Abstractions for timed automata

work done with F. Herbreteau, I. Walukiewicz and D.Kini

B. Srivathsan

Ph.D. defence

Jury

Ahmed Bouajjani

Patricia Bouyer

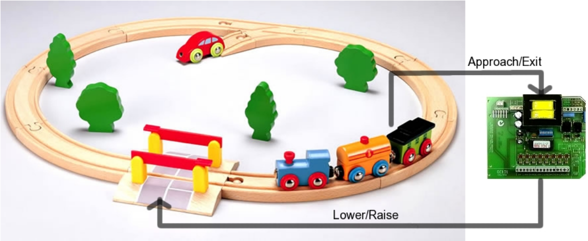
Bruno Courcelle

Frédéric Herbreteau Advisor

Joost-Pieter Katoen

Igor Walukiewicz Advisor

James Worrell

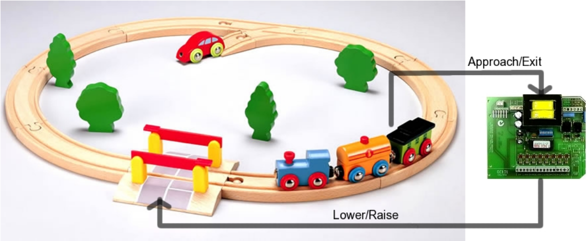


Reachability: Does something **bad** happen?

Liveness: Does something **good** happen **repeatedly**?

A THEORY OF TIMED AUTOMATA

R. Alur and D.L. Dill, *TCS'94*



Reachability: Does something **bad** happen?

UPPAAL, KRONOS, RED, IF, PAT, Rabbit ...

Liveness: Does something **good** happen **repeatedly**?

PROFOUNDER, CTAV ...

A THEORY OF TIMED AUTOMATA
R. Alur and D.L. Dill, *TCS'94*

In this thesis...

We revisit **reachability** and **liveness** problems for Alur-Dill timed automata

Reachability

Reachability

Liveness

Liveness

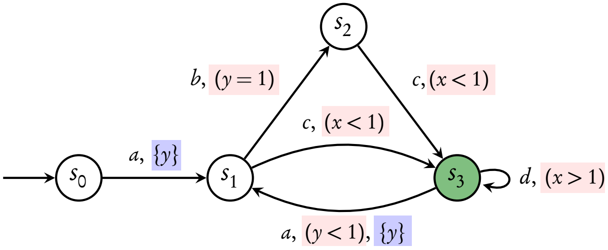
Reachability

Reachability

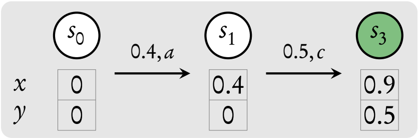
Liveness

Liveness

Timed Automata



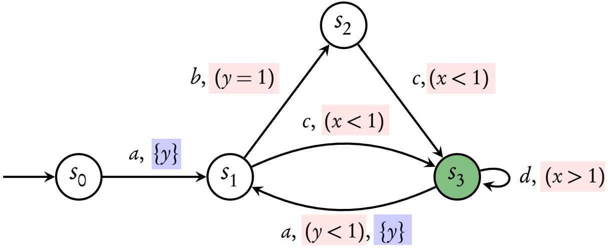
Run: finite sequence of transitions



- **accepting** if ends in green state

Reachability problem

Given a TA, does it **have** an **accepting** run

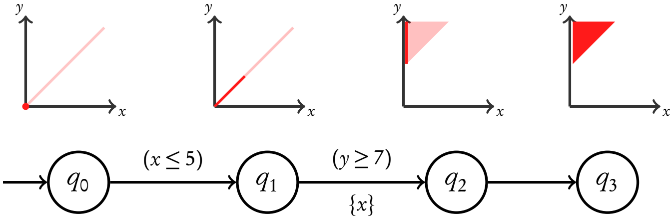


Theorem [AD94]

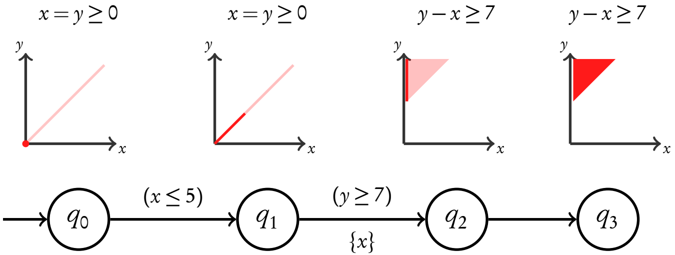
This problem is **PSPACE-complete**

first solution based on Regions

Key idea: Maintain **sets of valuations** reachable along a path

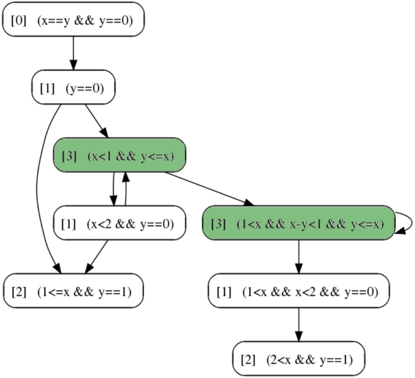


Key idea: Maintain **sets of valuations** reachable along a path



Easy to describe **convex** sets

Zones and zone graph



- Zone: set of valuations defined by conjunctions of constraints:

$$x \sim c$$
$$x - y \sim c$$

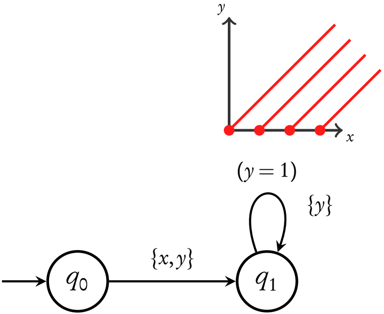
e.g. $(x - y \geq 1) \wedge (y < 2)$

- Representation: by DBM [Dil89]

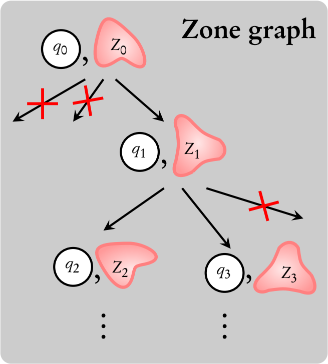
Sound and complete [DT98]

Zone graph preserves state **reachability**

Problem of non-termination

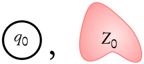


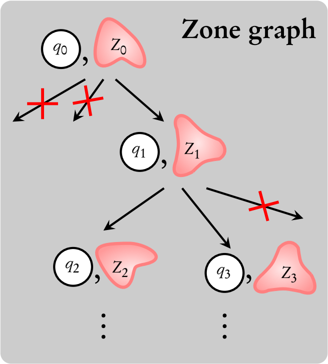
Abstractions



potentially infinite...

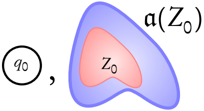
Abstractions

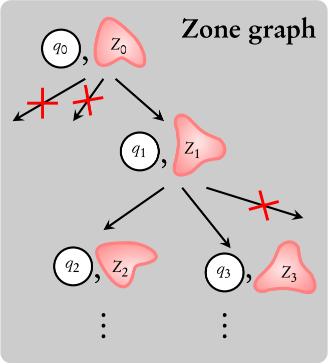




potentially infinite...

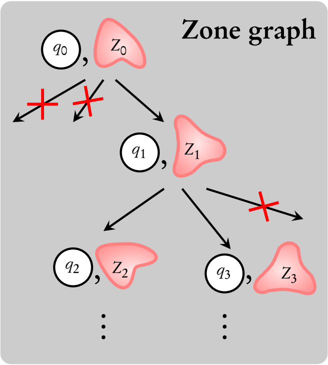
Abstractions



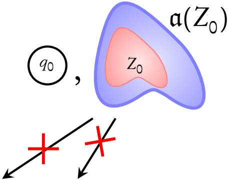


potentially infinite...

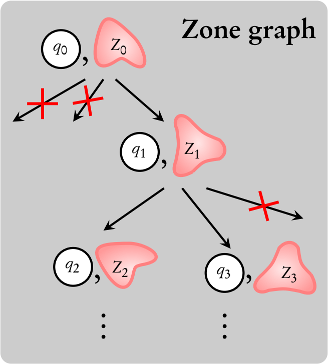
Abstractions



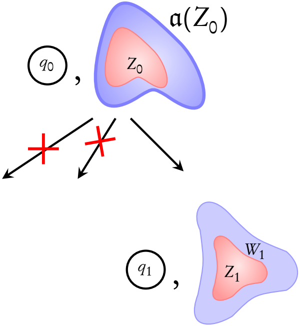
potentially infinite...



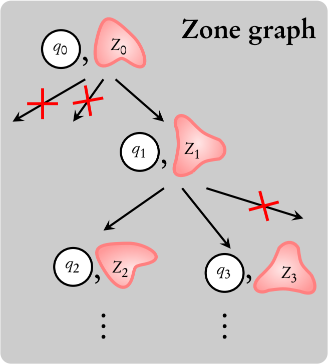
Abstractions



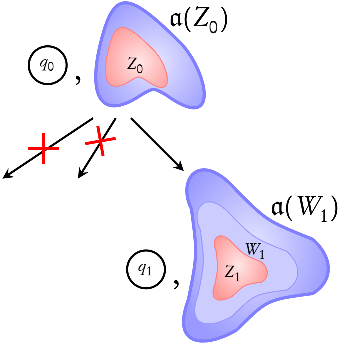
potentially infinite...



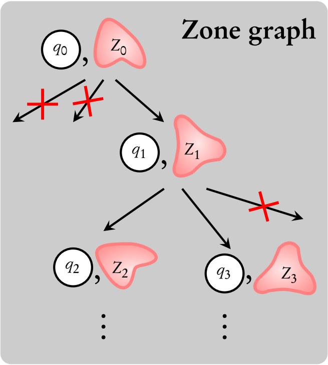
Abstractions



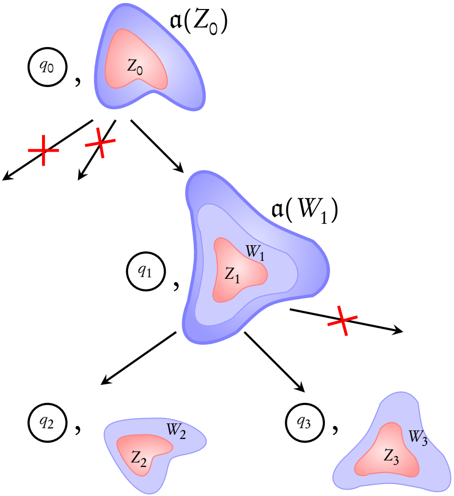
potentially infinite...



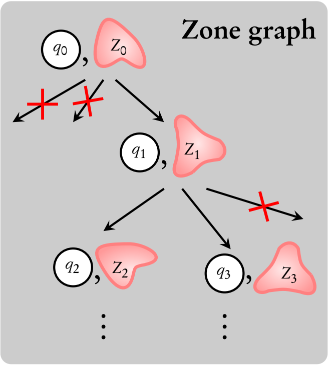
Abstractions



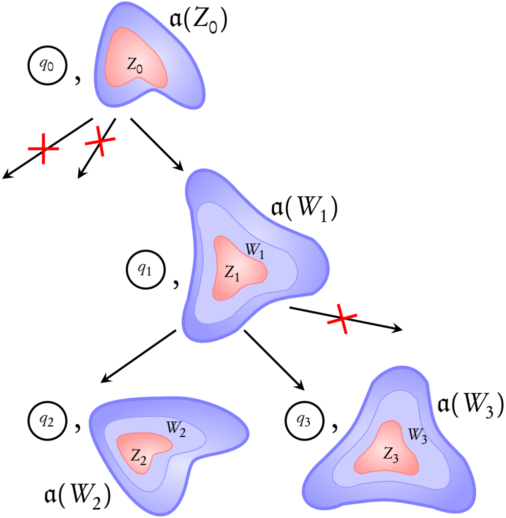
potentially infinite...



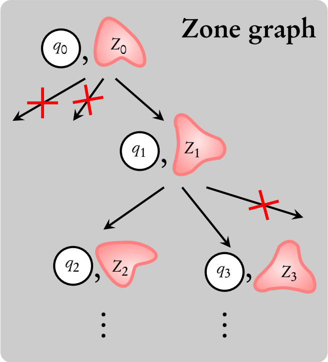
Abstractions



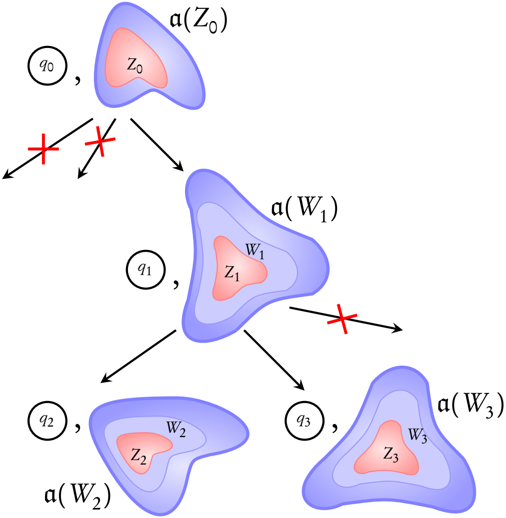
potentially infinite...



Abstractions

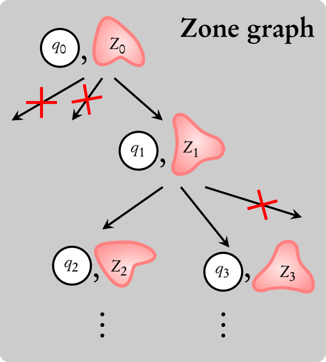


potentially infinite...

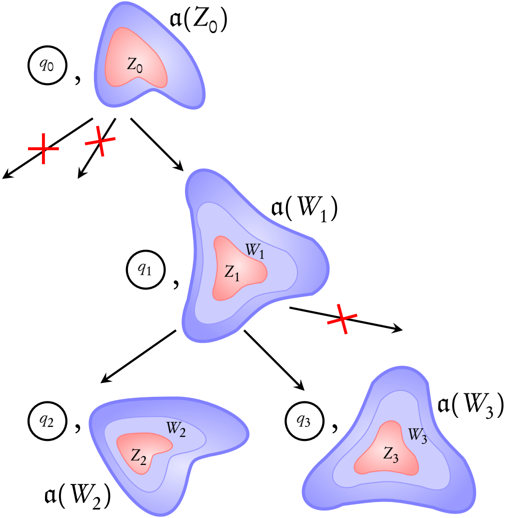


Find $\mathfrak{a}$ such that number of **abstracted** sets is **finite**

Abstractions



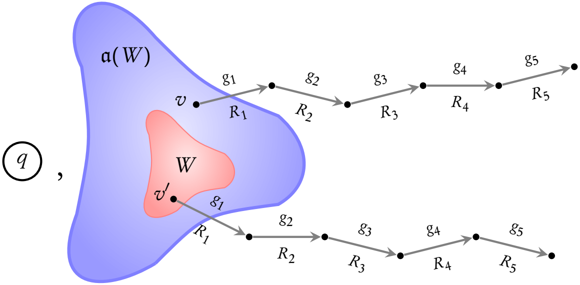
potentially infinite...



Coarser the abstraction, **smaller** the abstracted graph

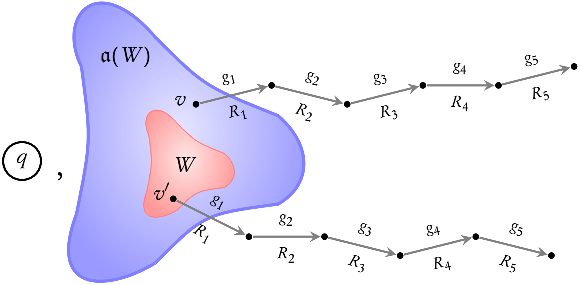
Condition 1: Abstractions should have **finite range**

Condition 2: Abstractions should be sound ⇒ $\mathfrak{a}(W)$ can contain only valuations **simulated** by W



Condition 1: Abstractions should have **finite range**

Condition 2: Abstractions should be sound $\Rightarrow$ $\mathfrak{a}(W)$ can contain only valuations **simulated** by W

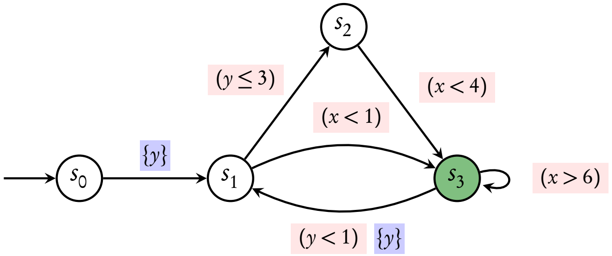


Question: Why not add **all** the valuations **simulated** by W?

Bounds and abstractions

Theorem [LS00]

Coarsest simulation relation is **EXPTIME-hard**



Bounds and abstractions

Theorem [LS00]

Coarsest simulation relation is **EXPTIME-hard**

$(y \leq 3)$

$(x < 4)$

$(x < 1)$

$(x > 6)$

$(y < 1)$

Bounds and abstractions

Theorem [LS00]

Coarsest simulation relation is **EXPTIME-hard**

$(y \leq 3)$

$(x < 4)$

$(x < 1)$

$(x > 6)$

$(y < 1)$

M-bounds [AD94]

$M(x) = 6,\ M(y) = 3$

$v \preccurlyeq_M v'$

Bounds and abstractions

Theorem [LS00]

Coarsest simulation relation is **EXPTIME-hard**

$(y \leq 3)$ $(x < 4)$

$(x < 1)$

$(x > 6)$

$(y < 1)$

M-bounds [AD94]

$M(x) = 6,\ M(y) = 3$

$v \preccurlyeq_M v'$

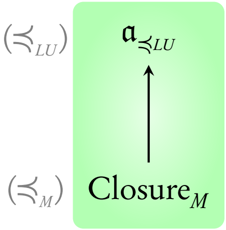
LU-bounds [BBLP04]

$L(x) = 6,\ L(y) = -\infty$

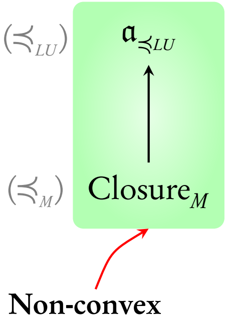
$U(x) = 4,\ U(y) = 3$

$v \preccurlyeq_{LU} v'$

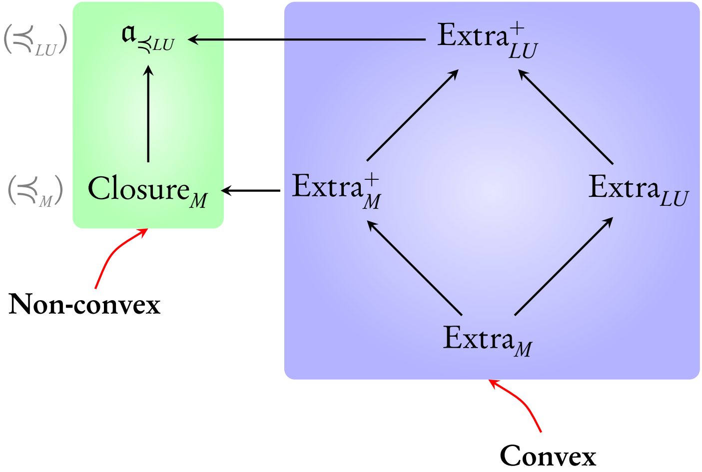
Abstractions in literature [BBLP04, Bou04]



Abstractions in literature [BBLP04, Bou04]



Abstractions in literature [BBLP04, Bou04]



Only convex abstractions used in **implementations!**

Non-convex abstr.

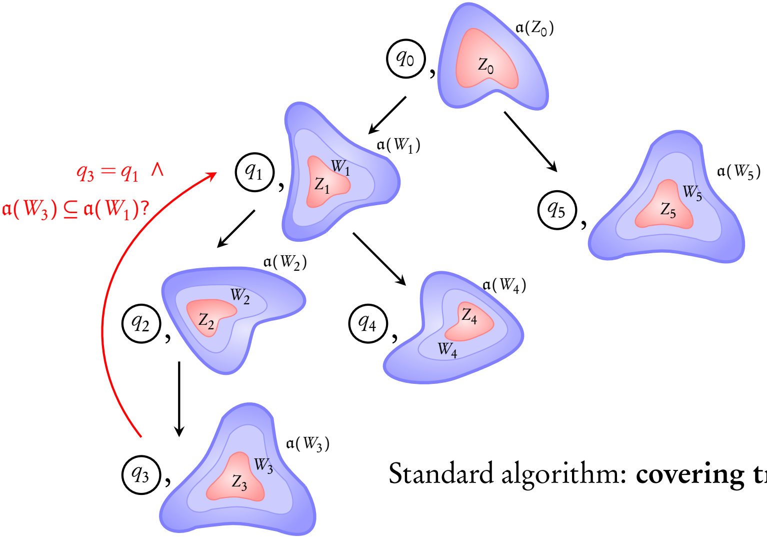
Reachability

Liveness

Liveness

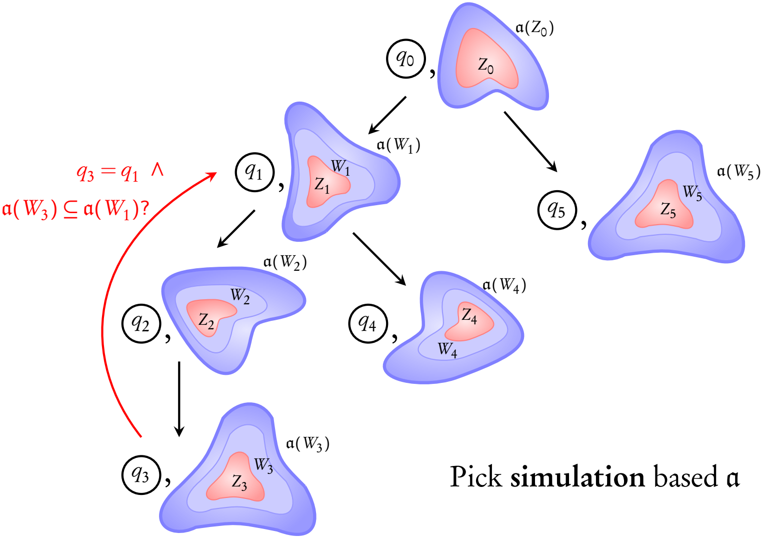
Step 1: We can use abstractions **without storing** them

Using non-convex abstractions



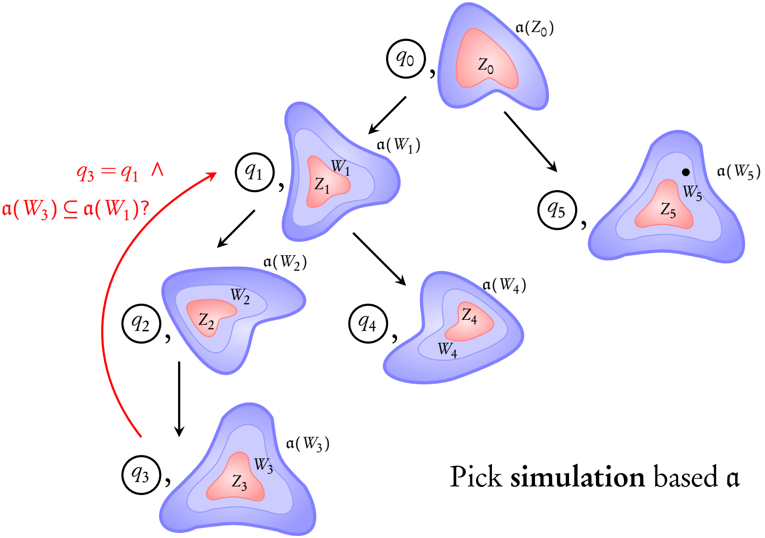
Standard algorithm: **covering tree**

Using non-convex abstractions



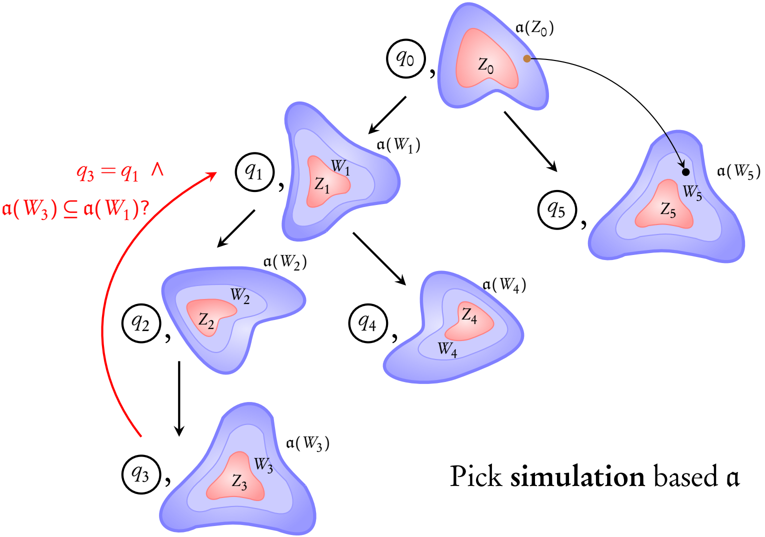
Pick **simulation** based $\mathfrak{a}$

Using non-convex abstractions



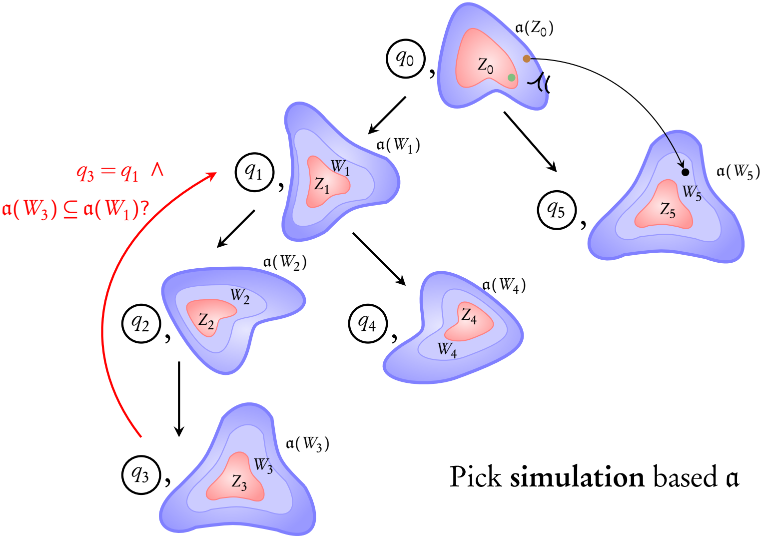
Pick **simulation** based $\mathfrak{a}$

Using non-convex abstractions



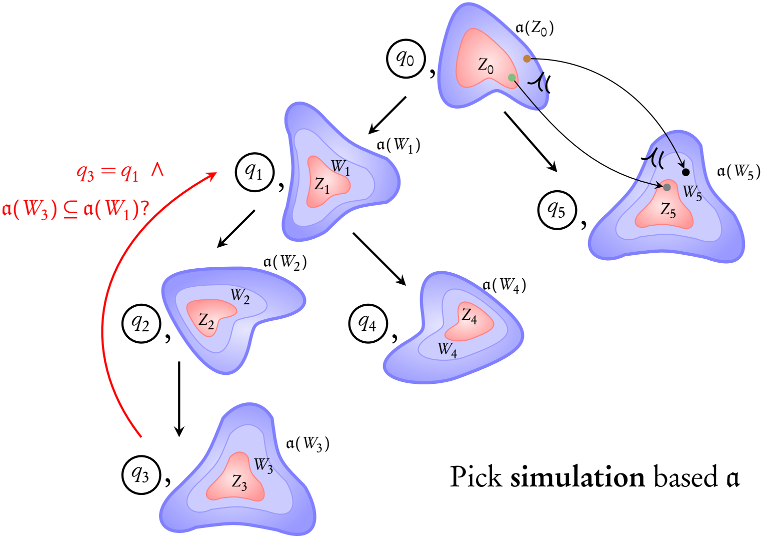
Pick **simulation** based $\mathfrak{a}$

Using non-convex abstractions



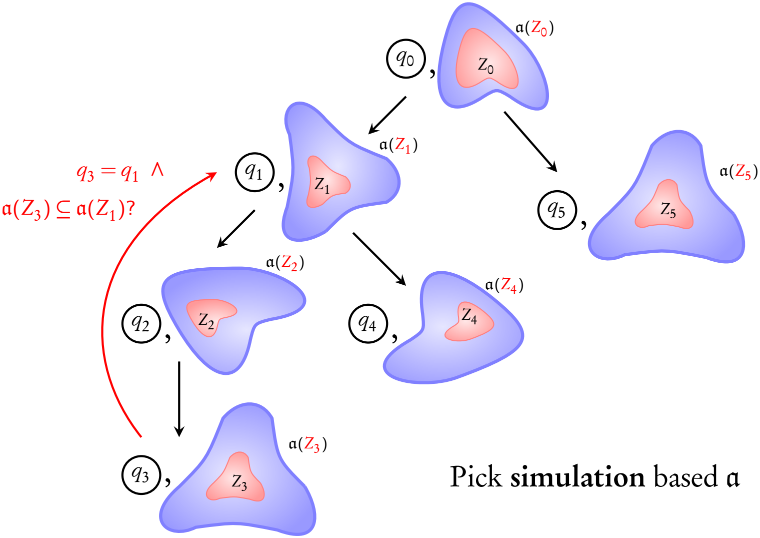
Pick **simulation** based $\mathfrak{a}$

Using non-convex abstractions



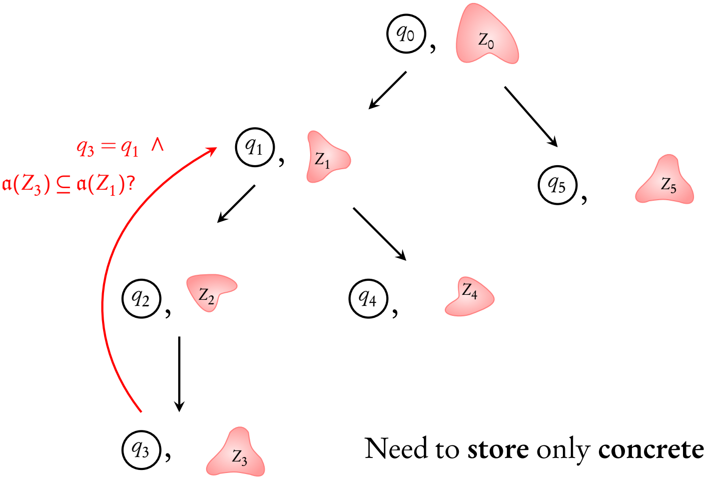
Pick **simulation** based $\mathfrak{a}$

Using non-convex abstractions



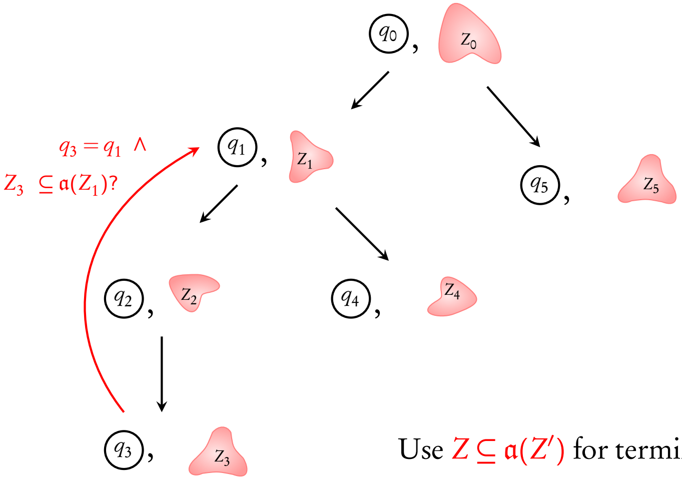
Pick **simulation** based $\mathfrak{a}$

Using non-convex abstractions



Need to **store** only **concrete** semantics

Using non-convex abstractions



Use $Z \subseteq \mathfrak{a}(Z')$ for termination

Step 1: We can use abstractions **without storing** them

Step 2: We can do the **inclusion** test **efficiently**

Efficient inclusion testing

Main result

$Z \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\mathbf{Proj}_{xy}(Z) \not\subseteq \mathfrak{a}_{\preccurlyeq_{LU}}(\mathbf{Proj}_{xy}(Z'))$$

Efficient inclusion testing

Main result

$Z \nsubseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\mathbf{Proj}_{xy}(Z) \nsubseteq \mathfrak{a}_{\preccurlyeq_{LU}}(\mathbf{Proj}_{xy}(Z'))$$

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

Efficient inclusion testing

Main result

$Z \nsubseteq \mathfrak{a}_{\preccurlyeq_{LU}}(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\mathbf{Proj}_{xy}(Z) \nsubseteq \mathfrak{a}_{\preccurlyeq_{LU}}(\mathbf{Proj}_{xy}(Z'))$$

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

Same complexity as $Z \subseteq Z'$!

Efficient inclusion testing

Main result

$Z \nsubseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ if and only if there **exist 2 clocks** x, y s.t.

$$\mathbf{Proj}_{xy}(Z) \nsubseteq \mathfrak{a}_{\preccurlyeq LU}(\mathbf{Proj}_{xy}(Z'))$$

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

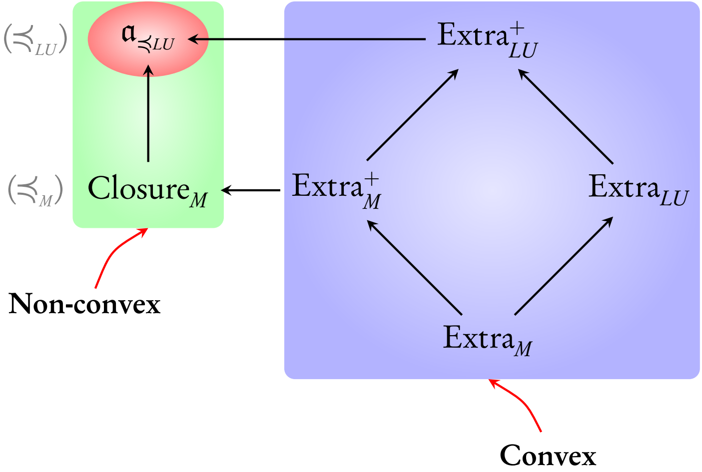
Same complexity as $Z \subseteq Z'$!

Slightly modified comparison works!

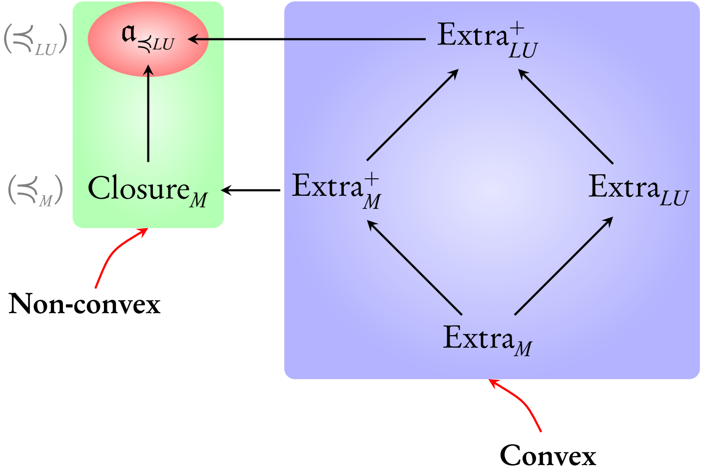
Step 1: We can use abstractions **without storing** them

Step 2: We can do the **inclusion** test **efficiently**

$\Rightarrow$ **new algorithm** for reachability



$(\precsim_{LU})$
$\mathfrak{a}_{\precsim LU}$
Extra^+_{LU}
$(\precsim_M)$
Closure_M
Extra^+_M
Extra_{LU}
Extra_M
Non-convex
Convex



Question: Can we do better than $\mathfrak{a}_{\preccurlyeq LU}$?

Optimality

LU-automata: automata with guards **determined by** L and U

Theorem

The $\mathfrak{a}_{\preccurlyeq_{LU}}$ abstraction is the **biggest abstraction** that is **sound** and **complete** for all LU-automata.

Non-convex abstr.

- Efficient use
- Optimality

Reachability

Liveness

Liveness

Non-convex abstr.

Efficient use

Optimality

Reachability

Liveness

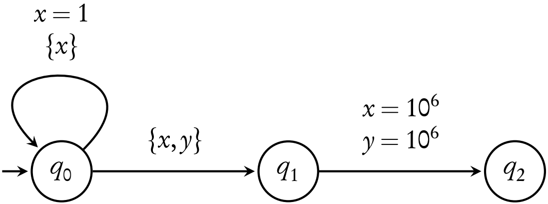
Liveness

Question: If $\mathfrak{a}_{\preccurlyeq_{LU}}$ is the best, can we do better?

Question: If $\mathfrak{a}_{\preccurlyeq_{LU}}$ is the best, can we do better?

Get better **LU-bounds!**

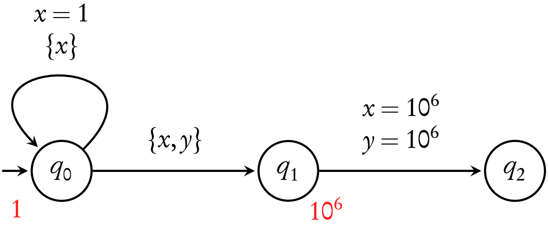
Global LU-bounds



Naive: $L_x = U_x = 10^6$, $L_y = U_y = 10^6$

Size of graph $\sim 10^6$

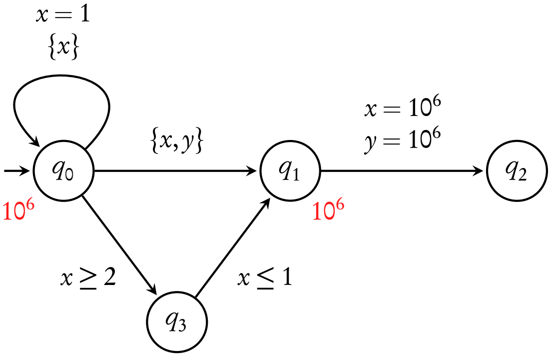
Static analysis: bounds for every q [BBFL03]



Size of graph < 10

Static analysis: bounds for every q

[BBFL03]

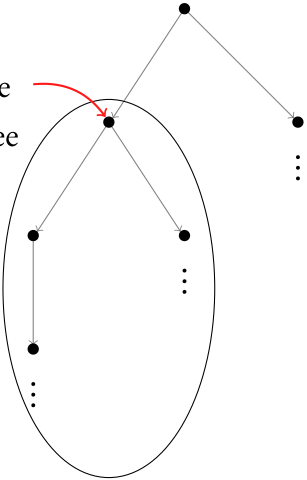


Size of graph $\sim 10^6$

Need to look at **semantics**...

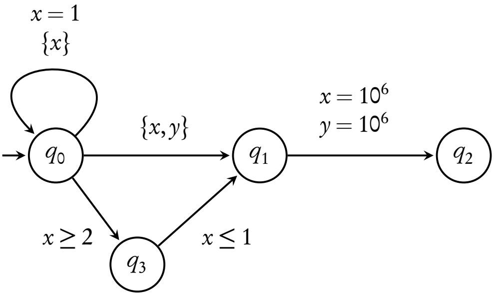
LU bounds for every (q, Z) in zone graph

constants at node
depend on the subtree



Constant propagation

Contribution: A new **on-the-fly** algorithm to **learn** constants during exploration



Theorem (Correctness)

An accepting state is reachable in $\mathscr{A}$ iff the constant propagation algorithm reaches a node with accepting state and a non-empty zone.

Non-convex abstr.

Efficient use

Optimality

Bounds

On-the-fly

Liveness

Liveness

Benchmarks

Model	Our algorithm		UPPAAL's algorithm		UPPAAL 4.1.3 (-n4 -C -o1)	
	nodes	s.	nodes	s.	nodes	s.
CSMA/CD7	5046	0.39	5923	0.30	–	T.O.
CSMA/CD8	16609	0.75	19017	1.16	–	T.O.
CSMA/CD9	54467	9.40	60783	4.53	–	T.O.
FDDI10	459	0.04	525	0.05	12049	2.43
FDDI20	1719	0.41	2045	0.82	–	T.O.
FDDI30	3779	1.70	4565	3.90	–	T.O.
Fischer7	7737	0.40	18353	0.48	18374	0.35
Fischer8	25080	1.50	85409	2.31	85438	1.53
Fischer9	81035	5.70	397989	12.05	398685	8.95
Fischer10	–	T.O.	–	T.O.	1827009	53.44

- $\mathbf{Extra}^{+}_{LU}$ and **static** analysis bounds in UPPAAL
- $\mathfrak{a}_{\preccurlyeq LU}$ and **otf** bounds in our algorithm

Non-convex abstr.

Efficient use

Optimality

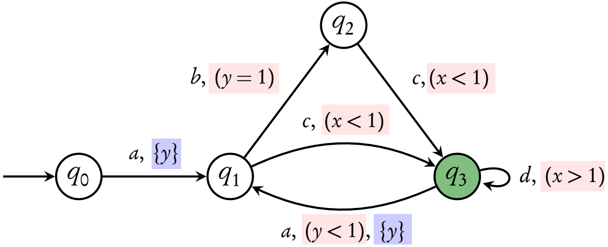
Bounds

On-the-fly

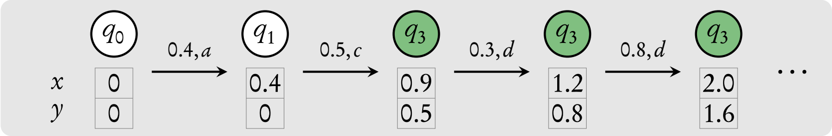
Liveness

Liveness

Timed Büchi automata



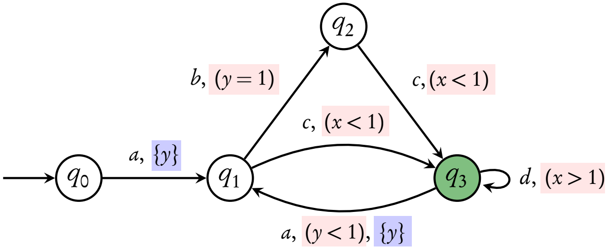
Run: infinite sequence of transitions



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

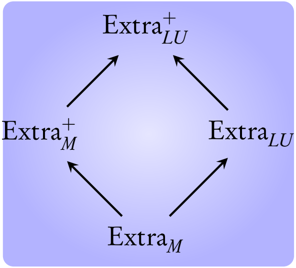
Büchi non-emptiness problem

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



Theorem [AD94]

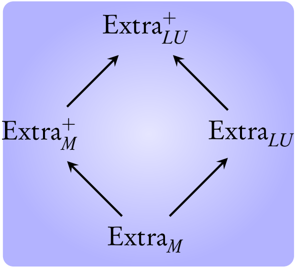
This problem is **PSPACE-complete**



$$
\begin{array}{lccccccc}
ZG^{\mathfrak{a}}(\mathscr{A}): & (q_0, Z_0) & \rightarrow & (q_1, Z_1) & \rightarrow & (q_2, Z_2) & \longrightarrow & \cdots \\
 & \Cup & & \Cup & & \Cup & & \\
\mathscr{A}: & (q_0, v_0) & \rightarrow & (q_1, v_1) & \rightarrow & (q_2, v_2) & \longrightarrow & \cdots
\end{array}
$$

Sound and complete [Tri09, Li09]

All the above abstractions preserve **repeated state reachability**



$$
\begin{array}{lccccccc}
ZG^{\mathfrak{a}}(\mathscr{A}): & (q_0, Z_0) & \rightarrow & (q_1, Z_1) & \rightarrow & (q_2, Z_2) & \longrightarrow & \cdots \\
 & \Cup & & \Cup & & \Cup & & \\
\mathscr{A}: & (q_0, v_0) & \rightarrow & (q_1, v_1) & \rightarrow & (q_2, v_2) & \longrightarrow & \cdots
\end{array}
$$

Sound and complete [Tri09, Li09]

All the above abstractions preserve **repeated state reachability**

What about **non-Zenoness**?

Adding a clock for non-Zenoness [TYB05]

$\mathbf{A}'$: strongly non-Zeno TBA

$|X| + 1$ clocks and at most $2 \cdot |Q|$ states

Theorem [TYB05]

A has a non-Zeno accepting run iff $ZG^{\mathfrak{a}}(\mathbf{A}')$ has an **accepting** run

Adding a clock for non-Zenoness [TYB05]

$\mathbf{A}'$: strongly non-Zeno TBA
$|X| + 1$ clocks and at most $2 \cdot |Q|$ states

Theorem [TYB05]

$\mathbf{A}$ has a non-Zeno accepting run iff $ZG^{\mathfrak{a}}(\mathbf{A}')$ has an **accepting** run

Question: Is this good enough?

Adding a clock for non-Zenoness [TYB05]

$\mathbf{A}'$: strongly non-Zeno TBA
$|X|+1$ clocks and at most $2 \cdot |Q|$ states

Theorem [TYB05]

$\mathbf{A}$ has a non-Zeno accepting run iff $ZG^{\mathfrak{a}}(\mathbf{A}')$ has an **accepting** run

Contribution: The construction can give exponential blowup

Theorem

There exists an automaton $\mathscr{A}_n$ with n clocks for which

$$|ZG^{\mathfrak{a}}(\mathscr{A}_n')| = \mathscr{O}(2^n) \cdot |ZG^{\mathfrak{a}}(\mathscr{A}_n)|$$

Non-convex abstr.

- Efficient use
- Optimality

Bounds

On-the-fly

Non-Zenoness

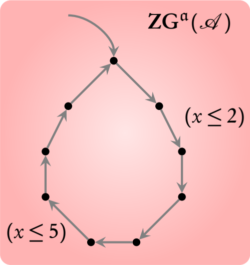
Adding 1 clock is costly

Liveness

Coming next: A **new construction** for non-Zenoness

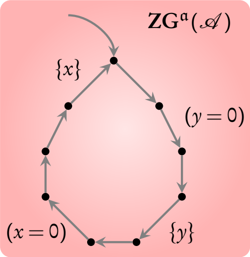
New construction

When does a path in $ZG^{a}(\mathscr{A})$ **yield only Zeno runs**?



Blocking clocks

x never reset but checked for upper bound



Zero-checks

x and y should be 0 all along the path

Zero-checks

Can time elapse here?

Zero-checks

Time can elapse at a node if

every zero-check is **preceded** by a reset

Zero-checks

Time can elapse at a node if

every zero-check is **preceded** by a reset

Guessing Zone Graph ($GZG^{\mathfrak{a}}(\mathcal{A})$) :

$$(q, Z, Y) \xrightarrow{\{x\}} (q', Z', Y \cup \{x\})$$

$$(q, Z, Y) \xrightarrow{(x=0)} \text{enabled only if } x \in Y$$

$$(q, Z, Y) \xrightarrow{\tau} (q, Z, \emptyset)$$

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** $Y = \emptyset$.

Complexity: $|\mathrm{GZG}^{\mathfrak{a}}(A)| \cdot (|X| + 1)$

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

$2^{|X|}$ **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_M, Extra_M^+ **preserve the order**.
- Y **respects** this order; only $|X| + 1$ sets needed.

$2^{|X|}$ **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_M, Extra_M^+ **preserve the order**.
- Y **respects** this order; only $|X| + 1$ sets needed.

Extra_{LU}, Extra_{LU}^+ **do not preserve order**

Theorem

Non-Zenoness from LU-abstract zone graphs is **NP-complete**

Theorem

A **slight weakening** of Extra_{LU}, Extra_{LU}^+ **preserves** order

Non-convex abstr.

Efficient use

Optimality

Bounds

On-the-fly

Non-Zenoness

Adding 1 clock is costly

New construction

NP-complete for LU

Liveness

Benchmarks

A	$ZG^{\mathfrak{a}}(A)$	$ZG^{\mathfrak{a}}(A')$		$GZG^{\mathfrak{a}}(A)$		
	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 $GZG^{\mathfrak{a}}(A)$ gives best results

Non-convex abstr.

Efficient use

Optimality

LICS'12, FSTTCS'11

Bounds

On-the-fly

FSTTCS'11

Non-Zenoness

Adding 1 clock is costly

New construction

NP-complete for LU

CAV'10 + ATVA'10 (FMSD'12), CONCUR'11

Zenoness

First complete algorithm

NP-complete for LU

CONCUR'11

Perspectives

- **More** than LU
- Automata with **diagonal** constraints
- Probabilistic timed automata, priced timed automata

- Non-Zeno strategies for **timed games**

References I

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

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. Larsen, and R. Pelánek.
Lower and upper bounds in zone based abstractions of timed automata.
Tools and Algorithms for the Construction and Analysis of Systems, pages 312–326, 2004.

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

D. Dill.
Timing assumptions and verification of finite-state concurrent systems.
In *AVMFSS*, volume 407 of *LNCS*, pages 197–212. Springer, 1989.

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.

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.

References II

François Laroussinie and Ph. Schnoebelen.
The state explosion problem from trace to bisimulation equivalence.
In *Proceedings of the Third International Conference on Foundations of Software Science and Computation Structures*, FOSSACS '00, pages 192–207. Springer-Verlag, 2000.

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.