Timed Automata and Friends

. . . and What They Can (and Cannot) Do for You

Uli Fahrenberg

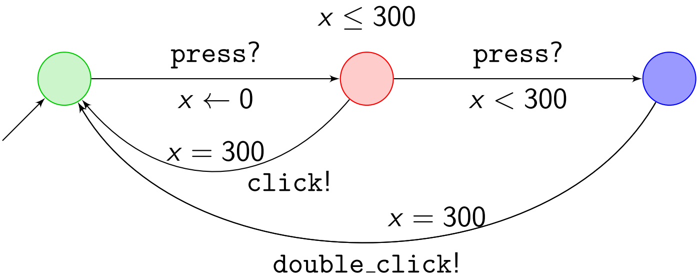
Inria Rennes (for now)

June 13, 2016

Timed Automata

- Invented by Rajeev Alur and David Dill (ICALP 1990 / TCS 1994)
- Popularized by Kim G. Larsen, Wang Yi and many others
- Robust tool support (TRL9): UPPAAL (Aalborg University, Denmark)
- in France: Cachan, Bordeaux, Grenoble
- for this talk: thanks to Kim G. Larsen, Claus Thrane, Patricia Bouyer, Nicolas Markey, and Benedikt Bollig

A Timed Automaton



Results

- Reachability is PSPACE-complete
- Emptiness is PSPACE-complete, universality is undecidable
- Decidability via regions; fast algorithms via zones
- Extensions: weighted timed automata; timed games

Overview

Timed Automata: Syntax

Definition

The set $\Phi(C)$ of clock constraints ϕ over a finite set C is defined by the grammar

$$\phi ::= x \bowtie k \mid x - y \bowtie k \mid \phi_1 \wedge \phi_2$$

$$(x, y \in C, k \in \mathbb{Z}, \bowtie \in \{\leq, <, \geq, >\}).$$

Definition

A timed automaton is a tuple $(L, \ell_0, C, \Sigma, I, E)$ consisting of a finite set L of locations, an initial location $\ell_0 \in L$, a finite set C of clocks, a finite set Σ of actions, a location invariants mapping $I : L \to \Phi(C)$, and a set $E \subseteq L \times \Phi(C) \times \Sigma \times 2^C \times L$ of edges.

Timed Automata: Semantics

Definition

The operational semantics of a timed automaton $A = (L, \ell_0, C, \Sigma, I, E)$ is the transition system $[\![A]\!] = (S, s_0, \Sigma \cup \mathbb{R}_{\geq 0}, T = T_s \cup T_d)$ given as follows:

$$S = \big\{(\ell, v) \in L \times \mathbb{R}_{\geq 0}^C \bigm| v \models I(\ell)\big\} \qquad s_0 = (\ell_0, v_0)$$

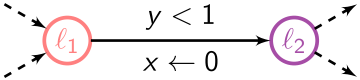
$$T_s = \big\{(\ell, v) \xrightarrow{a} (\ell', v') \bigm| \exists(\ell, \phi, a, r, \ell') \in E : v \models \phi, v' = v[r \leftarrow 0]\big\}$$

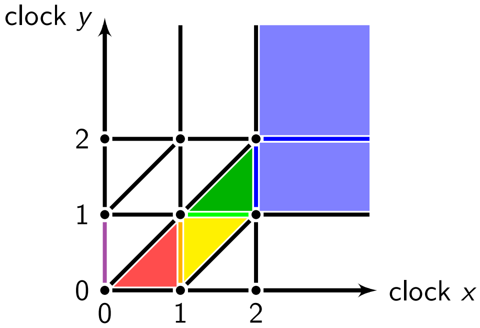
$$T_d = \big\{(\ell, v) \xrightarrow{d} (\ell, v + d) \bigm| \forall d' \in [0, d] : v + d' \models I(\ell)\big\}$$

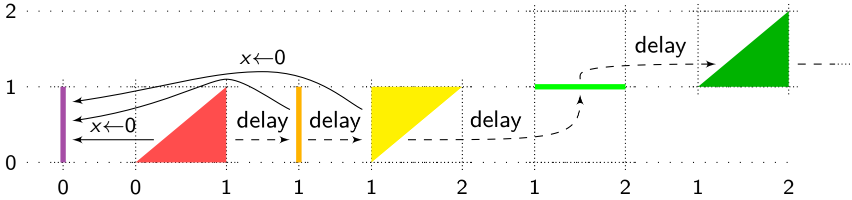
- $v \in \mathbb{R}_{\geq 0}^C$: clock valuation
- operations on clock valuations:

$$v[r \leftarrow 0](x) = \begin{cases} 0 & \text{if } x \in r \\ v(x) & \text{if } x \notin r \end{cases} \qquad (v + d)(x) = v(x) + d$$

Regions: Example







Regions: Definition

- Let $A = (L, \ell_0, C, \Sigma, I, E)$ be a timed automaton
- For each $x \in C$, let M_x be the maximal constant which x is compared to in A

Definition

Clock valuations $v, v' : C \to \mathbb{R}_{\geq 0}$ are region equivalent, denoted $v \cong v'$, if

- $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or $v(x), v'(x) > M_x$, for all $x \in C$, and
- $\langle v(x) \rangle = 0$ iff $\langle v'(x) \rangle = 0$, for all $x \in C$, and
- $\langle v(x) \rangle \leq \langle v(y) \rangle$ iff $\langle v'(x) \rangle \leq \langle v'(y) \rangle$ for all $x, y \in C$.

- $\lfloor v(x) \rfloor$: integer part; $\langle v(x) \rangle$: fractional part
- Extend to states by $(\ell, v) \cong (\ell', v')$ iff $\ell = \ell'$ and $v \cong v'$

Regions: Reachability

- Let $A = (L, \ell_0, C, \Sigma, I, E)$ be a timed automaton

Theorem

$\cong$ is a time-abstracted bisimulation: for all $(\ell_1, v_1) \cong (\ell_2, v_2)$:

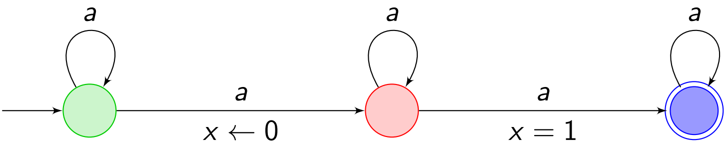
- *for all $(\ell_1, v_1) \xrightarrow{d} (\ell_1', v_1')$ there is $(\ell_2, v_2) \xrightarrow{d'} (\ell_2', v_2')$ such that $(\ell_1', v_1') \cong (\ell_2', v_2')$;*
- *for all $(\ell_1, v_1) \xrightarrow{a} (\ell_1', v_1')$ there is $(\ell_2, v_2) \xrightarrow{a} (\ell_2', v_2')$ such that $(\ell_1', v_1') \cong (\ell_2', v_2')$;*

and vice versa.

- Hence reachability can be decided in the quotient $[\![A]\!]/\cong$
- $[\![A]\!]/\cong$ is called the region automaton of A
- the number of states in $[\![A]\!]/\cong$ is bounded above by $|C|! \cdot 2^{|C|} \cdot \prod_{x \in C}(2M_x + 2)$, hence finite

Languages of Timed Automata

- A timed automaton accepts timed words: sequences $(a_1, t_1), (a_2, t_2), \dots$
  - symbols with time stamps: $t_1 \leq t_2 \leq \cdots$
- timed regular languages: closed under intersection and union, not under complement



- timed automata are not determinizable

Untimed Languages of Timed Automata

- $UL(A)$: untimed language of A
  - all projections of timed words to Σ^*
- Theorem: $UL(A) = L(\llbracket A \rrbracket / \cong)$
- Hence $UL(A)$ is regular
  - Conversely, for any L regular, there is a timed automaton A with $L = UL(A)$.
- Corollary: emptiness decidable; untimed regular model checking decidable

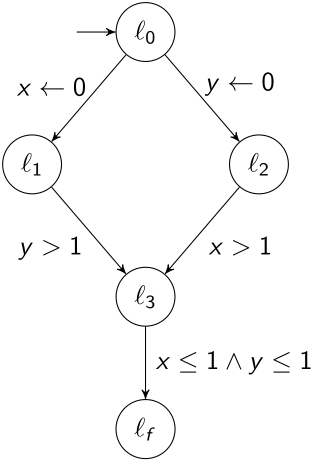
Zones: Definition

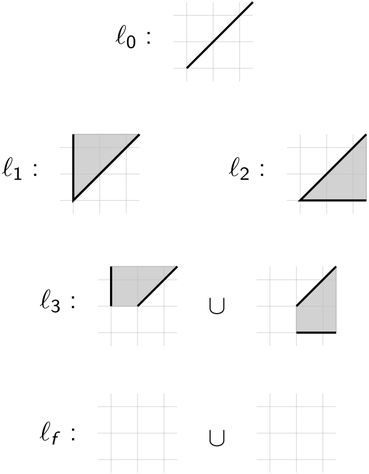
- The region automaton is too big to be practical
- All tools use zones: convex unions of regions
- Recall clock constraints:

$$\phi ::= x \bowtie k \mid x - y \bowtie k \mid \phi_1 \wedge \phi_2$$
$$(x, y \in C, k \in \mathbb{Z}, \bowtie \in \{\leq, <, \geq, >\}).$$

- The zone of ϕ: $[\![\phi]\!] = \{v : C \to \mathbb{R}_{\geq 0} \mid v \models \phi\}$
- (“half octagons”)

Zones: Example





Zones: Reachability Algorithm

Input: timed automaton $(L, \ell_0, C, \Sigma, I, E)$, $\ell_f \in L$

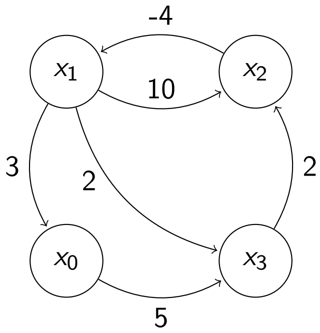
Output: **true** iff $\exists v_f : C \to \mathbb{R}_{\geq 0} : (\ell_0, v_0) \leadsto^* (\ell_f, v_f)$

$Waiting \leftarrow \{(\ell_0, \texttt{intersect}_{I(\ell_0)}(\texttt{delay}(\{v_0\})))\}$; $Passed \leftarrow \emptyset$
while $Waiting \neq \emptyset$ **do**
    Choose and remove (ℓ, v) from $Waiting$
    if $\ell = \ell_f$ **then**
        return true
    if (**not** $\texttt{is\_included}(v, v')$) for all $(\ell, v') \in Passed$ **then**
        $Passed \leftarrow Passed \cup \{(\ell, v)\}$
        for all $(\ell, \phi, a, r, \ell') \in E$ **do**
            $v' \leftarrow \texttt{intersect}_{I(\ell')}(\texttt{delay}(\texttt{reset}_r(\texttt{intersect}_\phi(v))))$
            if not $\texttt{is\_empty}(v')$ **then**
                $Waiting \leftarrow Waiting \cup \{(\ell', v')\}$
return false

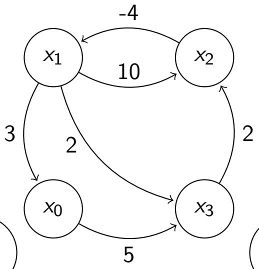
Zones: Representation

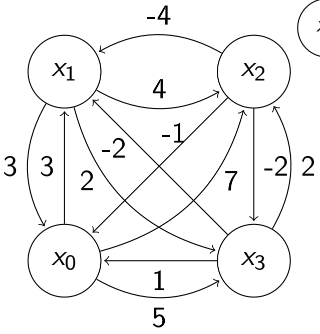
Zone $\rightsquigarrow$ digraph $\cong$ difference-bound matrix

$$Z = \begin{cases} x_1 \leq 3 \\ x_1 - x_2 \leq 10 \\ x_1 - x_2 \geq 4 \\ x_1 - x_3 \leq 2 \\ x_3 - x_2 \leq 2 \\ x_3 \geq -5 \end{cases}$$

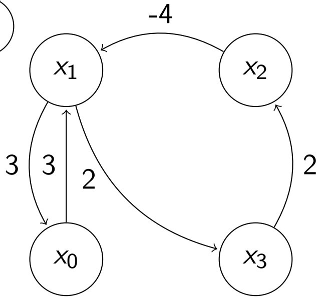


Zones: Representation





shortest-path closure



shortest-path reduction

Zones: Algorithms

- Using closures or reductions
- Delay, reset, intersection, inclusion check can be done in $O(|C|^3)$
- In practice: combined Passed-Waiting list
- Each location has a list of zones ($\cong$ union)
- Represented using clock decision diagrams
- Extract DBMs from CDD $\rightsquigarrow$ perform operations on each $\rightsquigarrow$ re-combine to new CDD
- Use max-plus polyhedra instead of zones? (Probably not!)

Timed Automata

- Useful for modeling synchronous real-time systems
- Reachability, emptiness, LTL model checking PSPACE-complete
- Universality undecidable
- Decidability via regions; undecidability via two-counter machines
- Fast on-the-fly algorithms, using zones, for reachability, liveness, and Timed CTL model checking
- Next: Extensions

Extensions

- More clock constraints, e.g. $x + y \bowtie k$: reachability undecidable
- Stopwatches: reachability undecidable
- Rectangular hybrid automata: reachability undecidable
  - Initialized rectangular automata: reachability decidable, but no zone-based algorithms

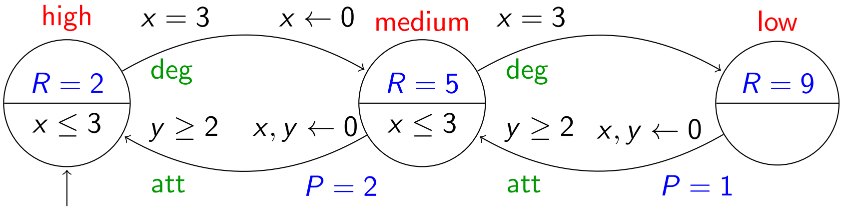
→ Weighted timed automata:

- Optimal reachability decidable; on-the fly zone algorithm
- Same for conditional optimal reachability for multi-weights
- Also other problems decidable, but no zones

→ Timed games:

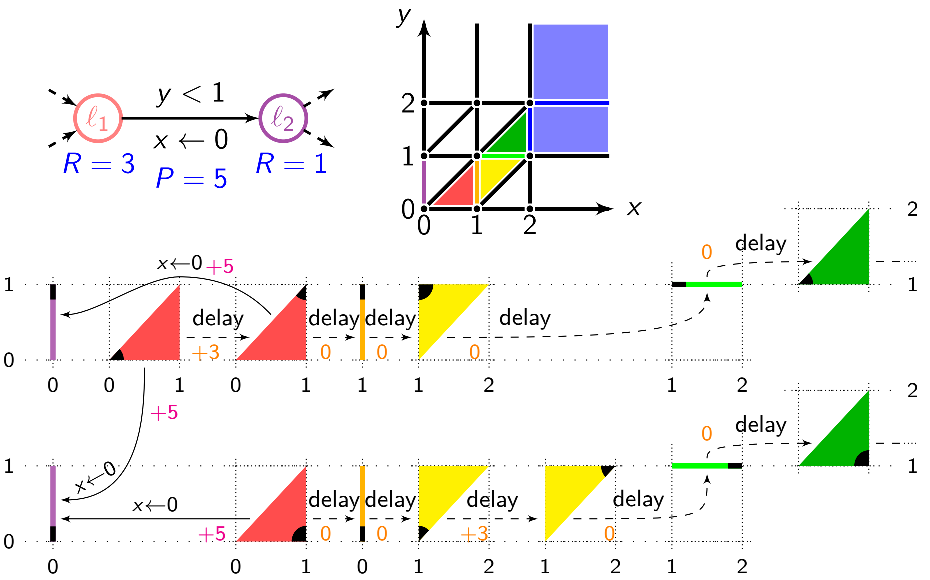
- Reachability and safety games decidable; on-the fly zone algorithm, but slow
- Also for partial observability
- Weighted timed games: very difficult

Weighted Timed Automata



- Models a plant with three modes of production
- Goal: incur lowest long-term cost
- Minimal cost-per-time: computable, but uses corner-point abstraction (finer than regions)
- No zone-based algorithm
- Same for minimal discounted cost

Corner-Point Abstraction: Example



Optimal Reachability

Problem

Given a weighted timed automaton A and $\epsilon > 0$, compute $W = \inf\{w(\rho) \mid \rho$ accepting run in $A\}$ and an accepting run ρ for which $w(\rho) < W + \epsilon$.

Theorem

The optimal reachability problem for weighted timed automata with non-negative weights is PSPACE-complete.

- Corollary: Time-optimal reachability for timed automata is also PSPACE-complete
- Fast on-the-fly algorithms using weighted zones: zones with affine cost functions
- But weighted zones may need to be split during exploration

Conditional Optimal Reachability

Problem

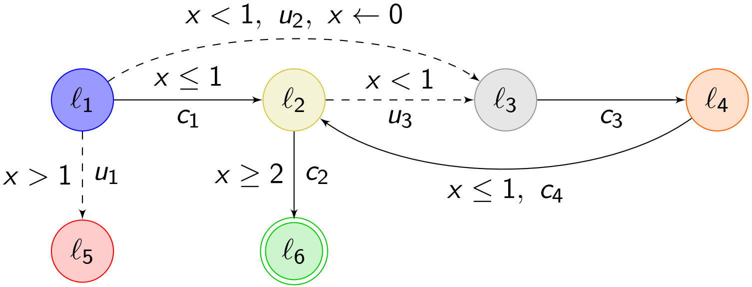
Given a doubly weighted timed automaton A, $M \in \mathbb{Z}$, and $\epsilon > 0$, compute $W = \inf \{ w_1(\rho) \mid \rho$ accepting run in $A, w_2(\rho) \leq M \}$ and an accepting run ρ for which $w_2(\rho) \leq M$ and $w_1(\rho) < W + \epsilon$.

Theorem

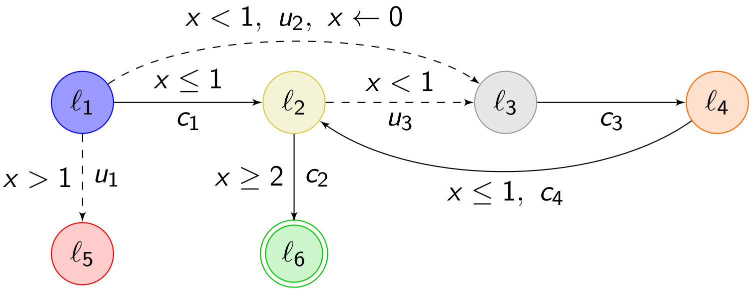
The conditional optimal reachability problem is computable for doubly weighted timed automata with non-negative weights.

- Can also compute Pareto frontier
- Fast on-the-fly algorithms using doubly weighted zones

Timed Games


$x < 1,\ u_2,\ x \leftarrow 0$
$x \leq 1$
c_1
$x < 1$
u_3
c_3
ℓ_1
ℓ_2
ℓ_3
ℓ_4
$x > 1$
u_1
$x \geq 2$
c_2
$x \leq 1,\ c_4$
ℓ_5
ℓ_6

Timed Games



Winning strategy:

$$\sigma(\ell_1, v) = \begin{cases} \delta & \text{if } v(x) \neq 1 \\ c_1 & \text{if } v(x) = 1 \end{cases} \qquad \sigma(\ell_2, v) = \begin{cases} \delta & \text{if } v(x) < 2 \\ c_2 & \text{if } v(x) \geq 2 \end{cases}$$

$$\sigma(\ell_3, v) = \begin{cases} \delta & \text{if } v(x) < 1 \\ c_3 & \text{if } v(x) \geq 1 \end{cases} \qquad \sigma(\ell_4, v) = \begin{cases} \delta & \text{if } v(x) \neq 1 \\ c_4 & \text{if } x(x) = 1 \end{cases}$$

Reachability and Safety Games

Lemma

If the player has a winning strategy in the reachability or safety game, then she has a memoryless winning strategy.

Theorem

The reachability and safety games for timed games are EXPTIME-complete.

- Same for time-optimal reachability and safety games
- On-the-fly algorithm using zones
- Forward and backwards exploration
- Needs to compute differences of zones $\rightsquigarrow$ state space explosion
- Use max-plus polyhedra instead of zones?

References

- P. Bouyer, U.F., K.G. Larsen, N. Markey. *Quantitative analysis of real-time systems*. Communications of the ACM, 54(9):78-87, 2011.
- U.F., K.G. Larsen, A. Legay. *Model-Based Verification, Optimization, Synthesis and Performance Evaluation of Real-Time Systems*. In Unifying Theories of Programming and Formal Engineering Methods, LNCS 8050, Springer 2013.
- X. Allamigeon, U.F., S. Gaubert, R. Katz, A. Legay. *Tropical Fourier-Motzkin Elimination, with an Application to Real-Time Verification*. International Journal of Algebra and Computation 24(5):569-607, 2014

Distributed Timed Automata with Independently Evolving Clocks

S. Akshay, B. Bollig, P. Gastin, M. Mukund, K.N. Kumar, Fundamenta Informaticae 130(4): 377-407, 2014

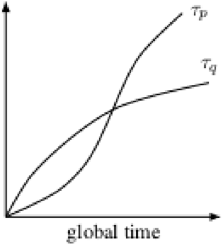
- Product of timed automata: Let $A_1 = (L_1, \ell_0^1, C_1, \Sigma_1, I_1, E_1)$, $A_2 = (L_2, \ell_0^2, C_2, \Sigma_2, I_2, E_2)$. Then $A_1 \times A_2 = (L_1 \times L_2, (\ell_0^1, \ell_0^2), C_1 \sqcup C_2, I, E)$, with

$$I(\ell_1, \ell_2) = I_1(\ell_1) \wedge I_2(\ell_2)$$

$$E = \{((\ell_1, \ell_2), \phi, a, r, (\ell_1', \ell_2)) \mid (\ell_1, \phi, a, r, \ell_1') \in E_1\}$$
$$\cup \{((\ell_1, \ell_2), \phi, a, r, (\ell_1, \ell_2')) \mid (\ell_2, \phi, a, r, \ell_2') \in E_2\}$$

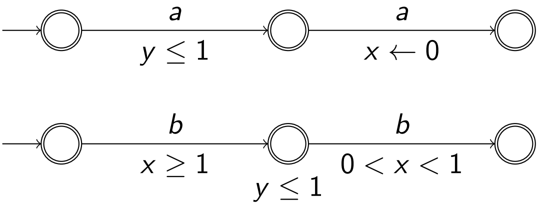
- Can be combined with different types of action synchronization
- Popular specification formalism e.g. in UPPAAL
- Clocks are synchronized

Distributed Timed Automata



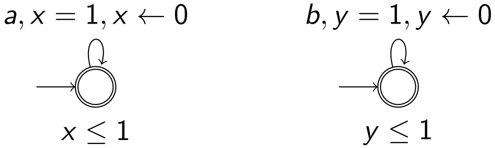
- Network $A = (A_1, \dots, A_n)$ of timed automata
- Together with local time rates $\tau_1, \dots, \tau_n : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$
  - all τ_i continuous, strictly increasing, diverging, with $\tau_i(0) = 0$
- Clocks in C_i can appear in constraints in all A_j, but can only be reset in A_i
  - i.e. $E_i \subseteq L_i \times \Phi(C_1 \sqcup \dots \sqcup C_n) \times \Sigma \times 2^{C_i} \times L_i$
  - (precise formalization in the paper is slightly different)
- $\tau_i = \text{id}$ for all i: standard product of timed automata
- Paper considers only untimed languages, for different types of clock synchronization constraints

Example

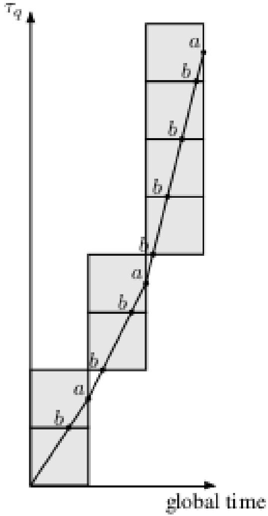


- $L_{\text{sync}} = \{\epsilon, a, aa, b, ab, ba, aba, baa, aab\}$
- x slower than y: $L = \{\epsilon, a, aa\}$
- x faster than y:
  $L = \{\epsilon, a, aa, b, ab, ba, aba, baa, aab, abab, baab\}$
- $L_{\exists} = \{\epsilon, a, aa, b, ab, ba, aba, baa, aab, abab, baab\}$
- $L_{\forall} = \{\epsilon, a, aa\}$

Non-Regular Behavior



- $\tau_2(t) \approx 2^t - .5$
- $L = \text{Pref}(bab^2ab^4ab^8a\dots)$



Existential Semantics

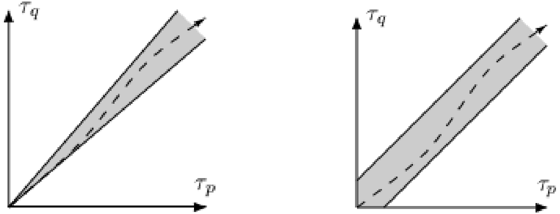
- $A = (A_1, \dots, A_n)$ network of timed automata
- For $\tau = (\tau_1, \dots, \tau_n)$ local time rates:
  $L(A, \tau) :=$ untimed language of A given τ
- $L_\exists(A) = \bigcup_\tau L(A, \tau)$
- Theorem: $L_\exists(A)$ is regular and can be obtained via a modified region construction
- Corollary: emptiness and regular model checking are decidable for the existential semantics

Universal Semantics

- $L_\forall(A) = \bigcap_\tau L(A, \tau)$
- Theorem: emptiness and universality undecidable
- Corollary: regular model checking undecidable

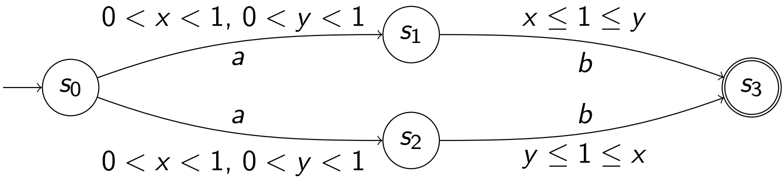
Bounded Clock Drift

- Restrict to two timed automata: $A = (A_1, A_2)$, $\tau = (\tau_1, \tau_2)$
- For $k \geq 1$: $R^{\text{rat}\leq k} = \left\{\tau \mid \forall t > 0 : \frac{1}{k} \leq \frac{\tau_1(t)}{\tau_2(t)} \leq k\right\}$
- For $d \geq 0$: $R^{\text{diff}\leq d} = \{\tau \mid \forall t > 0 : |\tau_1(t) - \tau_2(t)| \leq d\}$



- $L_\forall^{\text{rat}\leq 1}(A) = L_\forall^{\text{diff}\leq 0}(A) = UL(A)$, hence regular
- For $k > 1$, emptiness and universality of $L_\forall^{\text{rat}\leq k}(A)$ undecidable
- For $d > 0$, emptiness and universality of $L_\forall^{\text{diff}\leq d}(A)$ undecidable
- Nothing known about $L_\exists^{\text{rat}\leq k}(A)$ and $L_\exists^{\text{diff}\leq d}(A)$

Reactive Semantics



- Problem: $L_\forall(A) = \{ab\}$, but either through s_1 or s_2, depending on future local time rates
- Need to "know" future local time rates when deciding whether to go to s_1 or s_2
- Solution: reactive semantics $L_{\text{react}}(A)$: "choose future local time rates only when it's time"
  - (Formalization using games on region automaton; complicated)
- $L_{\text{react}}(A)$ is regular

Distributed Timed Automata with Independent Clocks

- Clocks within a component evolve in sync; clocks in different components are independent
- Untimed semantics: $L_{\text{react}} \subseteq L_\forall \subseteq UL \subseteq L_\exists$
- Useful: bounds on clock drift: $R^{\text{rat}\le k}$, $R^{\text{diff}\le d}$
- $L_\forall$, $L_\forall^{\text{rat}\le k}$ and $L_\forall^{\text{diff}\le d}$ seem difficult to work with
- L_{react} and $L_\exists$ are regular
- Nothing known about $L_{\text{react}}^{\text{rat}\le k}$, $L_{\text{react}}^{\text{diff}\le d}$, $L_\exists^{\text{rat}\le k}$, and $L_\exists^{\text{diff}\le d}$

- Useful as a starting point for distributed hybrid systems
- We also care about timed semantics
- For hybrid systems, we're beyond undecidability
- But zones are nice!