Efficient Reachability in Timed Automata

F. Herbreteau

Univ. Bordeaux, LaBRI, CNRS, UMR 5800

Séminaire DTIM Onera, Toulouse
January 28th, 2013

Joint work with: D. Kini, B. Srivathsan, I. Walukiewicz

Outline

Timed Automata and the Reachability Problem

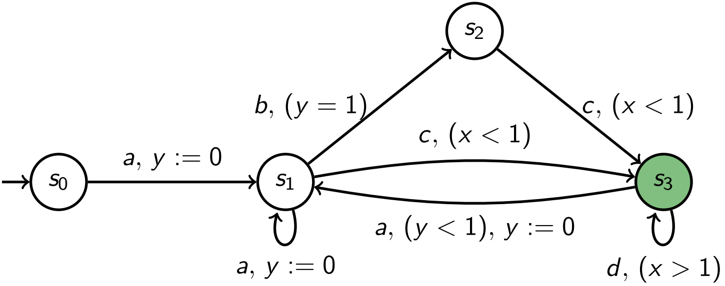
Symbolic semantics and abstractions

Bounds based abstractions

Small bounds for abstractions

Conclusions and future work

Timed Automata [AD94]



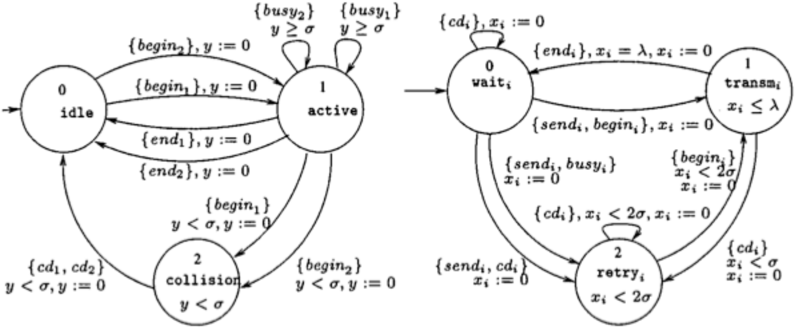
Run: finite **sequence** of transitions,

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

Accepting run (reachability): ends in a green state.

Example #1: the CSMA/CD protocol

Property to check: detection of **collisions**



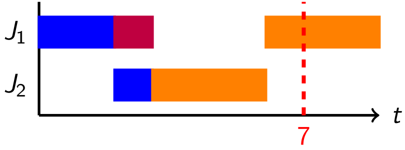
Reachability of a state with *collision* and *wait*$_1$ or *wait*$_2$?

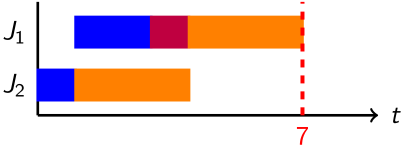
Example #2: scheduling jobs (1/2)

- Jobs **compete** to execute tasks on machines

$$J_1 : \ (m_1, 2)\,(m_2, 1)\,(m_3, 3) \qquad\qquad J_2 : \ (m_1, 1)\,(m_3, 3)$$

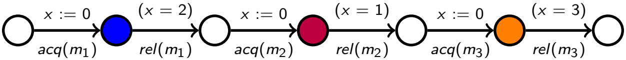
- Can the jobs be **scheduled within** $7s$?





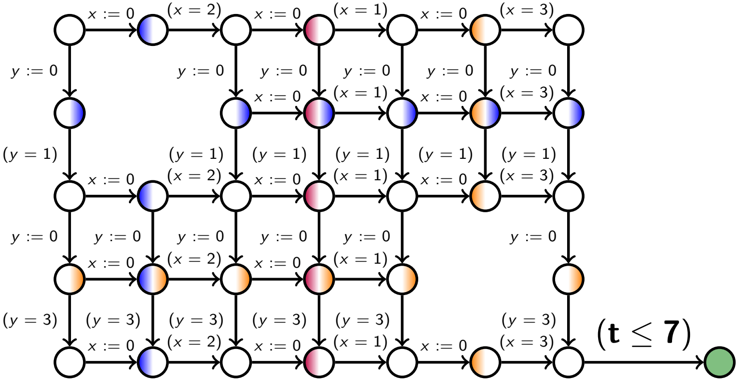
Example #2: scheduling jobs (2/2)

$J_1: \; (m_1, 2)\,(m_2, 1)\,(m_3, 3)$



Example #2: scheduling jobs (2/2)

J_1 : $(m_1, 2)\,(m_2, 1)\,(m_3, 3)$ J_2 : $(m_1, 1)\,(m_3, 3)$ within 7s.



Reachability of the green state?

The problem we are interested in ...

Problem (Emptiness/State reachability)

Given a TA and a state q, is q **reachable**?

The problem we are interested in ...

Problem (Emptiness/State reachability)

Given a TA and a state q, is q **reachable**?

Restriction: guards only involve **integer constants**

Theorem ([AD94, CY92])

This problem is **PSPACE-complete**

The problem we are interested in ...

Problem (Emptiness/State reachability)

Given a TA and a state q, is q **reachable**?

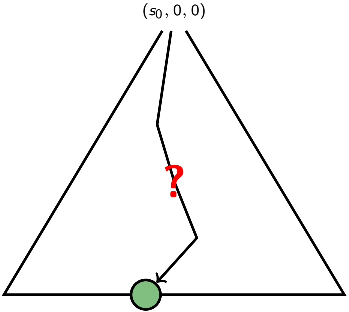
Restriction: guards only involve **integer constants**

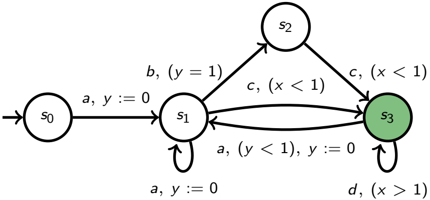
Theorem ([AD94, CY92])

This problem is **PSPACE-complete**

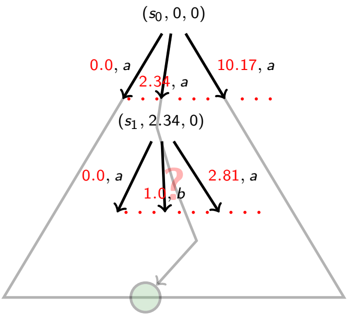
This talk: **challenges** and **advances** for solving reachability in Timed Automata **efficiently**

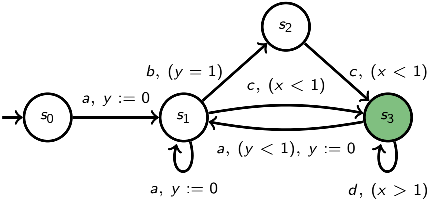
Solving the reachability problem





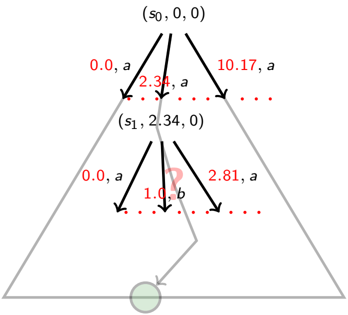
Solving the reachability problem

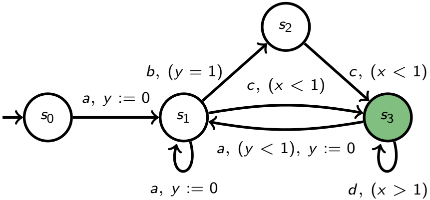




Search space = **reachability tree**

Solving the reachability problem





Search space = **reachability tree**

Uncountable branching due to **density** of time

Solution: tree over **sets of valuations** instead of valuations

Outline

Timed Automata and the Reachability Problem

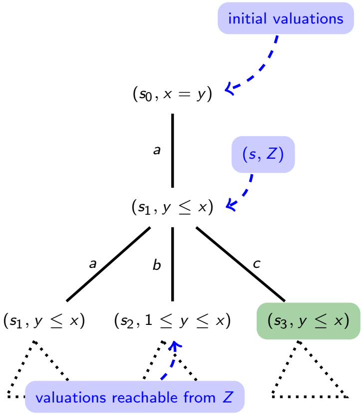
Symbolic semantics and abstractions

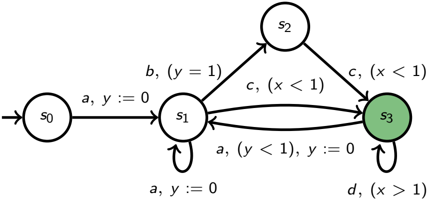
Bounds based abstractions

Small bounds for abstractions

Conclusions and future work

Symbolic reachability tree

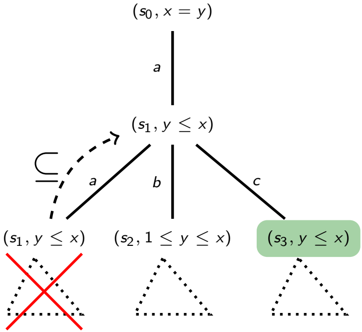


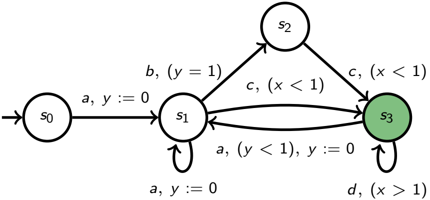


- **Zone:** set of valuations with **efficient symbolic representation** by DBMs

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

Symbolic reachability tree



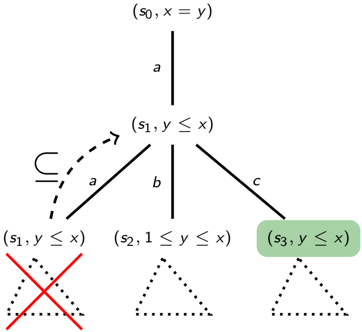


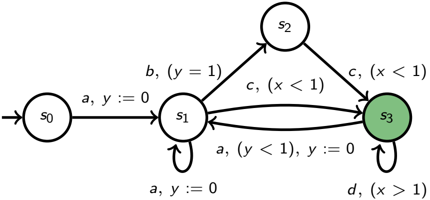
- **Zone:** set of valuations with **efficient symbolic representation** by DBMs

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

- **Covering tree** ($\subseteq$ wrt zones)

Symbolic reachability tree





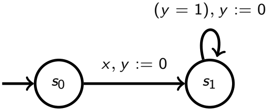
- **Zone:** set of valuations with **efficient symbolic representation** by DBMs

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

- **Covering tree** ($\subseteq$ wrt zones)

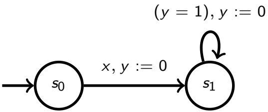
The tree may be **infinite**!

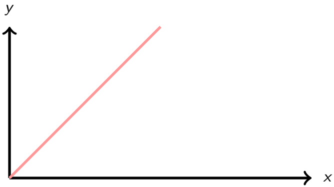
The tree may be infinite



$(s_0,\ x - y = 0)$

The tree may be infinite

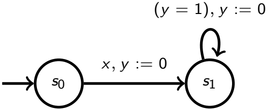


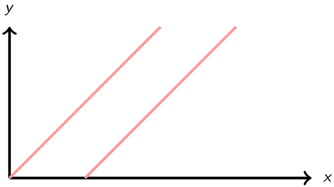


$(s_0,\ x - y = 0)$

$(s_1,\ x - y = 0)$

The tree may be infinite



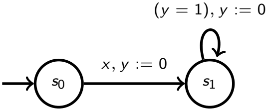


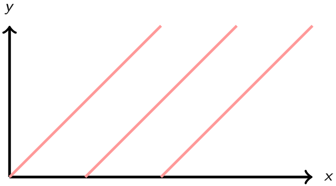
$(s_0, \ x - y = 0)$

$(s_1, \ x - y = 0)$

$(s_1, \ x - y = 1)$

The tree may be infinite





$(s_0,\ x - y = 0)$

|

$(s_1,\ x - y = 0)$

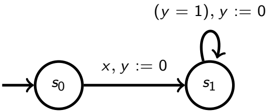
|

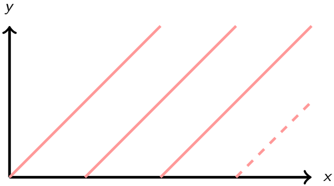
$(s_1,\ x - y = 1)$

|

$(s_1,\ x - y = 2)$

The tree may be infinite





$(s_0, \ x - y = 0)$

$(s_1, \ x - y = 0)$

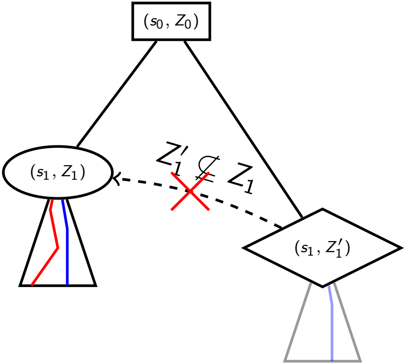
$(s_1, \ x - y = 1)$

$(s_1, \ x - y = 2)$

$(s_1, \ x - y = 3)$

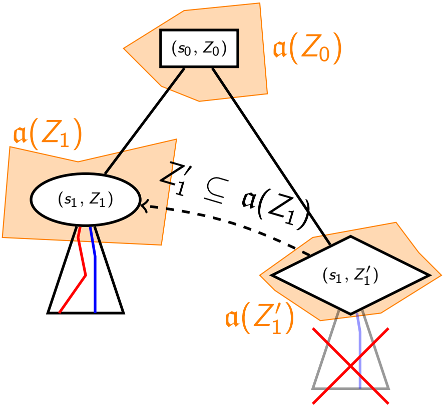
$\vdots$

Introducing abstractions



Don't explore (s_1, Z_1'): all its runs are possible from (s_1, Z_1)

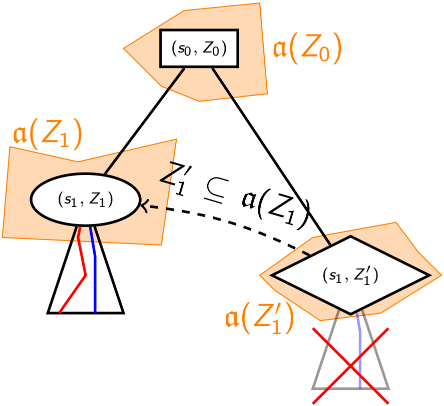
Introducing abstractions



Don't explore (s_1, Z_1'): all its runs are possible from (s_1, Z_1)

Correctness: abstractions **preserve runs**, only add **"equivalent"** valuations

Introducing abstractions



Don't explore (s_1, Z_1'): all its runs are possible from (s_1, Z_1)

Correctness: abstractions **preserve runs**, only add **"equivalent"** valuations

Termination: ensure **finitely many** abstracted zones

Outline

Timed Automata and the Reachability Problem

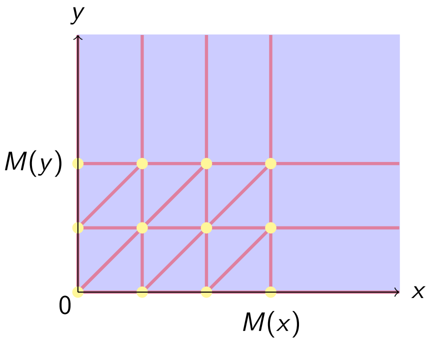
Symbolic semantics and abstractions

Bounds based abstractions

Small bounds for abstractions

Conclusions and future work

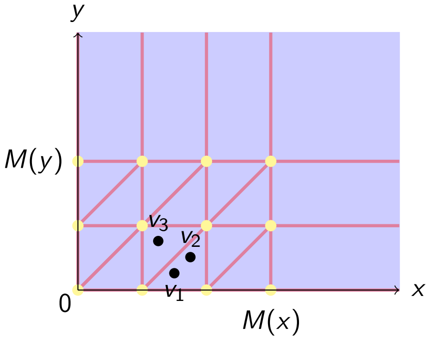
Regions [AD94]



Bound M: guards $x \leq c$, $x \geq c$ only use constants $c \leq M(x)$

Region: set of valuations that enable the **same sequences of transitions**

Regions [AD94]



Bound M: guards $x \leq c$, $x \geq c$ only use constants $c \leq M(x)$

Region: set of valuations that enable the **same sequences of transitions**

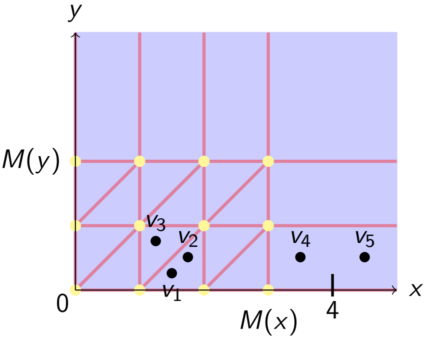
Correct:

$v_1 \xrightarrow{x \geq 2} \xrightarrow{y \leq 1}$

$v_2 \xrightarrow{x \geq 2} \xrightarrow{y \leq 1}$

$v_3 \xrightarrow{x \geq 2} \quad \xrightarrow{y \leq 1}$

Regions [AD94]



Bound M: guards $x \leq c$, $x \geq c$ only use constants $c \leq M(x)$

Region: set of valuations that enable the **same sequences of transitions**

Correct:

$v_1 \xrightarrow{x \geq 2} \xrightarrow{y \leq 1}$

$v_2 \xrightarrow{x \geq 2} \xrightarrow{y \leq 1}$

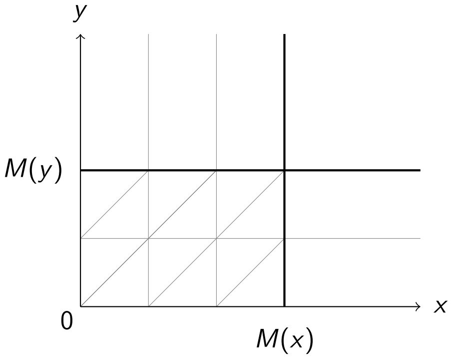
$v_3 \xrightarrow{x \geq 2} \xrightarrow{y \leq 1}$

Incorrect:

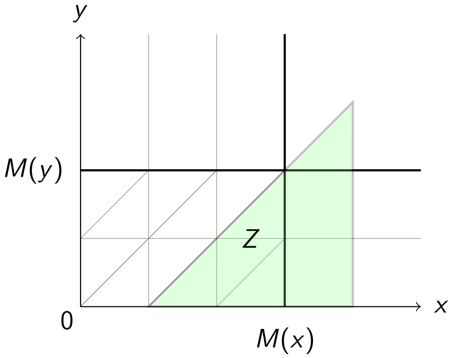
$v_4 \xrightarrow{x \leq 4}$

$v_5 \xrightarrow{x \leq 4}$

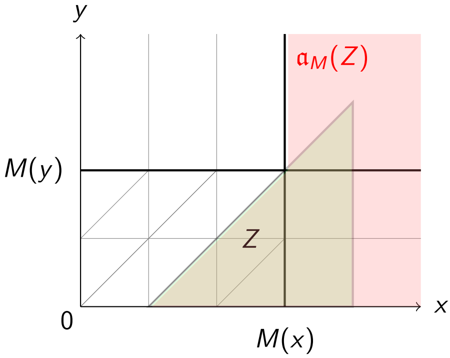
Region based abstraction



Region based abstraction

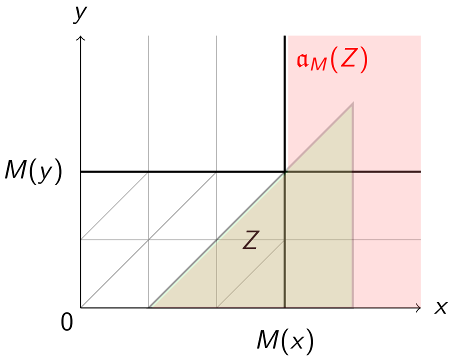


Region based abstraction



$\mathfrak{a}_M(Z)$ is the **union of regions** that Z intersects

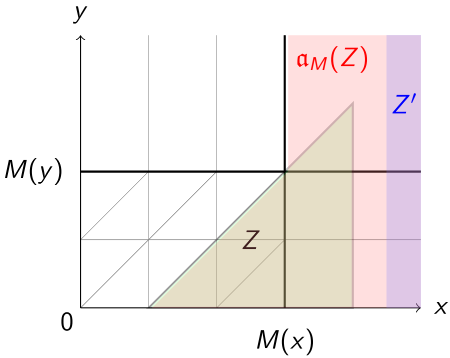
Region based abstraction

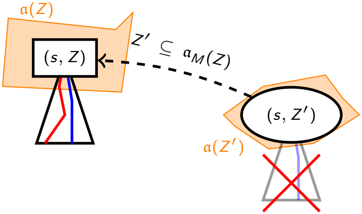


$\mathfrak{a}_M(Z)$ is the **union of regions** that Z intersects

- **Correctness:** Z and $\mathfrak{a}_M(Z)$ have the same executions
- **Termination:** finitely many regions

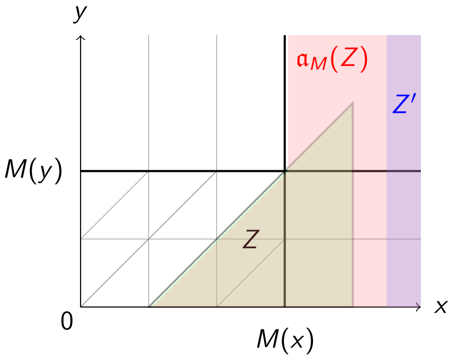
Region based abstraction

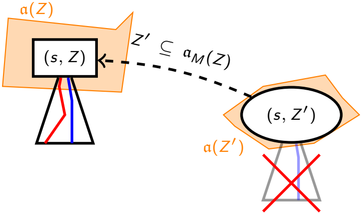




Early termination: $Z' \not\subseteq Z$ but $Z' \subseteq \mathfrak{a}_M(Z)$

Region based abstraction



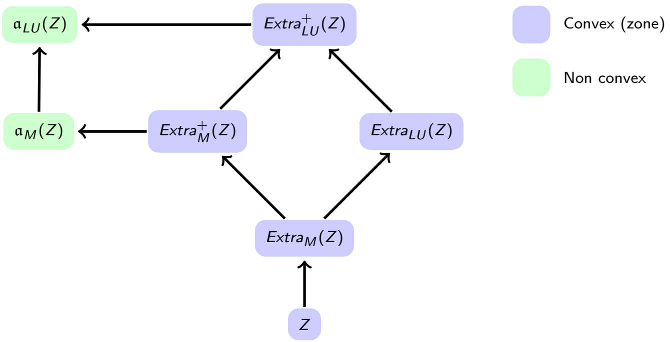


Early termination: $Z' \not\subseteq Z$ but $Z' \subseteq \mathfrak{a}_M(Z)$

$\mathfrak{a}_M(Z)$ may **not** be **convex**: how to check inclusion?

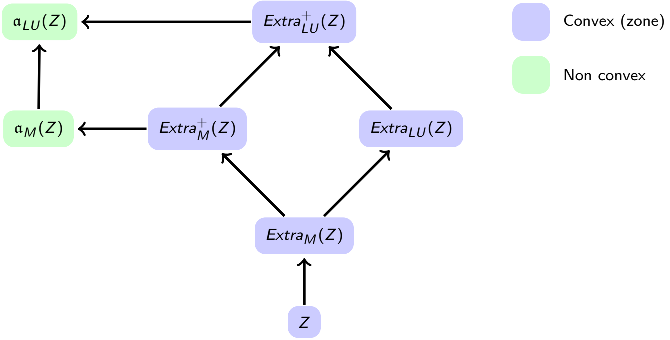
Abstractions [DT98, BBLP06]

Standard restriction: use abstractions such that $\mathfrak{a}(Z)$ **is a zone** (inclusion in $\mathcal{O}(|X|^2)$)



Abstractions [DT98, BBLP06]

Standard restriction: use abstractions such that $\mathfrak{a}(Z)$ **is a zone** (inclusion in $\mathcal{O}(|X|^2)$)



Can we check $Z \subseteq \mathfrak{a}_{LU}(Z')$ **efficiently**?

Efficient algorithm for $\mathfrak{a}_{LU}$ and $\mathfrak{a}_M$

Theorem

$Z \subseteq \mathfrak{a}_{LU}(Z')$ is decided in $\mathcal{O}(|X|^2)$ (same as $\subseteq$)

Idea: do not compute $\mathfrak{a}_{LU}(Z)$

- define $\subseteq_{\mathfrak{a}_{LU}}$ s.t. $Z \subseteq_{\mathfrak{a}_{LU}} Z'$ iff $Z \subseteq \mathfrak{a}_{LU}(Z')$
- $\subseteq_{\mathfrak{a}_{LU}}$ is easy for **2 clocks**
- n clocks: check **all pairs** of clocks

Theorem

$\mathfrak{a}_{LU}$ is the **coarsest abstraction** if bounds are the only parameter

Outline

Timed Automata and the Reachability Problem

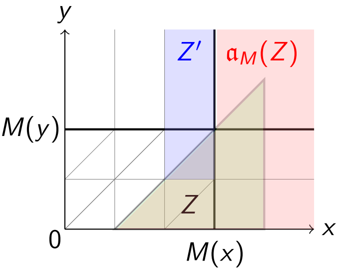
Symbolic semantics and abstractions

Bounds based abstractions

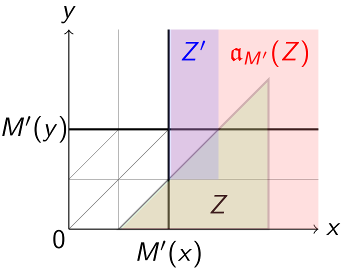
Small bounds for abstractions

Conclusions and future work

Back to regions

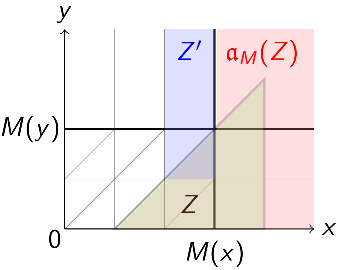


$Z' \not\subseteq \mathfrak{a}_M(Z)$

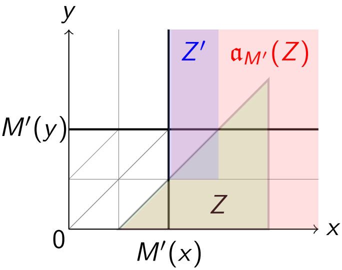


$Z' \subseteq \mathfrak{a}_{M'}(Z)$

Back to regions



$Z' \not\subseteq \mathfrak{a}_M(Z)$

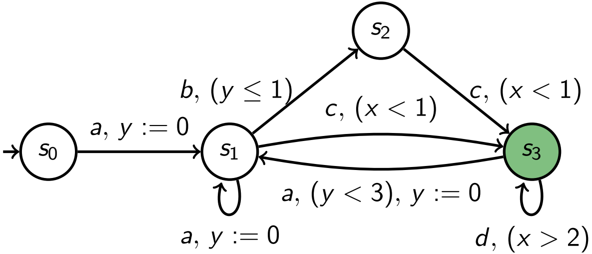


$Z' \subseteq \mathfrak{a}_{M'}(Z)$

Take the **smallest bounds** you can!

Recall: M **preserve** the sequence of transitions with guards $x \leq c$, $x \geq c$ that only use **constants** $c \leq M(x)$

Global bounds



Global bounds

$(y \leq 1)$ $(x < 1)$ $(x < 1)$

$(y < 3)$

$(x > 2)$

***M*-bounds [AD94]:**

	x	y
M	2	3

***LU*-bounds [BBLP06]:**

	x	y
L	2	$-\infty$
U	1	3

Global bounds

$(y \leq 1)$ $(x < 1)$ $(x < 1)$

$(y < 3)$

$(x > 2)$

M-bounds [AD94]:

	x	y
M	2	3

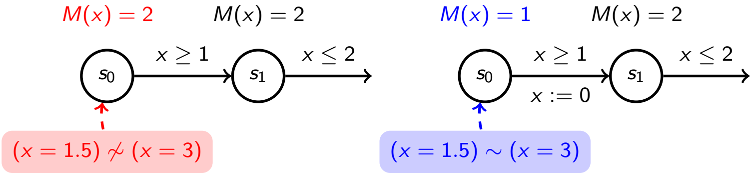
LU-bounds [BBLP06]:

	x	y
L	2	$-\infty$
U	1	3

$(s, v) \sim_M (s, v')$ iff they enable the **same sequences of transitions**

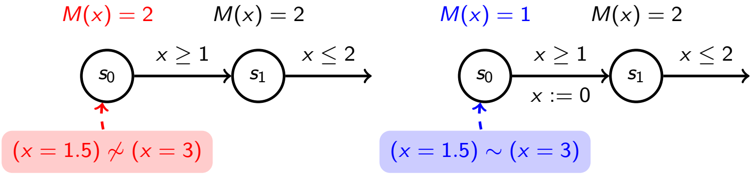
Local bounds [BBFL03]

Idea: bounds are **local to each state** in the automaton



Local bounds [BBFL03]

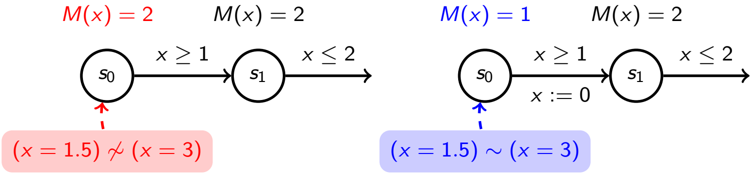
Idea: bounds are **local to each state** in the automaton



$(s, v) \sim_M (s, v')$ iff they enable the **same sequences of transitions**

Local bounds [BBFL03]

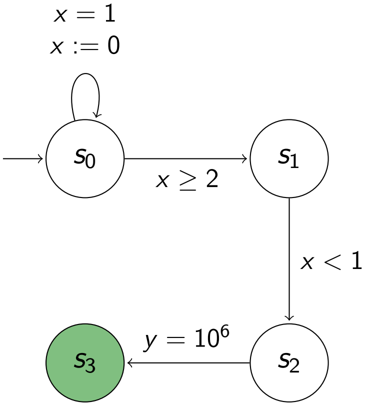
Idea: bounds are **local to each state** in the automaton



$(s, v) \sim_M (s, v')$ iff they enable the **same sequences of transitions**

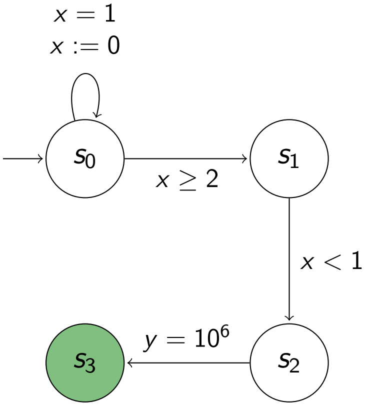
Computation: static analysis on the automaton

Better abstraction: look at semantics



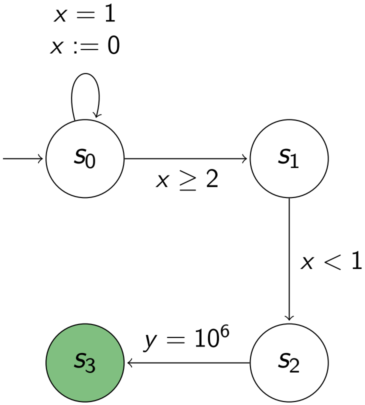
Better abstraction: look at semantics

Static analysis: $M(y) = 10^6$



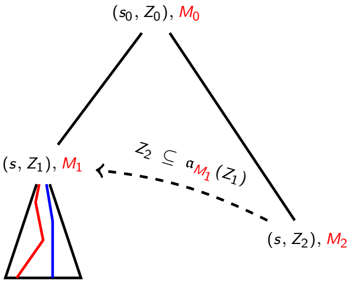
Better abstraction: look at semantics

Static analysis: $M(y) = 10^6$



More than $\mathbf{10^6}$ zones at s_0 **not necessary**!

On-the-fly bounds



- Bounds M **local to the nodes** in the reachability tree
- M are **updated on-the-fly** by propagation
- Abstraction using **local bounds**: $Z_2 \subseteq \mathfrak{a}_{M_1}(Z_1)$

Bounds can be **updated** as abstractions are not stored

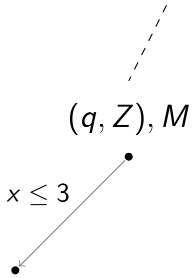
On-the-fly bounds propagation

$M(x) = -\infty$

$(q, Z), M$

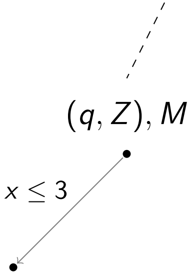
On-the-fly bounds propagation

$M(x) = -\infty$



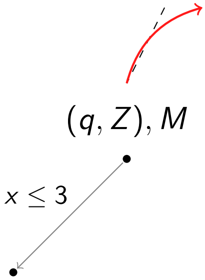
On-the-fly bounds propagation

$M(x) = \ 3$



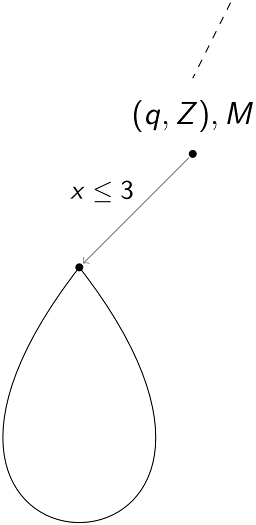
On-the-fly bounds propagation

$M(x) = 3$



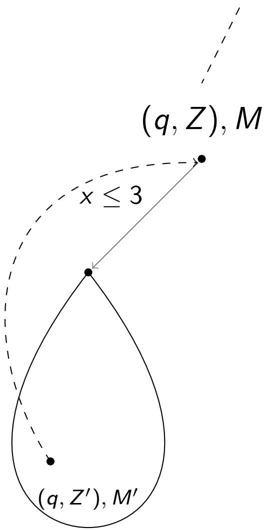
On-the-fly bounds propagation

$M(x) = \quad 5$



On-the-fly bounds propagation

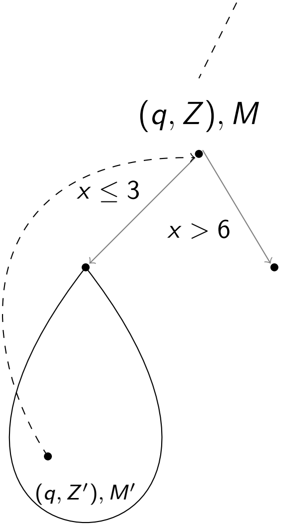
$M(x) = \quad 5$



$Z' \subseteq \mathfrak{a}_M(Z)$

On-the-fly bounds propagation

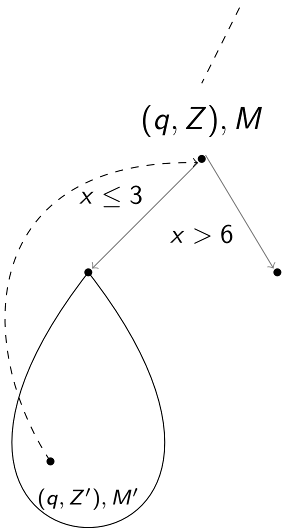
$M(x) = \quad 5$



$Z' \subseteq \mathfrak{a}_M(Z)$

On-the-fly bounds propagation

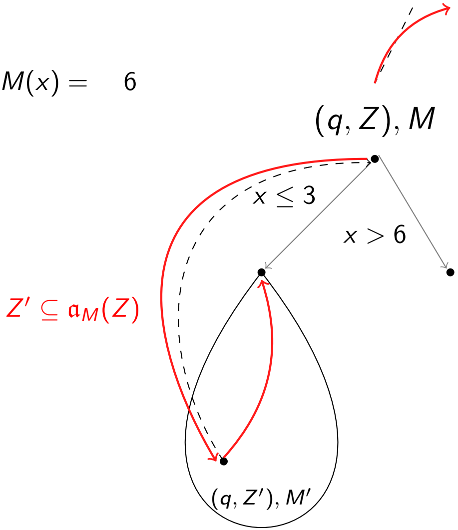
$M(x) = \quad 6$



$Z' \subseteq \mathfrak{a}_M(Z)$

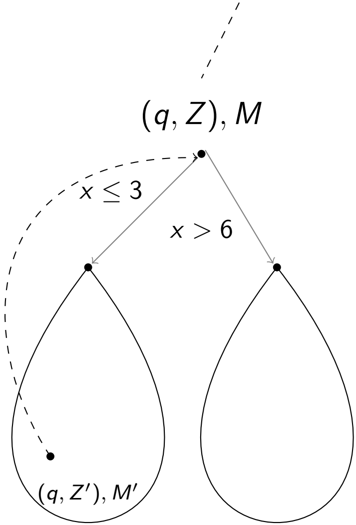
On-the-fly bounds propagation

$M(x) = \quad 6$



On-the-fly bounds propagation

$M(x) = \quad 6$

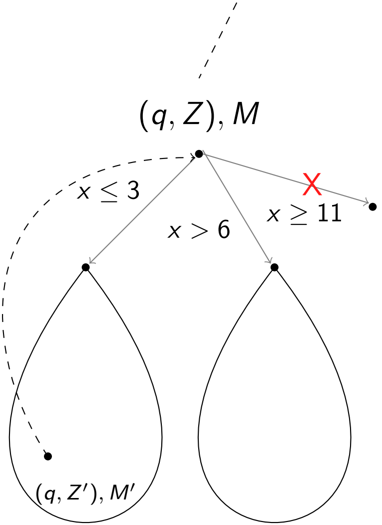


$Z' \subseteq \mathfrak{a}_M(Z)$

On-the-fly bounds propagation

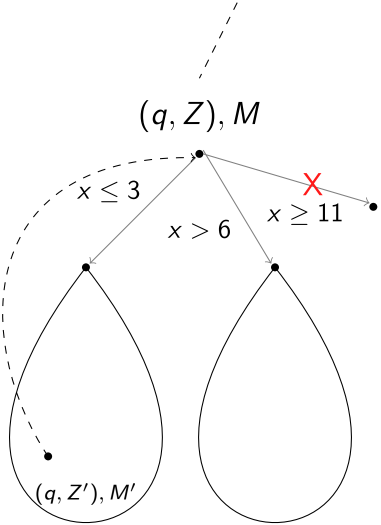
$M(x) = \quad 6$

$Z' \subseteq \mathfrak{a}_M(Z)$



On-the-fly bounds propagation

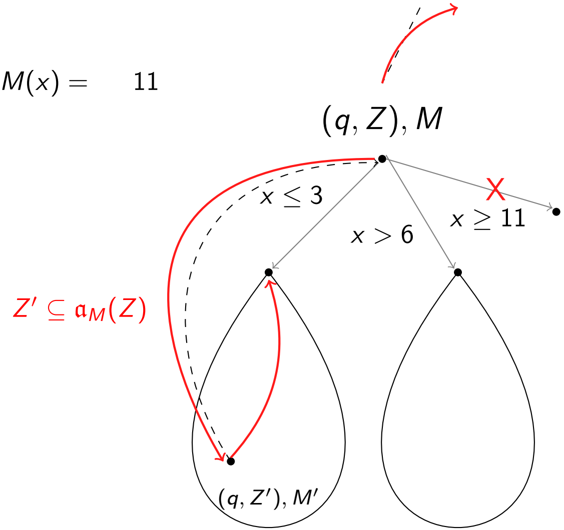
$M(x) = \quad 11$



$Z' \subseteq \mathfrak{a}_M(Z)$

On-the-fly bounds propagation

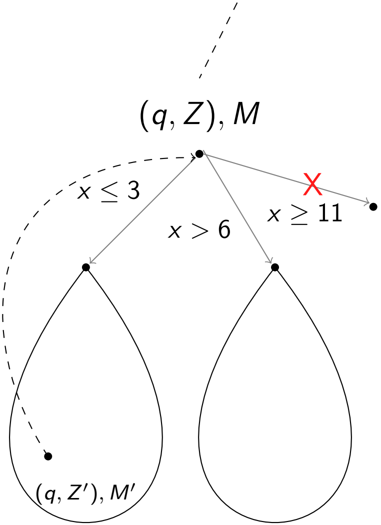
$M(x) = \quad 11$



On-the-fly bounds propagation

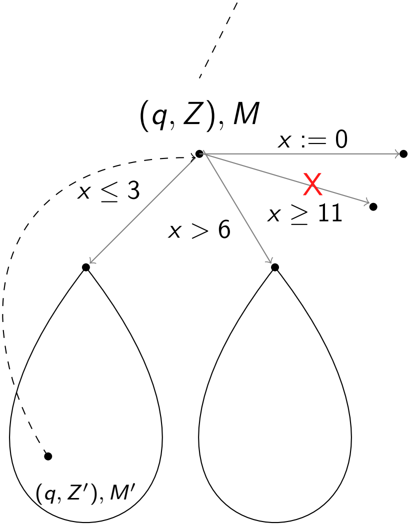
$M(x) = \quad 11$

$Z' \subseteq \mathfrak{a}_M(Z)$



On-the-fly bounds propagation

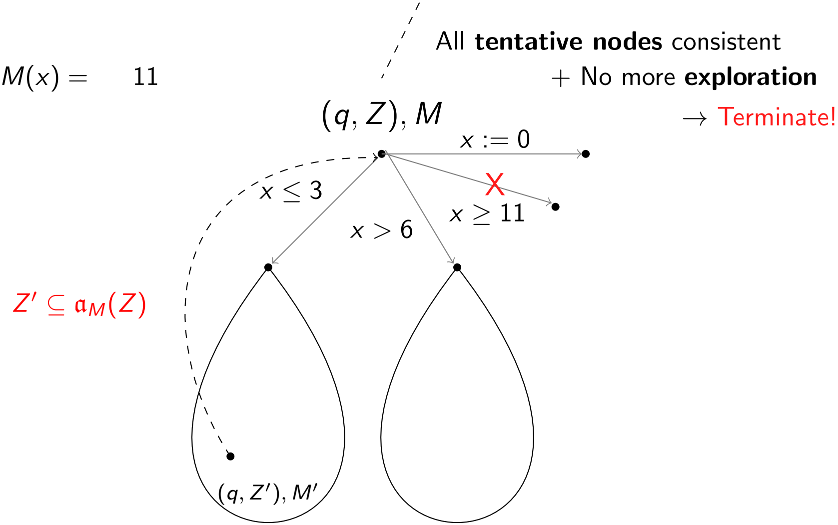
$M(x) = \quad 11$



$Z' \subseteq \mathfrak{a}_M(Z)$

On-the-fly bounds propagation

$M(x) = \quad 11$



On-the-fly bounds propagation (cont'd)

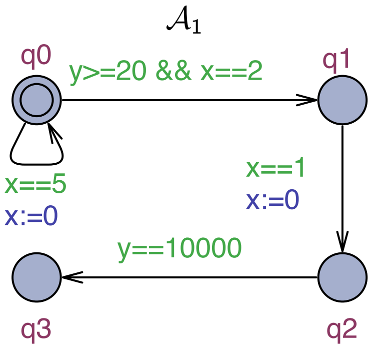
Theorem

The algorithm is correct and it terminates

- **Non tentative nodes:** $M = max\{M_{succ}\}$ (resets)
- **Tentative nodes:** $M = M_{covering}$
- M only increases and is bounded by [BBFL03]

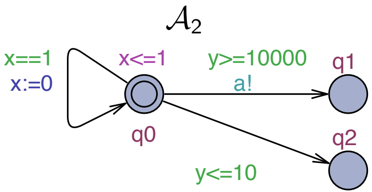
$(s, v) \sim_M (s, v')$ iff they enable the **same sequences of transitions**

Experiments I



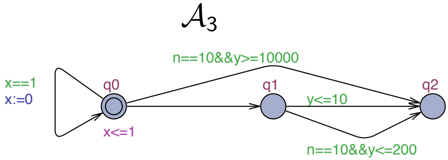
$\mathcal{A}_1$	nodes	s.
Extra^{+}_{LU}, **sa**	4001	6.16
$\mathfrak{a}_{LU}$, **otf**	9	0.00

Experiments II



$\mathcal{A}_2$	nodes	s.
Extra^{+}_{LU}, **sa**	10014	95.62
$\mathfrak{a}_{LU}$, **otf**	3	0.00

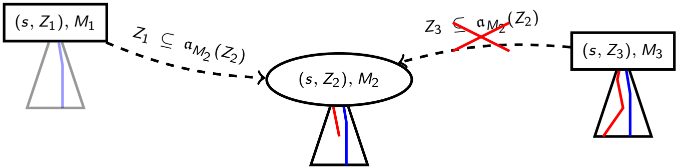
Experiments III



$\mathcal{A}_3$	nodes	s.
Extra^{+}_{LU}, **sa**	20006	99.26
$\mathfrak{a}_{LU}$, **otf**	4	0.00

But we can do even better...

Idea: only the **disabled edges** matter!



- Take bounds from the **disabled edges** only
- **Optimize** propagation

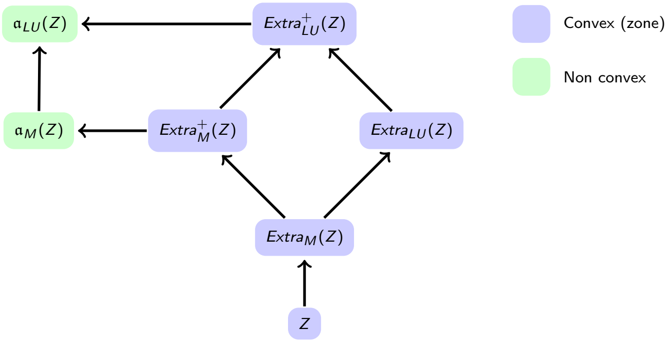
Outline

Benchmarks

Model	nb. of clocks	UPPAAL (-C) nodes	UPPAAL (-C) sec.	$Extra^{+}_{LU}$,sa nodes	$Extra^{+}_{LU}$,sa sec.	$\mathfrak{a}_{LU}$,otf nodes	$\mathfrak{a}_{LU}$,otf sec.	$\mathfrak{a}_{LU}$,dis. nodes	$\mathfrak{a}_{LU}$,dis. sec.
$Sched_7$	14	18654	11.6	18654	8.1	213	0.0	72	0.0
$Sched_8$	16					274	0.0	90	0.0
$Sched_{70}$	140							5112	1.9
CSMA/CD 10	11	120845	1.9	120844	6.3	78604	6.1	51210	4.0
CSMA/CD 11	12	311310	5.4	311309	16.8	198669	16.1	123915	10.2
CSMA/CD 12	13	786447	14.8	786446	44.0	493582	41.8	294924	25.2
FDDI 50	151	12605	52.9	12606	29.4	5448	14.7	401	0.8
FDDI 70	211							561	2.7
FDDI 140	421							1121	40.4
Fischer 9	9	135485	2.4	135485	8.9	135485	11.4	135485	14.8
Fischer 10	10	447598	10.1	447598	34.0	447598	42.8	447598	56.9
Fischer 11	11	1464971	40.4	1464971	126.8				
Stari 2	7	7870	0.1	6993	0.4	5779	0.4	4202	0.3
Stari 3	10	136632	1.7	113958	9.4	82182	8.2	29964	2.9
Stari 4	13	1323193	26.2	983593	109.0	602762	84.9	278120	37.6

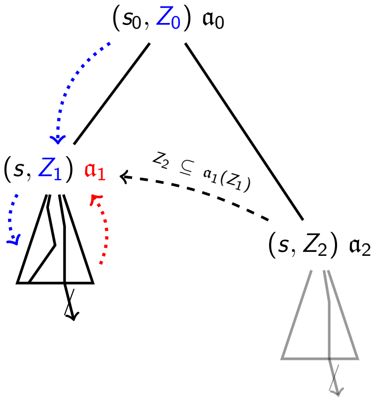
Both **non-convex abstractions** $\mathfrak{a}_M/\mathfrak{a}_{LU}$ and **on-the-fly bounds** computation help

Conclusions



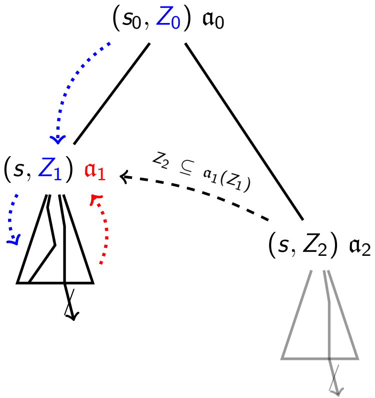
- New reachability algorithm with **non-convex abstractions** and **on-the-fly computation** of bounds
- **Optimal abstraction** when only bounds are considered
- **Tightening** of bounds

Future Work



Z_1 defines the subtree, which in turn defines $\mathfrak{a}_1$

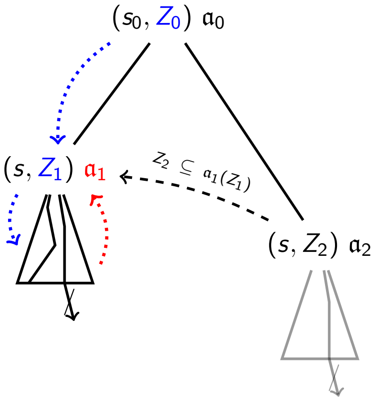
Future Work



Z_1 defines the subtree, which in turn defines $\mathfrak{a}_1$

- **Optimal** bounds: better propagation

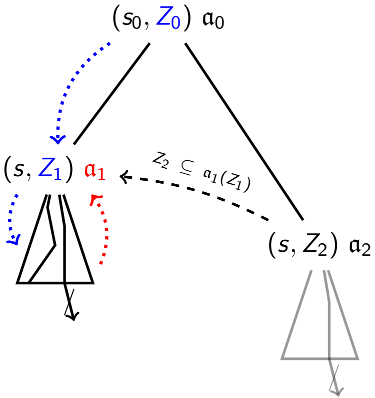
Future Work



Z_1 defines the subtree, which in turn defines $\mathfrak{a}_1$

- **Optimal** bounds: better propagation
- **Beyond bounds**: define $\mathfrak{a}$ from constraints, ...

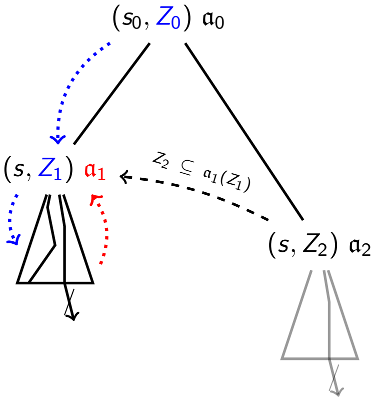
Future Work



Z_1 defines the subtree, which in turn defines $\mathfrak{a}_1$

- **Optimal** bounds: better propagation
- **Beyond bounds**: define $\mathfrak{a}$ from constraints, ...
- Extend to **infinite runs** (beyond reachability)

Future Work



Z_1 defines the subtree, which in turn defines $\mathfrak{a}_1$

- **Optimal** bounds: better propagation
- **Beyond bounds**: define $\mathfrak{a}$ from constraints, ...
- Extend to **infinite runs** (beyond reachability)
- Prototype **tool**

References

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.

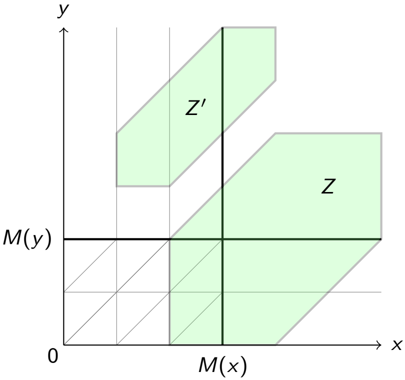
G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelanek.
Lower and upper bounds in zone-based abstractions of timed automata.
Int. Journal on Software Tools for Technology Transfer, 8(3):204–215, 2006.

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.

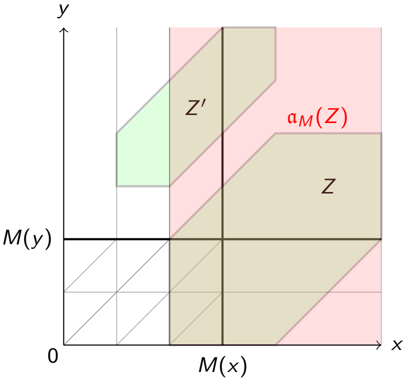
François Laroussinie and Philippe Schnoebelen.
The state explosion problem from trace to bisimulation equivalence.
In Jerzy Tiuryn, editor, *FoSSaCS*, volume 1784 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2000.

When is $Z' \subseteq \mathfrak{a}_M(Z)$?



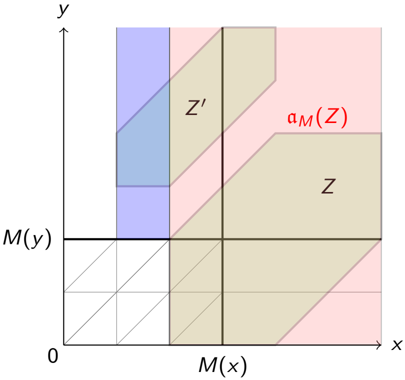
Recall: $\mathfrak{a}_M(Z)$ is the union of **regions that intersect** Z

When is $Z' \subseteq \mathfrak{a}_M(Z)$?



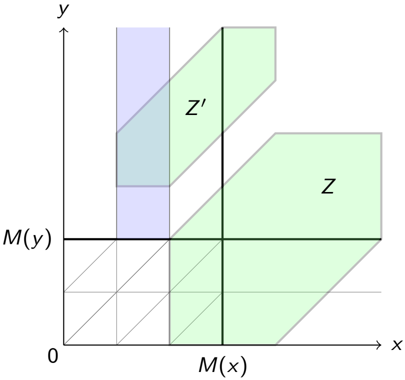
Recall: $\mathfrak{a}_M(Z)$ is the union of **regions that intersect** Z

When is $Z' \subseteq \mathfrak{a}_M(Z)$?



Recall: $\mathfrak{a}_M(Z)$ is the union of **regions that intersect** Z

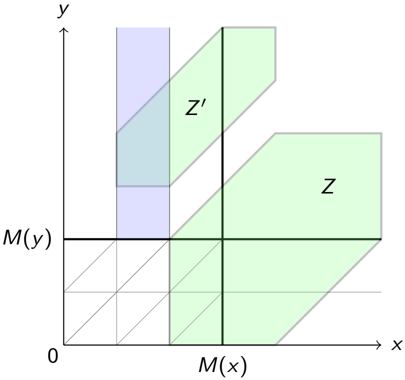
When is $Z' \subseteq \mathfrak{a}_M(Z)$?



Recall: $\mathfrak{a}_M(Z)$ is the union of **regions that intersect** Z

$Z' \not\subseteq \mathfrak{a}_M(Z)$ iff $\exists R.\ R$ **intersects Z'**, but R does **not intersect Z**

When is $Z' \subseteq \mathfrak{a}_M(Z)$?



Recall: $\mathfrak{a}_M(Z)$ is the union of **regions that intersect** Z

$Z' \not\subseteq \mathfrak{a}_M(Z)$ iff $\exists R.$ R **intersects** $\mathbf{Z}'$, but R does **not intersect** $\mathbf{Z}$

Theorem

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

$$\mathbf{Proj_{xy}}(Z') \not\subseteq \mathfrak{a}_M(\mathbf{Proj_{xy}}(Z))$$