Zone based verification of timed automata revisited

B. Srivathsan

Joint work with F. Herbreteau and I. Walukiewicz

LaBRI, Université de Bordeaux 1

Groupe de Travail Modélisation et Vérification

LIF, Marseille - November 2011

Outline

The reachability problem

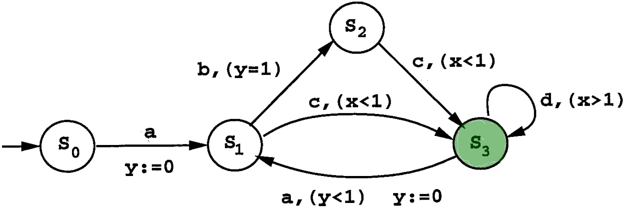
The liveness problem

Part 1: The reachability problem

Includes work done with

D. Kini
Indian Institute of Technology, Bombay

Timed Automata [AD94]



Run: finite **sequence** of transitions,

$$(s_0, \overbrace{0}^{x}, \overbrace{0}^{y}) \xrightarrow{0.4,a} (s_1, 0.4, 0) \xrightarrow{0.5,c} (s_3, 0.9, 0.5)$$

- A run is **accepting** if it ends in a green state.

The problem we are interested in ...

Given a TA, does there **exist** an **accepting run**?

The problem we are interested in ...

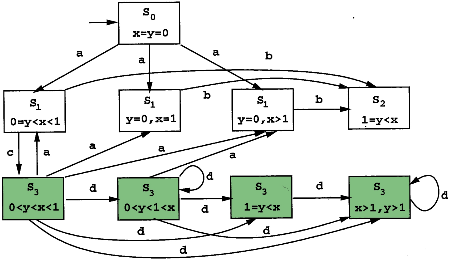
Given a TA, does there **exist** an **accepting run**?

Theorem [AD94, CY92]

This problem is **PSPACE-complete**

First solution to this problem

Key idea: Partition the space of valuations into a **finite** number of **regions**



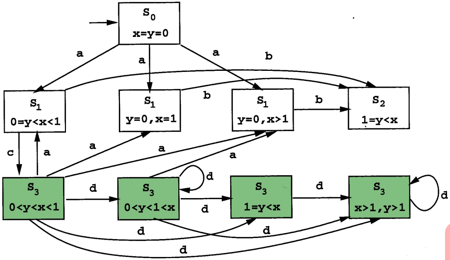
- Region: set of valuations satisfying the **same guards** w.r.t. time
- Finiteness: Parametrized by **maximal constant**

Sound and complete [AD94]

Region graph preserves state **reachability**

First solution to this problem

Key idea: Partition the space of valuations into a **finite** number of **regions**



- Region: set of valuations satisfying the **same guards** w.r.t. time
- Finiteness: Parametrized by **maximal constant**

$\mathcal{O}(|X|!.M^{|X|})$ many regions!

Sound and complete [AD94]

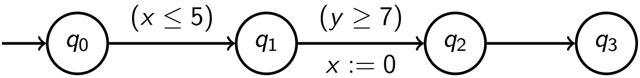
Region graph preserves state **reachability**

A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path

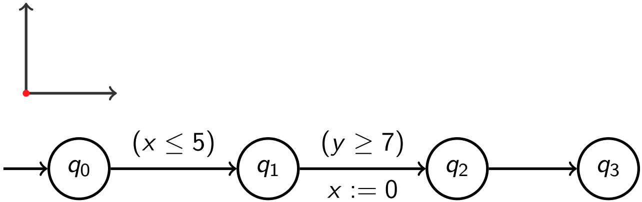
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



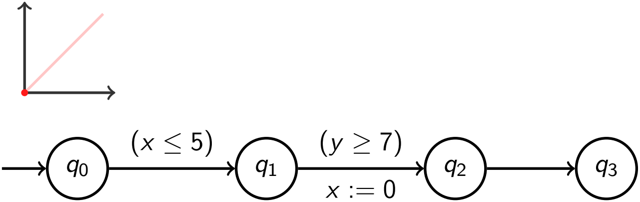
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



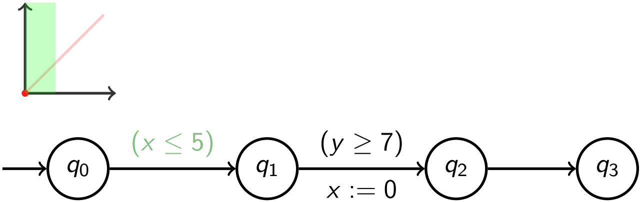
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



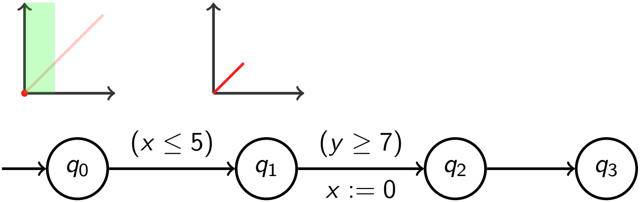
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



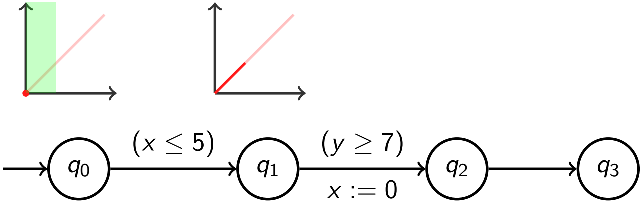
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



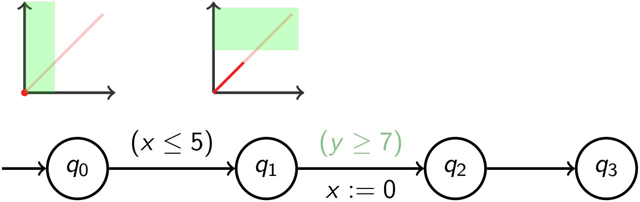
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



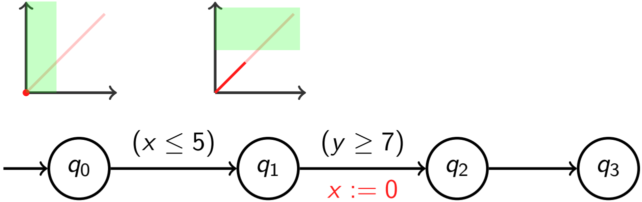
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



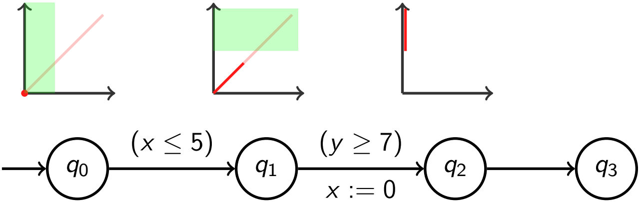
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



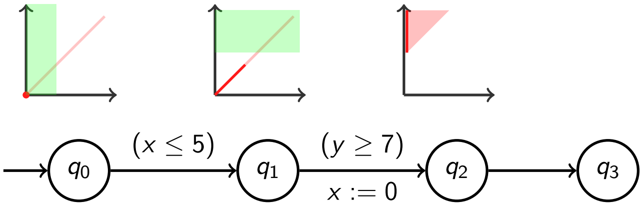
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



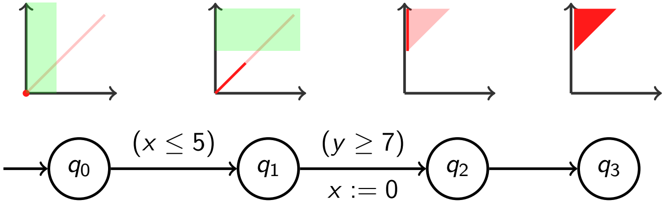
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



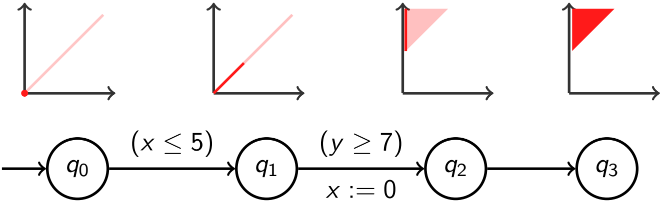
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



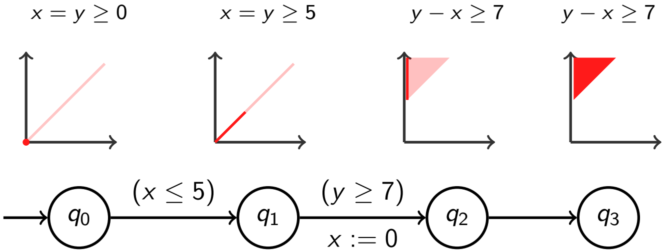
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path



A more efficient solution...

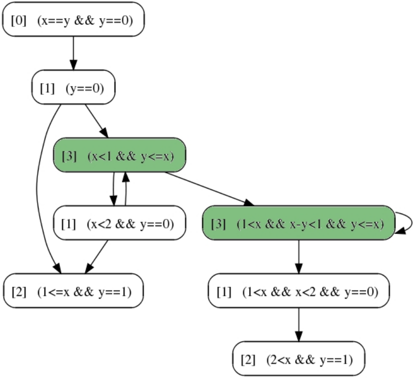
Key idea: Maintain **all valuations** reachable along a path



Zones and zone graph

- Zone: set of valuations defined by conjunctions of constraints:
  - $x \sim c$
  - $x - y \sim c$
  - e.g. $(x - y \geq 1) \wedge (y < 2)$
- Representation: by DBM

Zones and zone graph

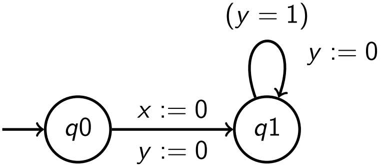


- Zone: set of valuations defined by conjunctions of constraints:
  - $x \sim c$
  - $x - y \sim c$
  - e.g. $(x - y \geq 1) \wedge (y < 2)$
- Representation: by DBM

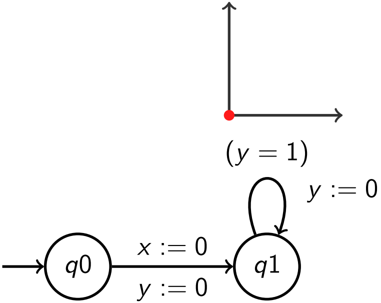
Sound and complete [DT98]

Zone graph preserves state **reachability**

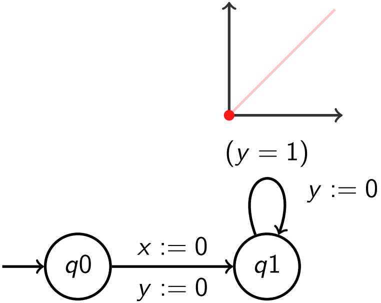
But the zone graph could be infinite ...



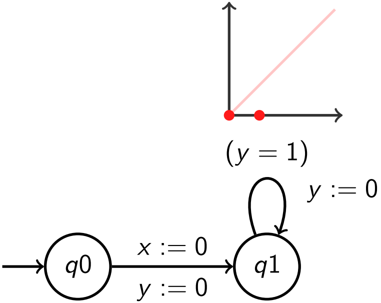
But the zone graph could be infinite ...



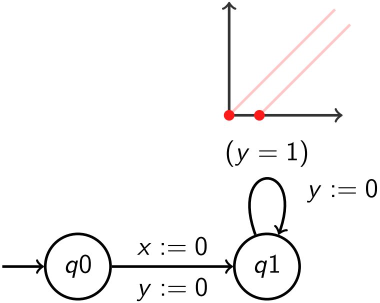
But the zone graph could be infinite ...



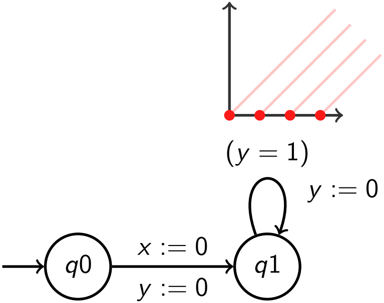
But the zone graph could be infinite ...



But the zone graph could be infinite ...

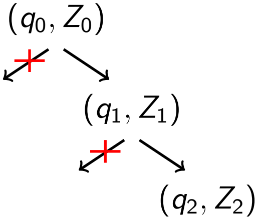


But the zone graph could be infinite ...



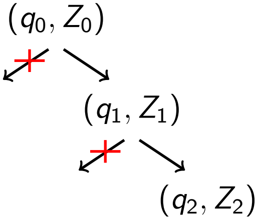
Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



(q_0, Z_0)

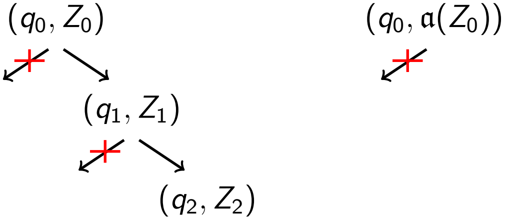
(q_1, Z_1)

(q_2, Z_2)

$(q_0, \mathfrak{a}(Z_0))$

Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



(q_0, Z_0)

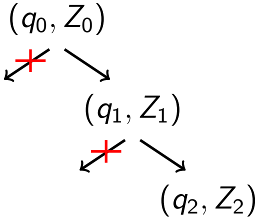
(q_1, Z_1)

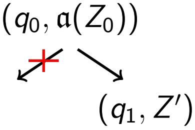
(q_2, Z_2)

$(q_0, \mathfrak{a}(Z_0))$

Use finite abstractions

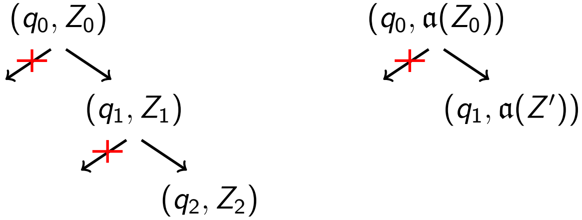
Key idea: **Abstract** each zone in a **sound** manner





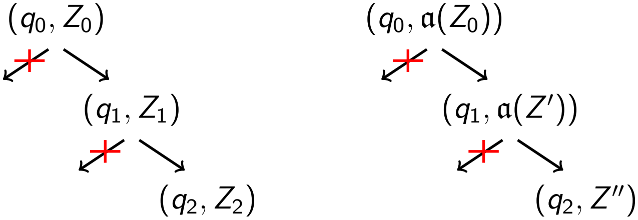
Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



(q_0, Z_0)

(q_1, Z_1)

(q_2, Z_2)

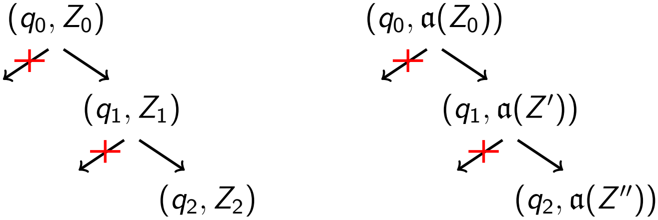
$(q_0, \mathfrak{a}(Z_0))$

$(q_1, \mathfrak{a}(Z'))$

(q_2, Z'')

Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



(q_0, Z_0)

(q_1, Z_1)

(q_2, Z_2)

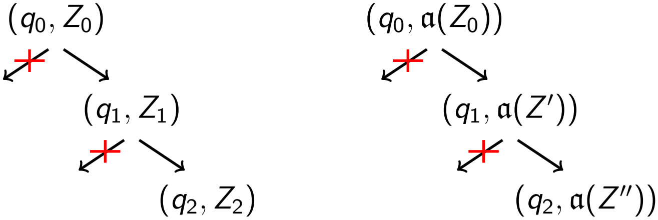
$(q_0, \mathfrak{a}(Z_0))$

$(q_1, \mathfrak{a}(Z'))$

$(q_2, \mathfrak{a}(Z''))$

Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner



(q_0, Z_0)

(q_1, Z_1)

(q_2, Z_2)

$(q_0, \mathfrak{a}(Z_0))$

$(q_1, \mathfrak{a}(Z'))$

$(q_2, \mathfrak{a}(Z''))$

- Number of **abstracted zones** is **finite**
- **Coarser** abstraction → **fewer** abstracted zones

Abstractions in literature [Bou04, BBLP06]

$$\begin{tikzcd}
 & \mathfrak{a}_{\preccurlyeq LU} & \\
\mathrm{Closure}_{\alpha} \arrow[ur] & & \mathrm{Extra}^{+}_{LU} \arrow[ul] \\
 & \mathrm{Extra}^{+}_{\alpha} \arrow[ul] \arrow[ur] & \\
 & \mathrm{Extra}_{\alpha} \arrow[u] &
\end{tikzcd}$$

Abstractions in literature [Bou04, BBLP06]

$$\begin{tikzcd}
 & \mathfrak{a}_{\preccurlyeq LU} & \\
\mathrm{Closure}_{\alpha} \arrow[ur] & & \mathrm{Extra}^{+}_{LU} \arrow[ul] \\
 & \mathrm{Extra}^{+}_{\alpha} \arrow[ul] \arrow[ur] & \\
 & \mathrm{Extra}_{\alpha} \arrow[u] &
\end{tikzcd}$$

Sound and complete

All the above abstractions preserve state **reachability**

Abstractions in literature [Bou04, BBLP06]

$$\begin{tikzcd}
 & \mathfrak{a}_{\preccurlyeq LU} & \\
\mathsf{Closure}_\alpha \arrow[ur] & & \mathsf{Extra}^+_{LU} \arrow[ul] \\
 & \mathsf{Extra}^+_\alpha \arrow[ul] \arrow[ur] & \\
 & \mathsf{Extra}_\alpha \arrow[u] &
\end{tikzcd}$$

Sound and complete

All the above abstractions preserve state **reachability**

But for **implementation** abstracted zone should be a zone

Abstractions in literature [Bou04, BBLP06]

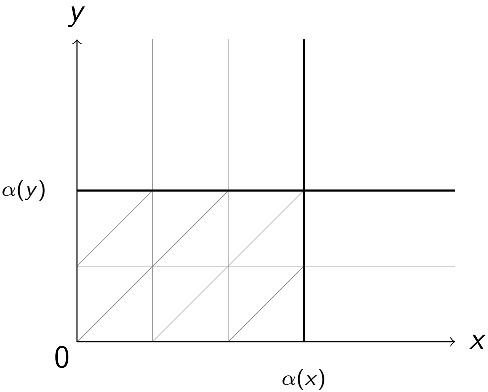
$$\begin{tikzcd}
 & \mathfrak{a}_{\preccurlyeq LU} & \\
\text{Closure}_{\alpha} \arrow[ur] & & \text{Extra}^{+}_{LU} \checkmark \arrow[ul] \\
 & \text{Extra}^{+}_{\alpha} \checkmark \arrow[ul] \arrow[ur] & \\
 & \text{Extra}_{\alpha} \checkmark \arrow[u] &
\end{tikzcd}$$

Only convex abstractions in **implementations**!

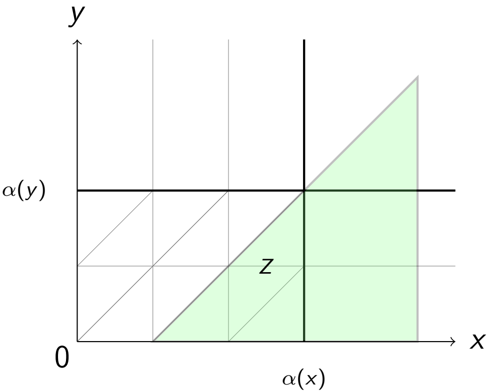
Here...

Efficient use of the **non-convex** Closure abstraction!

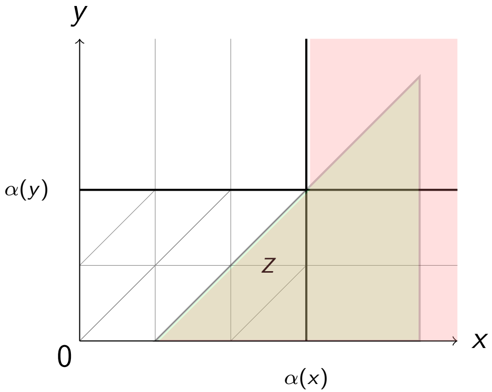
What is Closure$_\alpha$?



What is Closure$_\alpha$?

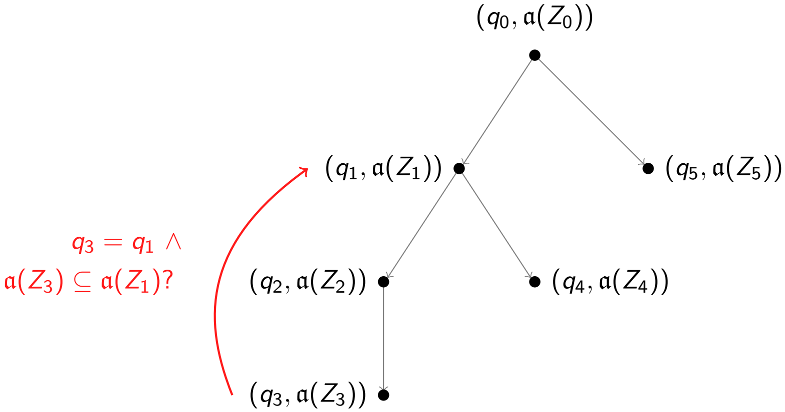


What is Closure$_\alpha$?



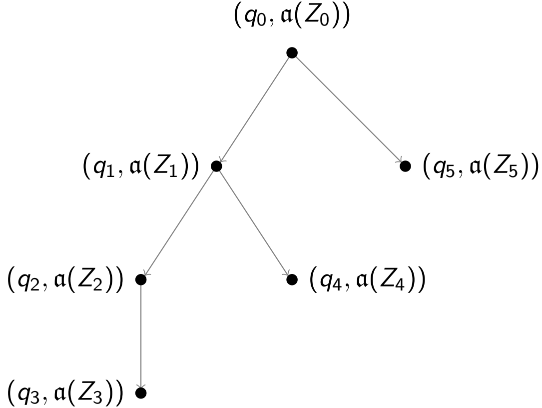
Closure$_\alpha(Z)$: set of regions that Z intersects

Using Closure$_\alpha$ for reachability



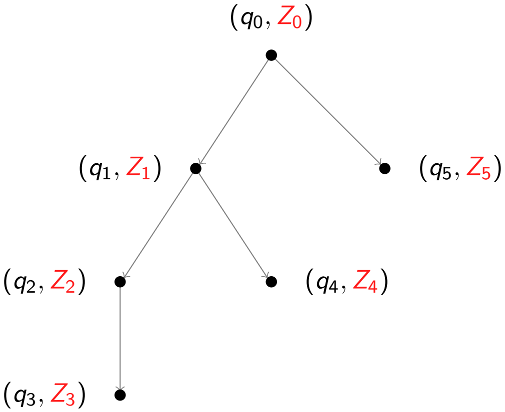
Standard algorithm: **covering tree**

Using Closure$_\alpha$ for reachability



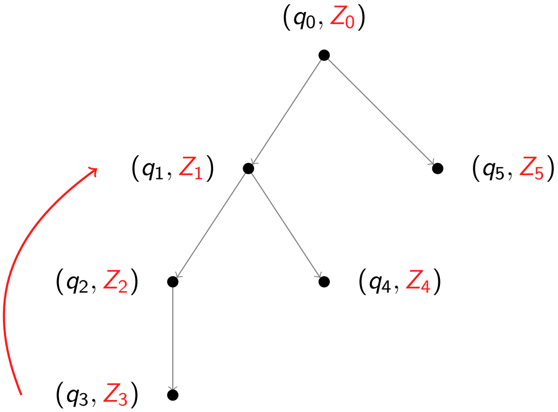
Closure$_\alpha(Z)$ **cannot be** efficiently **stored**

Using Closure$_\alpha$ for reachability



Do **not store** abstracted zones!

Using Closure_α for reachability

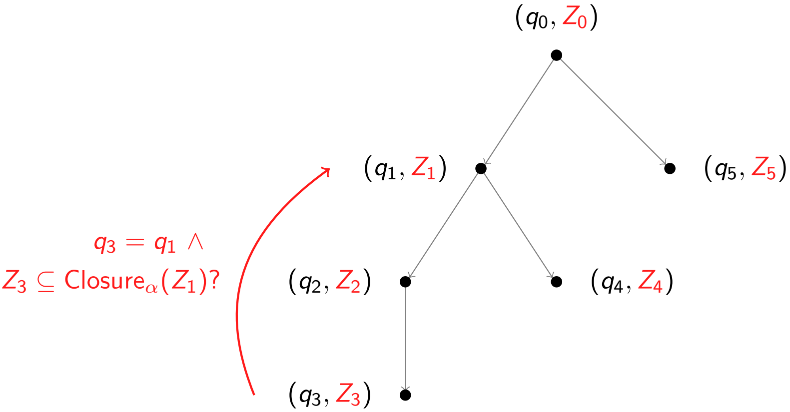


$q_3 = q_1 \ \wedge$

$Z_3 \subseteq \text{Closure}_\alpha(Z_1)$?

Use Closure for termination!

Using Closure$_\alpha$ for reachability



(q_0, Z_0)

(q_1, Z_1)

(q_5, Z_5)

(q_2, Z_2)

(q_4, Z_4)

(q_3, Z_3)

$q_3 = q_1 \wedge$
$Z_3 \subseteq \text{Closure}_\alpha(Z_1)$?

Need an **efficient** algorithm for $Z \subseteq \text{Closure}_\alpha(Z')$

Reduction to two clocks

Inspired by a crucial observation made in [Bou04]

Theorem

$Z \not\subseteq \text{Closure}_\alpha(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\mathbf{Proj}_{xy}(Z) \not\subseteq \text{Closure}_\alpha(\mathbf{Proj}_{xy}(Z'))$$

Reduction to two clocks

Inspired by a crucial observation made in [Bou04]

Theorem

$Z \not\subseteq \text{Closure}_\alpha(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\mathbf{Proj}_{xy}(Z) \not\subseteq \text{Closure}_\alpha(\mathbf{Proj}_{xy}(Z'))$$

Complexity: $\mathcal{O}(|X|^2)$, where X is the set of clocks

Reduction to two clocks

Inspired by a crucial observation made in [Bou04]

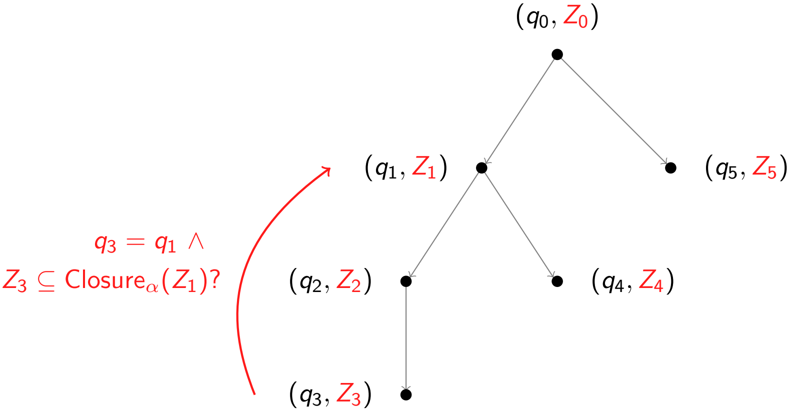
Theorem

$Z \not\subseteq \text{Closure}_\alpha(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\textbf{Proj}_{xy}(Z) \not\subseteq \text{Closure}_\alpha(\textbf{Proj}_{xy}(Z'))$$

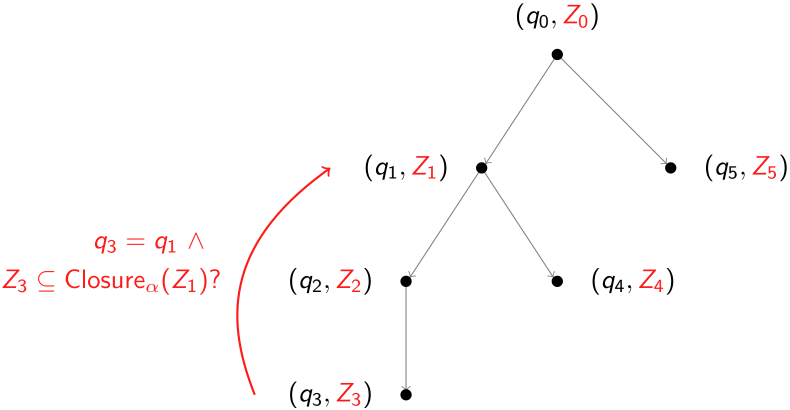
Same complexity as $Z \subseteq Z'$!

So what do we have now...



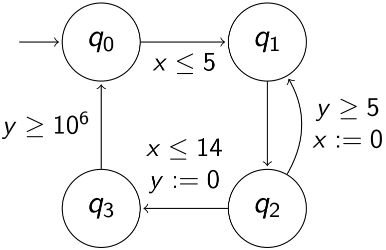
Efficient algorithm for $Z \subseteq \text{Closure}_\alpha(Z')$

So what do we have now...



Coming next: **prune** the **bound function** α!

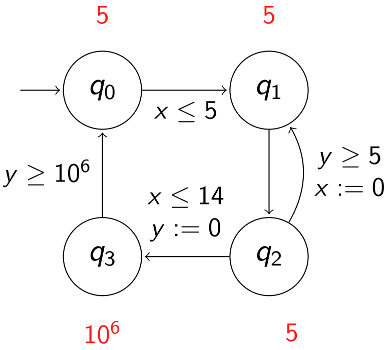
Bound function α



Naive: $\alpha(x) = 14$, $\alpha(y) = 10^6$

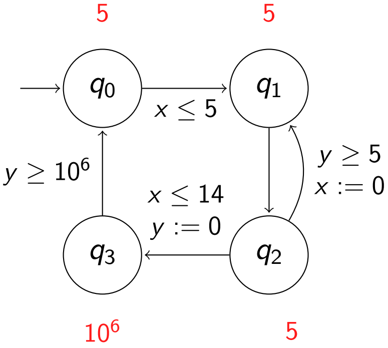
Size of graph $\sim 10^5$

Static analysis: bound function for every q [BBFL03]



Naive: $\alpha(x) = 14$, $\alpha(y) = 10^6$

Static analysis: bound function for every q [BBFL03]

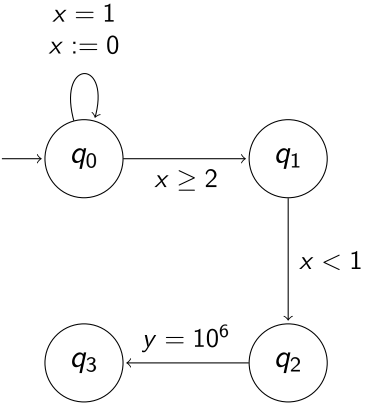


Naive: $\alpha(x) = 14$, $\alpha(y) = 10^6$

But this is not enough!

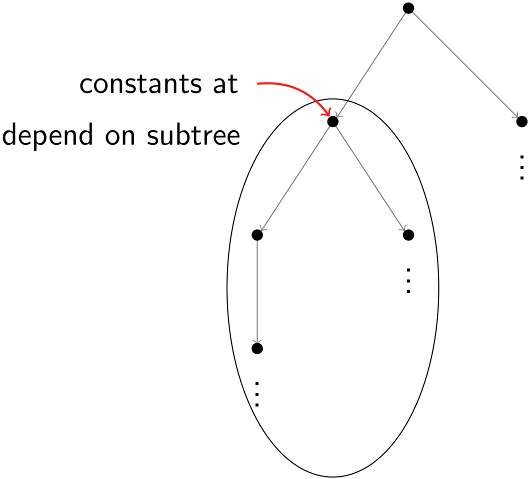
Need to look at semantics...

Static analysis: $\alpha(y) = 10^6$



More than $\mathbf{10^6}$ zones at q_0 **not necessary**!

Bound function for every (q, Z) in $\mathrm{ZG}(\mathcal{A})$



constants at

depend on subtree

Constant propagation

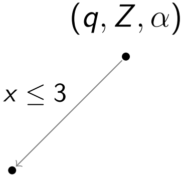
$\alpha(x) = -\infty$

(q, Z, α)

•

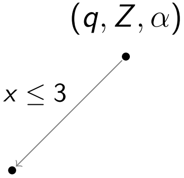
Constant propagation

$\alpha(x) = -\infty$



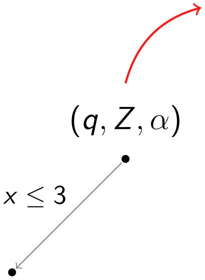
Constant propagation

$\alpha(x) = 3$



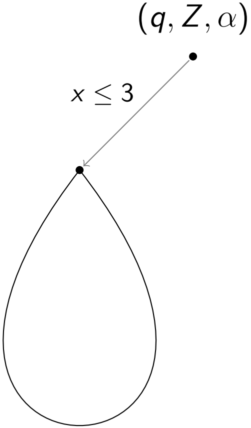
Constant propagation

$\alpha(x) = \ 3$



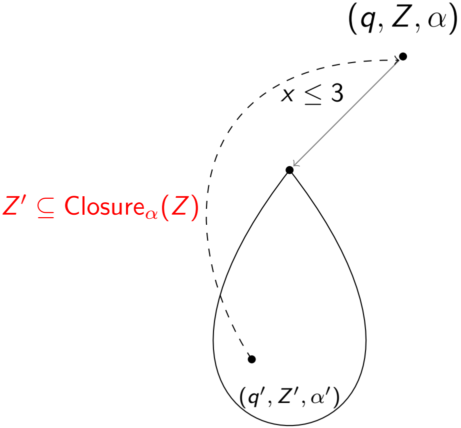
Constant propagation

$\alpha(x) = \quad 5$



Constant propagation

$\alpha(x) = \quad 5$



(q, Z, α)

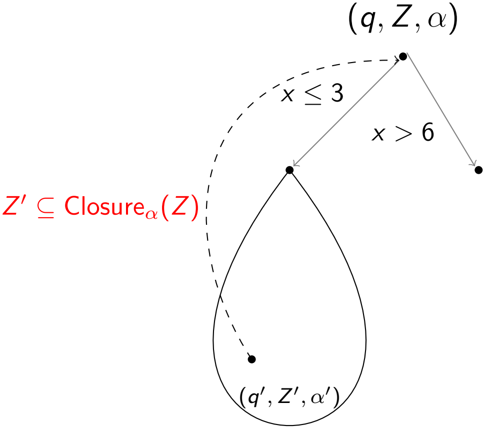
$x \leq 3$

$Z' \subseteq \mathsf{Closure}_{\alpha}(Z)$

(q', Z', α')

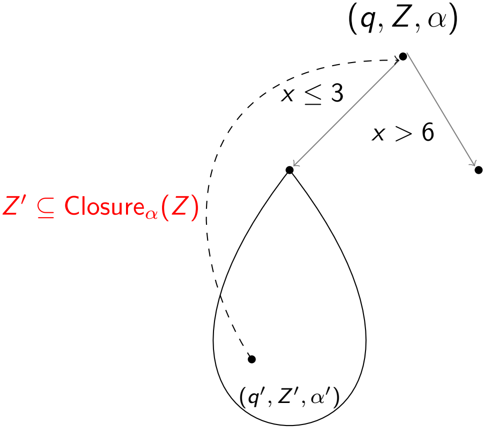
Constant propagation

$\alpha(x) = \quad 5$



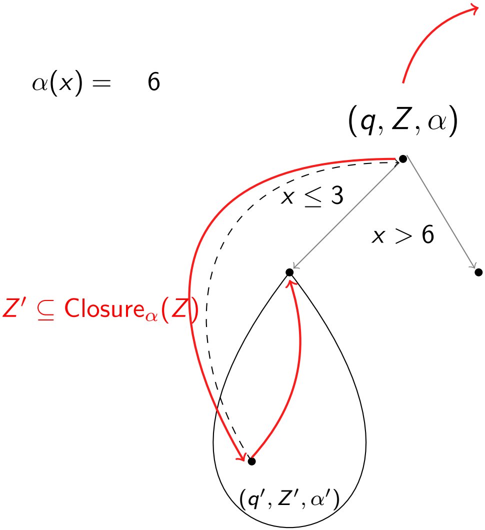
Constant propagation

$\alpha(x) = \quad 6$



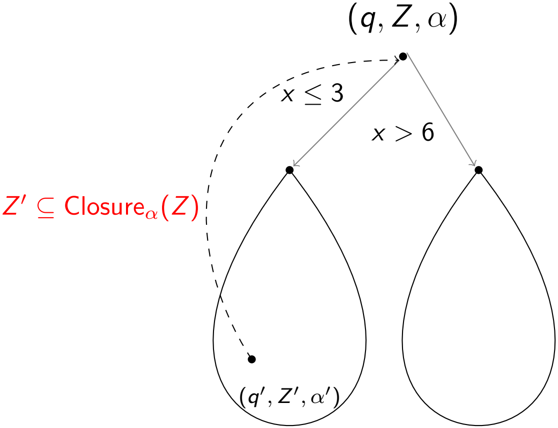
Constant propagation

$\alpha(x) = \quad 6$



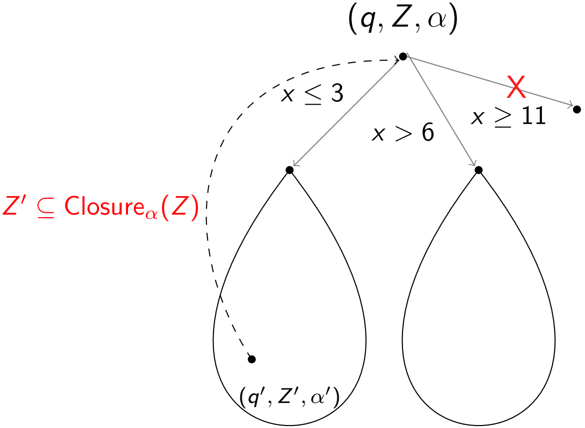
Constant propagation

$\alpha(x) = \quad 6$



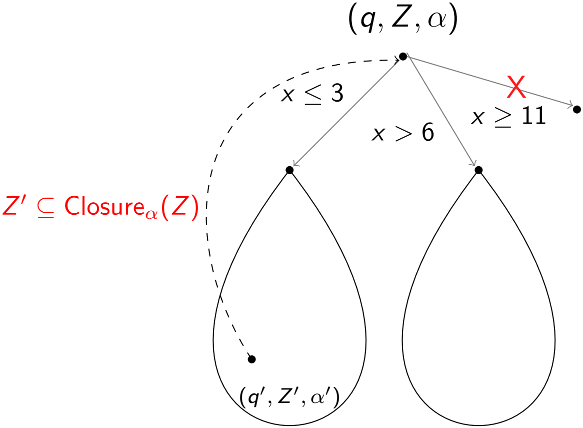
Constant propagation

$\alpha(x) = \quad 6$



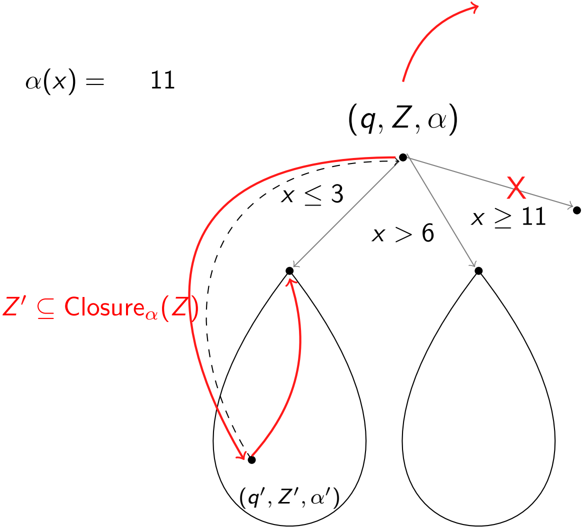
Constant propagation

$\alpha(x) = \quad 11$



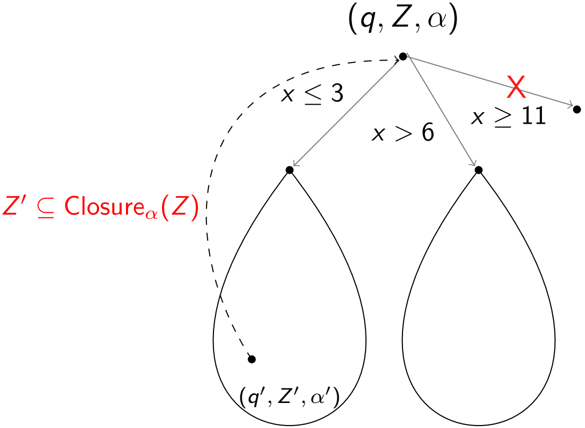
Constant propagation

$\alpha(x) = \quad 11$



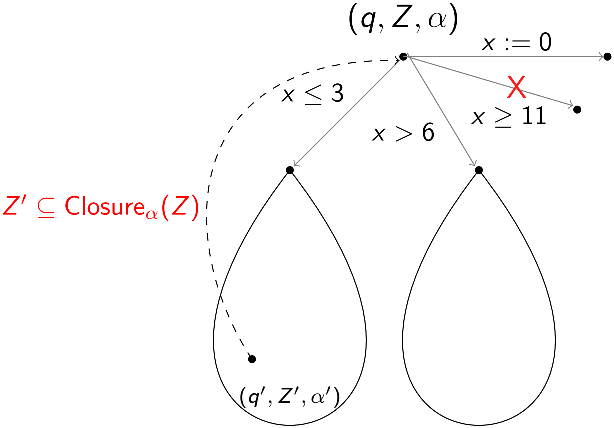
Constant propagation

$\alpha(x) = \quad 11$



Constant propagation

$\alpha(x) = \quad 11$



$Z' \subseteq \text{Closure}_\alpha(Z)$

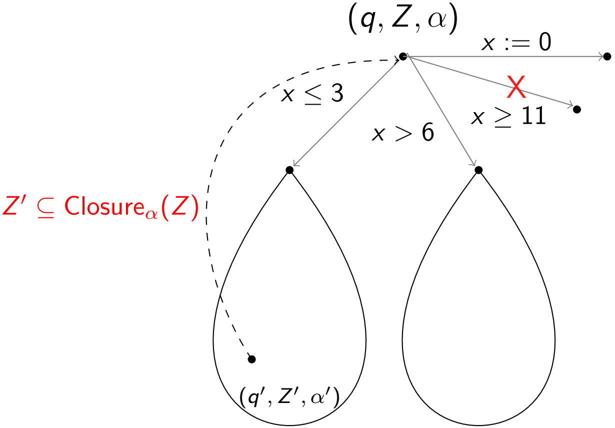
Constant propagation

All **tentative nodes** consistent

+ No more **exploration**

→ Terminate!

$\alpha(x) =$ 11



Invariants on the bounds

- Non tentative nodes: $\alpha = max\{\alpha_{succ}\}$ (modulo resets)
- Tentative nodes: $\alpha = \alpha_{covering}$

Theorem (Correctness)

An accepting state is reachable in ZG($\mathcal{A}$) iff the algorithm reaches a node with an accepting state and a non-empty zone.

Overall algorithm

- Compute $ZG(\mathcal{A})$: $Z \subseteq \text{Closure}_{\alpha'}(Z')$ for **termination**
- **Bounds** α calculated **on-the-fly**
- Abstraction Extra^+_{LU} can **also** be **handled**:

$$\begin{tikzcd}
& \mathfrak{a}_{\preccurlyeq LU} & \\
& \text{Closure}_\alpha \circ \text{Extra}^+_{LU} \arrow[u] & \\
\text{Closure}_\alpha \arrow[ur] & & \text{Extra}^+_{LU} \arrow[ul] \\
& \text{Extra}^+_\alpha \arrow[ul] \arrow[ur] & \\
& \text{Extra}_\alpha \arrow[u] &
\end{tikzcd}$$

An **efficient** $\mathcal{O}(|X|^2)$ procedure for $Z \subseteq \text{Closure}_\alpha(Extra^+_{LU}(Z'))$!

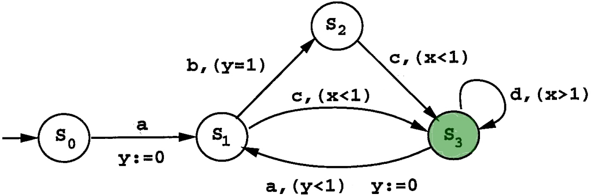
Benchmarks

Model	Our algorithm		UPPAAL's algorithm		UPPAAL 4.1.3 (-n4 -C -o1)	
	nodes	s.	nodes	s.	nodes	s.
CSMA/CD7	5031	0.32	5923	0.27	–	T.O.
CSMA/CD8	16588	1.36	19017	1.08	–	T.O.
CSMA/CD9	54439	6.01	60783	4.19	–	T.O.
FDDI10	459	0.02	525	0.06	12049	2.43
FDDI20	1719	0.29	2045	0.78	–	T.O.
FDDI30	3779	1.29	4565	4.50	–	T.O.
Fischer7	7737	0.42	20021	0.53	18374	0.35
Fischer8	25080	1.55	91506	2.48	85438	1.53
Fischer9	81035	5.90	420627	12.54	398685	8.95
Fischer10	–	T.O.	–	T.O.	1827009	53.44

- $\mathbf{Extra}^{+}_{LU}$ and **static** analysis bounds in UPPAAL
- $\mathbf{Closure}_{\alpha}(\mathbf{Extra}^{+}_{LU})$ and **otf** bounds in our algorithm

Part 2: The liveness problem

Timed Büchi Automata [AD94]



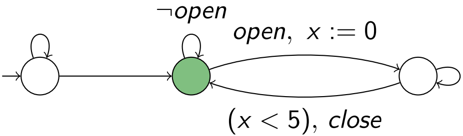
Run: infinite sequence of transitions

$$(s_0, \overbrace{0}^{x}, \overbrace{0}^{y}) \xrightarrow{0.4,a} (s_1, 0.4, 0) \xrightarrow{0.5,c} (s_3, 0.9, 0.5) \xrightarrow{0.3,d} (s_3, 1.2, 0.8) \xrightarrow{15,d} \cdots$$

- **accepting** if infinitely often green
- **non-Zeno** if time diverges ($\sum_{i\geq 0} \delta_i \to \infty$)

Model-Checking Real-Time Systems

Correctness: Safety + **Liveness** + **Fairness**



"Infinitely often, the gate is open for at least 5 s."

Realistic counter-examples: infinite **non-Zeno** runs

The problem that we consider

Given a TBA A, does it **have** a **non-Zeno** accepting run

Theorem [AD94]

Deciding if a TBA has a non-Zeno accepting run is **PSPACE-complete**

Once again abstract zone graph $\mathrm{ZG}^{\mathfrak{a}}(\mathcal{A})$

$$\begin{tikzcd}
 & \mathfrak{a}_{\preccurlyeq LU} & \\
\mathrm{Closure}_{\alpha} \arrow[ur] & & \mathrm{Extra}^{+}_{LU}\ \checkmark \arrow[ul] \\
 & \mathrm{Extra}^{+}_{\alpha}\ \checkmark \arrow[ul] \arrow[ur] & \\
 & \mathrm{Extra}_{\alpha}\ \checkmark \arrow[u] &
\end{tikzcd}$$

Sound and complete [Bou04, BBLP06, Tri09, Li09]

Extra_{α}, $\mathrm{Extra}^{+}_{\alpha}$, Extra^{+}_{LU} preserve **repeated state reachability**

Once again abstract zone graph $\mathrm{ZG}^{\mathfrak{a}}(\mathcal{A})$

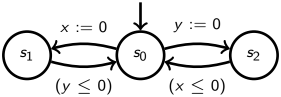
$$\begin{tikzcd}
 & \mathfrak{a}_{\preccurlyeq LU} & \\
\mathrm{Closure}_{\alpha} \arrow[ur] & & \mathrm{Extra}^{+}_{LU} \ \checkmark \arrow[ul] \\
 & \mathrm{Extra}^{+}_{\alpha} \ \checkmark \arrow[ul] \arrow[ur] & \\
 & \mathrm{Extra}_{\alpha} \ \checkmark \arrow[u] &
\end{tikzcd}$$

Sound and complete [Bou04, BBLP06, Tri09, Li09]

Extra_{α}, $\mathrm{Extra}^{+}_{\alpha}$, Extra^{+}_{LU} preserve **repeated state reachability**

What about **non-Zenoness**?

Finding non-Zeno Runs from Abstract Paths



Region graph:

$$\begin{tikzcd}
 & (s_1, 0 = x < y) & & (s_2, 0 = y < x) & \\
(s_0, 0 = x = y) \arrow[r] \arrow[ur] & (s_1, 0 = x = y) \arrow[r] & (s_0, 0 = x = y) \arrow[r] \arrow[ur] & (s_2, 0 = y = x) \arrow[r, dashed] & {}
\end{tikzcd}$$

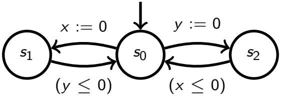
Zone graph with Extra$_\alpha$:

$$(s_0, 0 = x = y) \rightarrow (s_1, 0 = x \leq y) \rightarrow (s_0, 0 = x = y) \rightarrow (s_2, 0 = y \leq x) \dashrightarrow$$

Zone graph with Extra$^+_{LU}$:

$$(s_0, \top) \longrightarrow (s_1, \top) \longrightarrow (s_0, \top) \longrightarrow (s_2, \top) \dashrightarrow$$

Finding non-Zeno Runs from Abstract Paths



Region graph:

$$\begin{tikzcd}
& (s_1, 0 = x < y) & & (s_2, 0 = y < x) \\
(s_0, 0 = x = y) \arrow[r] \arrow[ur] & (s_1, 0 = x = y) \arrow[r] & (s_0, 0 = x = y) \arrow[r] \arrow[ur] & (s_2, 0 = y = x) \arrow[r, dashed] & {}
\end{tikzcd}$$

Zone graph with Extra_α:

$$(s_0, 0 = x = y) \rightarrow (s_1, 0 = x \leq y) \rightarrow (s_0, 0 = x = y) \rightarrow (s_2, 0 = y \leq x) \dashrightarrow$$

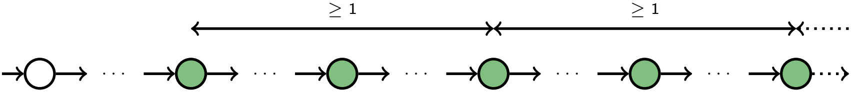
Zone graph with Extra^+_{LU}:

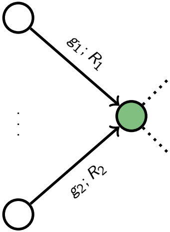
$$(s_0, \top) \longrightarrow (s_1, \top) \longrightarrow (s_0, \top) \longrightarrow (s_2, \top) \dashrightarrow$$

How to detect **non-Zeno runs** from **abstract zones**?

From TBA to Strongly non-Zeno TBA [TYB05]

Key Idea : reduce non-Zenoness to Büchi acceptation

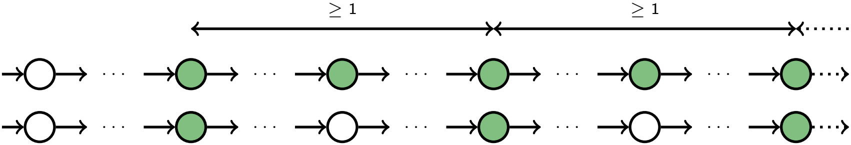


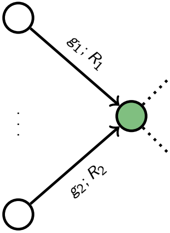


A

From TBA to Strongly non-Zeno TBA [TYB05]

Key Idea : reduce non-Zenoness to Büchi acceptation

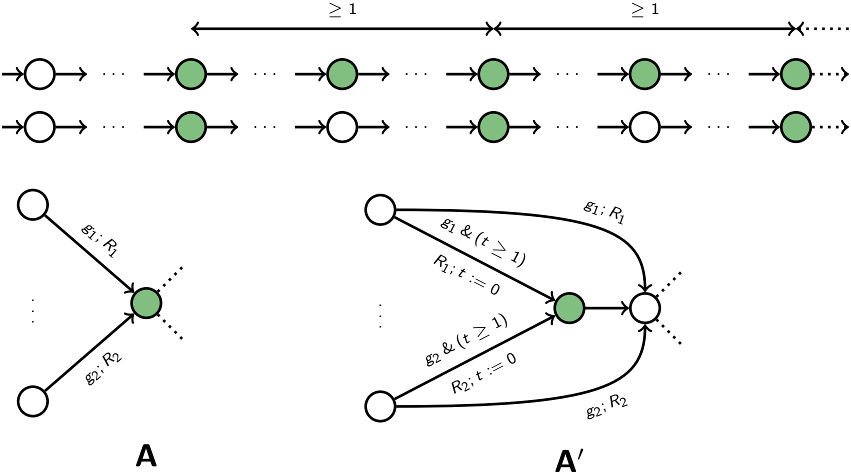




A

From TBA to Strongly non-Zeno TBA [TYB05]

Key Idea : reduce non-Zenoness to Büchi acceptation



Strongly non-Zeno TBA [Tri99, TYB05]

Definition

Strongly non-Zeno TBA: **all** accepting runs are **non-Zeno**

Theorem [TYB05]

For every TBA A, there exists a Strongly non-Zeno TBA A' that has an **accepting** run iff A has a **non-Zeno accepting** run

(size of A': $|X|+1$ clocks and at most $2|Q|$ states)

Theorem [Tri09]

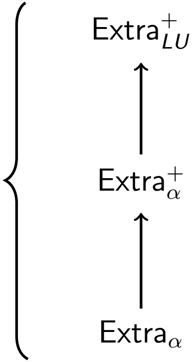
A has a non-Zeno accepting run iff $\mathrm{ZG}(A')$ has an accepting run

What we observe

Strongly non-Zeno Construction [TYB05]

$$\begin{tikzcd}
\text{Extra}^{+}_{LU} \\
\text{Extra}^{+}_{\alpha} \arrow[u] \\
\text{Extra}_{\alpha} \arrow[u]
\end{tikzcd}$$

What we observe



Strongly non-Zeno
Construction [TYB05]

Combinatorial
blowup

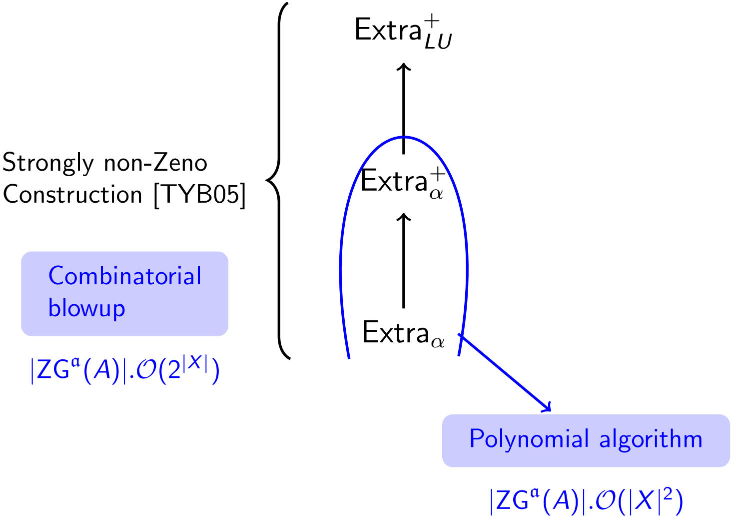
$|\mathrm{ZG}^{\mathfrak{a}}(A)|.\mathcal{O}(2^{|X|})$

Extra_{LU}^{+}

$\text{Extra}_{\alpha}^{+}$

Extra_{α}

and we propose...



Extra^+_{LU}

Extra^+_{α}

Extra_{α}

Strongly non-Zeno Construction [TYB05]

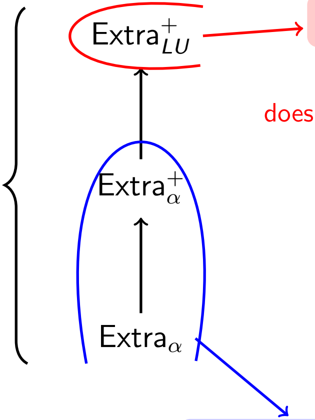
Combinatorial blowup

$|ZG^{\mathfrak{a}}(A)|.\mathcal{O}(2^{|X|})$

Polynomial algorithm

$|ZG^{\mathfrak{a}}(A)|.\mathcal{O}(|X|^2)$

and we propose...



Extra^+_{LU}

Extra^+_{α}

Extra_{α}

NP-complete

Given A, $\text{ZG}^{\text{Extra}^+_{LU}}$
does A have a non-Zeno run?

Strongly non-Zeno
Construction [TYB05]

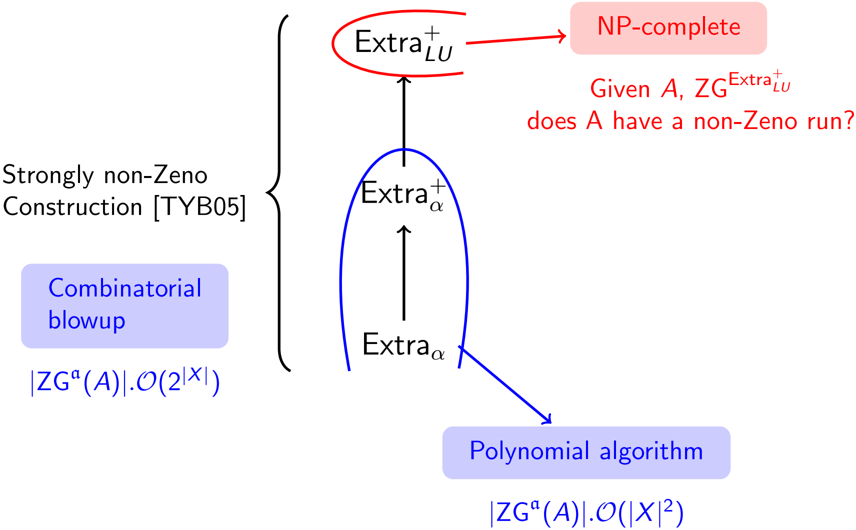
Combinatorial blowup

$|\text{ZG}^{\mathfrak{a}}(A)|.\mathcal{O}(2^{|X|})$

Polynomial algorithm

$|\text{ZG}^{\mathfrak{a}}(A)|.\mathcal{O}(|X|^2)$

and we propose...



Extra^+_{LU}

NP-complete

Given A, $ZG^{\text{Extra}^+_{LU}}$
does A have a non-Zeno run?

Extra^+_{α}

Extra_{α}

Strongly non-Zeno
Construction [TYB05]

Combinatorial blowup

$|ZG^{\mathfrak{a}}(A)|.\mathcal{O}(2^{|X|})$

Polynomial algorithm

$|ZG^{\mathfrak{a}}(A)|.\mathcal{O}(|X|^2)$

Coming next: the polynomial construction

Our approach to non-Zenoness

A path in $\mathrm{ZG}^{\mathfrak{a}}(A)$ **yields only Zeno runs** iff:

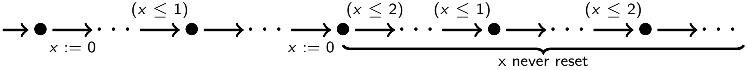
1. some clock x is **blocking**:

$$\to \bullet \xrightarrow[x := 0]{} \cdots \xrightarrow{(x \leq 1)} \bullet \longrightarrow \cdots \xrightarrow[x := 0]{} \underbrace{\bullet \xrightarrow{(x \leq 2)} \cdots \xrightarrow{(x \leq 1)} \bullet \longrightarrow \cdots \xrightarrow{(x \leq 2)} \bullet \longrightarrow \cdots}_{\text{x never reset}}$$

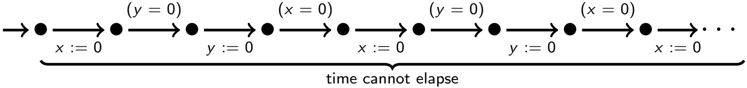
Our approach to non-Zenoness

A path in $\mathrm{ZG}^{\mathfrak{a}}(A)$ **yields only Zeno runs** iff:

1. some clock x is **blocking**:



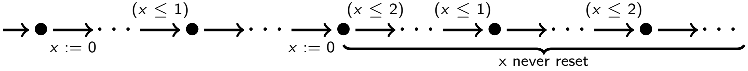
2. or **time cannot elapse** due to **zero-checks**:



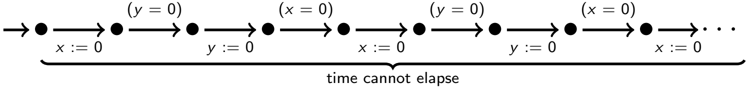
Our approach to non-Zenoness

A path in $ZG^{\mathfrak{a}}(A)$ **yields only Zeno runs** iff:

1. some clock x is **blocking**:

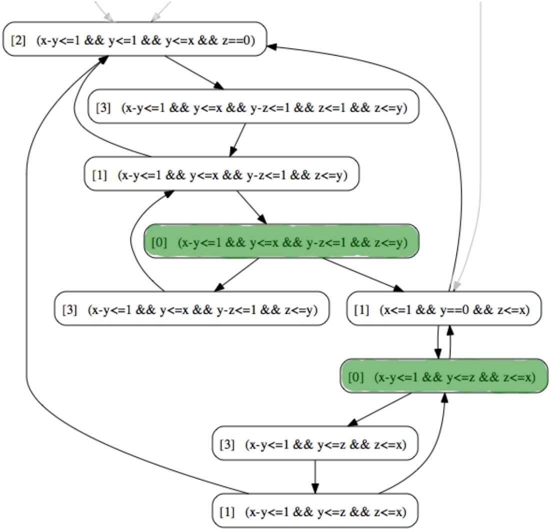


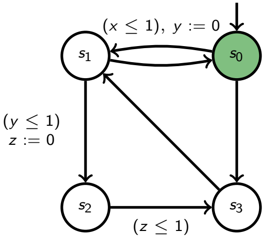
2. or **time cannot elapse** due to **zero-checks**:



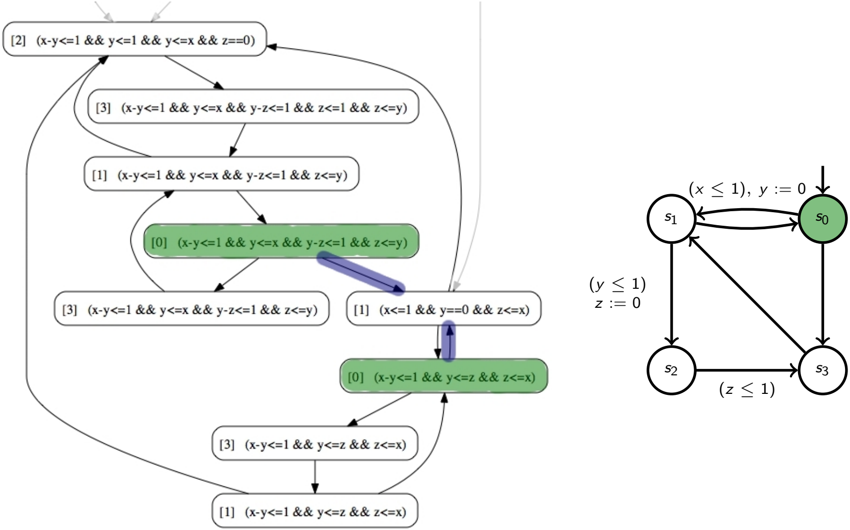
- Idea : define **conditions on SCC** in $ZG^{\mathfrak{a}}(A)$ to detect those two situations

The Case of Blocking Clocks (no $x = 0$)

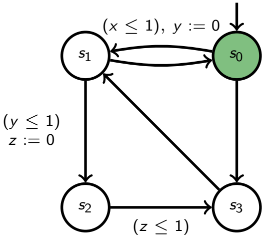




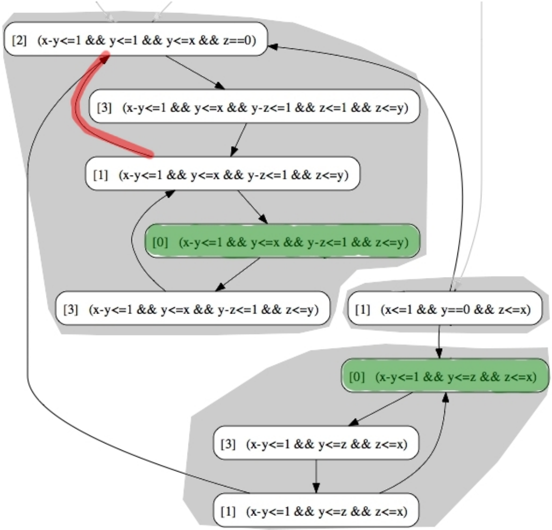
The Case of Blocking Clocks (no $x = 0$)



The Case of Blocking Clocks (no $x = 0$)



The Case of Blocking Clocks (no $x = 0$)



[2] (x-y<=1 && y<=1 && y<=x && z==0)

[3] (x-y<=1 && y<=x && y-z<=1 && z<=1 && z<=y)

[1] (x-y<=1 && y<=x && y-z<=1 && z<=y)

[0] (x-y<=1 && y<=x && y-z<=1 && z<=y)

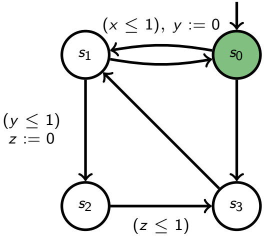
[3] (x-y<=1 && y<=x && y-z<=1 && z<=y)

[1] (x<=1 && y==0 && z<=x)

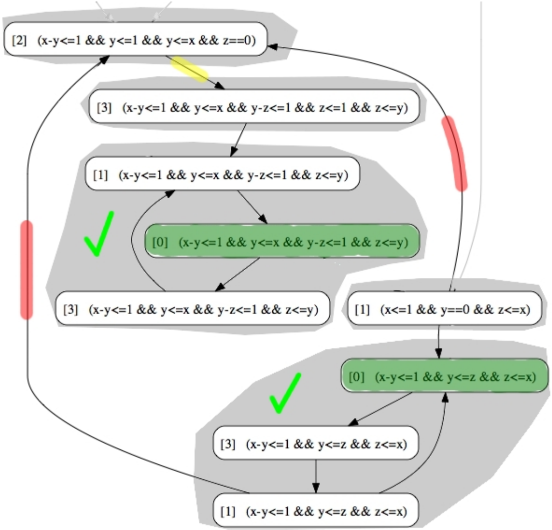
[0] (x-y<=1 && y<=z && z<=x)

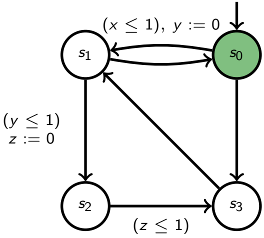
[3] (x-y<=1 && y<=z && z<=x)

[1] (x-y<=1 && y<=z && z<=x)

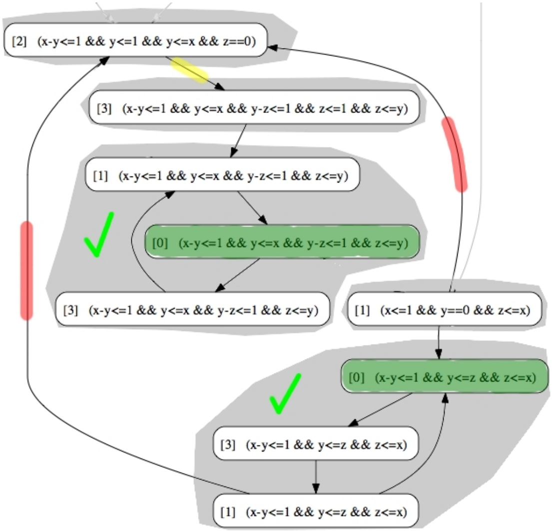


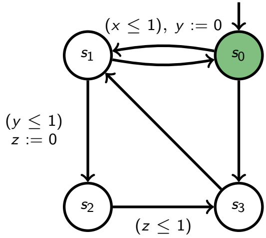
The Case of Blocking Clocks (no $x = 0$)





The Case of Blocking Clocks (no $x = 0$)



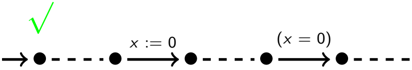


Blocking clocks are detected in time $|ZG^{\alpha}(A)|.(|X|+1)$

Detecting zero-checks ($x = 0$)

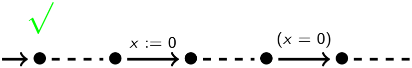
Can time **elapse** here?

Detecting zero-checks ($x = 0$)



Can time **elapse** here?

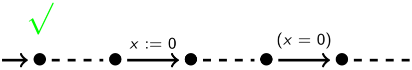
Detecting zero-checks ($x = 0$)



Problem: detect nodes where **time can elapse**

Solution: each zero-check must be **preceded** by a reset

Detecting zero-checks ($x = 0$)



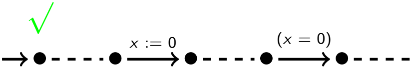
Problem: detect nodes where **time can elapse**

Solution: each zero-check must be **preceded** by a reset

Guessing zone graph ($\mathrm{GZG}^{\mathfrak{a}}$)

- Each node (q, Z, Y) has a **guess set** $Y \subseteq X$
- $(q, Z, Y) \xrightarrow{x:=0} (q', Z', Y \cup \{x\})$
- $(q, Z, Y) \xrightarrow{(x=0)}$ enabled if $x \in Y$

Detecting zero-checks ($x = 0$)



Problem: detect nodes where **time can elapse**

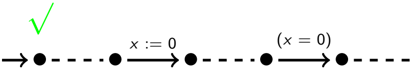
Solution: each zero-check must be **preceded** by a reset

Guessing zone graph ($\mathrm{GZG}^{\mathfrak{a}}$)

- Each node (q, Z, Y) has a **guess set** $Y \subseteq X$
- $(q, Z, Y) \xrightarrow{x:=0} (q', Z', Y \cup \{x\})$
- $(q, Z, Y) \xrightarrow{(x=0)}$ enabled if $x \in Y$

A node $(q, Z, \emptyset)$ is **clear** for **time elapse**.

Detecting zero-checks ($x = 0$)



Problem: detect nodes where **time can elapse**

Solution: each zero-check must be **preceded** by a reset

Guessing zone graph (GZG$^{\mathfrak{a}}$)

- Each node (q, Z, Y) has a **guess set** $Y \subseteq X$
- $(q, Z, Y) \xrightarrow{x:=0} (q', Z', Y \cup \{x\})$
- $(q, Z, Y) \xrightarrow{(x=0)}$ enabled if $x \in Y$
- $(q, Z, Y) \xrightarrow{\tau} (q, Z, \emptyset)$, to **forget** guesses

A node $(q, Z, \emptyset)$ is **clear** for **time elapse**.

Algorithm

Theorem

A has a non-Zeno run iff there is an **unblocked** path in $\mathrm{GZG}^{\mathfrak{a}}(A)$ with **infinitely many nodes that have** $Y = \emptyset$.

- Equivalent: find an **SCC** in $\mathrm{GZG}^{\mathfrak{a}}(A)$ that has an **accepting** node and a **clear** node, and that is **unblocked**
- Recall : blocking clocks can be detected in time $|\mathrm{GZG}^{\mathfrak{a}}(A)|.(|X| + 1)$

Size of $\mathsf{GZG}^{\mathfrak{a}}(A)$

$2^{|X|}$ **more nodes** in $\mathsf{GZG}^{\mathfrak{a}}(A)$ than in $\mathsf{ZG}^{\mathfrak{a}}(A)$ **due to** Y **sets**?

Size of $\mathrm{GZG}^{\mathfrak{a}}(A)$

$2^{|X|}$ **more nodes** in $\mathrm{GZG}^{\mathfrak{a}}(A)$ than in $\mathrm{ZG}^{\mathfrak{a}}(A)$ **due to** Y **sets**?

Theorem

- For each reachable node (q, Z), Z entails a **total order** on X.

Size of $\mathrm{GZG}^{\mathfrak{a}}(A)$

$2^{|X|}$ **more nodes** in $\mathrm{GZG}^{\mathfrak{a}}(A)$ than in $\mathrm{ZG}^{\mathfrak{a}}(A)$ **due to** Y **sets**?

Theorem

- For each reachable node (q, Z), Z entails a **total order** on X.
- Extra_{α} **preserves the order**.
- $\mathrm{Extra}_{\alpha}^{+}$ preserves order on **relevant** clocks.

Size of $\mathrm{GZG}^{\mathfrak{a}}(A)$

$2^{|X|}$ **more nodes** in $\mathrm{GZG}^{\mathfrak{a}}(A)$ than in $\mathrm{ZG}^{\mathfrak{a}}(A)$ **due to** Y **sets**?

Theorem

- For each reachable node (q, Z), Z entails a **total order** on X.
- Extra_{α} **preserves the order**.
- $\mathrm{Extra}_{\alpha}^{+}$ preserves order on **relevant** clocks.
- Y **respects** this order.

For every (q, Z) only $|X| + 1$ guess sets.

Size of $\mathrm{GZG}^{\mathfrak{a}}(A)$

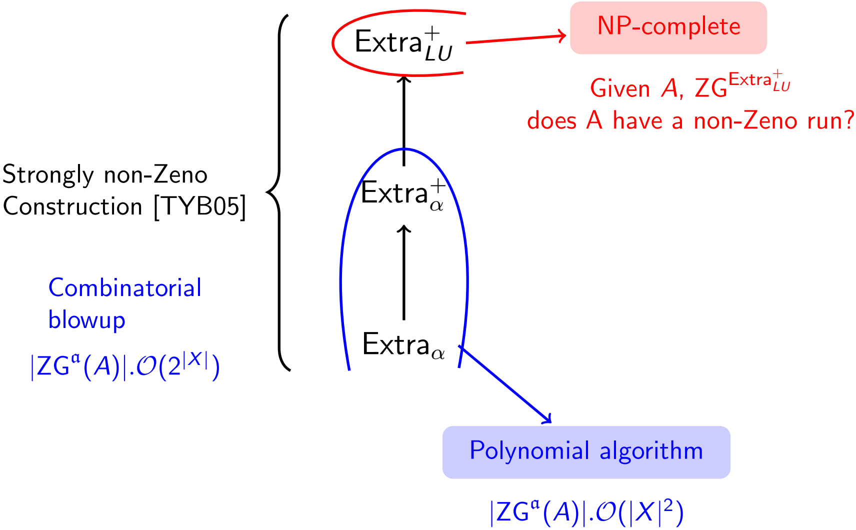
$2^{|X|}$ **more nodes** in $\mathrm{GZG}^{\mathfrak{a}}(A)$ than in $\mathrm{ZG}^{\mathfrak{a}}(A)$ **due to** Y **sets**?

Theorem

- For each reachable node (q, Z), Z entails a **total order** on X.
- Extra_{α} **preserves the order**.
- $\mathrm{Extra}_{\alpha}^{+}$ preserves order on **relevant** clocks.
- Y **respects** this order.

For every (q, Z) only $|X| + 1$ guess sets.

Extra_{LU}^{+} **does not preserve order** even on relevant clocks.



Extra^+_{LU}

NP-complete

Given A, $ZG^{\text{Extra}^+_{LU}}$
does A have a non-Zeno run?

Strongly non-Zeno
Construction [TYB05]

Extra^+_{α}

Combinatorial
blowup

$|ZG^{\mathfrak{a}}(A)|.\mathcal{O}(2^{|X|})$

Extra_{α}

Polynomial algorithm

$|ZG^{\mathfrak{a}}(A)|.\mathcal{O}(|X|^2)$

Benchmarks

A	$\mathrm{ZG}^{\mathfrak{a}}(A)$	$\mathrm{ZG}^{\mathfrak{a}}(A')$		$\mathrm{GZG}^{\mathfrak{a}}(A)$		
	size	size	otf	size	otf	opt
Train-Gate2 (mutex)	134	194	194	400	400	**134**
Train-Gate2 (bound. resp.)	988	**227482**	352	3840	1137	292
Train-Gate2 (liveness)	100	217	35	298	53	33
Fischer3 (mutex)	1837	3859	3859	7292	7292	**1837**
Fischer4 (mutex)	46129	96913	96913	229058	229058	**46129**
Fischer3 (liveness)	1315	4962	52	5222	64	40
Fischer4 (liveness)	33577	147167	223	166778	331	207
FDDI3 (liveness)	508	1305	44	3654	79	42
FDDI5 (liveness)	6006	15030	90	67819	169	88
FDDI3 (bound. resp.)	6252	41746	59	52242	114	60
CSMA/CD4 (collision)	4253	7588	7588	20146	20146	**4253**
CSMA/CD5 (collision)	45527	80776	80776	260026	260026	**45527**
CSMA/CD4 (liveness)	3038	9576	1480	14388	3075	832
CSMA/CD5 (liveness)	32751	120166	8437	186744	21038	4841

- Combinatorial explosion may **occur** in practice
- **Optimized** use of $\mathrm{GZG}(A)$ gives best results

What about existence of Zeno runs?

Problem : Given A, $ZG^{\mathfrak{a}}(A)$, does A **have a Zeno** run?

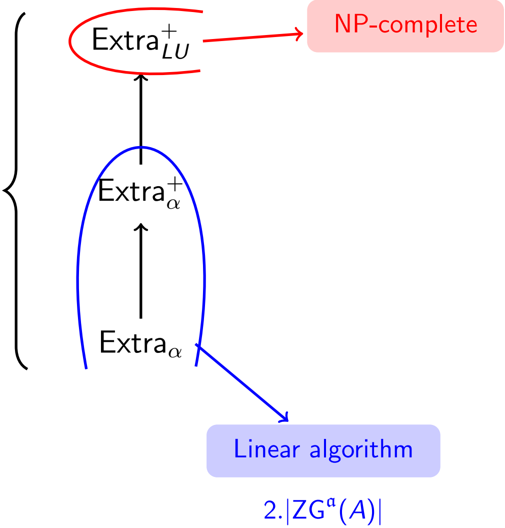
Syntactic criterion
on A [GB07]

Sufficient only

$$\begin{tikzcd}
\mathrm{Extra}^{+}_{LU} \\
\mathrm{Extra}^{+}_{\alpha} \arrow[u] \\
\mathrm{Extra}_{\alpha} \arrow[u]
\end{tikzcd}$$

What about existence of Zeno runs?

Problem : Given A, $ZG^{\mathfrak{a}}(A)$, does A **have a Zeno** run?



Conclusion & Future work

- Reachability : Efficient **implementation** of **non-convex** abstractions and **on-the-fly** learning of bounds
- Non-Zenoness :
  - **Combinatorial explosion** due to strongly non-Zeno construction
  - An $\mathcal{O}(|ZG^{\mathfrak{a}}(A)|.|X|^2)$ algorithm for Extra$_\alpha$, Extra$_\alpha^+$ and **NP-complete** for Extra$_{LU}^+$
- Zenoness : An $\mathcal{O}(|ZG^{\mathfrak{a}}|)$ algorithm for Extra$_\alpha$, Extra$_\alpha^+$ and **NP-complete** for Extra$_{LU}^+$

Future work

- Propagating **more** than constants
- Computing **non-Zeno** strategies for timed games
- Automata with **diagonal** constraints

Related papers

Using non-convex approximations for efficient analysis of timed automata
with F. Herbreteau, D. Kini, I. Walukiewicz (FSTTCS 2011)

Efficient emptiness check for timed Büchi automata
with F. Herbreteau, I. Walukiewicz (FMSD, CAV 2010 special issue)

Efficient on-the-fly emptiness check for timed Büchi automata
with F. Herbreteau (ATVA 2010)

Coarse abstractions make Zeno behaviours difficult to detect
with F. Herbreteau (CONCUR 2010)

Bibliography I

R. Alur and D.L. Dill.
A theory of timed automata.
Theoretical Computer Science, 126(2):183–235, 1994.

R. Alur and P. Madhusudan.
Decision problems for timed automata: A survey.
In *SFM-RT'04*, volume 3185 of *LNCS*, pages 1–24, 2004.

G. Behrmann, P. Bouyer, E. Fleury, and K. G. Larsen.
Static guard analysis in timed automata verification.
In *TACAS'03*, volume 2619 of *LNCS*, pages 254–270. Springer, 2003.

G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelanek.
Lower and upper bounds in zone-based abstractions of timed automata.
Int. Journal on Software Tools for Technology Transfer, 8(3):204–215, 2006.

P. Bouyer.
Forward analysis of updatable timed automata.
Form. Methods in Syst. Des., 24(3):281–320, 2004.

C. Courcoubetis and M. Yannakakis.
Minimum and maximum delay problems in real-time systems.
Form. Methods Syst. Des., 1(4):385–415, 1992.

C. Daws and S. Tripakis.
Model checking of real-time reachability properties using abstractions.
In *TACAS'98*, volume 1384 of *LNCS*, pages 313–329. Springer, 1998.

Bibliography II

R. Gómez and H. Bowman.
Efficient detection of zeno runs in timed automata.
In *Proc. 5th Int. Conf. on Formal Modeling and Analysis of Timed Systems, FORMATS 2007*, volume 4763 of *LNCS*, pages 195–210, 2007.

Guangyuan Li.
Checking timed büchi automata emptiness using lu-abstractions.
In Joël Ouaknine, editor, *Formal modeling and analysis of timed systems. 7th Int. Conf. (FORMATS)*, volume 5813 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2009.

S. Tripakis.
Verifying progress in timed systems.
In *Proc. 5th Int. AMAST Workshop, ARTS'99*, volume 1601 of *LNCS*, pages 299–314. Springer, 1999.

S. Tripakis.
Checking timed büchi emptiness on simulation graphs.
ACM Transactions on Computational Logic, 10(3):??–??, 2009.

S. Tripakis, S. Yovine, and A. Bouajjani.
Checking timed büchi automata emptiness efficiently.
Formal Methods in System Design, 26(3):267–292, 2005.