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

To appear in CONCUR'08

Logic & Algorithms
Edinburgh, 21-25 July 2008

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

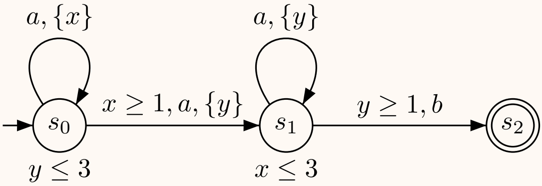
Region abstraction and existential semantics

Universal semantics and undecidability

Reactive (Game) Semantics

Timed automata (Alur & Dill)

Example: TA



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

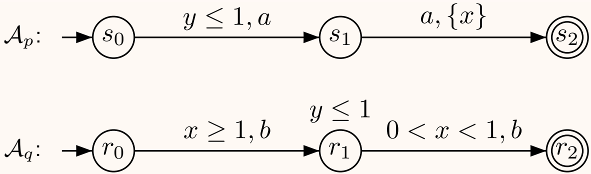
Distributed Timed automata

Definition: DTA

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

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

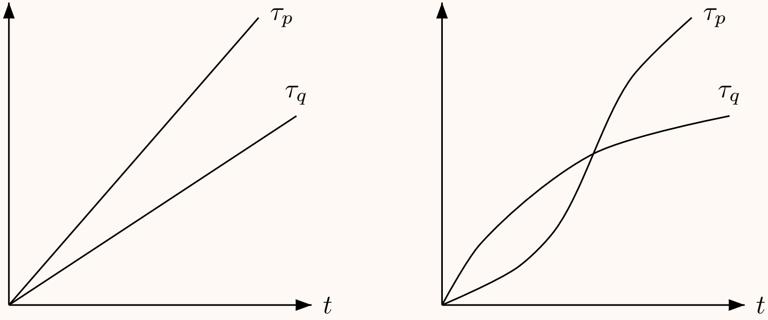
Local Times

- 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

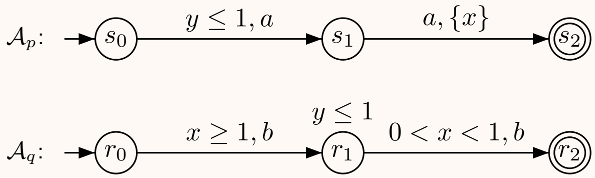
continuous, strictly increasing, diverging, $\tau_p(0) = 0$.

Example: Local Times



Runs of DTA's & Untimed Behaviours

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



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

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

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

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

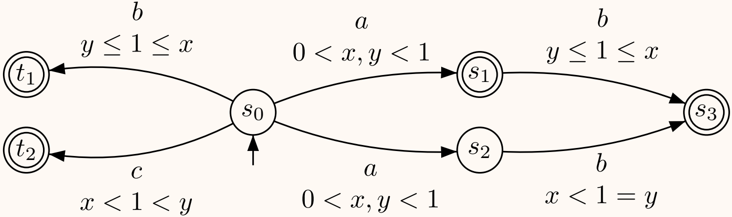
TA with independently evolving clocks

Definition: icTA

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

- $\mathcal{A}$ is a timed automaton
- $\pi : \mathcal{Z} \to Proc$ 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 $(\mathcal{A}_p)_{p \in Proc}$.

Formal Semantics of icTA's and DTA's

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

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

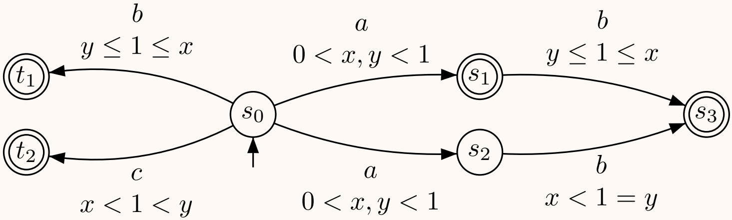
- States are tuples (q, t, v) where
  - q is a state of $\mathcal{A}$
  - $t \in \mathbb{R}_{\geq 0}$ is the absolute time
  - $v : \mathcal{Z} \to \mathbb{R}_{\geq 0}$ is the valuation of clocks.
- For $t < t'$ we define $v_{t,t'}(x) = v(x) + \tau_{\pi(x)}(t') - \tau_{\pi(x)}(t)$.
- Transitions : $(q, t, v) \xrightarrow{g,a,R} (q', t', v')$ if
  - $v_{t,t''} \models I(q)$ 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 $\mathrm{TS}(\mathcal{B}, \tau)$

$$(q_0, t_0, v_0) \xrightarrow{g_1,a_1,R_1} (q_1, t_1, v_1) \xrightarrow{g_2,a_2,R_2} \cdots \xrightarrow{g_n,a_n,R_n} (q_n, t_n, v_n)$$

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

Semantics of icTA's and DTA's

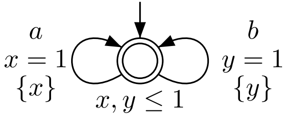
Example: icTA $\mathcal{B}$ with $\pi(x) = p$ and $\pi(y) = q$



- If $\tau_p = \tau_q$ then $b \in \mathcal{L}(\mathcal{B}, \tau)$ but $c \notin \mathcal{L}(\mathcal{B}, \tau)$.
- If $\tau_p < \tau_q$ then $b \notin \mathcal{L}(\mathcal{B}, \tau)$ but $c \in \mathcal{L}(\mathcal{B}, \tau)$.
- For all local times τ, we have $a \in \mathcal{L}(\mathcal{B}, \tau)$.
- For all local times τ, we have $ab \in \mathcal{L}(\mathcal{B}, \tau)$.

Unregular Behaviours

Consider the following icTA $\mathcal{B}$

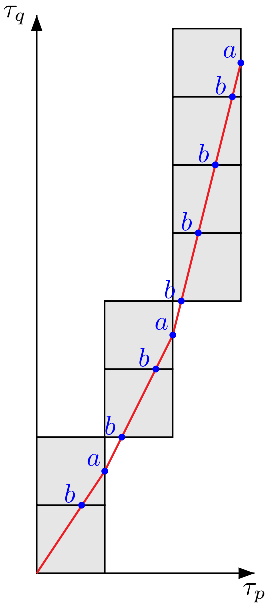


with $\pi(x) = p$ and $\pi(y) = q$
and the local times on the right.

a occurs every local time unit of p.

b occurs every local time unit of q.

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



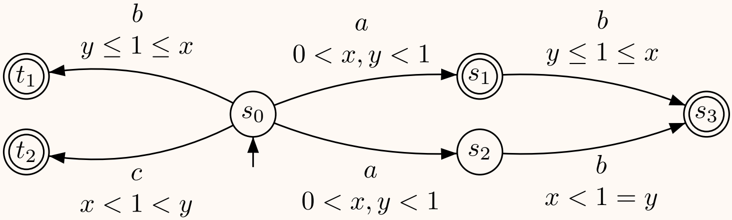
Existential & Universal Semantics

Definition: Existential & Universal Semantics

Let $\mathcal{B}$ be a DTA or an icTA.

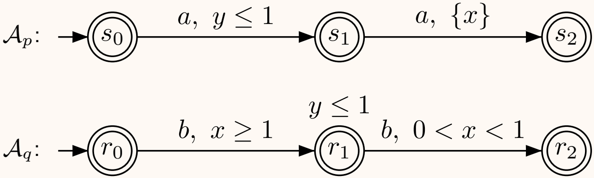
- $\mathcal{L}_\exists(\mathcal{B}) = \bigcup_\tau \mathcal{L}(\mathcal{B}, \tau)$
- $\mathcal{L}_\forall(\mathcal{B}) = \bigcap_\tau \mathcal{L}(\mathcal{B}, \tau)$

Example: $\mathcal{L}_\exists(\mathcal{B}) = \{a, b, c, ab\}$ $\quad\mathcal{L}_\forall(\mathcal{B}) = \{a, ab\}$



Existential & Universal Semantics

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



If $\tau_q < \tau_p$ then $baab \in \mathcal{L}(\mathcal{D}, \tau)$.

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

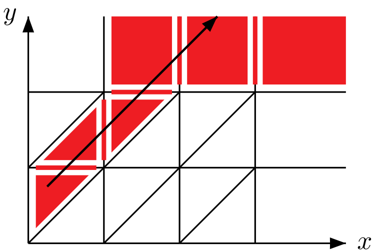
2. Region abstraction and existential semantics

Universal semantics and undecidability

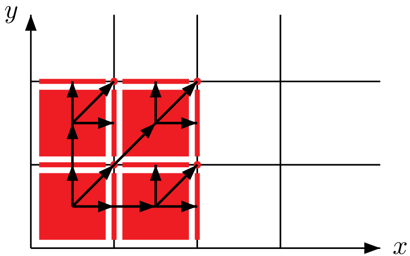
Reactive (Game) Semantics

Region abstraction for ∃-semantics

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



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



Region abstraction for ∃-semantics

Theorem: Region abstraction

Let $\mathcal{B}$ be an icTA (or a DTA). Let $\mathcal{R}_{\mathcal{B}}$ be its region abstraction.

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

and

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

Corollary: Negative specifications

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

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

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.

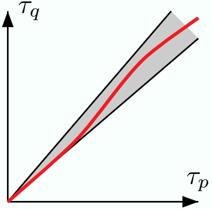
Let $\mathcal{D}$ be a DTA.

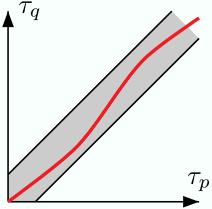
emptiness: $\mathcal{L}_\forall(\mathcal{D}) = \emptyset$ is undecidable.

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

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

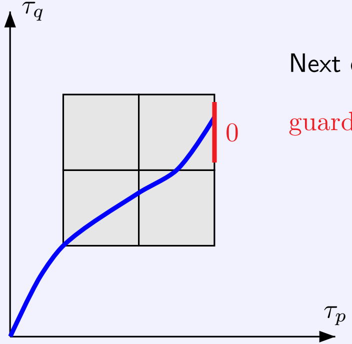
Undecidability of emptiness

Proof: Reduction from Post Correspondance Problem

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

Definition: Directions defined by local times

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



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

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

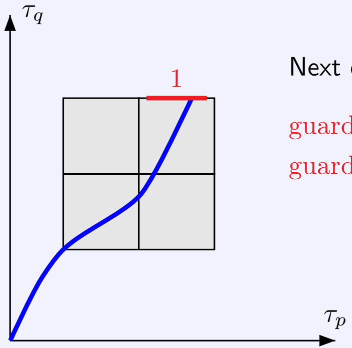
Undecidability of emptiness

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: Directions defined by local times

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



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

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

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

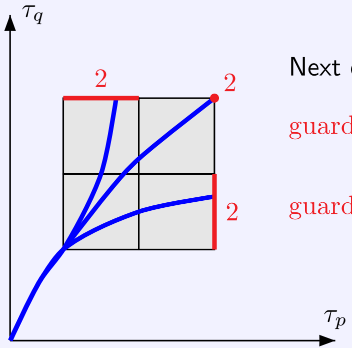
Undecidability of emptiness

Proof: Reduction from Post Correspondance Problem

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

Definition: Directions defined by local times

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



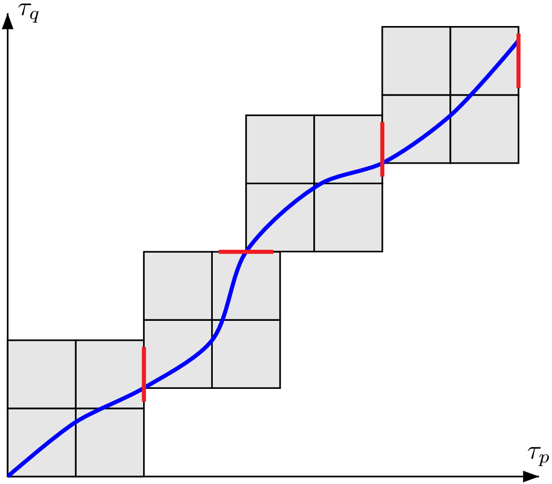
Next direction of $\mathrm{dir}(\tau)$ is 2

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

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

Directions defined by local times

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

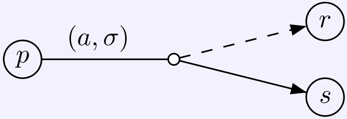


$$\mathrm{dir}(\tau) = 0100\cdots$$

Undecidability of emptiness

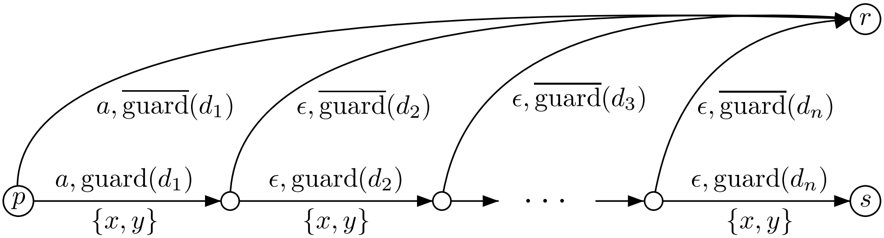
Definition: Macro transition

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

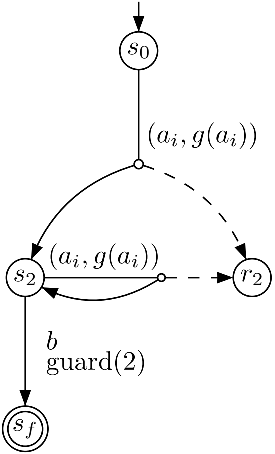


From p with $x = y = 0$, we reach

- s with $x = y = 0$ if local times τ evolve according to σ
- r otherwise



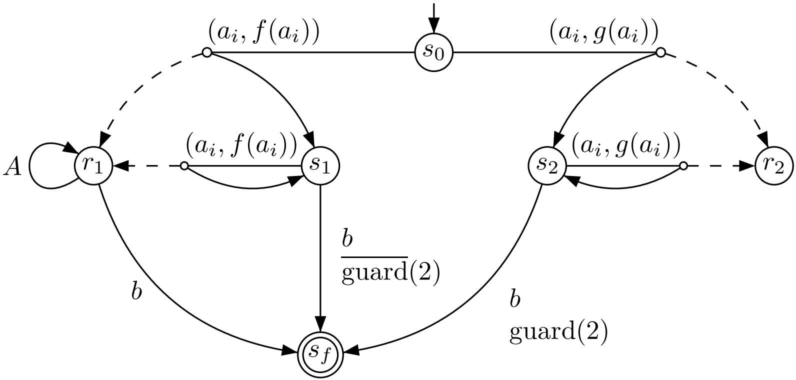
Undecidability of emptiness



Proposition:

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

Undecidability of emptiness



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

- $s_0 \xrightarrow{w} s_1$ 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

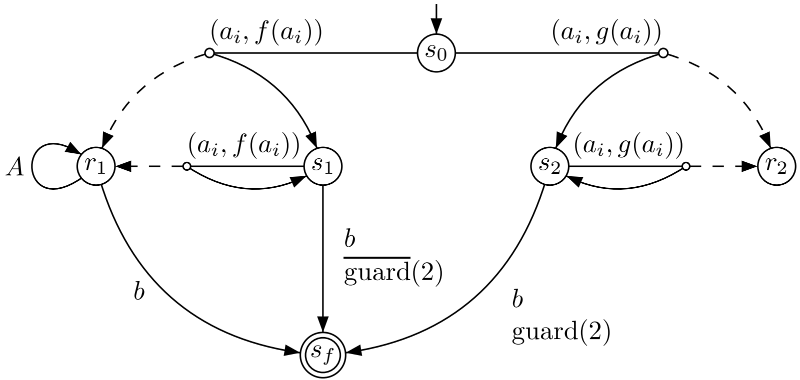
Universal semantics and undecidability

4. Reactive (Game) Semantics

Reactive (Game) Semantics

Remark: Positive Specifications and universal semantics

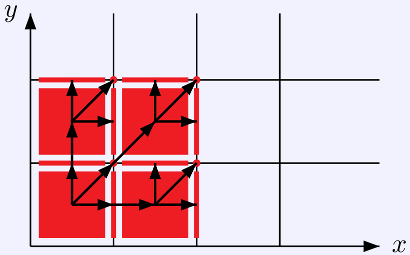
Good $\subseteq \mathcal{L}_\forall(\mathcal{D})$ does not imply that the system can be controlled in order to exhibit all Good behaviours, whatever local times are.



Reactive (Game) Semantics

Definition: Reactive (Game) Semantics

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



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

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

Decidability of the reactive semantics

Theorem: Regularity

Let $\mathcal{B}$ be an icTA or a DTA. $\mathcal{L}_{\text{react}}(\mathcal{B})$ is regular.

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

Corollary: Positive specifications

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

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

Proposition: Reactive vs. Universal

- $\mathcal{L}_{\text{react}}(\mathcal{B}) \subseteq \mathcal{L}_{\forall}(\mathcal{B})$ for all icTA's or DTA's $\mathcal{B}$.
- In general, $\mathcal{L}_{\text{react}}(\mathcal{B}) \subsetneq \mathcal{L}_{\forall}(\mathcal{B})$.
  Even for DTA's 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 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 $\mathcal{L}_{\mathrm{react}}(\mathcal{D}) = \mathrm{Spec}$