Lazy abstractions for timed automata

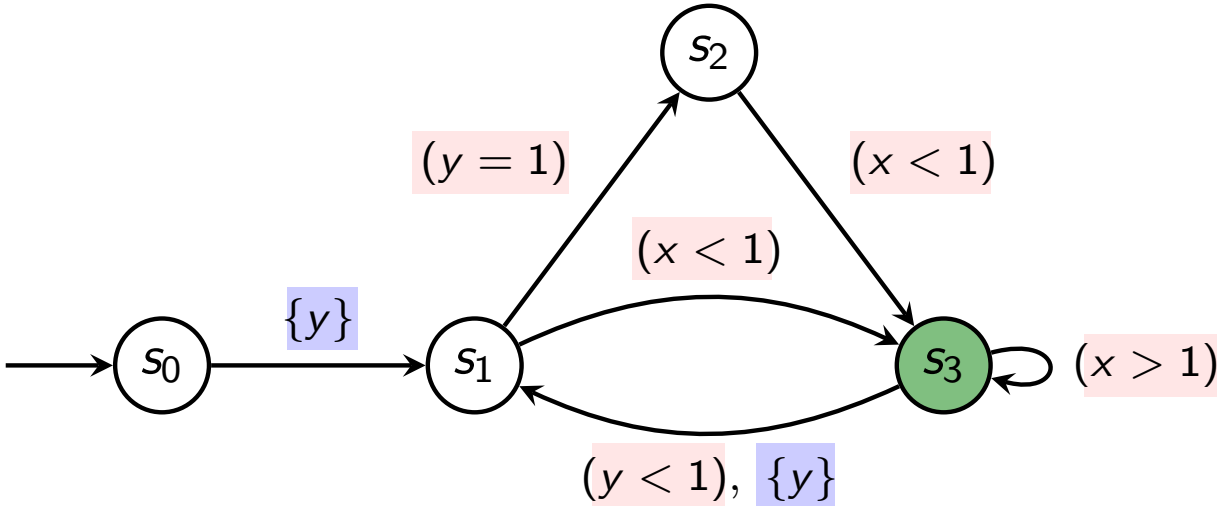
F. Herbreteau[1], B. Srivathsan[2], I. Walukiewicz[1]

LaBRI, Université Bordeaux 1

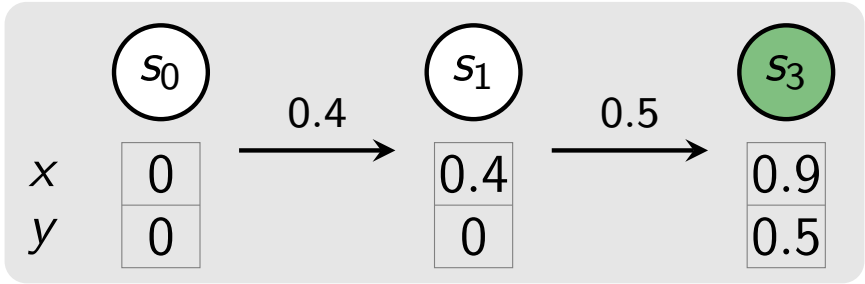
Software modeling and verification group, RWTH-Aachen

MOVES Seminar, March 2013

Timed Automata [AD94]



Run: finite sequence of transitions



- **accepting** if ends in 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]

This problem is **PSPACE-complete**

first solution based on Regions

Regions

Maximal bounds: $M : X \mapsto \mathbb{N} \cup \{-\infty\}$

Regions

Maximal bounds: $M : X \mapsto \mathbb{N} \cup \{-\infty\}$

$$M(x) = 3,\ M(y) = 2$$

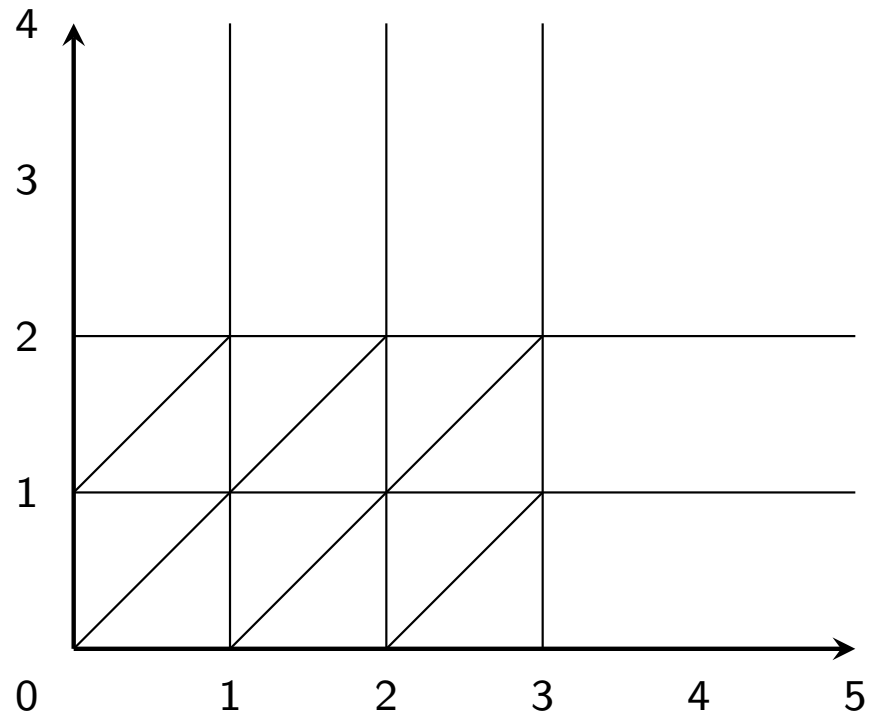
$$x \lessdot 0,\ x \gtrdot 0,\ x \lessdot 1,\ x \gtrdot 1,\ \ldots,\ x \lessdot 3,\ x \gtrdot 3$$
$$y \lessdot 0,\ y \gtrdot 0,\ \ldots,\ y \lessdot 2,\ y \gtrdot 2$$

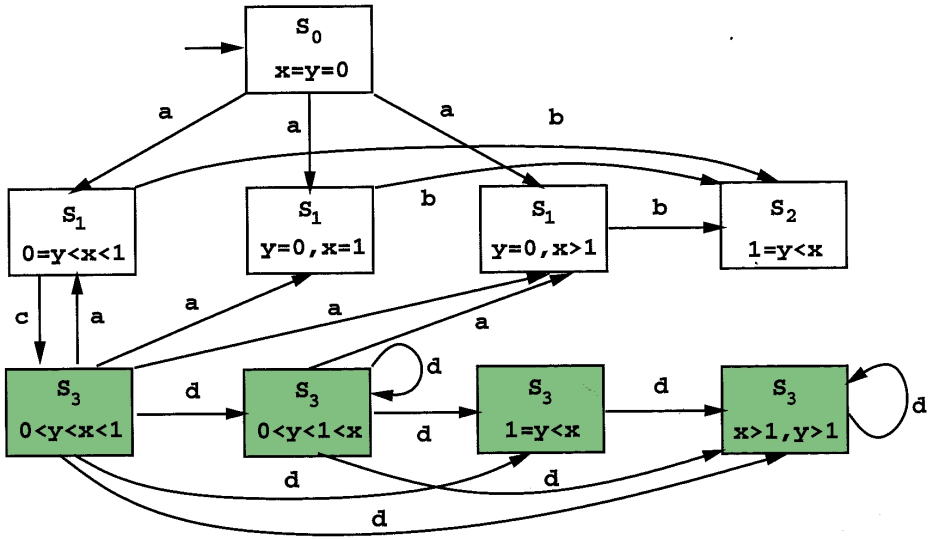
Regions

Maximal bounds: $M : X \mapsto \mathbb{N} \cup \{-\infty\}$

$$M(x) = 3,\ M(y) = 2$$

$$x \lessdot 0,\ x \gtrdot 0,\ x \lessdot 1,\ x \gtrdot 1,\ \ldots,\ x \lessdot 3,\ x \gtrdot 3$$
$$y \lessdot 0,\ y \gtrdot 0,\ \ldots,\ y \lessdot 2,\ y \gtrdot 2$$

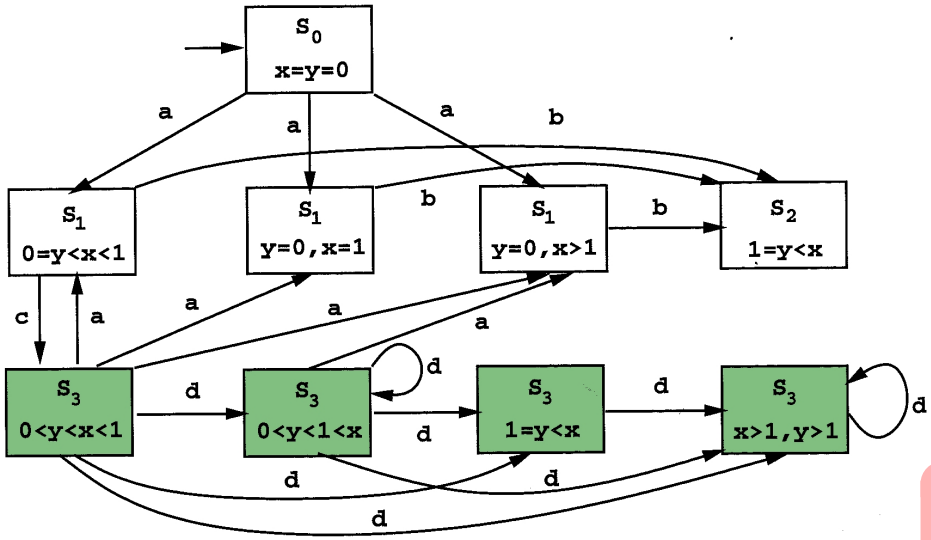




- Region: set of valuations satisfying the **same guards** w.r.t. time
- Finiteness: Parametrized by **maximal constant**

Sound and complete [AD94]

Region graph preserves state **reachability**



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

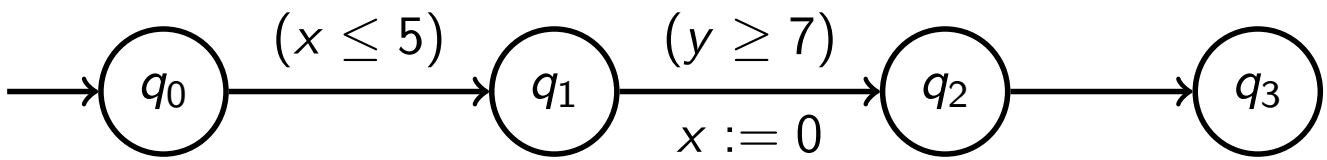
$\mathcal{O}(|X|!.M^{|X|})$ many regions!

Sound and complete [AD94]

Region graph preserves state **reachability**

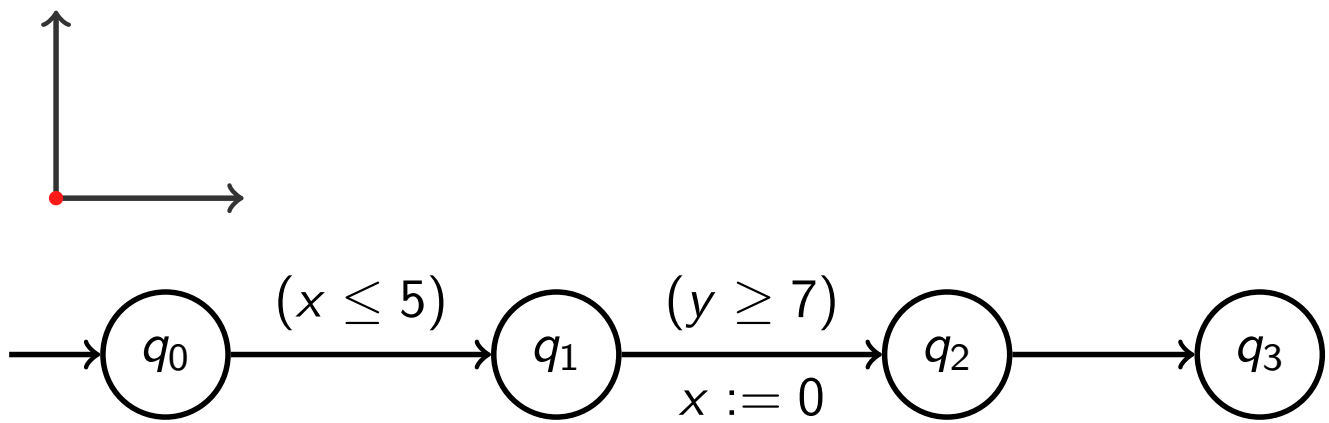
A more efficient solution...

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



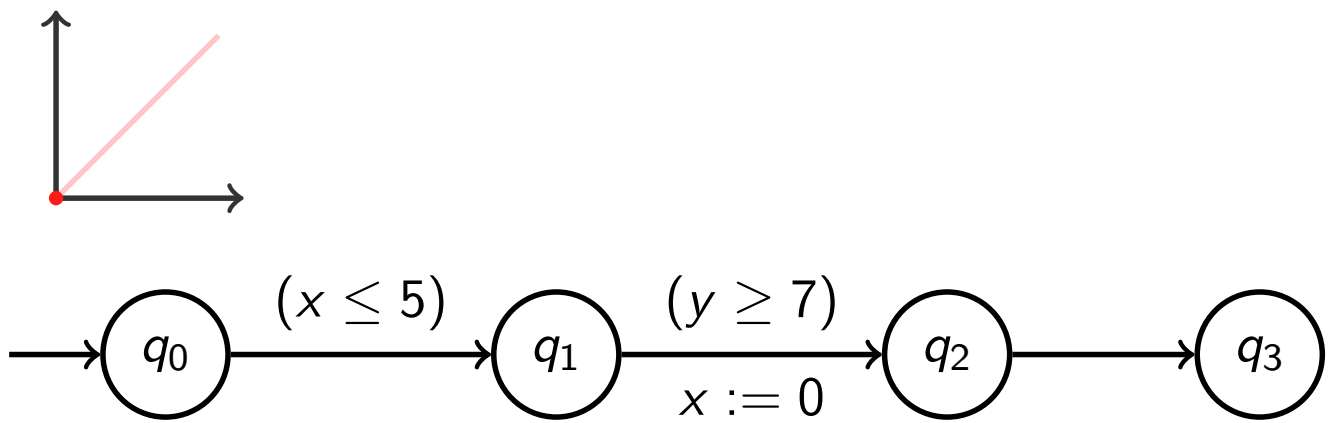
A more efficient solution...

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



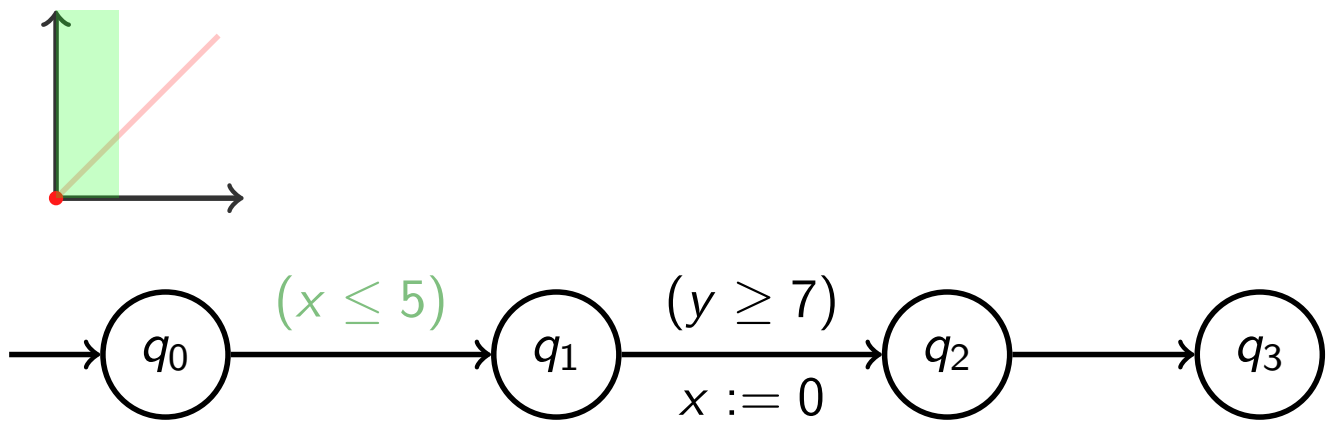
A more efficient solution...

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



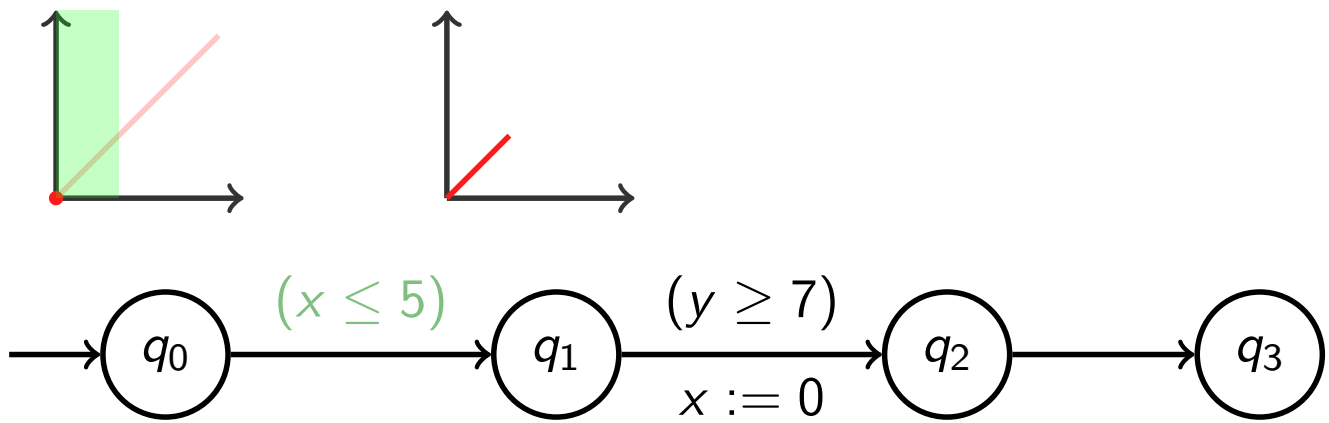
A more efficient solution...

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



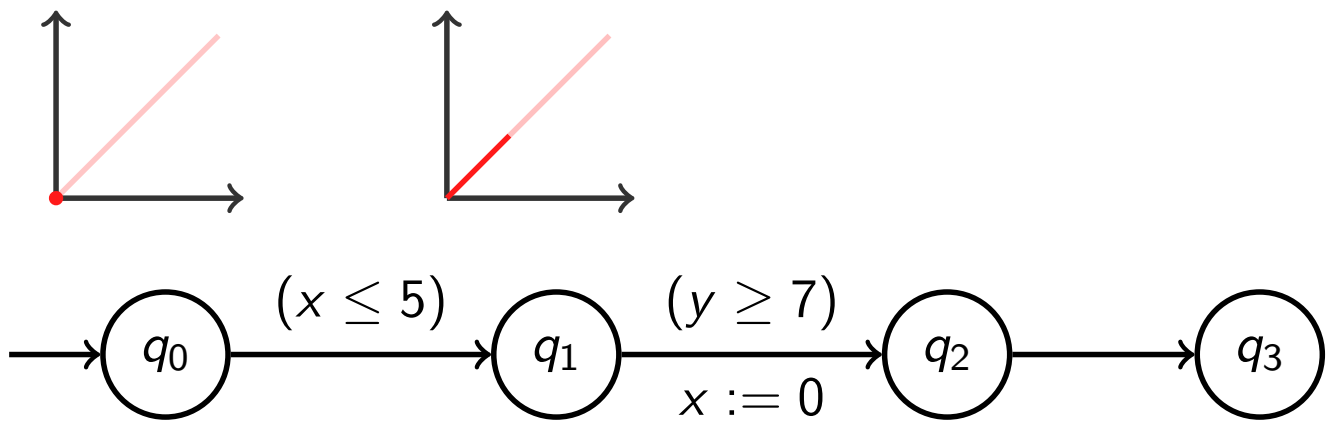
A more efficient solution...

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



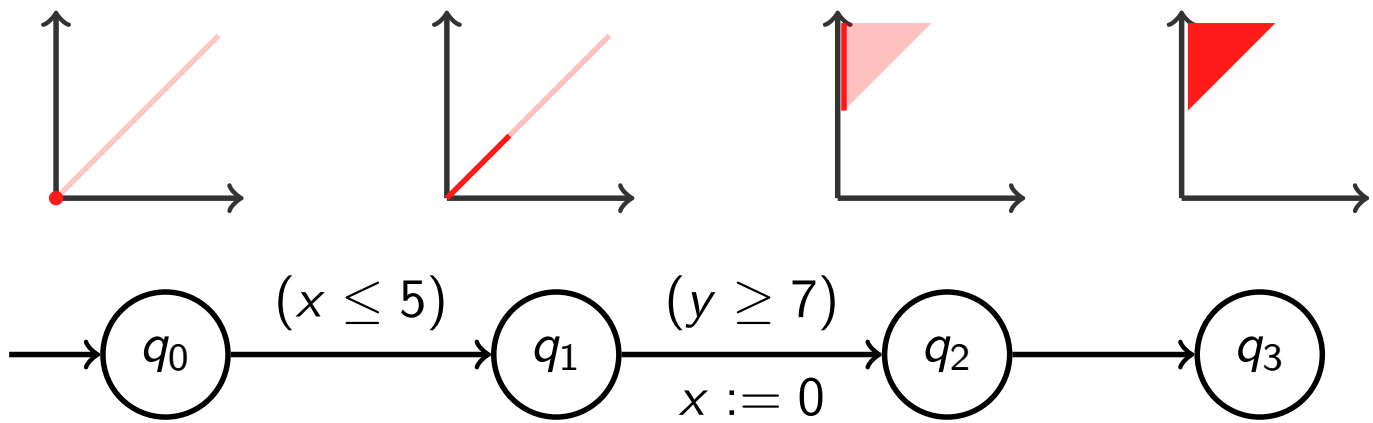
A more efficient solution...

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



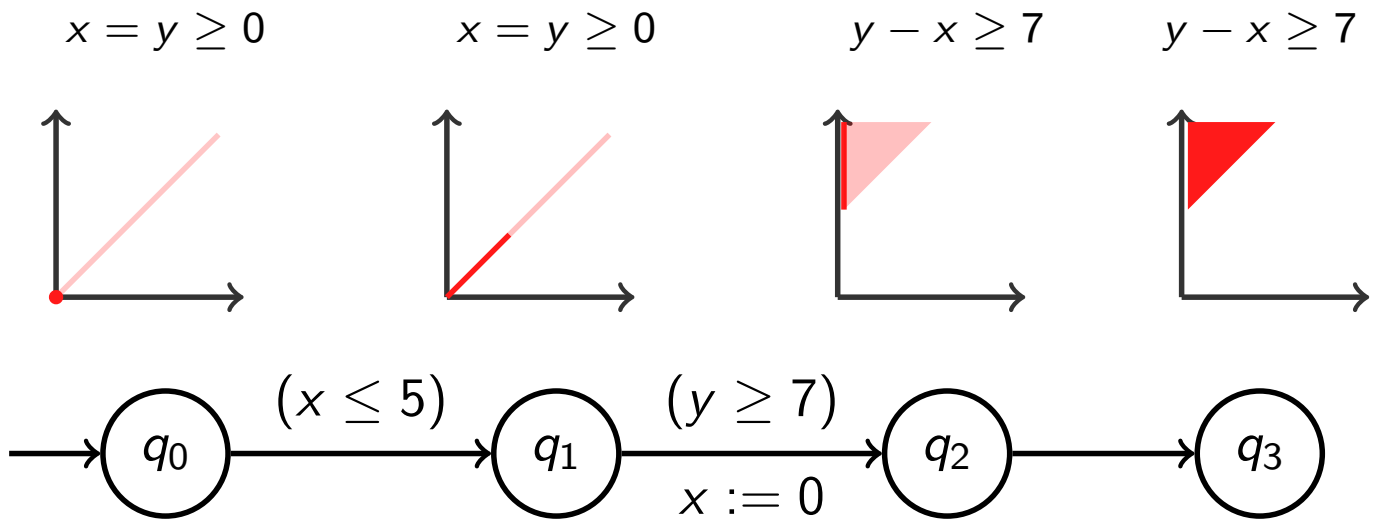
A more efficient solution...

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

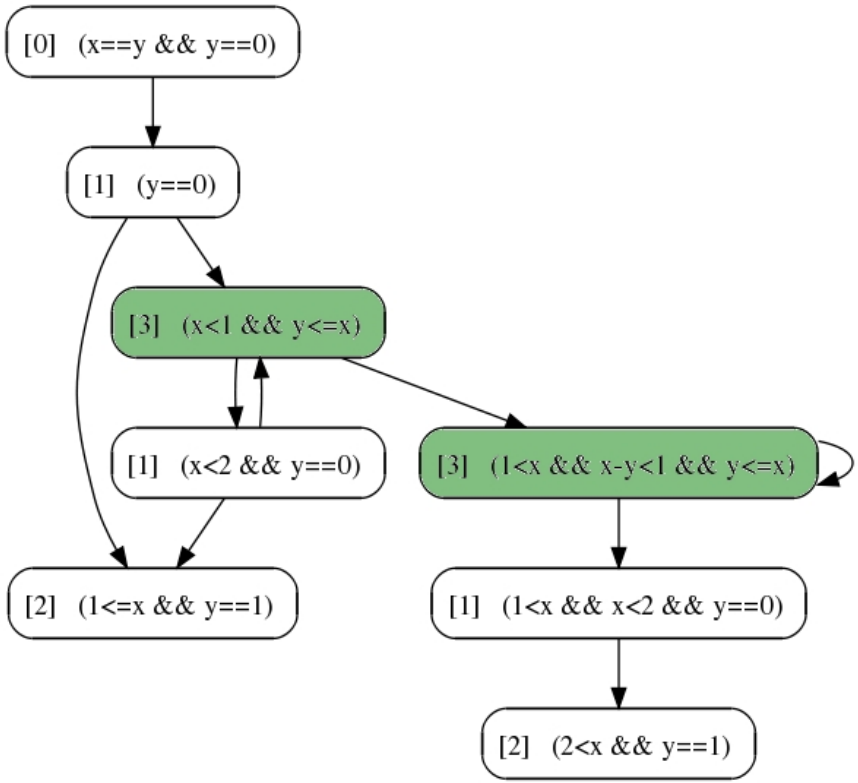


A more efficient solution...

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



Zones and zone graph

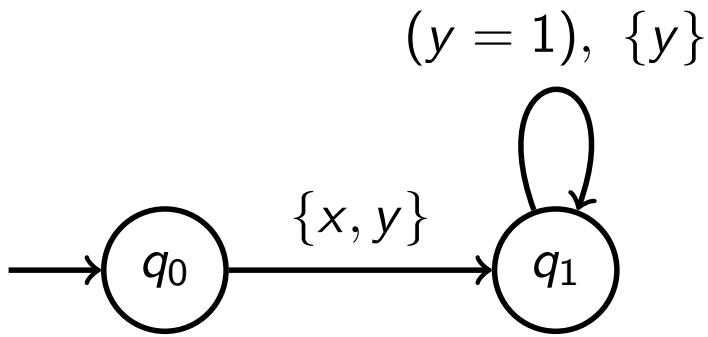


- Zone: set of valuations defined by conjunctions of constraints:
  - $x \sim c$
  - $x - y \sim c$
  - e.g. $(x - y \geq 1) \wedge y < 2$
- Representation: by DBM

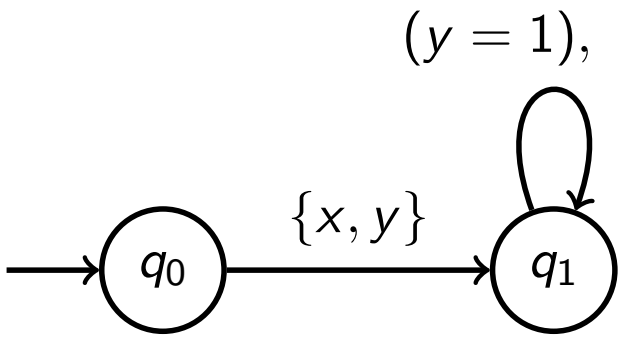
Sound and complete [DT98]

Zone graph preserves state **reachability**

But the zone graph could be infinite ...

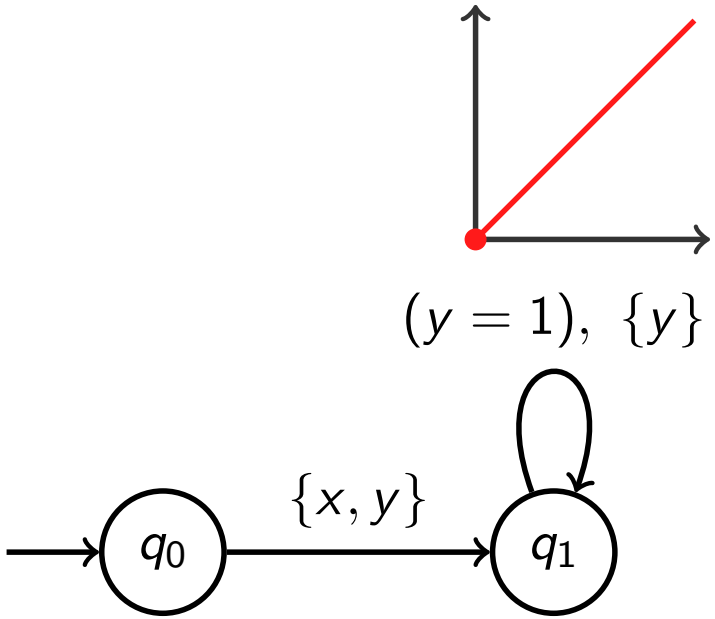


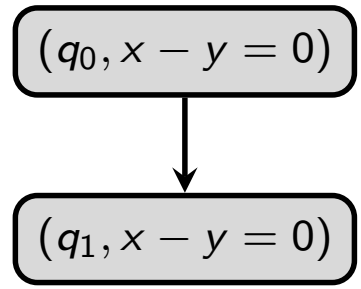
But the zone graph could be infinite ...



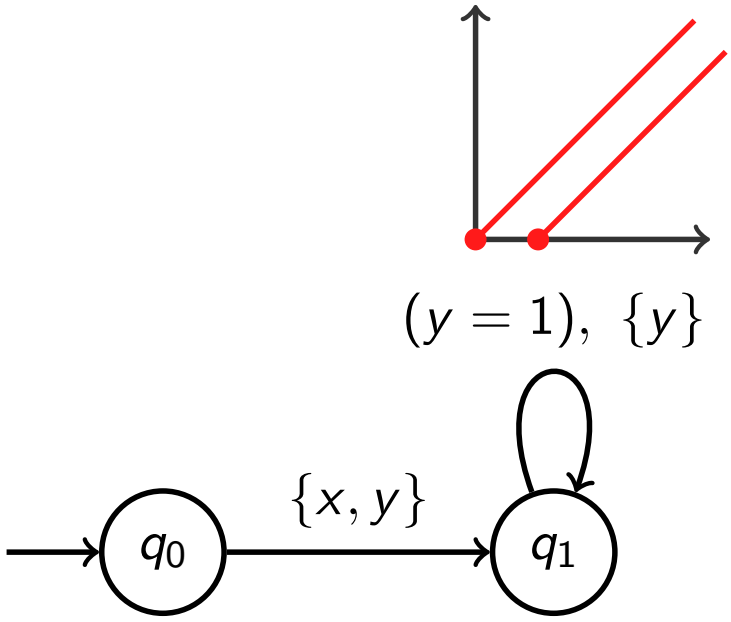
$(q_0, x - y = 0)$

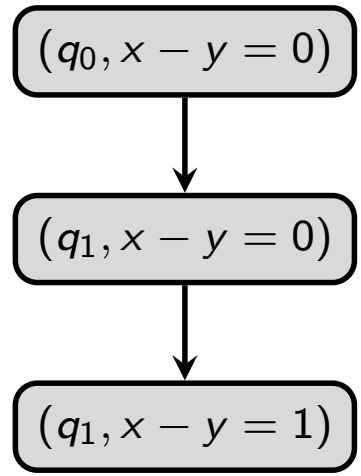
But the zone graph could be infinite ...



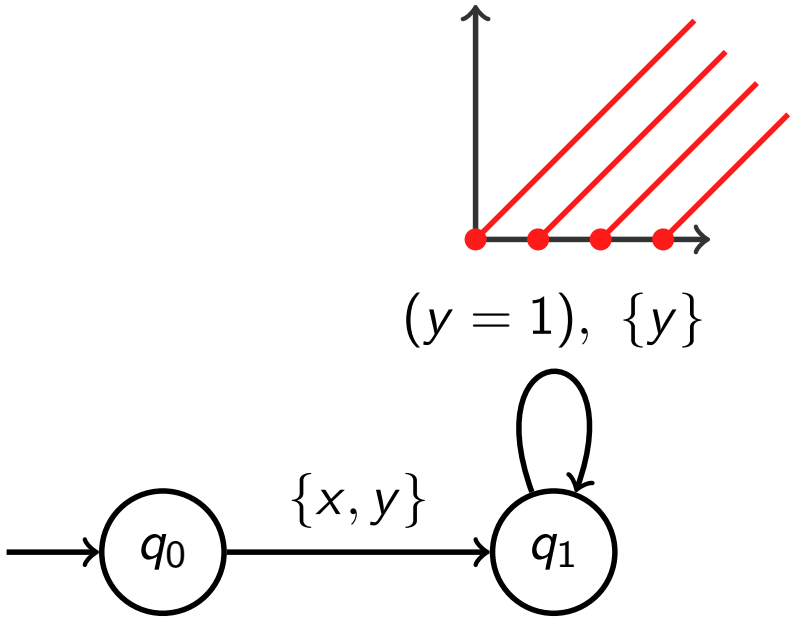


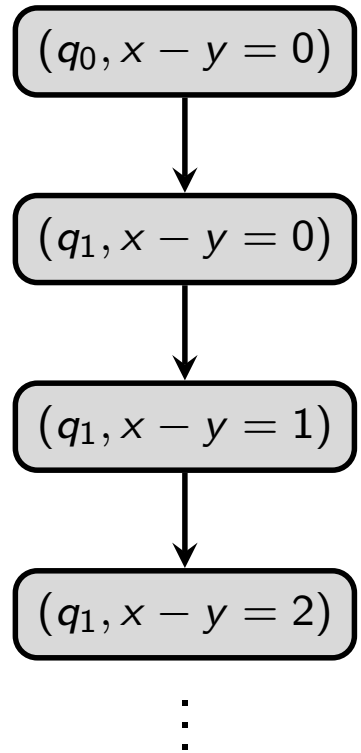
But the zone graph could be infinite ...



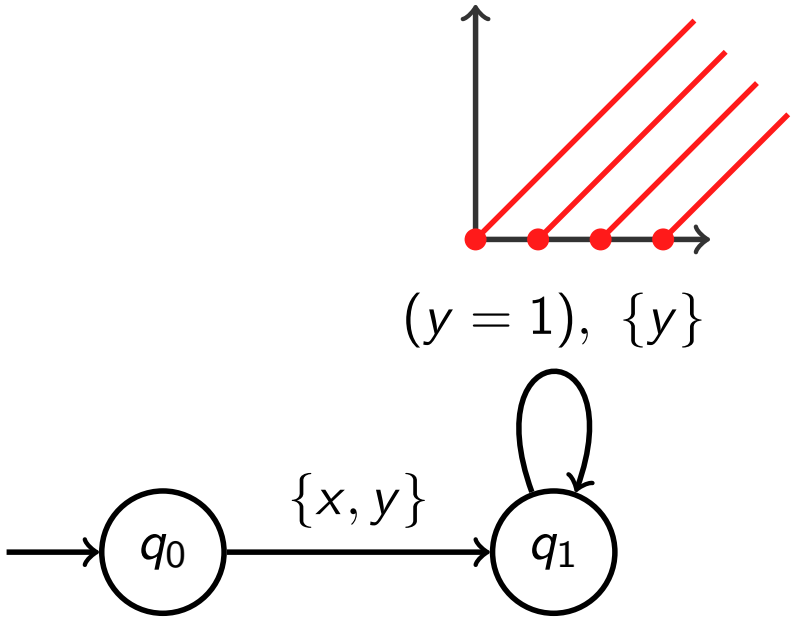


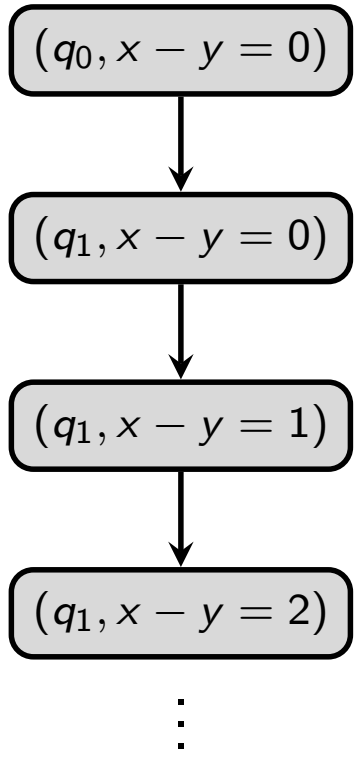
But the zone graph could be infinite ...





But the zone graph could be infinite ...

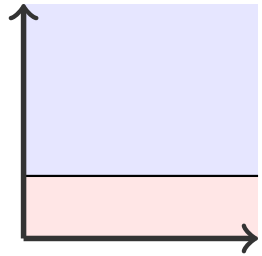




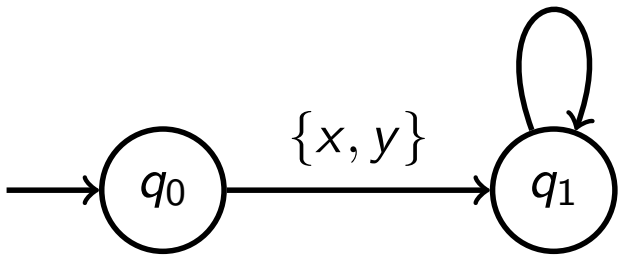
Abstract zone to its **region closure**

$M(x) = -\infty$

$M(y) = 1$

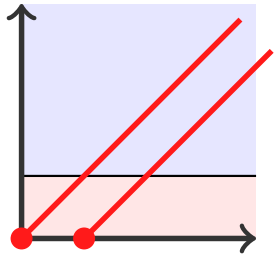


$(y = 1),\ \{y\}$

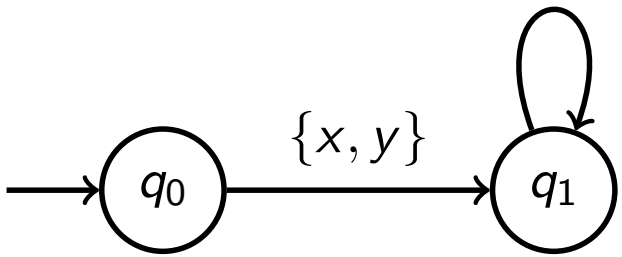


$M(x) = -\infty$

$M(y) = 1$

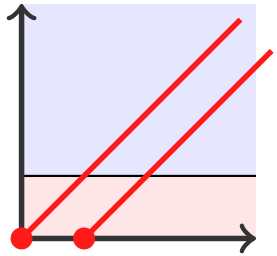


$(y = 1),\ \{y\}$

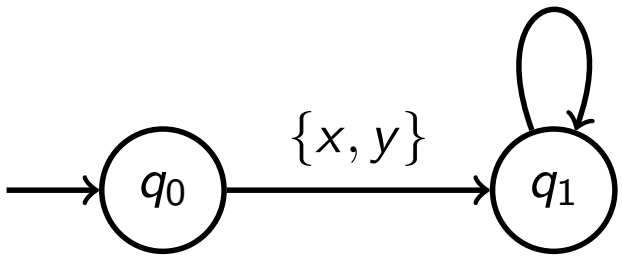


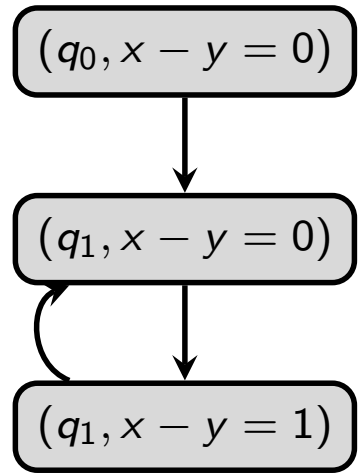
$M(x) = -\infty$

$M(y) = 1$



$(y = 1),\ \{y\}$

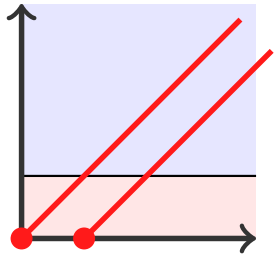




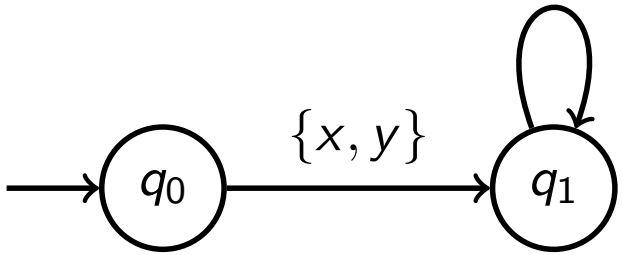
$$x - y = 1 \ \subseteq\ \mathsf{Closure}_M(x - y = 0)$$

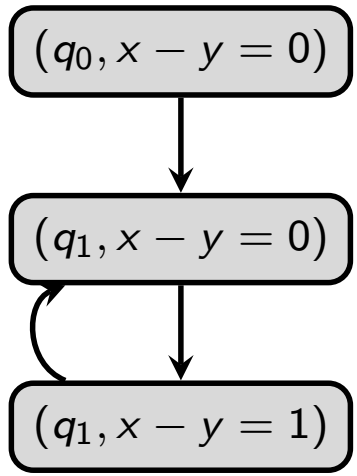
$M(x) = -\infty$

$M(y) = 1$



$(y = 1),\ \{y\}$





$$x - y = 1 \ \subseteq\ \text{Closure}_M(x - y = 0)$$

Using Closure

1. $Z \subseteq \text{Closure}_M(Z')$ can be done **efficiently** [HKSW11]
2. Given M, Closure_M is **optimal** [HSW12]

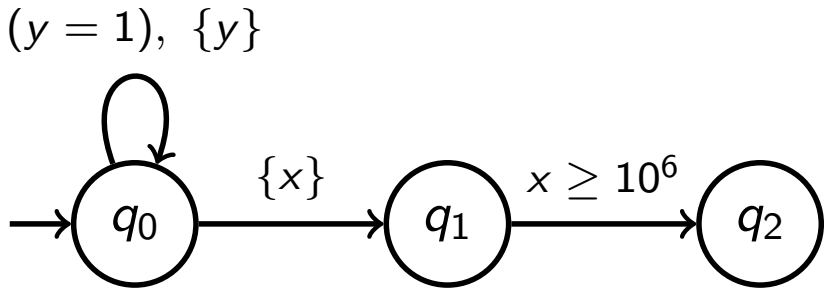
Reachability algorithm:

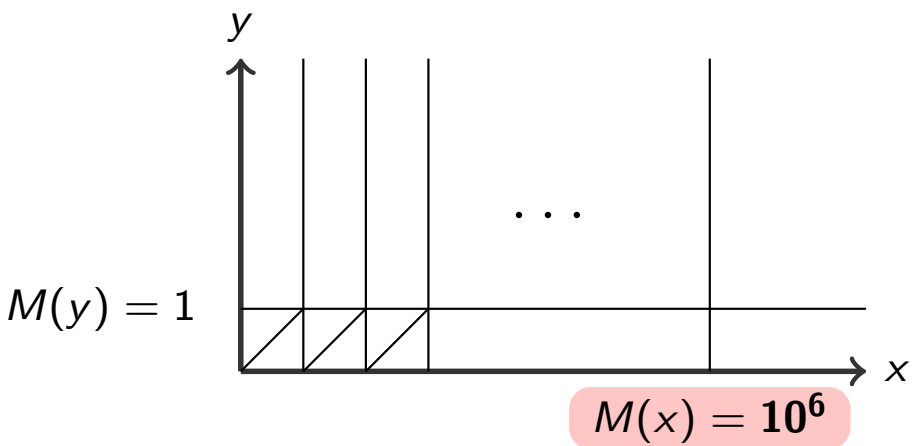
- Compute zones
- Use Closure$_M$ for termination
- Given M, Closure$_M$ is optimal

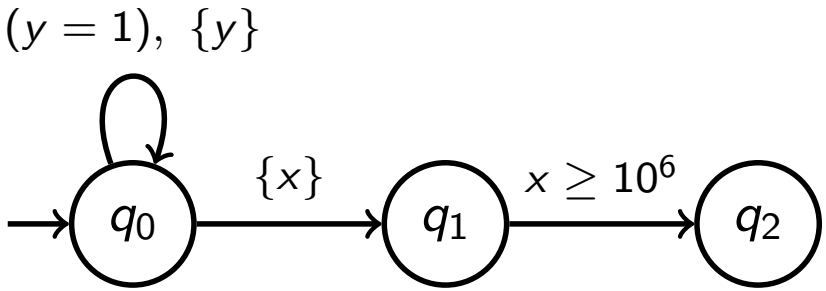
Reachability algorithm:

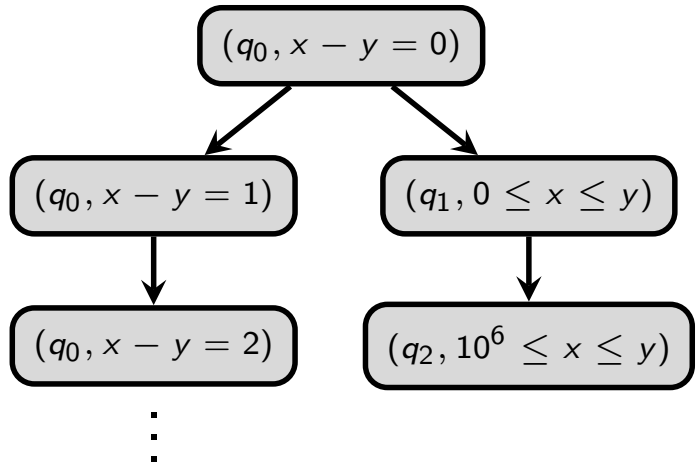
- Compute zones
- Use Closure_M for termination
- Given M, Closure_M is optimal

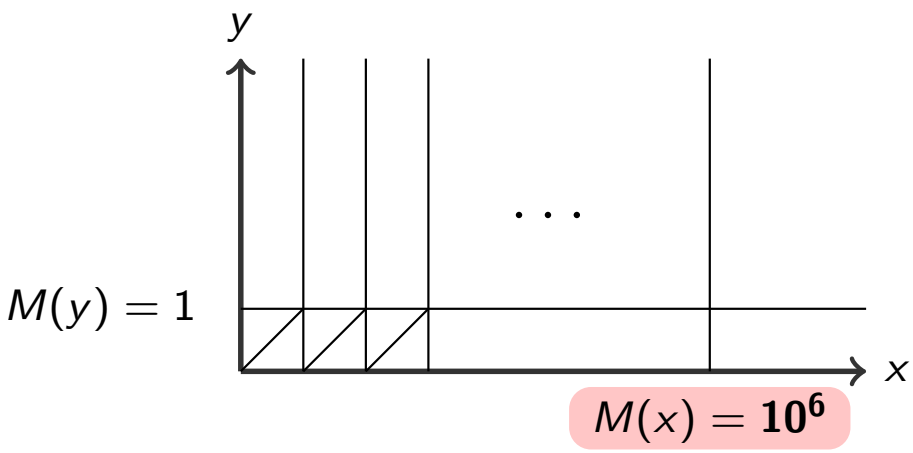
Coming next: get **better** M bounds!

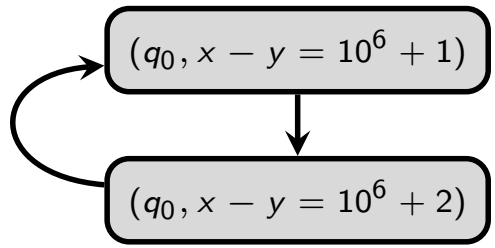

$(y = 1),\ \{y\}$
$\{x\}$
$x \geq 10^6$
q_0
q_1
q_2


y
$M(y) = 1$
x
$M(x) = \mathbf{10^6}$





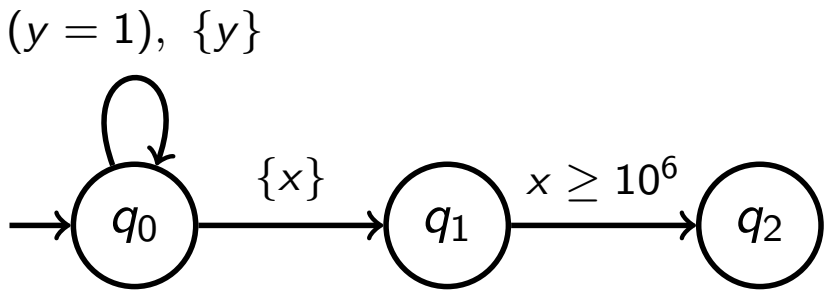




More than 10^6 nodes unnecessary

Static analysis [BBFL03]

Key idea: Bounds for every q of the automaton

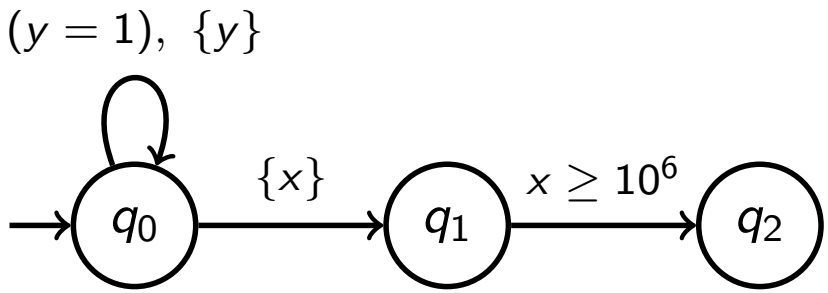


$M_0(x) = -\infty$ $\quad M_1(x) = 10^6$

$M_0(y) = 1$ $\quad M_1(y) = -\infty$

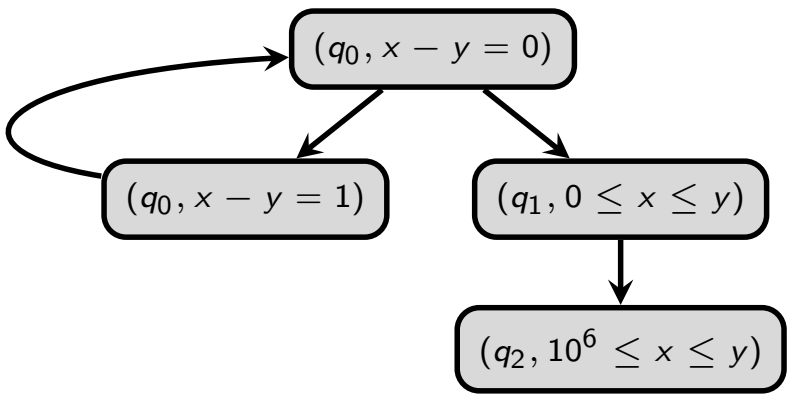
Static analysis [BBFL03]

Key idea: Bounds for every q of the automaton

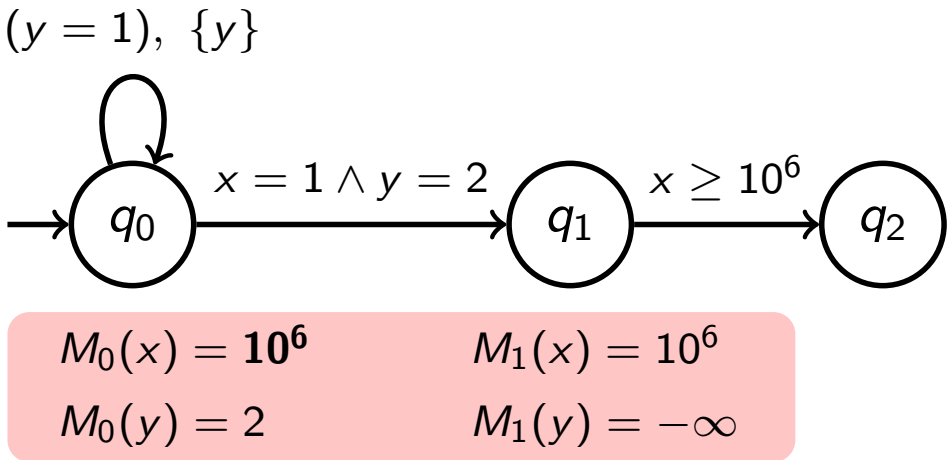


$M_0(x) = -\infty \quad M_1(x) = 10^6$

$M_0(y) = 1 \quad M_1(y) = -\infty$

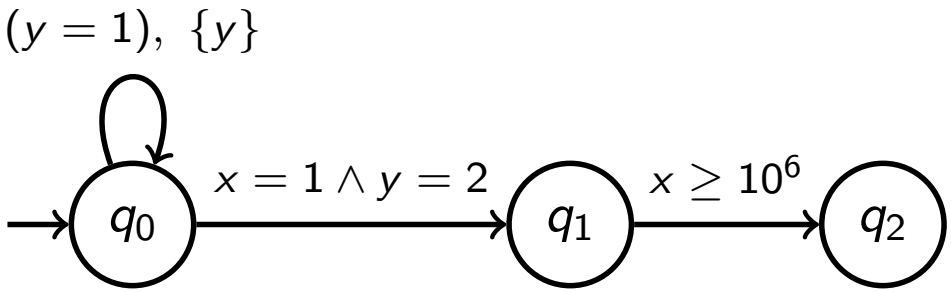


However...



Static analysis gives **more than** 10^6 nodes in the zone graph

However...



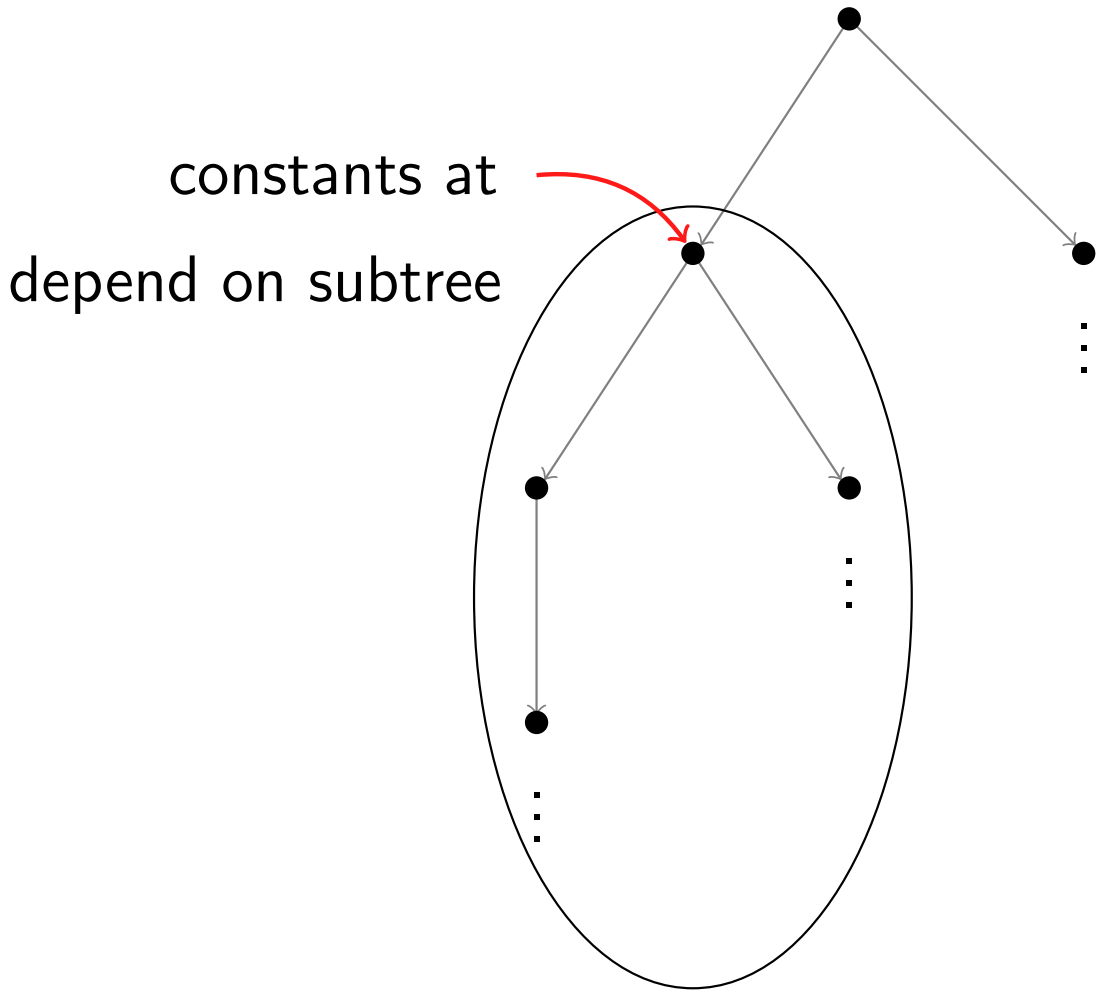
$M_0(x) = \mathbf{10^6}$	$M_1(x) = 10^6$
$M_0(y) = 2$	$M_1(y) = -\infty$

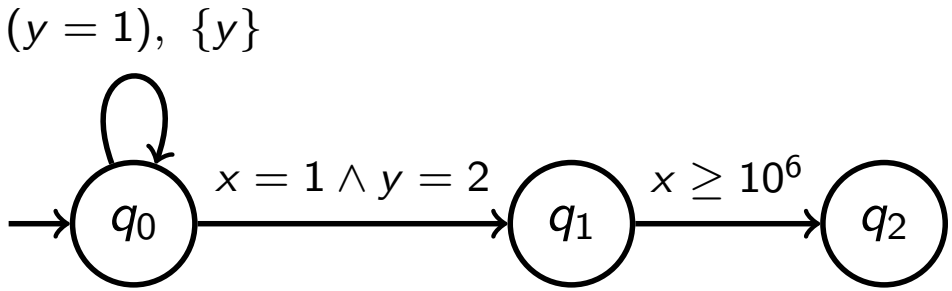
Static analysis gives **more than** 10^6 nodes in the zone graph

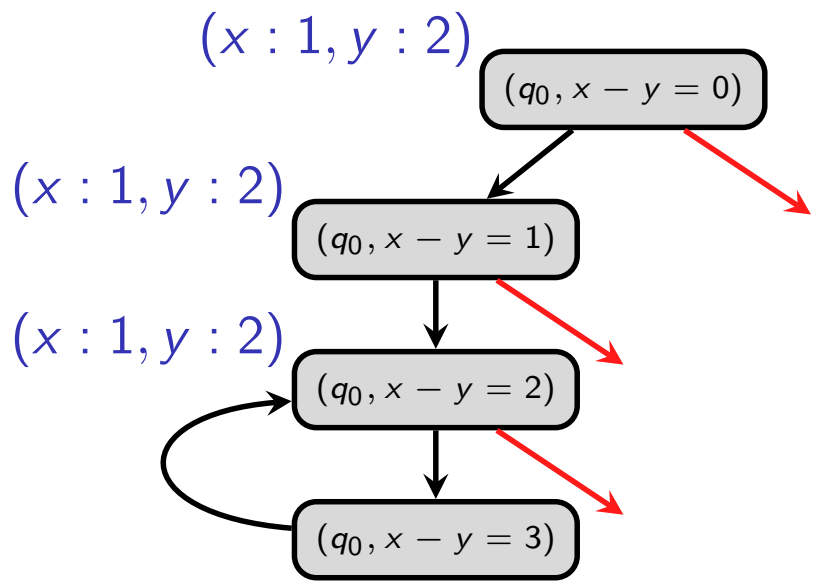
Need to look at **semantics**

On-the-fly bounds [HKSW11]

Key idea: Bounds for every (q, Z) of the zone graph




$(y = 1),\ \{y\}$
q_0
$x = 1 \wedge y = 2$
q_1
$x \geq 10^6$
q_2


$(x : 1, y : 2)$
$(q_0, x - y = 0)$
$(x : 1, y : 2)$
$(q_0, x - y = 1)$
$(x : 1, y : 2)$
$(q_0, x - y = 2)$
$(q_0, x - y = 3)$

Two ways of getting bounds:

- **Static** analysis (bounds for every q)
- **On-the-fly** propagation (bounds for every (q, Z))

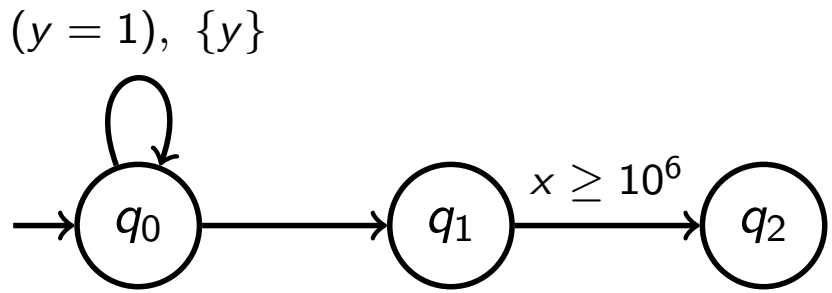
Two ways of getting bounds:

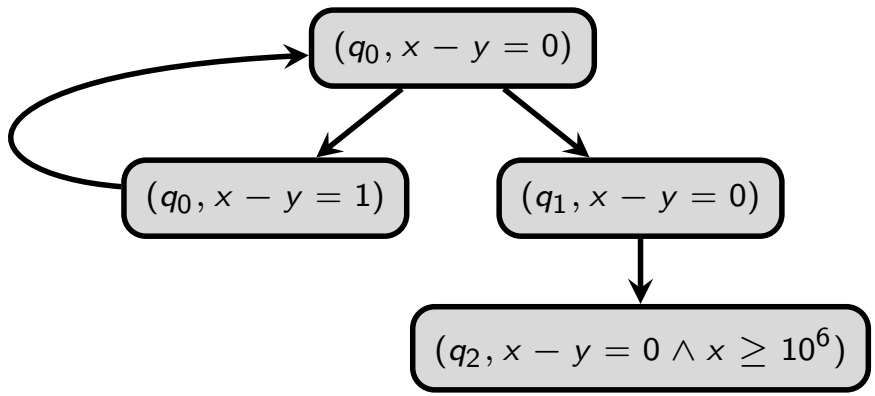
- **Static** analysis (bounds for every q)
- **On-the-fly** propagation (bounds for every (q, Z))

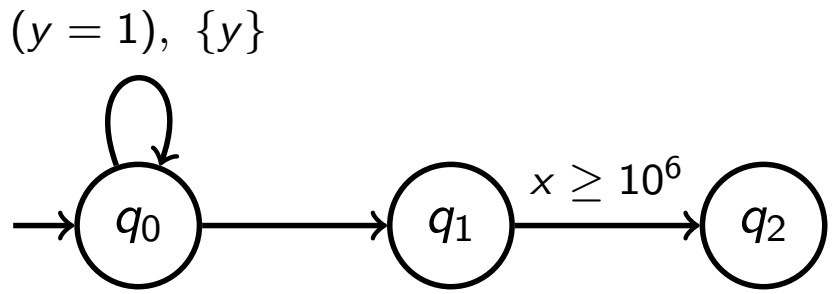
Coming next: Better bounds by exploiting **more semantics**

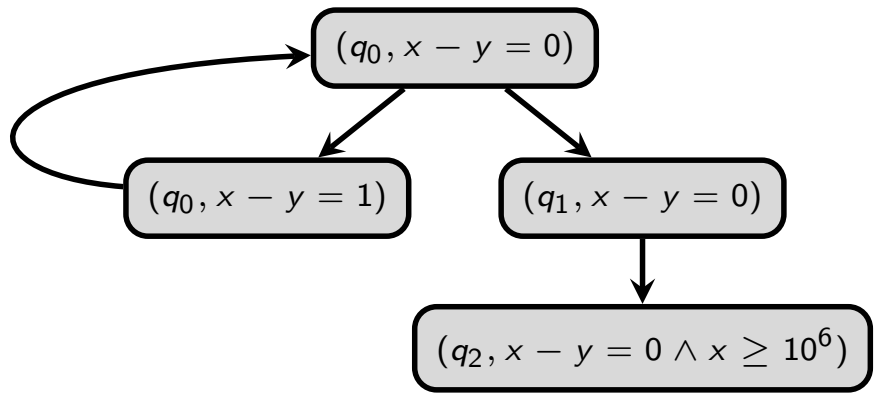
Observation 1:

If **all edges are enabled** in the zone graph, then we **don't need** bounds at all

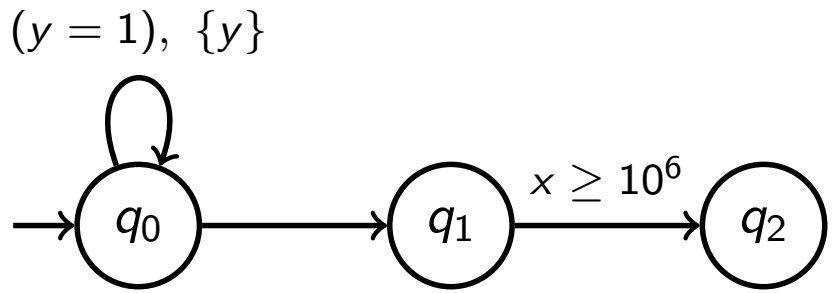

$(y = 1),\ \{y\}$
q_0
q_1
$x \geq 10^6$
q_2

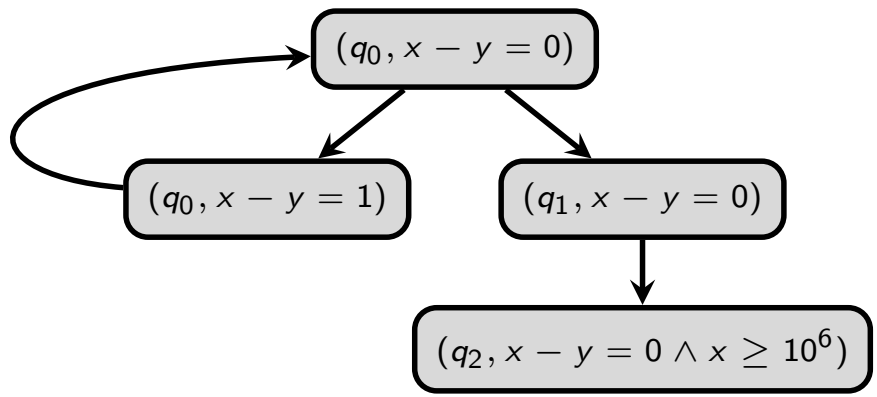

$(q_0, x - y = 0)$
$(q_0, x - y = 1)$
$(q_1, x - y = 0)$
$(q_2, x - y = 0 \wedge x \geq 10^6)$





Otf-propagation would have given $\sim 10^6$ nodes



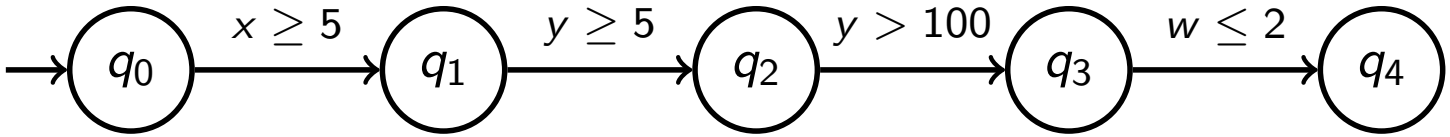


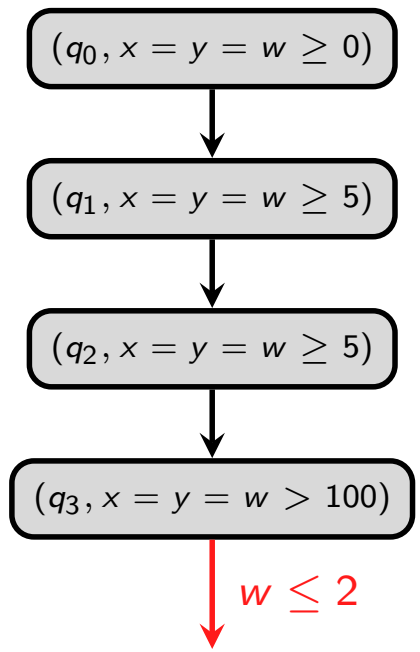
Otf-propagation would have given $\sim 10^6$ nodes

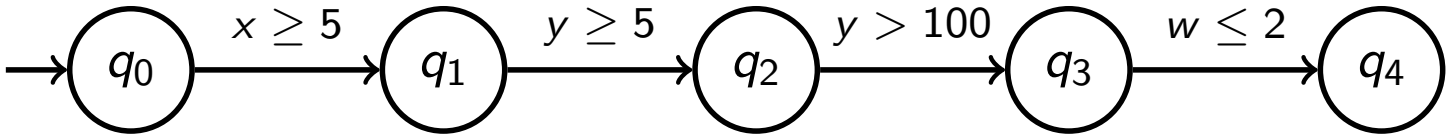
Trigger bounds propagation only when a **disabled edge** is seen

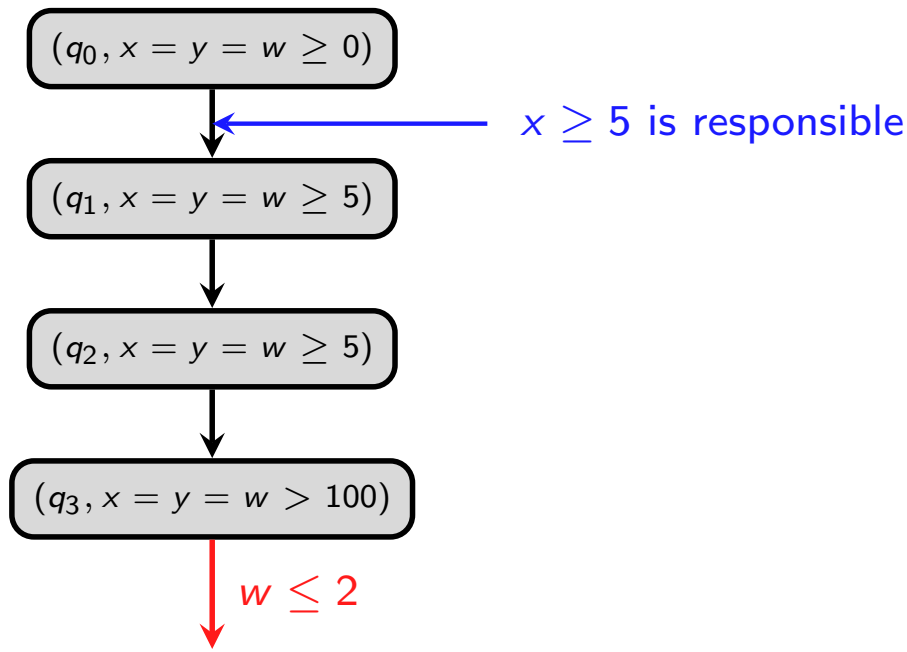
Observation 2:

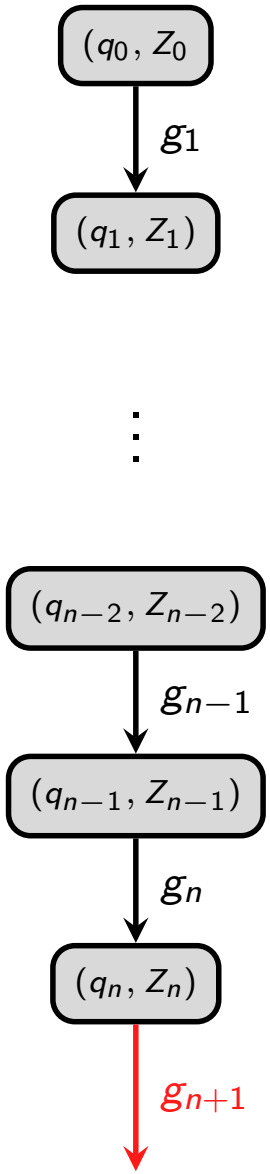
If **some edge is disabled** in the zone graph, it is enough to consider only the **guards that were responsible** for the edge to be disabled

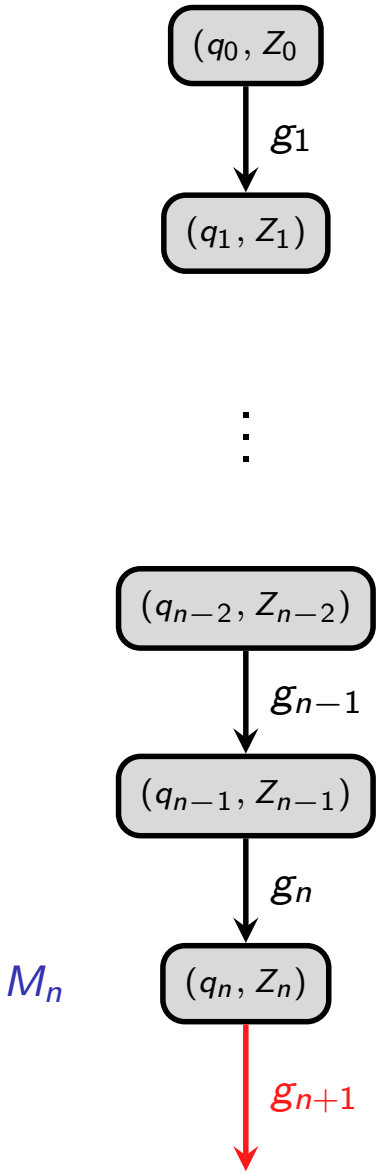

q_0
$x \geq 5$
q_1
$y \geq 5$
q_2
$y > 100$
q_3
$w \leq 2$
q_4


$(q_0, x = y = w \geq 0)$
$(q_1, x = y = w \geq 5)$
$(q_2, x = y = w \geq 5)$
$(q_3, x = y = w > 100)$
$w \leq 2$


q_0
$x \geq 5$
q_1
$y \geq 5$
q_2
$y > 100$
q_3
$w \leq 2$
q_4


$(q_0, x = y = w \geq 0)$
$x \geq 5$ is responsible
$(q_1, x = y = w \geq 5)$
$(q_2, x = y = w \geq 5)$
$(q_3, x = y = w > 100)$
$w \leq 2$

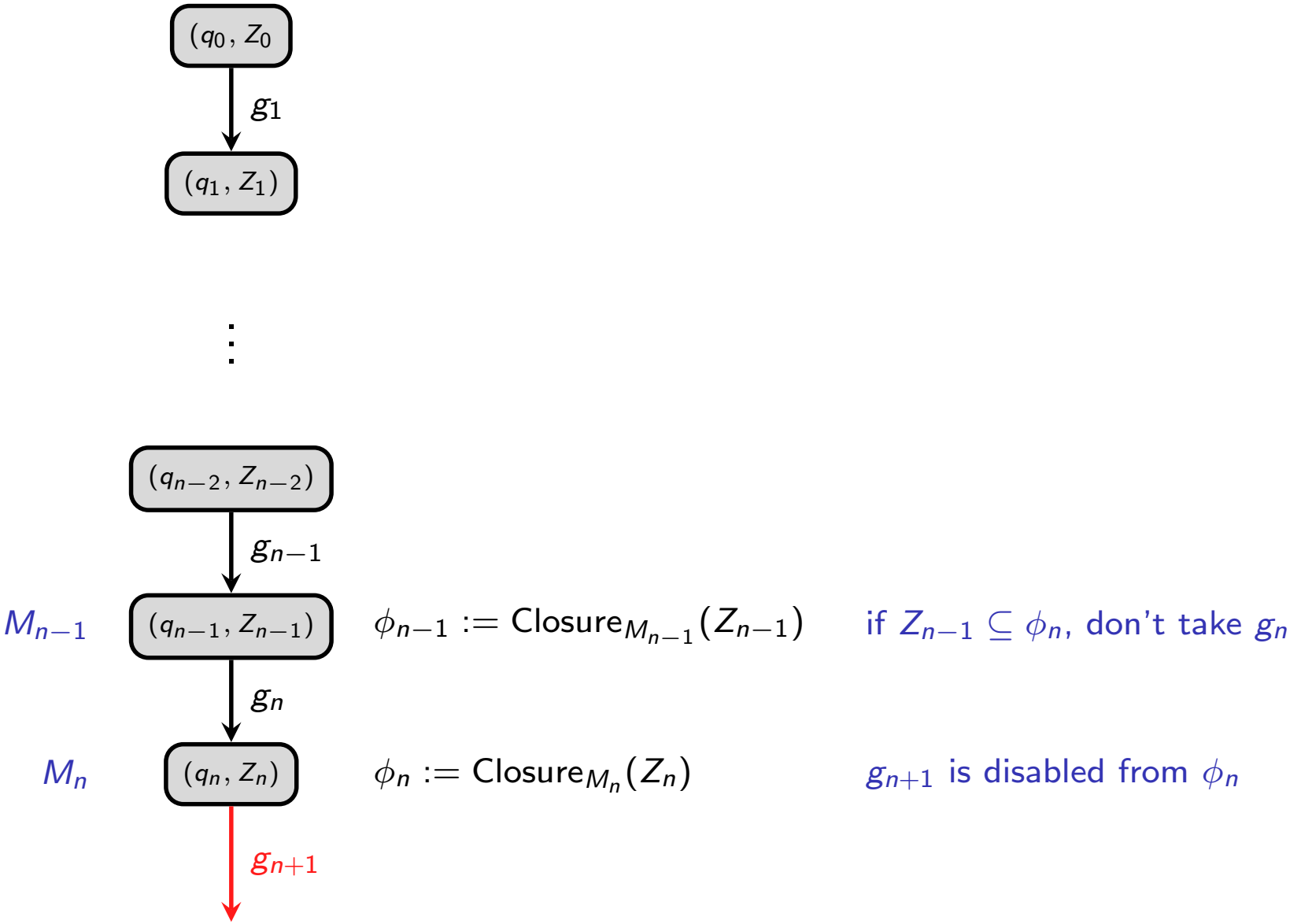

(q_0, Z_0)
g_1
(q_1, Z_1)
(q_{n-2}, Z_{n-2})
g_{n-1}
(q_{n-1}, Z_{n-1})
g_n
(q_n, Z_n)
g_{n+1}

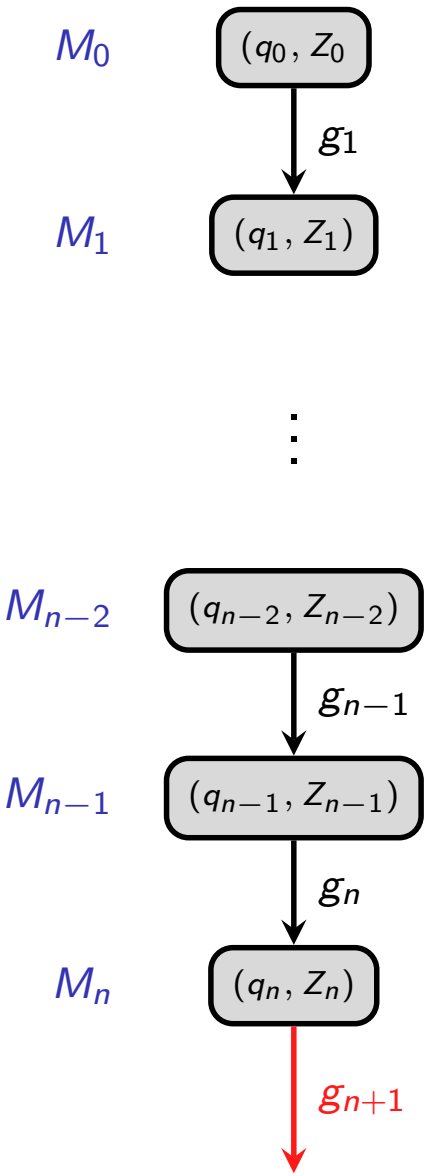


M_n

$\phi_n := \mathsf{Closure}_{M_n}(Z_n)$

g_{n+1} is disabled from ϕ_n


(q_0, Z_0)
g_1
(q_1, Z_1)
(q_{n-2}, Z_{n-2})
g_{n-1}
M_{n-1}
(q_{n-1}, Z_{n-1})
$\phi_{n-1} := \text{Closure}_{M_{n-1}}(Z_{n-1})$
if $Z_{n-1} \subseteq \phi_n$, don't take g_n
g_n
M_n
(q_n, Z_n)
$\phi_n := \text{Closure}_{M_n}(Z_n)$
g_{n+1} is disabled from ϕ_n
g_{n+1}



$\phi_1 := \text{Closure}_{M_0}(Z_0)$ if $Z_0 \subseteq \phi_1$, don't take g_1

$\phi_1 := \text{Closure}_{M_1}(Z_1)$ if $Z_1 \subseteq \phi_2$, don't take g_2

$\phi_{n-1} := \text{Closure}_{M_{n-2}}(Z_{n-2})$ if $Z_{n-2} \subseteq \phi_{n-1}$, don't take g_{n-1}

$\phi_{n-1} := \text{Closure}_{M_{n-1}}(Z_{n-1})$ if $Z_{n-1} \subseteq \phi_n$, don't take g_n

$\phi_n := \text{Closure}_{M_n}(Z_n)$ g_{n+1} is disabled from ϕ_n

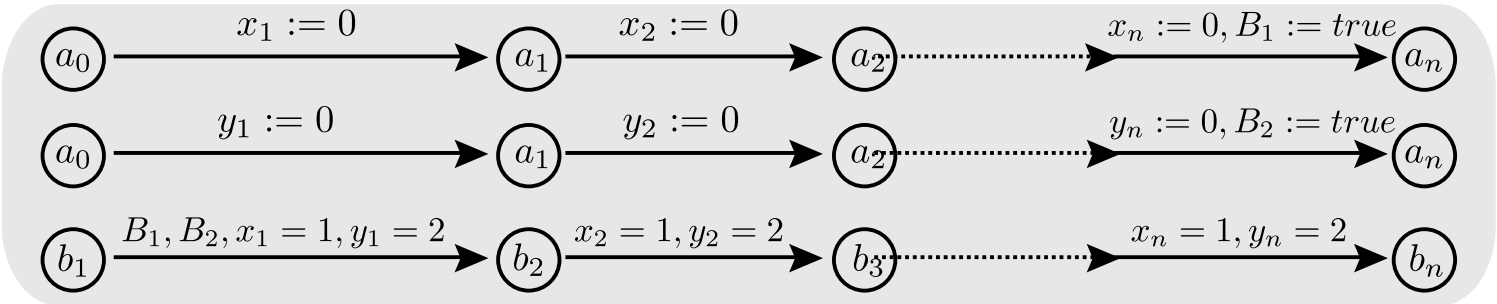
Lazy propagation

A new **efficient** propagation algorithm based on relation between **successor** zones

Lazy abstractions for timed automata (CoRR abs/1301.3127)

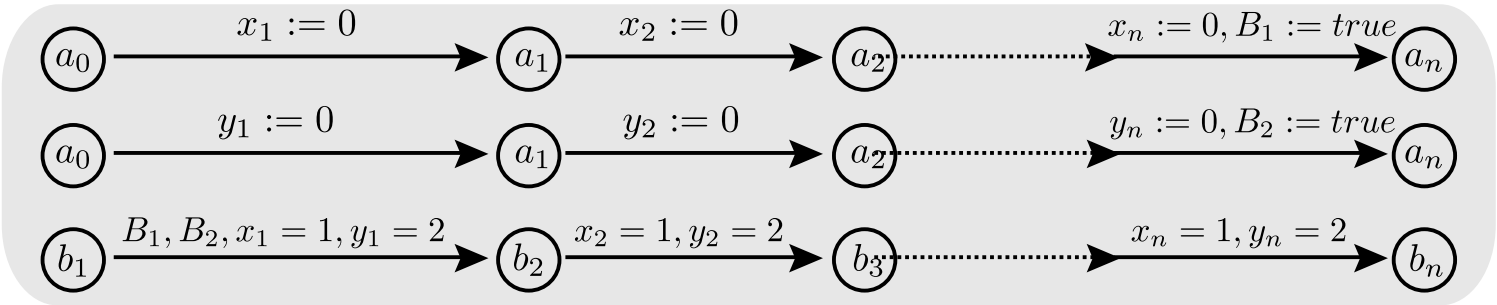
Exponential gain

$\mathcal{D}'_n$



Exponential gain

$\mathcal{D}'_n$



- **Lazy:** gives constants only for some pair (x_i, y_i) in any path **(quadratic zone graph)**
- **Static:** gives constants for all clocks **(exponential)**
- **Otf:** gives constants for k clocks depending on order of exploration **(exponential)**

Experiments

Model	nb. of clocks	UPPAAL (-C) nodes	UPPAAL (-C) sec.	static nodes	static sec.	otf nodes	otf sec.	lazy nodes	lazy sec.
CSMA/CD 10	11	120845	1.9						
CSMA/CD 11	12	311310	5.4						
CSMA/CD 12	13	786447	14.8						
FDDI 50	151	12605	52.9						
FDDI 70	211								
FDDI 140	421								
Fischer 9	9	135485	2.4						
Fischer 10	10	447598	10.1						
Fischer 11	11	1464971	40.4						
Stari 2	7	7870	0.1						
Stari 3	10	136632	1.7						
Stari 4	13	1323193	26.2						

- **UPPAAL (-C)** shows results from UPPAAL tool which uses static bounds
- **static** is our implementation of UPPAAL's algo
- Time out (150s), Memory out (1Gb)

Experiments

Model	nb. of clocks	UPPAAL (-C) nodes	UPPAAL (-C) sec.	static nodes	static sec.	otf nodes	otf sec.	lazy nodes	lazy sec.
CSMA/CD 10	11	120845	1.9	120844	6.3				
CSMA/CD 11	12	311310	5.4	311309	16.8				
CSMA/CD 12	13	786447	14.8	786446	44.0				
FDDI 50	151	12605	52.9	12606	29.4				
FDDI 70	211								
FDDI 140	421								
Fischer 9	9	135485	2.4	135485	8.9				
Fischer 10	10	447598	10.1	447598	34.0				
Fischer 11	11	1464971	40.4	1464971	126.8				
Stari 2	7	7870	0.1	6993	0.4				
Stari 3	10	136632	1.7	113958	9.4				
Stari 4	13	1323193	26.2	983593	109.0				

- **UPPAAL (-C)** shows results from UPPAAL tool which uses static bounds
- **static** is our implementation of UPPAAL's algo
- Time out (150s), Memory out (1Gb)

Experiments

Model	nb. of clocks	UPPAAL (-C) nodes	UPPAAL (-C) sec.	static nodes	static sec.	otf nodes	otf sec.	lazy nodes	lazy sec.
CSMA/CD 10	11	120845	1.9	120844	6.3				
CSMA/CD 11	12	311310	5.4	311309	16.8				
CSMA/CD 12	13	786447	14.8	786446	44.0				
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	37.6
Fischer 9	9	135485	2.4	135485	8.9				
Fischer 10	10	447598	10.1	447598	34.0				
Fischer 11	11	1464971	40.4	1464971	126.8				
Stari 2	7	7870	0.1	6993	0.4	5779	0.4	5113	0.5
Stari 3	10	136632	1.7	113958	9.4	82182	8.2	53178	7.8
Stari 4	13	1323193	26.2	983593	109.0	602762	84.9	342801	65.7

- **UPPAAL (-C)** shows results from UPPAAL tool which uses static bounds
- **static** is our implementation of UPPAAL's algo
- Time out (150s), Memory out (1Gb)

Experiments

Model	nb. of clocks	UPPAAL (-C) nodes	UPPAAL (-C) sec.	static nodes	static sec.	otf nodes	otf sec.	lazy nodes	lazy sec.
CSMA/CD 10	11	120845	1.9	120844	6.3	78604	6.1	74324	6.1
CSMA/CD 11	12	311310	5.4	311309	16.8	198669	16.1	188315	15.9
CSMA/CD 12	13	786447	14.8	786446	44.0	493582	41.8	469027	40.9
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	37.6
Fischer 9	9	135485	2.4	135485	8.9	135485	11.4	135485	24.7
Fischer 10	10	447598	10.1	447598	34.0	447598	42.8	447598	98.1
Fischer 11	11	1464971	40.4	1464971	126.8				
Stari 2	7	7870	0.1	6993	0.4	5779	0.4	5113	0.5
Stari 3	10	136632	1.7	113958	9.4	82182	8.2	53178	7.8
Stari 4	13	1323193	26.2	983593	109.0	602762	84.9	342801	65.7

- **UPPAAL (-C)** shows results from UPPAAL tool which uses static bounds
- **static** is our implementation of UPPAAL's algo
- Time out (150s), Memory out (1Gb)

Summary and future work

- A **new algorithm** for obtaining abstraction parameters
  - works the same way for LU-bounds and $\mathfrak{a}_{\preccurlyeq_{LU}}$ abstraction
- Theoretically, **exponential** gain possible
- Practical gains understood by **experiments**

- Better **data-structures** for zones
- Abstractions for **discrete** component
- Probabilistic systems, (rectangular) hybrid systems

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.

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.

F. Herbreteau, D. Kini, B. Srivathsan, and I. Walukiewicz.

Using non-convex approximations for efficient analysis of timed automata.
In *Proceedings of FSTTCS*, volume 13 of *LIPIcs*, pages 78–89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

F. Herbreteau, B. Srivathsan, and I. Walukiewicz.

Better abstractions for timed automata.
In *LICS*, 2012.