Non-convex on-the-fly abstractions for timed automata

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

Univ. Bordeaux, LaBRI, CNRS, UMR 5800

ANR VACSIM meeting, March 16th, 2012

FSTTCS 2011 + arXiv:1110.3705v3 (submitted)

Outline

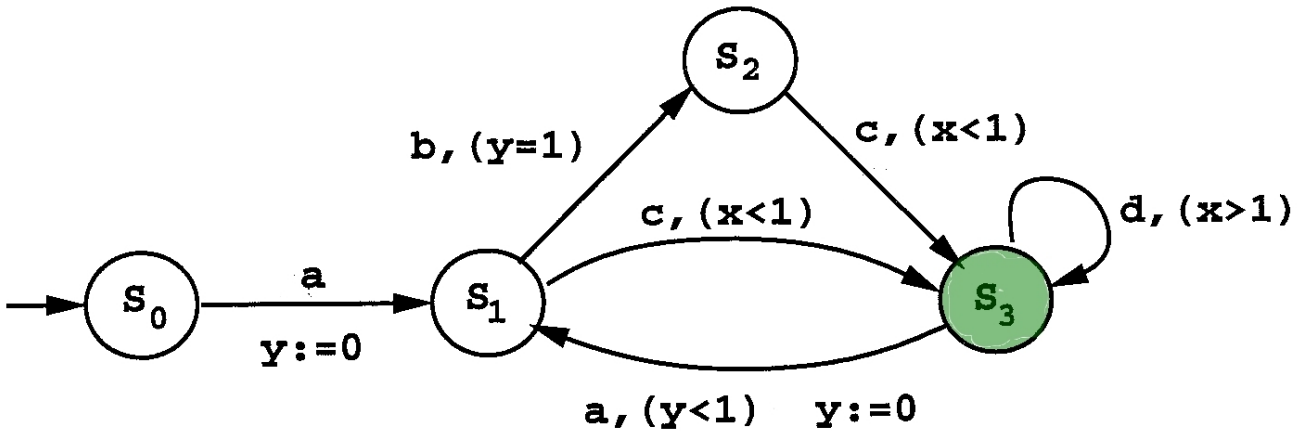
Timed Automata and the Reachability Problem

Symbolic Semantics

Efficient Use of non-Convex Abstraction Closure

On-the-fly Bounds Propagation

Timed Automata [AD94]



Run: finite **sequence** of transitions,

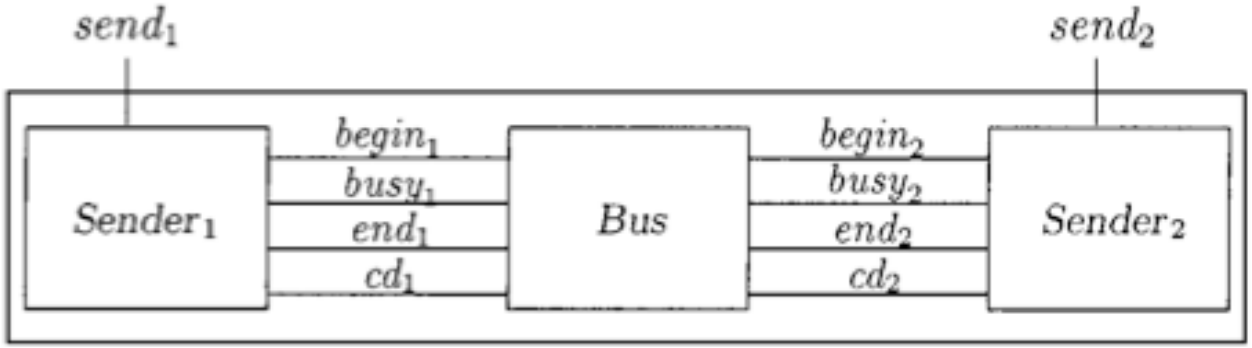
$$(s_0, \overbrace{0}^{x}, \overbrace{0}^{y}) \xrightarrow{0.4,a} (s_1, 0.4, 0) \xrightarrow{0.5,c} (s_3, 0.9, 0.5)$$

Accepting run: ends in a green state.

An example: the CSMA/CD protocol

- Before sending, a station **first listens** to the bus
  - if idle, the station begins transmitting
  - otherwise, the station waits (random) and retries
- **Collisions** may occur (propagation delays, etc)
  - then **all stations detect the collision**
  - and all station abort, wait (random) and start all over again.

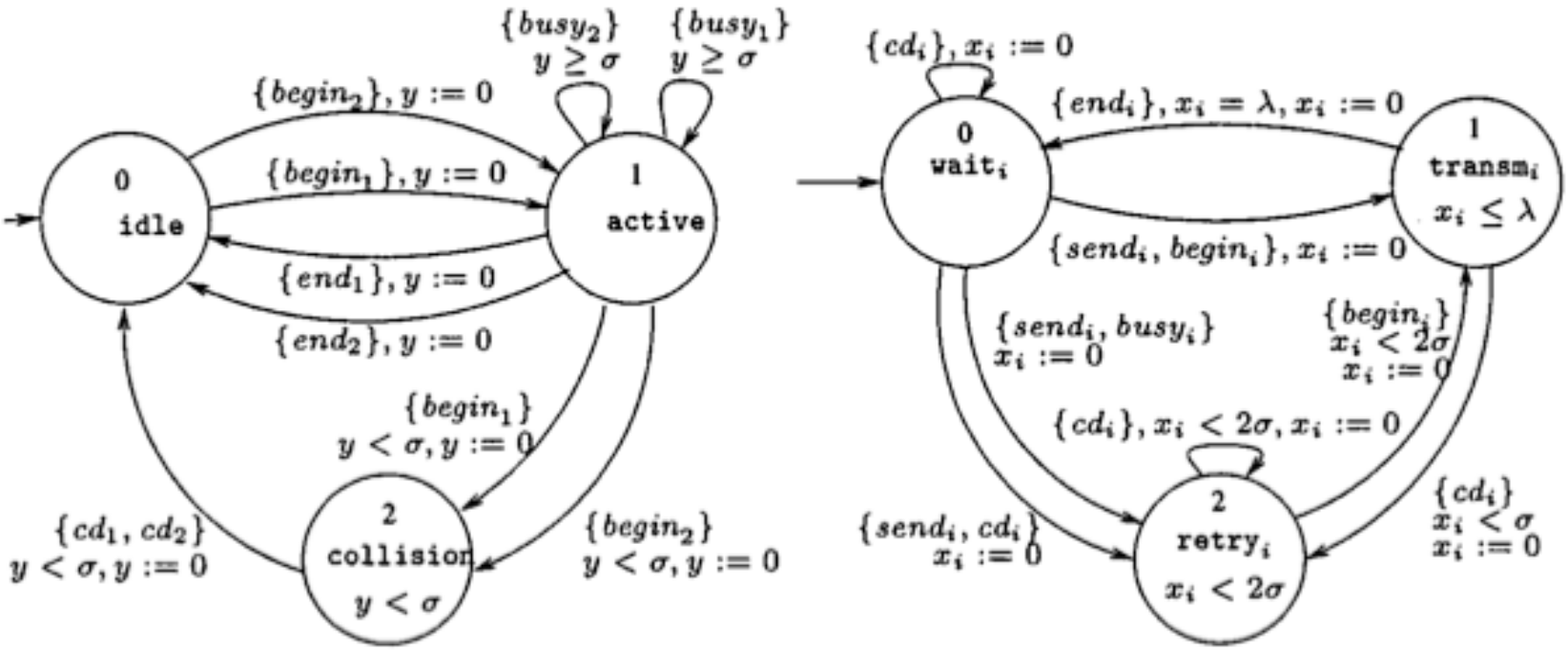
Modeling the CSMA/CD protocol [TY01]



- Error-free 10 megabits/sec bus
- Propagation delay σ at most $26\mu s$
- Fixed length 1024 bytes messages, send time λ is $808\mu s$

Property (to be checked): **all** stations detect collisions

Modeling the CSMA/CD protocol [TY01]



Propagation delay σ, Send time λ.

Property (to be checked): if $\texttt{transm}_1$ and $\texttt{transm}_2$ then `collision`.

The problem we are interested in ...

Problem (Emptiness/State reachability)

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

Example (CSMA/CD): can we have $\texttt{transm}_1$ and $\texttt{transm}_2$ but not `collision`?

The problem we are interested in ...

Problem (Emptiness/State reachability)

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

Example (CSMA/CD): can we have $\texttt{transm}_1$ and $\texttt{transm}_2$ but not `collision`?

Theorem ([AD94, CY92])

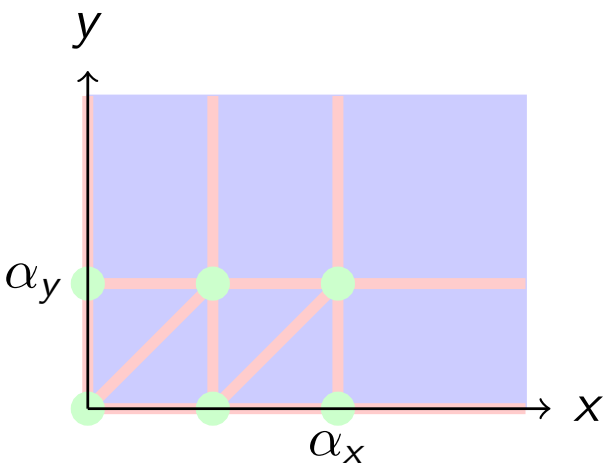
This problem is **PSPACE-complete**

How to handle the **uncountable state space**?

Outline

First solution to this problem: region graph

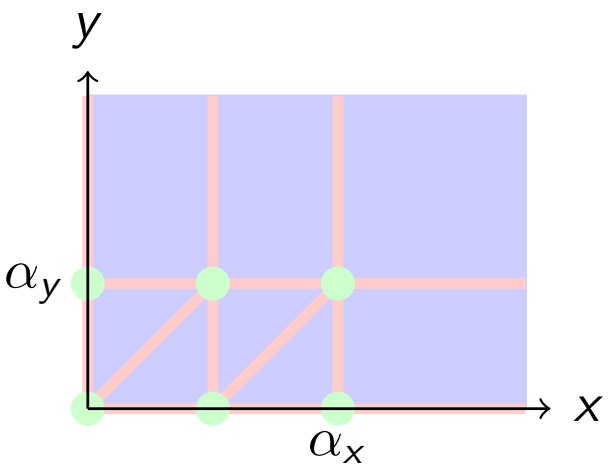
Key idea: Partition the space of valuations into a **finite** number of **regions**



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

First solution to this problem: region graph

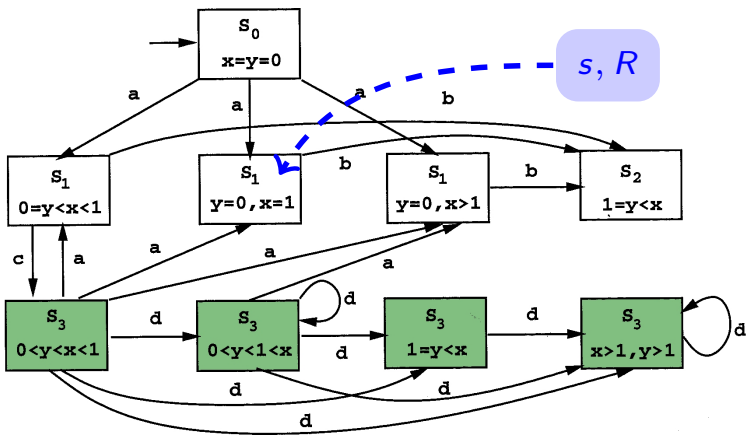
Key idea: Partition the space of valuations into a **finite** number of **regions**



- **Region:** set of valuations that enable the **same sequences of transitions**
- **Finiteness:** Parametrized by **maximal constants** in $\mathcal{A}$

First solution to this problem: region graph

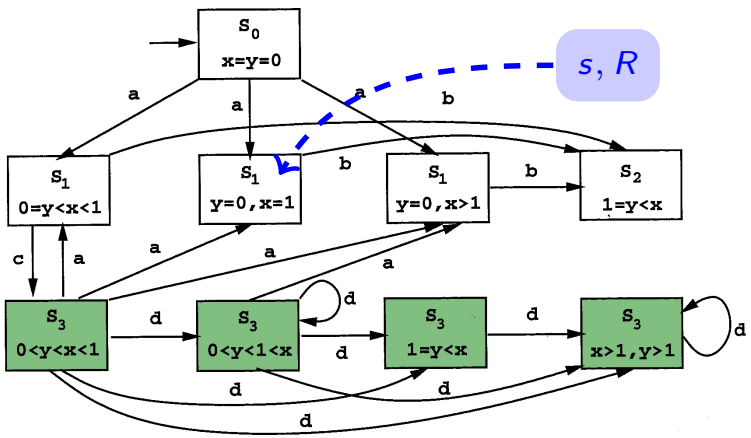
Key idea: Partition the space of valuations into a **finite** number of **regions**



- **Region:** set of valuations that enable the **same sequences of transitions**
- **Finiteness:** Parametrized by **maximal constants** in $\mathcal{A}$

First solution to this problem: region graph

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



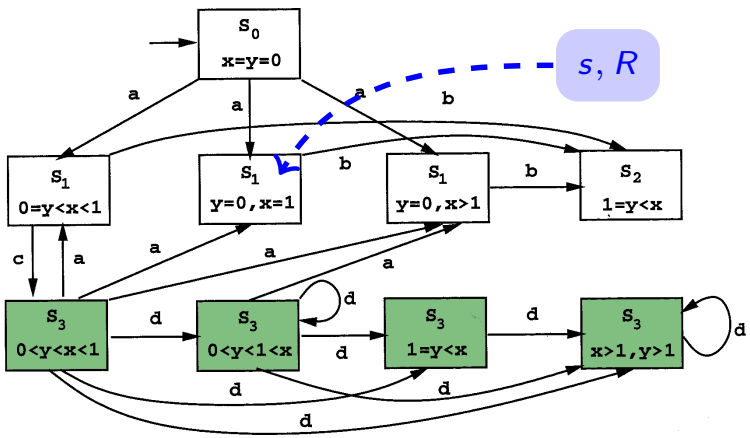
- **Region:** set of valuations that enable the **same sequences of transitions**
- **Finiteness:** Parametrized by **maximal constants** in $\mathcal{A}$

Theorem (Sound and complete [AD94])

Region graph preserves state **reachability**

First solution to this problem: region graph

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



- **Region:** set of valuations that enable the **same sequences of transitions**
- **Finiteness:** Parametrized by **maximal constants** in $\mathcal{A}$

Theorem (Sound and complete [AD94])

Region graph preserves state **reachability**

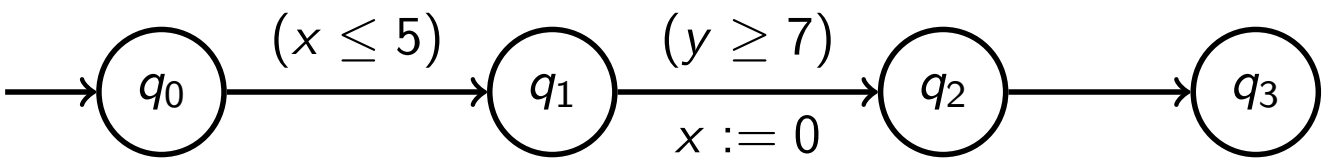
Too many regions: $\mathcal{O}(|X|!.\alpha^{|X|})$

A more efficient solution...

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

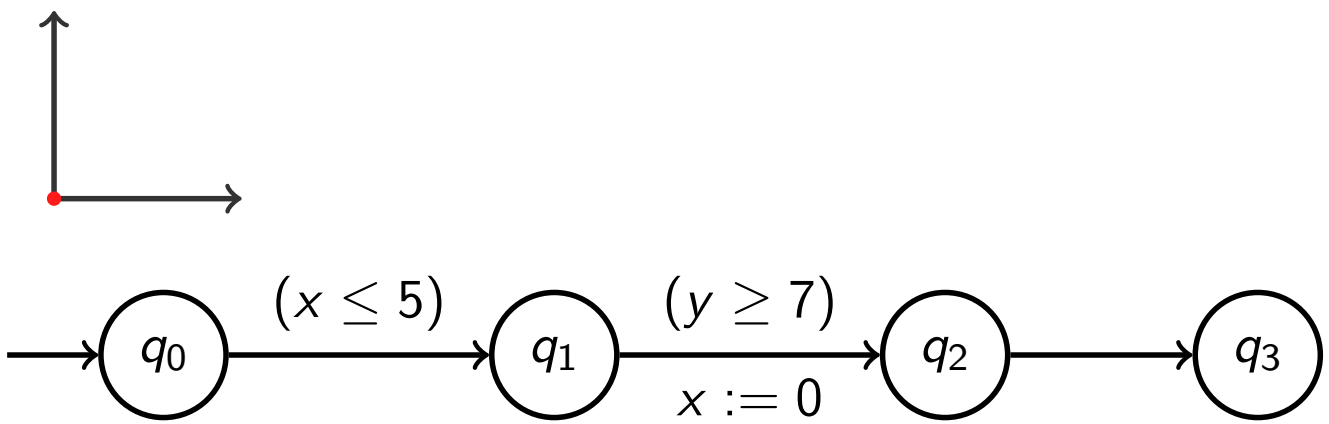
A more efficient solution...

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



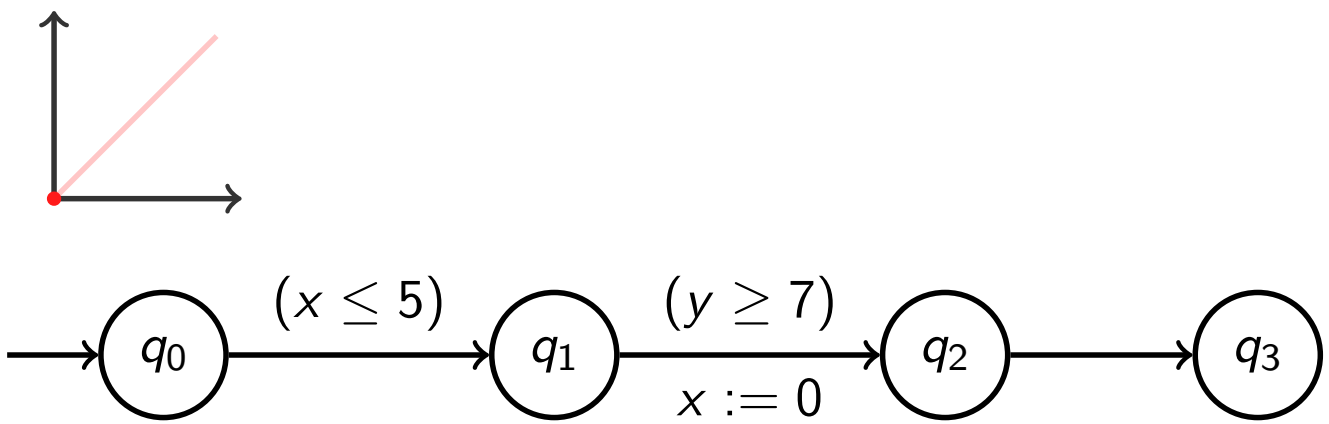
A more efficient solution...

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



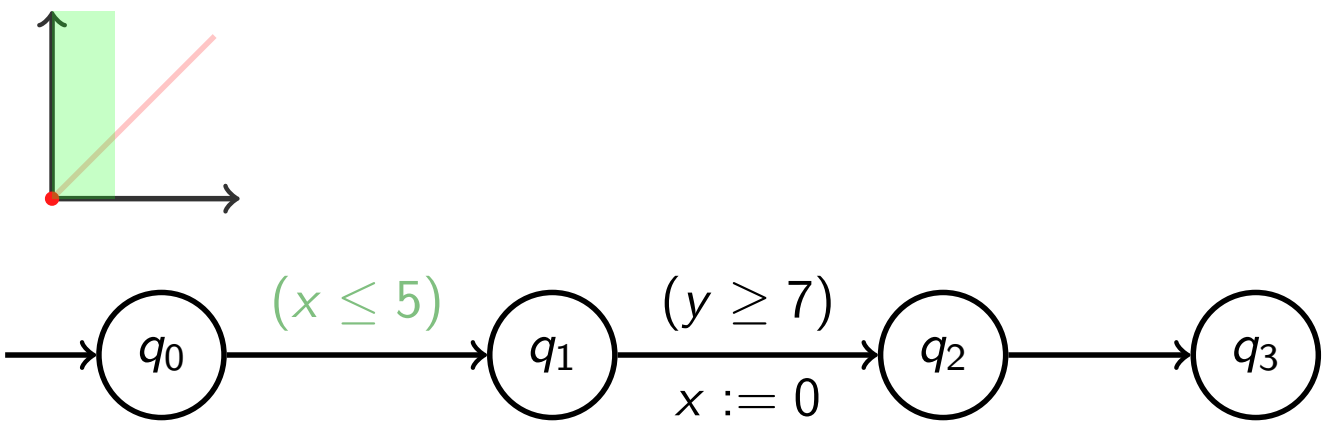
A more efficient solution...

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



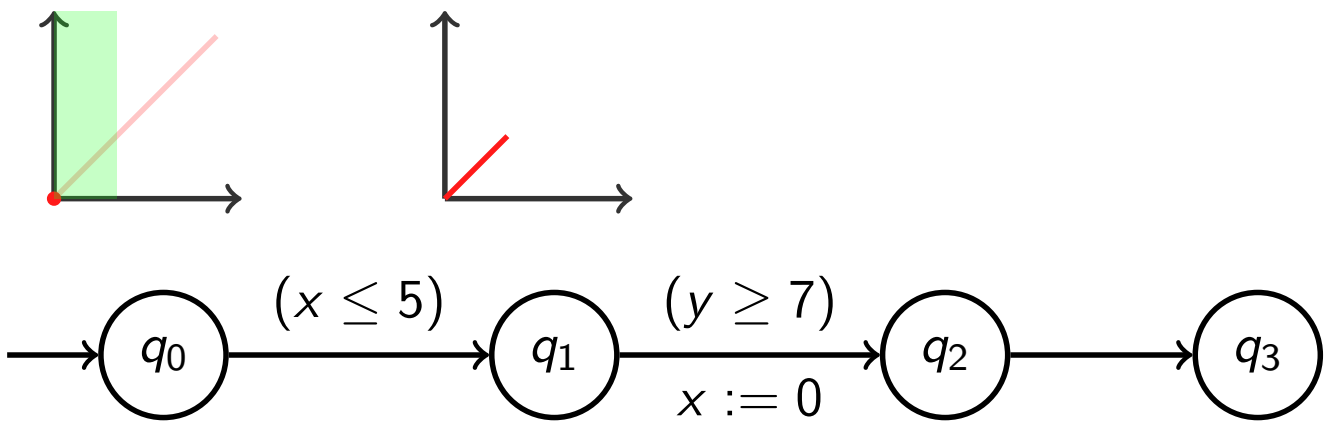
A more efficient solution...

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



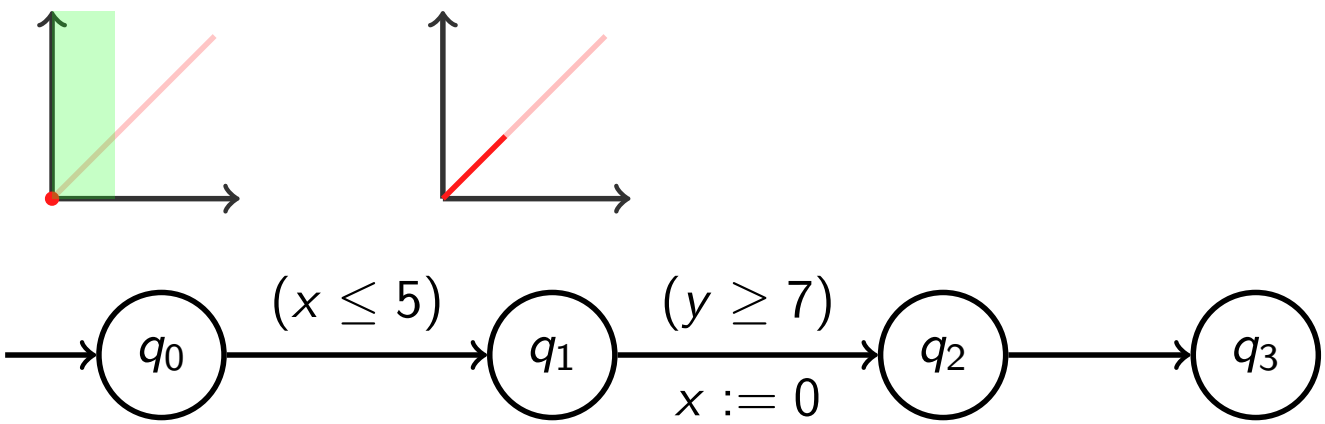
A more efficient solution...

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



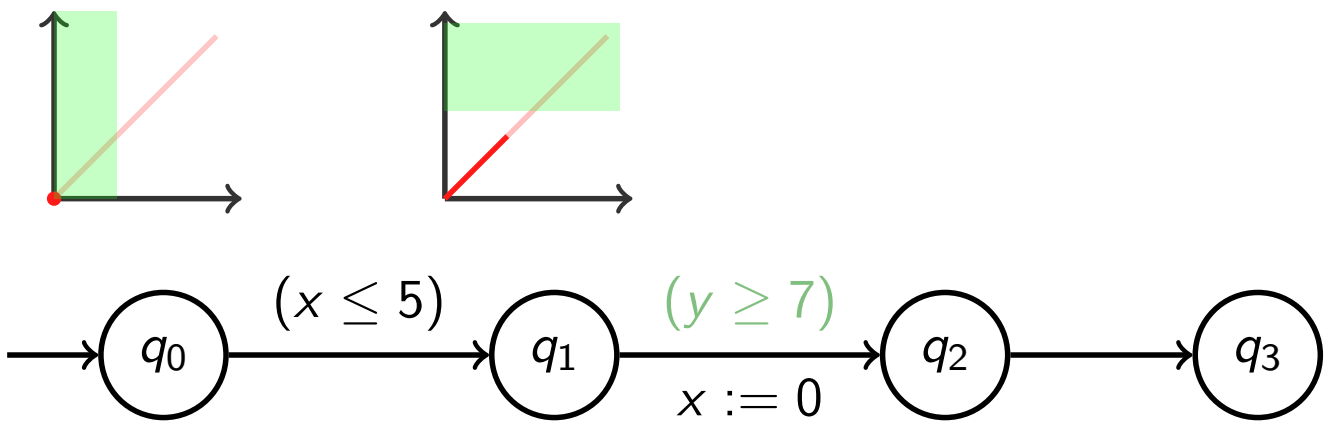
A more efficient solution...

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



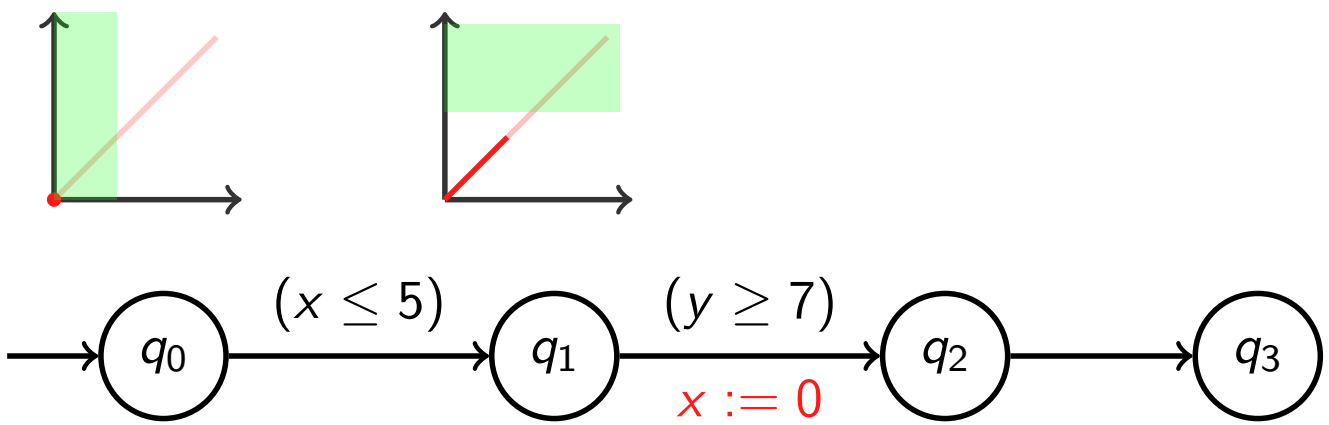
A more efficient solution...

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



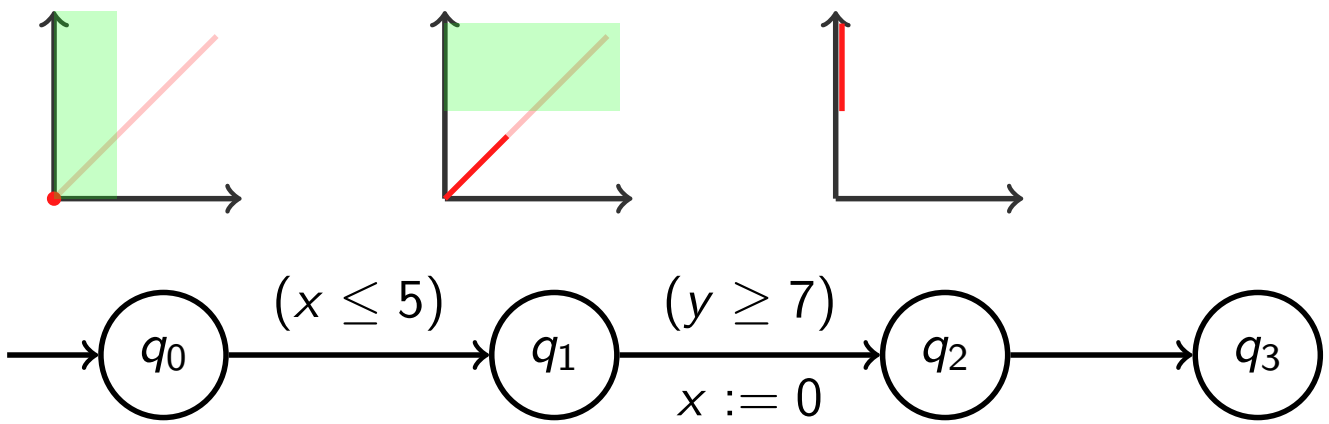
A more efficient solution...

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



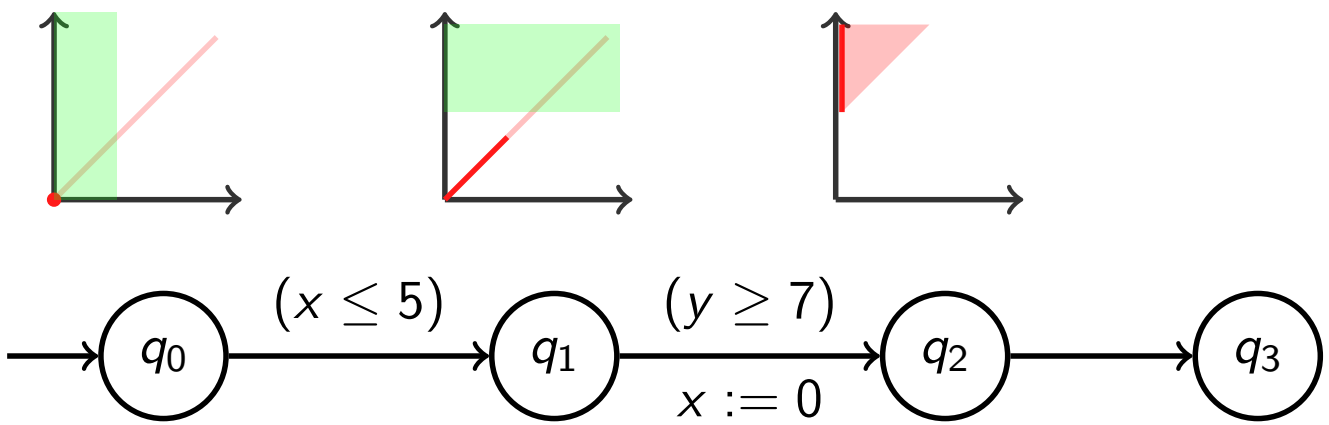
A more efficient solution...

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



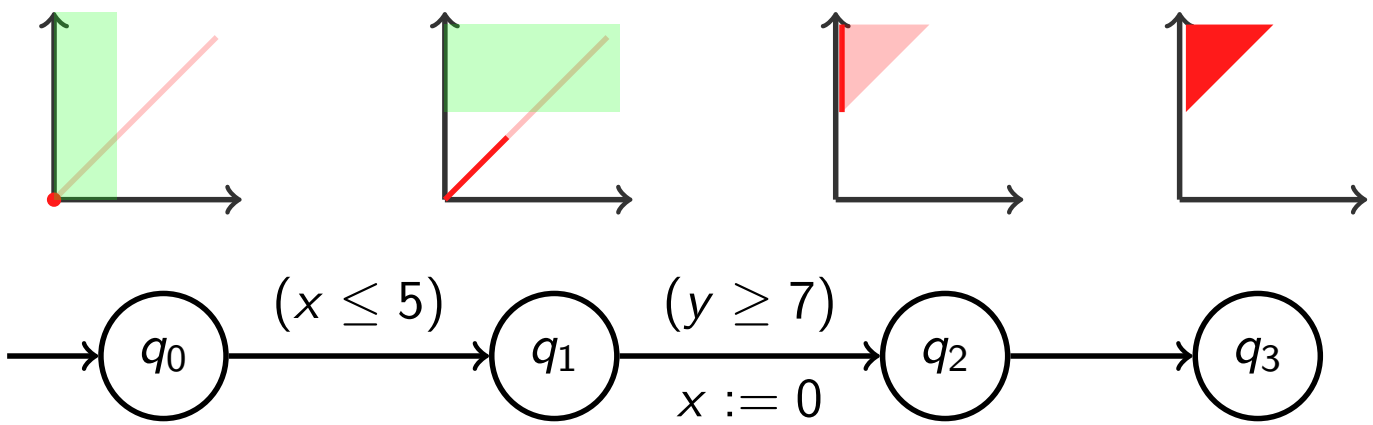
A more efficient solution...

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



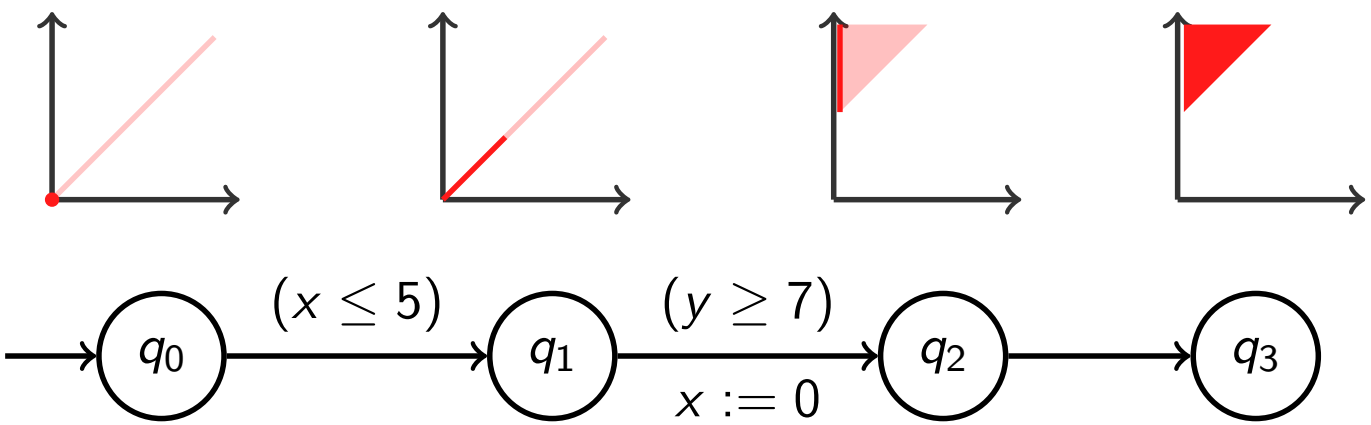
A more efficient solution...

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



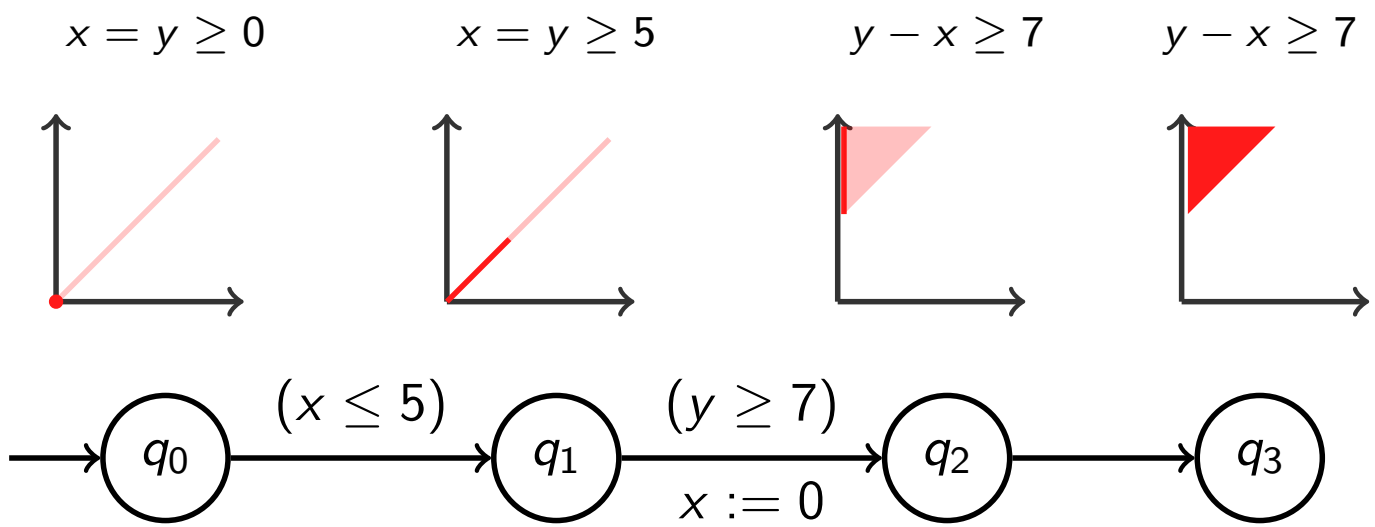
A more efficient solution...

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

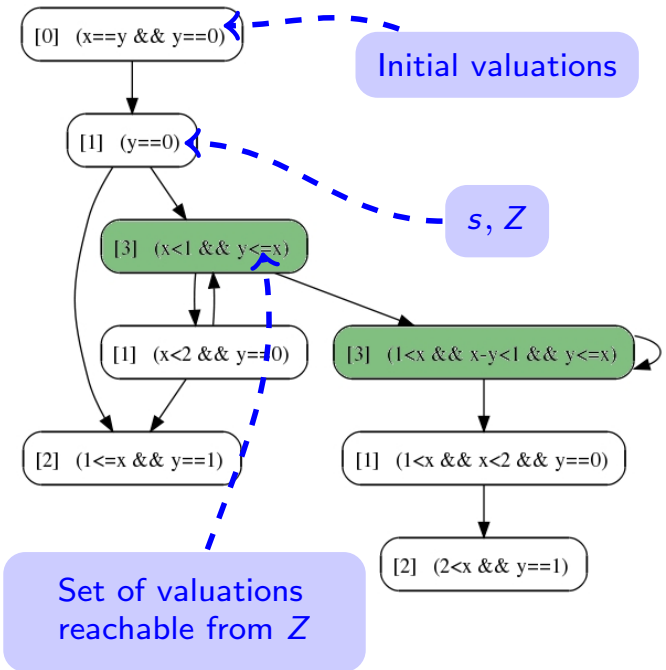


A more efficient solution...

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



A more efficient solution: zone graph ZG(A)



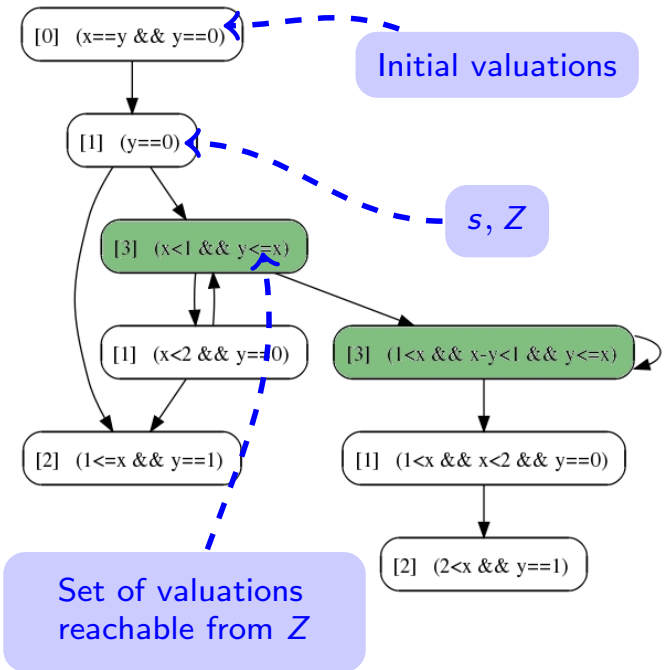
- **Reachability graph**
- **Zone:** set of valuations defined by conjunctions of constraints:

  e.g. $(x - y \geq 1) \wedge y < 2$
- **Efficient representation** of zones by DBM

Theorem (Sound and complete [DT98])

Zone graph preserves state **reachability**

A more efficient solution: zone graph ZG(A)



- **Reachability graph**
- **Zone:** set of valuations defined by conjunctions of constraints:

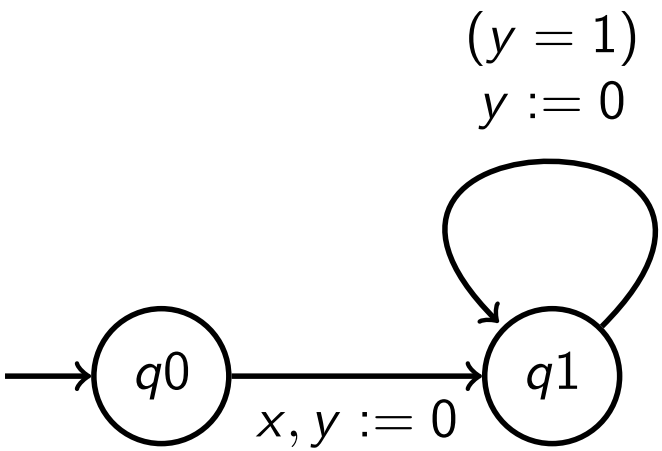
  e.g. $(x - y \geq 1) \wedge y < 2$
- **Efficient representation** of zones by DBM

Theorem (Sound and complete [DT98])

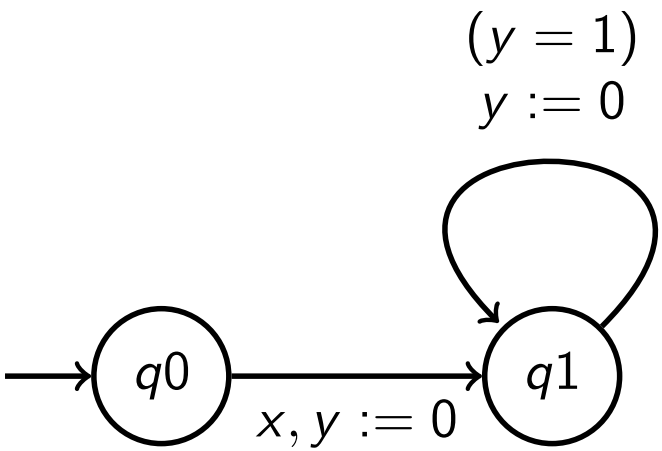
Zone graph preserves state **reachability**

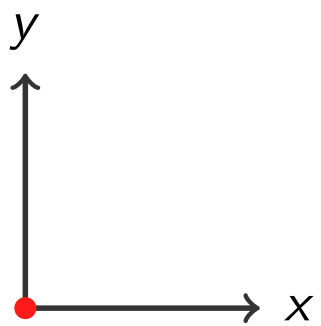
Maybe **infinite!**

Example of infinite zone graph

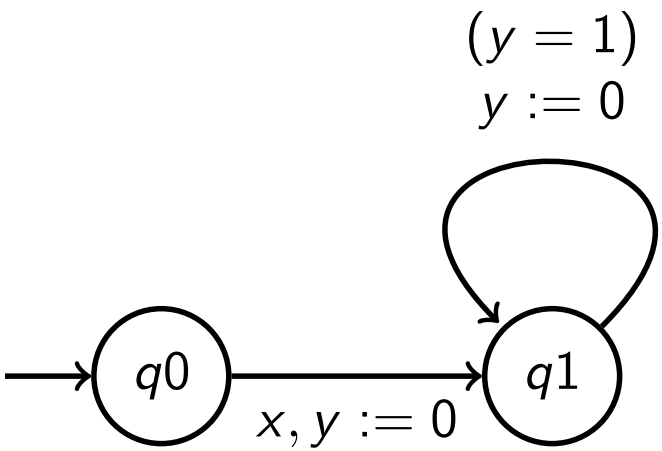


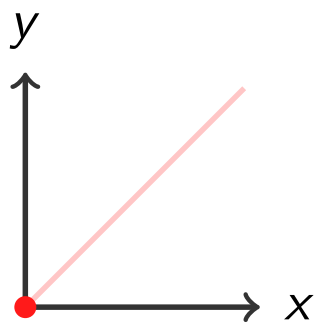
Example of infinite zone graph



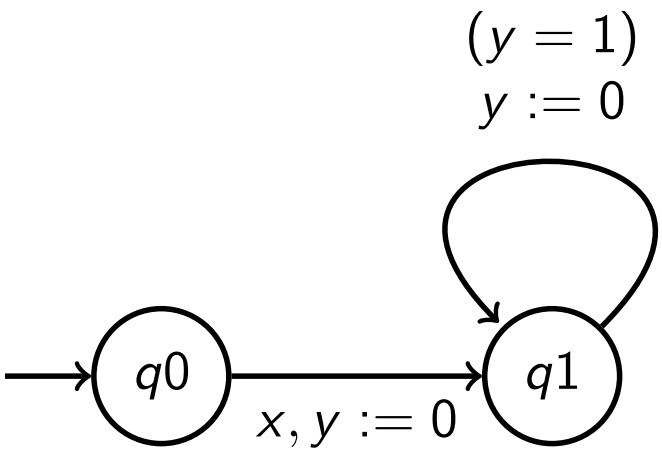


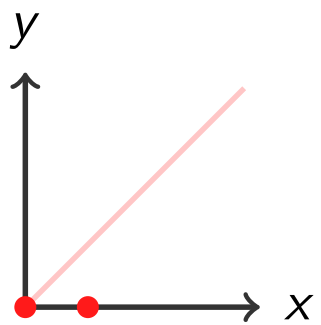
Example of infinite zone graph



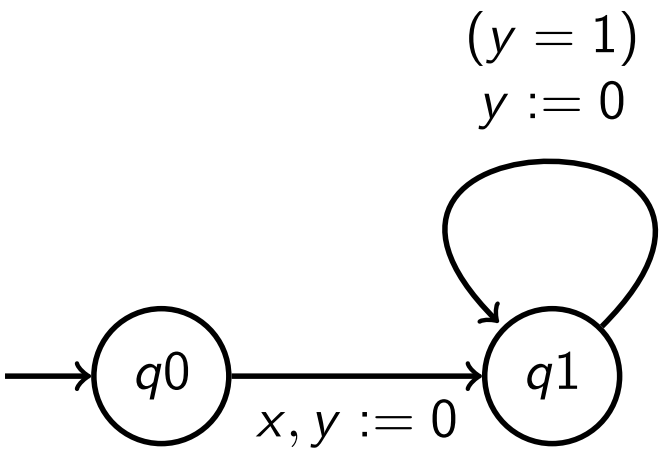


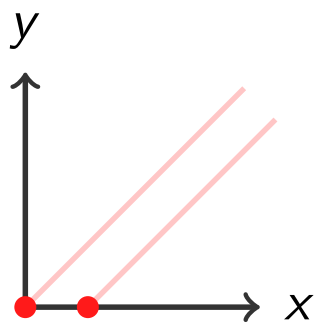
Example of infinite zone graph



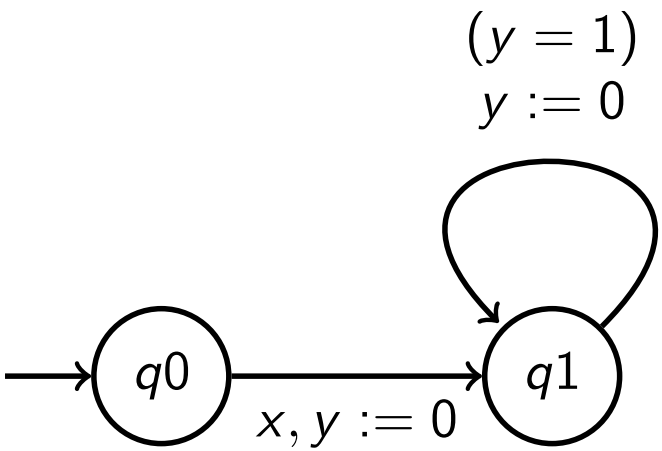


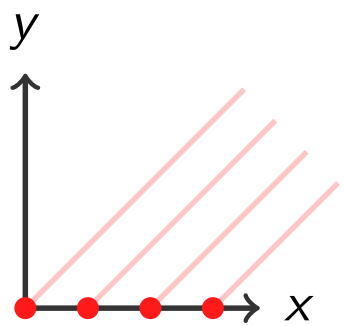
Example of infinite zone graph





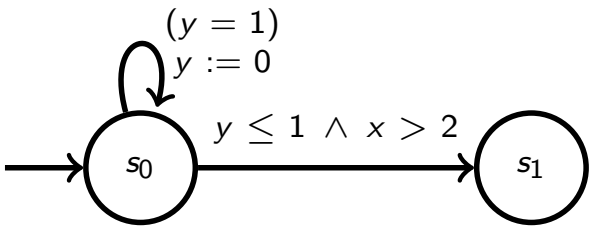
Example of infinite zone graph

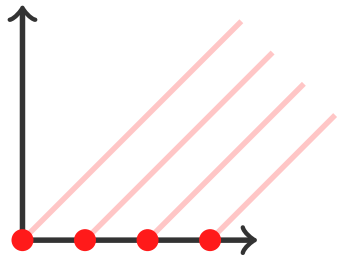




But zones above α can be abstracted away!

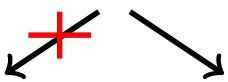
Use finite and sound abstraction $\mathfrak{a}$



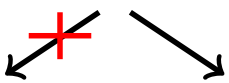


Key idea: **Abstract** each zone in a **sound** manner

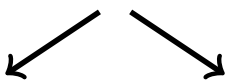
$s_0, x = y$



$s_0, 1 \leq x \wedge x - y = 1$

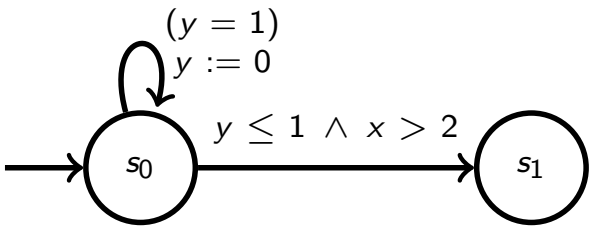


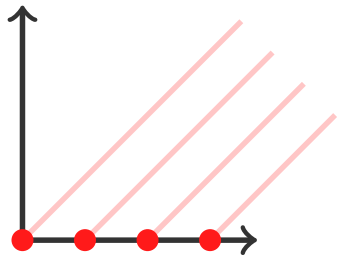
$s_0, 2 \leq x \wedge x - y = 2$



$s_0, \mathfrak{a}(x = y)$

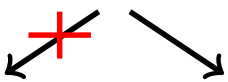
Use finite and sound abstraction α



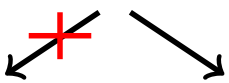


Key idea: **Abstract** each zone in a **sound** manner

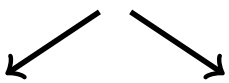
$s_0, x = y$



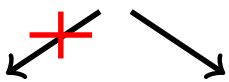
$s_0, 1 \leq x \wedge x - y = 1$



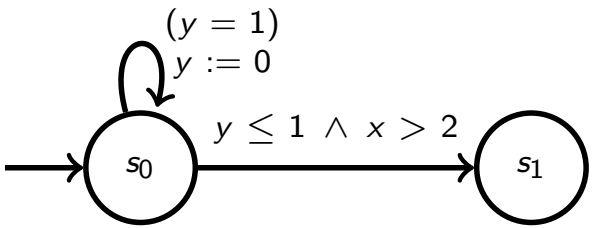
$s_0, 2 \leq x \wedge x - y = 2$

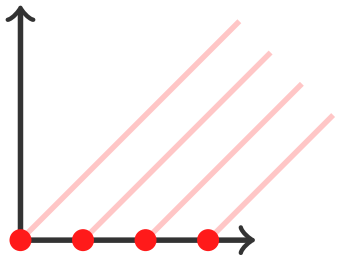


$s_0, \alpha(x = y)$



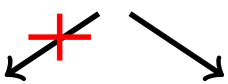
Use finite and sound abstraction α



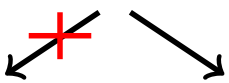


Key idea: **Abstract** each zone in a **sound** manner

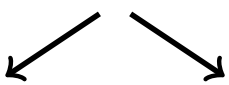
$s_0, x = y$



$s_0, 1 \leq x \wedge x - y = 1$



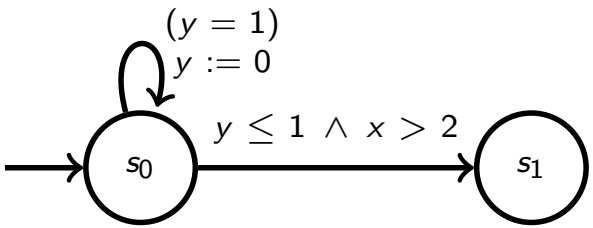
$s_0, 2 \leq x \wedge x - y = 2$

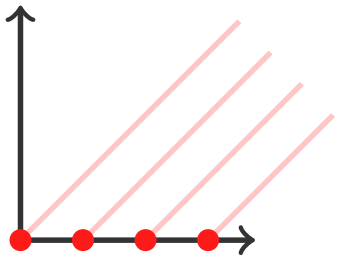


$s_0, \alpha(x = y)$

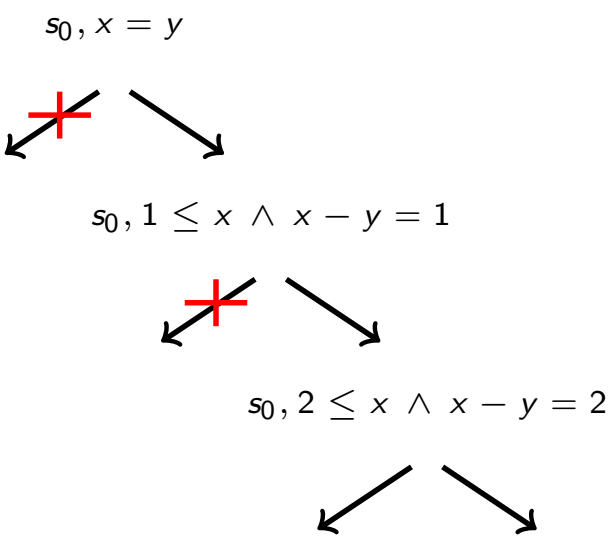
$s_0, \alpha(x - y \leq 1)$

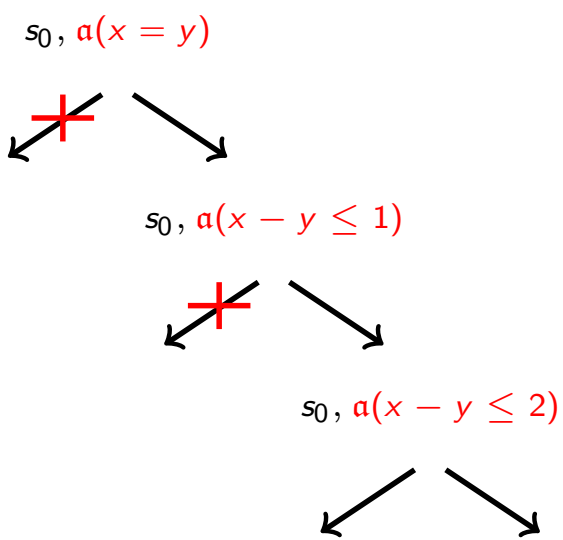
Use finite and sound abstraction $\mathfrak{a}$



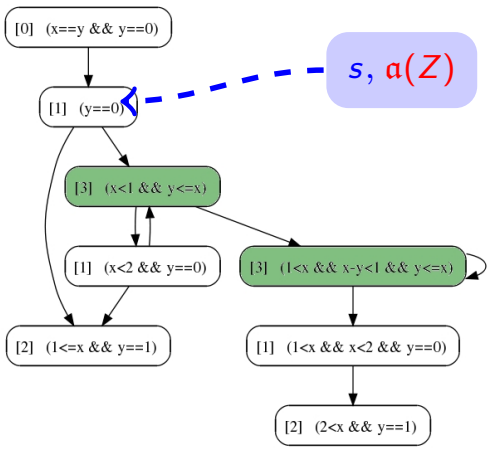


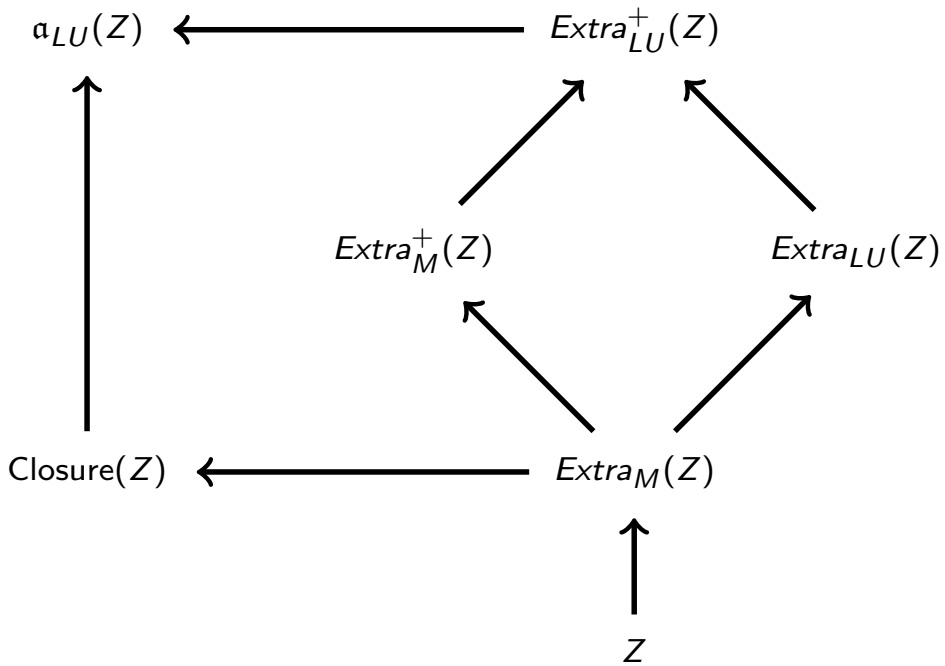
Key idea: **Abstract** each zone in a **sound** manner





Abstract Zone Graph $\mathrm{ZG}^{\mathfrak{a}}(A)$



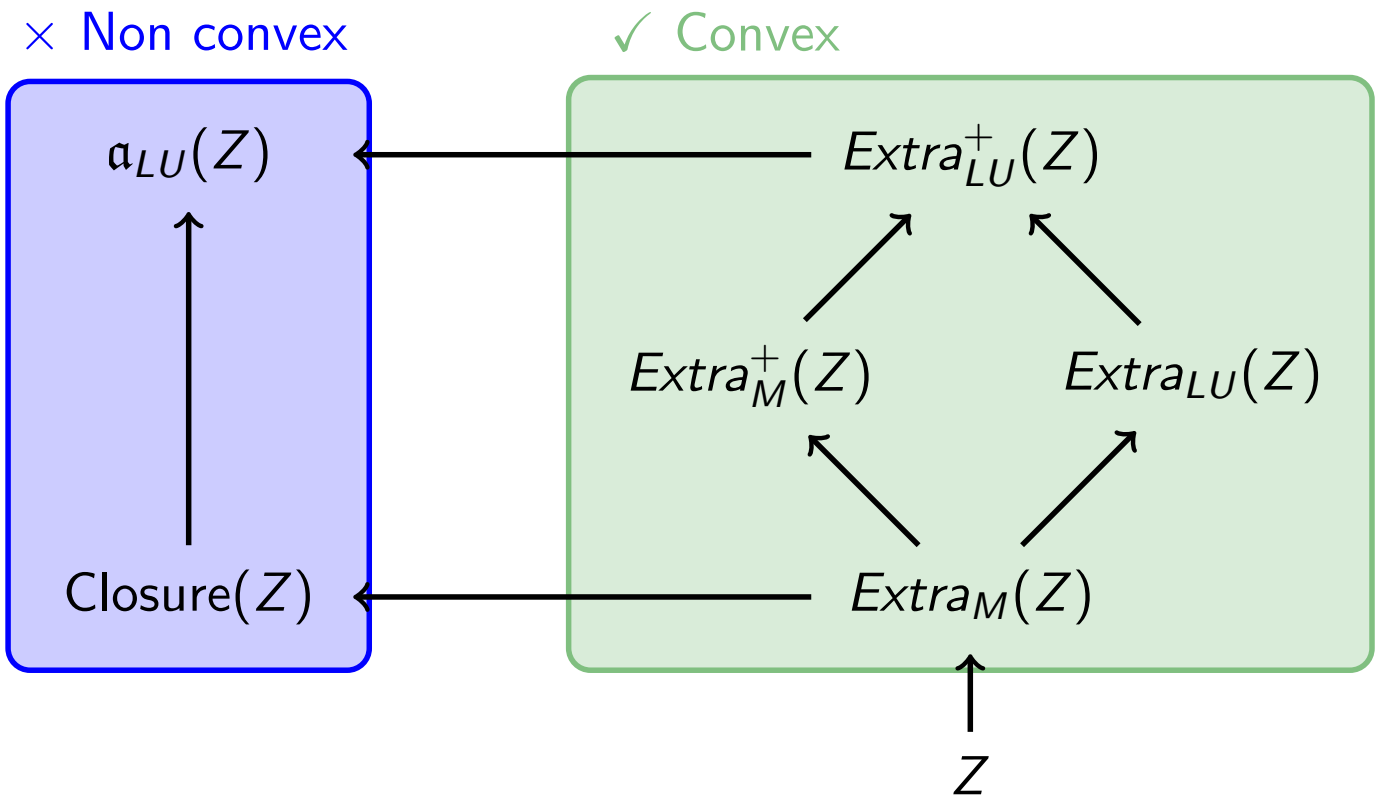


Theorem ([Bou04, BBLP06])

All these abstractions are **finite**, **sound** and **complete**

$|\mathrm{ZG}^{\mathfrak{a}}(A)|$ may vary **exponentially** depending on $\mathfrak{a}$

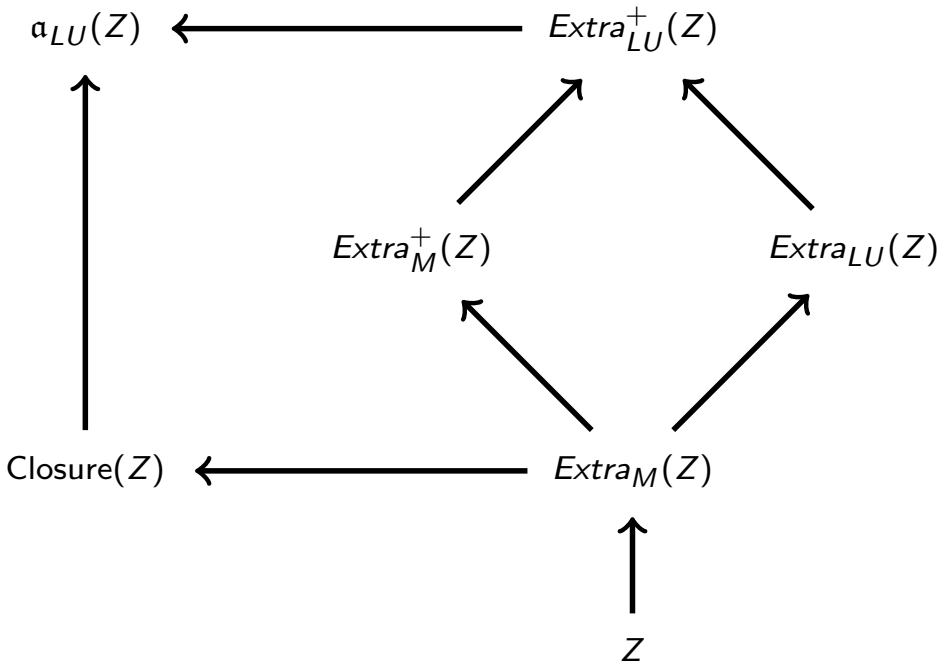
Convex abstractions



Only convex abstractions in **implementations**: $\mathfrak{a}(Z)$ must be a zone in order to use DBMs!

Outline

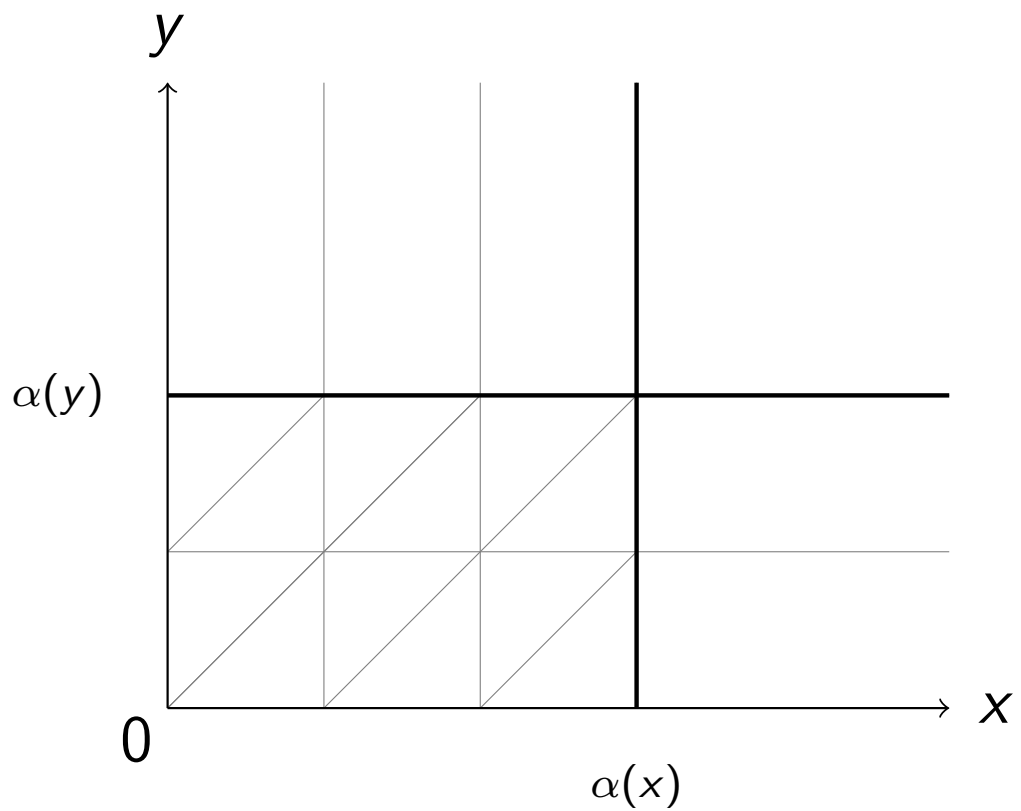
Using a non-convex abstraction



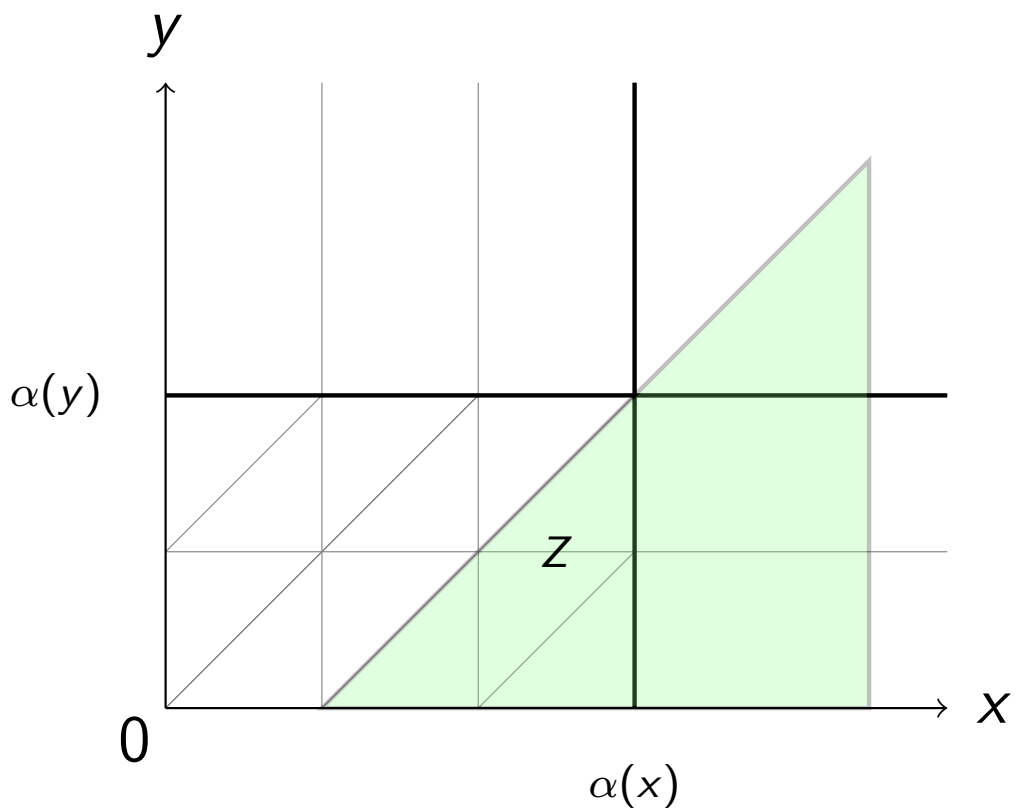
Contribution

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

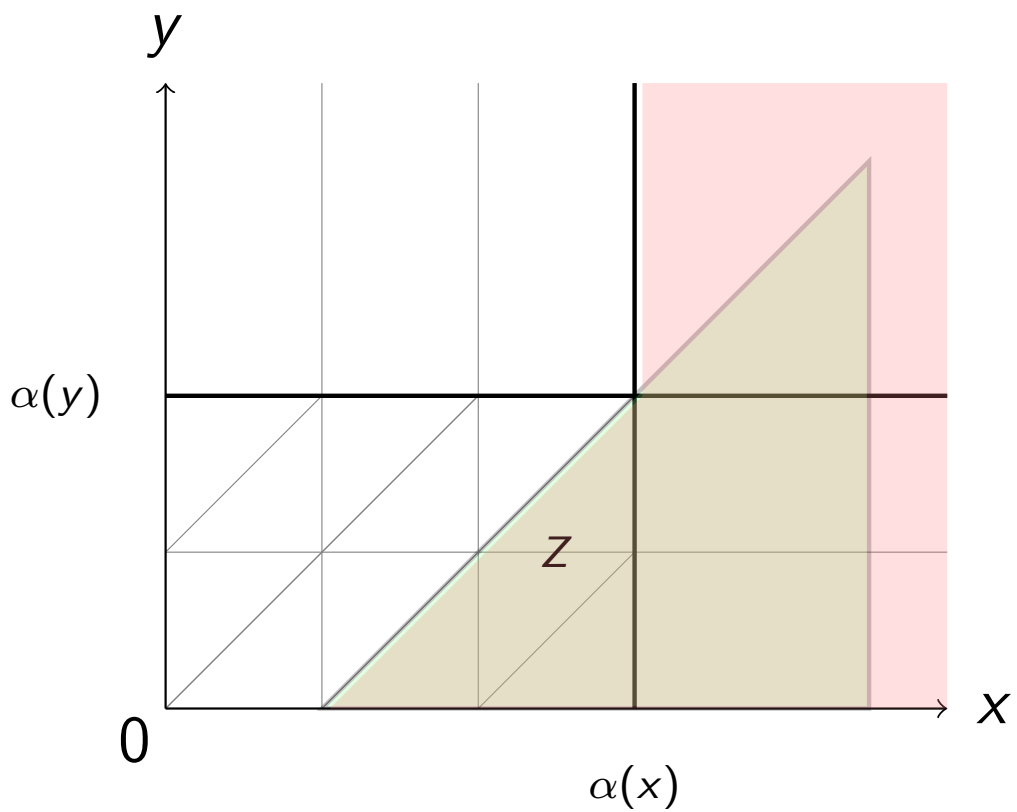
What is Closure$_\alpha$?



What is Closure$_\alpha$?

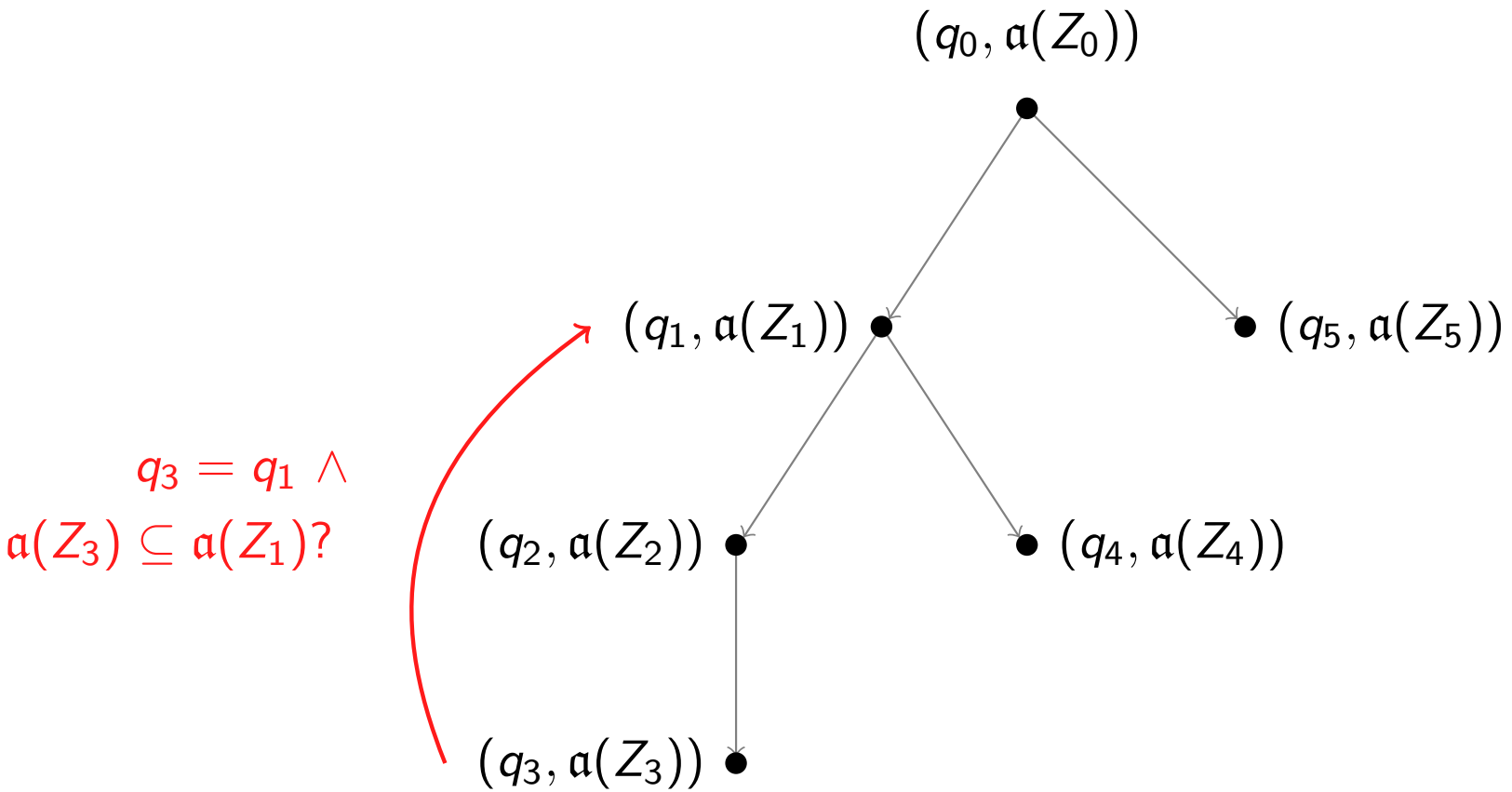


What is Closure_α?



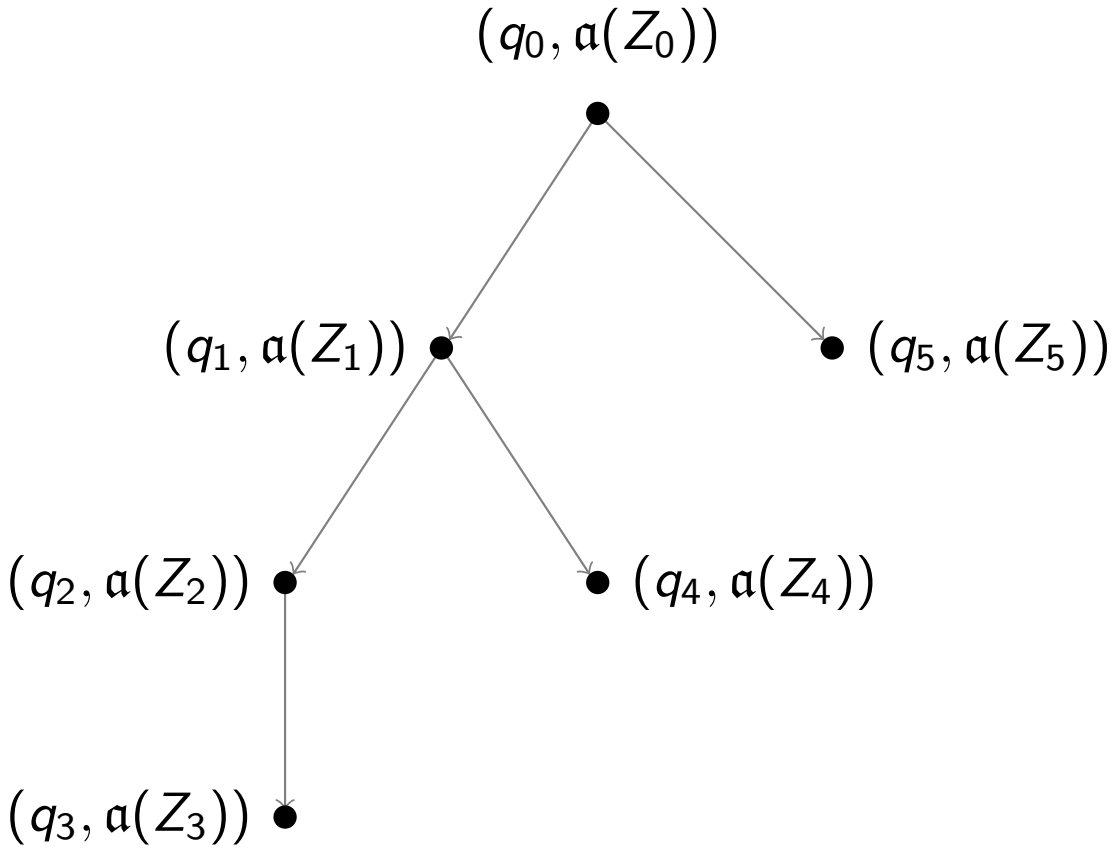
$\text{Closure}_\alpha(Z)$: union of regions that Z intersects

Using $\mathrm{Closure}_\alpha$ for reachability



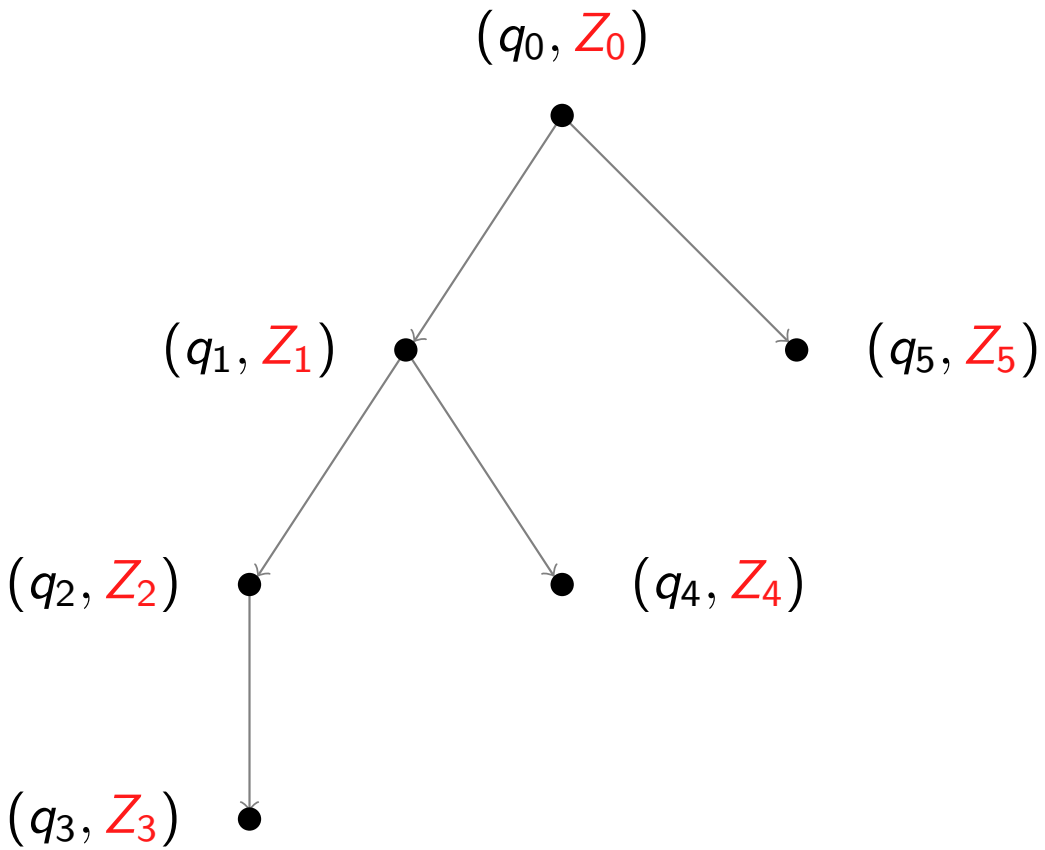
Standard algorithm: **covering tree**

Using $\mathrm{Closure}_\alpha$ for reachability



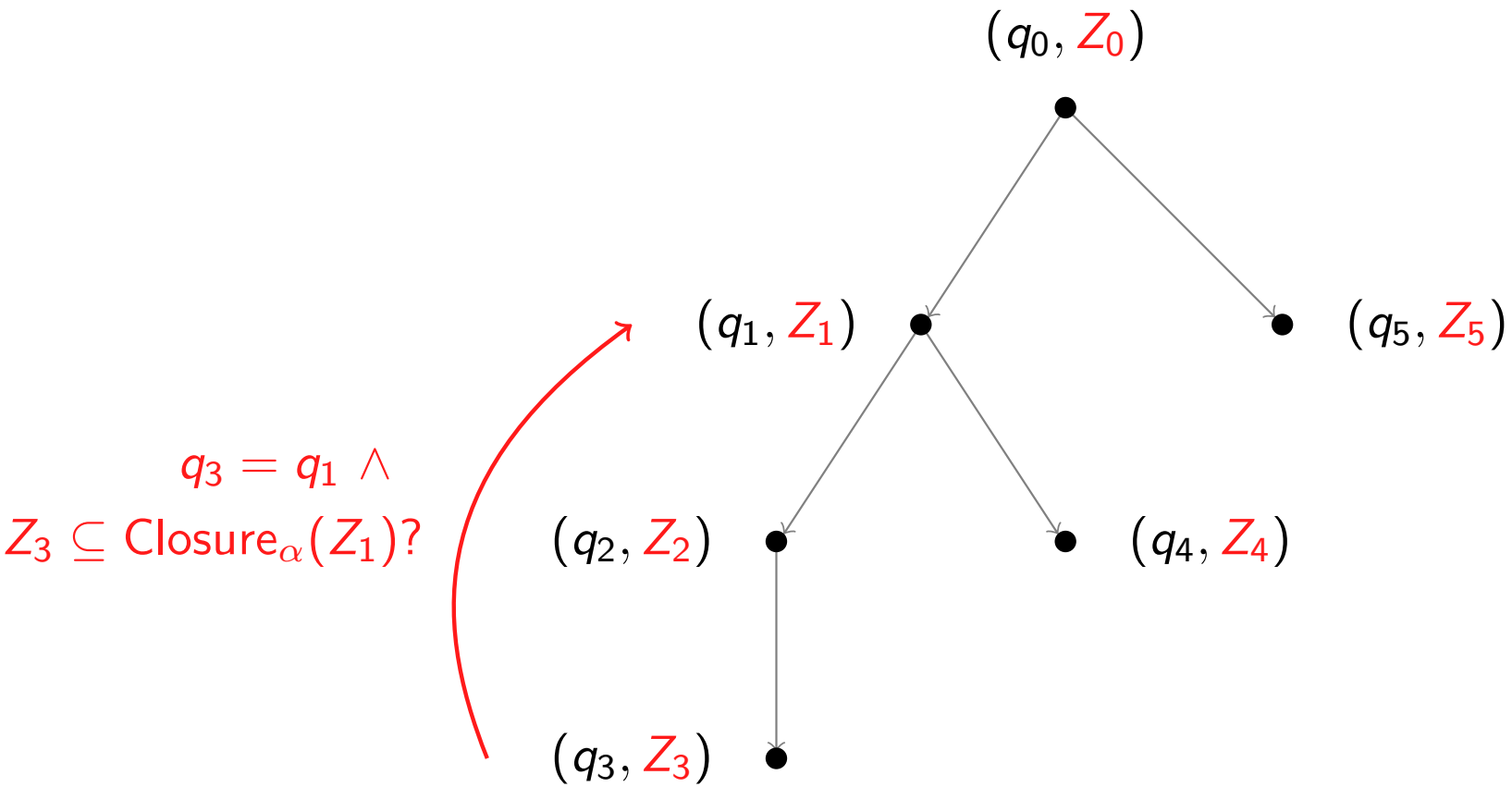
$\mathrm{Closure}_\alpha(Z)$ **cannot be** efficiently **stored**

Using Closure_α for reachability



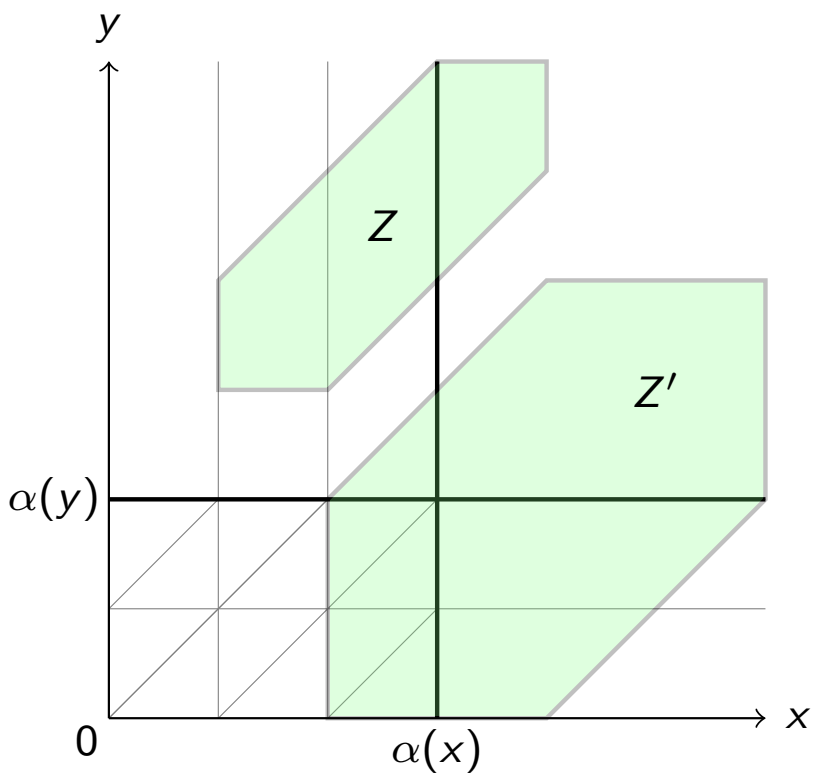
Do **not store** abstracted zones: use $\text{ZG}(A)$ instead of $\text{ZG}^{\alpha}(A)$!

Using $\mathrm{Closure}_\alpha$ for reachability



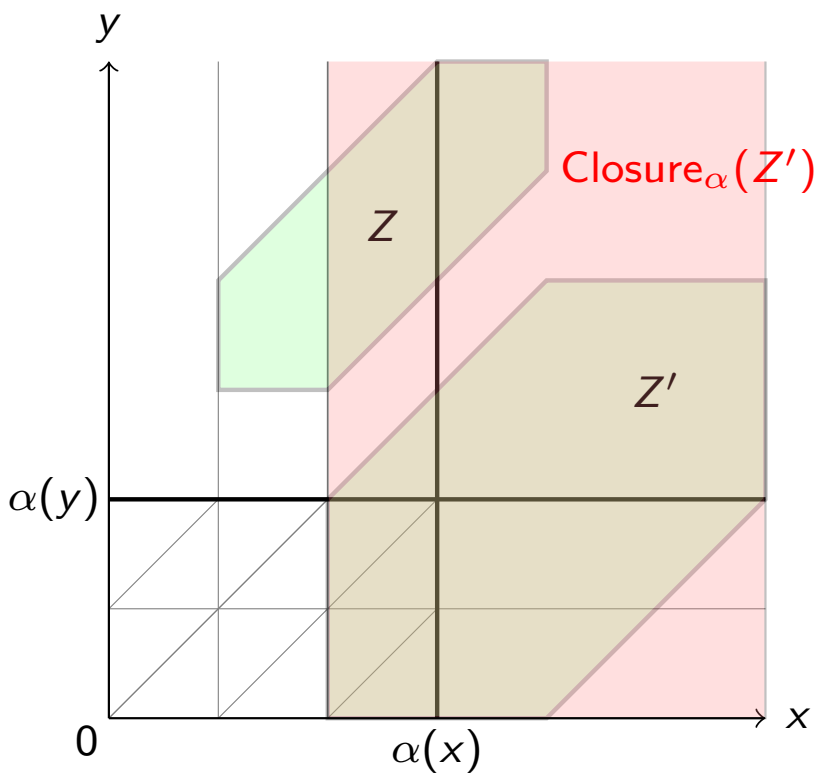
Use Closure for **termination**!

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



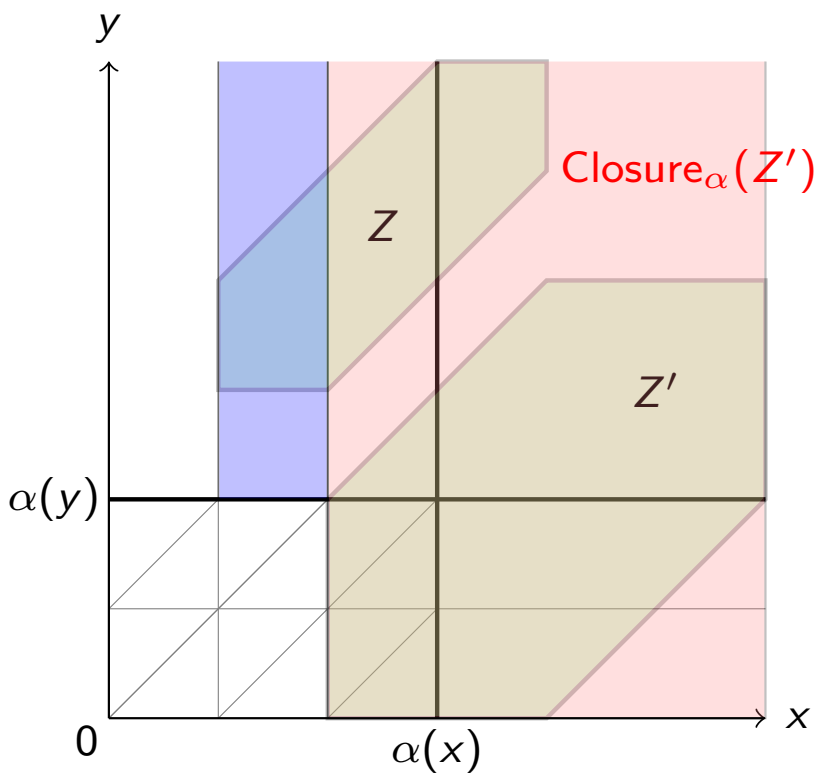
Recall: $\text{Closure}_\alpha(Z')$ is the union of **regions that intersect** Z'

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



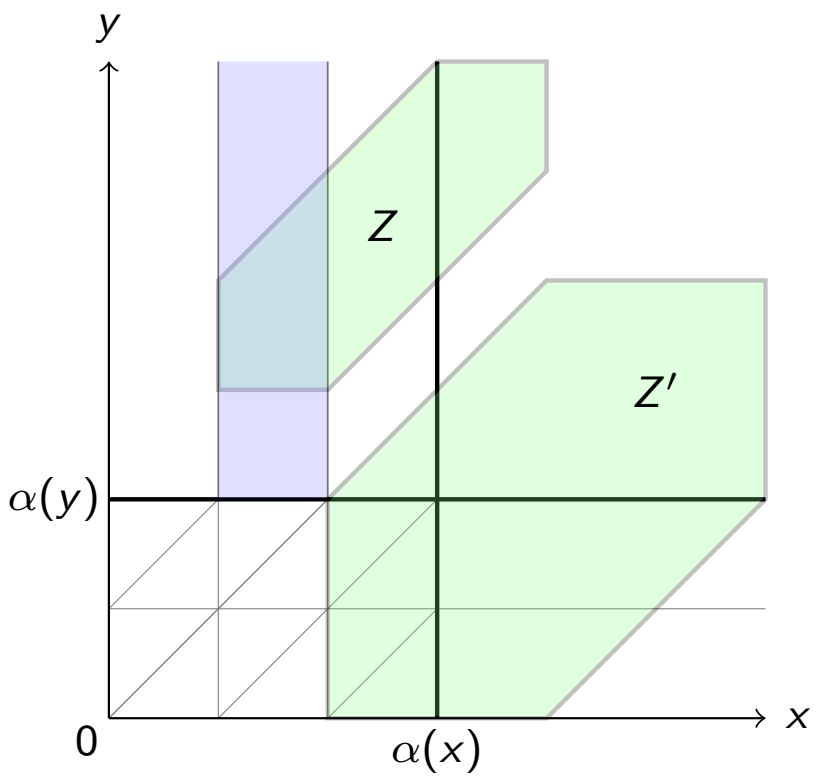
Recall: $\text{Closure}_\alpha(Z')$ is the union of **regions that intersect** Z'

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



Recall: $\text{Closure}_\alpha(Z')$ is the union of **regions that intersect** Z'

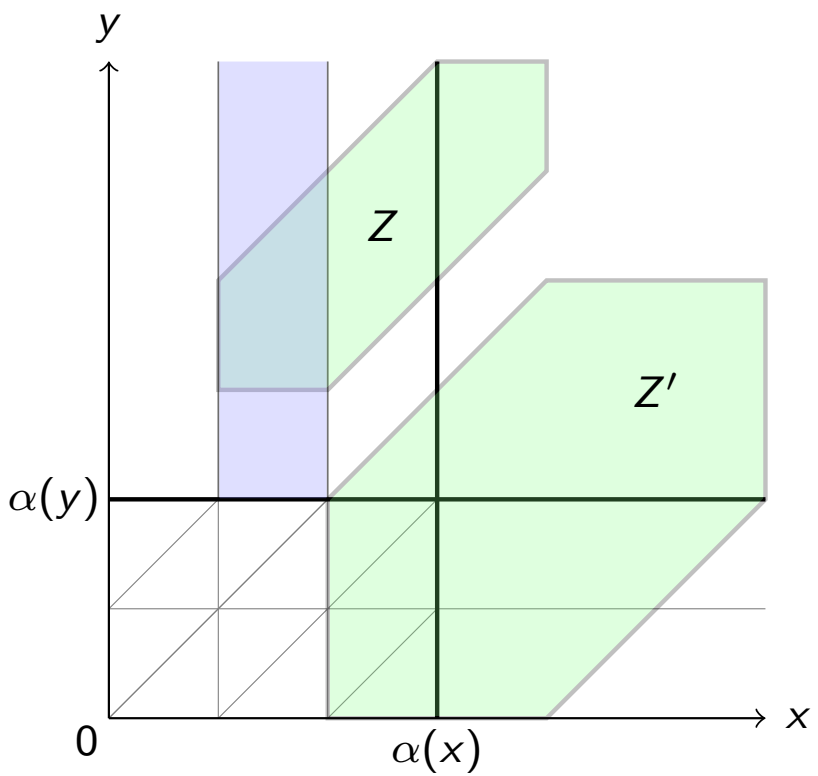
When is $Z \subseteq \text{Closure}_\alpha(Z')$?



Recall: $\text{Closure}_\alpha(Z')$ is the union of **regions that intersect** Z'

$Z \not\subseteq \text{Closure}_\alpha(Z')$ iff $\exists R.$ R **intersects Z**, but R does **not intersect Z′**

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



Recall: $\text{Closure}_\alpha(Z')$ is the union of **regions that intersect** Z'

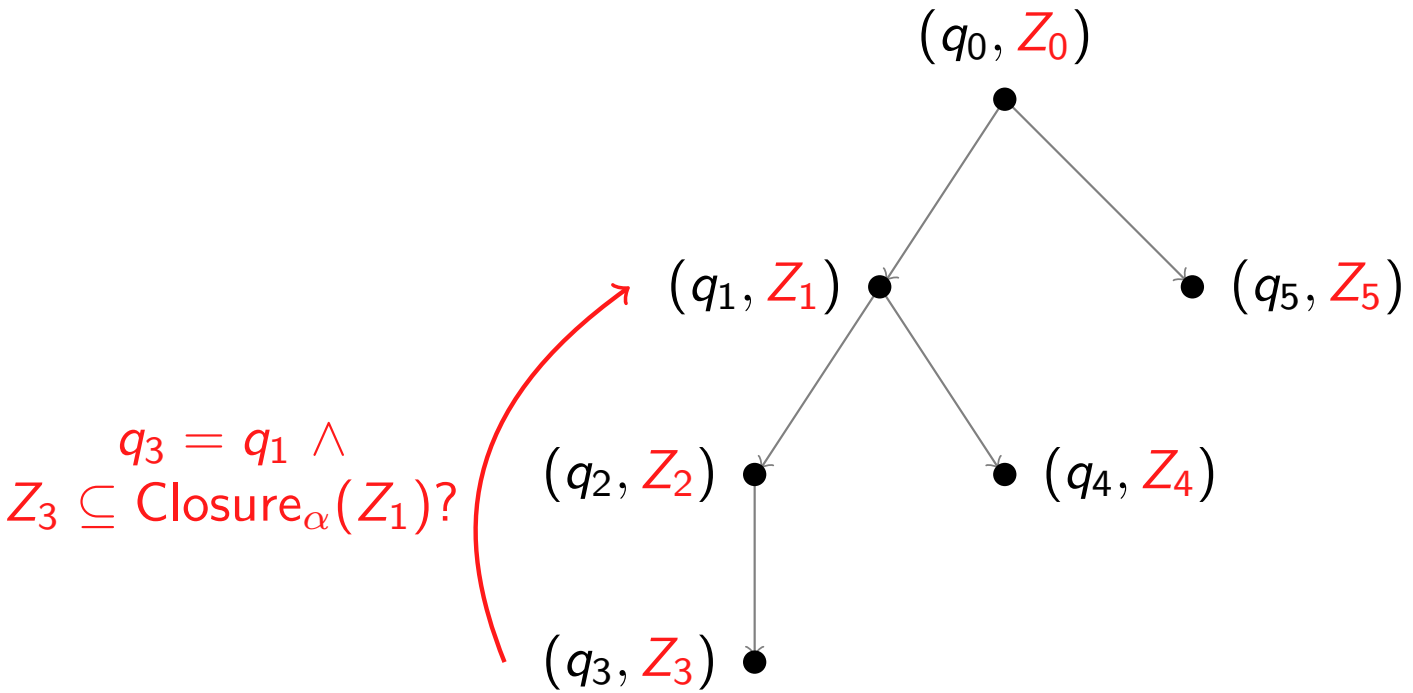
$Z \not\subseteq \text{Closure}_\alpha(Z')$ iff $\exists R.$ R **intersects Z**, but R does **not intersect Z′**

Theorem

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

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

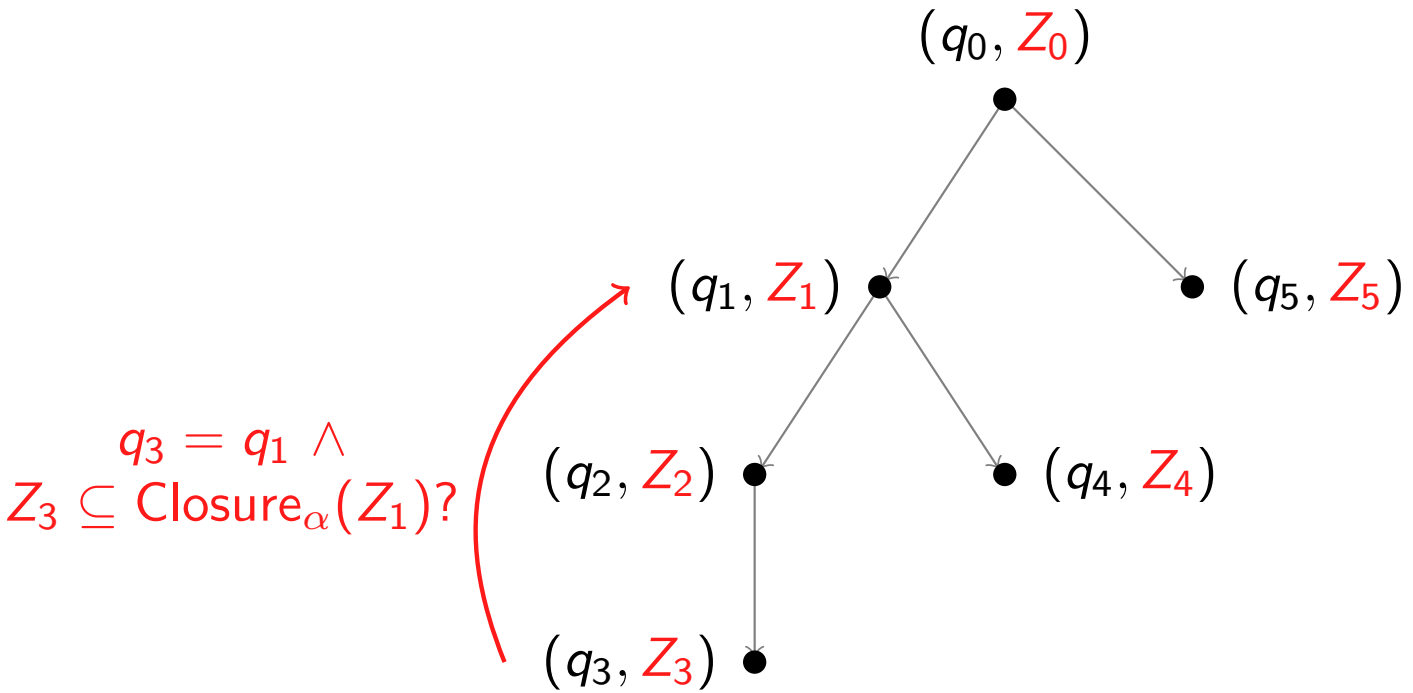
So what do we have now...



Theorem

$Z \subseteq \text{Closure}_\alpha(Z')$ can be decided in $\mathcal{O}(|X|^2)$ (same as $\subseteq$)

So what do we have now...



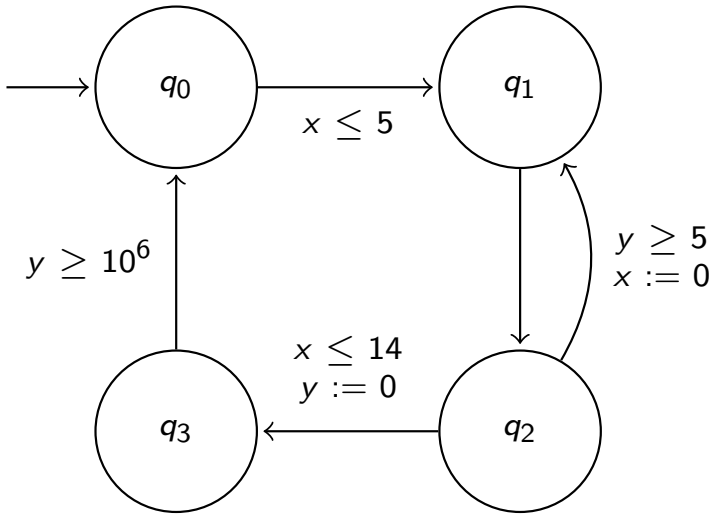
Theorem

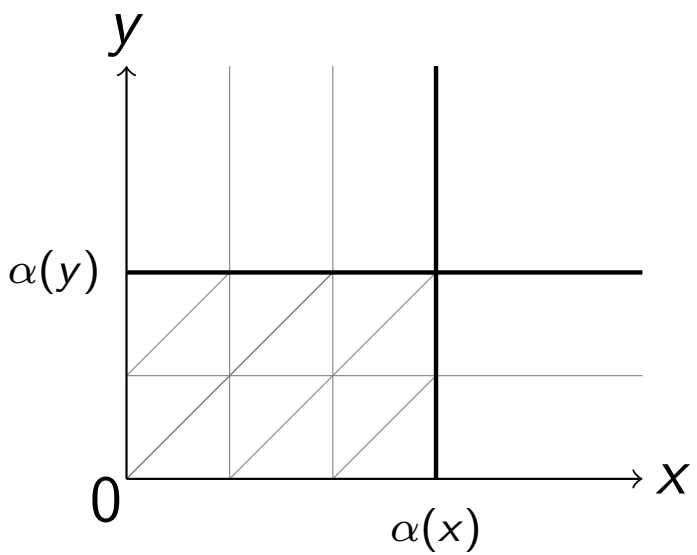
$Z \subseteq \text{Closure}_\alpha(Z')$ can be decided in $\mathcal{O}(|X|^2)$ (same as $\subseteq$)

Efficient algorithm to compute α?

Outline

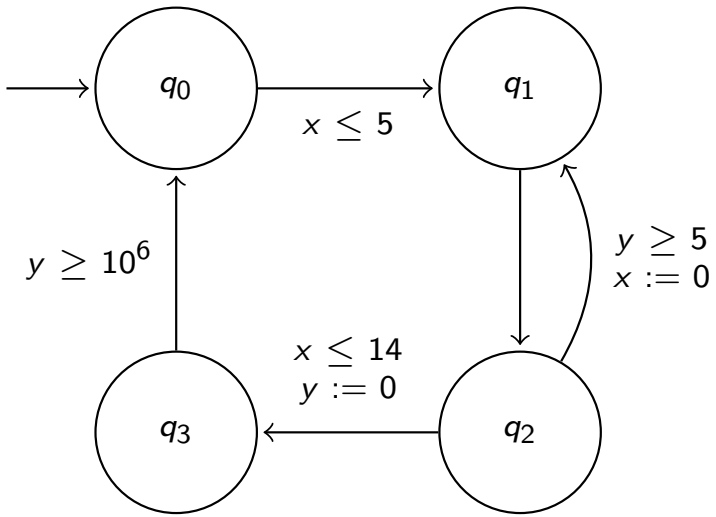
Bound function α





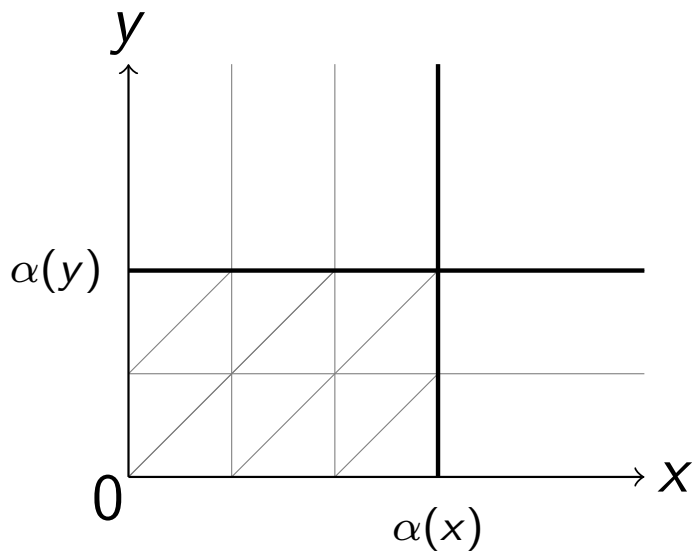
Naive: $\alpha(x) = 14$, $\alpha(y) = 10^6$. Yields a zone graph with $\sim$ **10^5 nodes**

Static analysis: α for every q [BBFL03]

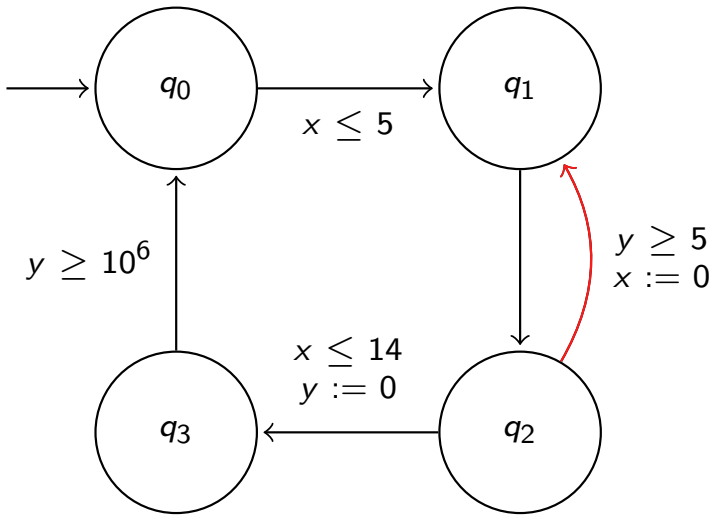


$\nu(y) = 10$

$\nu'(y) = 10^7$

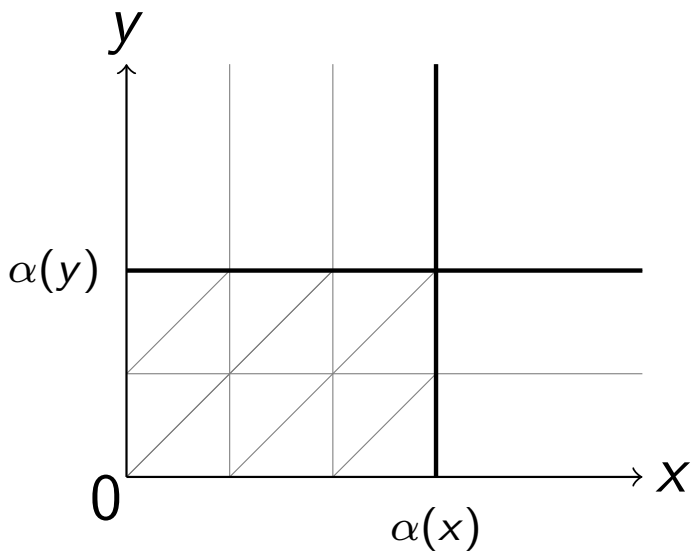


Static analysis: α for every q [BBFL03]

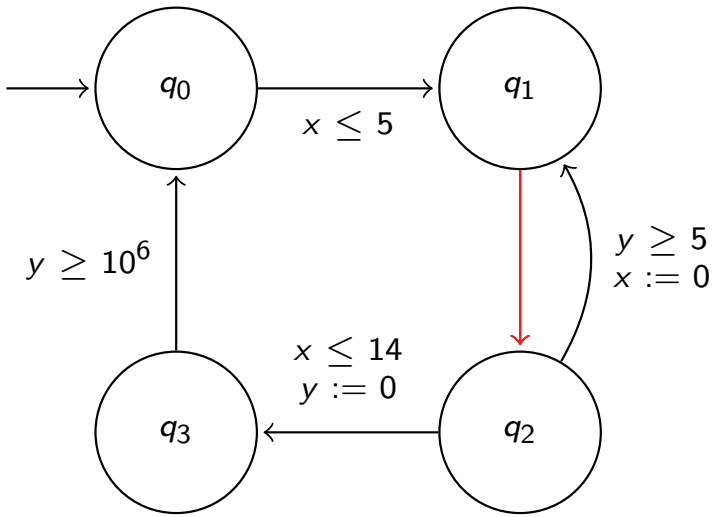


$\nu(y) = 10$

$\nu'(y) = 10^7$

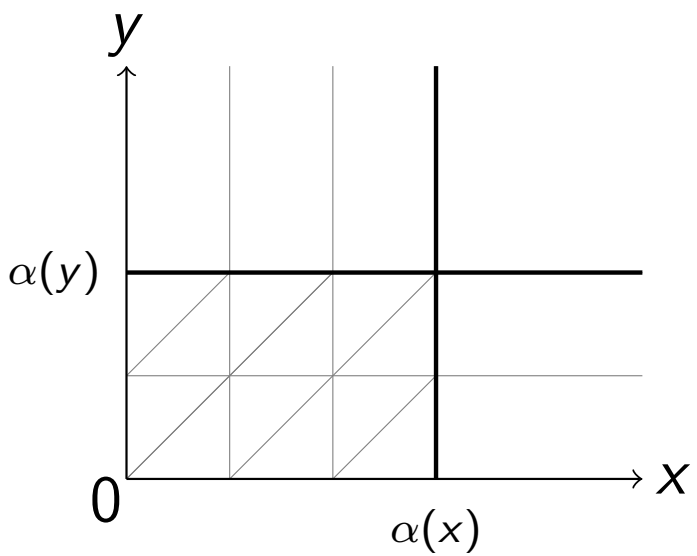


Static analysis: α for every q [BBFL03]

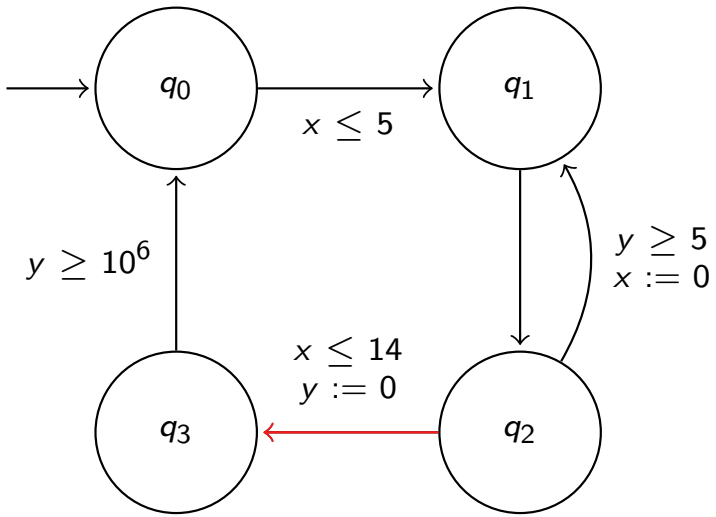


$\nu(y) = 10$

$\nu'(y) = 10^7$

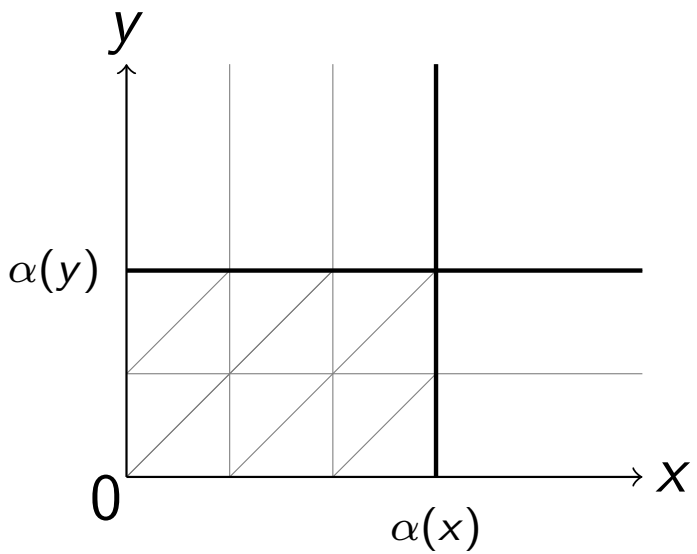


Static analysis: α for every q [BBFL03]

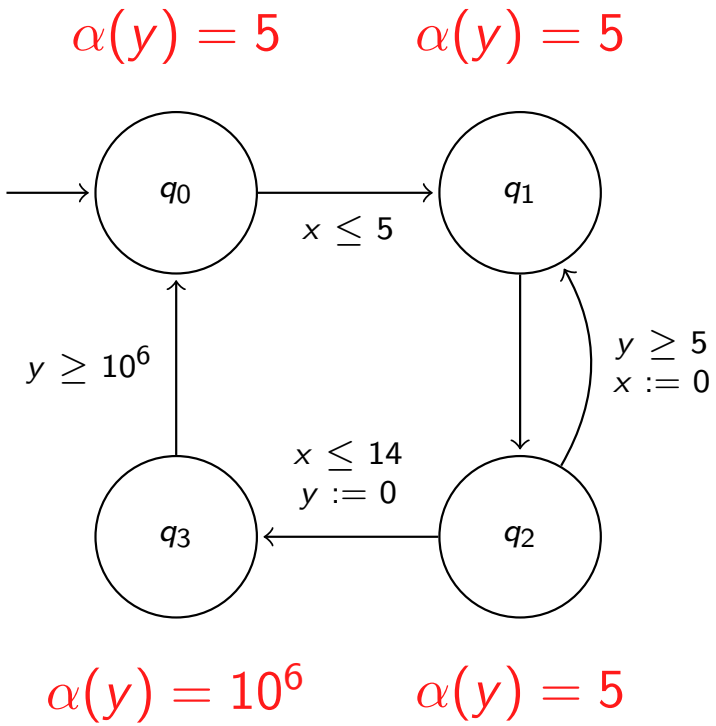


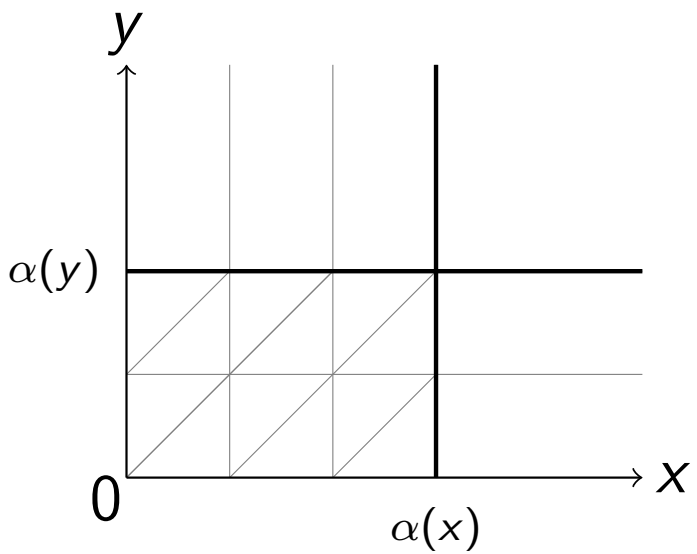
$\nu(y) = 10$

$\nu'(y) = 10^7$



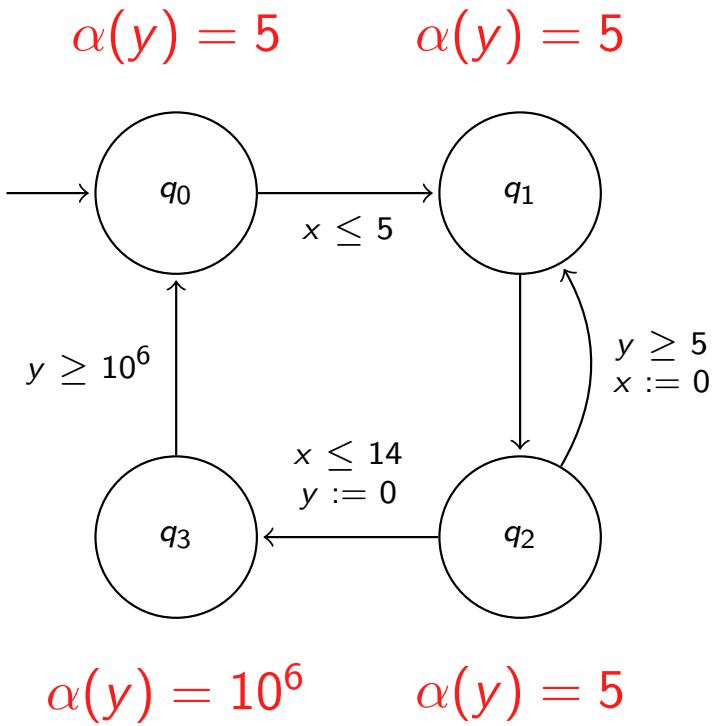
Static analysis: α for every q [BBFL03]

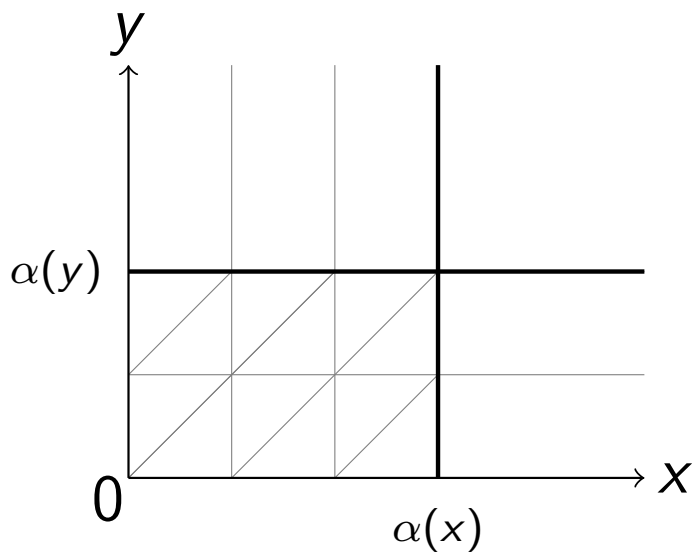




Each state has its own region partitioning

Static analysis: α for every q [BBFL03]

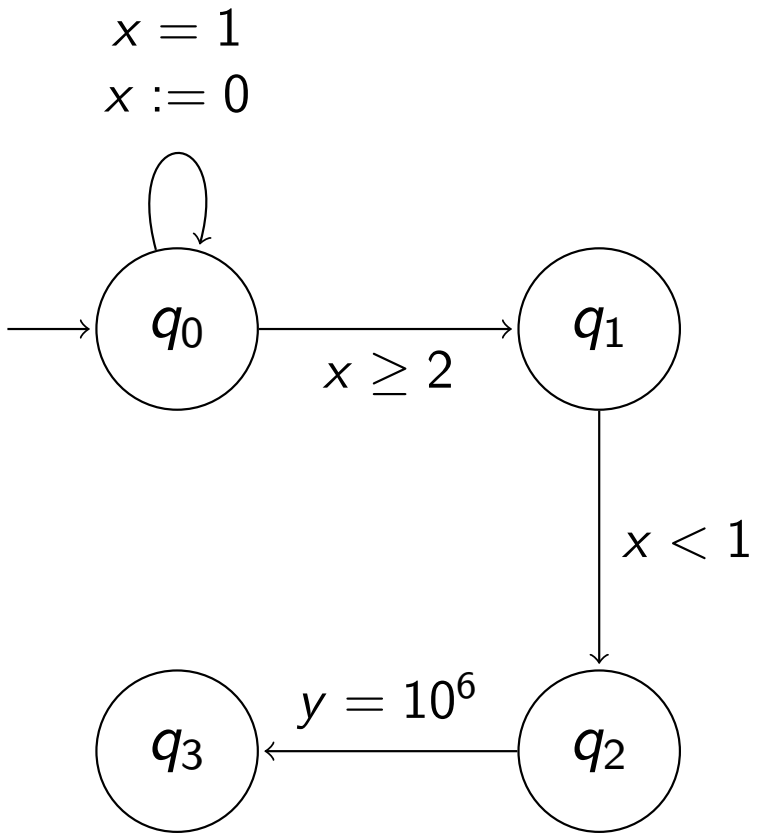




Each state has its own region partitioning

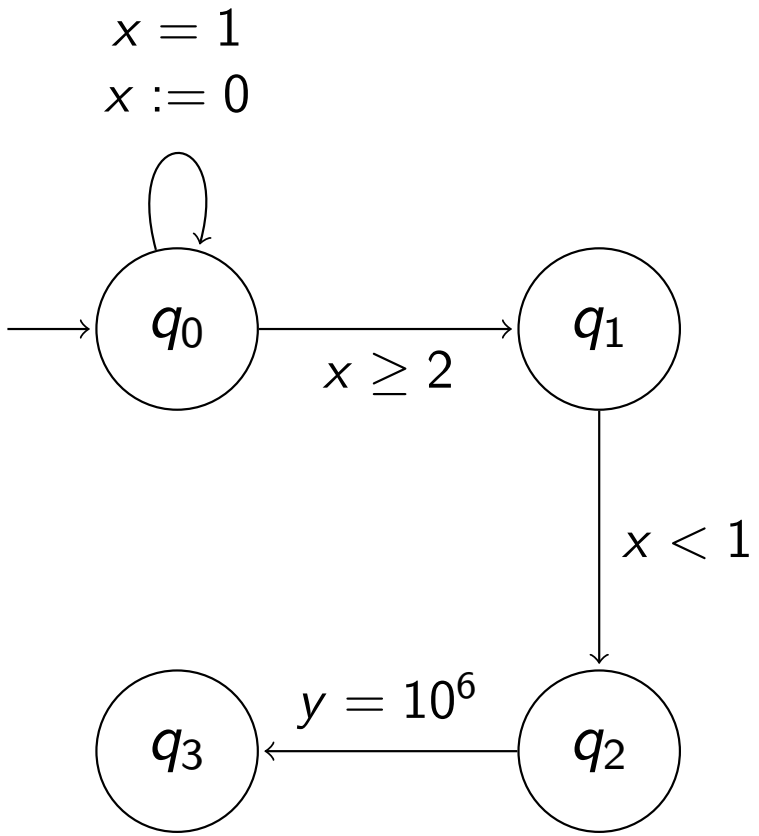
But this is not enough!

Need to look at semantics...



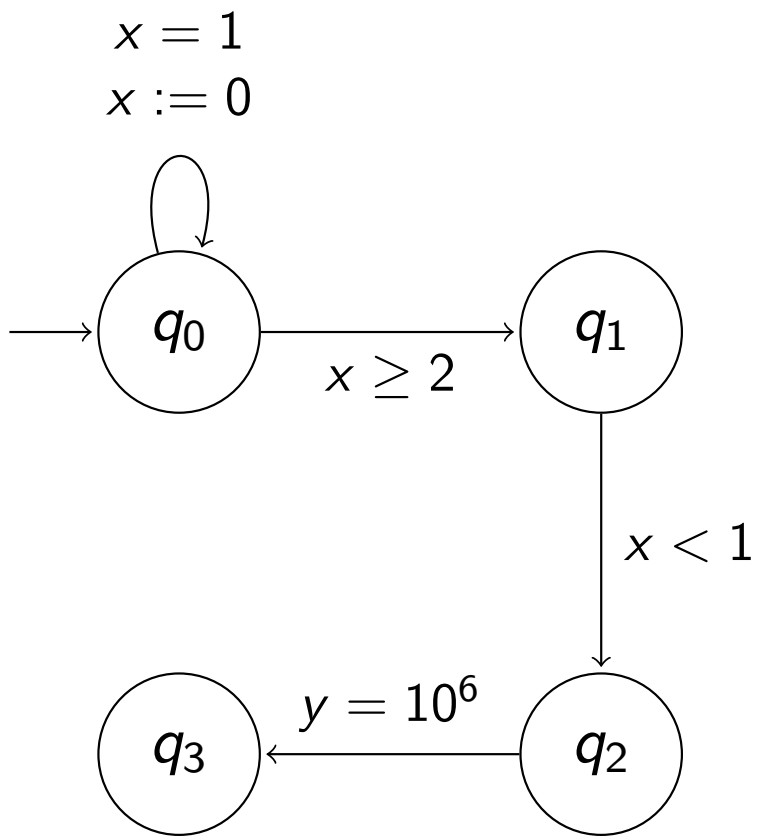
Need to look at semantics...

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



Need to look at semantics...

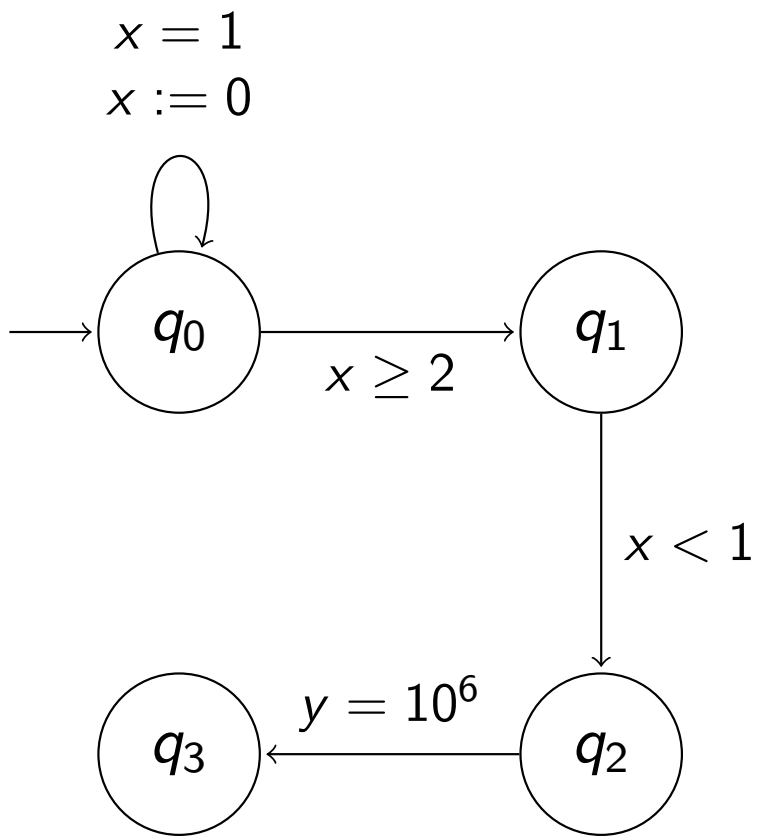
Static analysis: $\alpha(y) = 10^6$



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

Need to look at semantics...

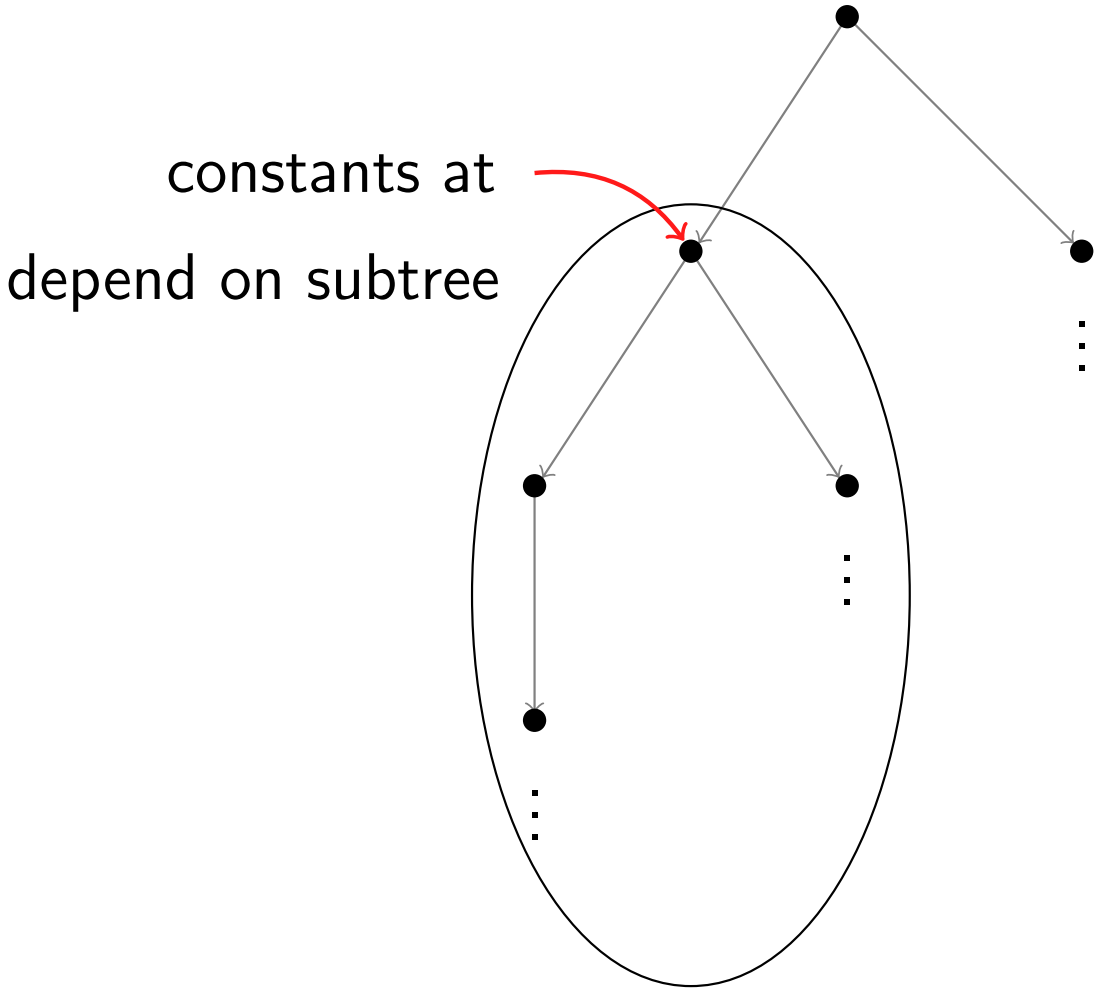
Static analysis: $\alpha(y) = 10^6$



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

Next: make **each node** have its own region partitioning!

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



Abstract zones are **not stored**: bounds can be computed **on-the-fly**

Constant propagation

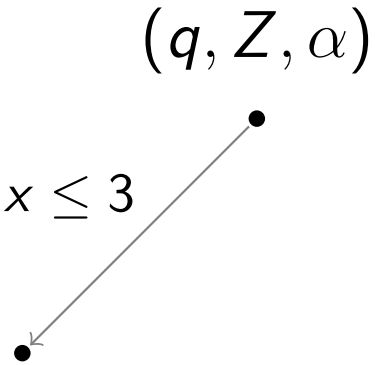
$\alpha(x) = -\infty$

(q, Z, α)

•

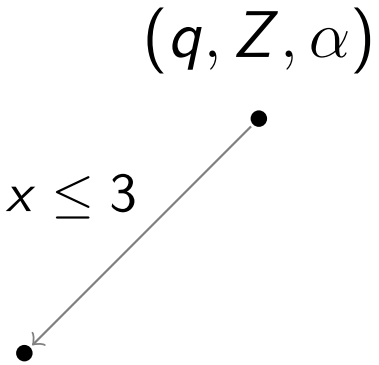
Constant propagation

$\alpha(x) = -\infty$



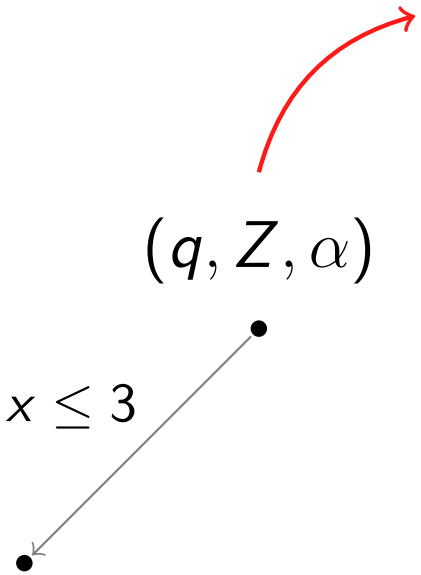
Constant propagation

$\alpha(x) = 3$



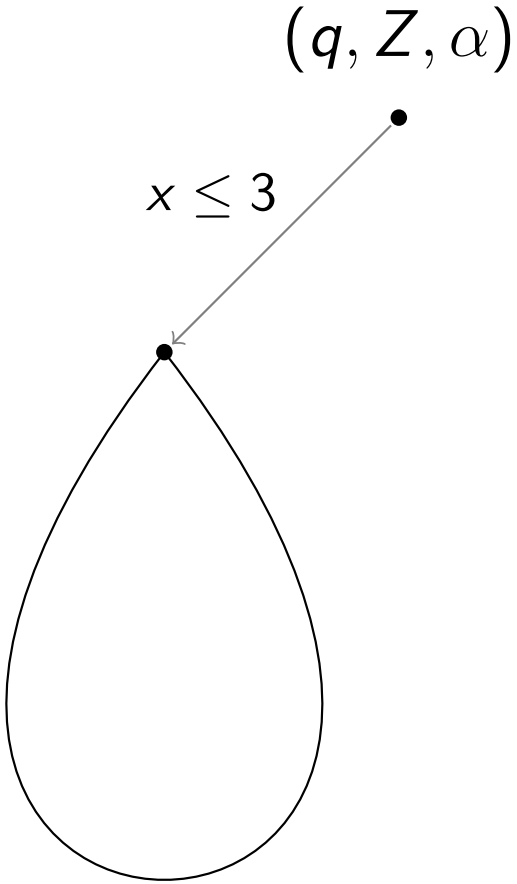
Constant propagation

$\alpha(x) = 3$



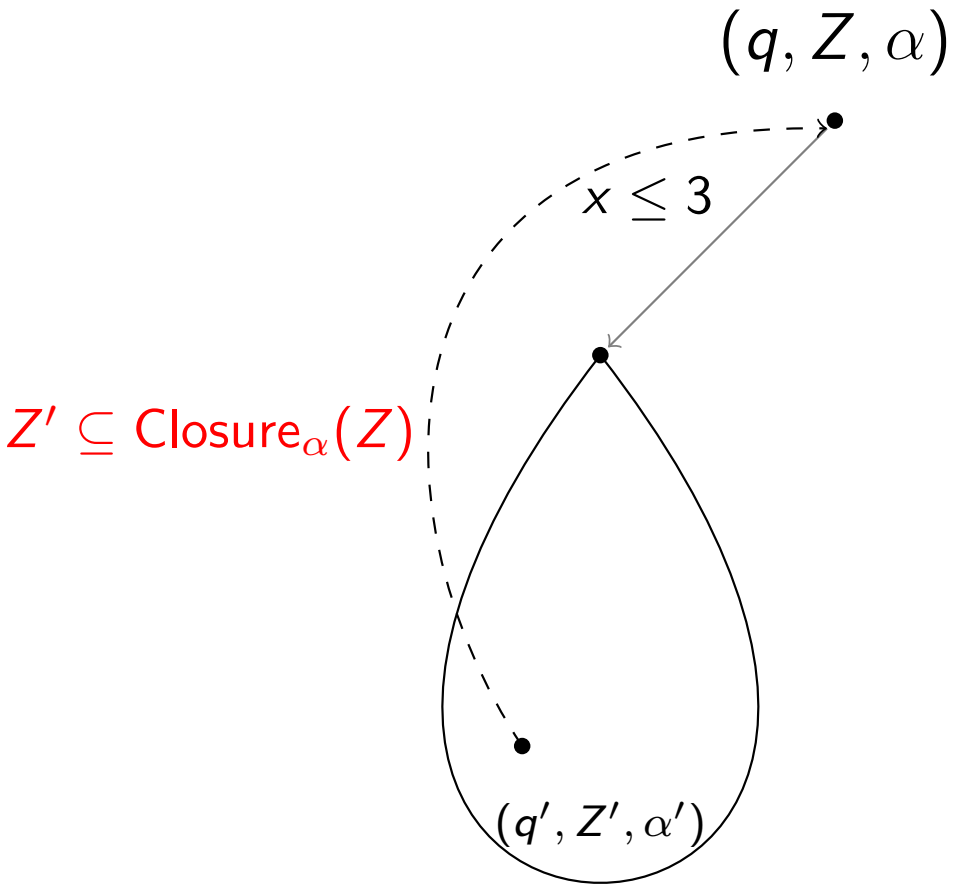
Constant propagation

$\alpha(x) = \quad 5$



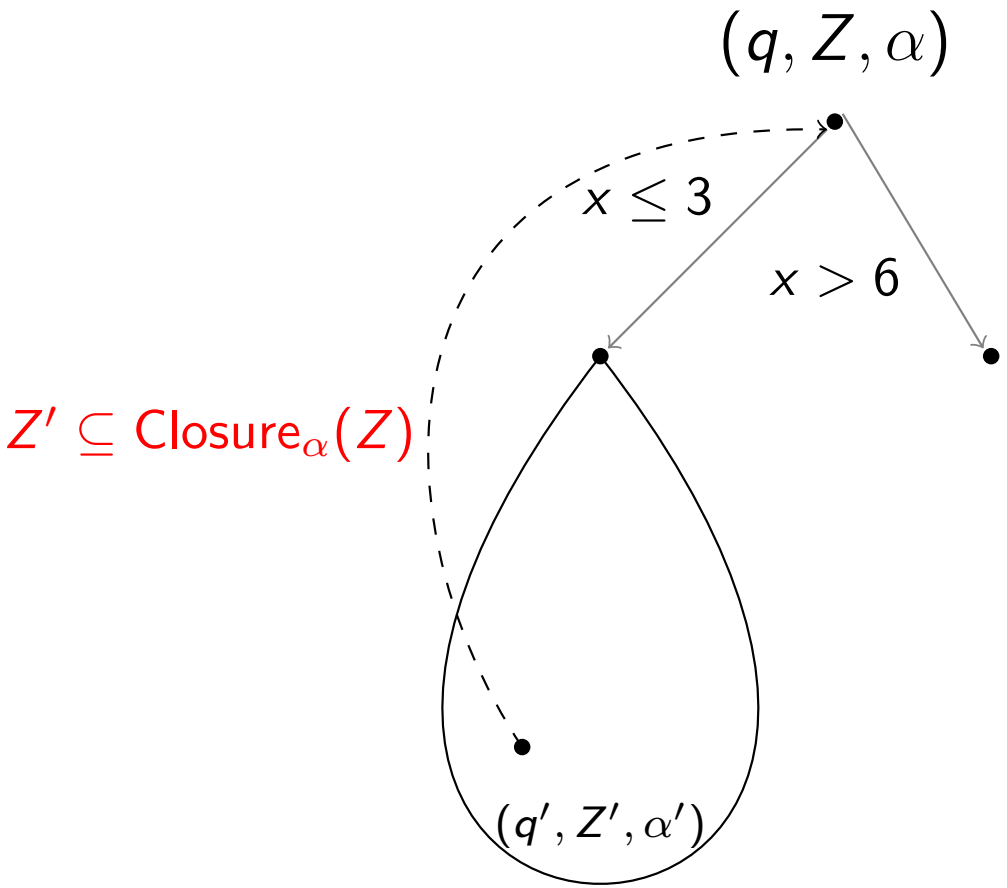
Constant propagation

$\alpha(x) = \quad 5$



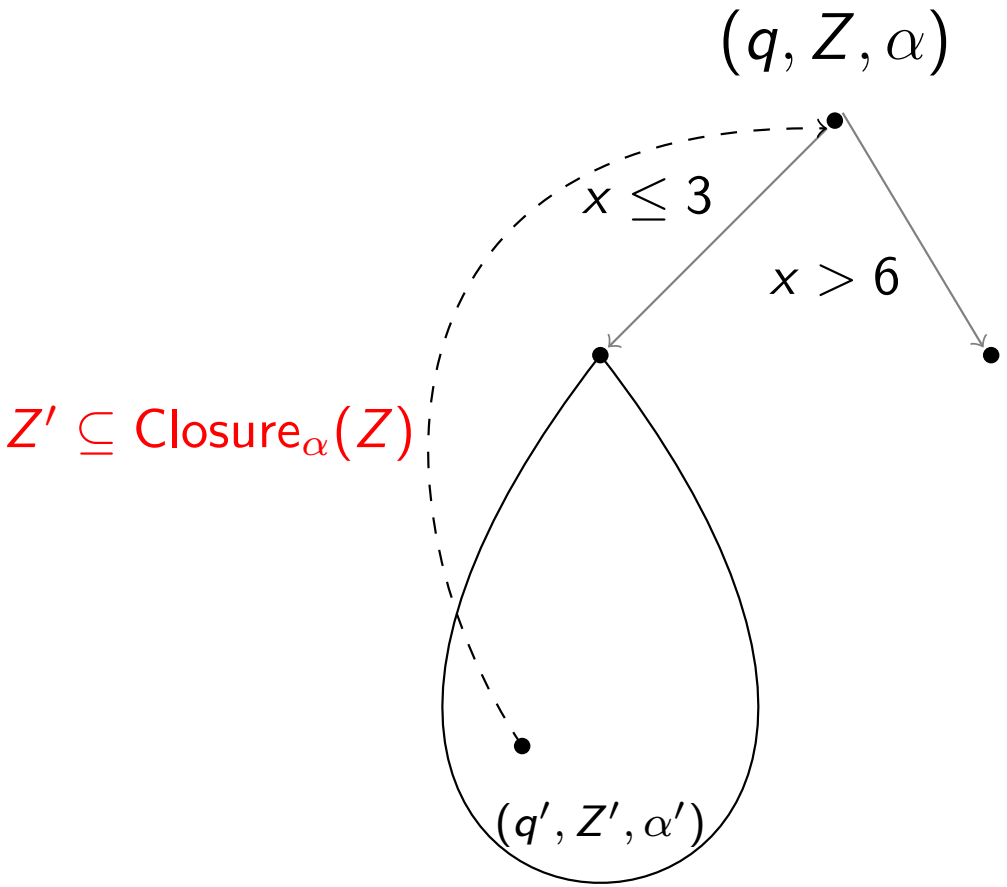
Constant propagation

$\alpha(x) = \quad 5$

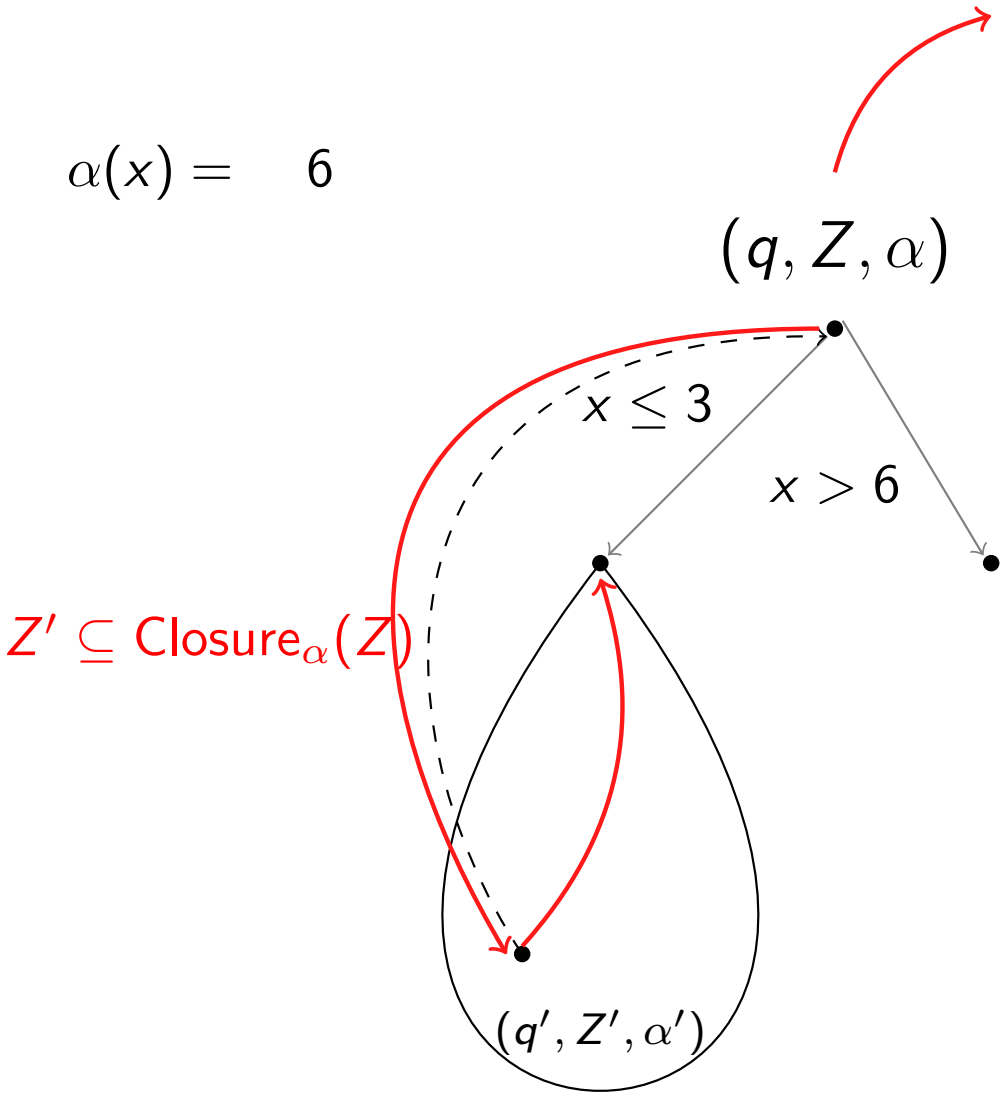


Constant propagation

$\alpha(x) = \quad 6$

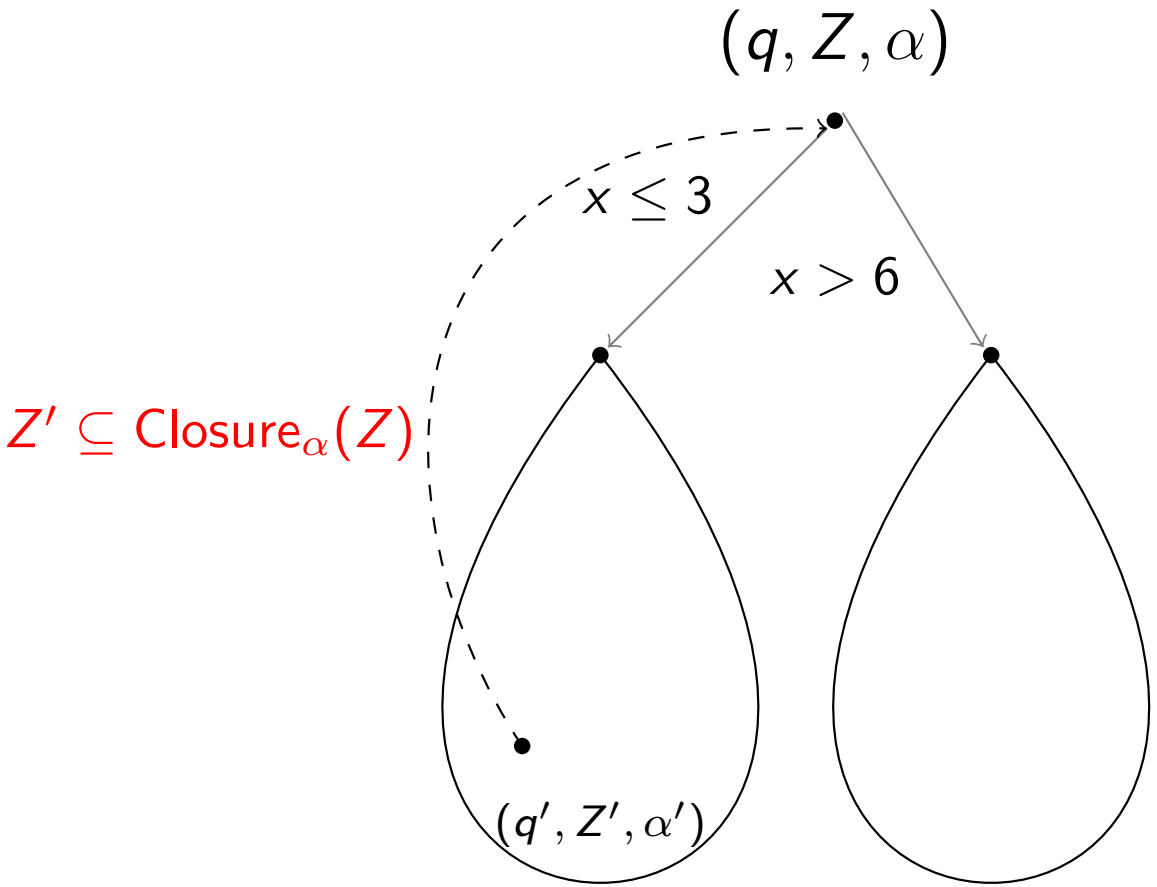


Constant propagation



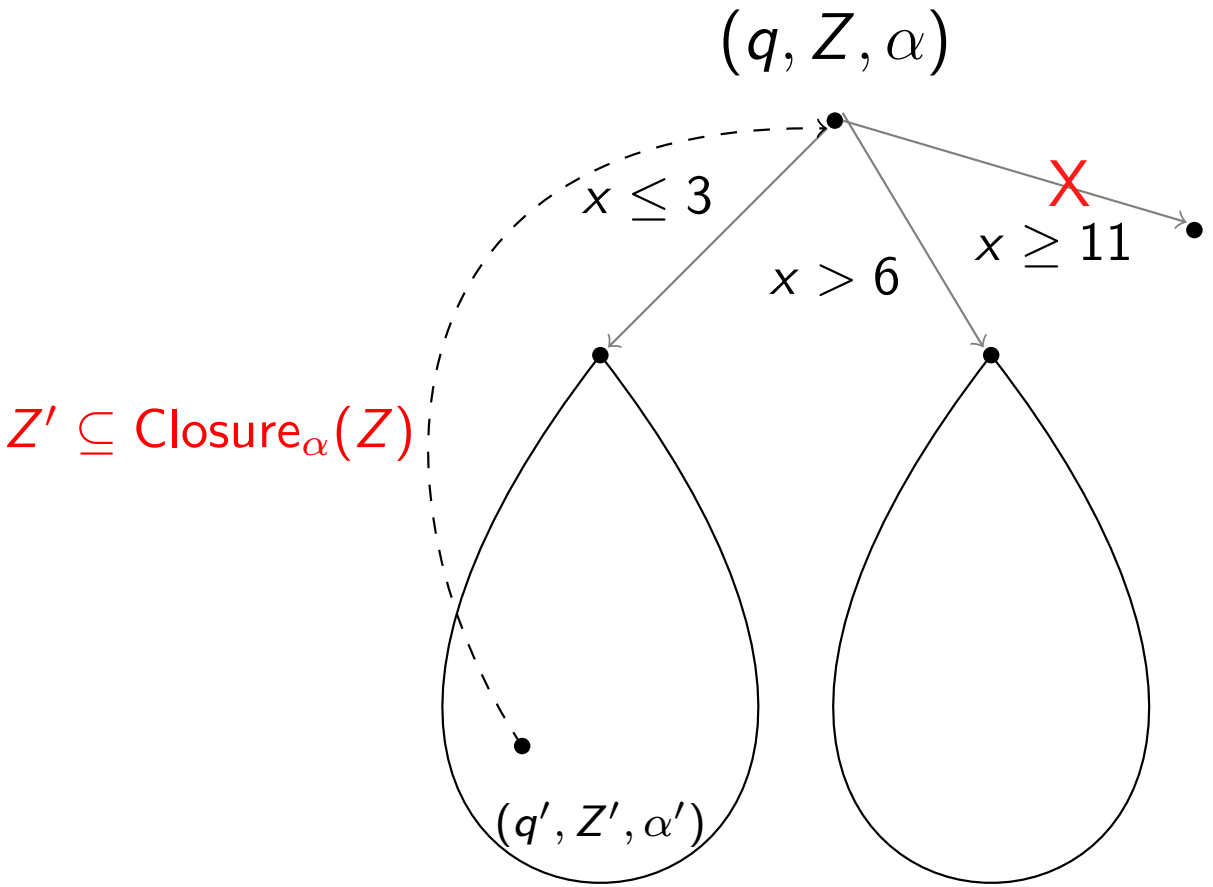
Constant propagation

$\alpha(x) = \quad 6$



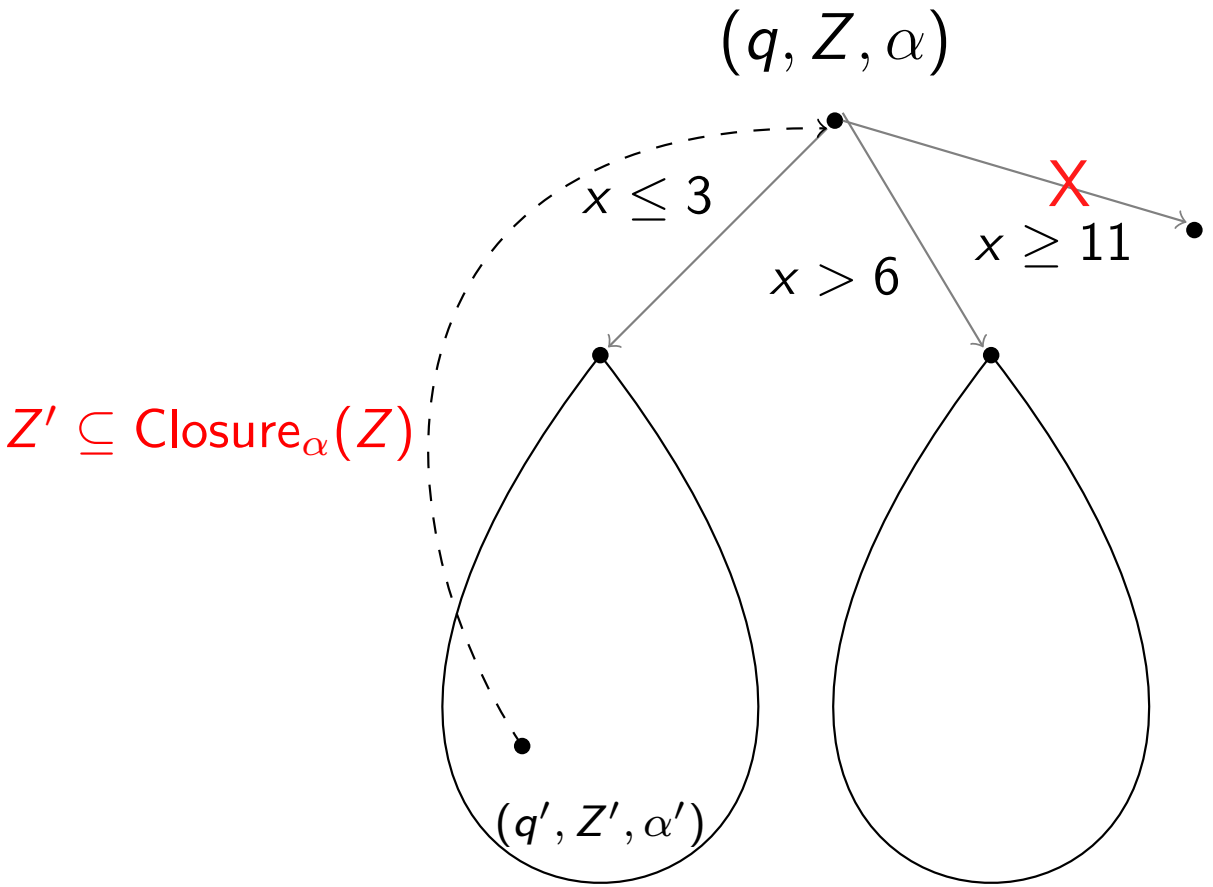
Constant propagation

$\alpha(x) = \quad 6$

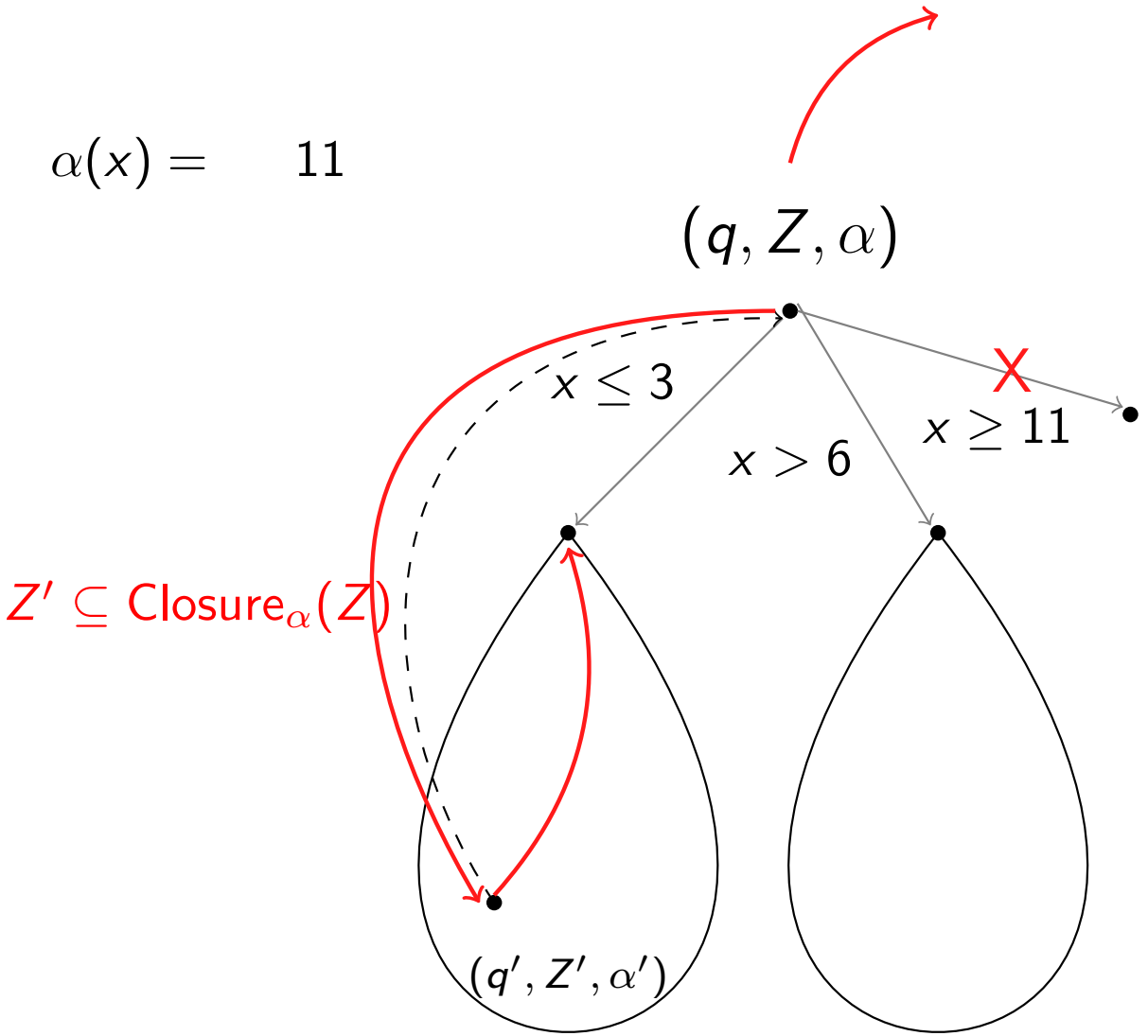


Constant propagation

$\alpha(x) = \quad 11$

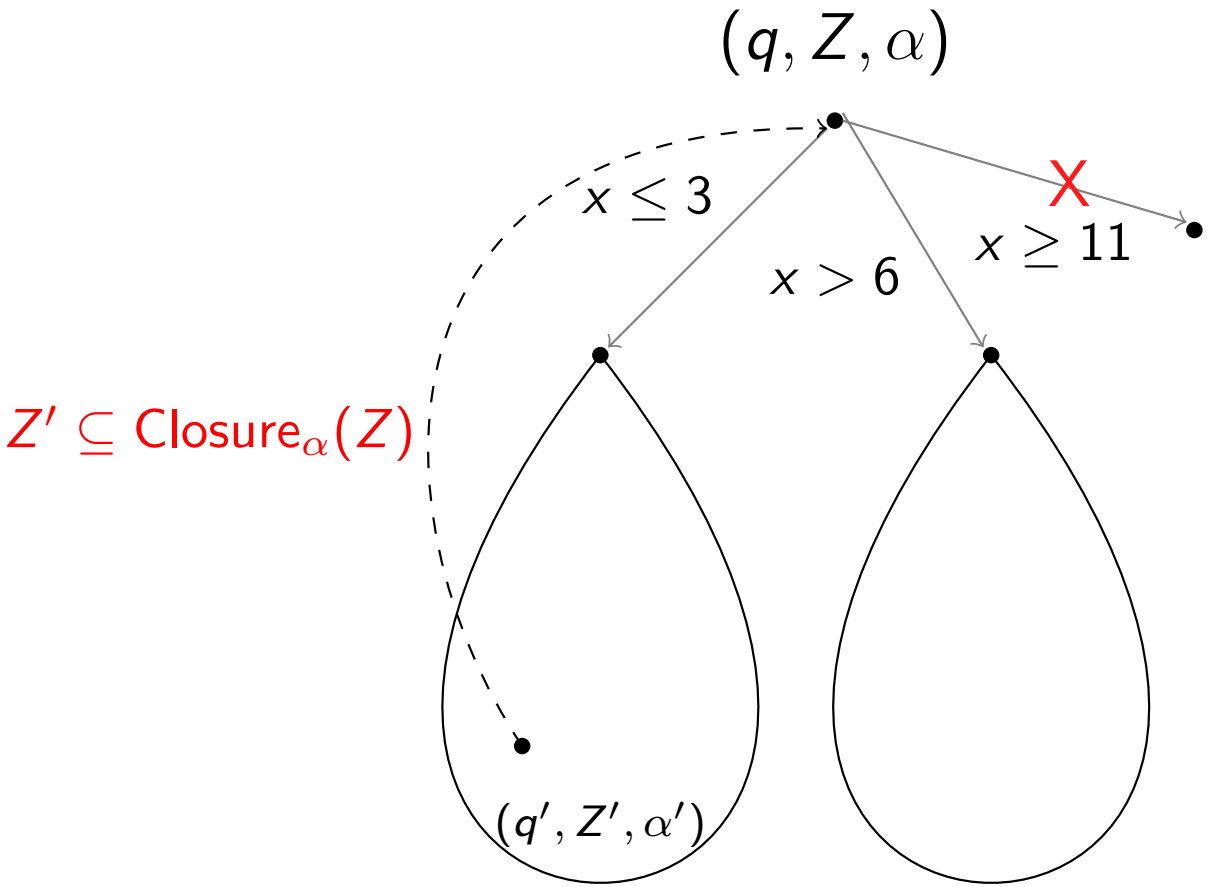


Constant propagation



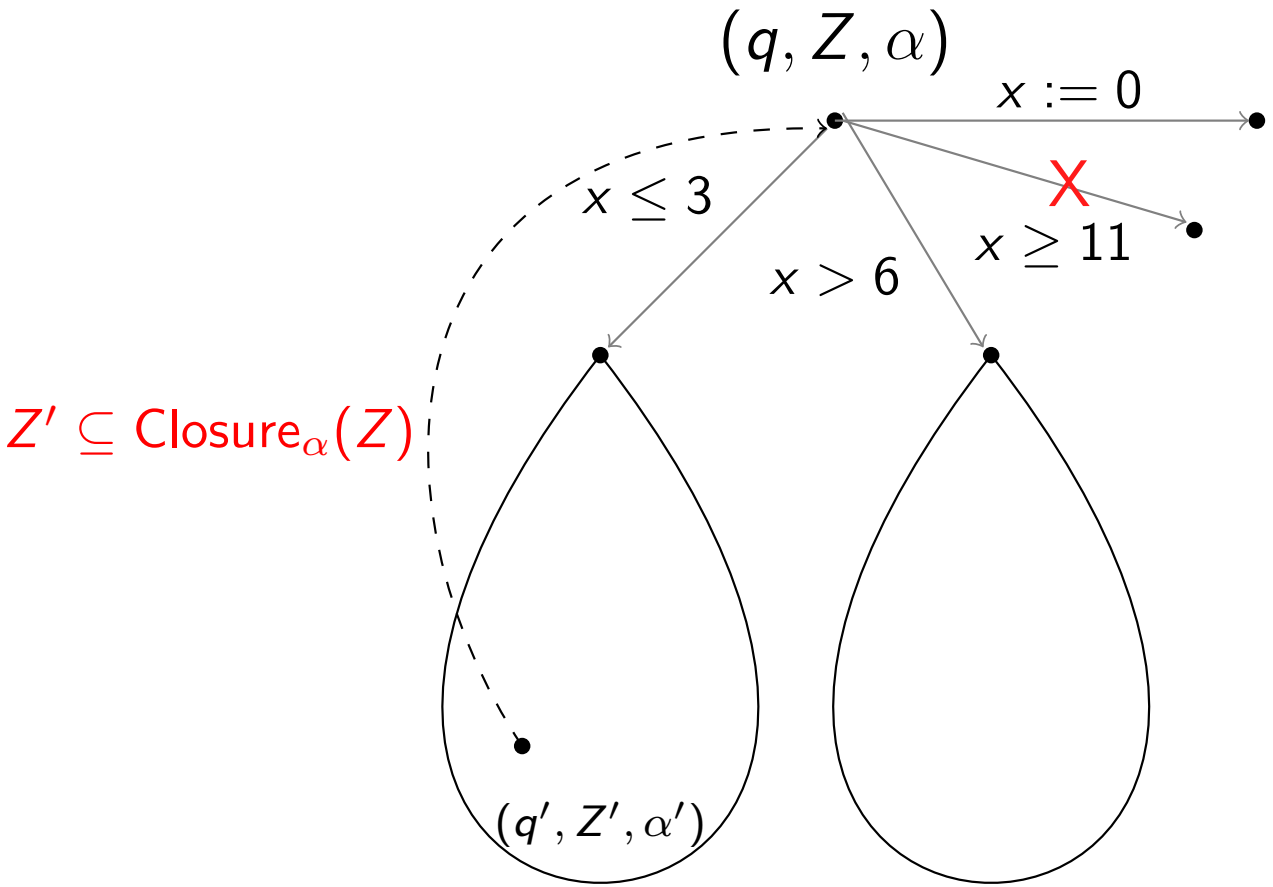
Constant propagation

$\alpha(x) = \quad 11$



Constant propagation

$\alpha(x) = \quad 11$



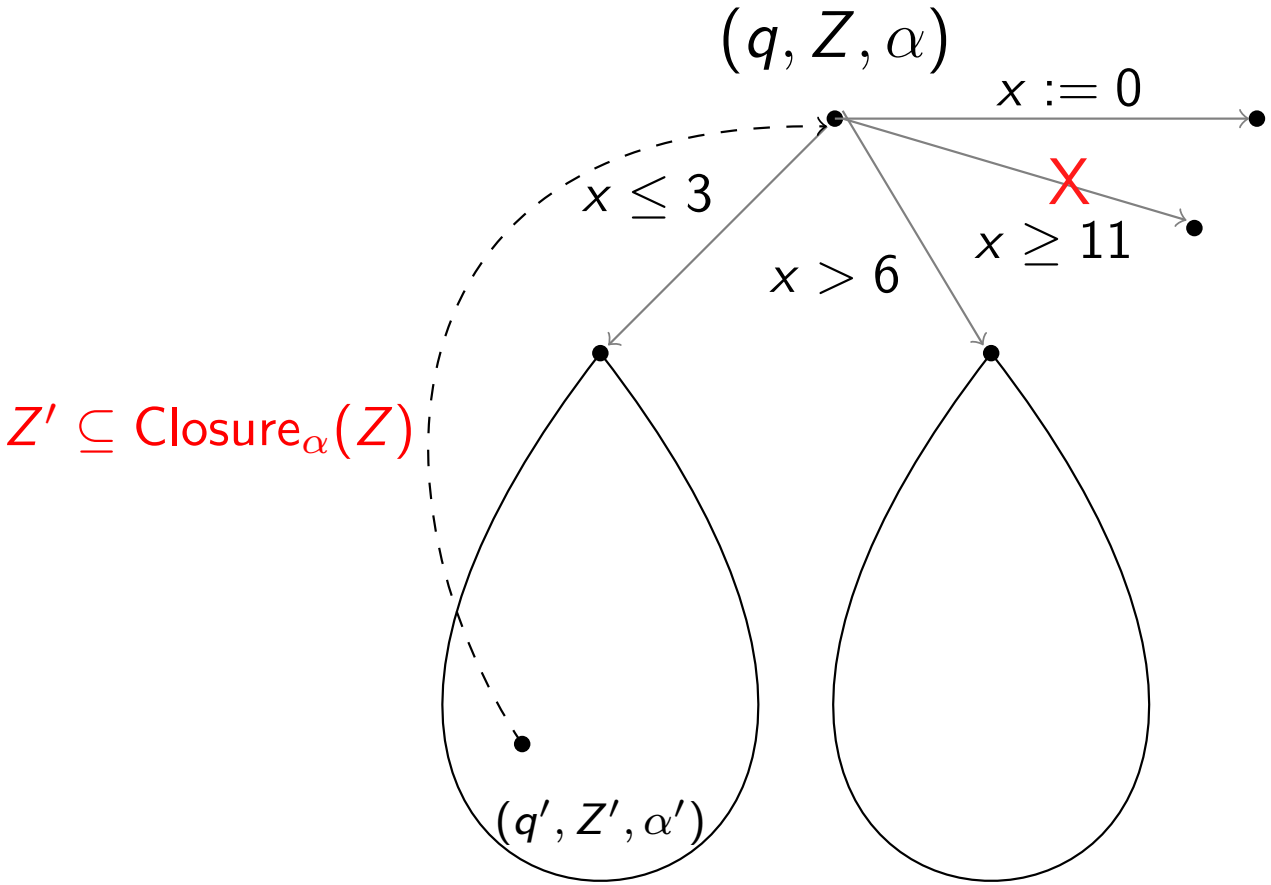
Constant propagation

$\alpha(x) = \quad 11$

All **tentative nodes** consistent

\+ No more **exploration**

$\rightarrow$ Terminate!



Invariants on the bounds

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

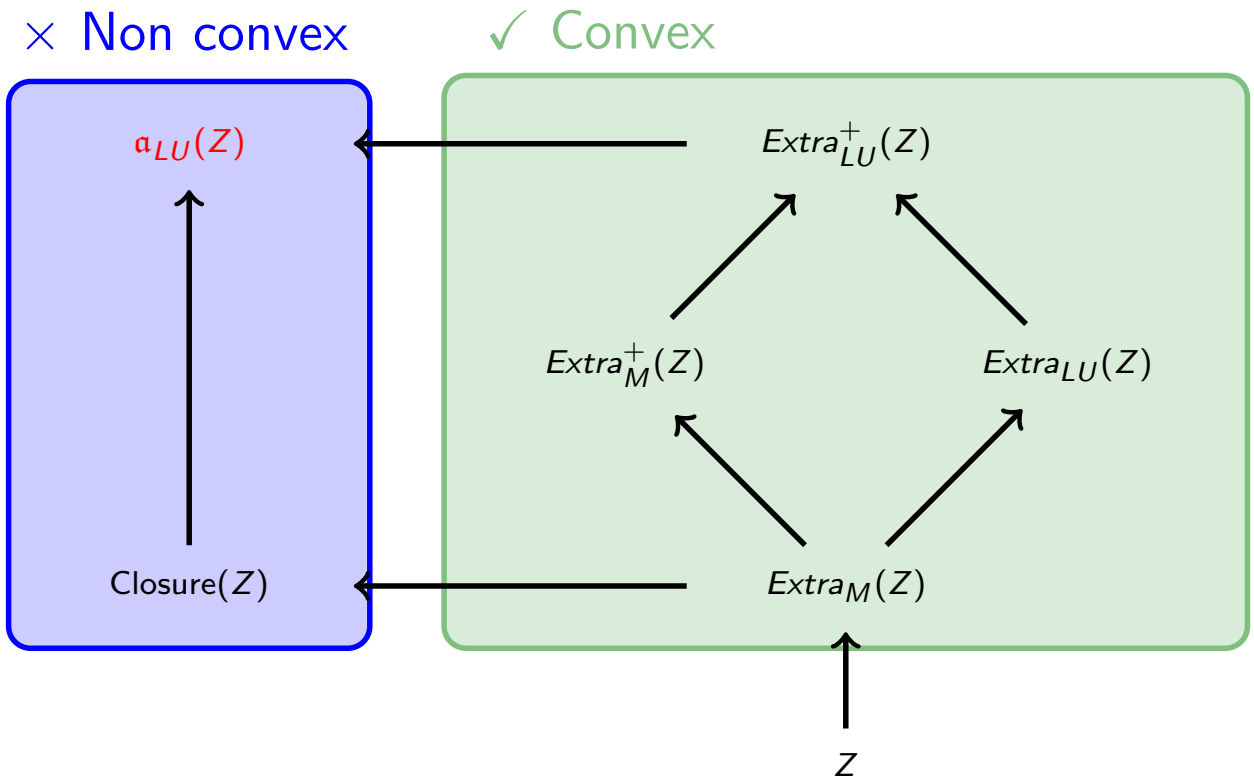
Invariants on the bounds

- **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



Theorem

An **efficient** $\mathcal{O}(|X|^2)$ procedure for $Z \subseteq \mathfrak{a}_{LU}(Z')$ (same as $\subseteq$)

Compute $ZG(\mathcal{A})$ and α **on-the-fly**, use $Z \subseteq \mathfrak{a}_{LU}(Z')$ for **termination**

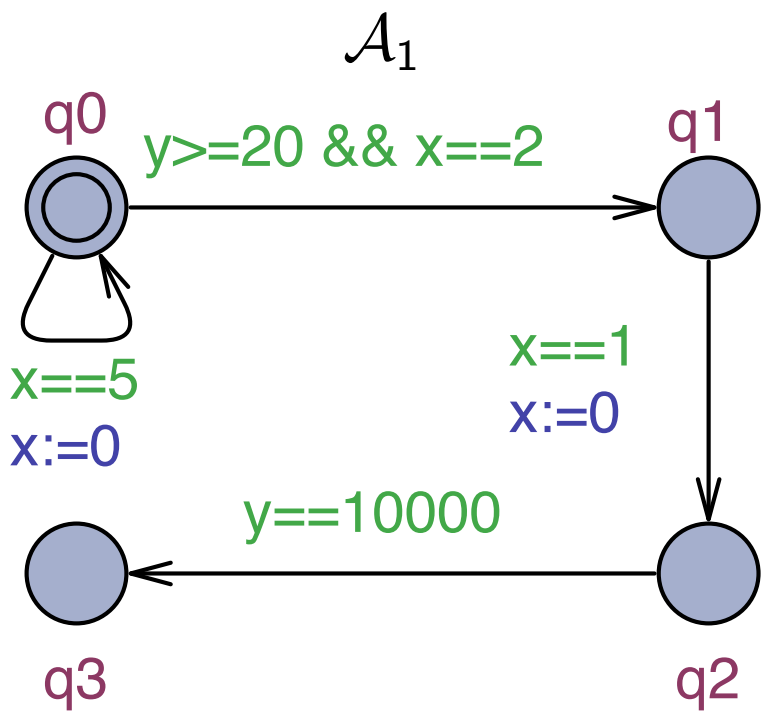
Benchmarks

Model	E^+_{LU},sa		a_{LU},sa		a_{LU},otf	
	nodes	s.	nodes	s.	nodes	s.
Fi7	20004	0.47	20020	0.50	7737	0.35
Fi8	91491	2.26	91532	2.56	25080	1.31
Fi9	419959	11.50	420280	14.63	81035	5.04
C7	5923	0.25	5923	0.27	5046	0.32
C8	19017	0.97	19017	1.13	16609	1.36
C9	60783	3.79	60783	5.01	54467	6.01
FD10	525	0.06	459	0.05	459	0.04
FD20	2045	0.69	1719	0.63	1719	0.35
FD30	4565	3.86	3779	3.41	3779	1.42

E^+_{LU}, sa is currently used in **UPPAAL**

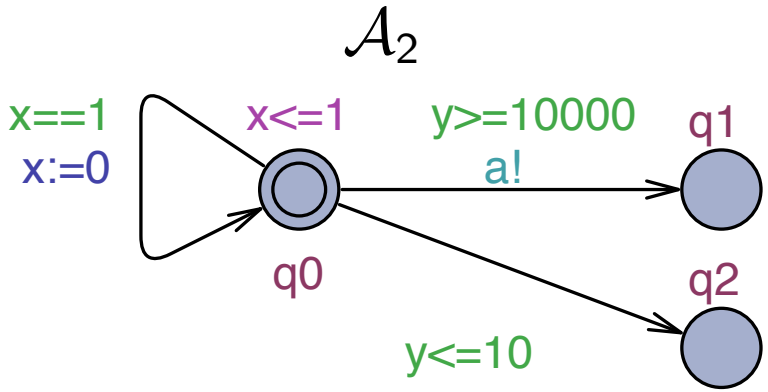
- Closure and $\mathfrak{a}_{LU}$ can be **efficiently implemented**
- **Both** Closure/$\mathfrak{a}_{LU}$ and otf bounds help

Experiments I



$\mathcal{A}_1$	nodes	s.
Extra^+_{LU}, **sa**	4001	6.16
Closure^+_{LU}, **sa**	4001	3.64
Closure^+_{LU}, **otf**	9	0.00

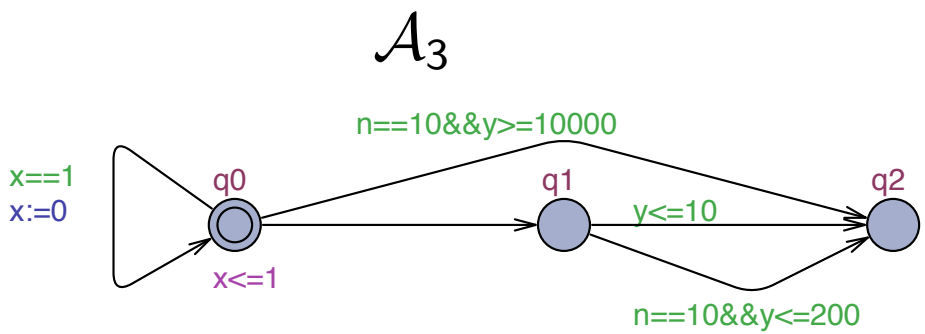
Experiments II



$\mathcal{A}_2$	nodes	s.
Extra^+_{LU}, **sa**	10014	95.62
Closure^+_{LU}, **sa**	10014	48.33
Closure^+_{LU}, **otf**	3	0.00

Illustrates the gain on the CSMA/CD example.

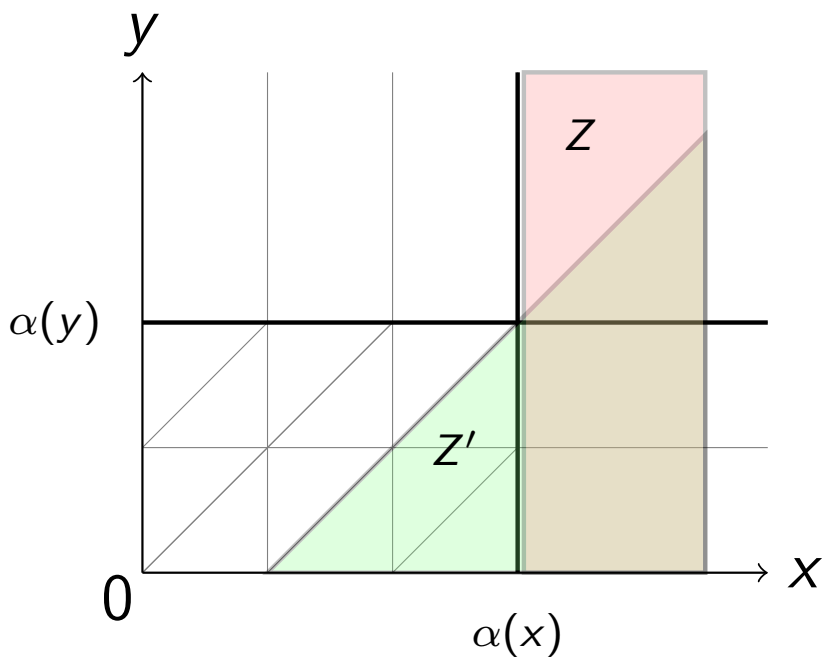
Experiments III



$\mathcal{A}_3$	nodes	s.
Extra^+_{LU}, **sa**	20006	99.26
Closure^+_{LU}, **sa**	20006	51.82
Closure^+_{LU}, **otf**	4	0.00

Illustrates the gain on the Fischer's protocol analysis.

Experiments IV



$$Z' : \ x - y \geq 1$$
$$Z : \ x > \alpha(x)$$

Illustrates the gain on the FDDI Example.

Conclusions & Perspectives

- **Efficient implementation** of a non-convex approximation that **subsumes** current ones in use
- **On-the-fly learning** of bounds that is **better** than the current static analysis

Conclusions & Perspectives

- **Efficient implementation** of a non-convex approximation that **subsumes** current ones in use
- **On-the-fly learning** of bounds that is **better** than the current static analysis

- More **sophisticated** abstraction parameters: **constraint** propagation, ...
- Adapt to **quantitative** properties (probabilities, quantities, etc)
- Handle **optimization** queries (shortest time, etc)

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.

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.

S. Tripakis and S. Yovine.

Analysis of timed systems using time-abstracting bisimulations.
Form. Methods Syst. Des., 18:25–68, 2001.