

Distributed Timed Automata with Independently Evolving Clocks

Benedikt Bollig

LSV, ENS Cachan, CNRS

joint work with S. Akshay, Paul Gastin, Madhavan Mukund, K. Narayan Kumar

Groupe de travail Modélisation et Vérification
LaBRI, Bordeaux

September 16, 2010

Motivations

Aim

Verify properties of a distributed timed system.

- ▶ No explicit communication or synchronization.
- ▶ Clocks as a synchronization mechanism.
- ▶ Clocks on different processes evolve independently according to local times (which might depend on external factors).

Plan

① Distributed Timed Automata

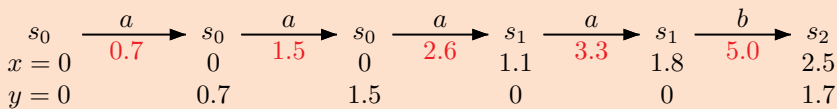
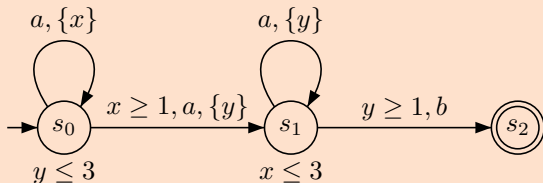
Existential semantics and region abstraction

Universal semantics and undecidability

Reactive (Game) Semantics

Timed automata (Alur & Dill)

Example: TA



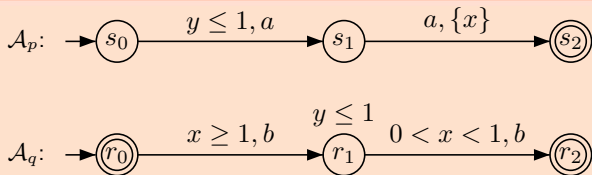
Distributed Timed automata

Definition: DTA

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

- ▶ each \mathcal{A}_p is a classical timed automaton
- ▶ $\pi : \mathcal{Z} \rightarrow Proc$ 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

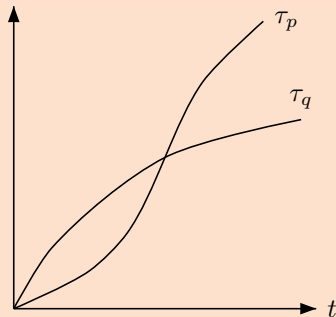
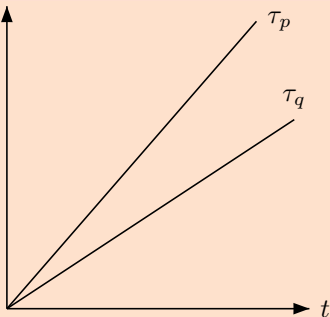
Local Times

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

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

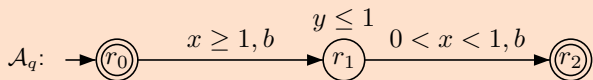
continuous, strictly increasing, diverging, $\tau_p(0) = 0$.

Example: Local Times



Runs of DTA & Untimed Behaviours

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

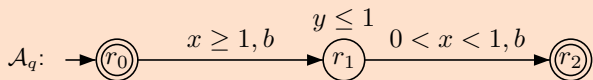
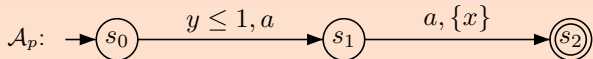


If $\tau_p > \tau_q$ then $abab \in \mathcal{L}(\mathcal{D}, \tau)$ (e.g. $\tau_p(t) = 2t$ and $\tau_q(t) = t$)

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

Runs of DTA & Untimed Behaviours

Example: DTA \mathcal{D} 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	$\xrightarrow{0.2}$	r_0	$\xrightarrow{0.6}$	r_1	$\xrightarrow{0.7}$	r_1	$\xrightarrow{0.8}$	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	$\xrightarrow{0.2}$	r_0	$\xrightarrow{0.5}$	r_1	$\xrightarrow{0.5}$	r_1
$x = 0$		0.4		1		0
$y = 0$		0.4		1		1

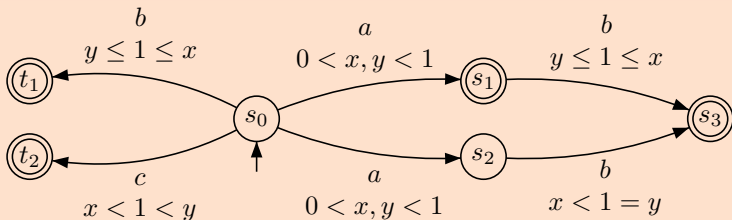
TA with independently evolving clocks

Definition: icTA

$\mathcal{B} = (\mathcal{A}, \pi)$ where

- ▶ \mathcal{A} is a timed automaton
- ▶ $\pi : \mathcal{Z} \rightarrow Proc$ assigns “processes” to clocks.
- ▶ If $\pi(x) = p$ then clock x evolves according to local time τ_p .

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



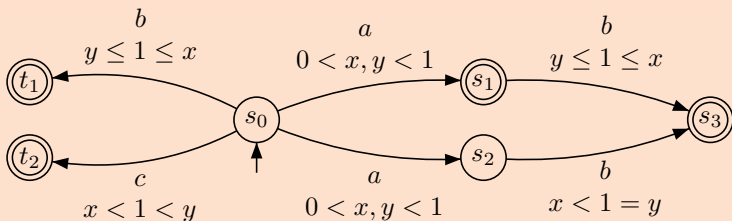
TA with independently evolving clocks

Definition: icTA

$\mathcal{B} = (\mathcal{A}, \pi)$ where

- ▶ \mathcal{A} is a timed automaton
- ▶ $\pi : \mathcal{Z} \rightarrow Proc$ assigns “processes” to clocks.
- ▶ If $\pi(x) = p$ then clock x evolves according to local time τ_p .

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

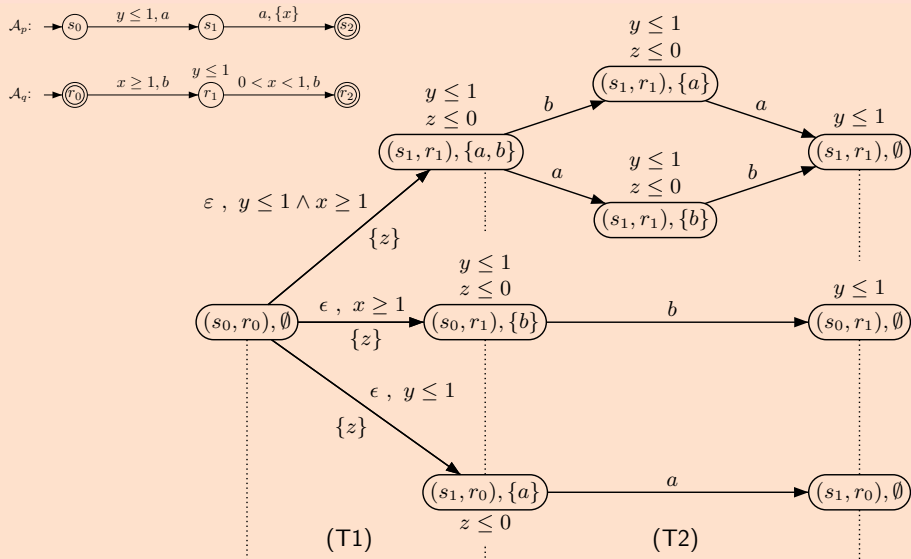


Remark: From DTA to icTA

Each DTA $\mathcal{D} = ((\mathcal{A}_p)_{p \in Proc}, \pi)$ can be viewed as an icTA $\mathcal{B} = (\mathcal{A}, \pi')$ where \mathcal{A} is the asynchronous product of $(\mathcal{A}_p)_{p \in Proc}$.

Semantics of DTA \mathcal{D}

Example: Part of the icTA $\mathcal{B}_{\mathcal{D}}$



Formal Semantics of icTA and DTA

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

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

- ▶ States are tuples (q, t, v) where
 - ▶ q is a state of \mathcal{A}
 - ▶ $t \in \mathbb{R}_{\geq 0}$ is the absolute time
 - ▶ $v : \mathcal{Z} \rightarrow \mathbb{R}_{\geq 0}$ is the valuation of clocks.

Formal Semantics of icTA and DTA

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

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

- ▶ States are tuples (q, t, v) where
 - ▶ q is a state of \mathcal{A}
 - ▶ $t \in \mathbb{R}_{\geq 0}$ is the absolute time
 - ▶ $v : \mathcal{Z} \rightarrow \mathbb{R}_{\geq 0}$ 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 icTA and DTA

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

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

- ▶ States are tuples (q, t, v) where
 - ▶ q is a state of \mathcal{A}
 - ▶ $t \in \mathbb{R}_{\geq 0}$ is the absolute time
 - ▶ $v : \mathcal{Z} \rightarrow \mathbb{R}_{\geq 0}$ 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: $(q, t, v) \xrightarrow{a} (q', t', v')$ if there are g and R such that
 - ▶ (q, g, a, R, q') is a transition of \mathcal{A} ,
 - ▶ $v_{t,t''} \models I(q)$ for all $t \leq t'' \leq t'$,
 - ▶ $v_{t,t'} \models g$
 - ▶ $v' = v_{t,t'}[R]$ (clocks in R are reset)
 - ▶ $v' \models I(q')$.

Formal Semantics of icTA and DTA

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

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

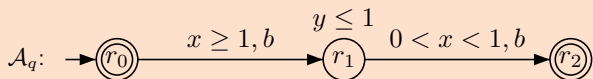
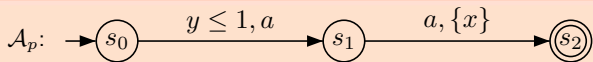
- ▶ States are tuples (q, t, v) where
 - ▶ q is a state of \mathcal{A}
 - ▶ $t \in \mathbb{R}_{\geq 0}$ is the absolute time
 - ▶ $v : \mathcal{Z} \rightarrow \mathbb{R}_{\geq 0}$ is the valuation of clocks.
- ▶ For $t < t'$ we define $v_{t,t'}(x) = v(x) + \tau_{\pi(x)}(t') - \tau_{\pi(x)}(t)$.
- ▶ Transitions: $(q, t, v) \xrightarrow{a} (q', t', v')$ if there are g and R such that
 - ▶ (q, g, a, R, q') is a transition of \mathcal{A} ,
 - ▶ $v_{t,t'} \models I(q)$ for all $t \leq t'' \leq t'$,
 - ▶ $v_{t,t'} \models g$
 - ▶ $v' = v_{t,t'}[R]$ (clocks in R are reset)
 - ▶ $v' \models I(q')$.
- ▶ $w = a_1 \dots a_n \in \mathcal{L}(\mathcal{B}, \tau)$ (with $a_i \in \Sigma \cup \{\varepsilon\}$) if there is a run in $TS(\mathcal{B}, \tau)$

$$(q_0, t_0, v_0) \xrightarrow{a_1} (q_1, t_1, v_1) \xrightarrow{a_2} \dots \xrightarrow{a_n} (q_n, t_n, v_n)$$

with q_0 initial, $t_0 = 0$, $v_0(x) = 0$ for all $x \in \mathcal{Z}$ and q_n final.

Semantics of DTA

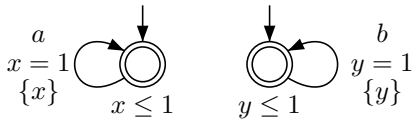
Example: DTA \mathcal{D} with $\pi(x) = p$ and $\pi(y) = q$



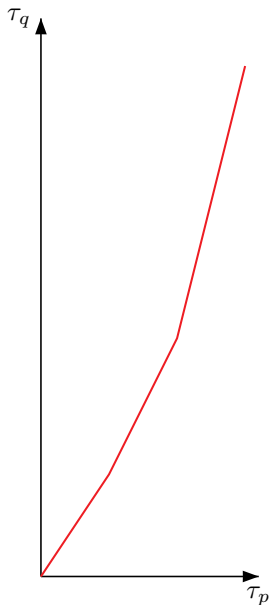
- ▶ If $\tau_p > \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa, abab, baab\}$.
- ▶ If $\tau_p = \tau_q$ then $\mathcal{L}(\mathcal{D}, \tau) = \{aa\}$.

Unregular Behaviours

Consider the following DTA \mathcal{D}

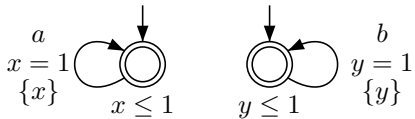


with $\pi(x) = p$ and $\pi(y) = q$
and the **local times** on the right.



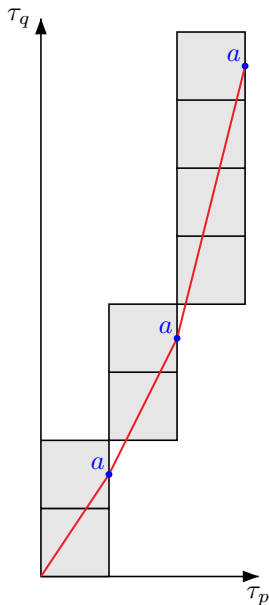
Unregular Behaviours

Consider the following DTA \mathcal{D}



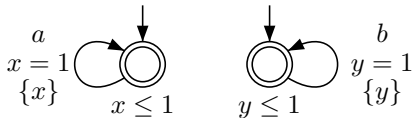
with $\pi(x) = p$ and $\pi(y) = q$
and the **local times** on the right.

a occurs every local time unit of p .



Unregular Behaviours

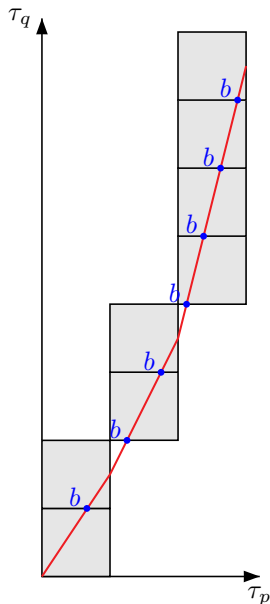
Consider the following DTA \mathcal{D}



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 .



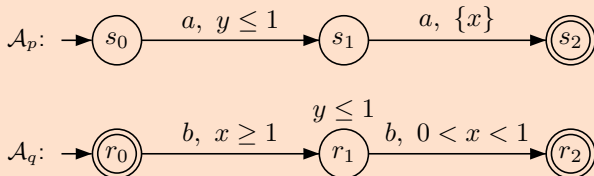
Existential & Universal Semantics

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**

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

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**

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

Definition: Positive Specifications (Liveness)

Given a set **Good** of desired behaviours,

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

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

Plan

Distributed Timed Automata

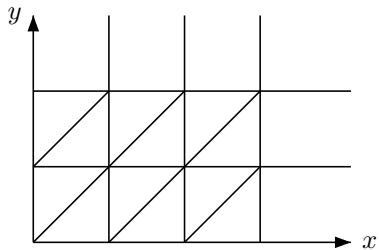
- 2 Existential semantics and region abstraction

Universal semantics and undecidability

Reactive (Game) Semantics

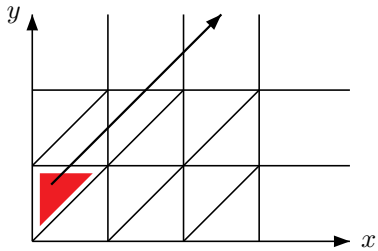
Region abstraction for \exists -semantics

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



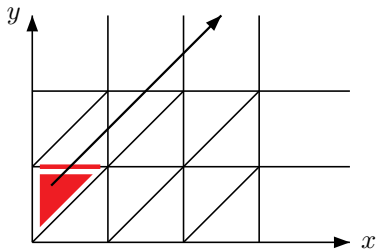
Region abstraction for \exists -semantics

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



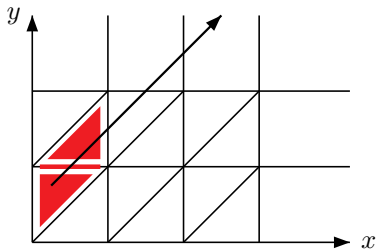
Region abstraction for \exists -semantics

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



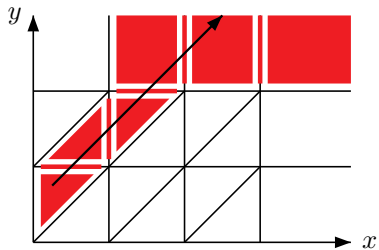
Region abstraction for \exists -semantics

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



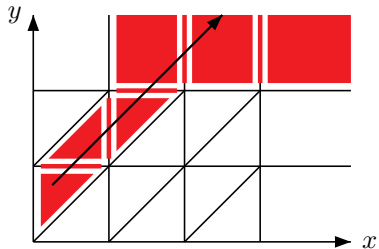
Region abstraction for \exists -semantics

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

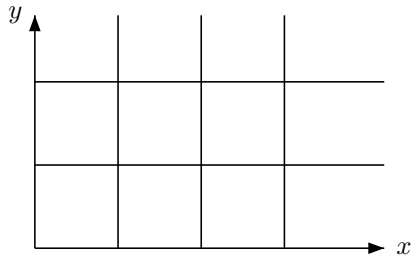


Region abstraction for \exists -semantics

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

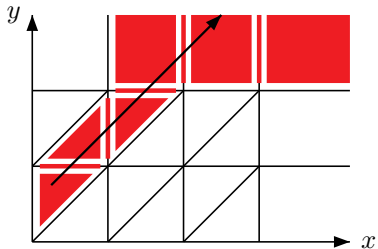


Regions when $\pi(x) \neq \pi(y)$

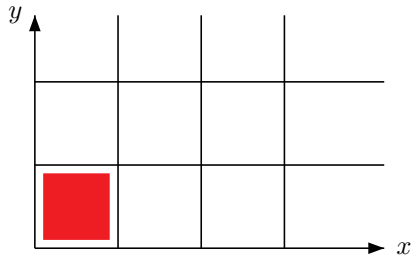


Region abstraction for \exists -semantics

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

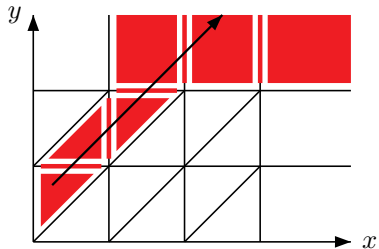


Regions when $\pi(x) \neq \pi(y)$

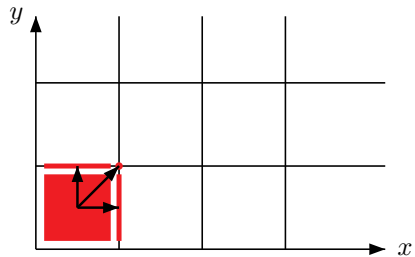


Region abstraction for \exists -semantics

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

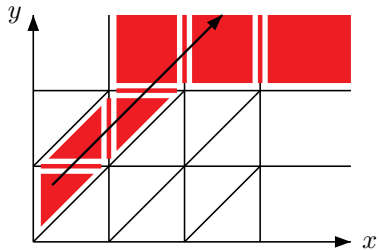


Regions when $\pi(x) \neq \pi(y)$

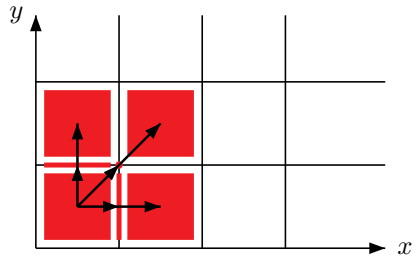


Region abstraction for \exists -semantics

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

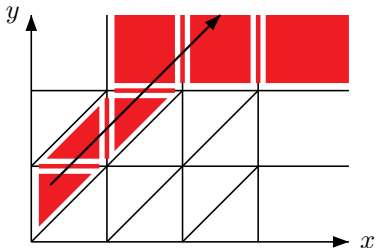


Regions when $\pi(x) \neq \pi(y)$

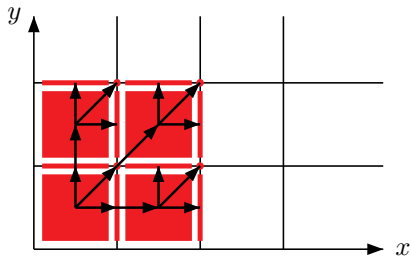


Region abstraction for \exists -semantics

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



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



Proposition:

The region equivalence is a **time-abstract bisimulation** for its \exists -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

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

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

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

Corollary: Negative specifications

Model checking **regular negative specifications** for DTA is decidable.

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

Plan

Distributed Timed Automata

Existential semantics and region abstraction

3 Universal semantics and undecidability

Reactive (Game) Semantics

Undecidability of the universal semantics

Theorem: Undecidability

Skip proof.

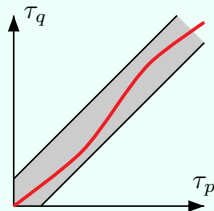
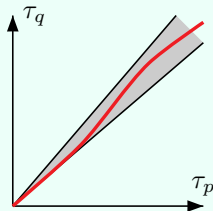
Let \mathcal{D} be a DTA.

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: $\exists \alpha > 0, \forall t > 0,$

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



Corollary: Positive specifications

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

Model checking **regular positive specifications** for DTA is **undecidable**.

Undecidability of emptiness

Proof: Reduction from Post Correspondence Problem

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

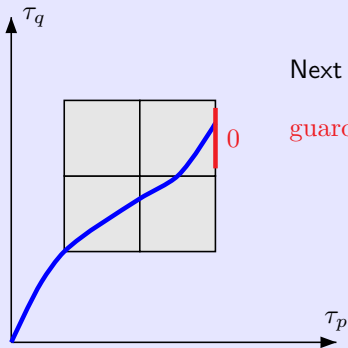
Undecidability of emptiness

Proof: Reduction from Post Correspondence Problem

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

Definition: Directions defined by local times

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



Next direction of $\text{dir}(\tau)$ is 0

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

$$G(\tau) = \{(\tau_p(t), \tau_q(t)) \mid t \in \mathbb{R}_{\geq 0}\}$$

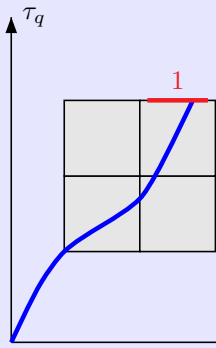
Undecidability of emptiness

Proof: Reduction from Post Correspondence Problem

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

Definition: Directions defined by local times

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



Next direction of $\text{dir}(\tau)$ is 1

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

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

$$G(\tau) = \{(\tau_p(t), \tau_q(t)) \mid t \in \mathbb{R}_{\geq 0}\}$$

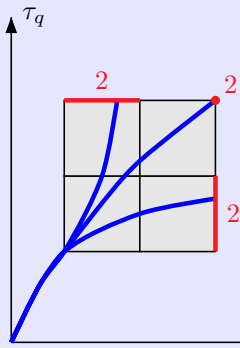
Undecidability of emptiness

Proof: Reduction from Post Correspondence Problem

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

Definition: Directions defined by local times

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



Next direction of $\text{dir}(\tau)$ is 2

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

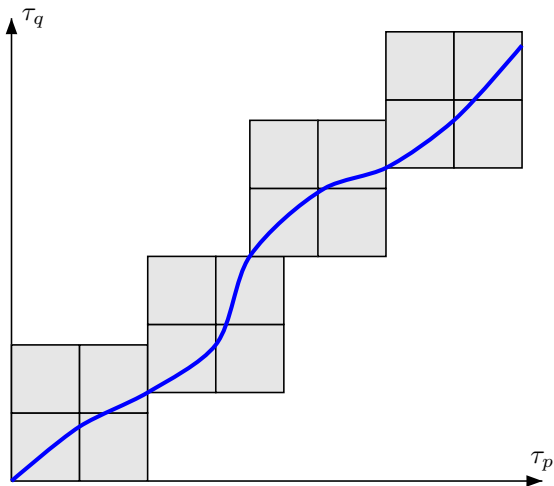
$$\text{guard}(1) := 1 < x < 2 \wedge y = 2$$

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

$$G(\tau) = \{(\tau_p(t), \tau_q(t)) \mid t \in \mathbb{R}_{\geq 0}\}$$

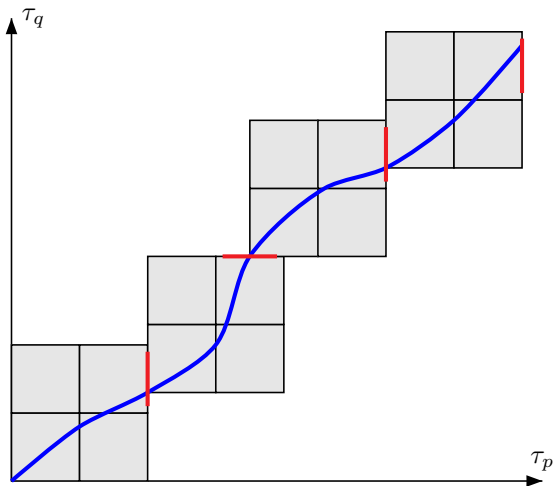
Directions defined by local times

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



Directions defined by local times

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



$$\text{dir}(\tau) = 0100 \dots$$

Undecidability of emptiness

Recall that we are given two morphisms

$$f, g : A^* \rightarrow \{0, 1\}^*$$

Undecidability of emptiness

Recall that we are given two morphisms

$$f, g : A^* \rightarrow \{0, 1\}^*$$

1. We construct icTA \mathcal{B}_f and \mathcal{B}_g such that for all local times $\tau = (\tau_p, \tau_q)$

$$L(\mathcal{B}_f, \tau) = \{ w \in A^+ \mid f(w) \not\leq \text{dir}(\tau) \}$$

$$L(\mathcal{B}_g, \tau) = \{ w \in A^+ \mid g(w) \leq \text{dir}(\tau) \}$$

Undecidability of emptiness

Recall that we are given two morphisms

$$f, g : A^* \rightarrow \{0, 1\}^*$$

1. We construct icTA \mathcal{B}_f and \mathcal{B}_g such that for all local times $\tau = (\tau_p, \tau_q)$

$$L(\mathcal{B}_f, \tau) = \{ w \in A^+ \mid f(w) \not\leq \text{dir}(\tau) \}$$

$$L(\mathcal{B}_g, \tau) = \{ w \in A^+ \mid g(w) \leq \text{dir}(\tau) \}$$

2. Let $\mathcal{B} = \mathcal{B}_f \vee \mathcal{B}_g$:

$$L_{\vee}(\mathcal{B}) = \{ w \in A^+ \mid f(w) = g(w) \}$$

Undecidability of emptiness

Recall that we are given two morphisms

$$f, g : A^* \rightarrow \{0, 1\}^*$$

1. We construct icTA \mathcal{B}_f and \mathcal{B}_g such that for all local times $\tau = (\tau_p, \tau_q)$

$$L(\mathcal{B}_f, \tau) = \{ w \in A^+ \mid f(w) \not\leq \text{dir}(\tau) \}$$

$$L(\mathcal{B}_g, \tau) = \{ w \in A^+ \mid g(w) \leq \text{dir}(\tau) \}$$

2. Let $\mathcal{B} = \mathcal{B}_f \vee \mathcal{B}_g$:

$$L_{\vee}(\mathcal{B}) = \{ w \in A^+ \mid f(w) = g(w) \}$$

3. Let $\mathcal{D} = \mathcal{B}[x] \parallel \mathcal{B}[y]$ where we add invariants $x \leq 1 \wedge y \leq 1$:

$$L_{\vee}(\mathcal{D}) = L_{\vee}(\mathcal{B})$$

Undecidability of existential semantics wrt. fixed slopes

Theorem:

Checking **emptiness of the \exists -semantics is undecidable** for DTA wrt. **fixed slopes**.

Undecidability of existential semantics wrt. fixed slopes

Theorem:

Checking **emptiness of the \exists -semantics is undecidable** for DTA wrt. **fixed slopes**.

Proof is by reduction from the following problem:

Theorem: Cassez & Henzinger & Raskin, 2002

The following problem is undecidable:

INPUT: Timed automaton.

QUESTION: Is there a rational number $\beta \geq 0$ such that some timed word is accepted where each time elapse equals β ?

Plan

Distributed Timed Automata

Existential semantics and region abstraction

Universal semantics and undecidability

4 Reactive (Game) Semantics

Semantics as a game

Existential semantics: 1-Player game

- ▶ Player 1 controls both transitions and time
- ▶ $\mathcal{L}_{\exists}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Semantics as a game

Existential semantics: 1-Player game

- ▶ Player 1 controls both transitions and time
- ▶ $\mathcal{L}_{\exists}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Universal semantics: 2-Player game

- ▶ Player 1 controls transitions
- ▶ Player 2 controls time
- ▶ $\mathcal{L}_{\forall}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Semantics as a game

Existential semantics: 1-Player game

- ▶ Player 1 controls both transitions and time
- ▶ $\mathcal{L}_{\exists}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Universal semantics: 2-Player game with imperfect information

- ▶ Player 1 controls transitions
- ▶ Player 2 controls time with imperfect information
- ▶ $\mathcal{L}_{\forall}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Semantics as a game

Existential semantics: 1-Player game

- ▶ Player 1 controls both transitions and time
- ▶ $\mathcal{L}_{\exists}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Universal semantics: 2-Player game with imperfect information

- ▶ Player 1 controls transitions
- ▶ Player 2 controls time with imperfect information
- ▶ $\mathcal{L}_{\forall}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

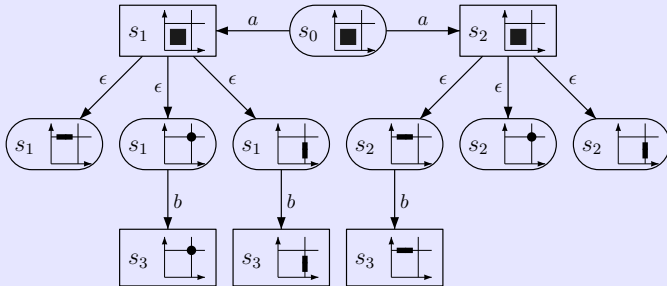
Reactive semantics: 2-Player game

- ▶ Player 1 controls transitions
- ▶ Player 2 controls time
- ▶ $\mathcal{L}_{\text{react}}(\mathcal{D}) = \{w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w\}$

Reactive (Game) Semantics

Definition: Reactive (Game) Semantics

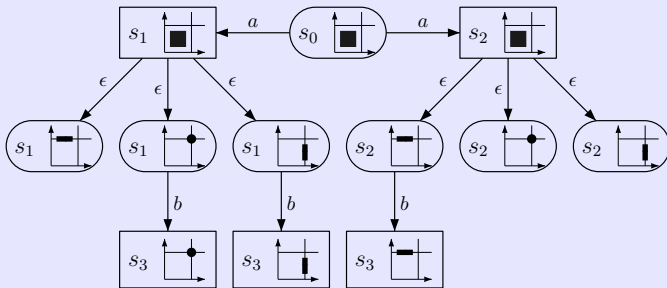
- ▶ System observes current region and controls discrete transitions
- ▶ Environment controls how local times evolve (time-elapse transitions)
- ▶ Not turn-based: system may execute several discrete transitions



Reactive (Game) Semantics

Definition: Reactive (Game) Semantics

- ▶ System observes current region and controls discrete transitions
- ▶ Environment controls how local times evolve (time-elapse transitions)
- ▶ Not turn-based: system may execute several discrete transitions



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

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

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

Conclusion

Summary

- ▶ Distributed system using clocks with local times to synchronize
- ▶ Regular existential semantics suited for negative specifications
- ▶ Regular reactive semantics suited for positive specifications
- ▶ Undecidable universal semantics

Conclusion

Summary

- ▶ Distributed system using clocks with local times to synchronize
- ▶ Regular existential semantics suited for negative specifications
- ▶ Regular reactive semantics suited for positive specifications
- ▶ Undecidable universal semantics

Further work: Synthesis Problem

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

