When are timed automata determinizable?

C. Baier[1], N. Bertrand[2], P. Bouyer[3], T. Brihaye[4]

[1]Technische Universität Dresden – Germany

[2]INRIA Rennes Bretagne Atlantique – France

[3]LSV – CNRS & ENS Cachan – France

[4]Université de Mons – Belgium

DistribCom Meeting – February 27th 2009

Outline

1. Timed automata
2. A determinization procedure
   - Unfolding into an infinite tree
   - Region equivalence
   - Symbolic determinization
   - Clock reduction
   - Location reduction
3. Summary

Syntax and semantics

Timed automata

A timed automaton is a tuple $\mathcal{A} = (L, \Sigma, X, E)$ with

- L finite set of locations
- X finite set of clocks
- Σ finite alphabet
- $E \subseteq L \times \Sigma \times \mathcal{G} \times 2^X \times L$ set of edges

where $\mathcal{G} = \{\bigwedge x \sim c \mid x \in X, c \in \mathbb{N}\}$ is the set of guards.

States of $\mathcal{A}$: $L \times (\mathbb{R}_+)^X$

Transitions between states of $\mathcal{A}$:

- Delay transitions: $(\ell, v) \xrightarrow{t} (\ell, v + t)$
- Discrete transitions: $(\ell, v) \xrightarrow{a} (\ell', v')$ if $\exists(\ell, a, g, Y, \ell') \in E$ with $v \models g$, $v'(x) = 0$ if $x \in Y$, and $v'(x) = v(x)$ otherwise.

Run of $\mathcal{A}$:

$(\ell_0, v_0) \xrightarrow{\tau_0} (\ell_0, v_0 + \tau_0) \xrightarrow{a_0} (\ell_1, v_1) \xrightarrow{\tau_1} (\ell_1, v_1 + \tau_1) \xrightarrow{a_1} (\ell_2, v_2) \ldots$

or simply: $(\ell_0, v_0) \xrightarrow{\tau_0, a_0} (\ell_1, v_1) \xrightarrow{\tau_1, a_1} (\ell_2, v_2) \ldots$

Timed language

Timed word: $w = (a_0, t_0)(a_1, t_1) \dots (a_k, t_k)$
with $a_i \in \Sigma$ and $(t_i)_{0 \le i \le k}$ nondecreasing sequence in $\mathbb{R}_+$.

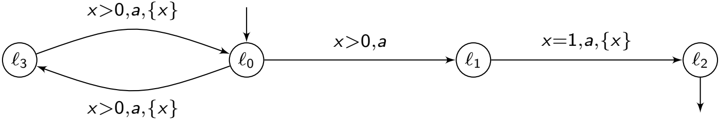
$\mathcal{A} = (L, \ell_0, L_{acc}, \Sigma, X, E)$ timed automaton equipped with ℓ_0 initial location, and L_{acc} set of accepting locations.

Accepted timed word

A timed word $w = (a_0, t_0)(a_1, t_1) \dots (a_k, t_k)$ is accepted in $\mathcal{A}$, if there is a run $\rho = (\ell_0, v_0) \xrightarrow{\tau_0, a_0} (\ell_1, v_1) \xrightarrow{\tau_1, a_1} \dots (\ell_{k+1}, v_{k+1})$ in $\mathcal{A}$ with $\ell_{k+1} \in L_{acc}$, and $t_i = \sum_{j<i} \tau_j$.

Accepted timed language: $\mathcal{L}(\mathcal{A}) = \{w \mid w \text{ accepted by } \mathcal{A}\}$.

A running example



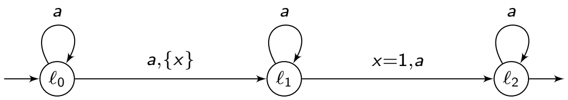
$$\mathcal{L}(A) = \{(a, t_1)(a, t_2)\cdots(a, t_{2n}) \mid 0 < t_1 < t_2 < \cdots < t_{2n-1} \text{ and } t_{2n} - t_{2n-2} = 1\}$$

Deterministic timed automata

Deterministic timed automata

$\mathcal{A}$ is deterministic whenever for every timed word w, there is at most one initial run on w in $\mathcal{A}$.

Some timed automata are not determinizable [AD90].



$\mathcal{L}(\mathcal{A}) = \{(a, t_1) \dots (a, t_n) \mid n \geq 2 \text{ and } \exists i < j \text{ s.t. } t_j - t_i = 1\}$
$\longrightarrow$ infinitely many clocks needed

Theorem [Finkel 06]

Checking whether a given timed automata is determinizable is undecidable.

About universality

$\mathcal{A}$ is universal if $\mathcal{L}(\mathcal{A}) = (\Sigma \times \mathbb{R}_+)^*$

Theorem [AD90]

Universality is undecidable for timed automata.

However, universality is decidable for some subclasses

- event-clock timed automata [AFH94]
- one-clock timed automata [OW04]

Event-clock timed automata

For every $a \in \Sigma$ there is a clock x_a reset at each occurrence of a.

Strong timed bisimulation

Strong timed (bi)simulation

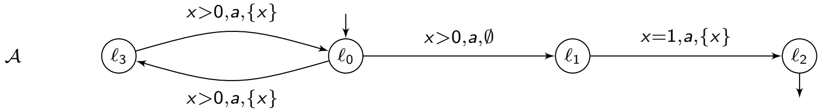
$\mathfrak{R}$ is a strong timed simulation between transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$ if for every $s_1 \; \mathfrak{R} \; s_2$ and $s_1 \xrightarrow{t_1,a} s_1'$ for some $t_1 \in \mathbb{R}_+$ and $a \in \Sigma$, then there exists $s_2' \in S_2$ such that $s_2 \xrightarrow{t_1,a} s_2'$ and $s_1' \; \mathfrak{R} \; s_2'$.
$\mathfrak{R}$ is a strong timed bisimulation if $\mathfrak{R}$ and $\mathfrak{R}^{-1}$ are strong timed simulations.

Strong timed bisimulation (preserving initial and accepting states) implies language equivalence.

Outline

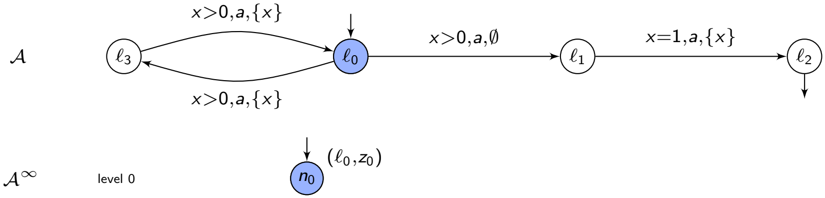
Unfolding

- $\mathcal{A}$ unfolded into a tree $\mathcal{A}^\infty$ with a fresh clock at each step.
- clocks of $\mathcal{A}$ are mapped to their reference in the new set of clocks.



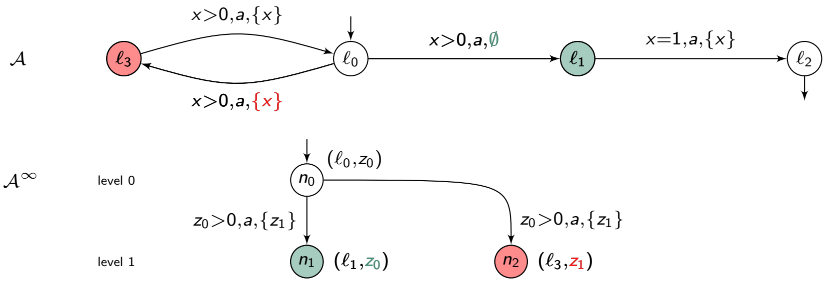
Unfolding

- $\mathcal{A}$ unfolded into a tree $\mathcal{A}^\infty$ with a fresh clock at each step.
- clocks of $\mathcal{A}$ are mapped to their reference in the new set of clocks.



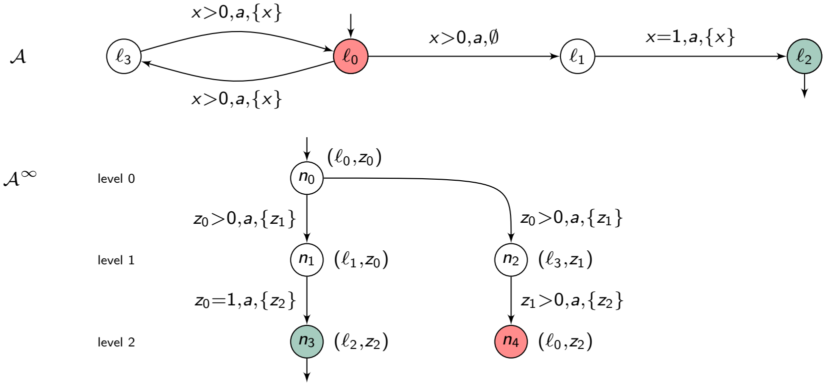
Unfolding

- $\mathcal{A}$ unfolded into a tree $\mathcal{A}^\infty$ with a fresh clock at each step.
- clocks of $\mathcal{A}$ are mapped to their reference in the new set of clocks.

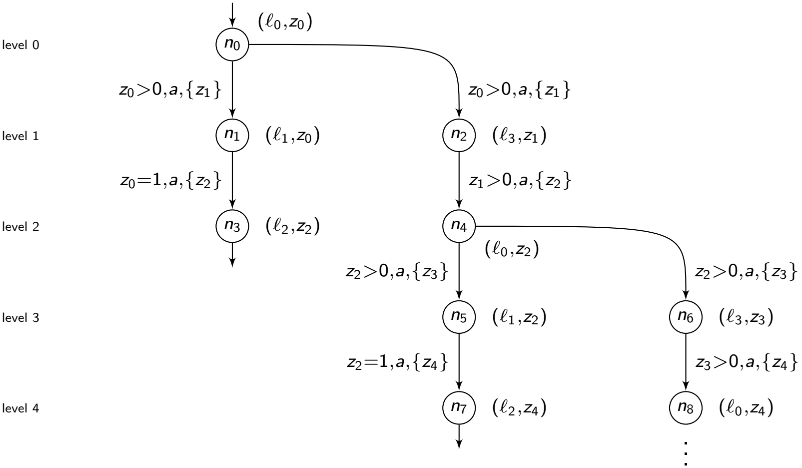


Unfolding

- $\mathcal{A}$ unfolded into a tree $\mathcal{A}^\infty$ with a fresh clock at each step.
- clocks of $\mathcal{A}$ are mapped to their reference in the new set of clocks.



Unfolding


level 0
level 1
level 2
level 3
level 4
(ℓ_0, z_0)
n_0
$z_0 > 0, a, \{z_1\}$
$z_0 > 0, a, \{z_1\}$
n_1
(ℓ_1, z_0)
n_2
(ℓ_3, z_1)
$z_0 = 1, a, \{z_2\}$
$z_1 > 0, a, \{z_2\}$
n_3
(ℓ_2, z_2)
n_4
(ℓ_0, z_2)
$z_2 > 0, a, \{z_3\}$
$z_2 > 0, a, \{z_3\}$
n_5
(ℓ_1, z_2)
n_6
(ℓ_3, z_3)
$z_2 = 1, a, \{z_4\}$
$z_3 > 0, a, \{z_4\}$
n_7
(ℓ_2, z_4)
n_8
(ℓ_0, z_4)

Properties of the unfolding

Input-determinacy property:
for every timed word w, there is a unique valuation v_w s.t. every initial run on w ends in some (n, v_w) with $level(n) = |w|$.

Lemma

$\mathcal{A}$ and $\mathcal{A}^\infty$ are strongly timed bisimilar; in particular $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^\infty)$.

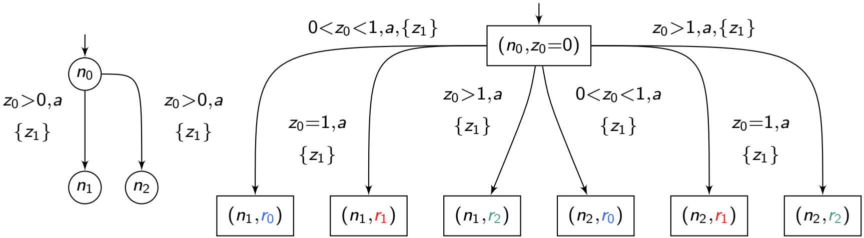
Drawbacks:

- $\mathcal{A}^\infty$ has infinitely many locations.
- $\mathcal{A}^\infty$ has infinitely many clocks.

Outline

Region equivalence

Region construction on $\mathcal{A}^\infty$: at level i regions over $\{z_0, \cdots, z_i\}$.



where $r_0{=}0{=}z_1{<}z_0{<}1$, $r_1{=}0{=}z_1{<}z_0{=}1$ and $r_2{=}0{=}z_1{<}1{<}z_0$

Lemma

$\mathcal{A}^\infty$ and $R(\mathcal{A}^\infty)$ are strongly timed bisimilar; thus $\mathcal{L}(\mathcal{A}) = \mathcal{L}(R(\mathcal{A}^\infty))$.

Outline

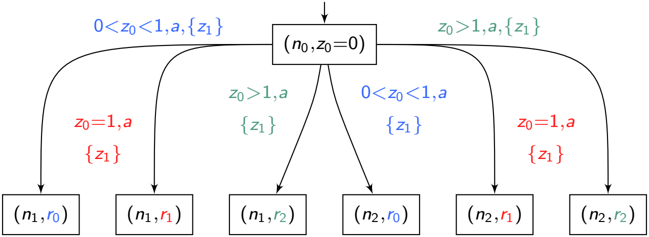
1 Timed automata

2 A determinization procedure
- Unfolding into an infinite tree
- Region equivalence
- Symbolic determinization
- Clock reduction
- Location reduction

3 Summary

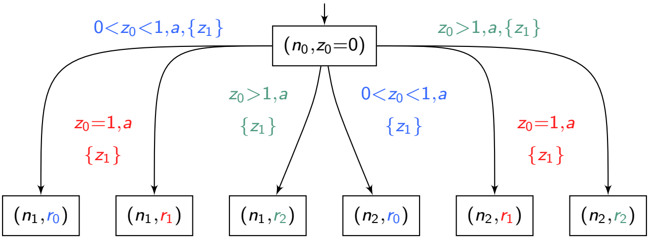
Symbolic determinization

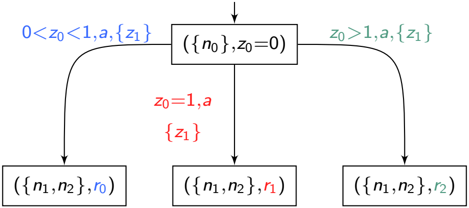
Determinization at level i on the alphabet $\mathrm{Reg}_i \times \Sigma \times Z$.



Symbolic determinization

Determinization at level i on the alphabet $\mathrm{Reg}_i \times \Sigma \times Z$.





Properties of the symbolic determinization

The symbolic determinization corresponds to determinization of the timed system.

SymbDet($\mathcal{A}$) is deterministic!

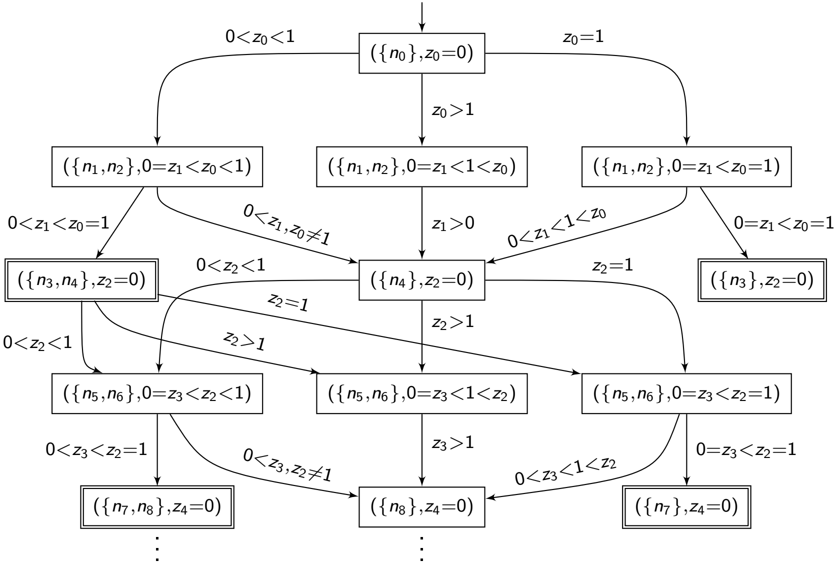
Lemma

$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{SymbDet}(R(\mathcal{A}^\infty)))$.

Drawbacks:

- SymbDet($R(\mathcal{A}^\infty)$) has infinitely many locations.
- SymbDet($R(\mathcal{A}^\infty)$) has infinitely many clocks.

Symbolic determinization on the example



Outline

1. Timed automata
2. A determinization procedure
   - Unfolding into an infinite tree
   - Region equivalence
   - Symbolic determinization
   - Clock reduction
   - Location reduction
3. Summary

Clock reduction

Active clocks: given a node of SymbDet($R(\mathcal{A})$), its active clocks is the set of clocks appearing in the region of the node.

Clock boundedness

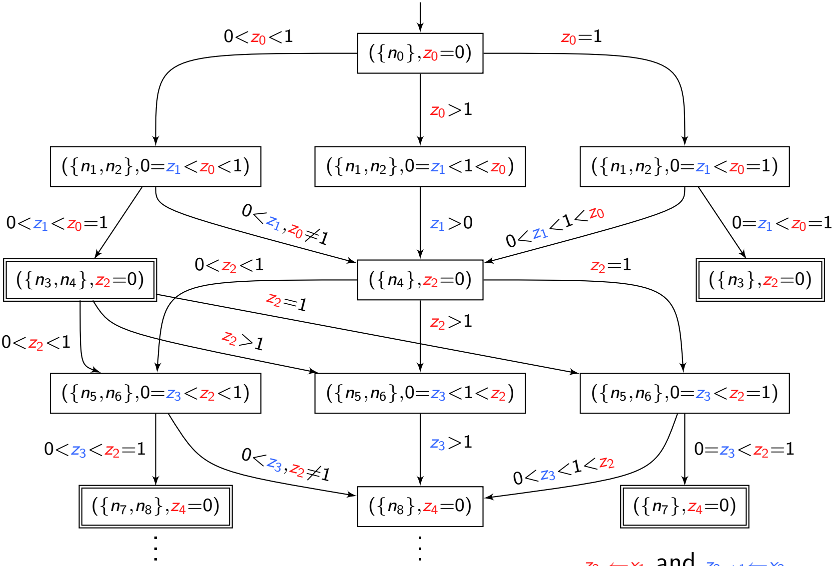
SymbDet($R(\mathcal{A}^\infty)$) is γ-clock bounded if in every node the number of active clocks is bounded by γ.

Under the clock-boundedness assumption: $\Gamma_\gamma(\text{SymbDet}(R(\mathcal{A}^\infty)))$ = reduction of SymbDet($R(\mathcal{A}^\infty)$) to set of clocks $\{x_1, \cdots, x_\gamma\}$.

Lemma

$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\Gamma_\gamma(\text{SymbDet}(R(\mathcal{A}^\infty))))$

Clock reduction on the example



$z_{2n} \longleftarrow x_1$ and $z_{2n+1} \longleftarrow x_2$

Outline

1. Timed automata
2. A determinization procedure
   - Unfolding into an infinite tree
   - Region equivalence
   - Symbolic determinization
   - Clock reduction
   - Location reduction
3. Summary

Location reduction

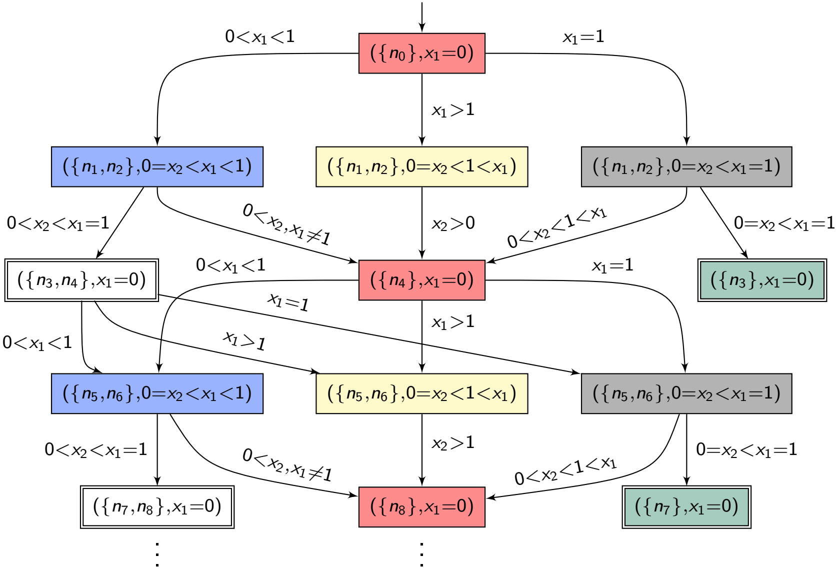
Property of $\Gamma_\gamma(\mathsf{SymbDet}(R(\mathcal{A}^\infty)))$:
Nodes sharing the same label (= set of locations + region + assignment of the clocks) are isomorphic.

$\mathcal{B}_{\mathcal{A},\gamma}$: $\Gamma_\gamma(\mathsf{SymbDet}(R(\mathcal{A}^\infty)))$ after merging isomorphic nodes.

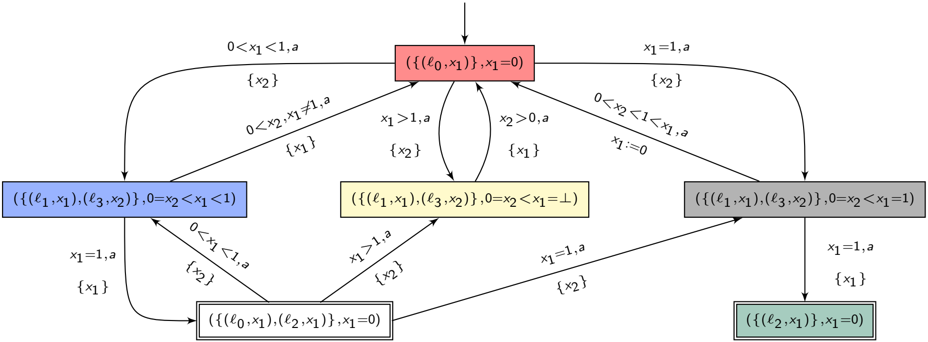
Theorem

$\mathcal{B}_{\mathcal{A},\gamma}$ is a deterministic timed automaton such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B}_{\mathcal{A},\gamma})$.

Back to the example



A deterministic version of the example



Outline

1. Timed automata
2. A determinization procedure
   - Unfolding into an infinite tree
   - Region equivalence
   - Symbolic determinization
   - Clock reduction
   - Location reduction
3. Summary

Recap of the procedure

1. Unfolding into a timed tree with infinitely many clocks and nodes
2. Region construction on the timed tree
   (still infinitely many clocks and nodes)
3. Symbolic determinization of the region tree
   (corresponding to a determinization of the timed system)
4. Reduction of the number of clocks
   (under the γ-clock bounded hypothesis)
5. Reduction of the number of locations

Determinizable classes

p-assumption

Let $p \in \mathbb{N}$. $\mathcal{A}$ satisfies the p-assumption if for every $n \geq p$, for every run

$$\rho = (\ell_0, v_0) \xrightarrow{\tau_1, a_1} (\ell_1, v_1) \ldots \xrightarrow{\tau_n, a_n} (\ell_n, v_n)$$

for every clock x, either x is reset along ρ, of $v_n(x) > M$.

If $\mathcal{A}$ satisfies the p-assumption, then SymbDet($R(\mathcal{A}^\infty)$) is p-clock bounded.

Classes of determinizable TA

- Event-clock TA ($|\Sigma|$-clock bounded)
- Strongly non-Zeno TA (satisfy the p-assumption)
- TA with integer resets

Complexity issues

Upper bound

Universality for timed automata can be checked in nondeterministic space logarithmic in the size of the deterministic timed automaton.

Given $\mathcal{A} = (L, \ell_0, L_{acc}, X, M, E)$ such that SymbDet($R(\mathcal{A}^\infty)$) is γ-clock bounded, $\mathcal{B}_{\mathcal{A},\gamma}$ has $2^{|L|} \cdot \gamma^{|X|} \cdot \left((2M+2)^{(\gamma+1)^2} \cdot \gamma!\right)$ locations.

Universality can be decided in EXPSPACE for timed automata satisfying the p-assumption, and for integer resets timed automata.

Lower bound

Checking universality in timed automata either satisfying the p-assumption or with integer resets is EXPSPACE-hard.

Summary complexity

	size of the det. TA	universality problem	inclusion problem
TA_p	*doubly exp.*	*EXPSPACE-compl.*	*EXPSPACE-compl.*
SnZTA	*doubly exp.*	trivial	*EXPSPACE-compl.*
ECTA	exp.	PSPACE-compl.	PSPACE-compl.
IRTA	doubly exp.	EXPSPACE-*compl.*	EXPSPACE-*compl.*