The monitoring problem for timed automata

Filip Mazowiecki

Max Planck Institute for Software Systems,
Saarbrücken Germany

based on joint work with A. Grez, M. Pilipczuk, G. Puppis and C. Riveros

The monitoring problem for timed automata

Filip Mazowiecki

Max Planck Institute for Software Systems,
Saarbrücken Germany

based on joint work with A. Grez, M. Pilipczuk, G. Puppis and C. Riveros

Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

- A property P

  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

- A property P

  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

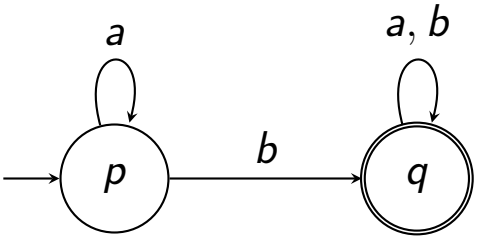
- A property P

  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

  Example "Is there a $d_i = b$?"



Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

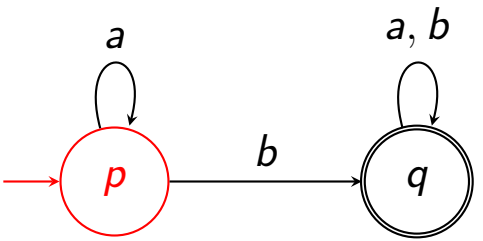
- A property P

  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

  Example "Is there a $d_i = b$?"



Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

- A property P

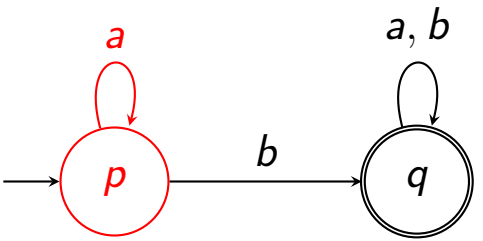
  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

  Example "Is there a $d_i = b$?"

  a



Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

- A property P

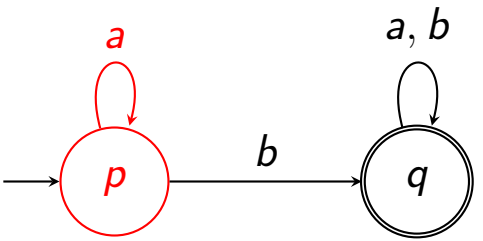
  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

  Example "Is there a $d_i = b$?"

  a a



Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

- A property P

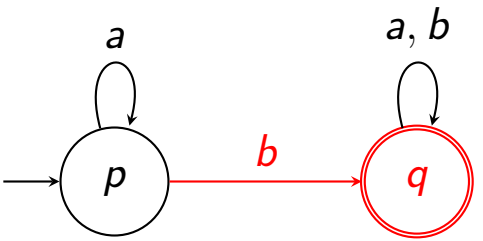
  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

  Example "Is there a $d_i = b$?"

  $a\ a\ b$



Monitoring problems

- Stream $d_0 d_1 d_2 d_3 \ldots \quad d_i \in \Sigma$

  finite alphabet: $\Sigma = \{a, b\}$

  set of numbers: $\Sigma = \mathbb{N}$

- A property P

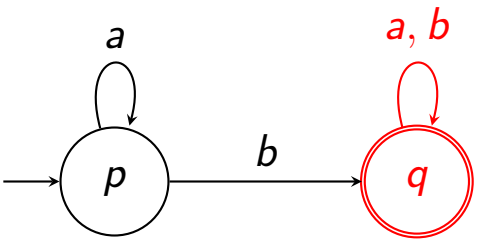
  boolean: "Is there a $d_i = b$?"

  quantitative: "What is the maximal d_i?"

  Goal: design a structure that dynamically verifies P

  Example "Is there a $d_i = b$?"

  $a\ a\ b\ a \ldots$



Challenges

- Output for P
  - Always correct (in this talk)

Challenges

- Output for P
  - Always correct (in this talk)

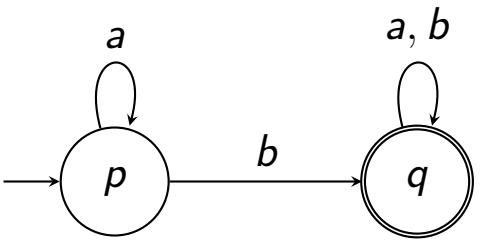
- Efficiency
  - Preprocessing time
  - Answer time
  - Update time
  - Space

Challenges

- Output for P
  - Always correct (in this talk)

- Efficiency
  - Preprocessing time
  - Answer time
  - Update time
  - Space

- Example $P =$ "Is there a $d_i = b$?"

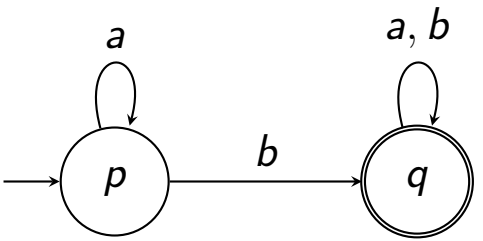


Challenges

- Output for P
  - Always correct (in this talk)

- Efficiency
  - Preprocessing time
  - Answer time
  - Update time
  - Space

- Example $P =$ "Is there a $d_i = b$?"
  - Preprocessing: $|P|$

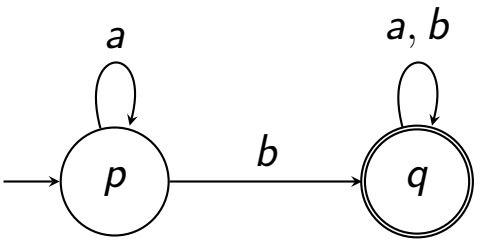


Challenges

- Output for P
  - Always correct (in this talk)

- Efficiency
  - Preprocessing time
  - Answer time
  - Update time
  - Space

- Example $P =$ "Is there a $d_i = b$?"
  - Preprocessing: $|P|$
  - Answer: constant

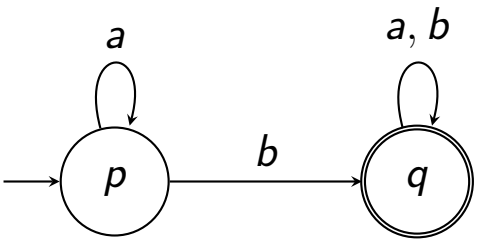


Challenges

- Output for P
  - Always correct (in this talk)

- Efficiency
  - Preprocessing time
  - Answer time
  - Update time
  - Space

- Example $P =$ "Is there a $d_i = b$?"
  - Preprocessing: $|P|$
  - Answer: constant
  - Update: constant

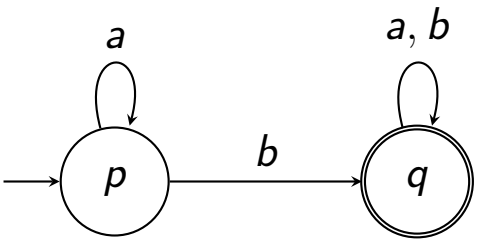


Challenges

- Output for P
  - Always correct (in this talk)

- Efficiency
  - Preprocessing time
  - Answer time
  - Update time
  - Space

- Example $P =$ "Is there a $d_i = b$?"
  - Preprocessing: $|P|$
  - Answer: constant
  - Update: constant
  - Space: $|P|$

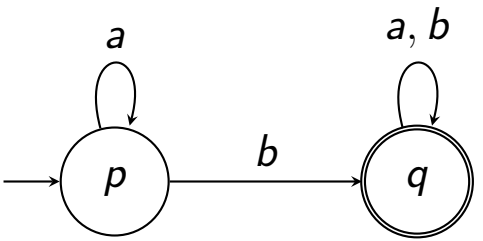


Challenges

- Output for P
  - Always correct (in this talk)

- Efficiency — Goals:
  - Preprocessing time — constant
  - Answer time — constant
  - Update time — constant
  - Space — irrelevant (for now)

- Example $P =$ "Is there a $d_i = b$?"
  - Preprocessing: $|P|$
  - Answer: constant
  - Update: constant
  - Space: $|P|$



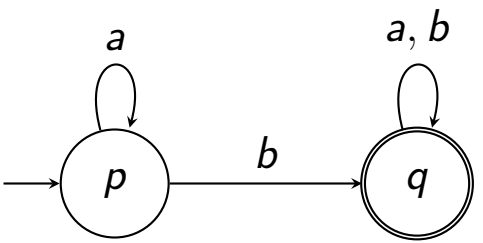
Challenges

- Output for P
  - Always correct (in this talk)

$|P|$ usually means constant

- Efficiency — Goals:
  - Preprocessing time — constant ← ignored in this talk
  - Answer time — constant
  - Update time — constant
  - Space — irrelevant (for now)

- Example $P =$ "Is there a $d_i = b$?"
  - Preprocessing: $|P|$
  - Answer: constant
  - Update: constant
  - Space: $|P|$



Windows

Does the property P hold within a window frame?

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"?

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? YES

- Example $N = 10$

$$ab\underbrace{baabbababaa}_{N}$$

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? YES

- Example $N = 10$

$$ab\underbrace{baabbababaa}_{N}a$$

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? NO

- Example $N = 10$

$$abb\underbrace{aabbababaaa}_{N}$$

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? NO

- Example $N = 10$

  $abb\underbrace{aabbababaaa}_{N}$

- Answer and update in constant time?

  $|P|$ is small (constant)

  but N is big

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? NO

- Example $N = 10$

  $abb\underbrace{aabbababaaa}_{N}$

- Answer and update in constant time?

  $|P|$ is small (constant)

  but N is big

- Motivation: new data is more valuable

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? NO

- Example $N = 10$

  $abba\underbrace{abbababaaab}_{N}$

- Answer and update in constant time?

  $|P|$ is small (constant)

  but N is big

- Motivation: new data is more valuable

Windows

Does the property P hold within a window frame?

- Given window size N

  $P =$ "does the window start with b"? YES

- Example $N = 10$

  $abbaa\underbrace{bbababaaabb}_{N}$

- Answer and update in constant time?

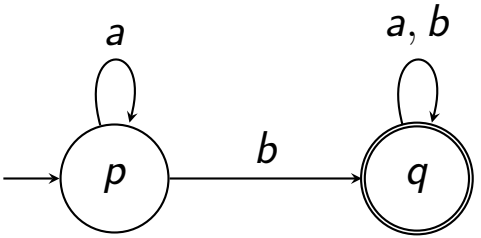
  $|P|$ is small (constant)

  but N is big

- Motivation: new data is more valuable

Timed automata

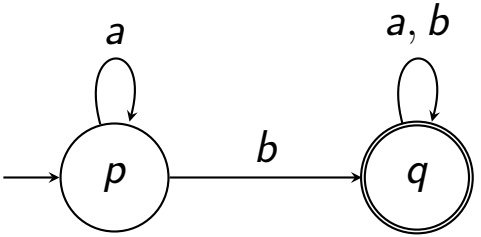
An extension of finite automata



+

Timed automata

An extension of finite automata

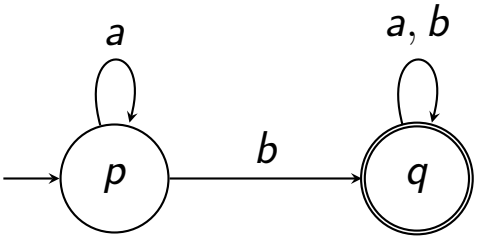


+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

Timed automata

An extension of finite automata



+

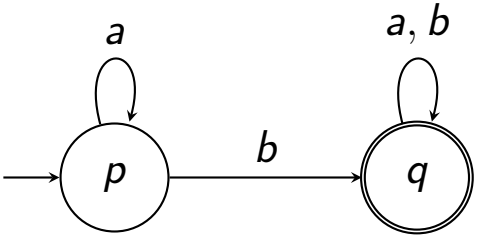
- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

Here we simplify

$$aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$$

Timed automata

An extension of finite automata

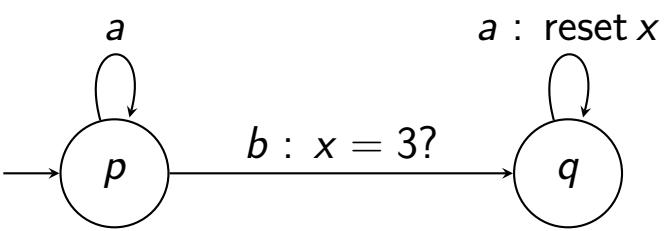


+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

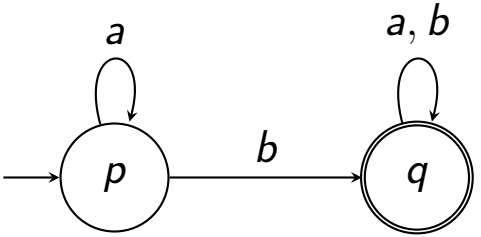
  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

- There are clocks x, y



Timed automata

An extension of finite automata



+

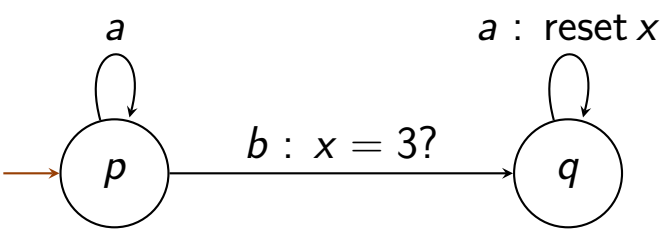
- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

- There are clocks x, y

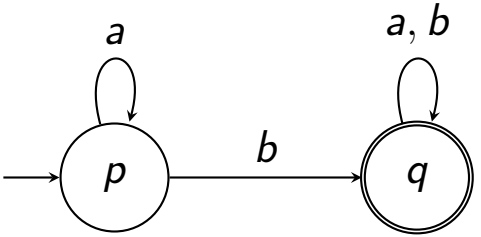
$x \quad 0$

$aaba$



Timed automata

An extension of finite automata

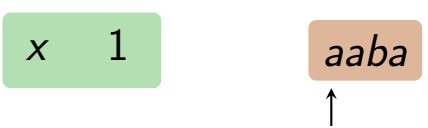


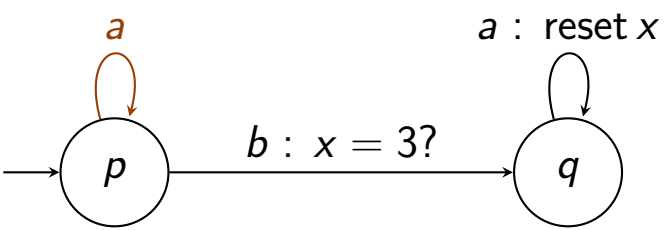
+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

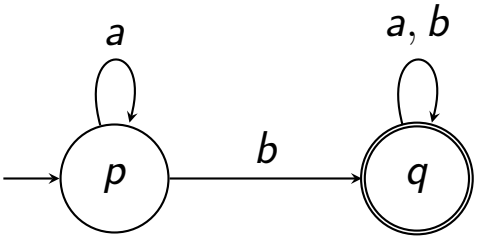
- There are clocks x, y





Timed automata

An extension of finite automata

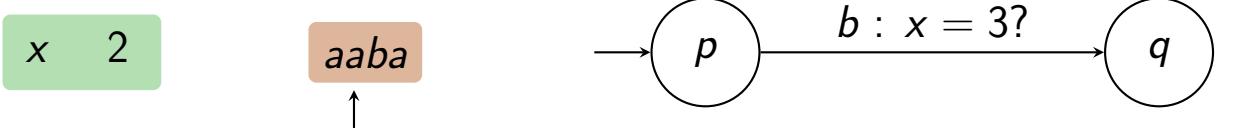


+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

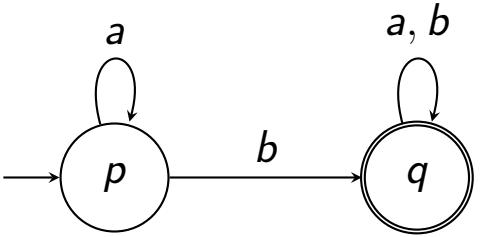
  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

- There are clocks x, y



Timed automata

An extension of finite automata

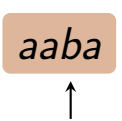


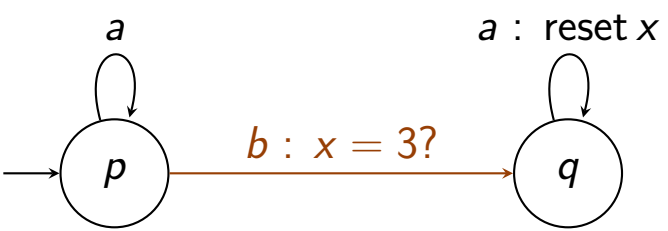
+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

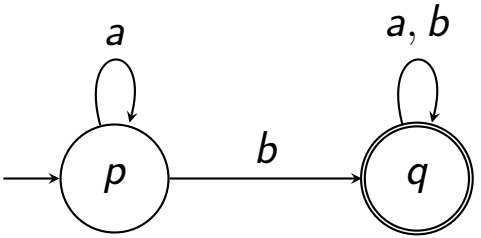
- There are clocks x, y





Timed automata

An extension of finite automata

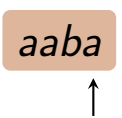


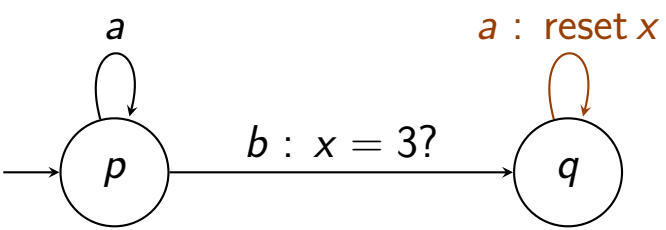
+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

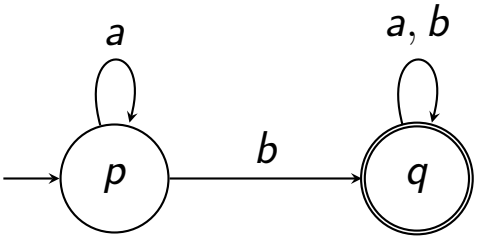
- There are clocks x, y





Timed automata

An extension of finite automata



+

- Letters arrive with timestamps $\binom{a}{01.03.2020\ 12:34}\binom{a}{01.03.2020\ 16:42}\binom{b}{02.03.2020\ 10:11}$

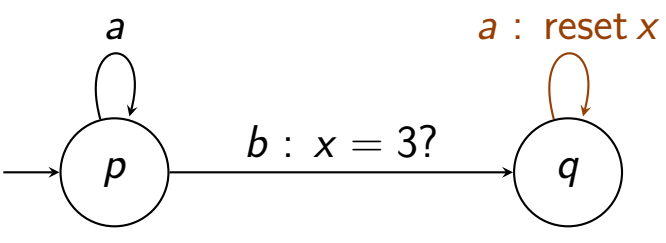
  Here we simplify $aab\ldots = \binom{a}{1}\binom{a}{2}\binom{b}{3}\ldots$

- There are clocks x, y

$\text{reset}\, x,\ x \leq N,\ x = N,\ x \geq N$

$x \quad 0$

$aaba$



“Real” example

Professor Joe

“Real” example

Professor Joe

Angie

Marco

Dan

“Real” example

Professor Joe

Angie

Marco

Dan

Thor

“Real” example

Professor Joe Angie Marco Dan Thor

- Joe receives many emails

“Real” example

Professor Joe Angie Marco Dan Thor

- Joe receives many emails

  Wants to always reply to Thor in 24h $= 1440$ minutes

"Real" example

Professor Joe Angie Marco Dan Thor

- Joe receives many emails

  Wants to always reply to Thor in 24h $= 1440$ minutes

- Joe sets a device checking his email box every minute

  b – email from Thor, a – no email from Thor

"Real" example

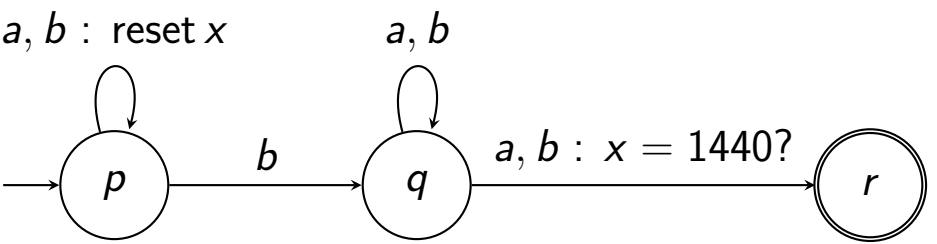
Professor Joe Angie Marco Dan Thor

- Joe receives many emails

  Wants to always reply to Thor in 24h = 1440 minutes

- Joe sets a device checking his email box every minute

  b – email from Thor, a – no email from Thor



"Real" example




Professor Joe Angie Marco Dan Thor

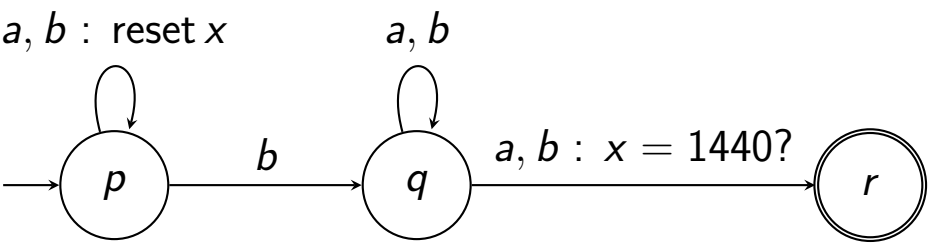
- Joe receives many emails

  Wants to always reply to Thor in 24h = 1440 minutes

- Joe sets a device checking his email box every minute

  b – email from Thor, a – no email from Thor

$a\ a\ b\ a\ a\ b\ a \ldots$



"Real" example




Professor Joe Angie Marco Dan Thor

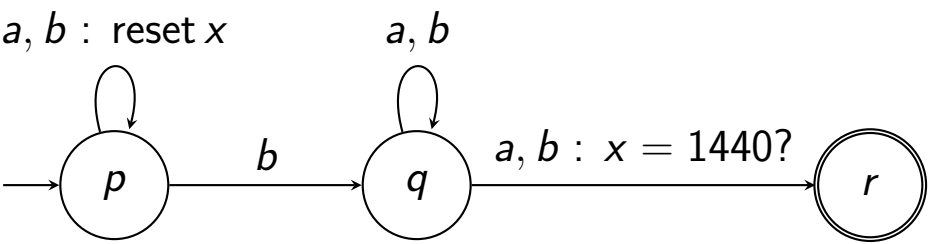
- Joe receives many emails

  Wants to always reply to Thor in 24h = 1440 minutes

- Joe sets a device checking his email box every minute

  b – email from Thor, a – no email from Thor

	a	a	b	a	a	b	a ...
x	0	0	1	2	3	4	5 ...



"Real" example

Professor Joe Angie Marco Dan Thor

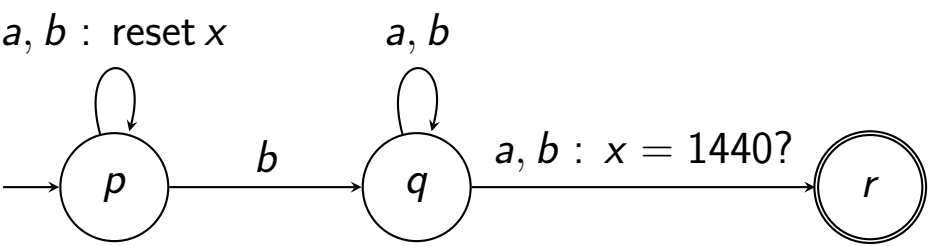
- Joe receives many emails

  Wants to always reply to Thor in 24h = 1440 minutes

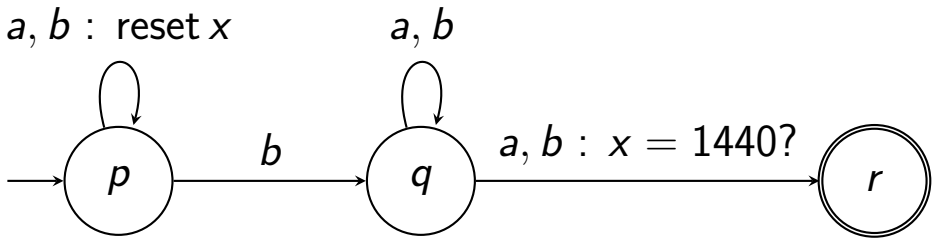
- Joe sets a device checking his email box every minute

  b – email from Thor, a – no email from Thor

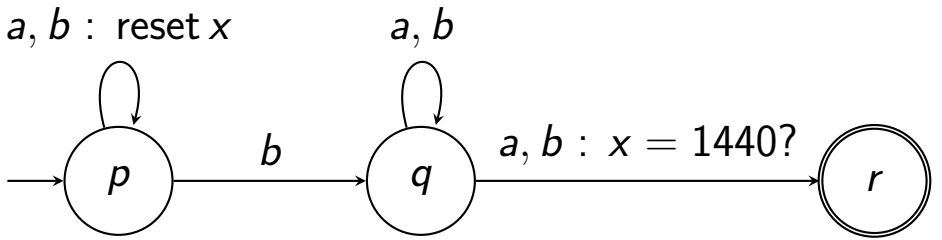
	a	a	b	a	a	b	a	$\ldots$
x	0	0	1	2	3	4	5	$\ldots$
x	0	0	0	0	0	1	2	$\ldots$



Structure for the example (not depending on 1440)



Structure for the example (not depending on 1440)

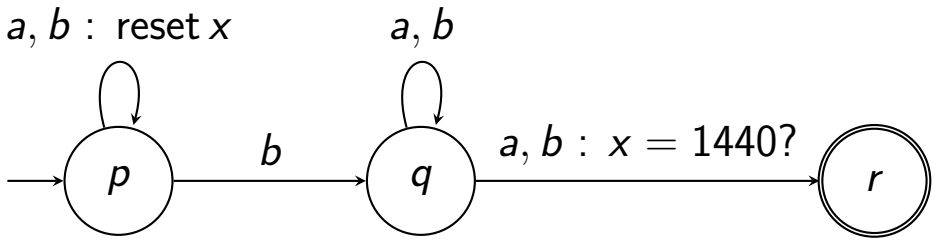


Output NO

Global clock 0 List

Input

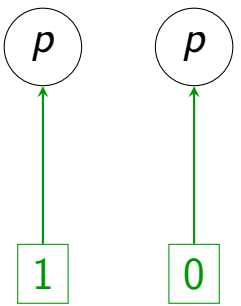
Structure for the example (not depending on 1440)



Output NO

Global clock 1

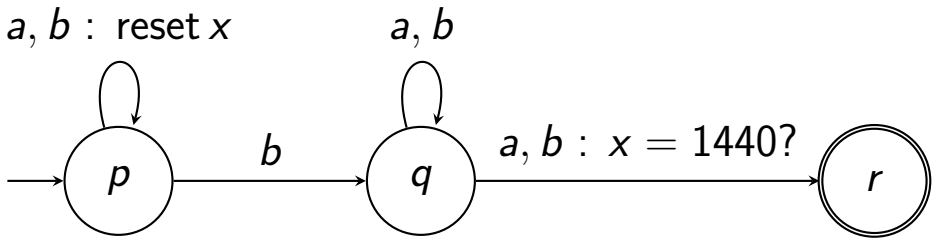
List



(reset)

Input b

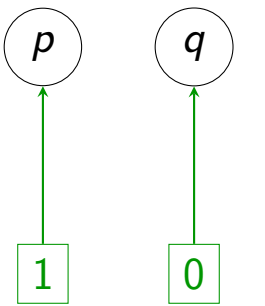
Structure for the example (not depending on 1440)



Output NO

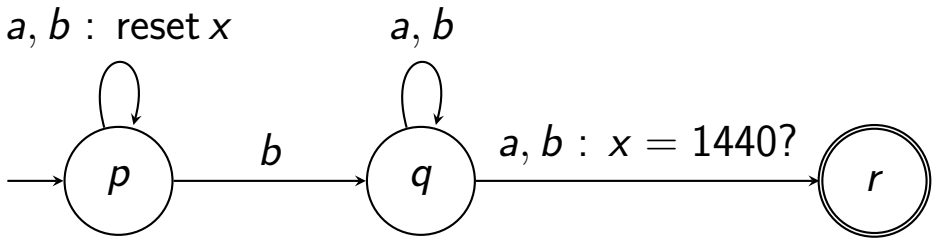
Global clock 1

List

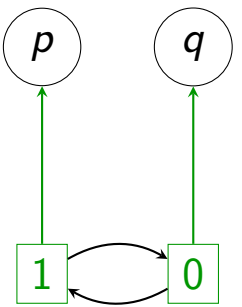


Input b

Structure for the example (not depending on 1440)



Output NO

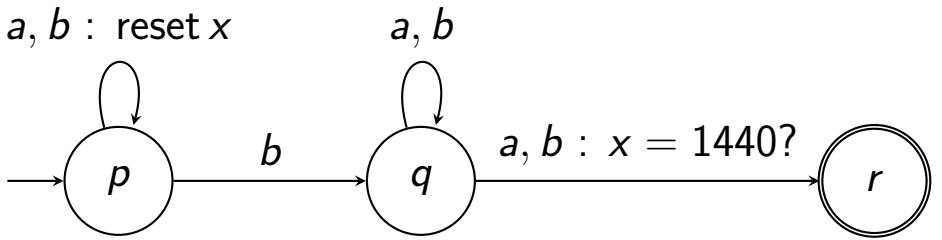


Global clock 1 List

(reset) (other)

Input b

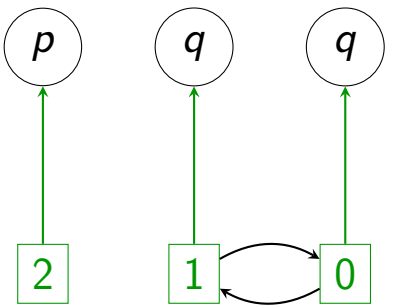
Structure for the example (not depending on 1440)



Output NO

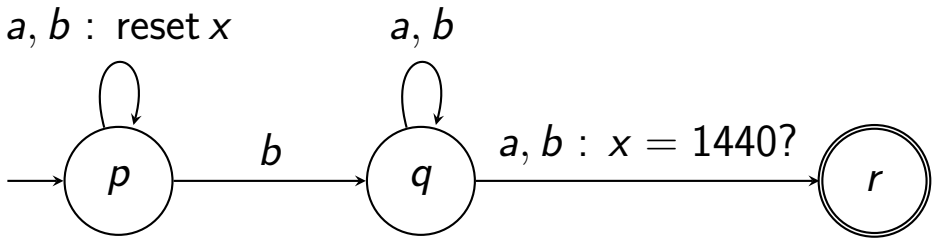
Global clock 2

List

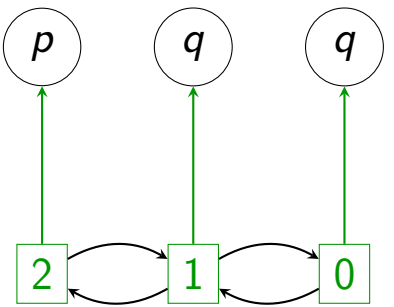


Input *bb*

Structure for the example (not depending on 1440)



Output NO

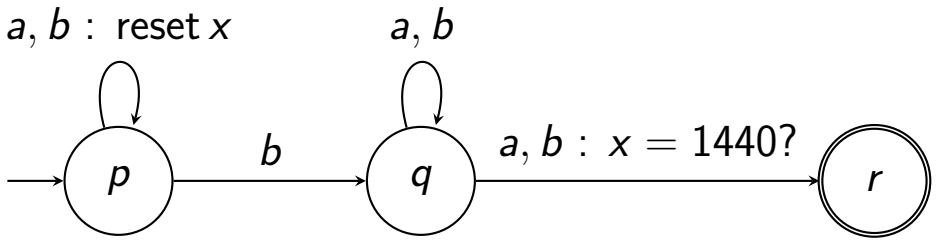


Global clock 2 List

(reset) (other)

Input *bb*

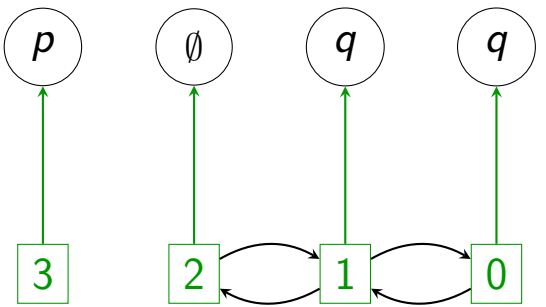
Structure for the example (not depending on 1440)



Output NO

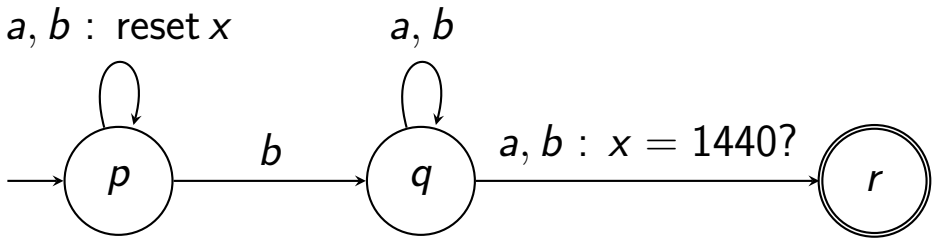
Global clock 3

List

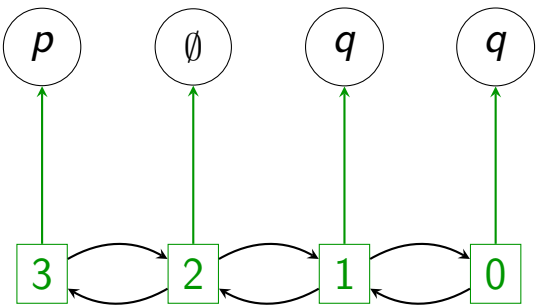


Input *bba*

Structure for the example (not depending on 1440)



Output NO

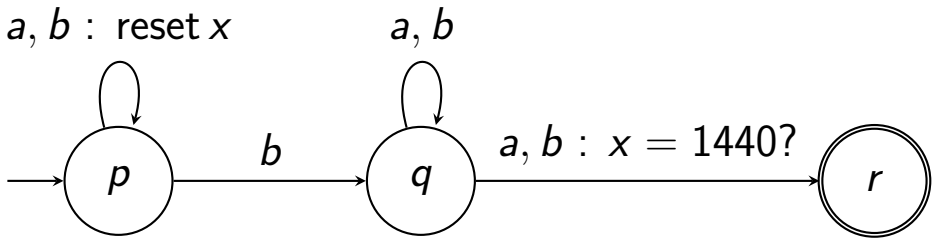


Global clock 3 List

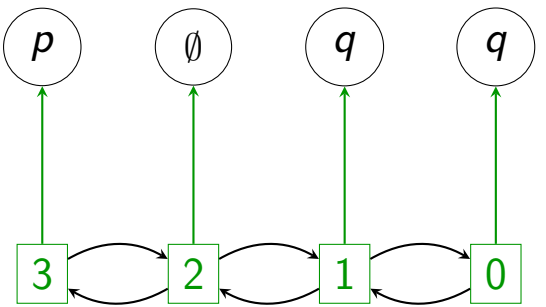
(reset) (other)

Input *bba*

Structure for the example (not depending on 1440)



Output YES

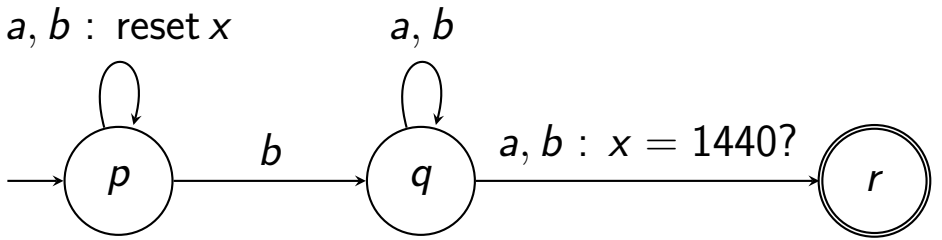


Global clock 1440 List ...

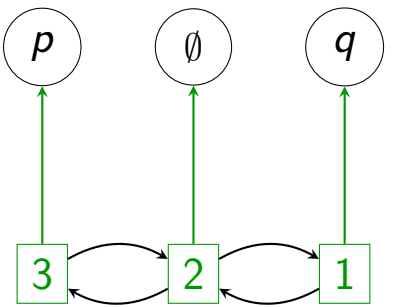
(reset) (other)

Input *bba* . . .

Structure for the example (not depending on 1440)



Output YES

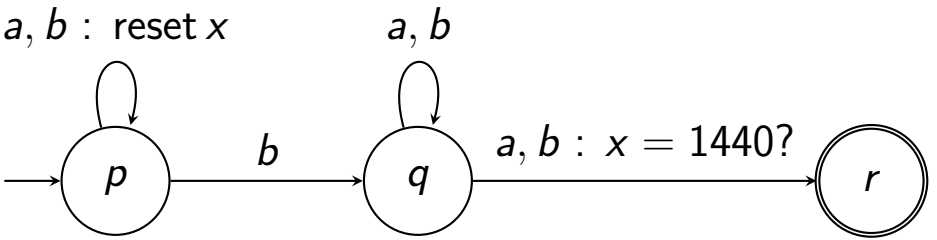


Global clock 1440 List ...

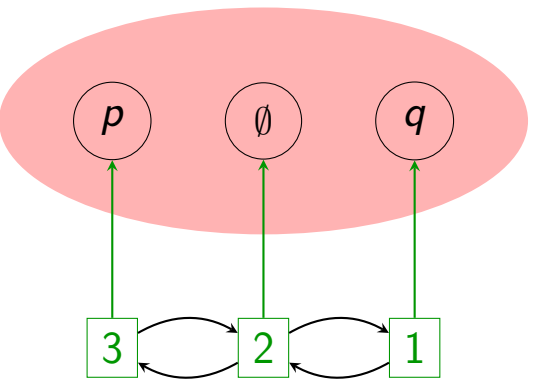
(reset) (other)

Input *bba* . . .

Structure for the example (not depending on 1440)



Output YES



Global clock 1440 List ...

(reset) (other)

Input *bba* . . .

- Issue: updating states

Current work

Theorem (Positive)

For 1 clock there is always a structure supporting amortised constants time.

When time is discrete $(0, 1, 2, \ldots)$ it supports (non-amortised) constant time.

Current work

Theorem (Positive)

For 1 clock there is always a structure supporting amortised constants time.

When time is discrete $(0, 1, 2, \ldots)$ it supports (non-amortised) constant time.

Theorem (Negative)

For 2+ clocks and extended timed automata no structure supports constant time (subject to some fine-grained complexity conjecture).

Current work

Theorem (Positive)

For 1 clock there is always a structure supporting amortised constants time.

When time is discrete $(0, 1, 2, \ldots)$ it supports (non-amortised) constant time.

Theorem (Negative)

For 2+ clocks and extended timed automata no structure supports constant time (subject to some fine-grained complexity conjecture).

- Can we deal with 2+ clocks?

Current work

Theorem (Positive)

For 1 clock there is always a structure supporting amortised constants time.

When time is discrete $(0, 1, 2, \ldots)$ it supports (non-amortised) constant time.

Theorem (Negative)

For 2+ clocks and extended timed automata no structure supports constant time (subject to some fine-grained complexity conjecture).

- Can we deal with 2+ clocks?

  Needed for complex event processing (CEP) [Grez, Ugarte, Riveros 2019]

$$\varphi \;:=\; a \;\mid\; \varphi;\varphi \;\mid\; \varphi \text{ WITHIN } t$$

Current work

Theorem (Positive)

For 1 clock there is always a structure supporting amortised constants time.

When time is discrete $(0, 1, 2, \ldots)$ it supports (non-amortised) constant time.

Theorem (Negative)

For 2+ clocks and extended timed automata no structure supports constant time (subject to some fine-grained complexity conjecture).

- Can we deal with 2+ clocks?

  Needed for complex event processing (CEP) [Grez, Ugarte, Riveros 2019]

$$\varphi \;:=\; a \;\mid\; \varphi;\varphi \;\mid\; \varphi \text{ WITHIN } t$$

- Can we restrict timed automata?

Other monitoring problems

- How much space is needed? (in terms of window size N)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  2

  $(2, 2)$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  25

  $(5, 5)$

  $(2, 2)$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  $(3,5)$
  $(5,5)$
  $(2,2)$ 253

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  $(3,5)$
  $(5,5)$
  $(2,2)$

  2 5 3

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

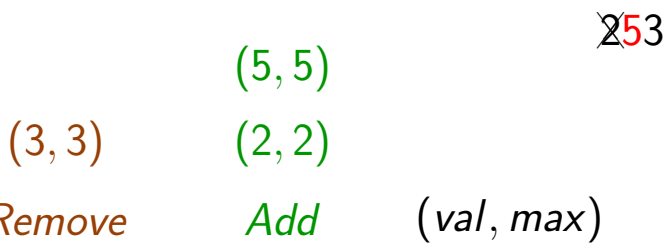
  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks



Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  ~~2~~53

  $(5, 5)$

  $(3, 3)$ $(2, 2)$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  $(2,5)$

  $(5,5)$

  $(3,3)$

  $\not{2}53$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  53

  $(5, 5)$

  $(3, 3)$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  534

  $(5, 5)$

  $(3, 3)$ $(4, 4)$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  34

  $(3,3)$ $(4,4)$

  Remove *Add* (val, max)

Other monitoring problems

- How much space is needed? (in terms of window size N)

  $a\Sigma^*$ $\Theta(N)$ [Moses Ganardi et. al]

  (Georg's future postdoc)

- Maximum of an integer Works for any monoid

  FIFO using insert() and evict()

  Amortised constant time using two stacks

  34

  $(3, 3)$ $(4, 4)$

  Remove *Add* (val, max)

  Can be even (non-amortised) constant [Tangwongsan, Hirzel, Schneider 2017]