On the Relationship between Reachability Problems in Timed and Counter Automata

Christoph Haase[1,2] Joël Ouaknine[2] James Worrell[2]

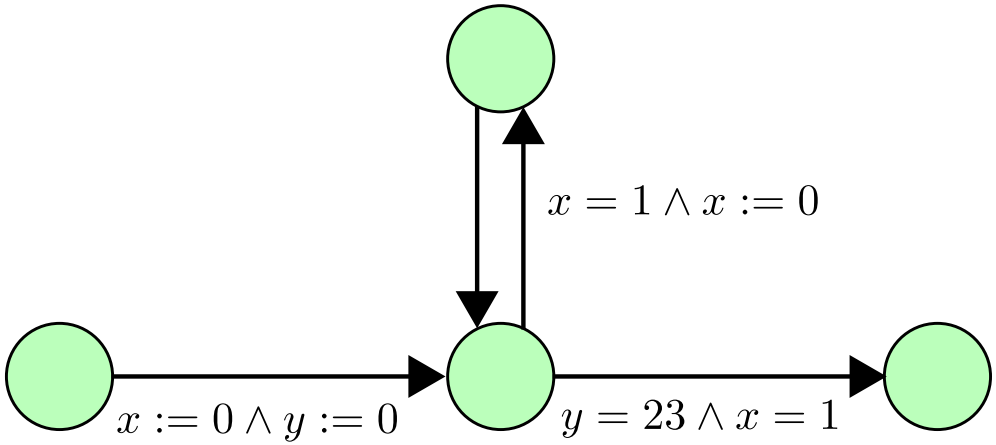
[1] *now at* LSV, CNRS & ENS de Cachan, France

[2] Department of Computer Science, University of Oxford, UK

Reachability Problems '12 — September 18, 2012

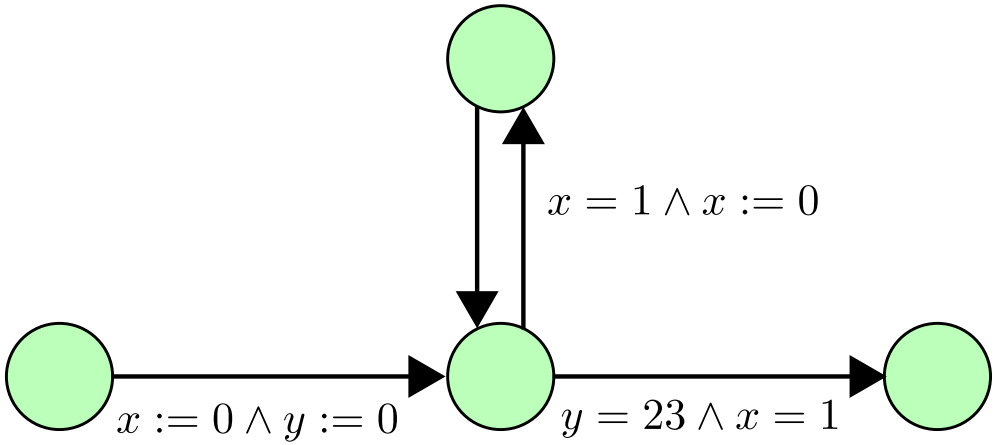
Introduction

Timed Automata



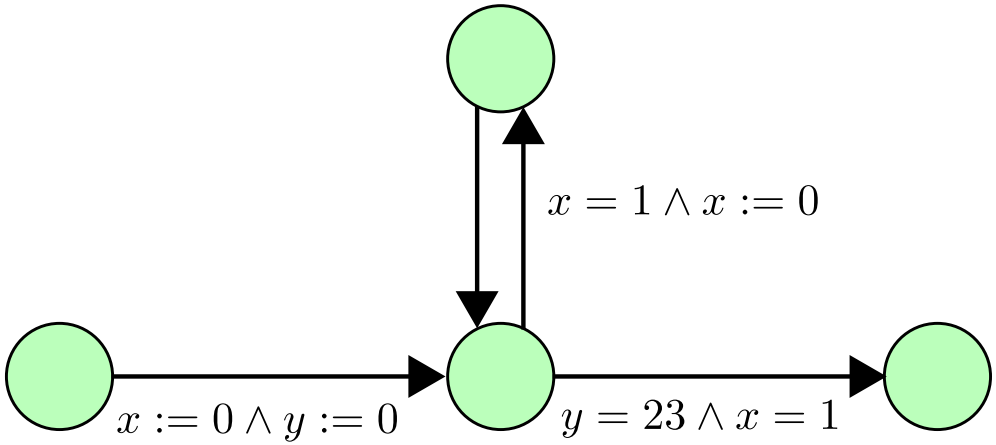
- Comprise a finite-state controller with a finite number of clocks ranging of $\mathbb{R}_{\geq 0}$
- Along transitions clocks can be compared to constants and reset
- Constants are encoded in binary

Timed Automata



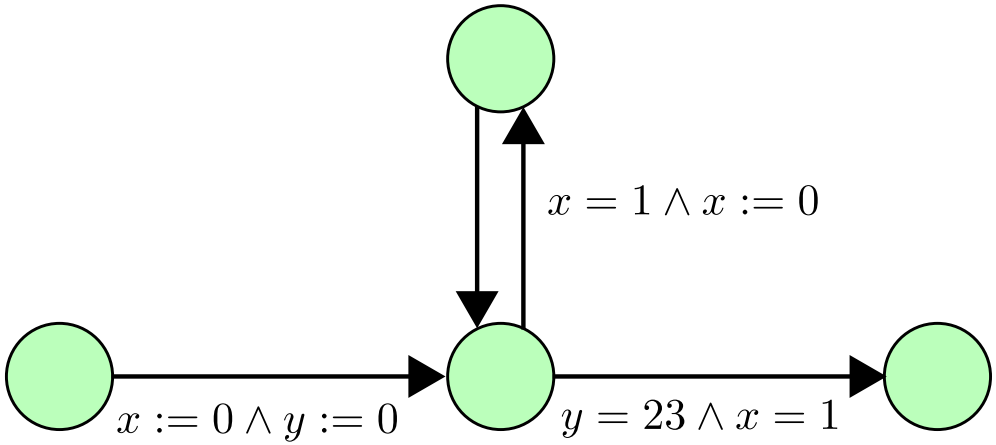
- Comprise a finite-state controller with a finite number of clocks ranging of $\mathbb{R}_{\geq 0}$
- Along transitions clocks can be compared to constants and reset
- Constants are encoded in binary

Timed Automata



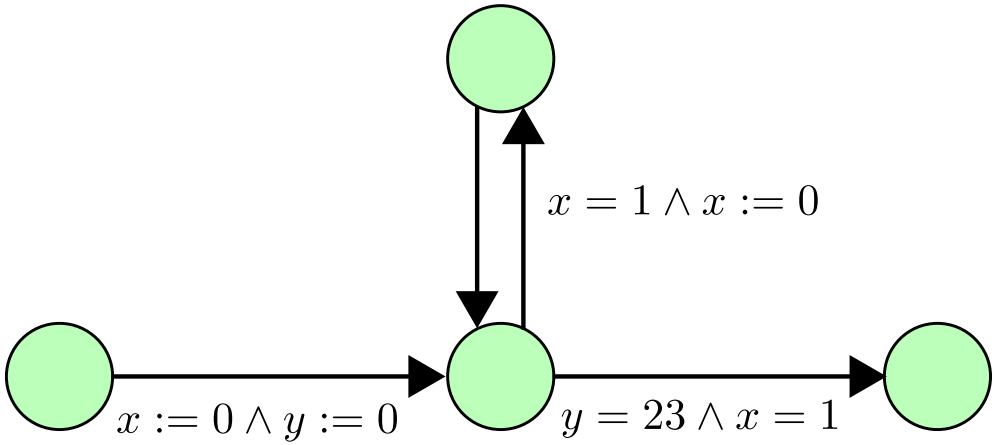
- Comprise a finite-state controller with a finite number of clocks ranging of $\mathbb{R}_{\geq 0}$
- Along transitions clocks can be compared to constants and reset
- Constants are encoded in binary

Timed Automata



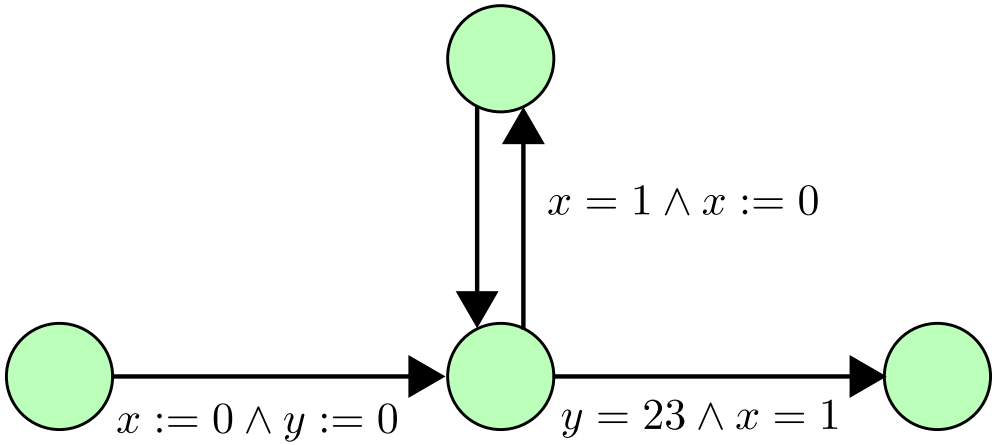
- Comprise a finite-state controller with a finite number of clocks ranging of $\mathbb{R}_{\geq 0}$
- Along transitions clocks can be compared to constants and reset
- Constants are encoded in binary

Timed Automata



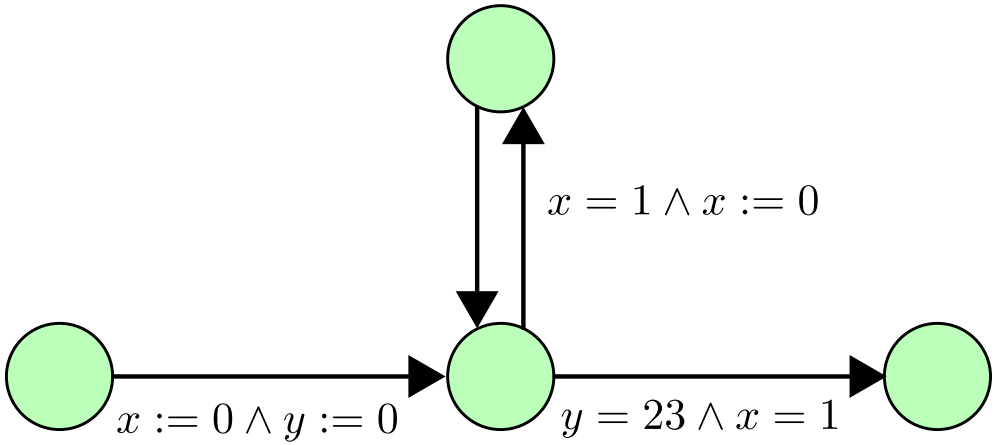
- Comprise a finite-state controller with a finite number of clocks ranging of $\mathbb{R}_{\geq 0}$
- Along transitions clocks can be compared to constants and reset
- Constants are encoded in binary

Timed Automata

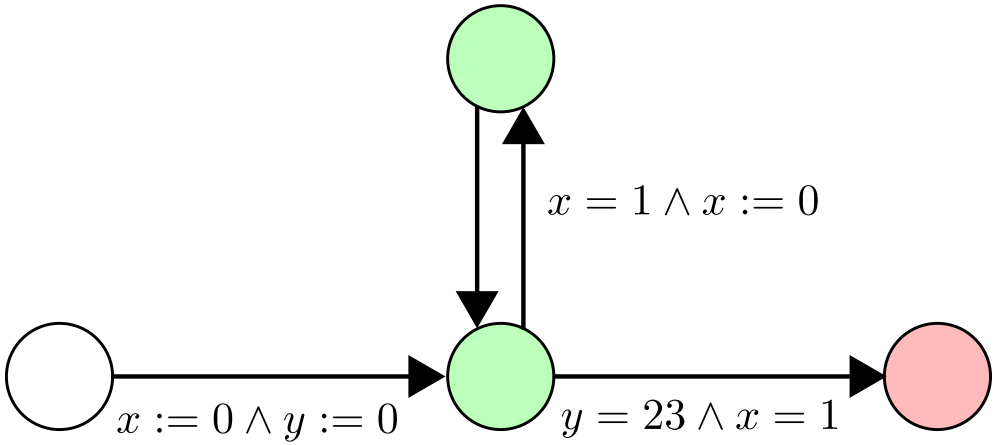


- Comprise a finite-state controller with a finite number of clocks ranging of $\mathbb{R}_{\geq 0}$
- Along transitions clocks can be compared to constants and reset
- Constants are encoded in binary

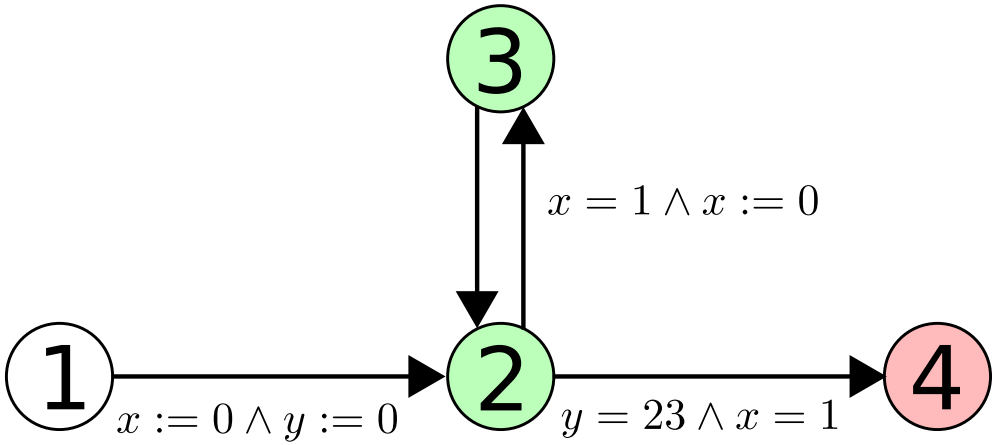
Reachability in Timed Automata



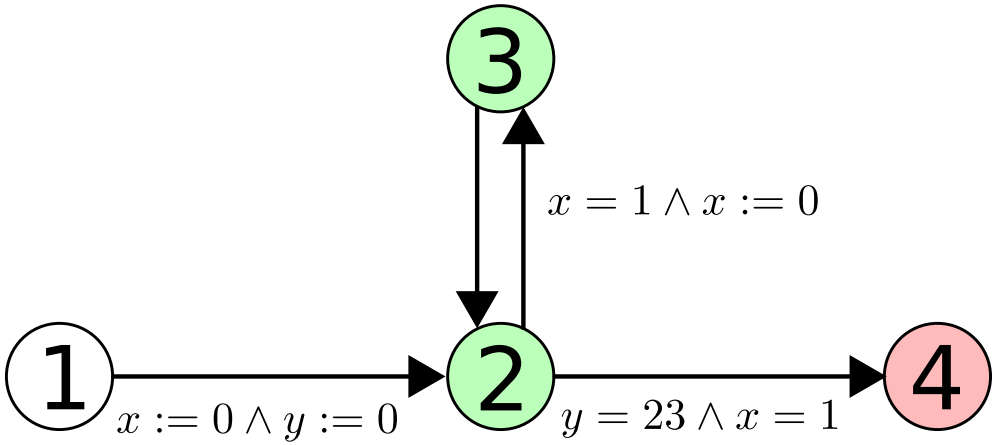
Reachability in Timed Automata



Reachability in Timed Automata

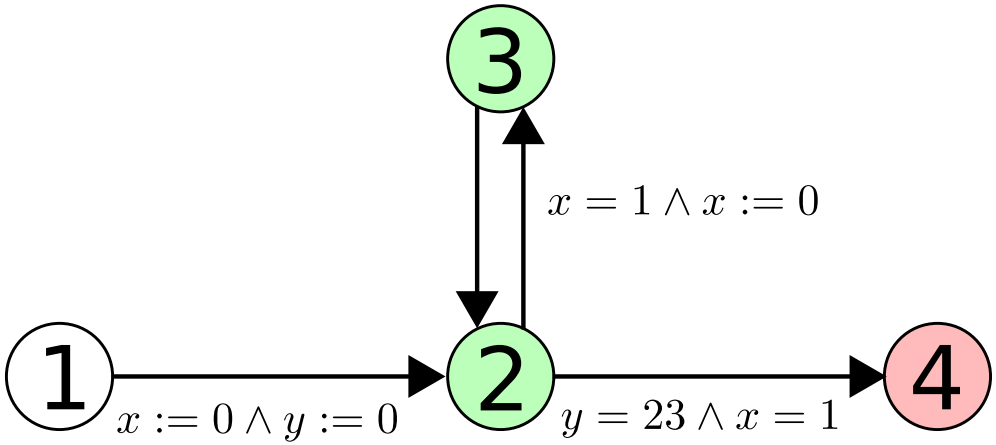


Reachability in Timed Automata



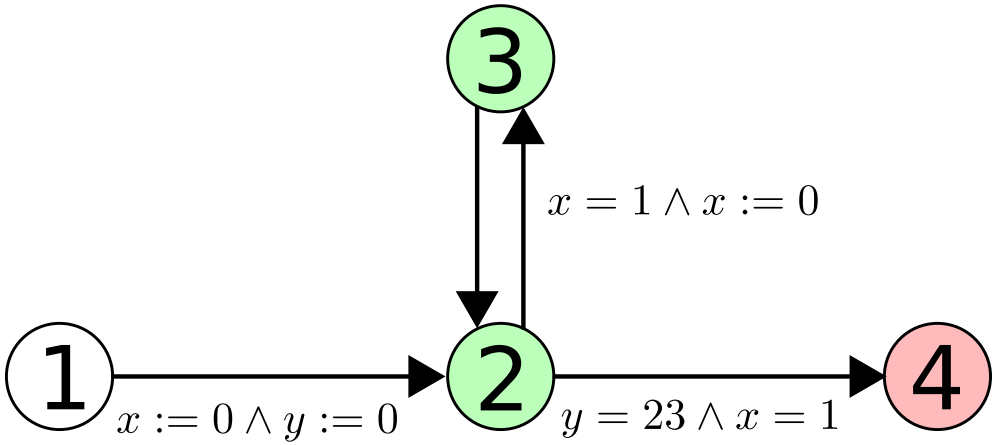
Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

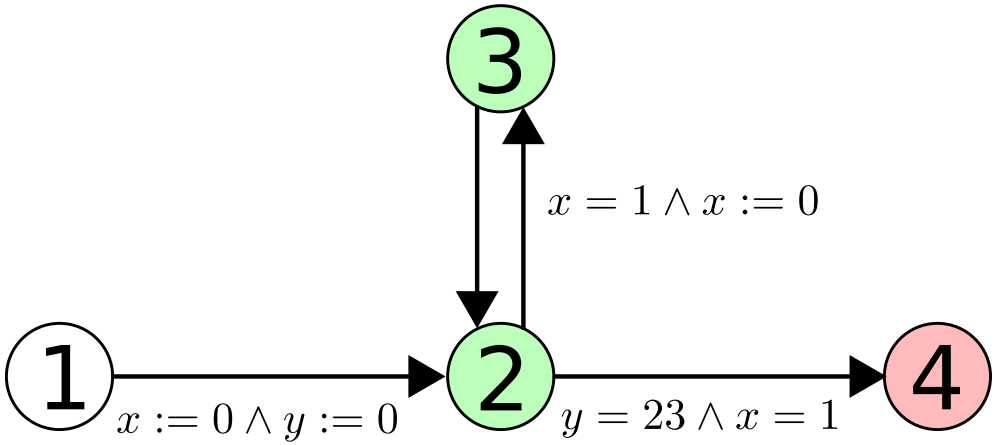
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

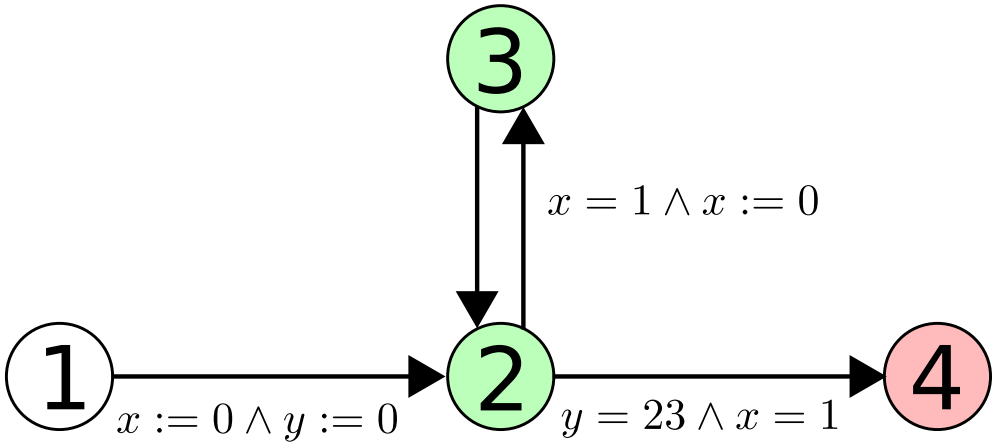
Yes we can!

Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

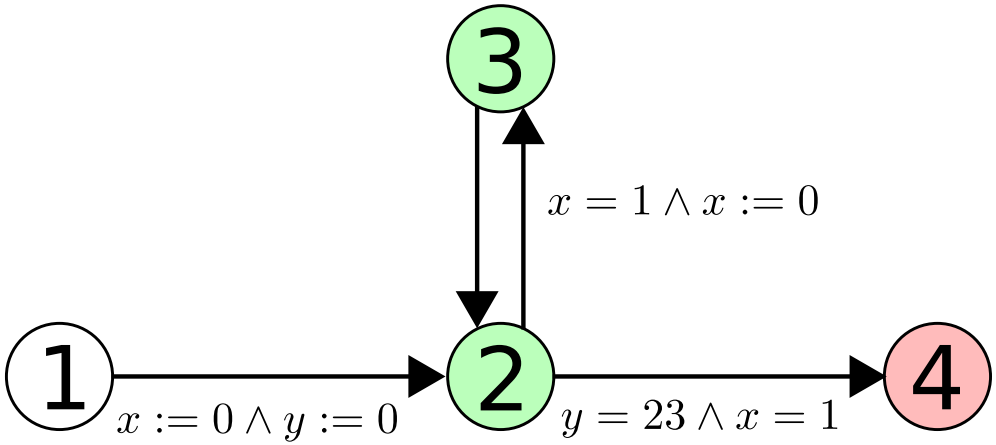
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$(1, x \mapsto 4, y \mapsto 2)$

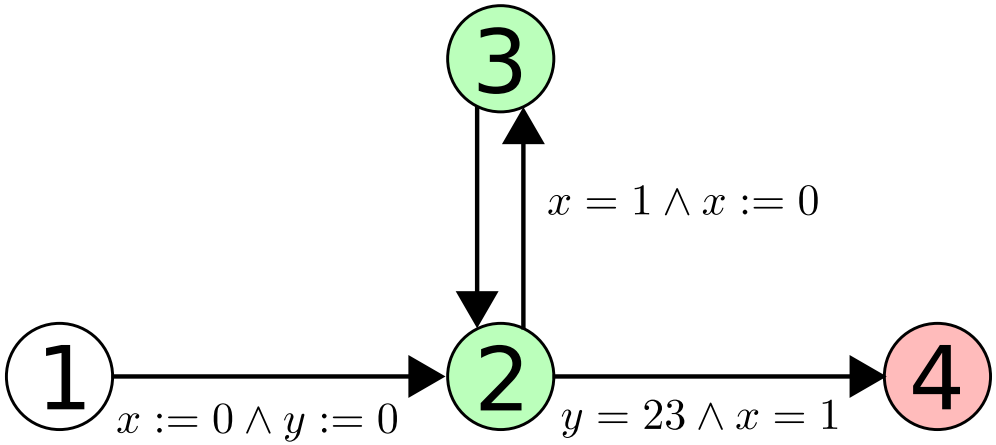
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$(1, x \mapsto 4, y \mapsto 2) \rightarrow (2, x \mapsto 0, y \mapsto 0)$

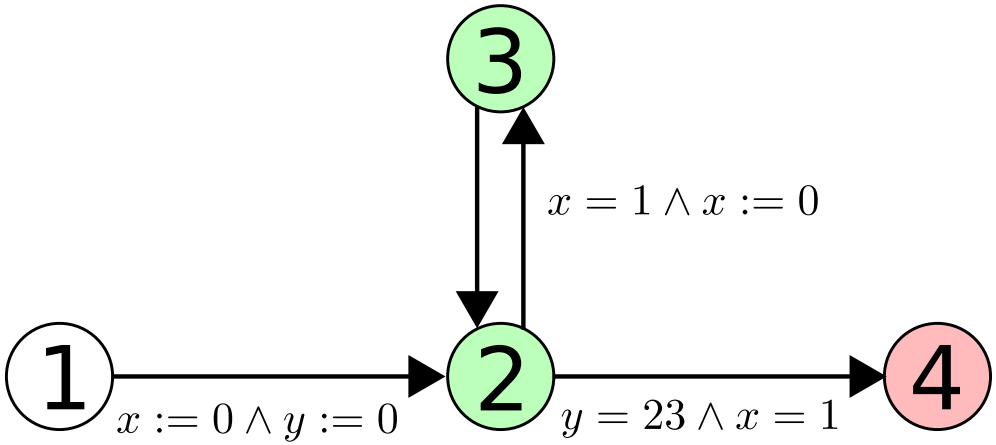
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$(1, x \mapsto 4, y \mapsto 2) \rightarrow (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$

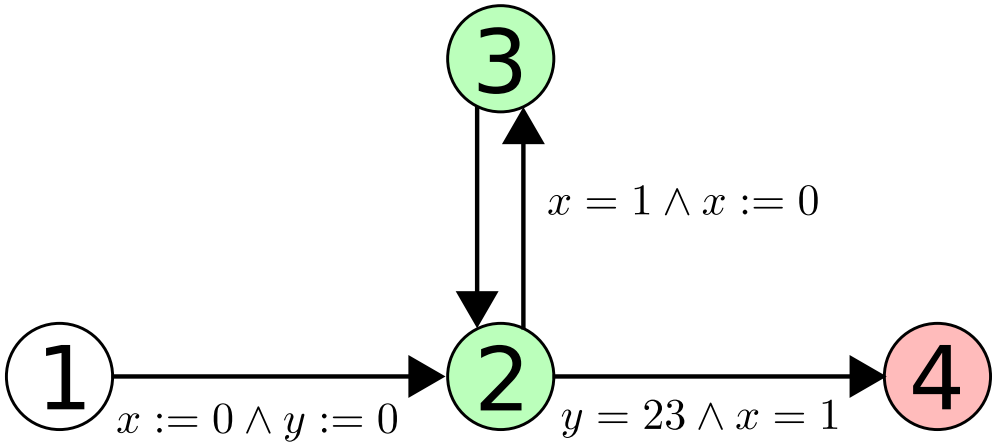
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \to (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\to (3, x \mapsto 0, y \mapsto 1)$$

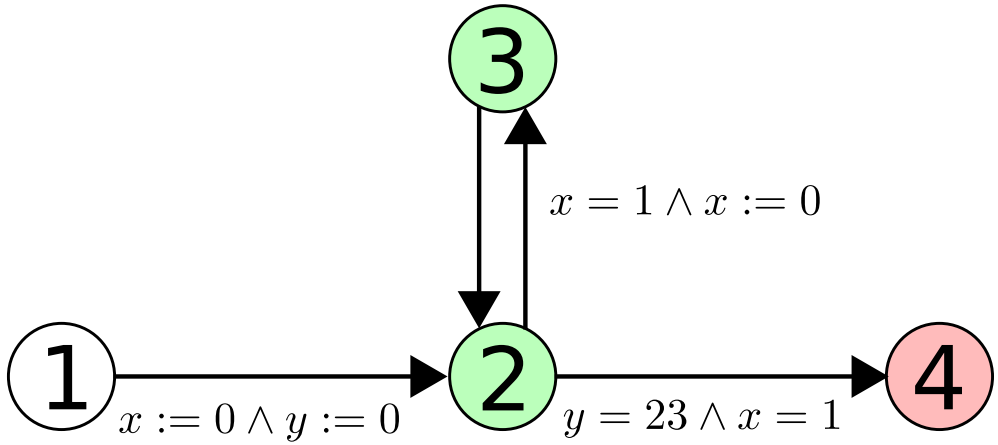
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \rightarrow (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\rightarrow (3, x \mapsto 0, y \mapsto 1) \rightsquigarrow (3, x \mapsto 0.5, y \mapsto 1.5)$$

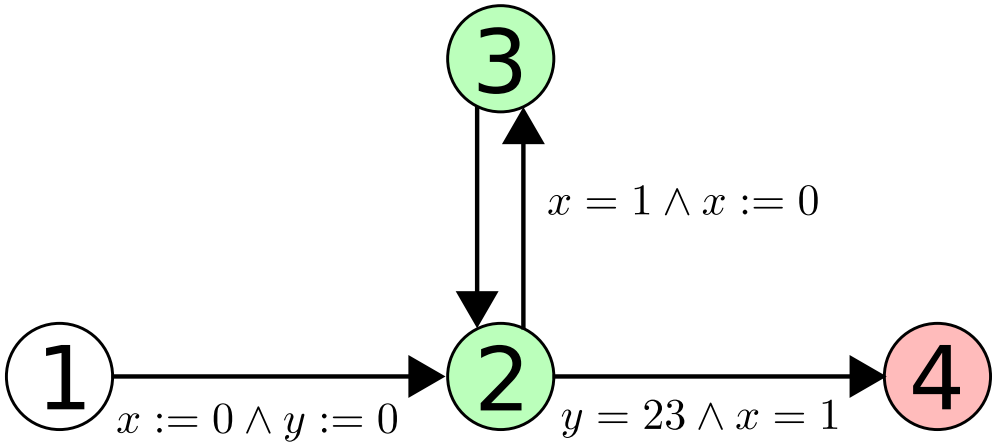
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \to (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\to (3, x \mapsto 0, y \mapsto 1) \rightsquigarrow (3, x \mapsto 0.5, y \mapsto 1.5)$$
$$\to (2, x \mapsto 0.5, y \mapsto 1.5)$$

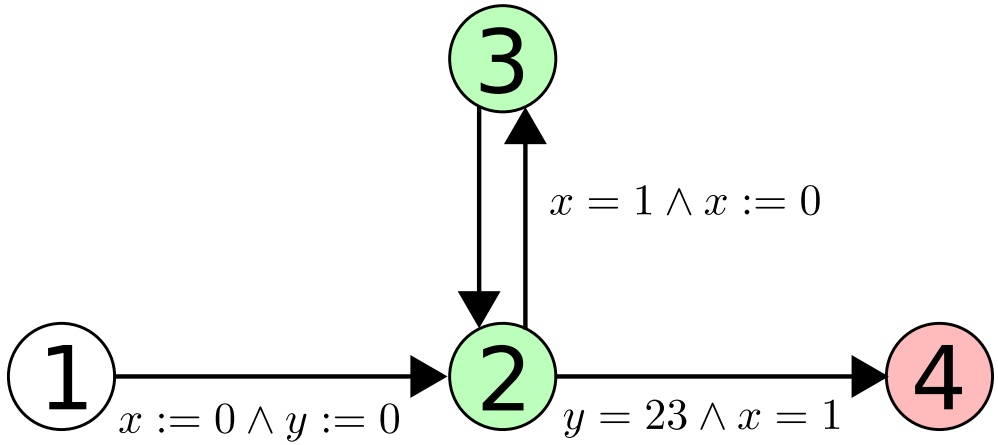
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \to (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\to (3, x \mapsto 0, y \mapsto 1) \rightsquigarrow (3, x \mapsto 0.5, y \mapsto 1.5)$$
$$\to (2, x \mapsto 0.5, y \mapsto 1.5) \rightsquigarrow (2, x \mapsto 1, y \mapsto 2)$$

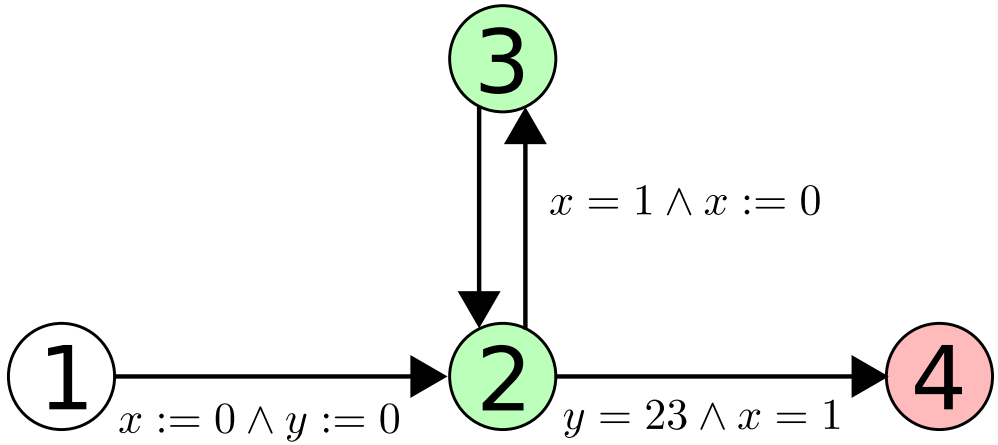
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \to (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\to (3, x \mapsto 0, y \mapsto 1) \rightsquigarrow (3, x \mapsto 0.5, y \mapsto 1.5)$$
$$\to (2, x \mapsto 0.5, y \mapsto 1.5) \rightsquigarrow (2, x \mapsto 1, y \mapsto 2) \to \cdots$$

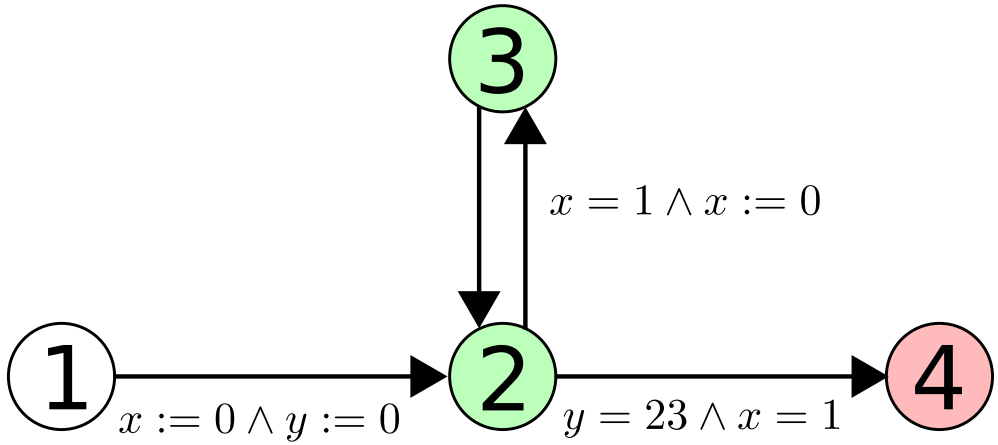
Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \to (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\to (3, x \mapsto 0, y \mapsto 1) \rightsquigarrow (3, x \mapsto 0.5, y \mapsto 1.5)$$
$$\to (2, x \mapsto 0.5, y \mapsto 1.5) \rightsquigarrow (2, x \mapsto 1, y \mapsto 2) \to \cdots$$
$$\to (2, x \mapsto 1, y \mapsto 23)$$

Reachability in Timed Automata



Can we reach $(4, x \mapsto 1, y \mapsto 23)$ starting in $(1, x \mapsto 4, y \mapsto 2)$

$$(1, x \mapsto 4, y \mapsto 2) \to (2, x \mapsto 0, y \mapsto 0) \rightsquigarrow (2, x \mapsto 1, y \mapsto 1)$$
$$\to (3, x \mapsto 0, y \mapsto 1) \rightsquigarrow (3, x \mapsto 0.5, y \mapsto 1.5)$$
$$\to (2, x \mapsto 0.5, y \mapsto 1.5) \rightsquigarrow (2, x \mapsto 1, y \mapsto 2) \to \cdots$$
$$\to (2, x \mapsto 1, y \mapsto 23) \to (4, x \mapsto 1, y \mapsto 23)$$

Reachability in Timed Automata

- General reachability problem is PSPACE-complete [Alur, Dill 1994]
- Reachability is PSPACE-complete for 3 clocks, or for an unbounded number of clocks and constants from $\{0, 1\}$ [Courcoubetis, Yannakakis, 1992]
- Reachability is NLOGSPACE-complete for one clock and NP-hard for two clocks [Laroussinie, Markey, Schnoebelen, 2004]

Reachability in Timed Automata

- General reachability problem is PSPACE-complete [Alur, Dill 1994]
- Reachability is PSPACE-complete for 3 clocks, or for an unbounded number of clocks and constants from $\{0, 1\}$ [Courcoubetis, Yannakakis, 1992]
- Reachability is NLOGSPACE-complete for one clock and NP-hard for two clocks [Laroussinie, Markey, Schnoebelen, 2004]

Reachability in Timed Automata

- General reachability problem is PSPACE-complete [Alur, Dill 1994]
- Reachability is PSPACE-complete for 3 clocks, or for an unbounded number of clocks and constants from $\{0, 1\}$ [Courcoubetis, Yannakakis, 1992]
- Reachability is NLOGSPACE-complete for one clock and NP-hard for two clocks [Laroussinie, Markey, Schnoebelen, 2004]

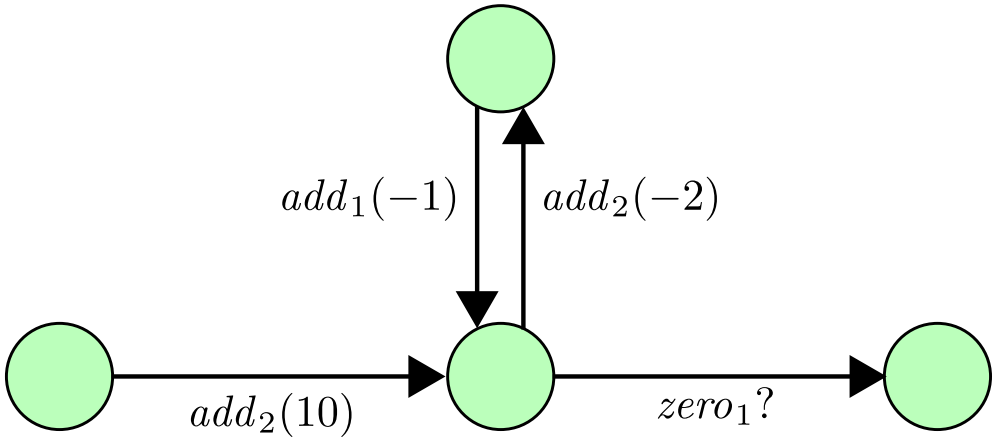
Reachability in Timed Automata

- General reachability problem is PSPACE-complete [Alur, Dill 1994]
- Reachability is PSPACE-complete for 3 clocks, or for an unbounded number of clocks and constants from $\{0, 1\}$ [Courcoubetis, Yannakakis, 1992]
- Reachability is NLOGSPACE-complete for one clock and NP-hard for two clocks [Laroussinie, Markey, Schnoebelen, 2004]

Reachability in Timed Automata

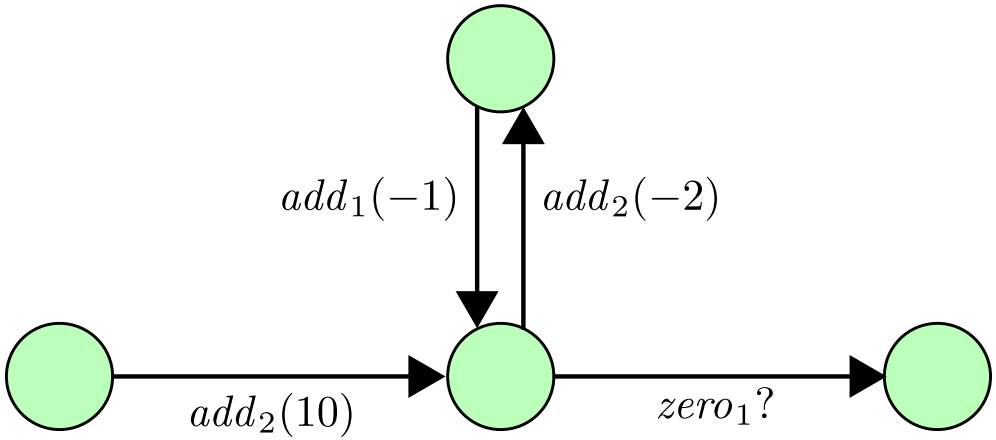
- General reachability problem is PSPACE-complete [Alur, Dill 1994]
- Reachability is PSPACE-complete for 3 clocks, or for an unbounded number of clocks and constants from $\{0, 1\}$ [Courcoubetis, Yannakakis, 1992]
- Reachability is NLOGSPACE-complete for one clock and NP-hard for two clocks [Laroussinie, Markey, Schnoebelen, 2004]

Counter Automata



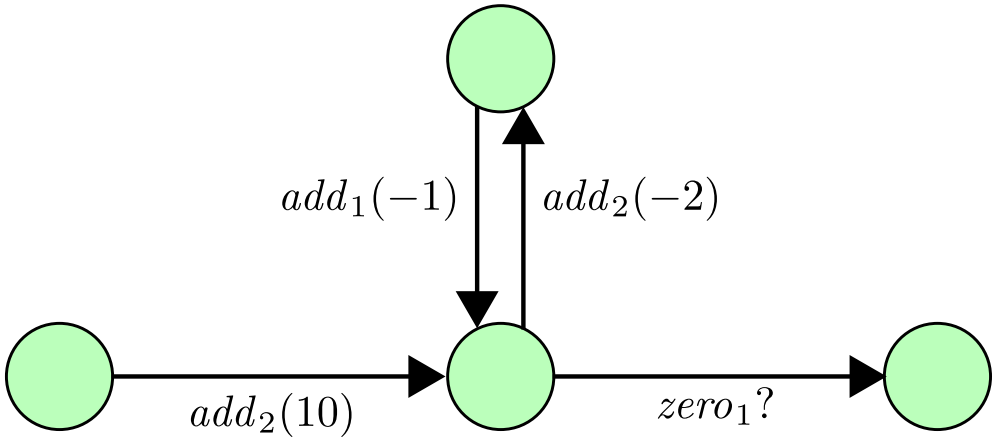
- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

Counter Automata



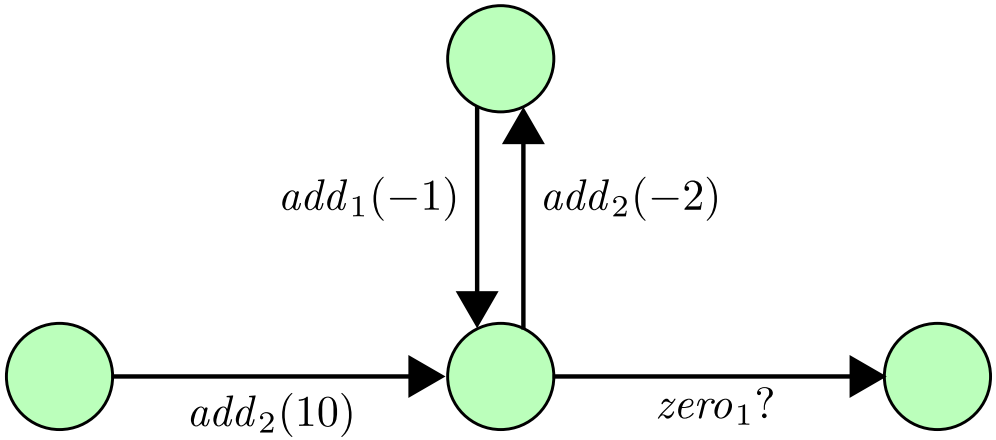
- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

Counter Automata



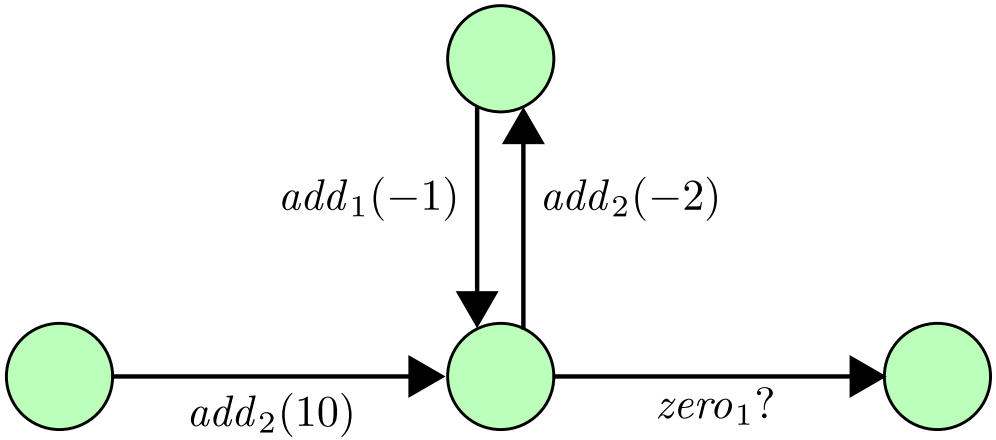
- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

Counter Automata



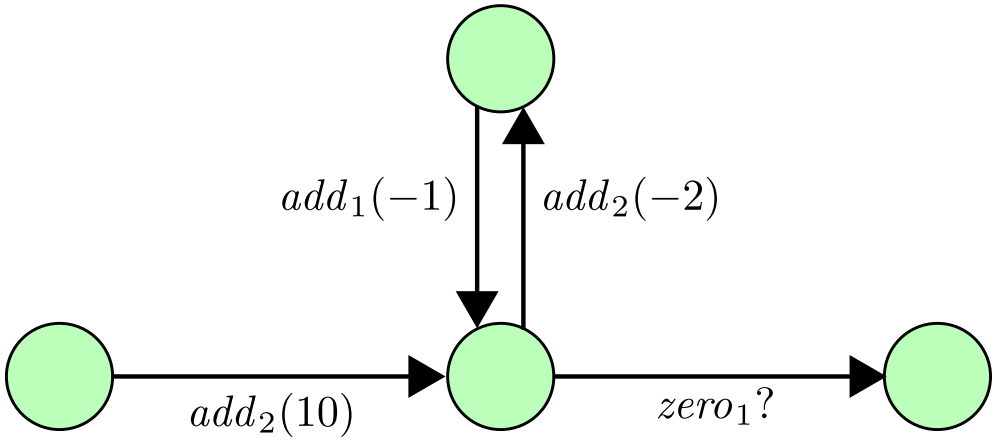
- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

Counter Automata



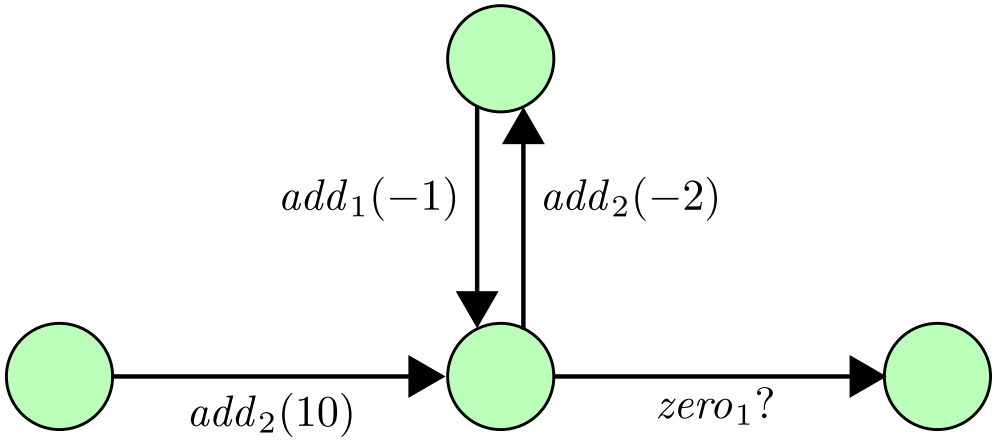
- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

Counter Automata



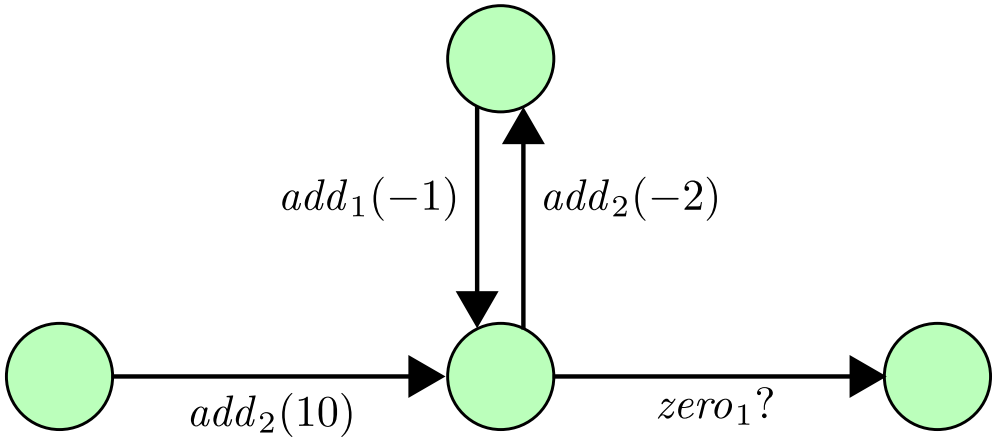
- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

Counter Automata

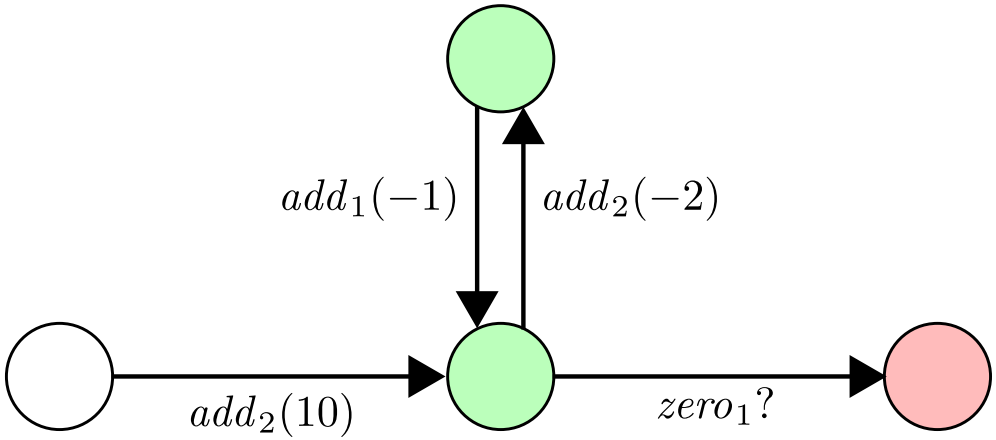


- Comprise a finite-state controller with a finite number of counters ranging of $\mathbb{N}$
- Along transitions counters can be incremented, decremented or tested for zero
- Constants are encoded in binary

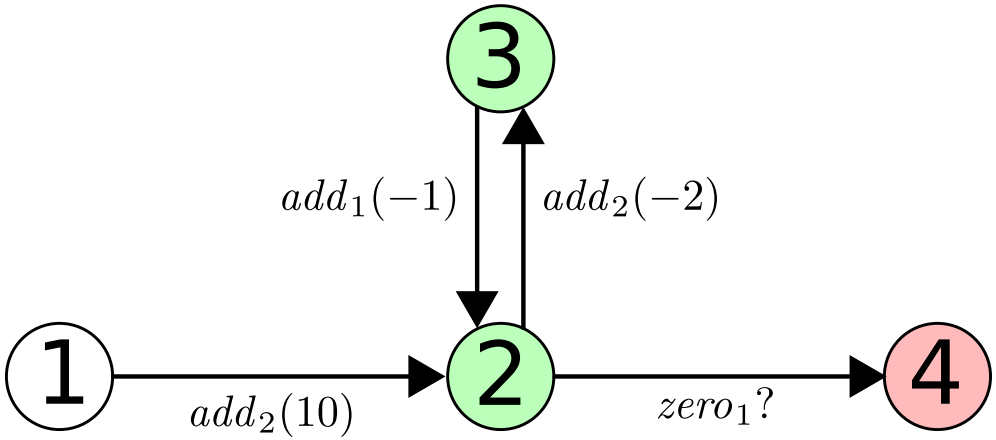
Reachability in Counter Automata



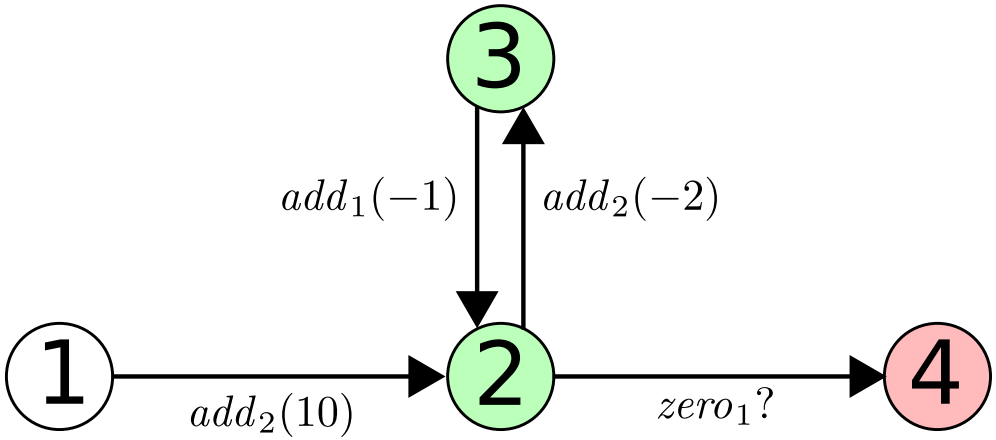
Reachability in Counter Automata



Reachability in Counter Automata

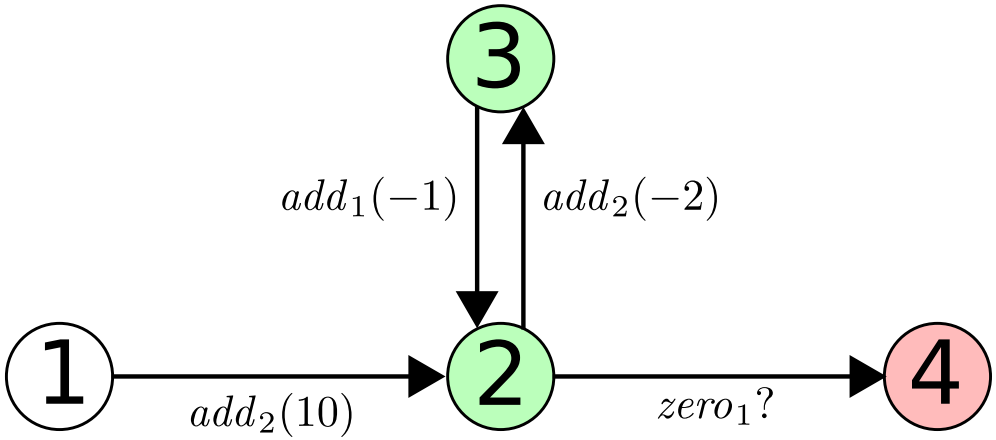


Reachability in Counter Automata



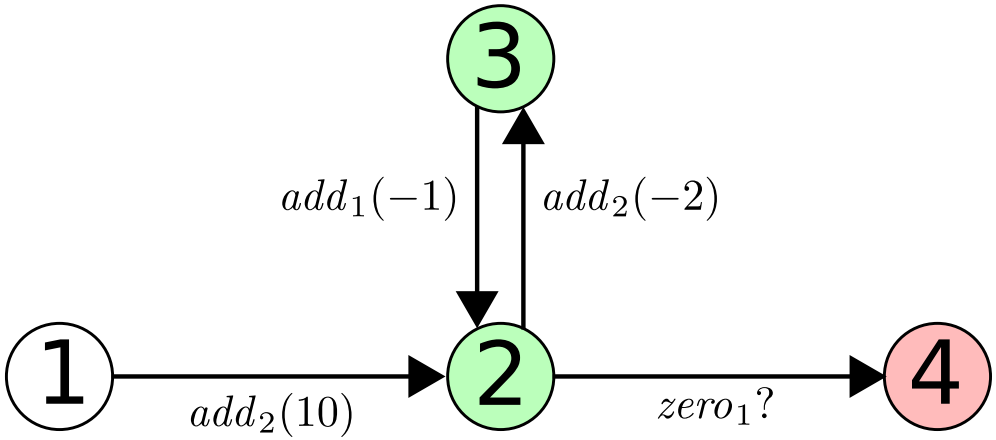
Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

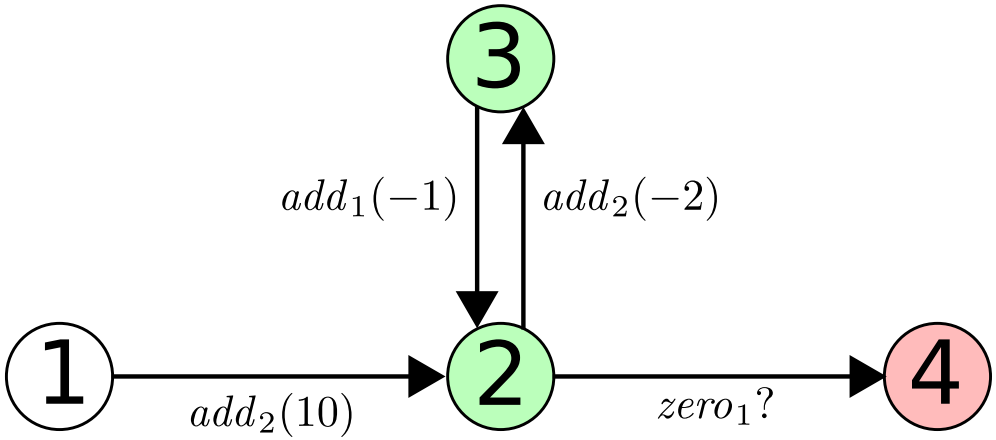
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

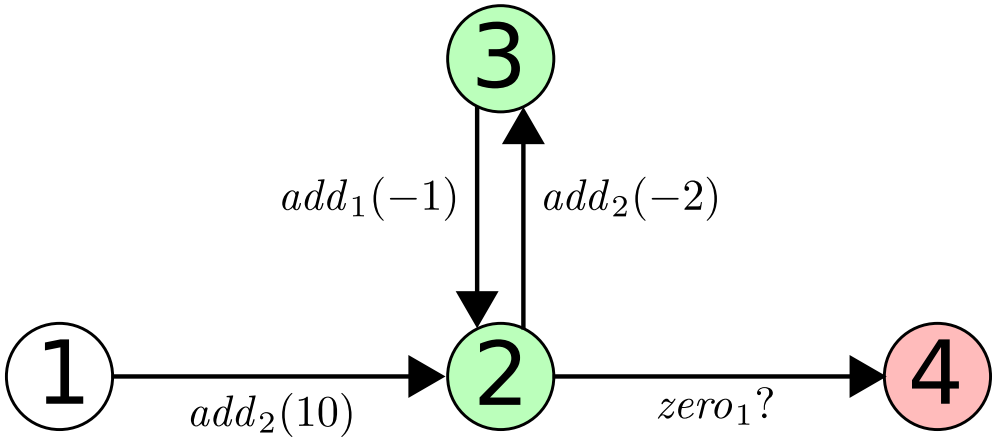
No we cannot!

Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

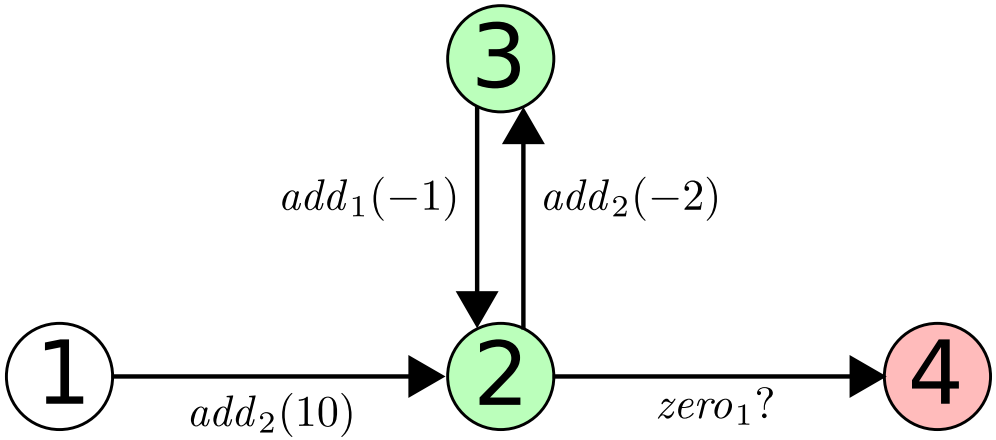
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$(1, c_1 \mapsto 6, c_2 \mapsto 0)$

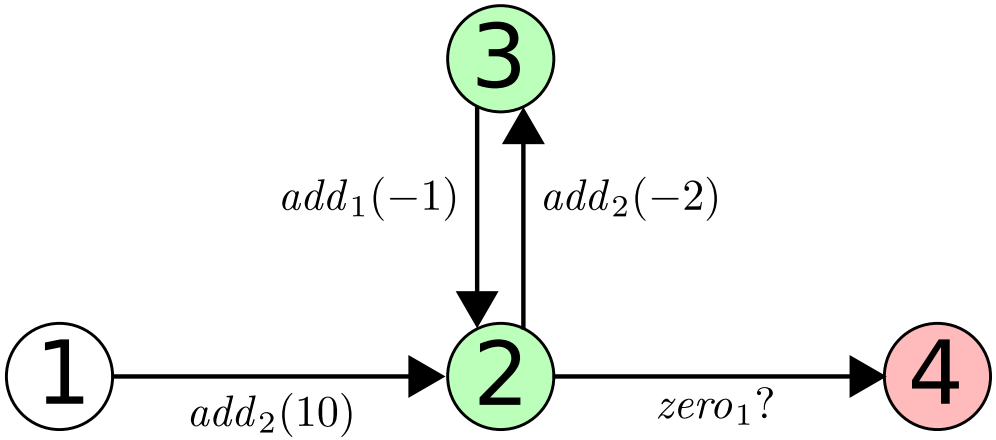
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$(1, c_1 \mapsto 6, c_2 \mapsto 0) \rightarrow (2, c_1 \mapsto 6, c_2 \mapsto 10)$

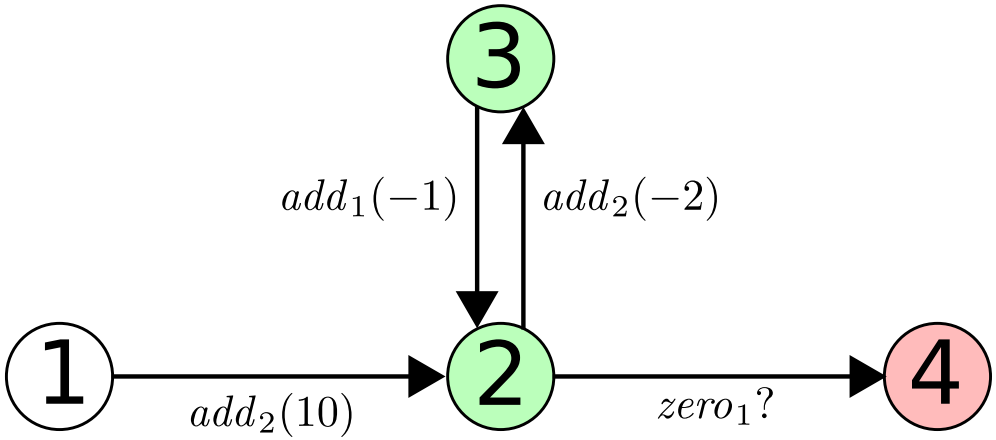
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$(1, c_1 \mapsto 6, c_2 \mapsto 0) \rightarrow (2, c_1 \mapsto 6, c_2 \mapsto 10) \rightarrow (3, c_1 \mapsto 6, c_2 \mapsto 8)$

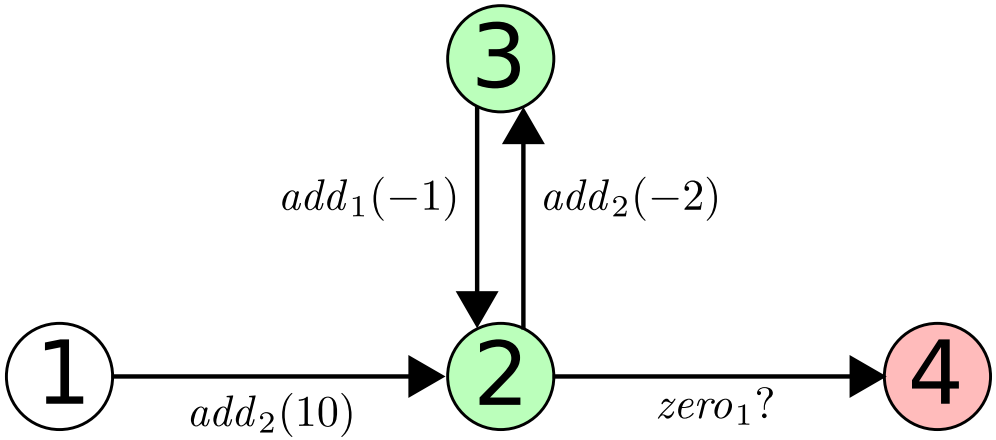
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$(1, c_1 \mapsto 6, c_2 \mapsto 0) \rightarrow (2, c_1 \mapsto 6, c_2 \mapsto 10) \rightarrow (3, c_1 \mapsto 6, c_2 \mapsto 8)$
$\rightarrow (2, c_1 \mapsto 5, c_2 \mapsto 8)$

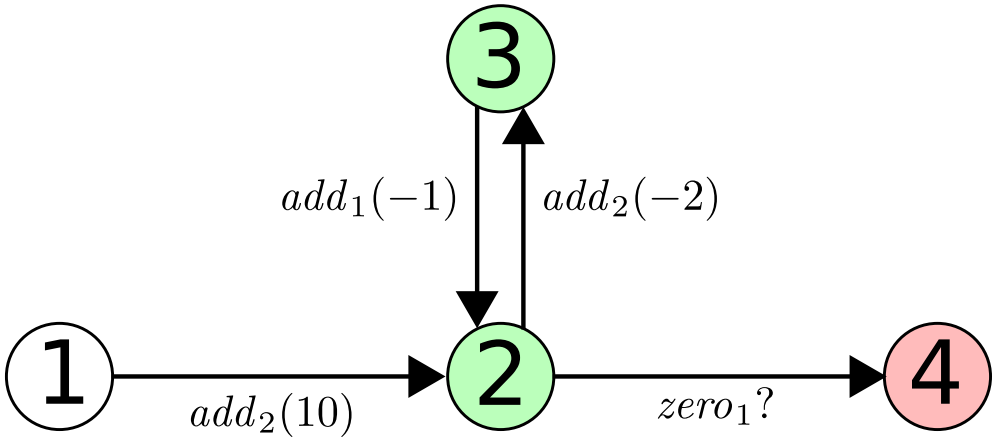
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$(1, c_1 \mapsto 6, c_2 \mapsto 0) \to (2, c_1 \mapsto 6, c_2 \mapsto 10) \to (3, c_1 \mapsto 6, c_2 \mapsto 8)$
$\to (2, c_1 \mapsto 5, c_2 \mapsto 8) \to \cdots$

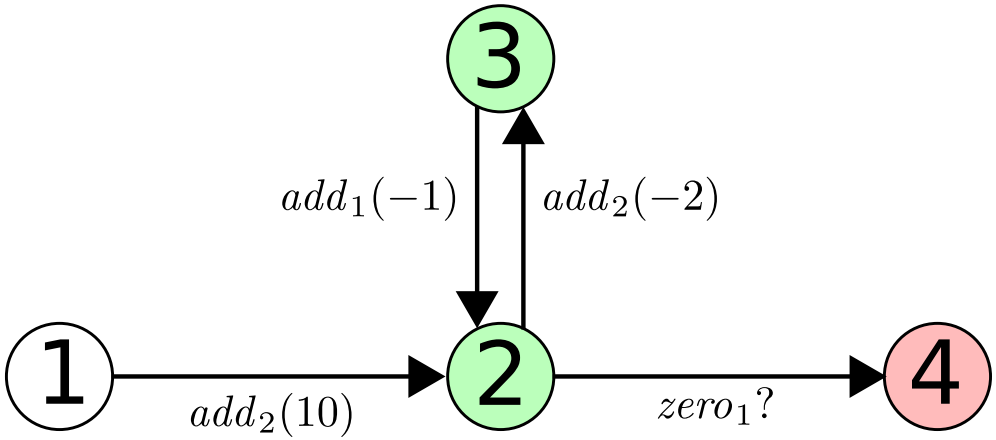
Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$(1, c_1 \mapsto 6, c_2 \mapsto 0) \to (2, c_1 \mapsto 6, c_2 \mapsto 10) \to (3, c_1 \mapsto 6, c_2 \mapsto 8)$
$\to (2, c_1 \mapsto 5, c_2 \mapsto 8) \to \cdots \to (2, c_1 \mapsto 1, c_2 \mapsto 0)$

Reachability in Counter Automata



Can we reach $(4, c_1 \mapsto 0, c_2 \mapsto 0)$ starting in $(1, c_1 \mapsto 6, c_2 \mapsto 0)$?

$$(1, c_1 \mapsto 6, c_2 \mapsto 0) \to (2, c_1 \mapsto 6, c_2 \mapsto 10) \to (3, c_1 \mapsto 6, c_2 \mapsto 8)$$
$$\to (2, c_1 \mapsto 5, c_2 \mapsto 8) \to \cdots \to (2, c_1 \mapsto 1, c_2 \mapsto 0) \to \lightning$$

Reachability in Counter Automata

- Reachability in counter automata is undecidable already for two counters [Minsky, 1961]
- Reachability is NP-complete for one counter [H., Kreutzer, O., W., 2009]
- Reachability is NLOGSPACE-complete for one counter with numbers encoded in unary [Lafourcade, Lugiez, Treinen, 2004]

Reachability in Counter Automata

- Reachability in counter automata is undecidable already for two counters [Minsky, 1961]
- Reachability is NP-complete for one counter [H., Kreutzer, O., W., 2009]
- Reachability is NLOGSPACE-complete for one counter with numbers encoded in unary [Lafourcade, Lugiez, Treinen, 2004]

Reachability in Counter Automata

- Reachability in counter automata is undecidable already for two counters [Minsky, 1961]
- Reachability is NP-complete for one counter [H., Kreutzer, O., W., 2009]
- Reachability is NLOGSPACE-complete for one counter with numbers encoded in unary [Lafourcade, Lugiez, Treinen, 2004]

Reachability in Counter Automata

- Reachability in counter automata is undecidable already for two counters [Minsky, 1961]
- Reachability is NP-complete for one counter [H., Kreutzer, O., W., 2009]
- Reachability is NLOGSPACE-complete for one counter with numbers encoded in unary [Lafourcade, Lugiez, Treinen, 2004]

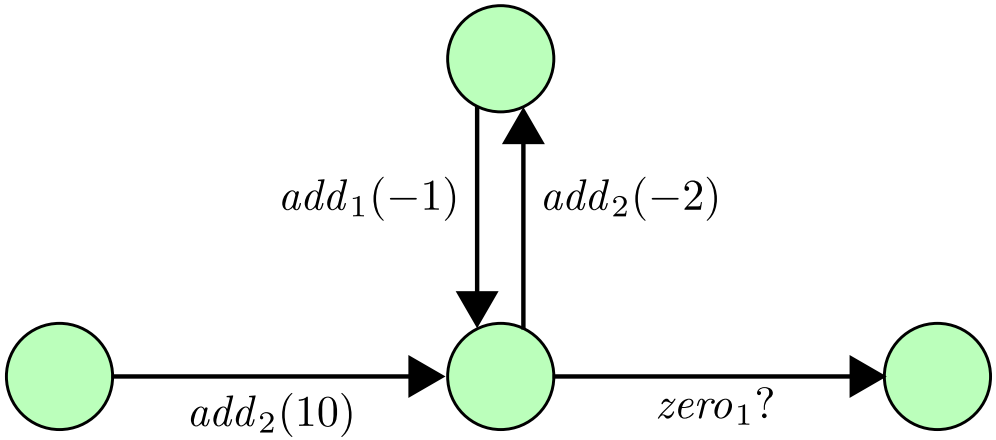
This talk:

Can we naturally relate reachability problems in timed and counter automata?

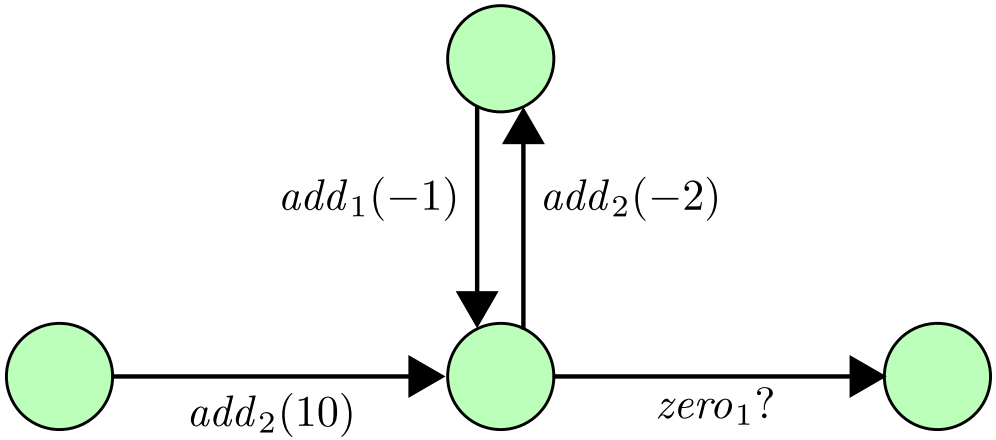
This talk:

Can we naturally relate reachability problems in timed and counter automata?

Bounded Counter Automata

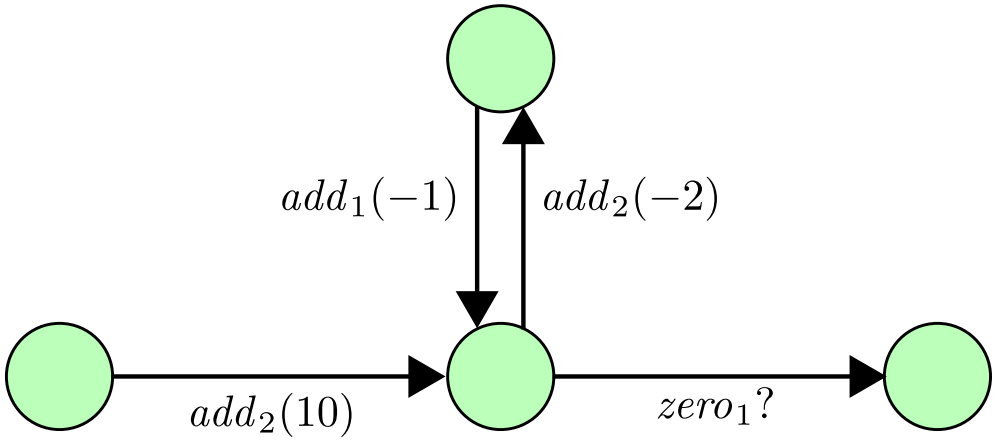


Bounded Counter Automata



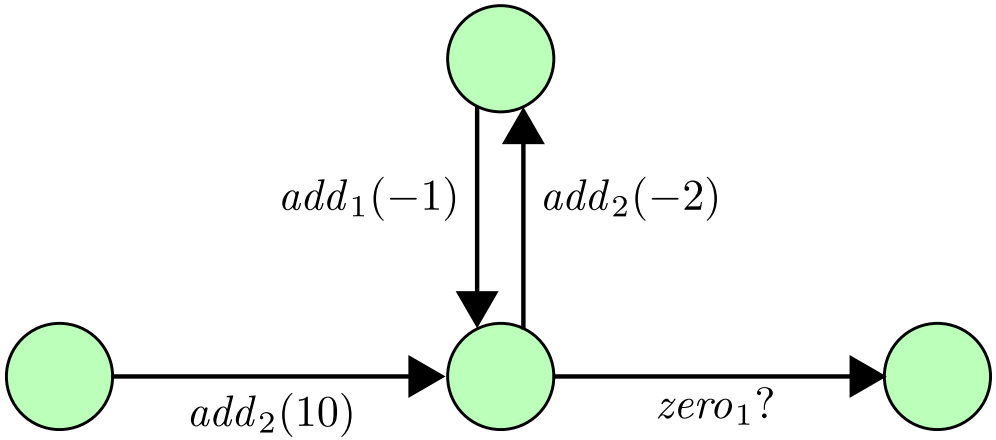
- Counters are constrained to take values from bounded intervals from $\mathbb{N}$
- Zero tests can be discarded
- Can be viewed as strongly-bounded VASS as defined by [Memmim, Roucairol, 1980]

Bounded Counter Automata



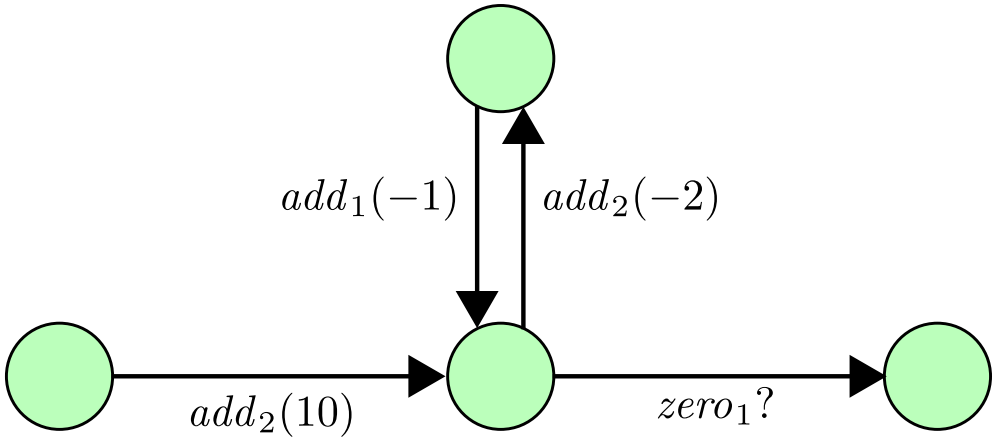
- Counters are constrained to take values from bounded intervals from $\mathbb{N}$
- Zero tests can be discarded
- Can be viewed as strongly-bounded VASS as defined by [Memmim, Roucairol, 1980]

Bounded Counter Automata



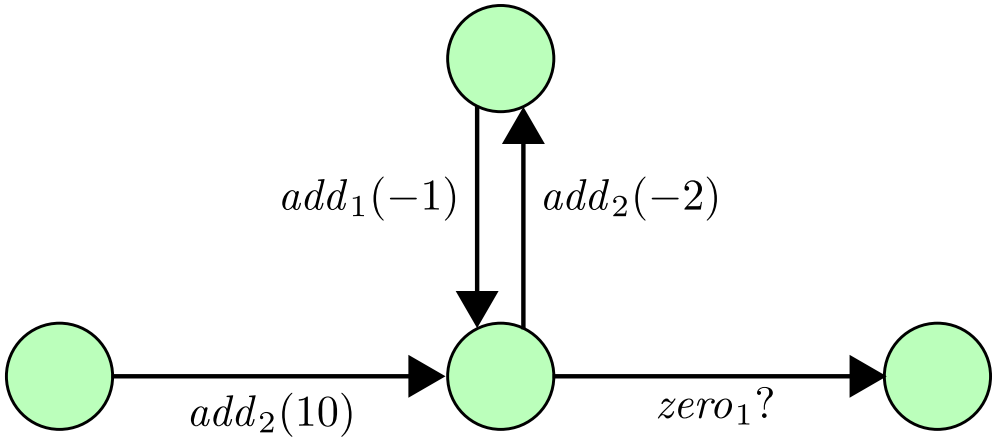
- Counters are constrained to take values from bounded intervals from $\mathbb{N}$
- Zero tests can be discarded
- Can be viewed as strongly-bounded VASS as defined by [Memmim, Roucairol, 1980]

Bounded Counter Automata



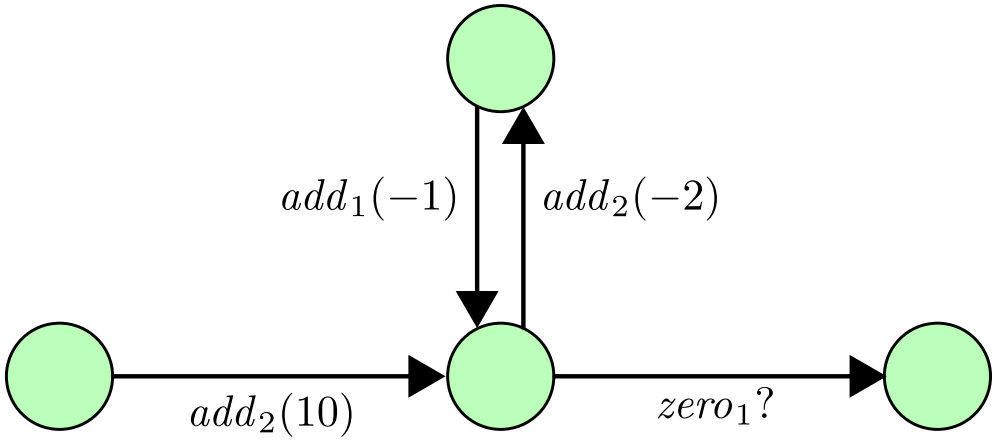
- Reachability is trivially decidable and in PSPACE
- Reachability with one counter is NP-hard and in PSPACE [Bouyer *et al.*, 2008]
- Reachability with one counter inter-reducible with model considered by Demri and Gascon where counter ranges over $\mathbb{Z}$ and sign tests are allowed

Bounded Counter Automata



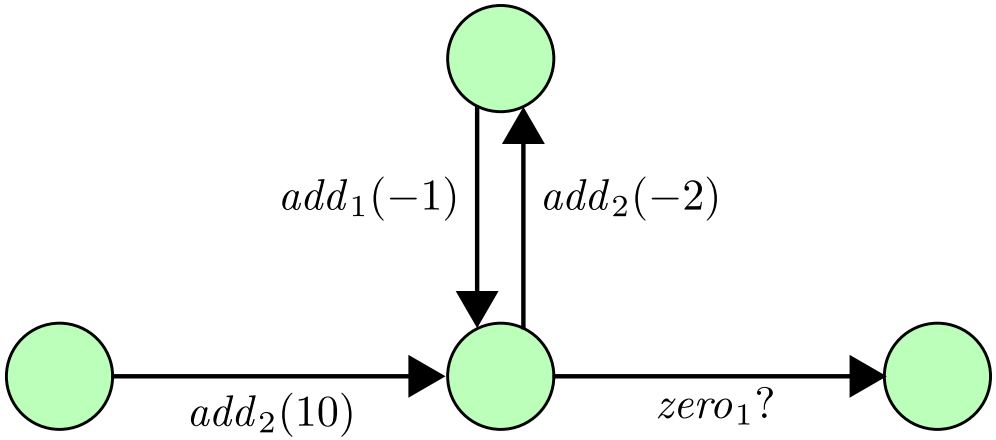
- Reachability is trivially decidable and in PSPACE
- Reachability with one counter is NP-hard and in PSPACE [Bouyer *et al.*, 2008]
- Reachability with one counter inter-reducible with model considered by Demri and Gascon where counter ranges over $\mathbb{Z}$ and sign tests are allowed

Bounded Counter Automata



- Reachability is trivially decidable and in PSPACE
- Reachability with one counter is NP-hard and in PSPACE [Bouyer *et al.*, 2008]
- Reachability with one counter inter-reducible with model considered by Demri and Gascon where counter ranges over $\mathbb{Z}$ and sign tests are allowed

Bounded Counter Automata



- Reachability is trivially decidable and in PSPACE
- Reachability with one counter is NP-hard and in PSPACE [Bouyer *et al.*, 2008]
- Reachability with one counter inter-reducible with model considered by Demri and Gascon where counter ranges over $\mathbb{Z}$ and sign tests are allowed

Remainder of this talk

Relating Reachability in Timed and Bounded Counter Automata with respect to logspace reductions:

Reach. in n-clock TA, $n \geq 3$	$\Longleftrightarrow$	Reach. in bounded 2-CA
Reach. in 2-clock TA	$\Longleftrightarrow$	Reach. in bounded 1-CA
Reach. in 1-clock TA	$\Longleftrightarrow$	Reach. in bounded 0-CA

Remainder of this talk

Relating Reachability in Timed and Bounded Counter Automata with respect to logspace reductions:

Reach. in n-clock TA, $n \geq 3$	$\Longleftrightarrow$	Reach. in bounded 2-CA
Reach. in 2-clock TA	$\Longleftrightarrow$	Reach. in bounded 1-CA
Reach. in 1-clock TA	$\Longleftrightarrow$	Reach. in bounded 0-CA

Remainder of this talk

Relating Reachability in Timed and Bounded Counter Automata with respect to logspace reductions:

Reach. in n-clock TA, $n \geq 3$	$\Longleftrightarrow$	Reach. in bounded 2-CA
Reach. in 2-clock TA	$\Longleftrightarrow$	Reach. in bounded 1-CA
Reach. in 1-clock TA	$\Longleftrightarrow$	Reach. in bounded 0-CA

Remainder of this talk

Relating Reachability in Timed and Bounded Counter Automata with respect to logspace reductions:

Reach. in n-clock TA, $n \geq 3$	$\Longleftrightarrow$	Reach. in bounded 2-CA
Reach. in 2-clock TA	$\Longleftrightarrow$	Reach. in bounded 1-CA
Reach. in 1-clock TA	$\Longleftrightarrow$	Reach. in bounded 0-CA

Remainder of this talk

Relating Reachability in Timed and Bounded Counter Automata with respect to logspace reductions:

Reach. in n-clock TA, $n \geq 3$	$\Longleftrightarrow$	Reach. in bounded 2-CA
Reach. in 2-clock TA	$\Longleftrightarrow$	Reach. in bounded 1-CA
Reach. in 1-clock TA	$\Longleftrightarrow$	Reach. in bounded 0-CA

Bounded Two-Counter Automata and *n*-Clock Timed Automata

Bounded Two-Counter Automata and n-Clock Timed Automata

bounded n-counter automata

$\Downarrow$

bounded two-counter automata

$\Downarrow$

n-clock timed automata, $n \geq 3$

$\Downarrow$

bounded $(2n+2)$-counter automata

Bounded Two-Counter Automata and n-Clock Timed Automata

bounded n-counter automata

$\Downarrow$

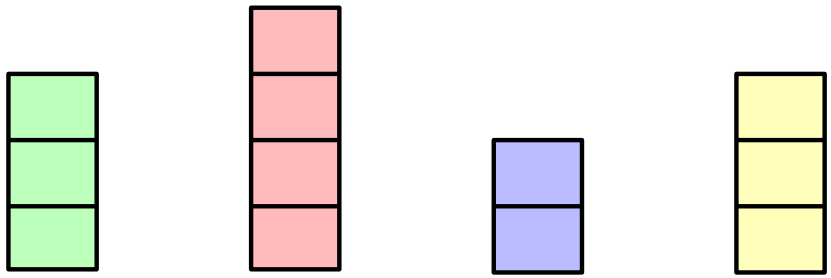
bounded two-counter automata

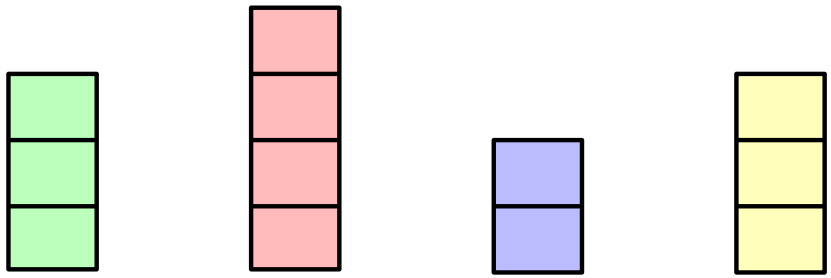
$\Downarrow$

n-clock timed automata, $n \geq 3$

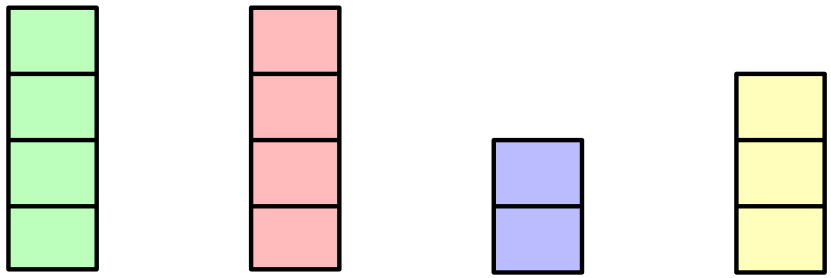
$\Downarrow$

bounded $(2n+2)$-counter automata

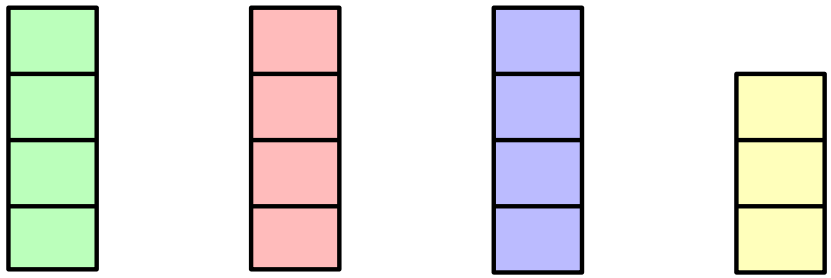




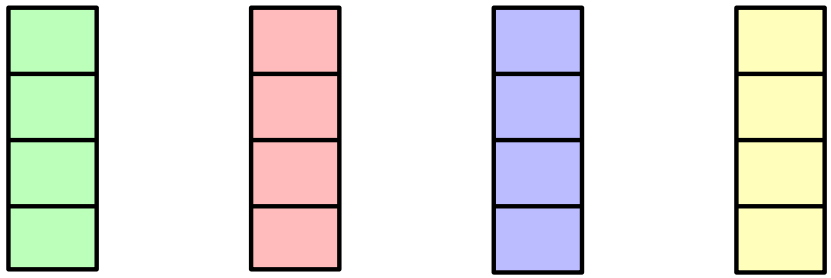
make bounds equal



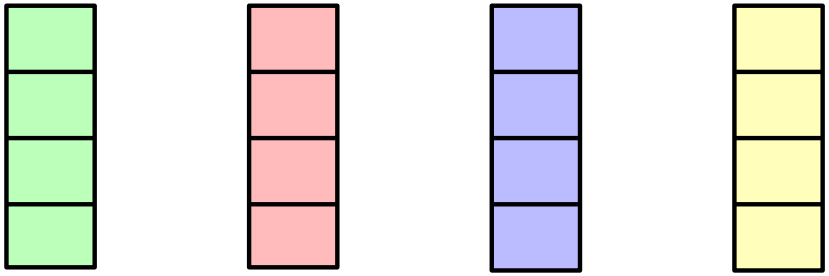
make bounds equal

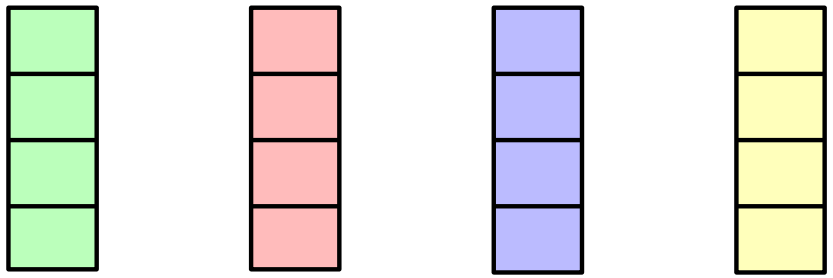


make bounds equal

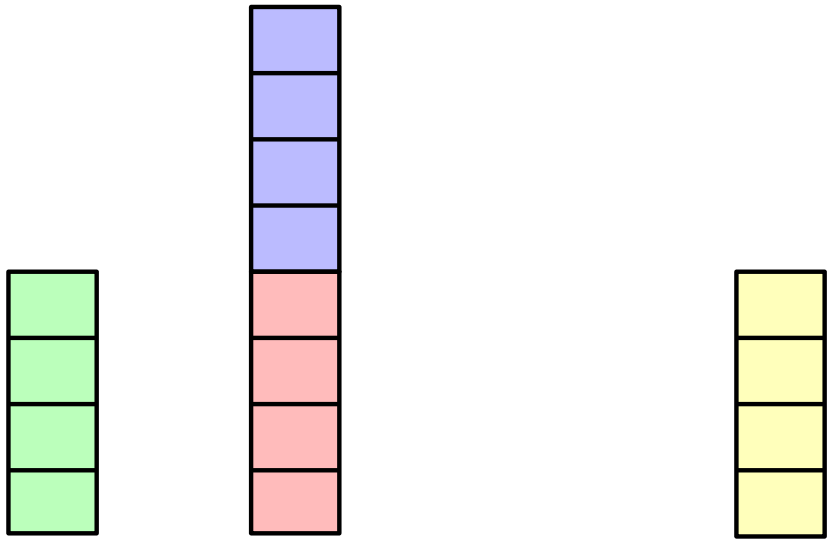


make bounds equal

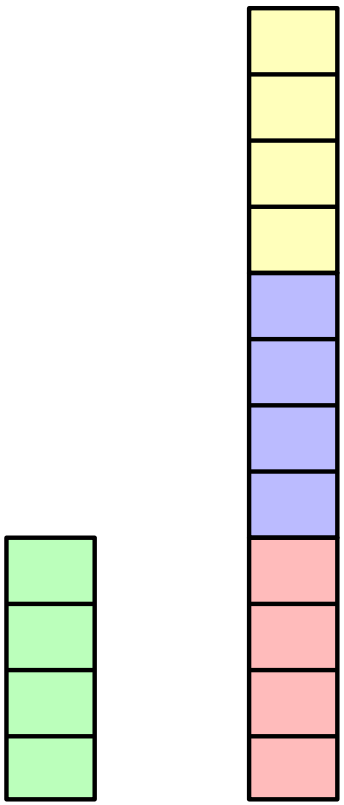




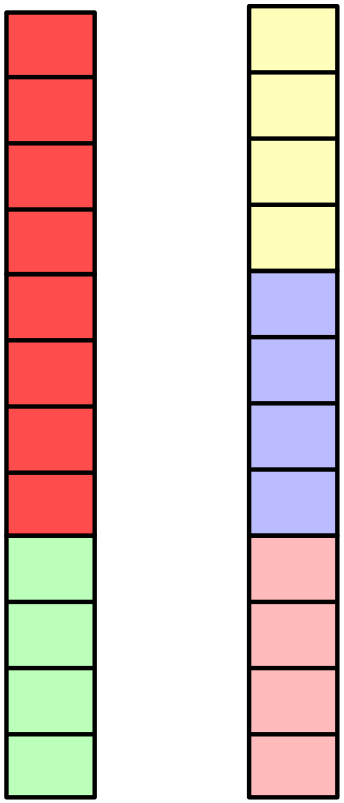
encode additional counters into second counter



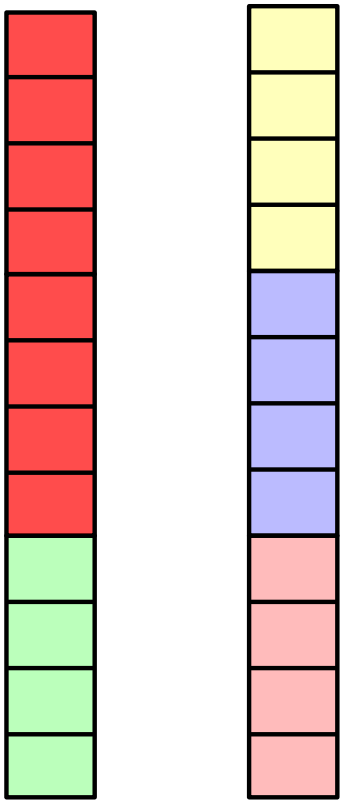
encode additional counters into second counter

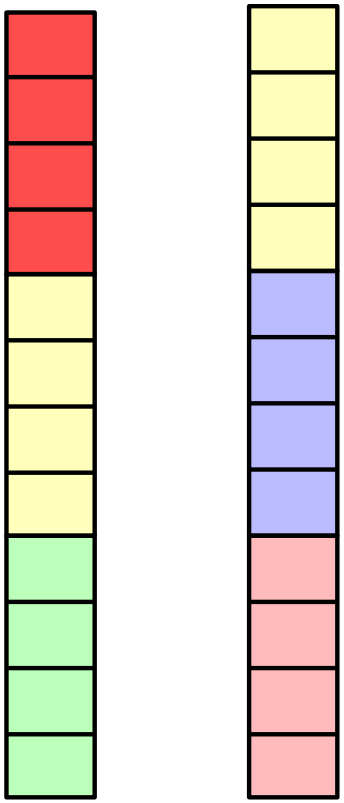


encode additional counters into second counter

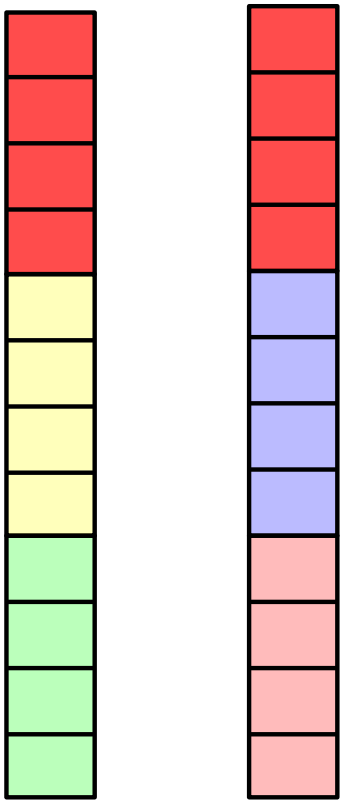


“reserve” temporary storage on first counter

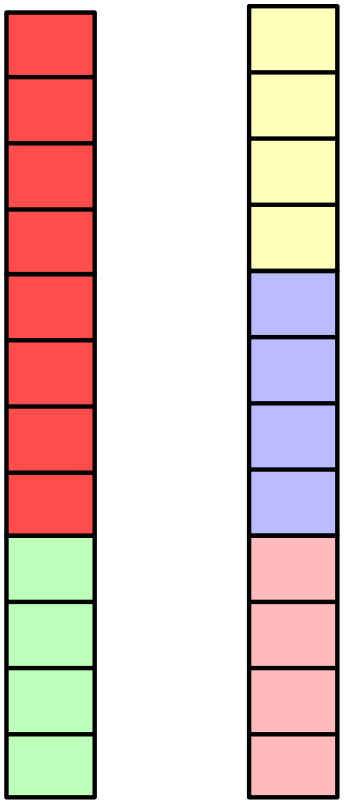




move “higher” counter values to temporary storage



block “upper” bits and simulate operation



move temporarily stored counters back

Bounded Two-Counter Automata and n-Clock Timed Automata

bounded n-counter automata

$\Downarrow$

bounded two-counter automata

$\Downarrow$

n-clock timed automata, $n \geq 3$

$\Downarrow$

bounded $(2n+2)$-counter automata

Bounded Two-Counter Automata and n-Clock Timed Automata

bounded n-counter automata

$\Downarrow$

bounded two-counter automata

$\Downarrow$

n-clock timed automata, $n \geq 3$

$\Downarrow$

bounded $(2n+2)$-counter automata

Simulating Bounded Two-Counter Automata with Timed Automata

Main idea:

- Assume uniform bound b on two counters
- Store values of counters in difference of clock values
- If $x = b$ then $x - y$ represents value of the first counter and $x - z$ the value of the second counter
- Replace in- and decrements by gadgets

Simulating Bounded Two-Counter Automata with Timed Automata

Main idea:

- Assume uniform bound b on two counters
- Store values of counters in difference of clock values
- If $x = b$ then $x - y$ represents value of the first counter and $x - z$ the value of the second counter
- Replace in- and decrements by gadgets

Simulating Bounded Two-Counter Automata with Timed Automata

Main idea:

- Assume uniform bound b on two counters
- Store values of counters in difference of clock values
- If $x = b$ then $x - y$ represents value of the first counter and $x - z$ the value of the second counter
- Replace in- and decrements by gadgets

Simulating Bounded Two-Counter Automata with Timed Automata

Main idea:

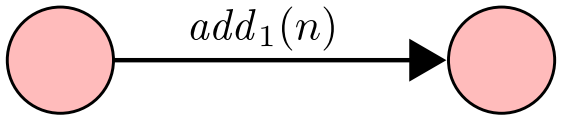
- Assume uniform bound b on two counters
- Store values of counters in difference of clock values
- If $x = b$ then $x - y$ represents value of the first counter and $x - z$ the value of the second counter
- Replace in- and decrements by gadgets

Simulating Bounded Two-Counter Automata with Timed Automata

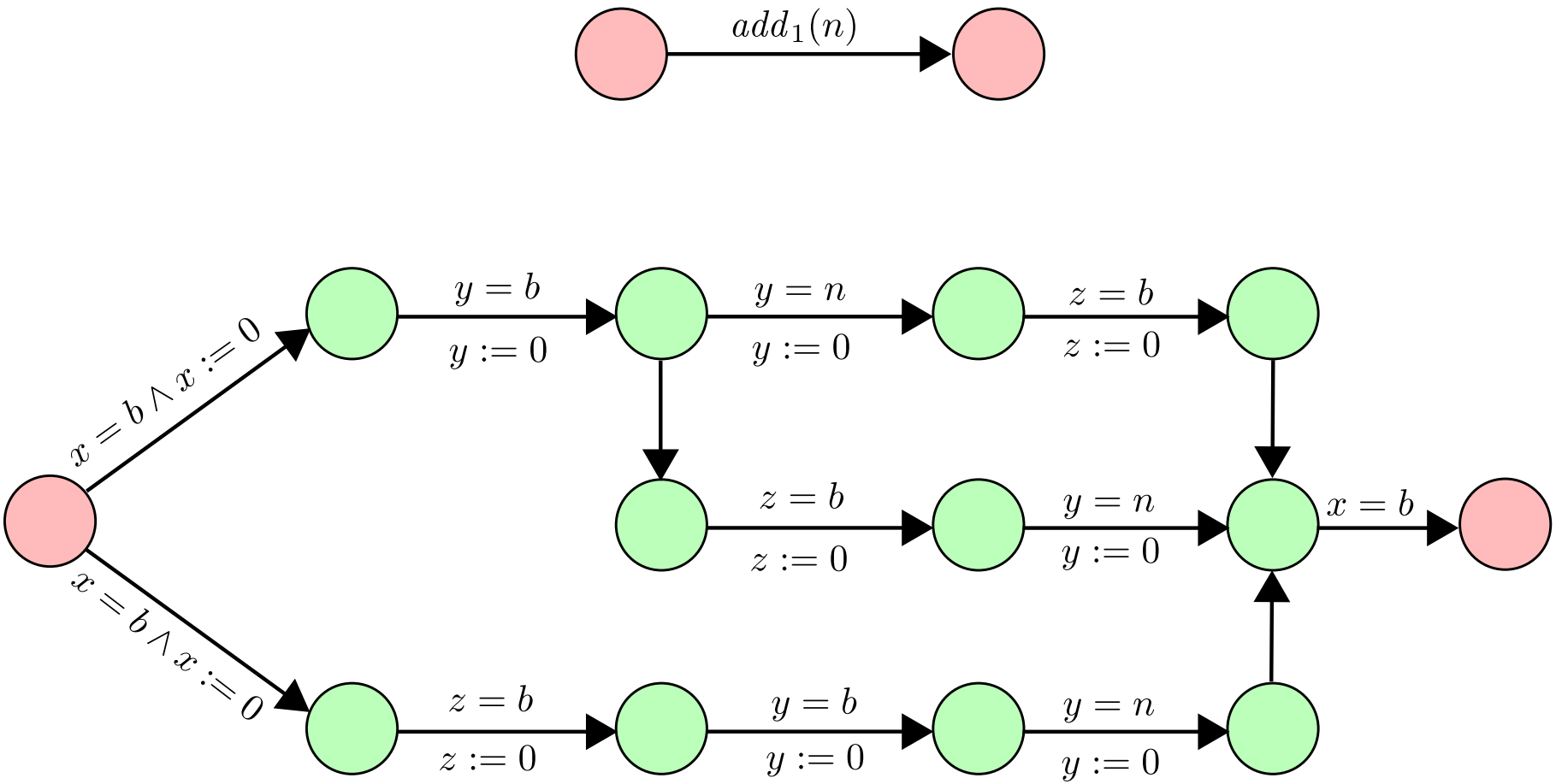
Main idea:

- Assume uniform bound b on two counters
- Store values of counters in difference of clock values
- If $x = b$ then $x - y$ represents value of the first counter and $x - z$ the value of the second counter
- Replace in- and decrements by gadgets

Simulating Bounded Two-Counter Automata with Timed Automata



Simulating Bounded Two-Counter Automata with Timed Automata



Bounded Two-Counter Automata and n-Clock Timed Automata

bounded n-counter automata

$\Downarrow$

bounded two-counter automata

$\Downarrow$

n-clock timed automata, $n \geq 3$

$\Downarrow$

bounded $(2n+2)$-counter automata

Bounded Two-Counter Automata and n-Clock Timed Automata

bounded n-counter automata

$\Downarrow$

bounded two-counter automata

$\Downarrow$

n-clock timed automata, $n \geq 3$

$\Downarrow$

bounded $(2n+2)$-counter automata

Simulating n-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters

Simulating n-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters

Simulating n-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters
- Region abstraction treats two configurations as equivalent if
  - **(a)** their control locations are the same
  - **(b)** the integral parts of each clock with a value below the maximum constant are the same
  - **(c)** the relative orders of the fractional parts of the values of the clocks are the same
  - **(d)** the clocks with fractional part 0 are the same

Simulating n-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters
- Region abstraction treats two configurations as equivalent if
  - **(a)** their control locations are the same
  - **(b)** the integral parts of each clock with a value below the maximum constant are the same
  - **(c)** the relative orders of the fractional parts of the values of the clocks are the same
  - **(d)** the clocks with fractional part 0 are the same

Simulating *n*-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters
- Region abstraction treats two configurations as equivalent if
  - **(a)** their control locations are the same
  - **(b)** the integral parts of each clock with a value below the maximum constant are the same
  - **(c)** the relative orders of the fractional parts of the values of the clocks are the same
  - **(d)** the clocks with fractional part 0 are the same

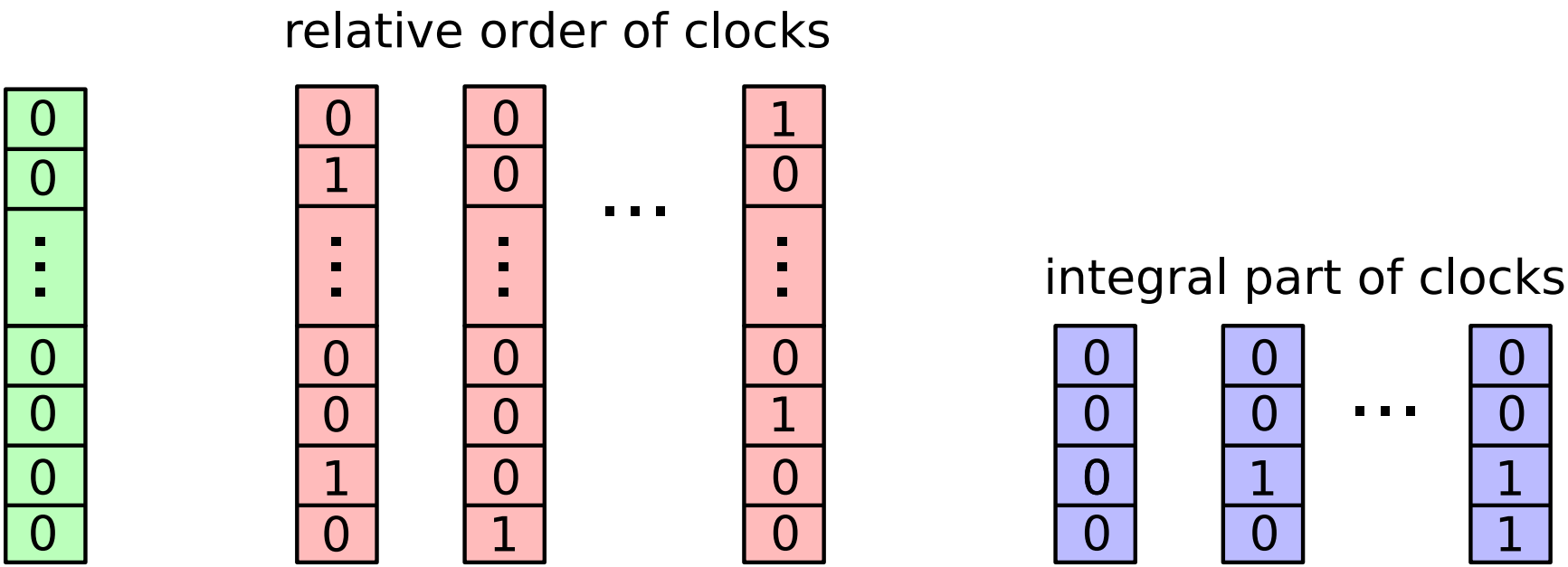
Simulating n-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters
- Region abstraction treats two configurations as equivalent if
  - **(a)** their control locations are the same
  - **(b)** the integral parts of each clock with a value below the maximum constant are the same
  - **(c)** the relative orders of the fractional parts of the values of the clocks are the same
  - **(d)** the clocks with fractional part 0 are the same

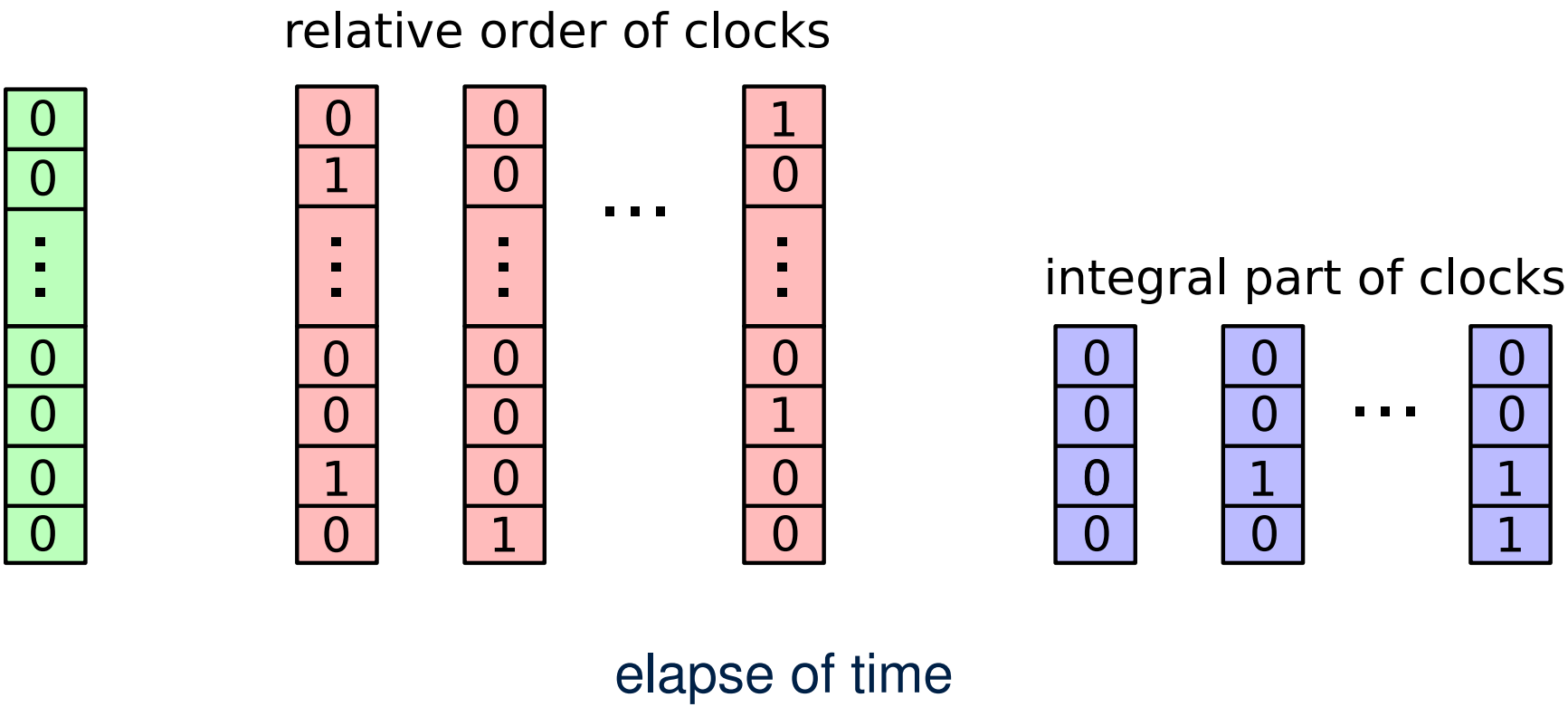
Simulating n-Clock Timed Automata with Bounded Counter Automata

- Main idea: simulate region abstraction on the counters
- Region abstraction treats two configurations as equivalent if
  - **(a)** their control locations are the same
  - **(b)** the integral parts of each clock with a value below the maximum constant are the same
  - **(c)** the relative orders of the fractional parts of the values of the clocks are the same
  - **(d)** the clocks with fractional part 0 are the same

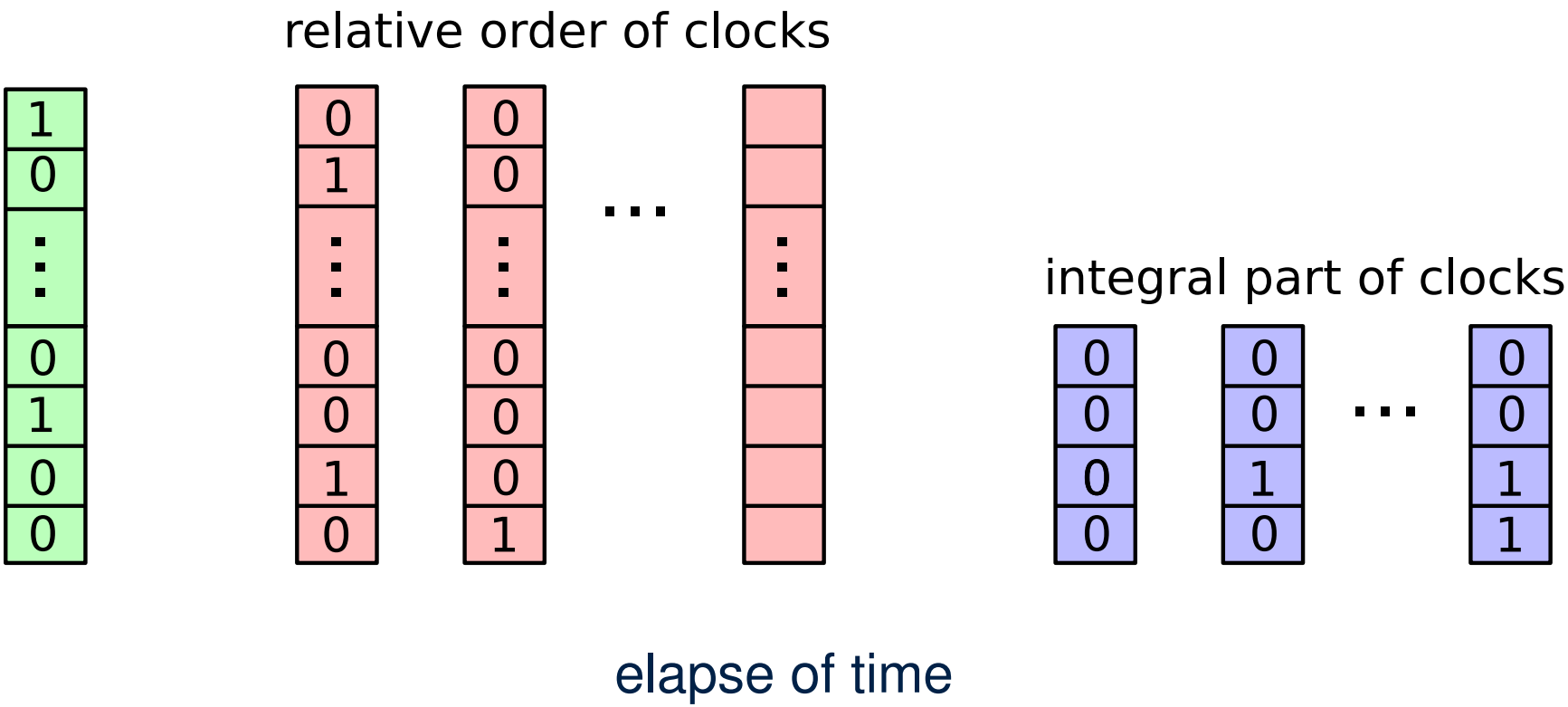
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



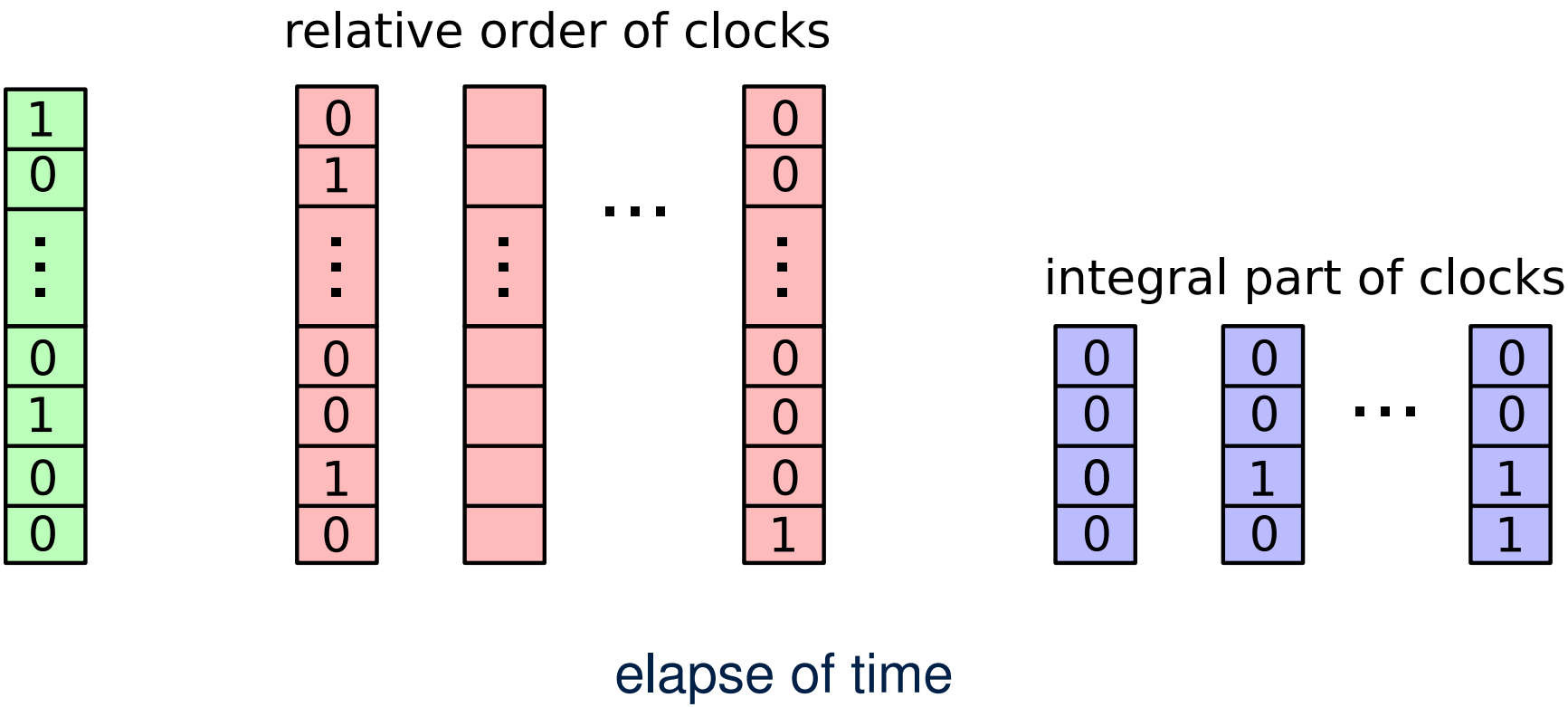
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



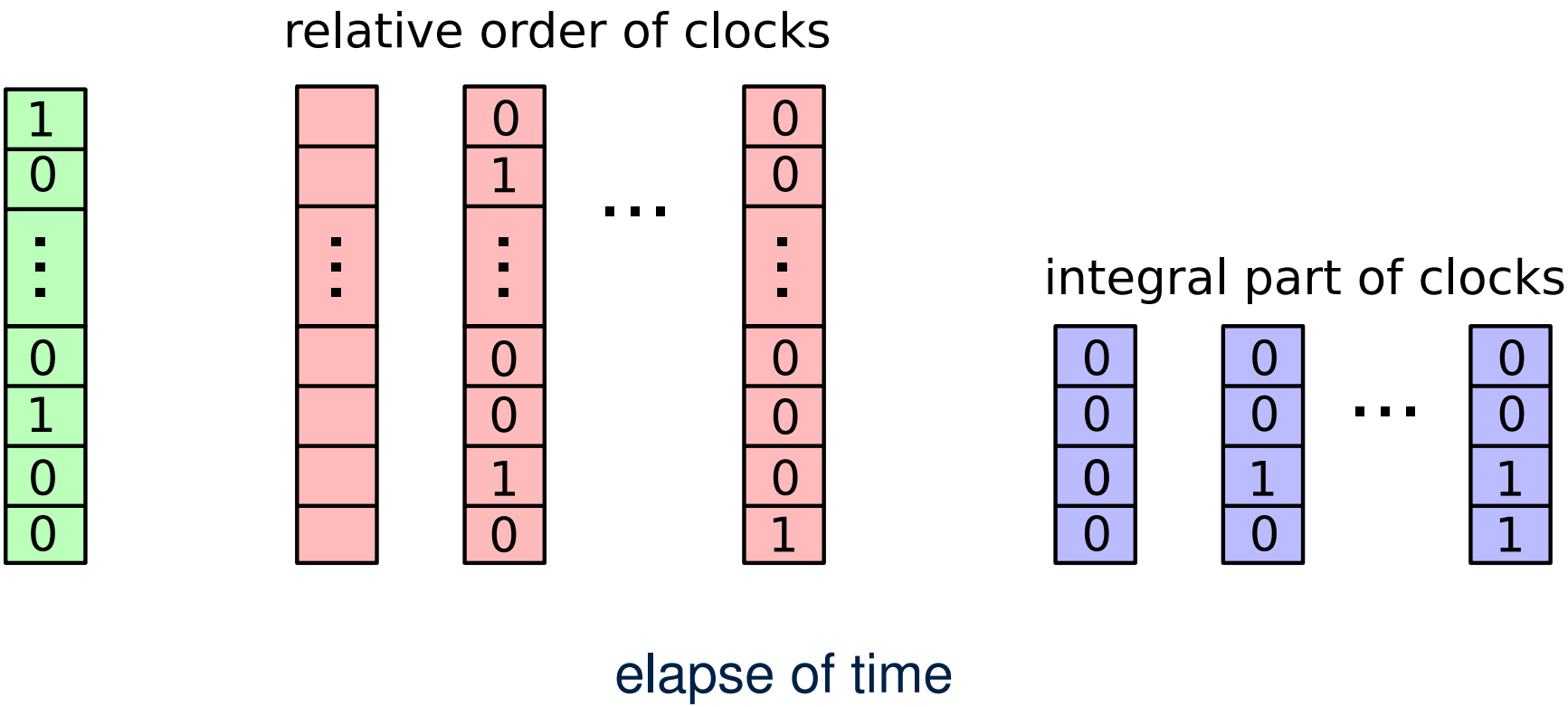
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



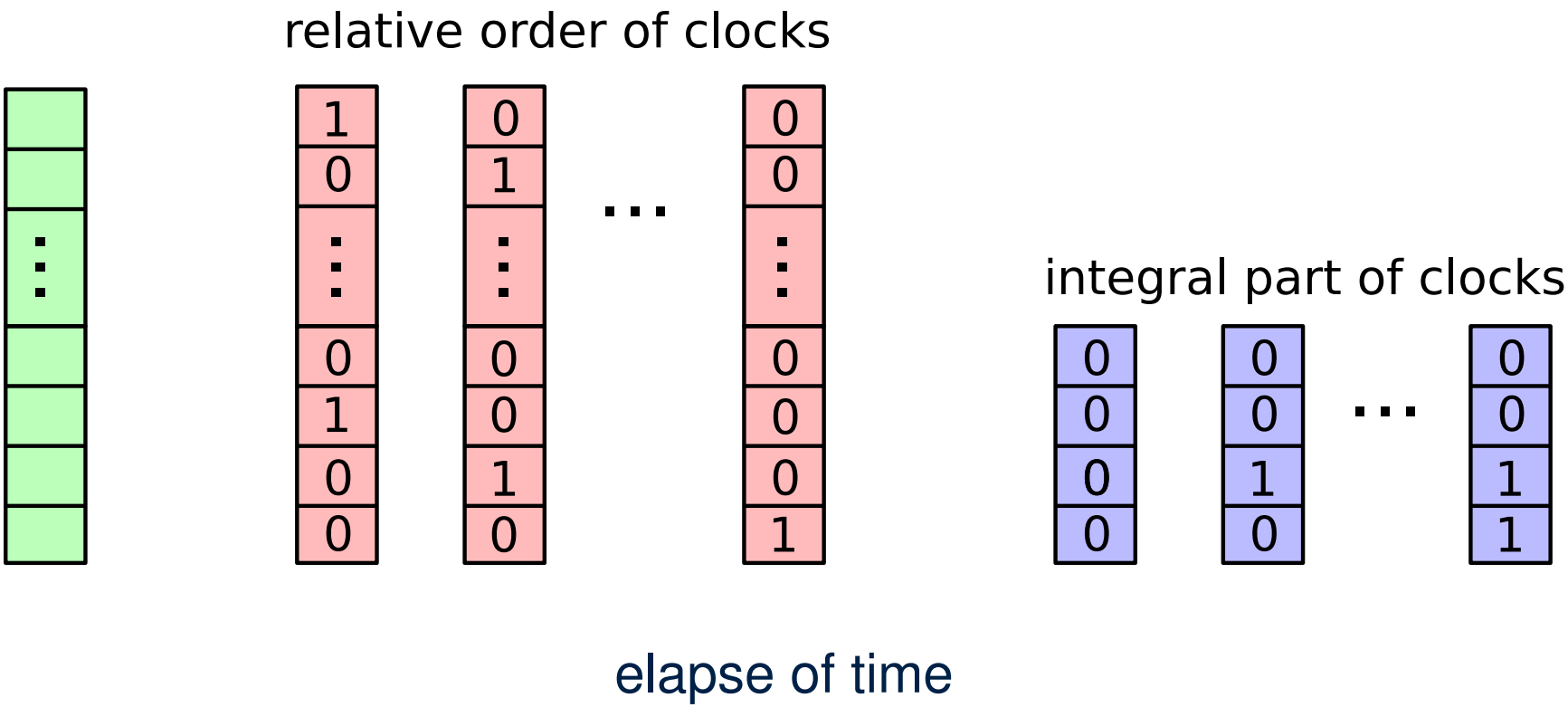
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



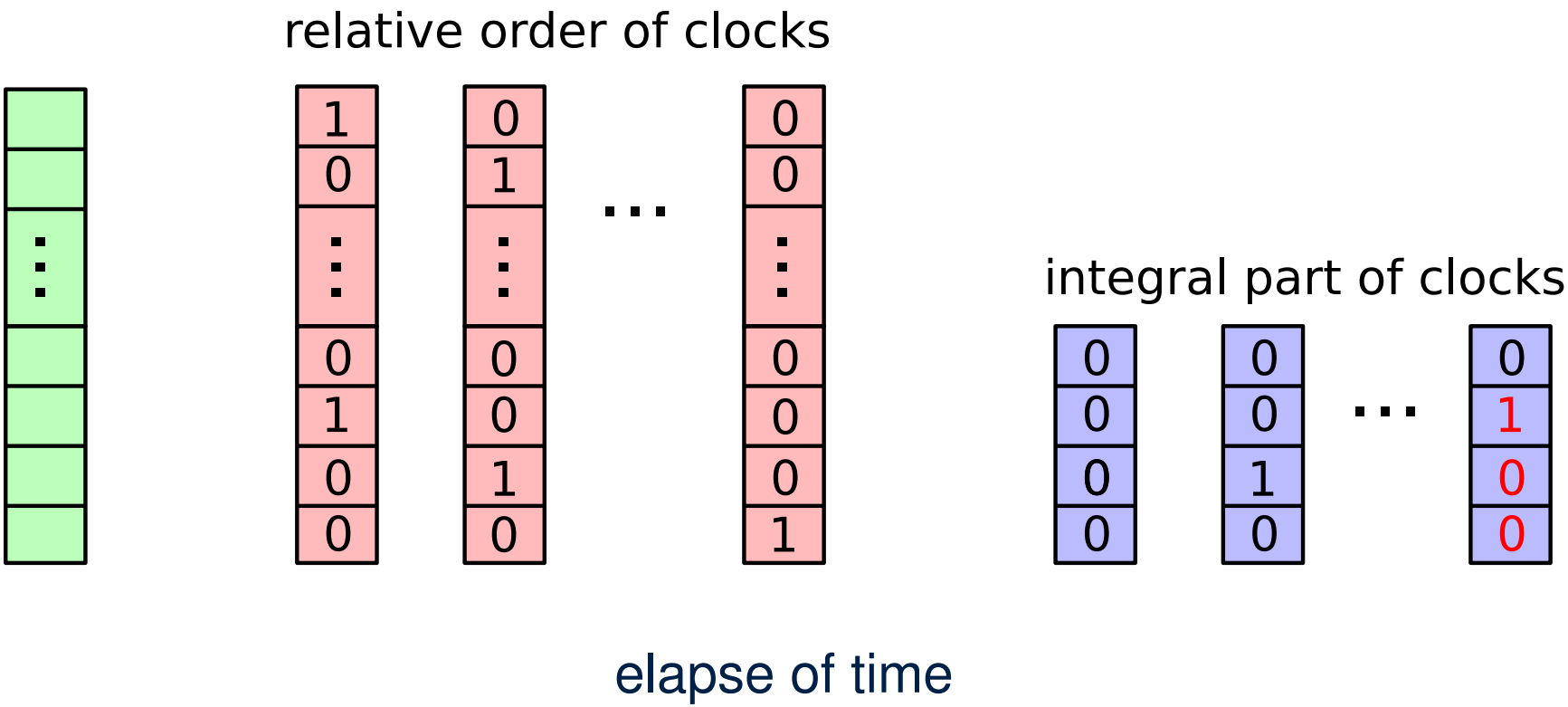
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



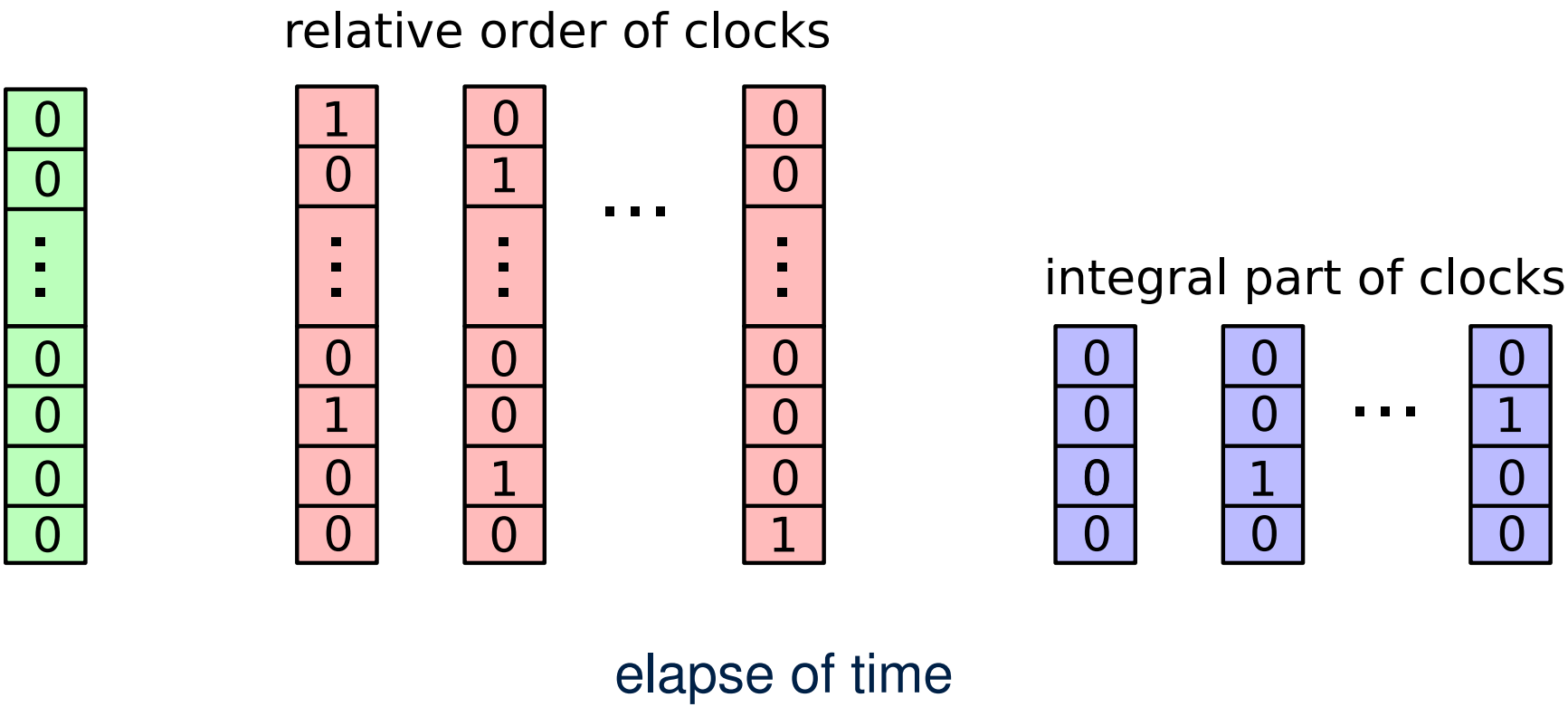
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



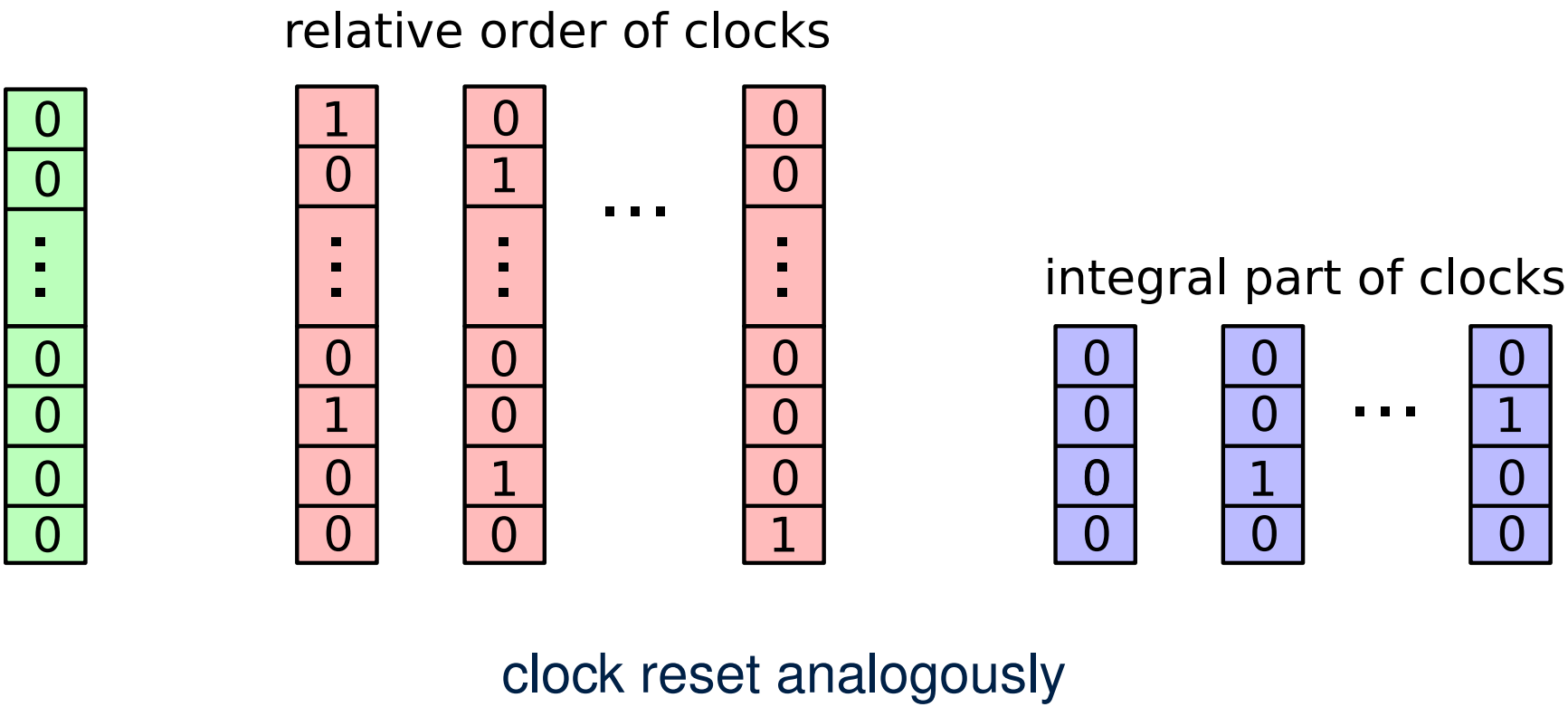
Simulating *n*-Clock Timed Automata with Bounded Counter Automata



Simulating *n*-Clock Timed Automata with Bounded Counter Automata



Simulating *n*-Clock Timed Automata with Bounded Counter Automata



Theorem

Reachability in k-clock timed automata with $k \geq 3$ is logarithmic-space inter-reducible with reachability in bounded two-counter automata.

Theorem

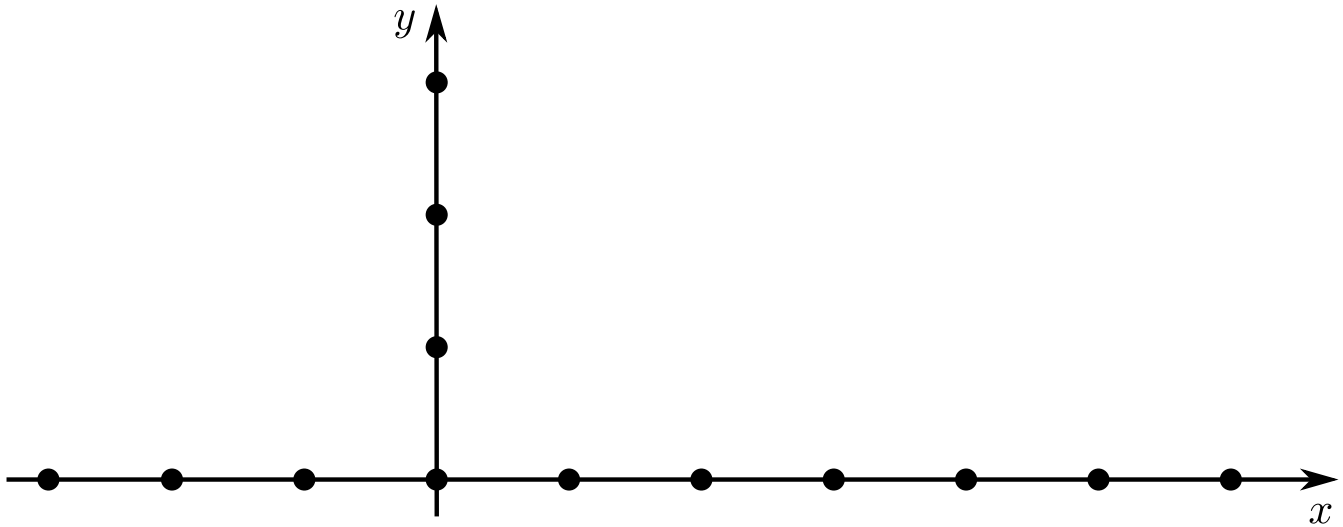
Reachability in k-clock timed automata with $k \geq 3$ is logarithmic-space inter-reducible with reachability in bounded two-counter automata.

Corollary

Reachability in bounded k-counter automata is PSPACE*-complete for $k \geq 2$.*

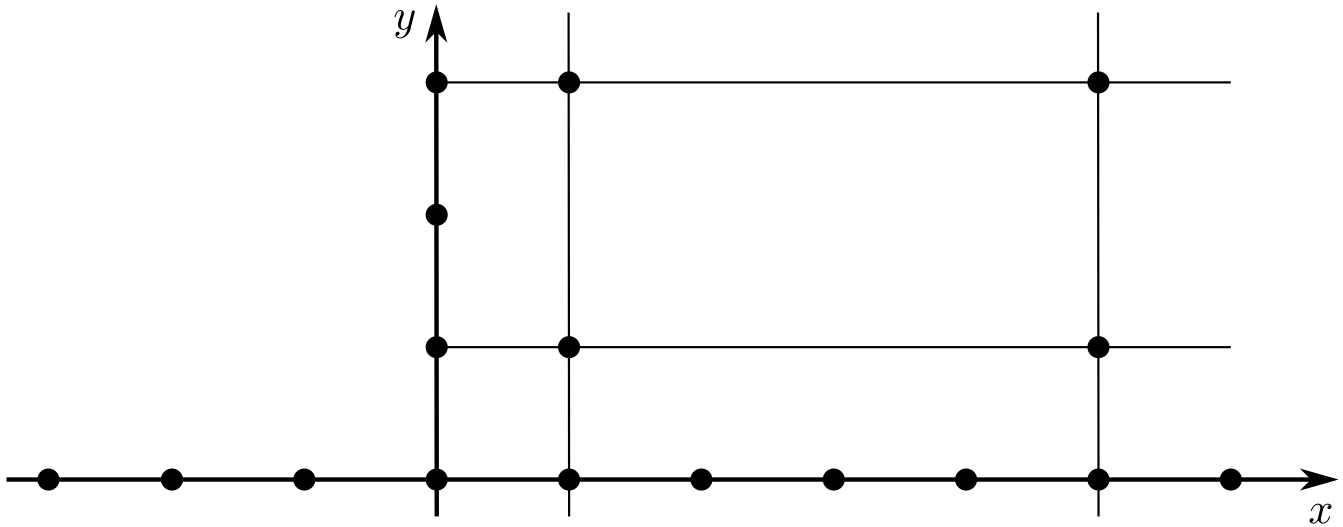
Bounded One-Counter Automata and Two-Clock Timed Automata

Two-Clock Timed Automata to Bounded One-Counter Automata



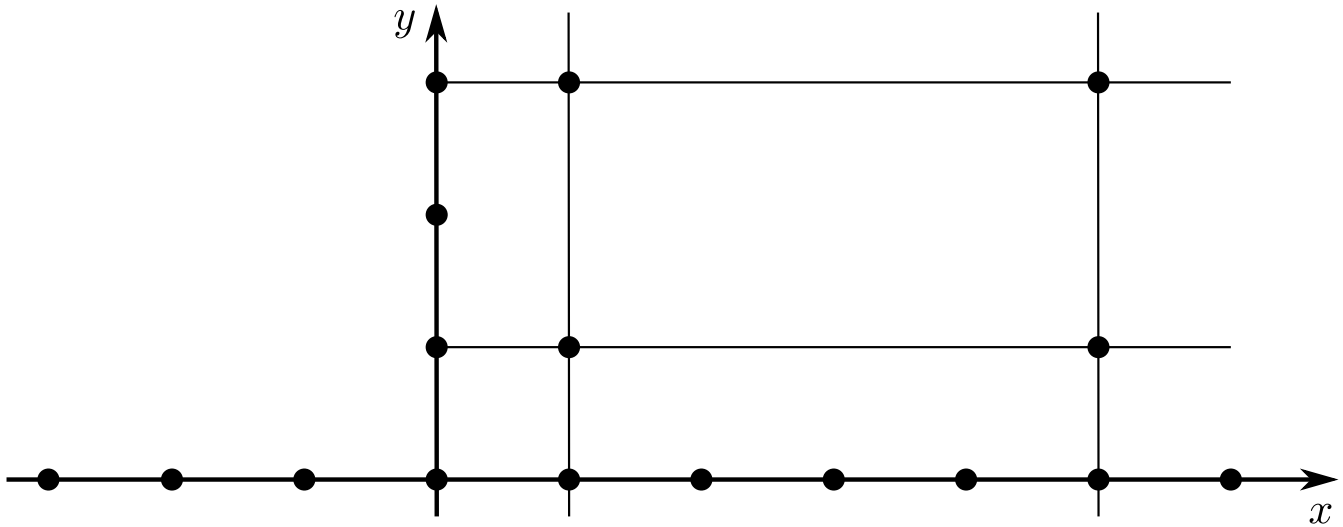
Given a timed automaton with x-constants $\{0, 1, 5\}$ and y-constants $\{0, 1, 3\}$

Two-Clock Timed Automata to Bounded One-Counter Automata



Given a timed automaton with x-constants $\{0, 1, 5\}$ and y-constants $\{0, 1, 3\}$

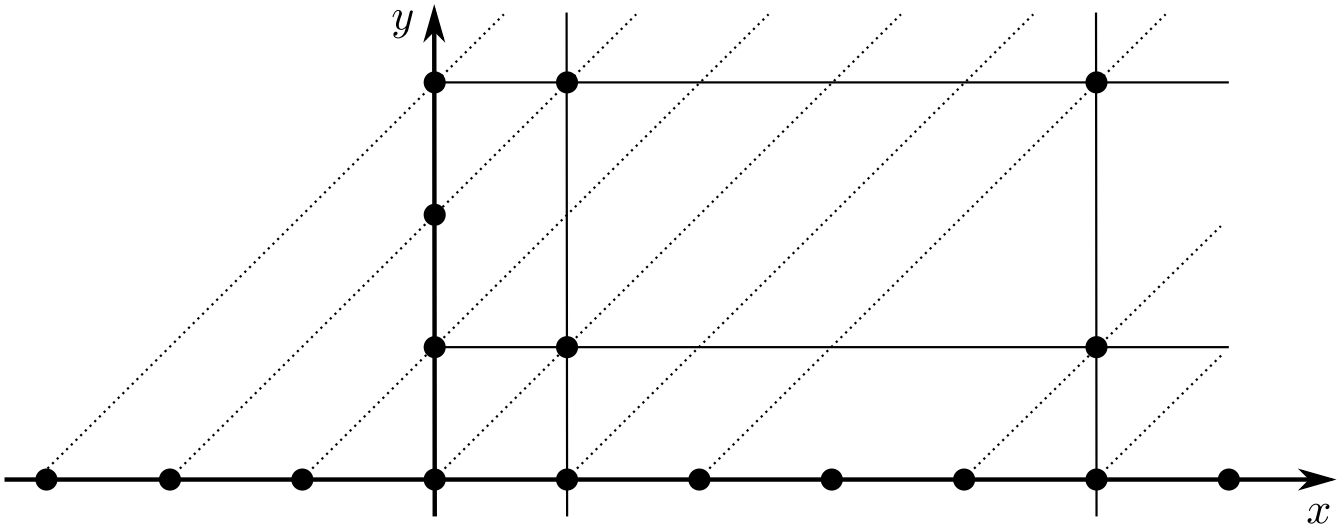
Two-Clock Timed Automata to Bounded One-Counter Automata



Given a timed automaton with x-constants $\{0, 1, 5\}$ and y-constants $\{0, 1, 3\}$

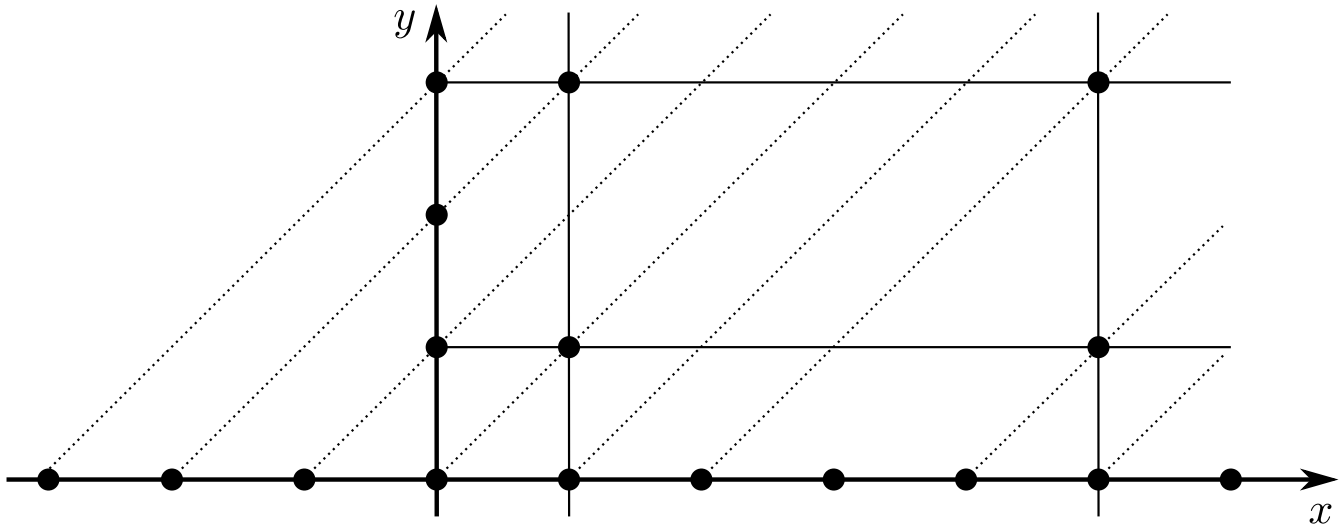
$\rightsquigarrow$ regions of the automaton

Two-Clock Timed Automata to Bounded One-Counter Automata



Given a timed automaton with x-constants $\{0, 1, 5\}$ and y-constants $\{0, 1, 3\}$

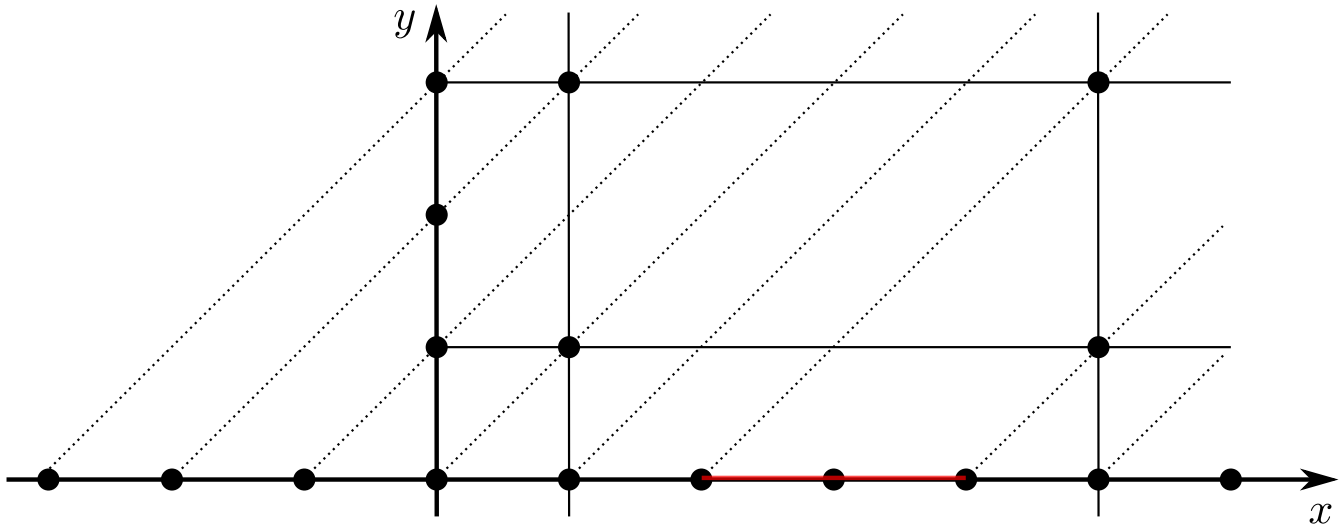
Two-Clock Timed Automata to Bounded One-Counter Automata



Given a timed automaton with x-constants $\{0, 1, 5\}$ and y-constants $\{0, 1, 3\}$

$\rightsquigarrow$ regions and clock difference zones of the automaton

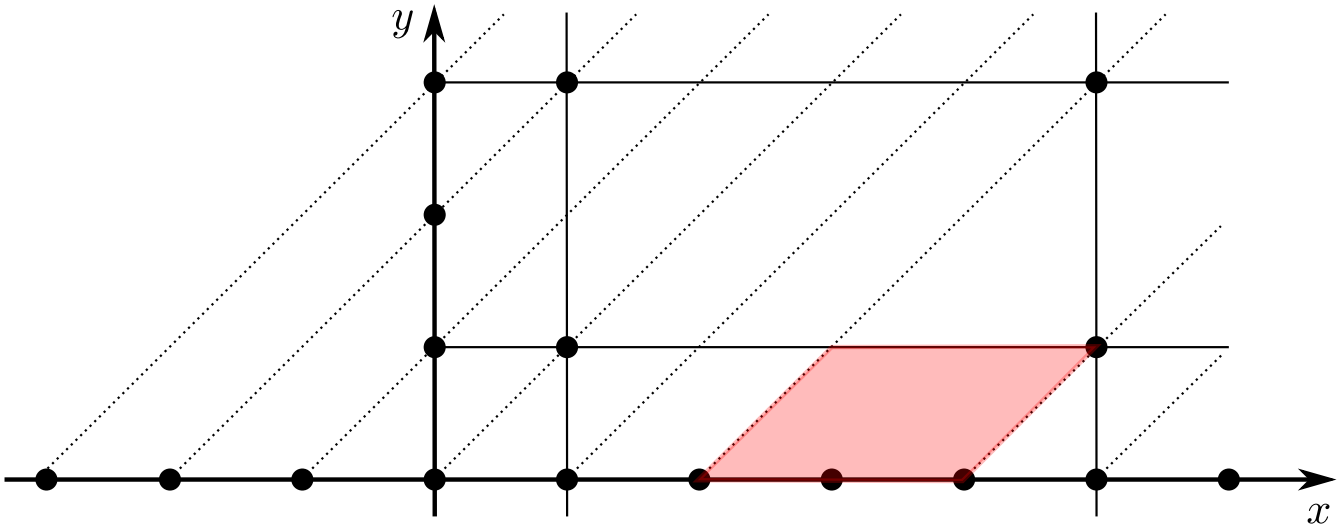
Two-Clock Timed Automata to Bounded One-Counter Automata



Elapse of time

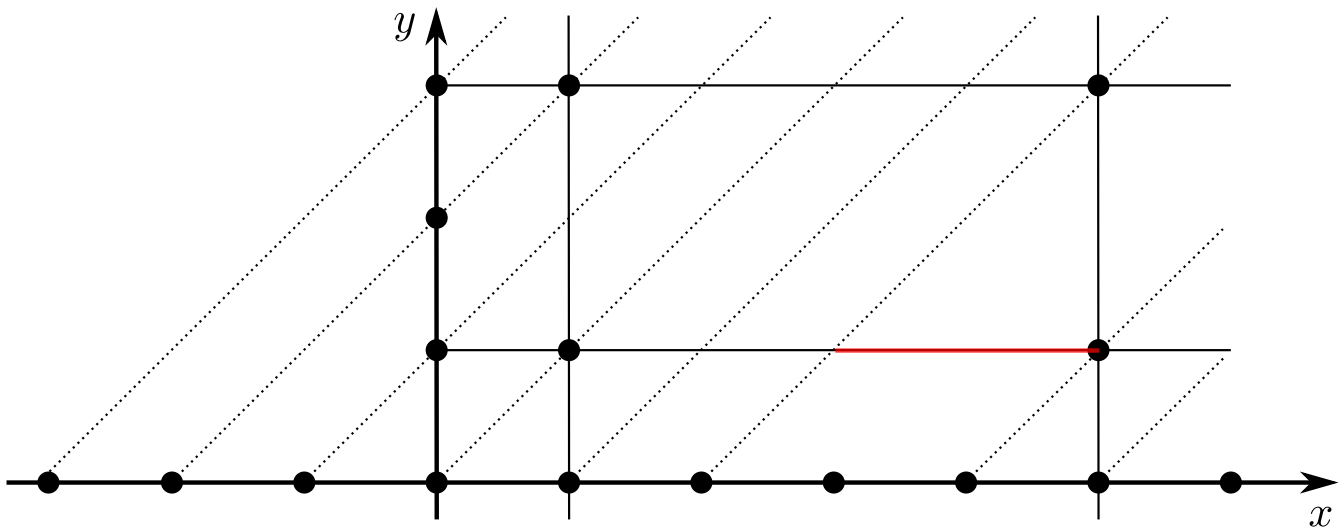
$x \in (2, 3), y \in [0, 0]$

Two-Clock Timed Automata to Bounded One-Counter Automata



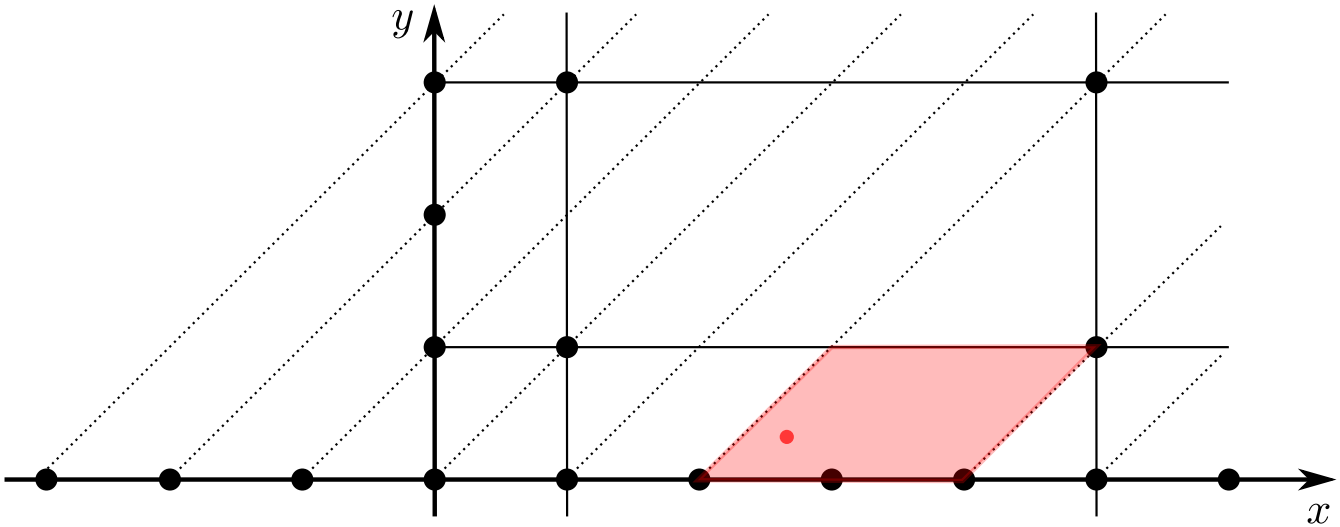
Elapse of time
$x \in (2,3), y \in (0,1)$

Two-Clock Timed Automata to Bounded One-Counter Automata



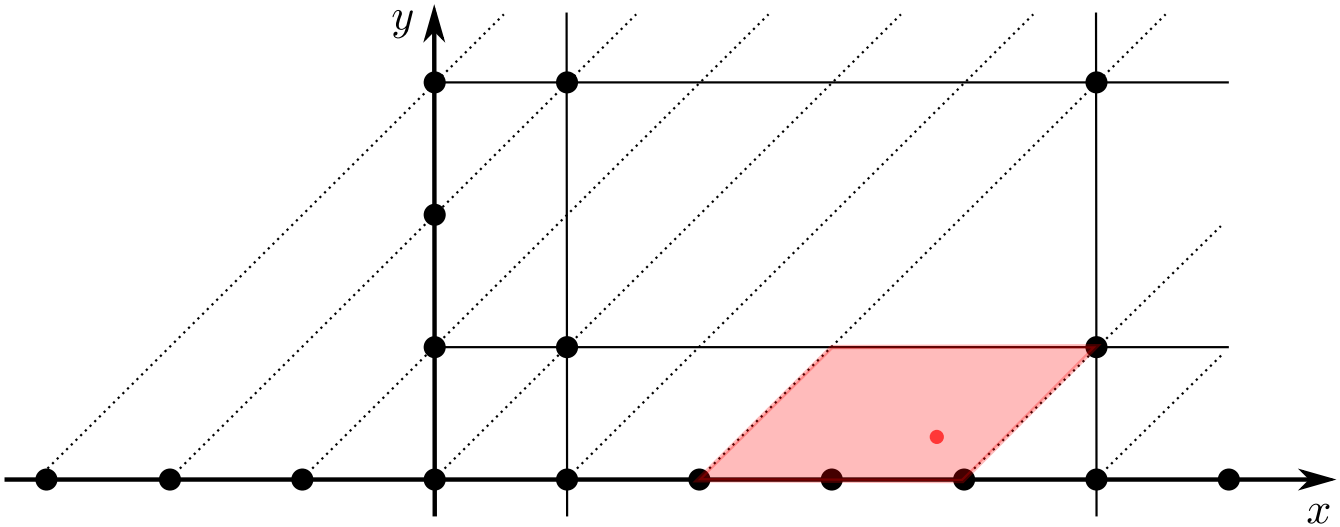
Elapse of time
$x \in (2, 3), y \in [1, 1]$

Two-Clock Timed Automata to Bounded One-Counter Automata



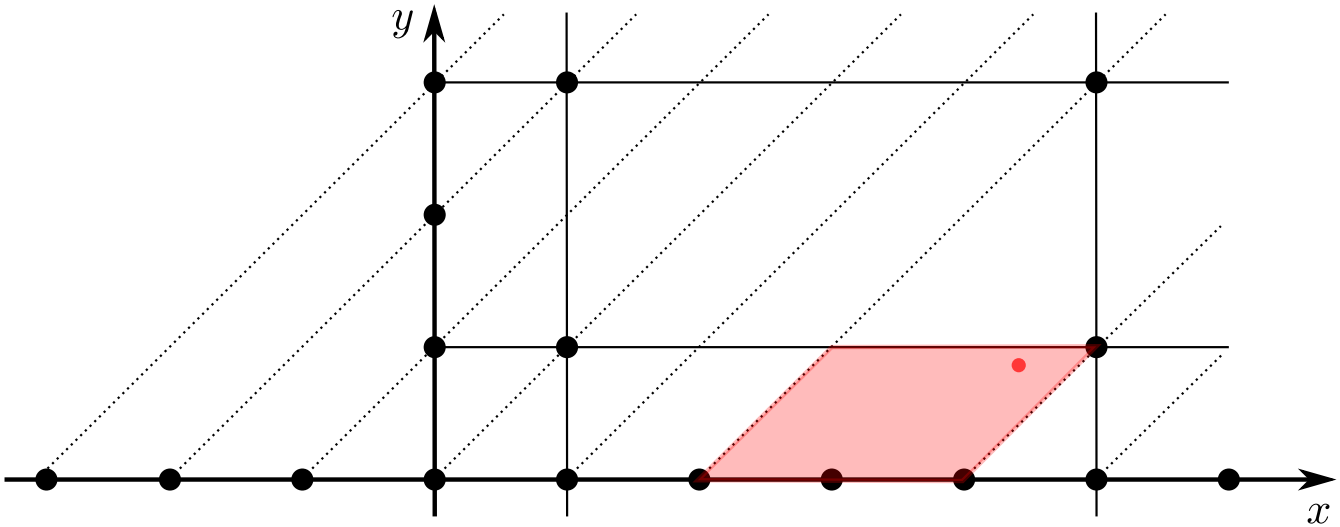
Regions and clock difference zones are too coarse to fully capture reachability properties

Two-Clock Timed Automata to Bounded One-Counter Automata



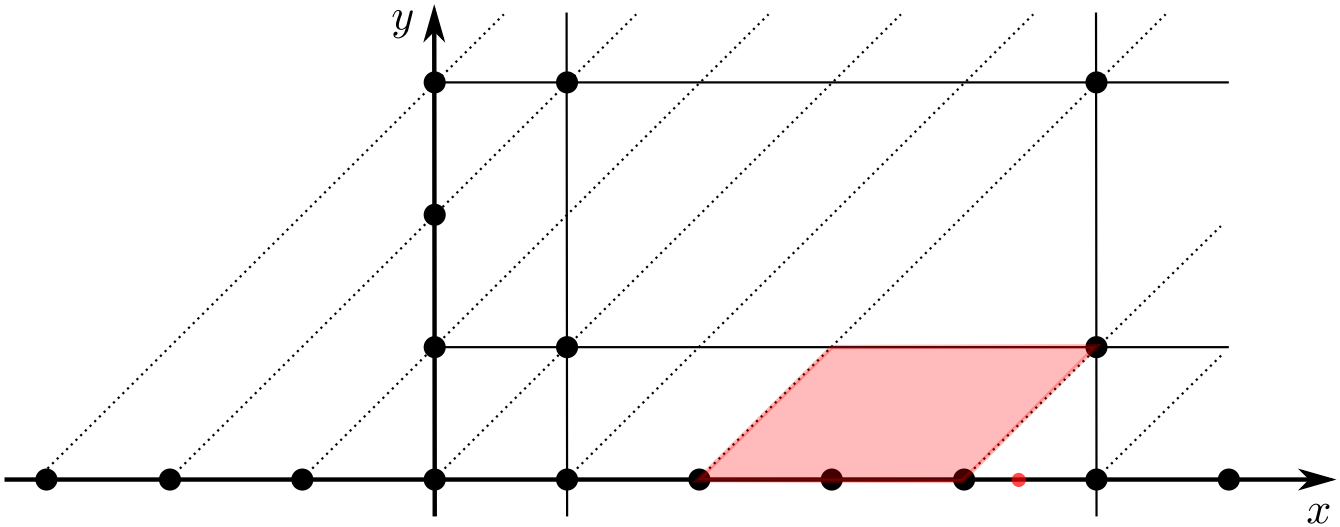
Regions and clock difference zones are too coarse to fully capture reachability properties

Two-Clock Timed Automata to Bounded One-Counter Automata



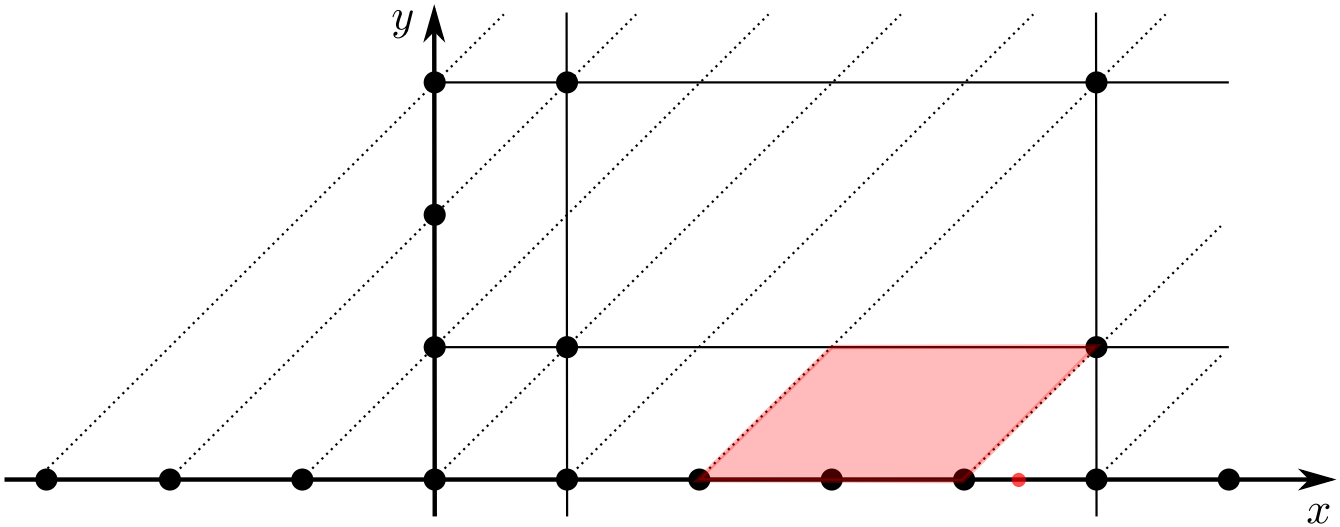
Regions and clock difference zones are too coarse to fully capture reachability properties

Two-Clock Timed Automata to Bounded One-Counter Automata



Regions and clock difference zones are too coarse to fully capture reachability properties

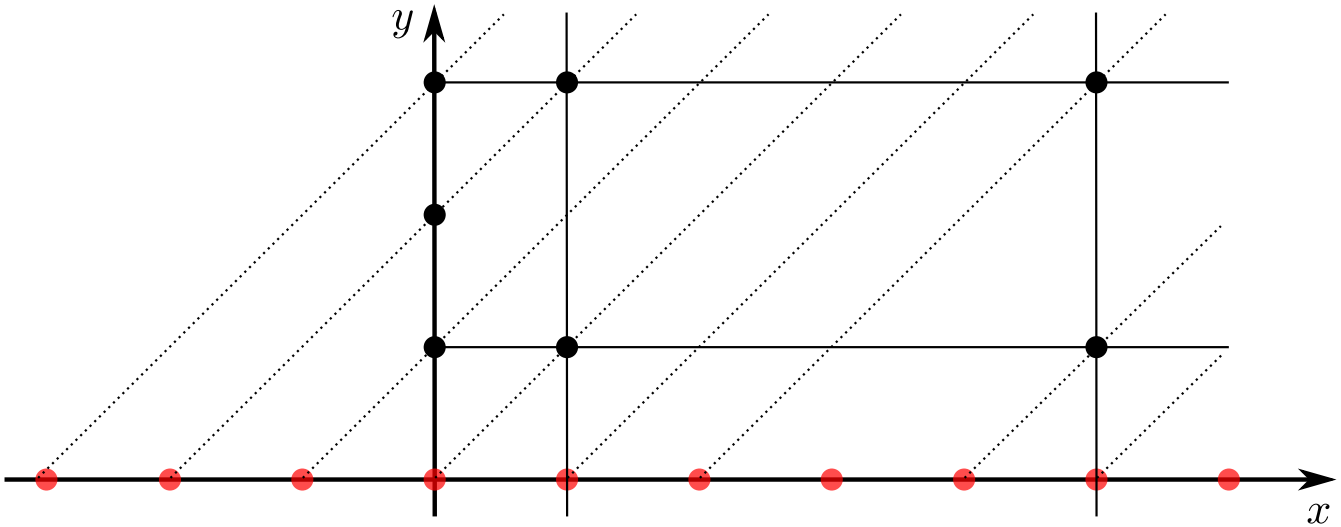
Two-Clock Timed Automata to Bounded One-Counter Automata



Regions and clock difference zones are too coarse to fully capture reachability properties

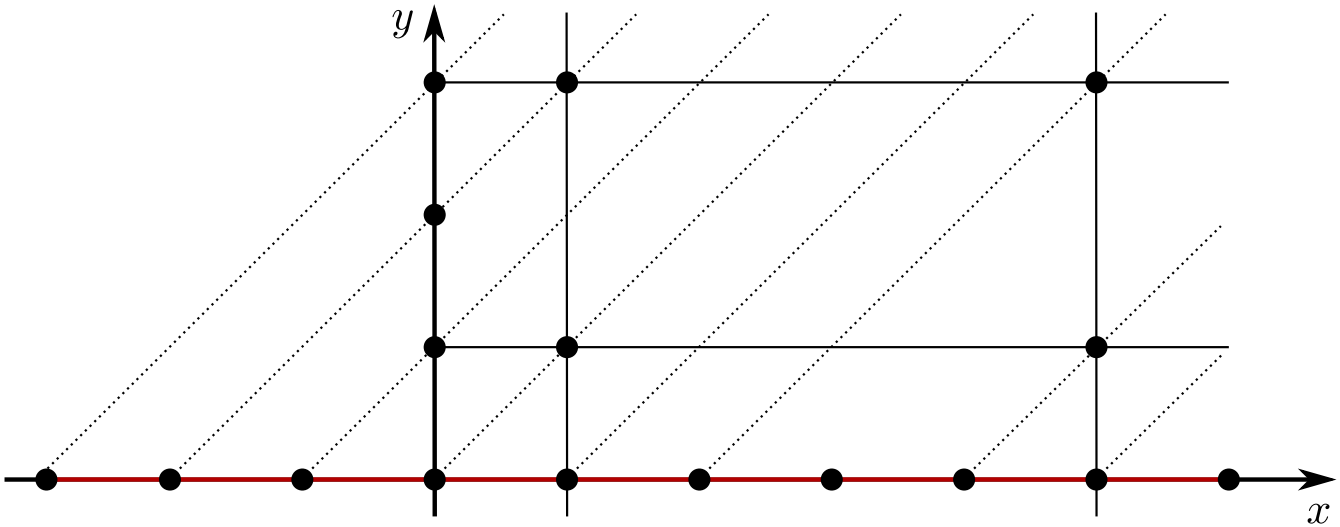
$\rightsquigarrow$ use counter in order to store difference between x and y

Two-Clock Timed Automata to Bounded One-Counter Automata



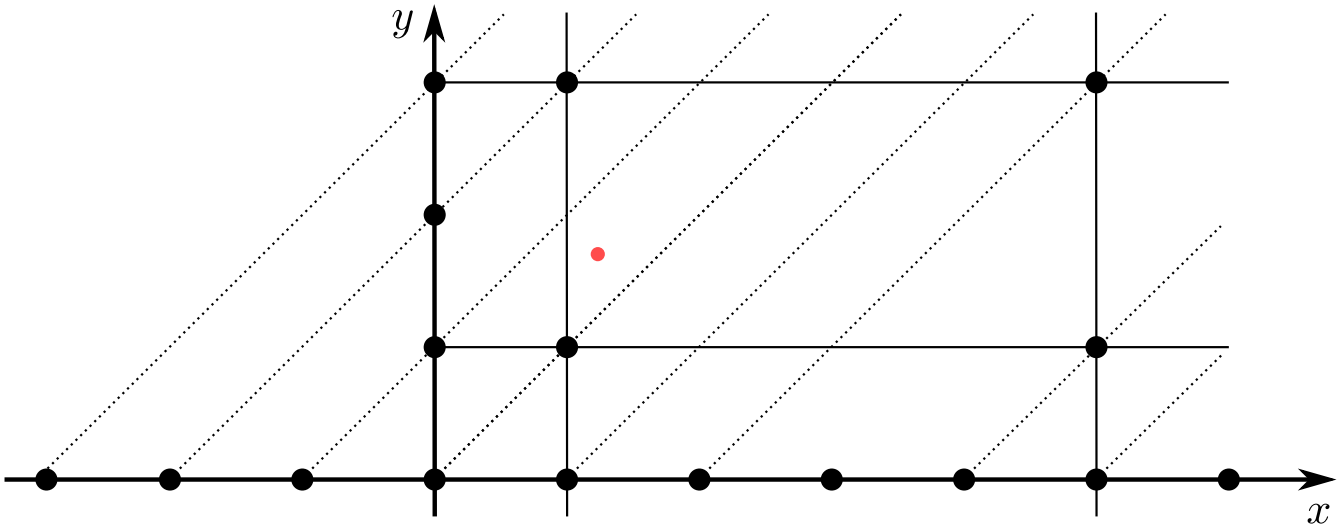
Counter encodes the difference between clocks when it is an integral value...

Two-Clock Timed Automata to Bounded One-Counter Automata



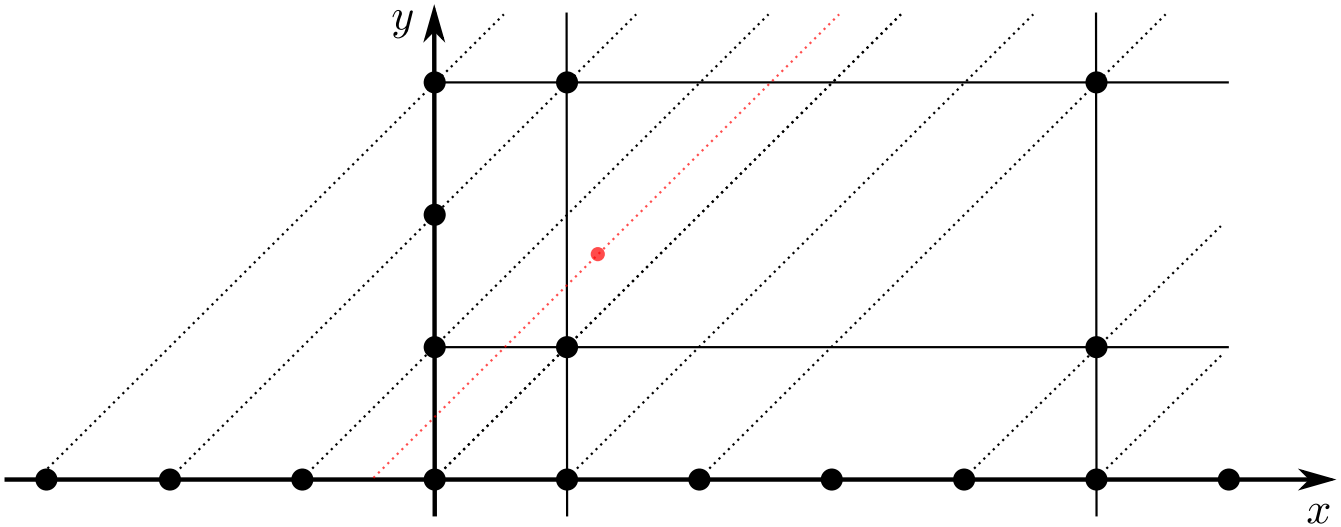
...and two consecutive integers if the difference between clocks lies in between those integral values

Two-Clock Timed Automata to Bounded One-Counter Automata

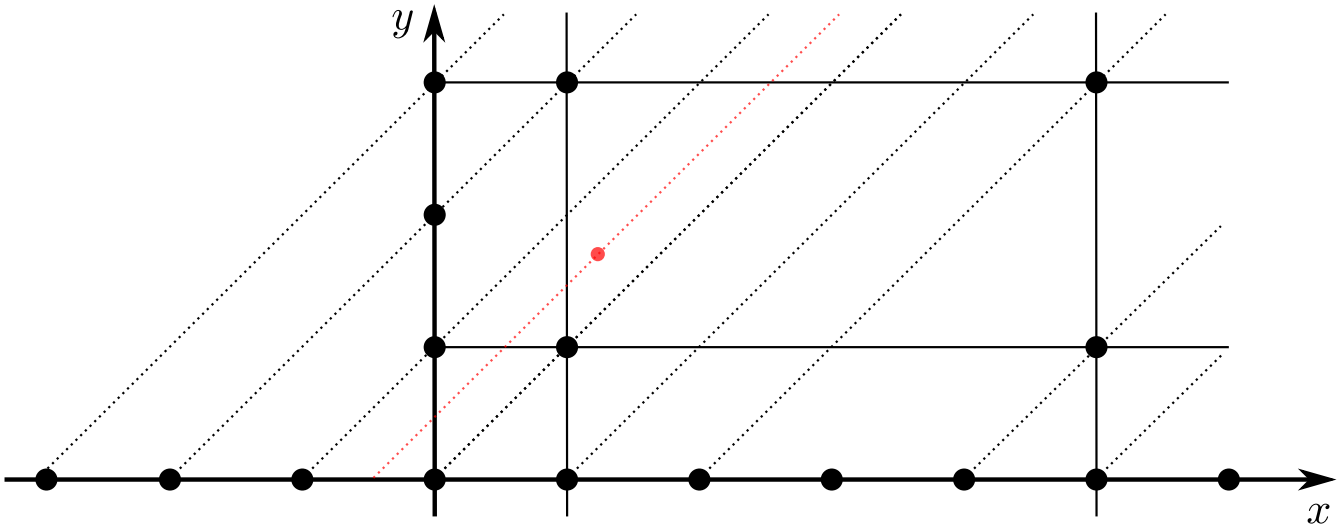


Suppose we wish to reset clock *y* only

Two-Clock Timed Automata to Bounded One-Counter Automata

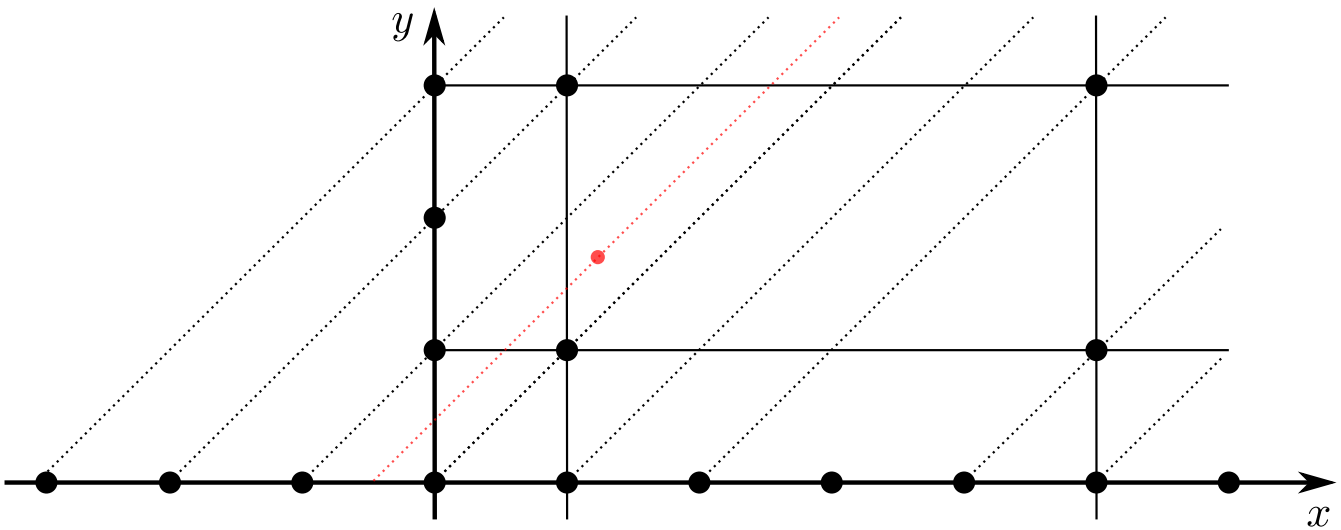


Two-Clock Timed Automata to Bounded One-Counter Automata



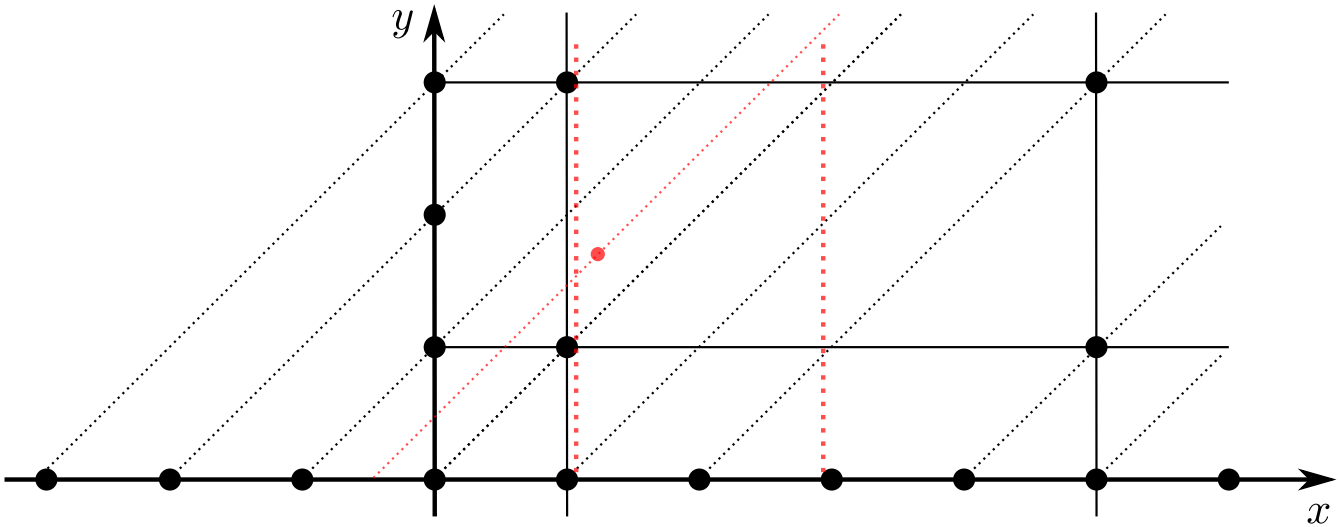
- Resulting counter must be smaller than $z + y_u$

Two-Clock Timed Automata to Bounded One-Counter Automata



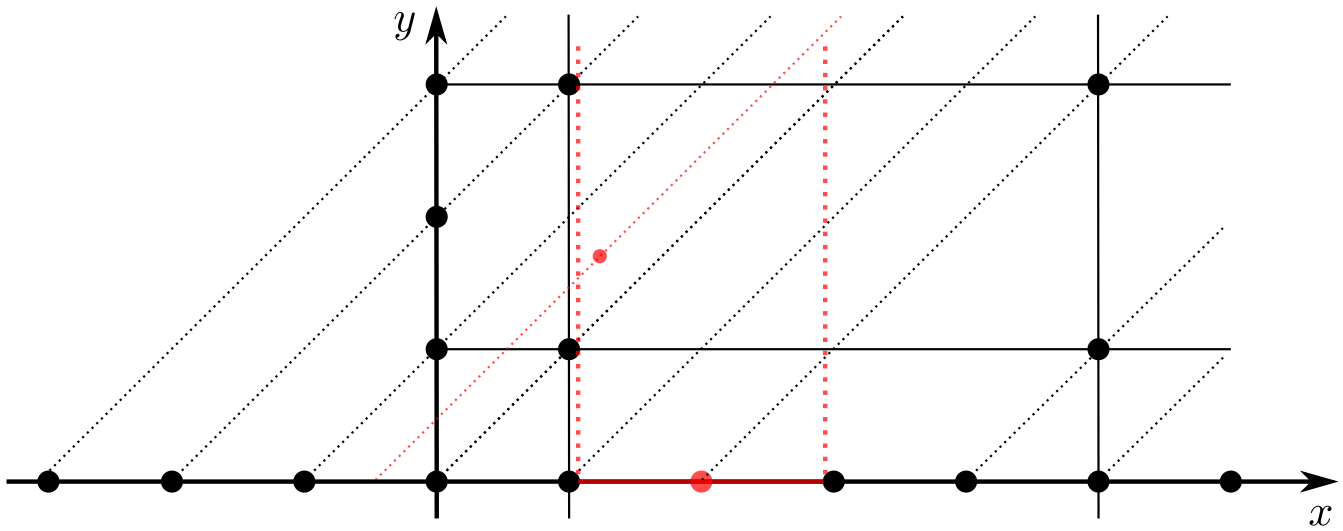
- Resulting counter must be smaller than $z + y_u$
- Resulting counter must be above x_l

Two-Clock Timed Automata to Bounded One-Counter Automata



- Resulting counter must be smaller than $z + y_u$
- Resulting counter must be above x_l

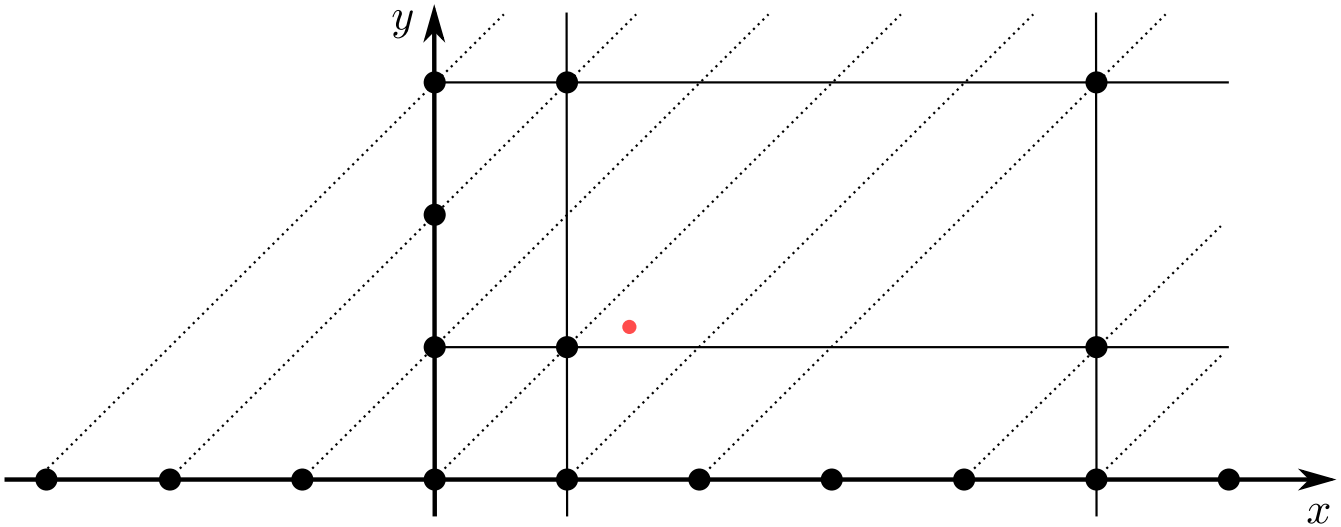
Two-Clock Timed Automata to Bounded One-Counter Automata



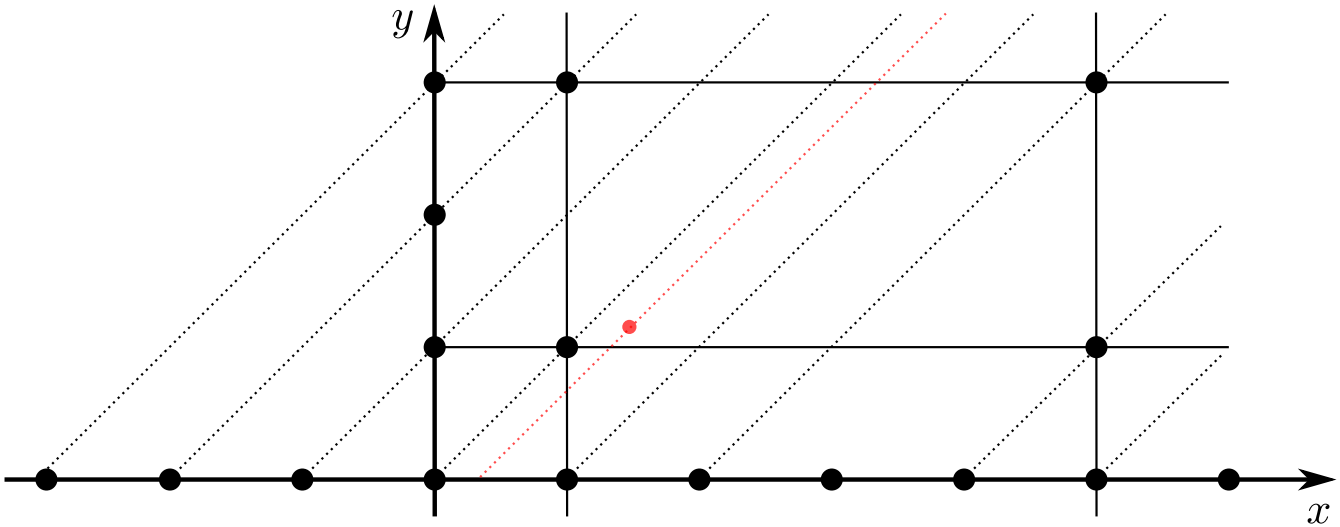
- Resulting counter must be smaller than $z + y_u$
- Resulting counter must be above x_l

$\rightsquigarrow$ add y_u to the counter, non-deterministically decrement counter and then check it is greater than x_l

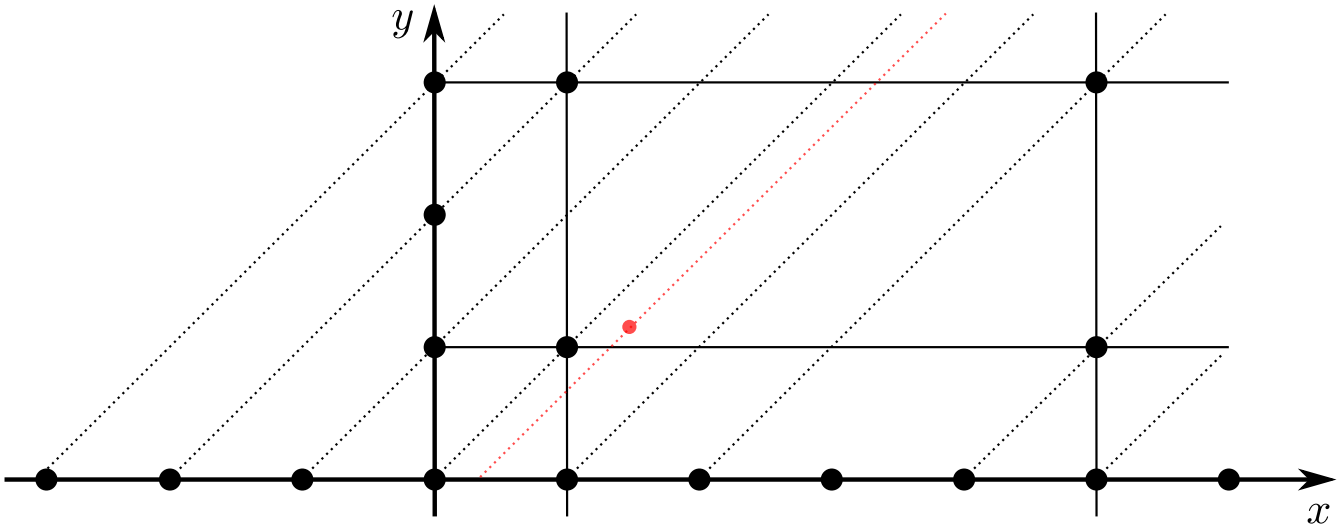
Two-Clock Timed Automata to Bounded One-Counter Automata



Two-Clock Timed Automata to Bounded One-Counter Automata

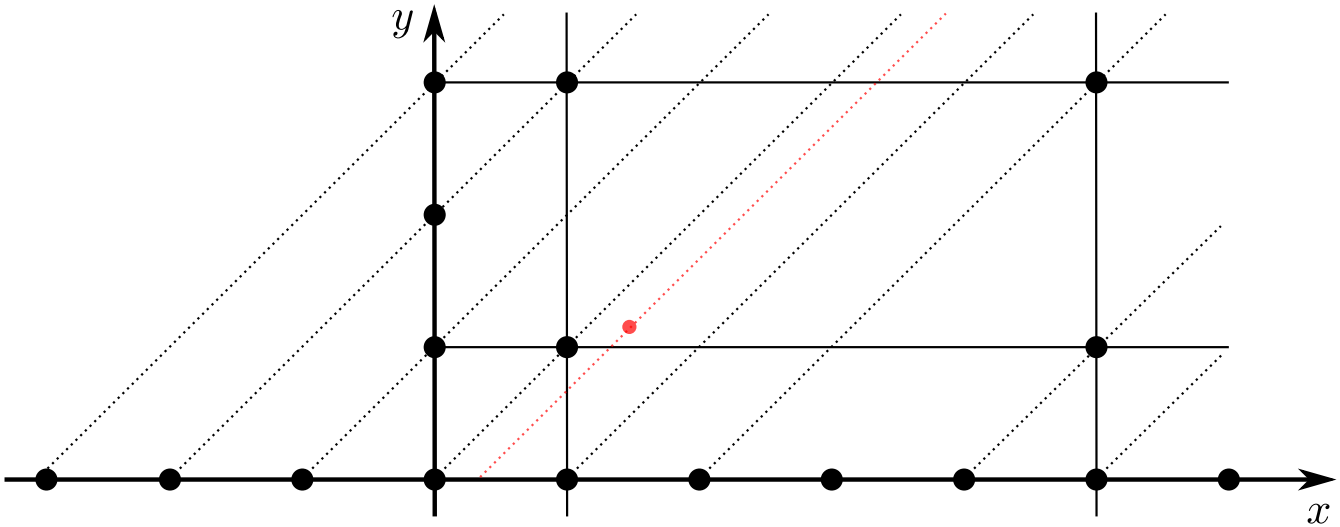


Two-Clock Timed Automata to Bounded One-Counter Automata



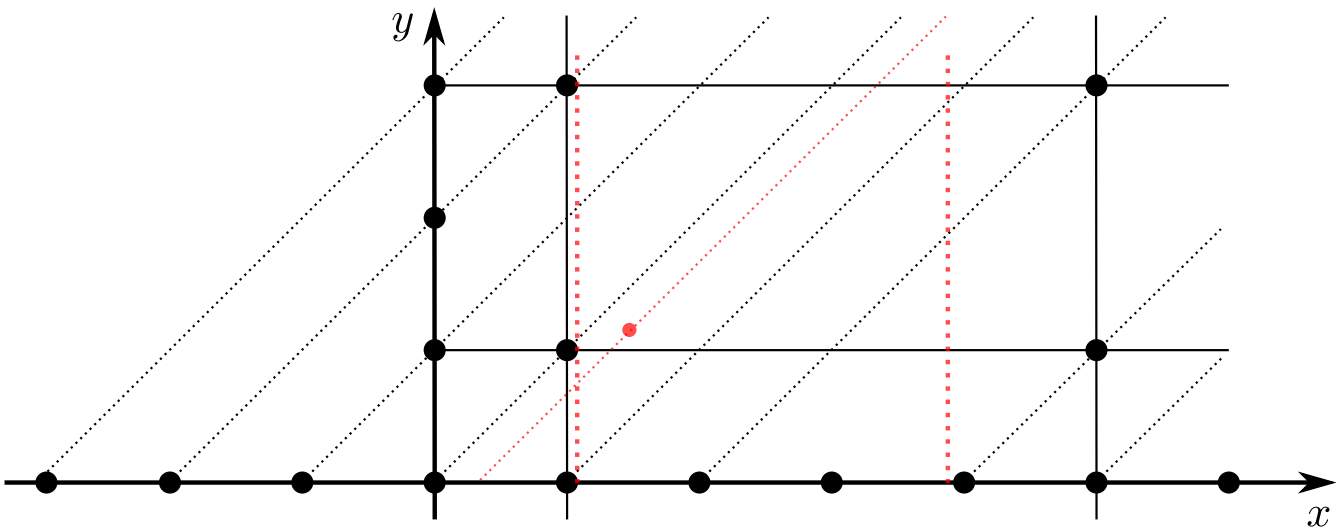
- counter must be below $n + y_u$

Two-Clock Timed Automata to Bounded One-Counter Automata



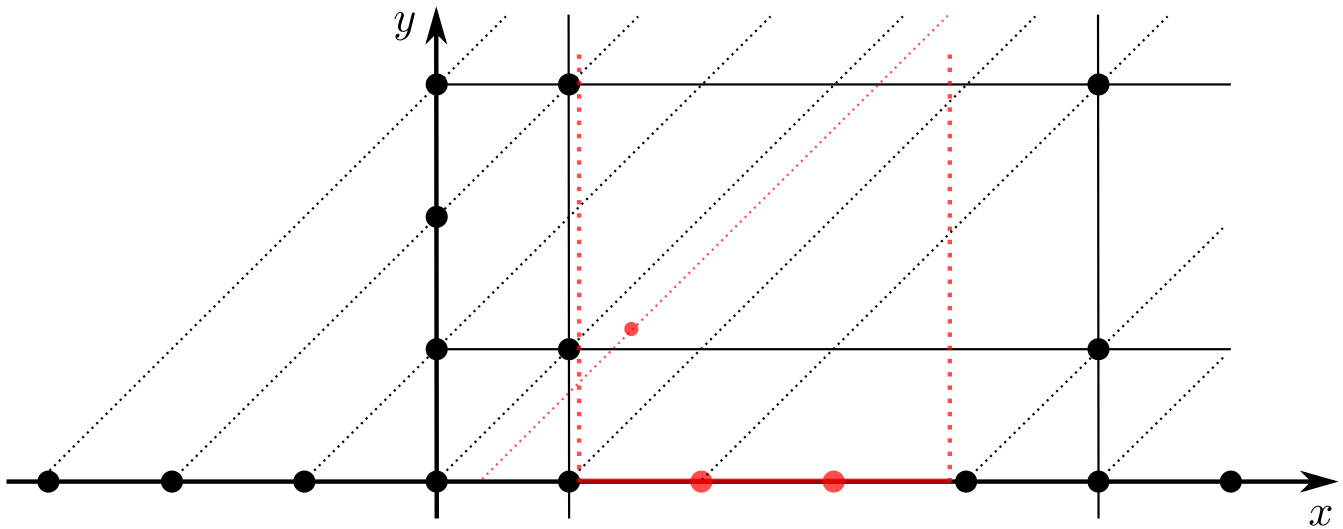
- counter must be below $n + y_u$
- counter must be above $n + y_l$

Two-Clock Timed Automata to Bounded One-Counter Automata



- counter must be below $n + y_u$
- counter must be above $n + y_l$

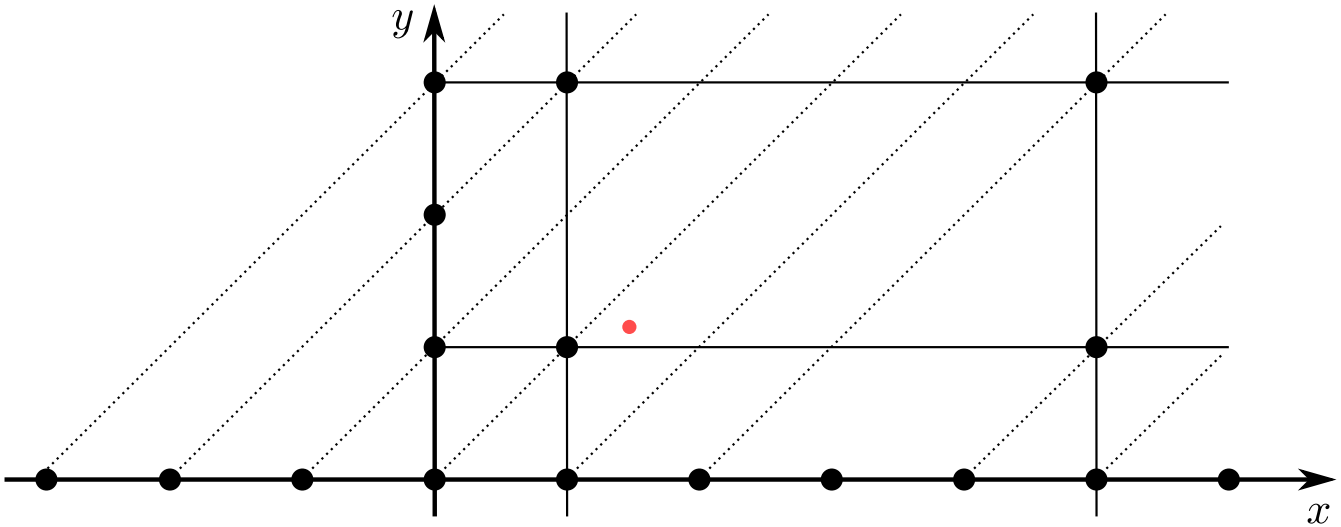
Two-Clock Timed Automata to Bounded One-Counter Automata



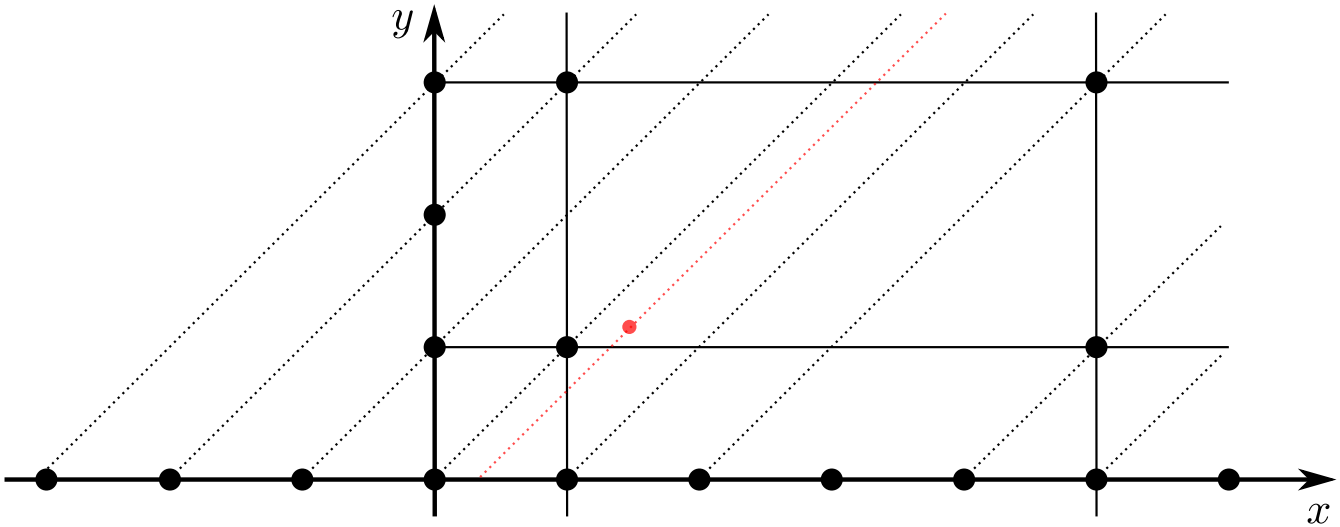
- counter must be below $n + y_u$
- counter must be above $n + y_l$

$\rightsquigarrow$ add value from the interval $[y_l, y_u]$ to the counter (requires gadget)

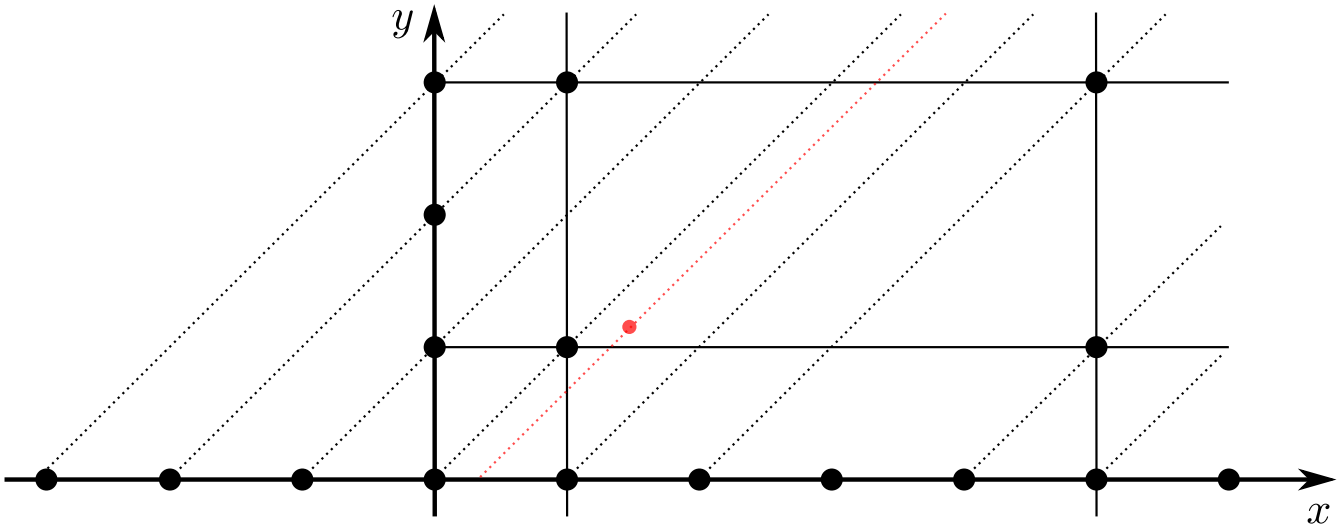
Two-Clock Timed Automata to Bounded One-Counter Automata



Two-Clock Timed Automata to Bounded One-Counter Automata

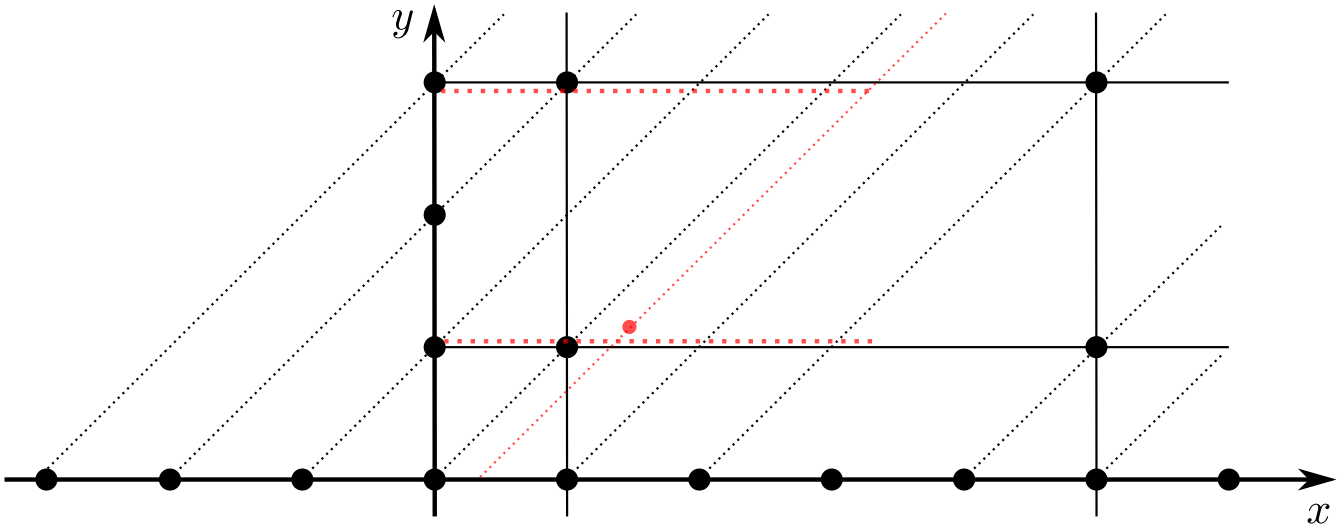


Two-Clock Timed Automata to Bounded One-Counter Automata



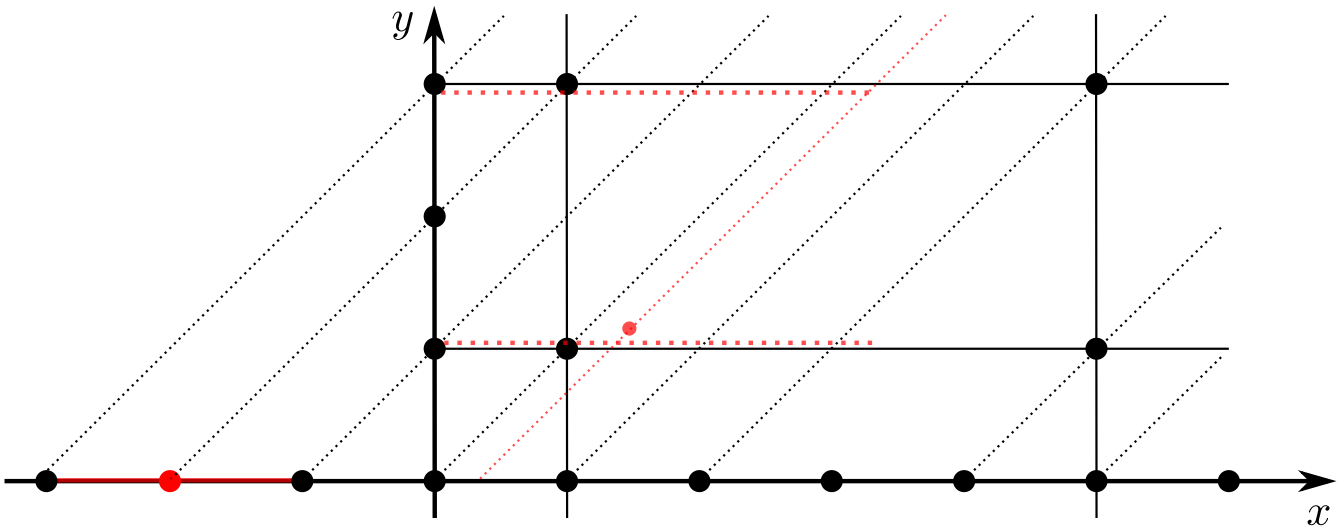
- counter must be below y_l
-
- counter must be above y_l

Two-Clock Timed Automata to Bounded One-Counter Automata



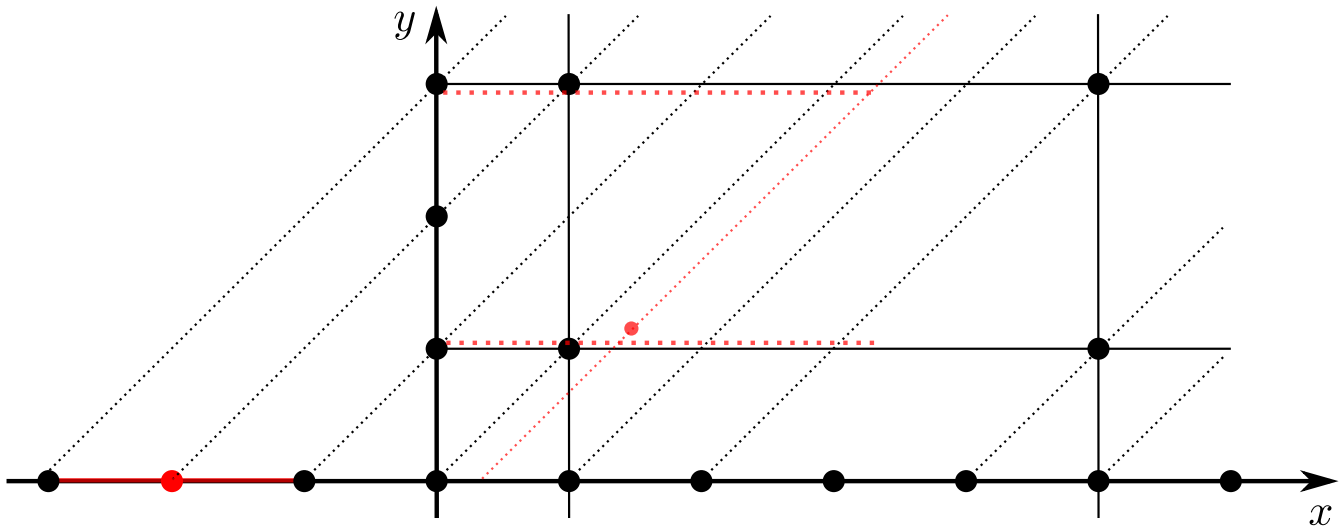
- counter must be below y_l
- counter must be above y_l

Two-Clock Timed Automata to Bounded One-Counter Automata



- counter must be below y_l
- counter must be above y_l

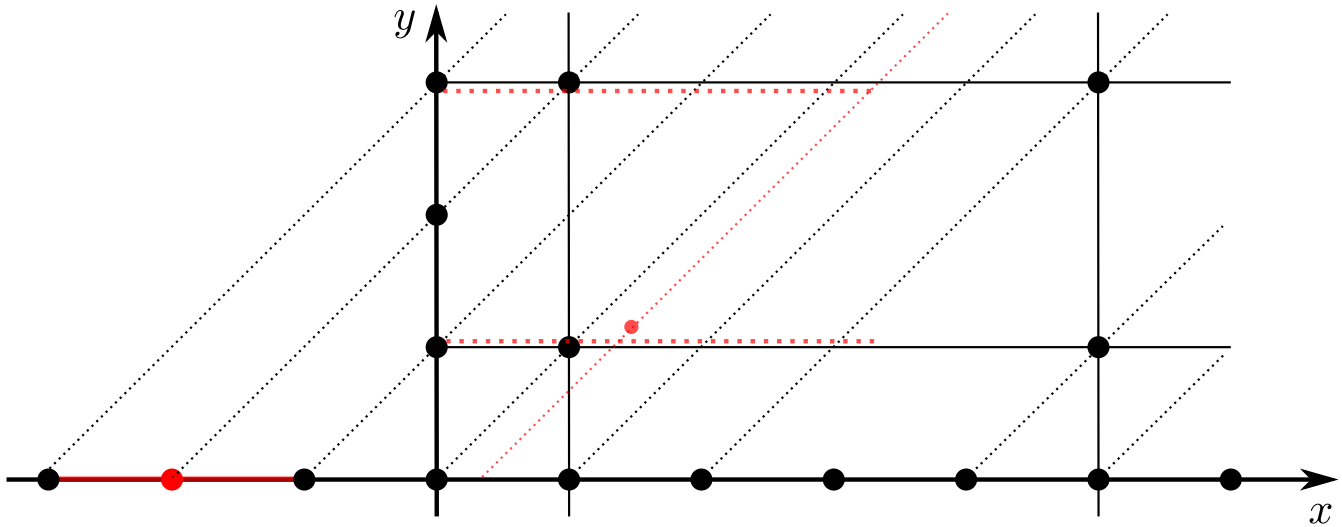
Two-Clock Timed Automata to Bounded One-Counter Automata



- counter must be below y_l
- counter must be above y_l

$\rightsquigarrow$ connect to a gadget which non-deterministically decrements the counter and then verifies that it is in $(-y_u, -y_l)$

Two-Clock Timed Automata to Bounded One-Counter Automata



Remaining polynomially many cases follow analogously

Bounded One-Counter Automata to Two-Clock Timed Automata to

- Other direction follows straightforwardly by encoding counter as the difference of two clocks, similar to the case with two counters

Bounded One-Counter Automata to Two-Clock Timed Automata to

- Other direction follows straightforwardly by encoding counter as the difference of two clocks, similar to the case with two counters

Theorem

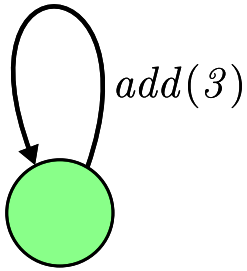
Reachability in two-clock timed automata is logarithmic-space inter-reducible with reachability in bounded one-counter automata.

Answering the Pólya Question

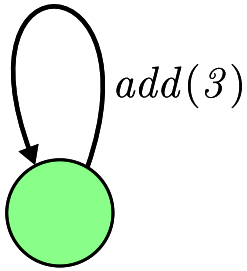
George Pólya (1887-1985)

“If there is a problem you can’t solve, then there is an easier problem you can solve: find it.”

One control location, one self-loop

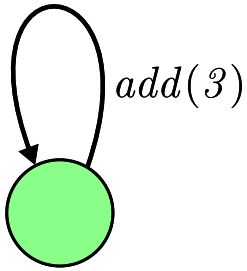


One control location, one self-loop



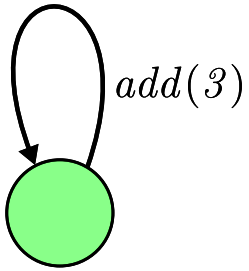
Reachability is NP-hard if the number of edges is unbounded and numbers are encoded in binary

One control location, one self-loop



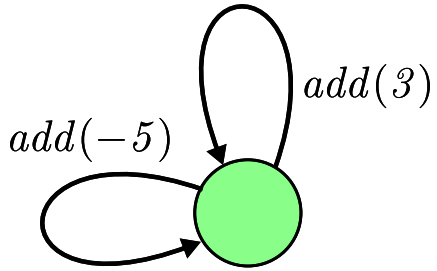
Given a bound and a target, reachability is clearly decidable in polynomial time

One control location, one self-loop



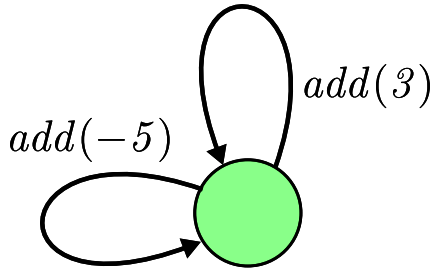
Given a bound and a target, reachability is clearly decidable in polynomial time $\surd$

One control location, two self-loops



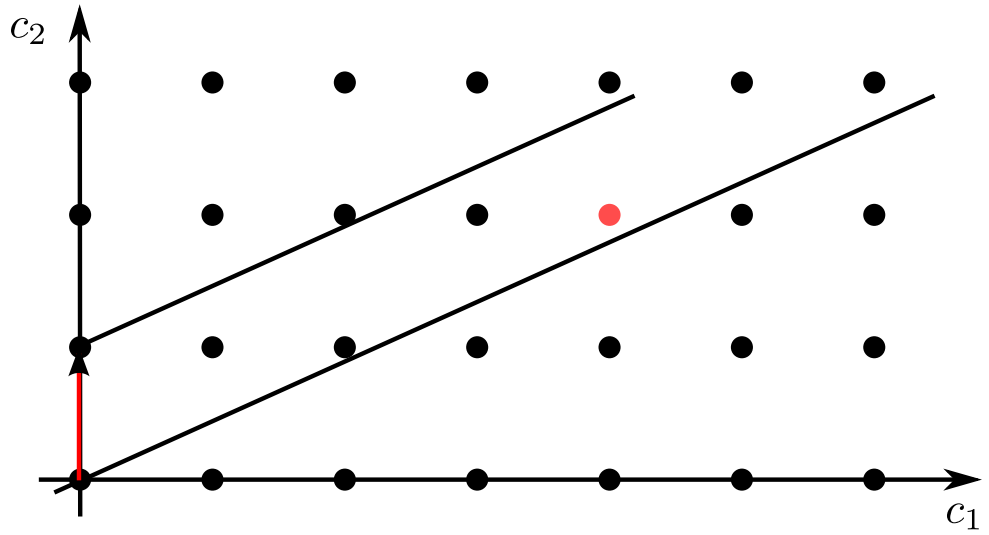
Given a bound a target *and the Parikh image of a reaching run*, reachability is decidable in polynomial time

One control location, two self-loops



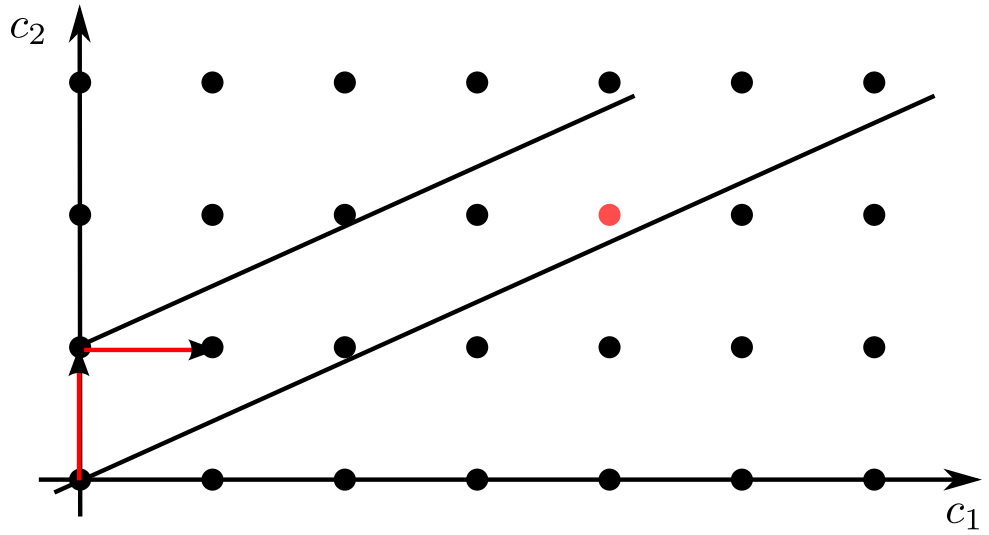
Given a bound a target *and the Parikh image of a reaching run*, reachability is decidable in polynomial time $\surd$

Reachability via Lattice Paths

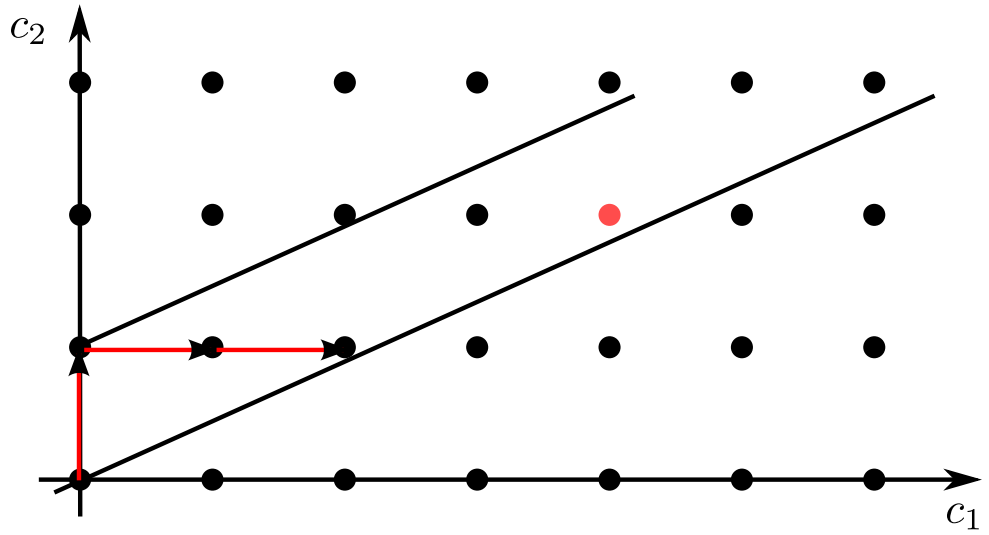


Idea: transform the reachability question into a question about the existence of lattice path in a convex polygon

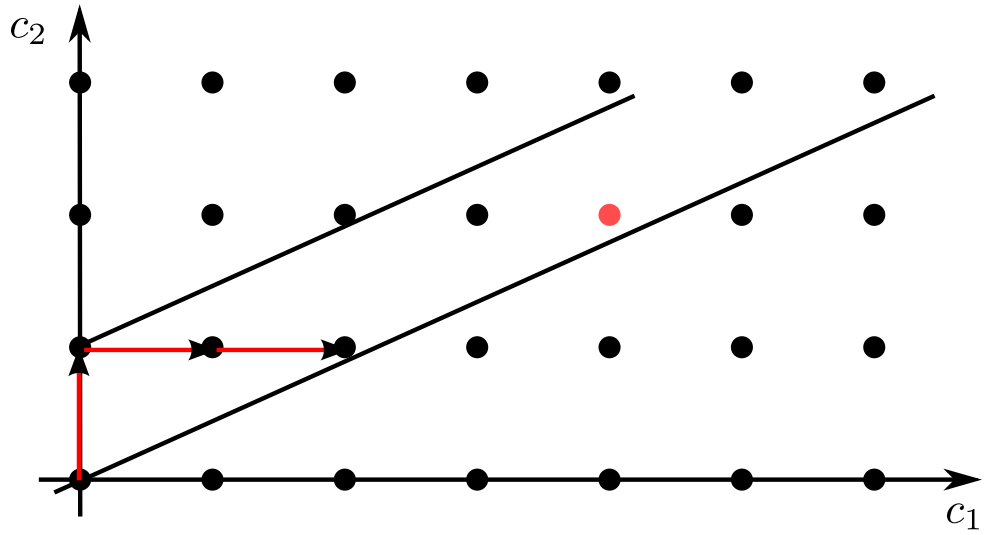
Reachability via Lattice Paths



Reachability via Lattice Paths

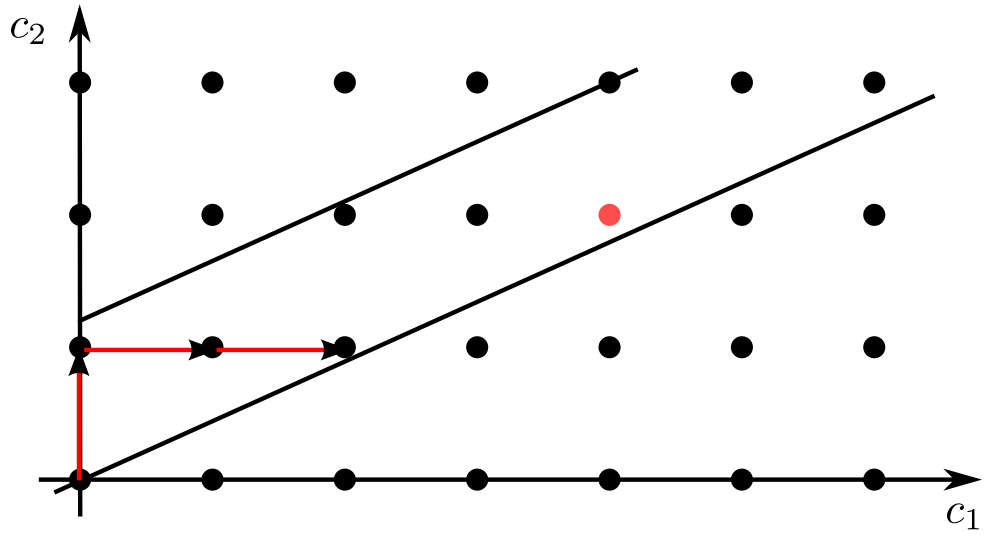


Reachability via Lattice Paths

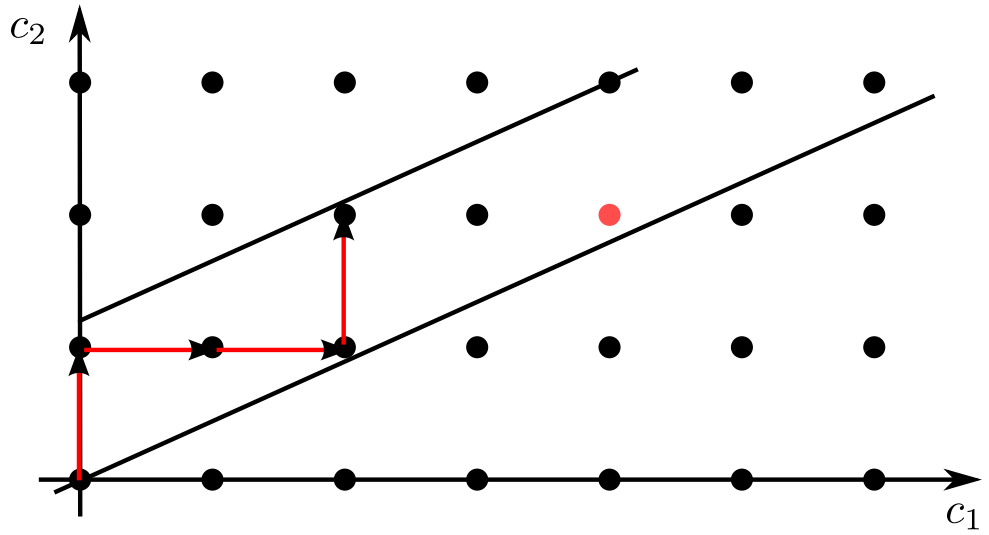


We get stuck since bound is too tight

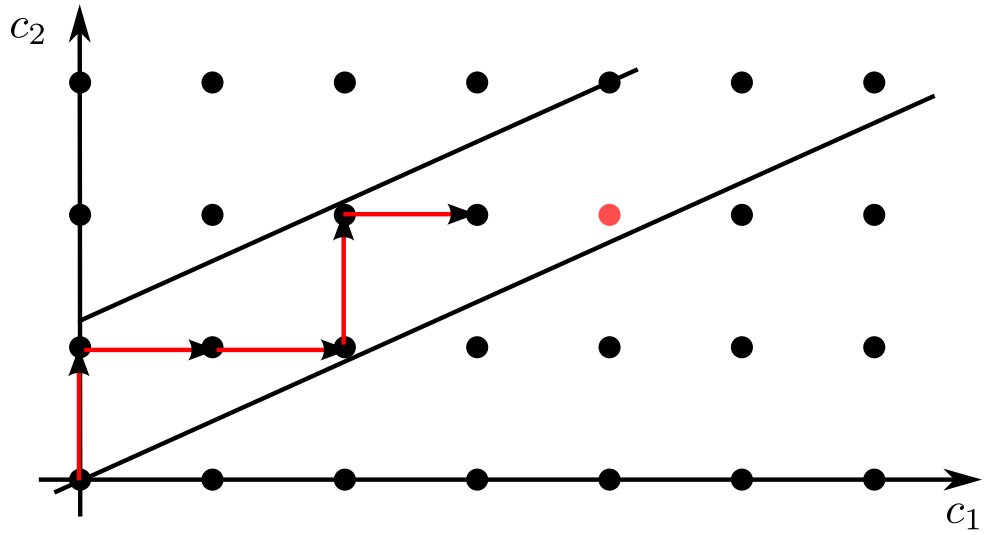
Reachability via Lattice Paths



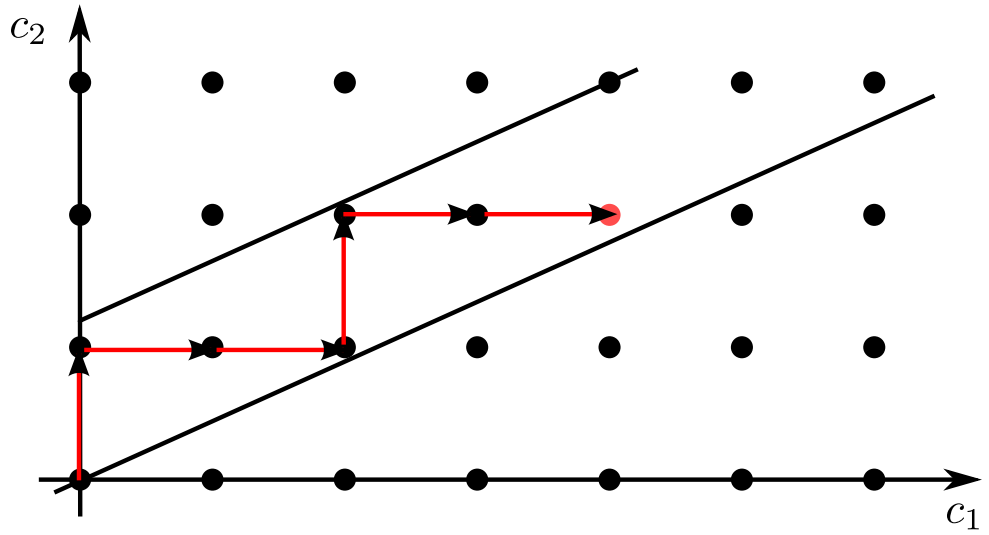
Reachability via Lattice Paths



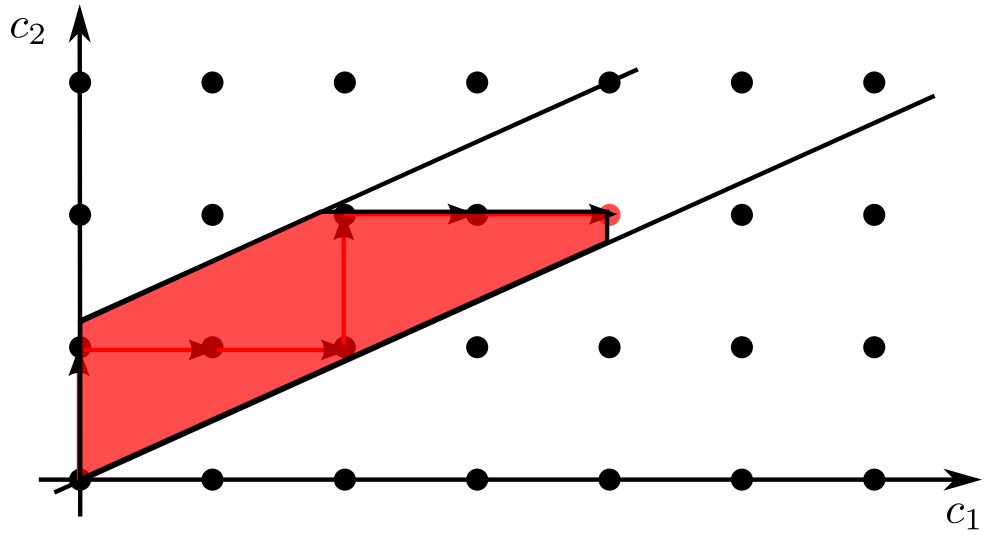
Reachability via Lattice Paths



Reachability via Lattice Paths



Reachability via Lattice Paths

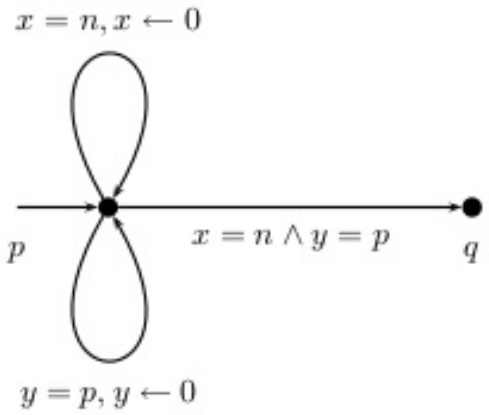


There exists a lattice path reaching a particular point (x, y) if, and only if, the number of lattice points in the polygon is at least

$$x + y + 1$$

Implications for Two-Clock Timed Automata

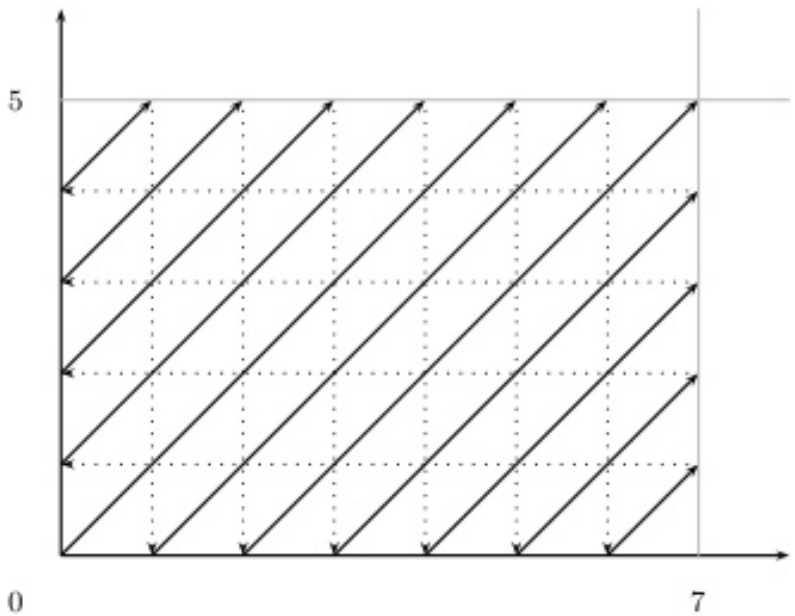
FIG. 6.1 – Automate de Bezout



Bézout automaton introduced in [Naves, 2006]

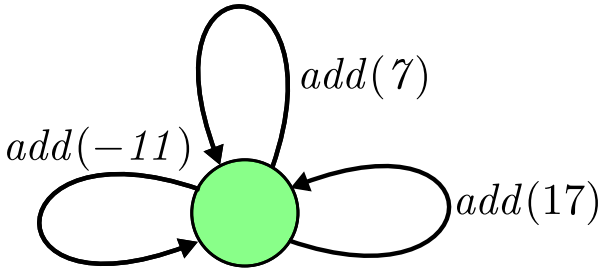
Implications for Two-Clock Timed Automata

Exemple :



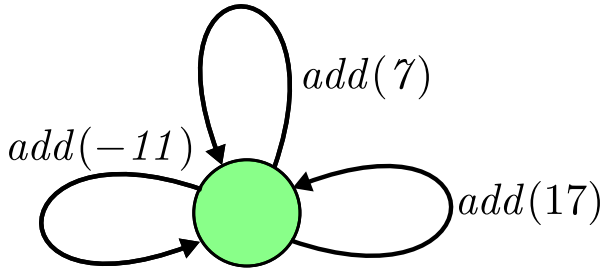
Pour $n = 7$ et $p = 5$, une représentation graphique de la plus courte exécution atteignant p. Les transitions de délai sont en trait continu, les transitions d'action en pointillés. L'exécution passe dans toutes les régions entières sauf $(7,0)$ et $(0,5)$. Les remises à zéro successives sont faites sur les horloges $y, x, y, x, y, y, x, y, x$, et enfin y.

One control location, three self-loops

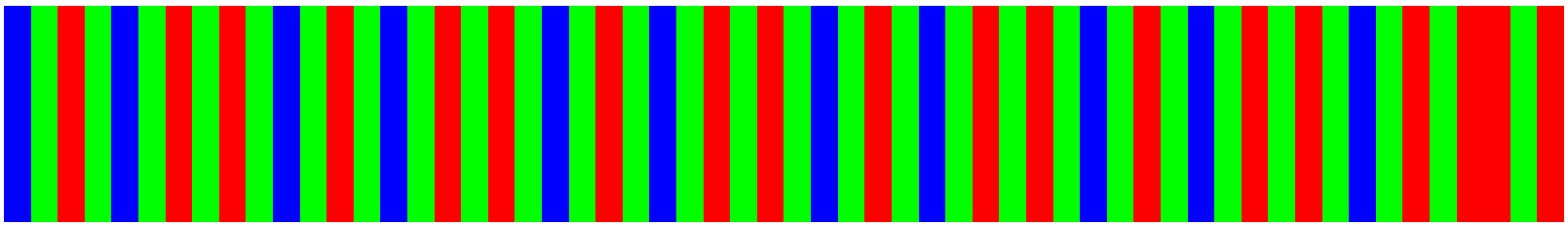


Let the bound be 20 and the target be 12

One control location, three self-loops



Let the bound be 20 and the target be 12



Example of a reaching run where red=7, green=-11 and blue=17

Conclusion

This talk showed

- a relationship between reachability problems in timed and bounded counter automata with respect to the resources available
- equivalence between two major problems that have been stated as open
- a simple class of bounded one-counter automata for which reachability is open

Conclusion

This talk showed

- a relationship between reachability problems in timed and bounded counter automata with respect to the resources available
- equivalence between two major problems that have been stated as open
- a simple class of bounded one-counter automata for which reachability is open

Conclusion

This talk showed

- a relationship between reachability problems in timed and bounded counter automata with respect to the resources available
- equivalence between two major problems that have been stated as open
- a simple class of bounded one-counter automata for which reachability is open

Conclusion

This talk showed

- a relationship between reachability problems in timed and bounded counter automata with respect to the resources available
- equivalence between two major problems that have been stated as open
- a simple class of bounded one-counter automata for which reachability is open