathrm{Good} \subseteq \mathcal{L}_{\mathrm{react}}(\mathcal{D}) \subseteq \mathcal{L}_{\exists}(\mathcal{D}) \subseteq \overline{\mathrm{Bad}}$$