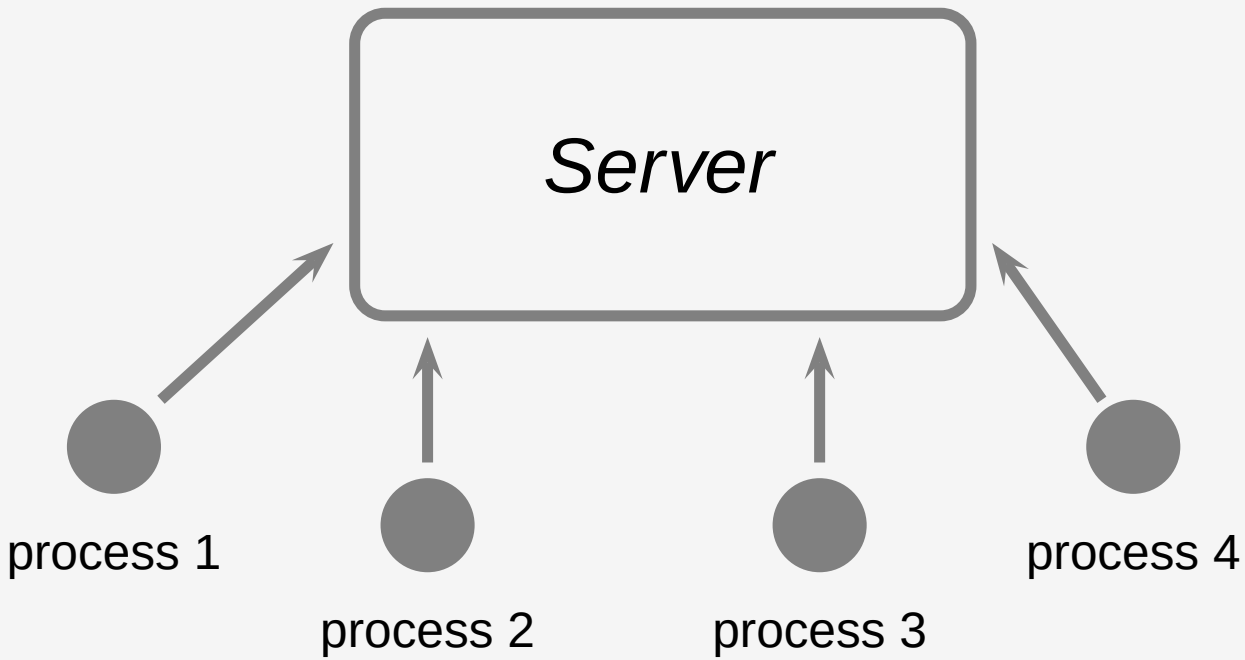
Feasible Automata for Two-Variable Logic with Successor on Data Words

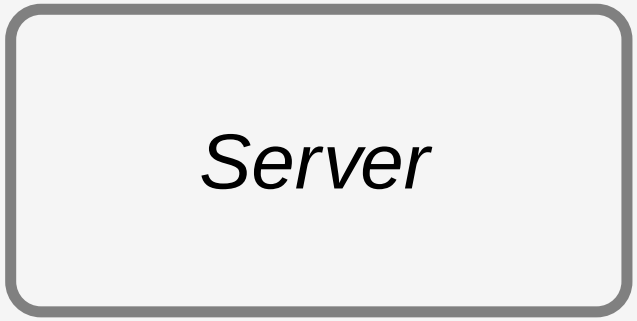
Ahmet Kara, Thomas Schwentick, and Tony Tan

LATA 2012, A Coruña

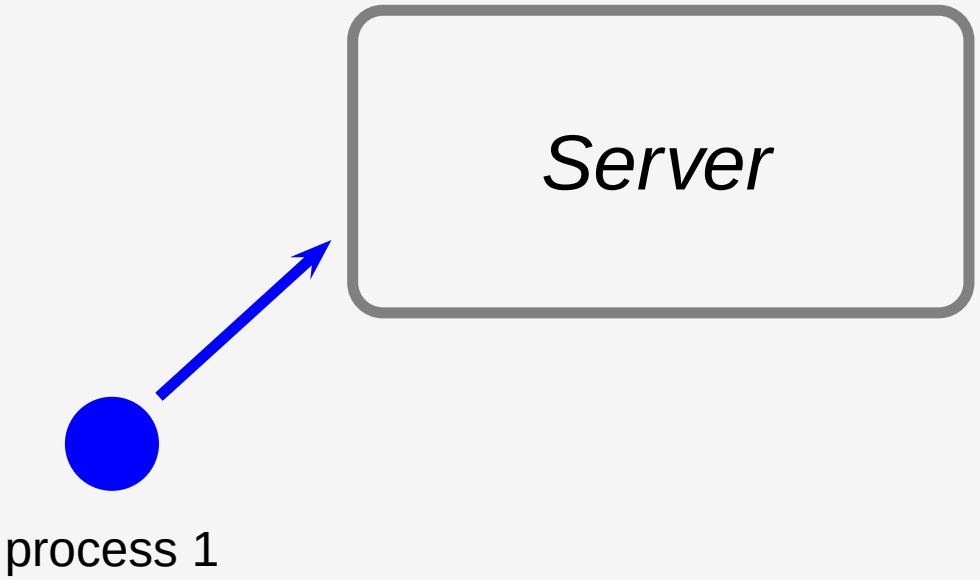
Systems with Unboundedly Many Processes



Systems with Unboundedly Many Processes



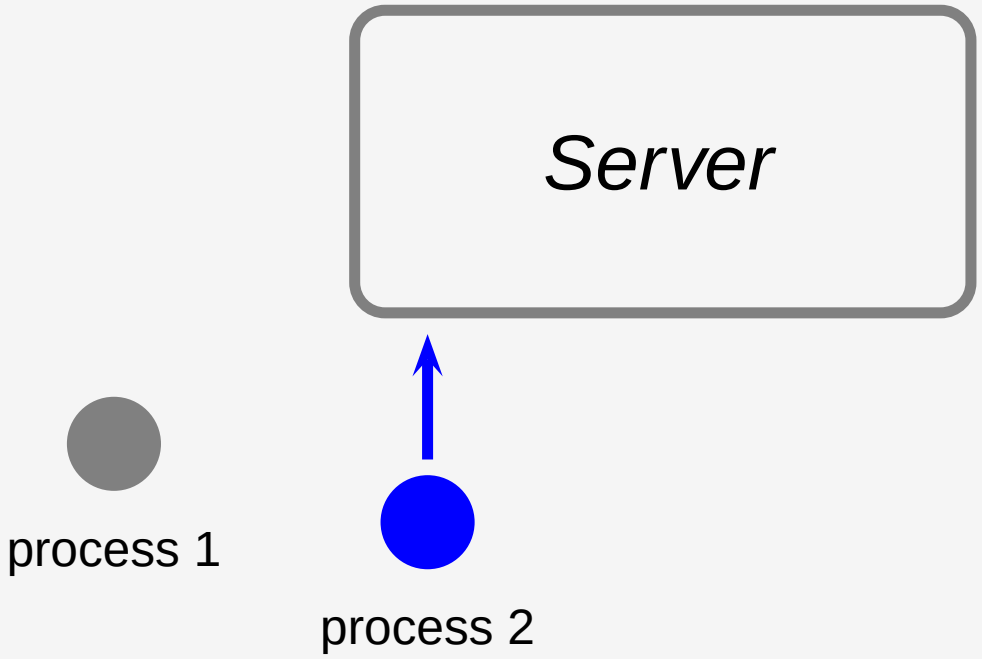
Systems with Unboundedly Many Processes



- A system run

req
1

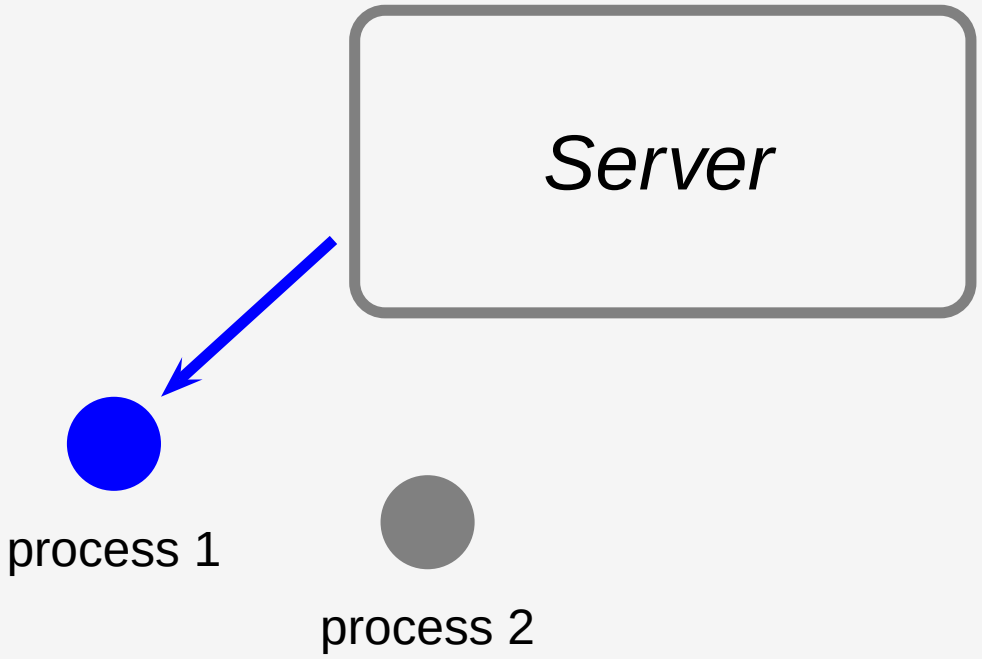
Systems with Unboundedly Many Processes



- A system run

req	*req*
1	2

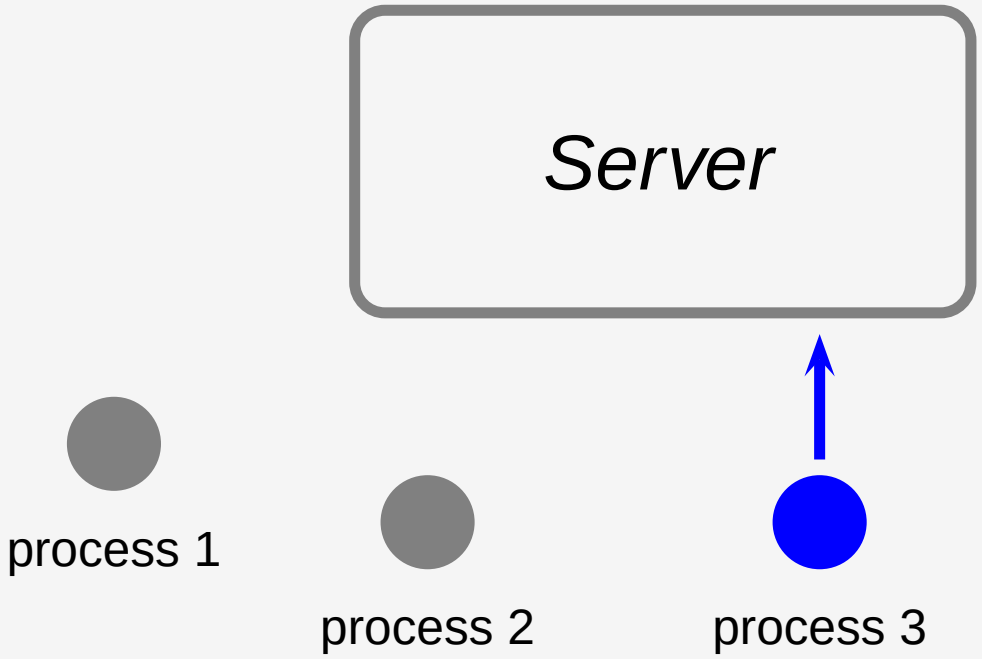
Systems with Unboundedly Many Processes



- A system run

req	*req*	*ack*
1	2	1

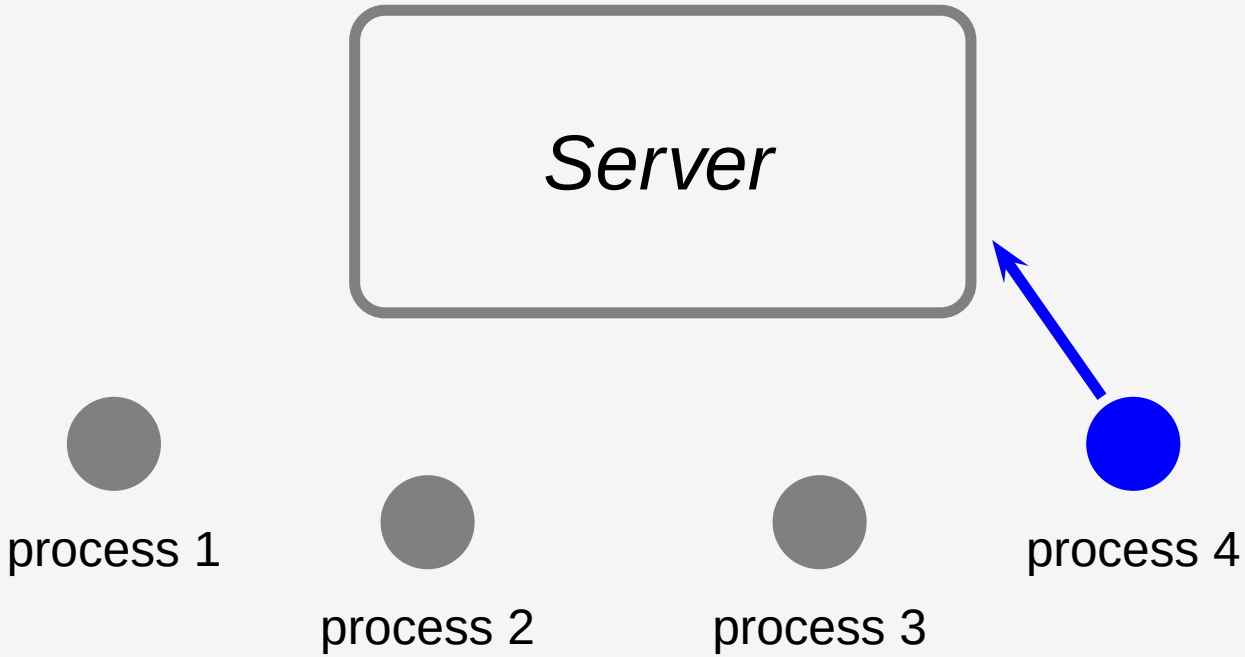
Systems with Unboundedly Many Processes



- A system run

req	*req*	*ack*	*req*
1	2	1	3

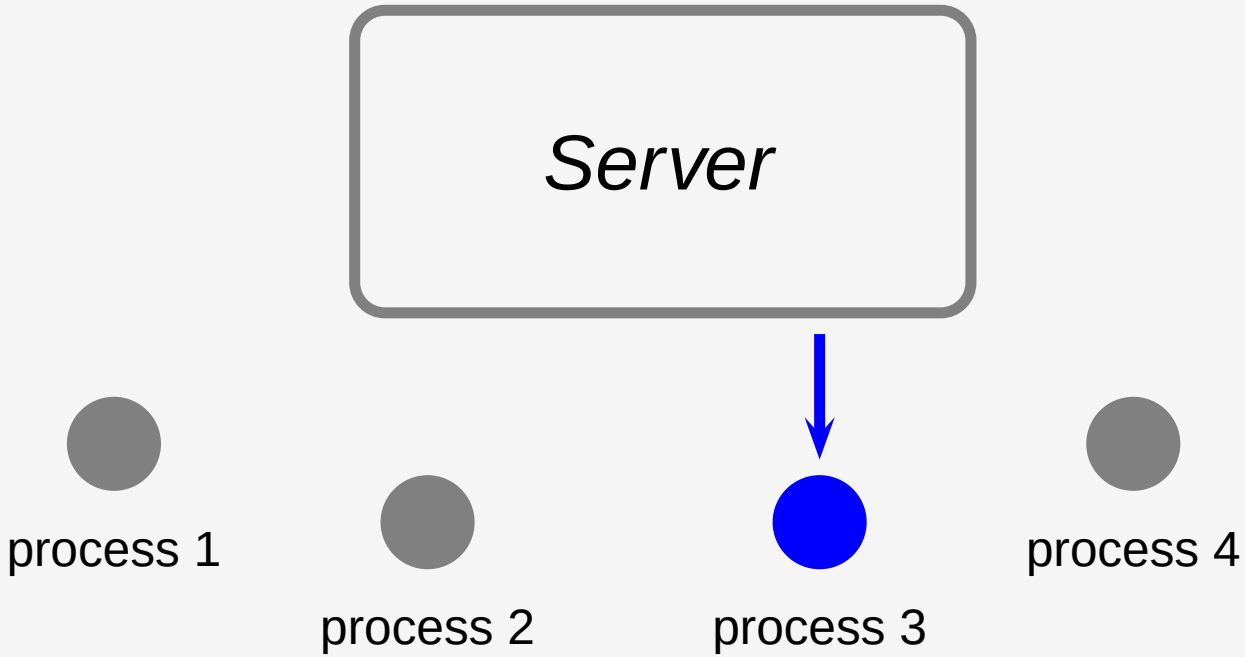
Systems with Unboundedly Many Processes



- A system run

req	*req*	*ack*	*req*	*req*
1	2	1	3	4

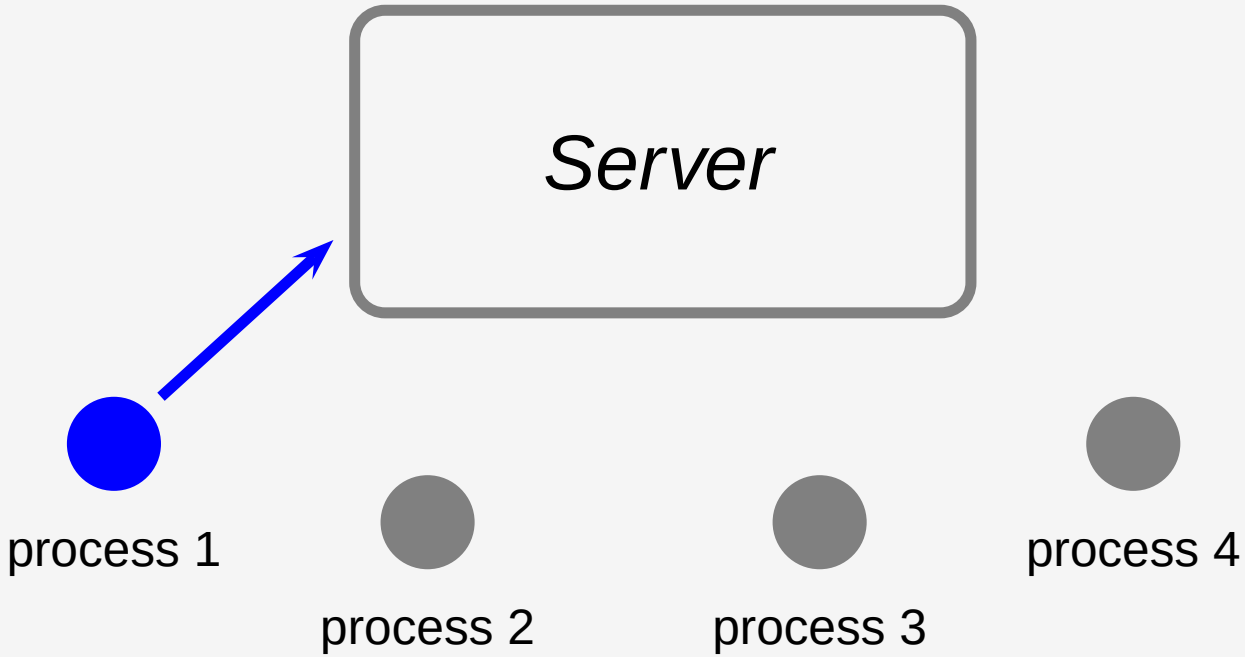
Systems with Unboundedly Many Processes



- A system run

req	*req*	*ack*	*req*	*req*	*ack*
1	2	1	3	4	3

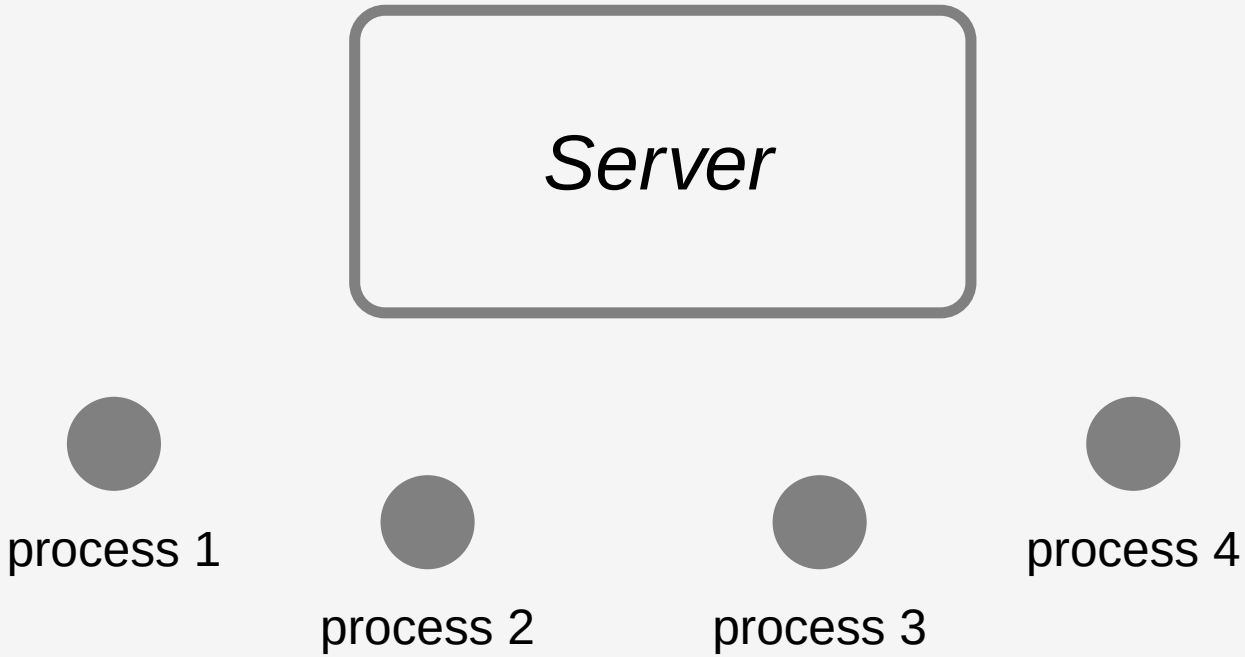
Systems with Unboundedly Many Processes



- A system run

req	*req*	*ack*	*req*	*req*	*ack*	*req*
1	2	1	3	4	3	1

Systems with Unboundedly Many Processes

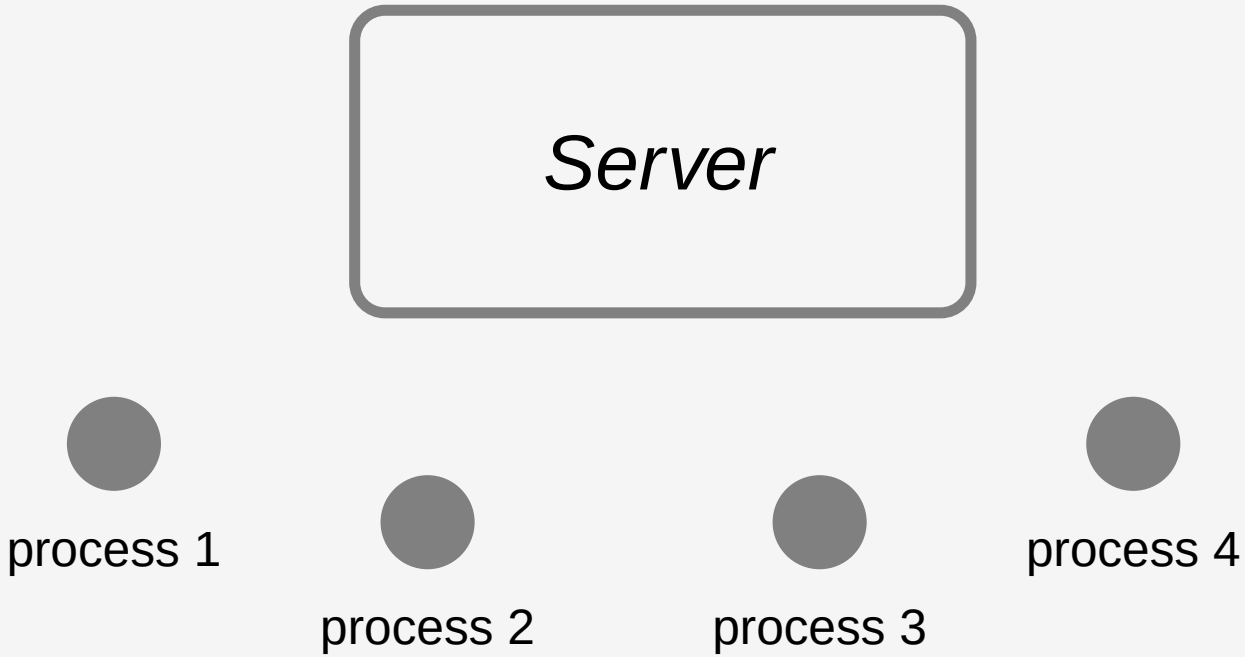


- A system run

req	*req*	*ack*	*req*	*req*	*ack*	*req*
1	2	1	3	4	3	1

...

Systems with Unboundedly Many Processes



- A system run

req	*req*	*ack*	*req*	*req*	*ack*	*req*
1	2	1	3	4	3	1

...

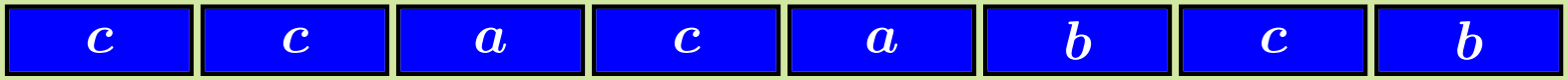
- A system property

„Every *req* of a process is followed by some *ack* for the same process."

$$\forall x \exists y (\mathit{req}(x) \rightarrow (x < y \wedge \mathit{ack}(y) \wedge x \sim y))$$

Words and Data Words

A Word over $\Sigma = \{a, b, c\}$



Words and Data Words

A Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b

A Data Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

Definition: Data Words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times \mathcal{D})^*$ is a data word over Σ

Words and Data Words

A Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b

A Data Word over $\Sigma = \{a, b, c\}$

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

- $\mathcal{D} = \{1, 2, 3, \ldots\}$

Definition: Data Words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times \mathcal{D})^*$ is a data word over Σ

Logics on Data Words – Data Logics

- Even very weak logics on data words have an undecidable satisfiability problem.
  - $\mathbf{LTL}$ in general not decidable [Demri et al. 06]
  - $\mathbf{FO^3}$ (with only three variables) not decidable [Bojańczyk et al. 06]
  - $\mathbf{FO^2}$ decidable but exact complexity not known [Bojańczyk et al. 06]

Known Results

DECIDABLE

$$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$$

Known Results

DECIDABLE

$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$

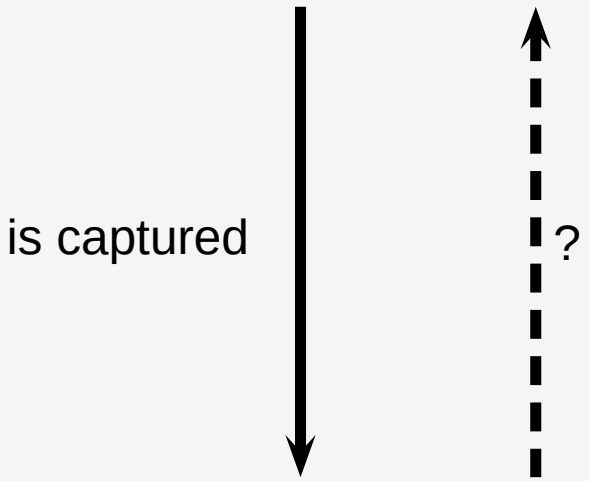
is captured

Data Automata

Known Results

DECIDABLE

$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$



Data Automata

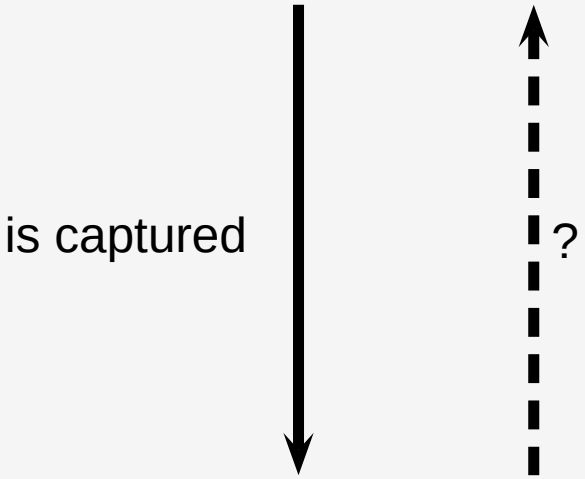
Known Results

2NEXPTIME

$\mathbf{EMSO^2}(\sim, \mathbf{Suc})$

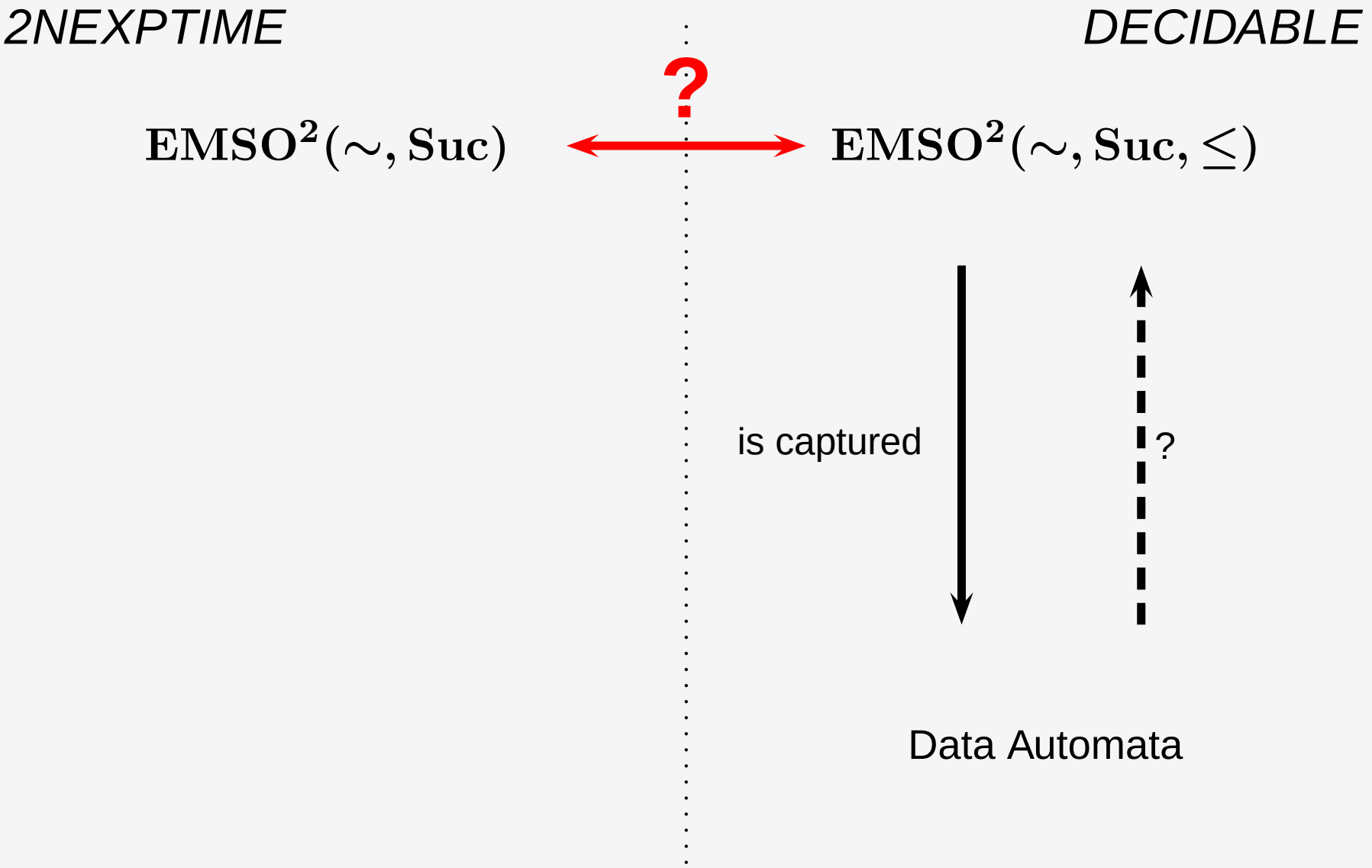
DECIDABLE

$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$

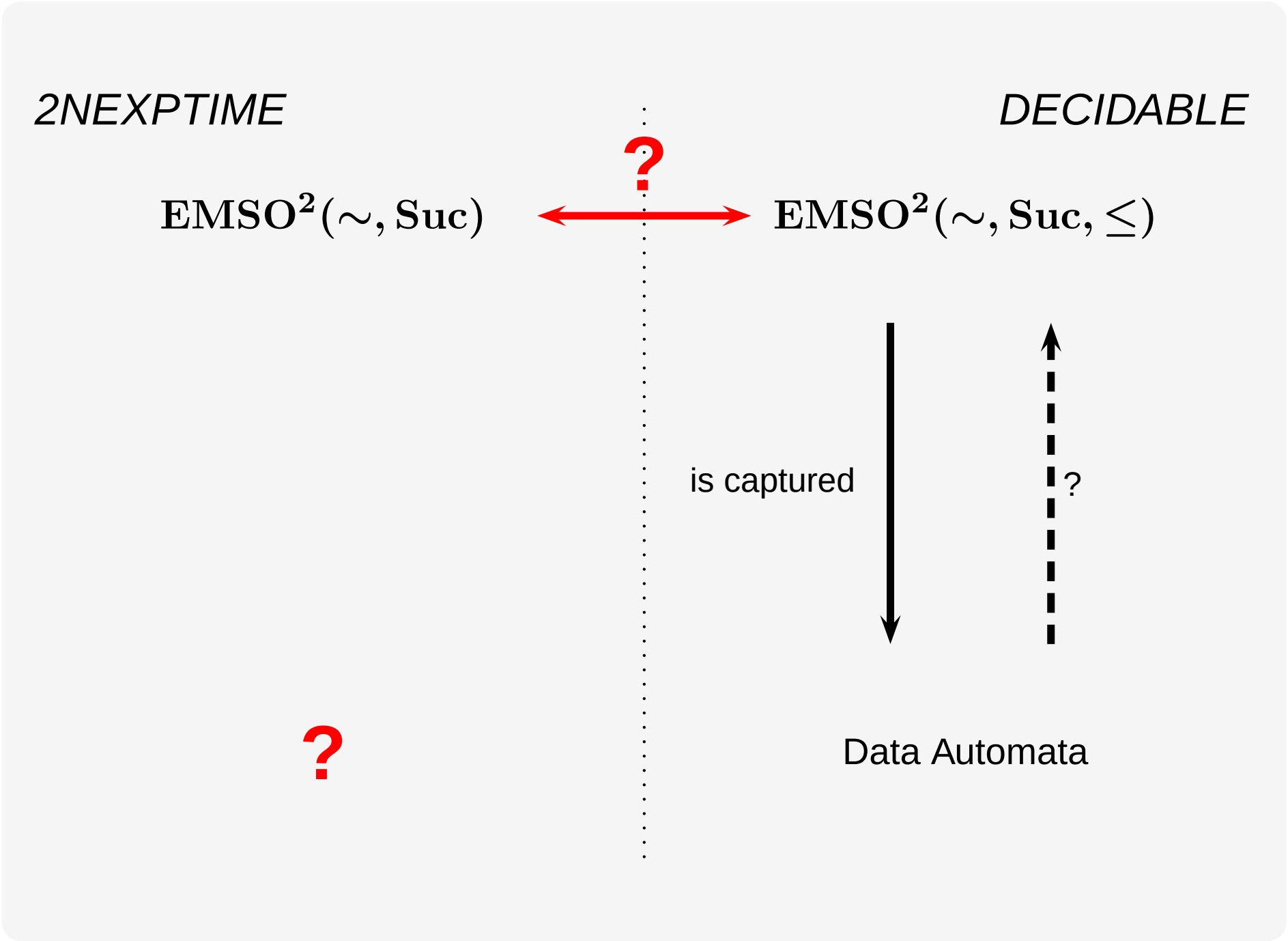


Data Automata

Known Results



Known Results



Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form
  $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$ where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$ where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$ where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$ where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2
		x					

$\forall x$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$ where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2
		x			y		

$\forall x \exists y$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2
		x			y		

$\forall x \exists y \big(a(x)$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value.”

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y$$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y)\big)$$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y) \wedge x \sim y) \big)$$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y) \wedge x \sim y) \big)$$

Example

„There are two neighbouring positions with different data values."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y) \wedge x \sim y) \big)$$

Example

„There are two neighbouring positions with different data values."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

x

$\exists x$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y) \wedge x \sim y) \big)$$

Example

„There are two neighbouring positions with different data values."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \quad y$

$$\exists x \exists y$$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y) \wedge x \sim y)\big)$$

Example

„There are two neighbouring positions with different data values."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \quad y$

$$\exists x \exists y \big(\mathbf{Suc}(x, y)\big)$$

Existential Monadic Second Order Logic on Data Words

Existential Monadic Second Order Logic with two position variables ($\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$):

- formulas are of the form $\exists M_1 \ldots \exists M_n \, \varphi$
  where φ is a first order formula
  - uses only two position variables
  - can test whether a position is contained in some set M_i
  - can test whether a position is labelled by some symbol
  - can use order and successor relations on positions
  - can test whether the data values at two positions are equal

Example

„Every a is followed by some b with the same data value."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \longrightarrow y$

$$\forall x \exists y \big(a(x) \rightarrow (x < y \wedge b(y) \wedge x \sim y)\big)$$

Example

„There are two neighbouring positions with different data values."

c	c	a	c	a	b	c	b
1	4	3	2	2	3	7	2

$x \quad y$

$$\exists x \exists y \big(\mathbf{Suc}(x, y) \wedge \neg(x \sim y)\big)$$

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$

σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8
1	1	3	2	2	3	1	2

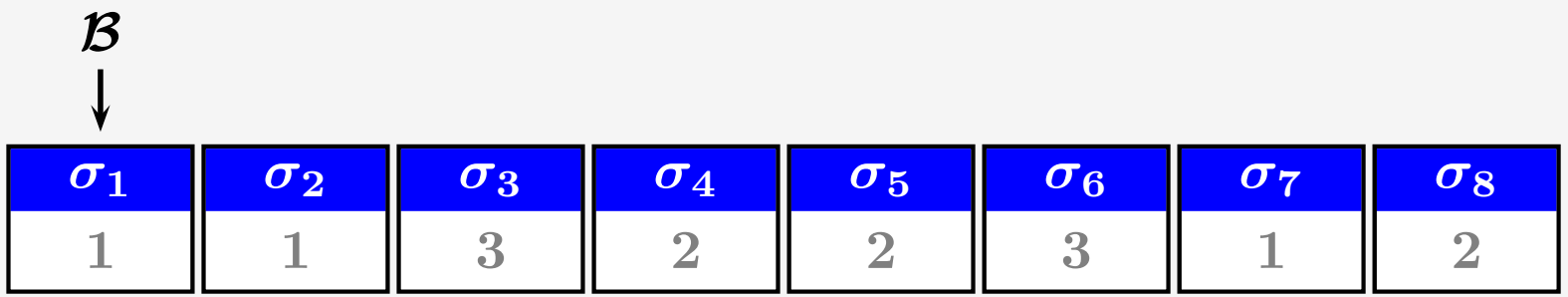
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



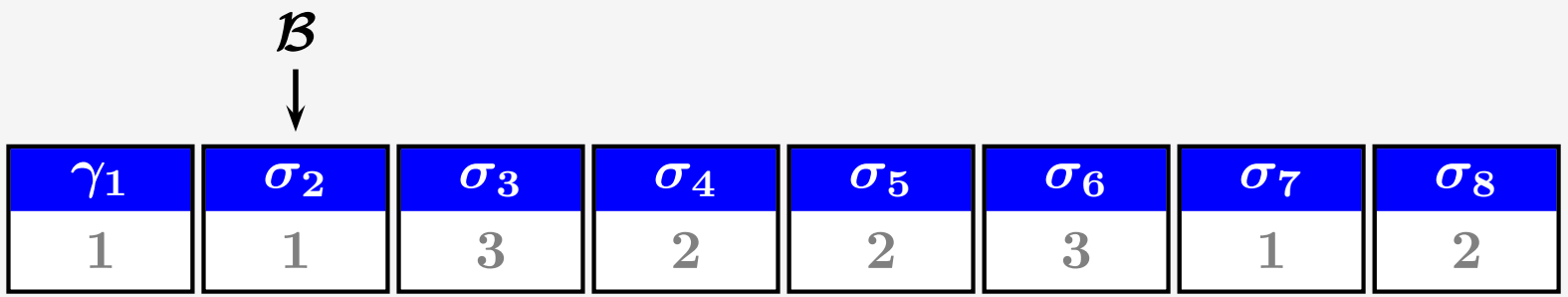
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



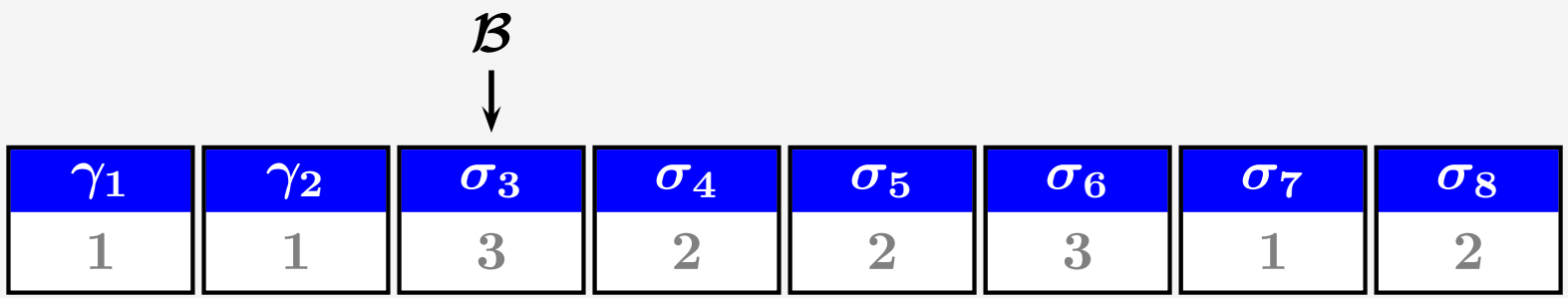
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



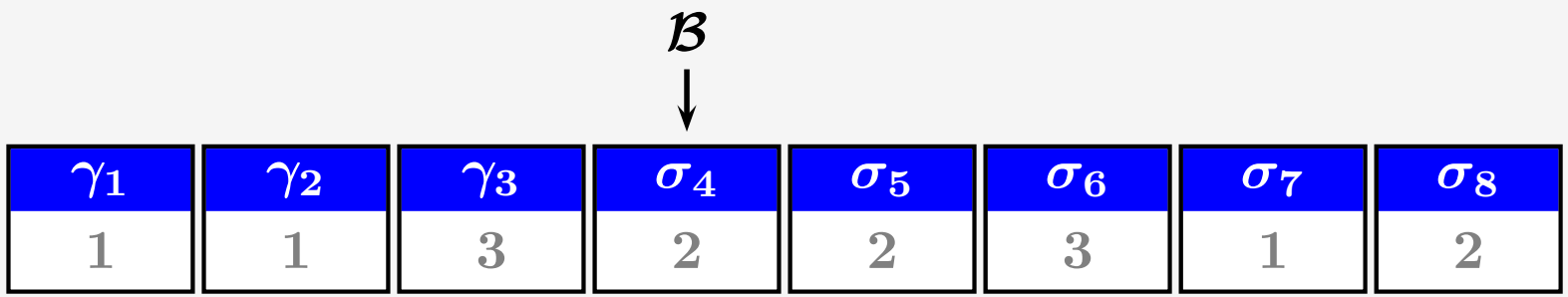
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



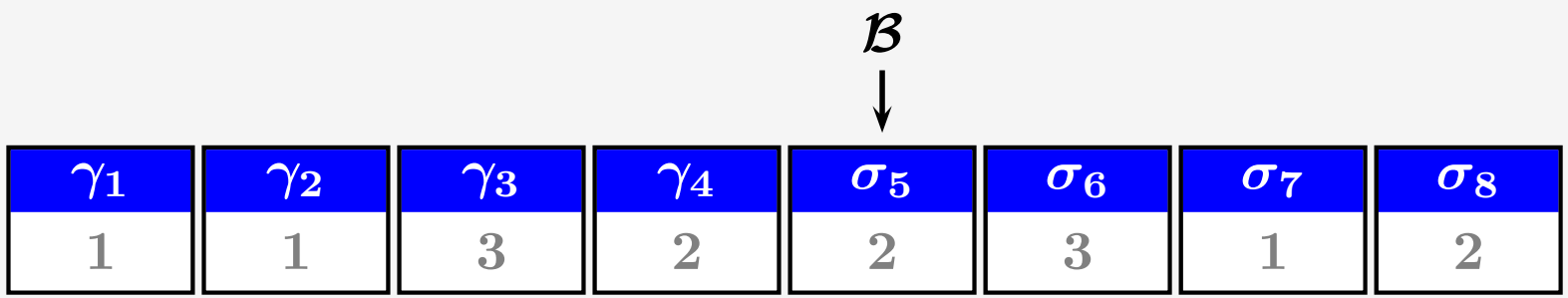
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



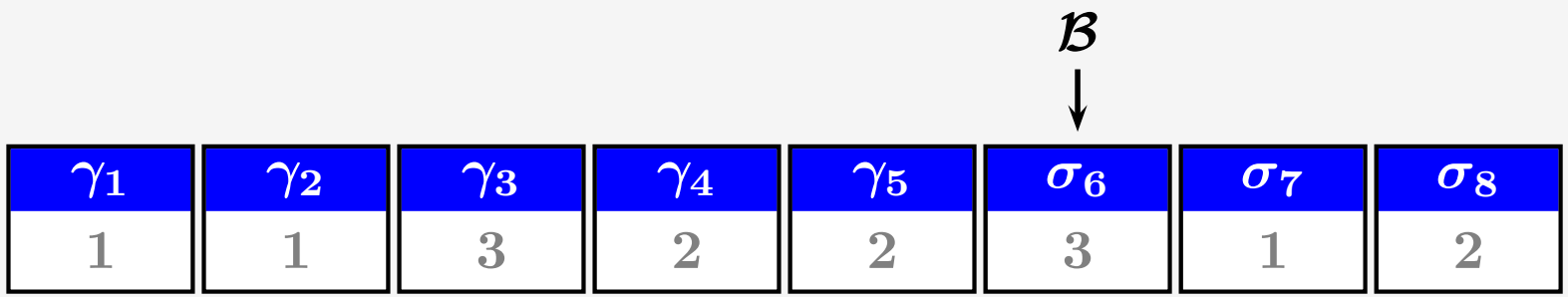
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



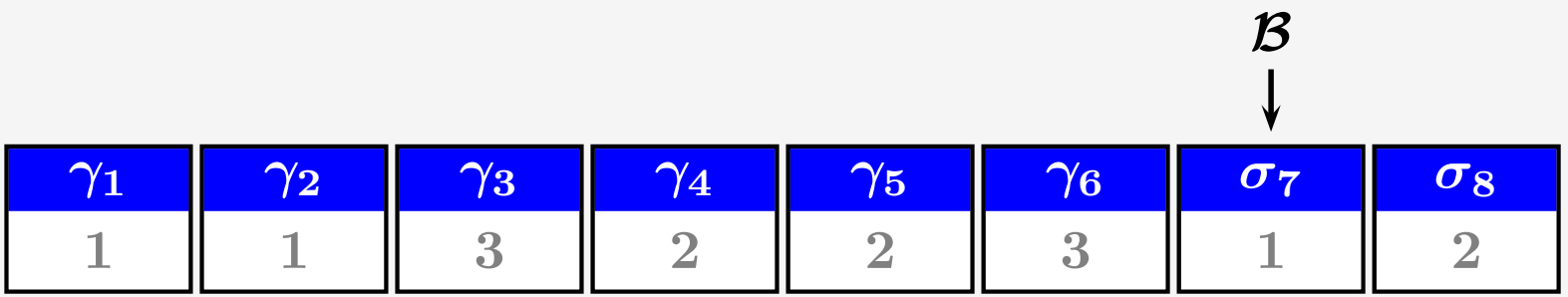
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



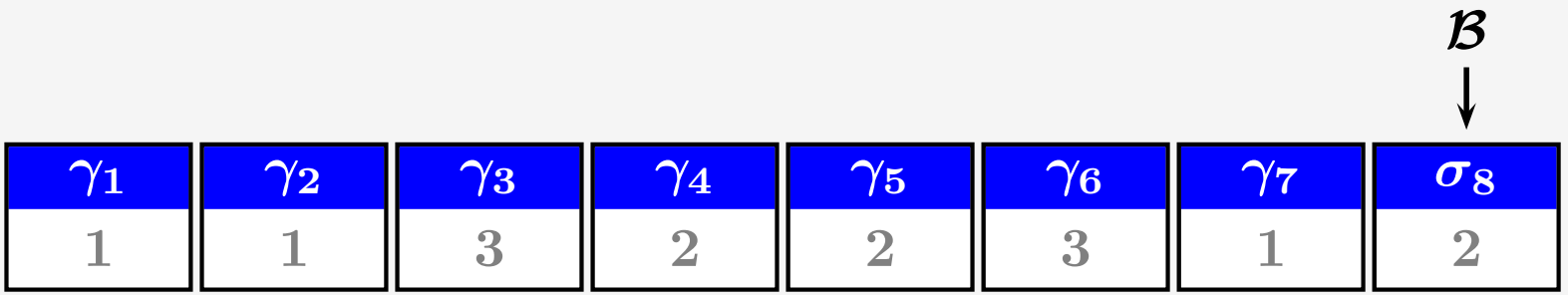
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
1	1	3	2	2	3	1	2

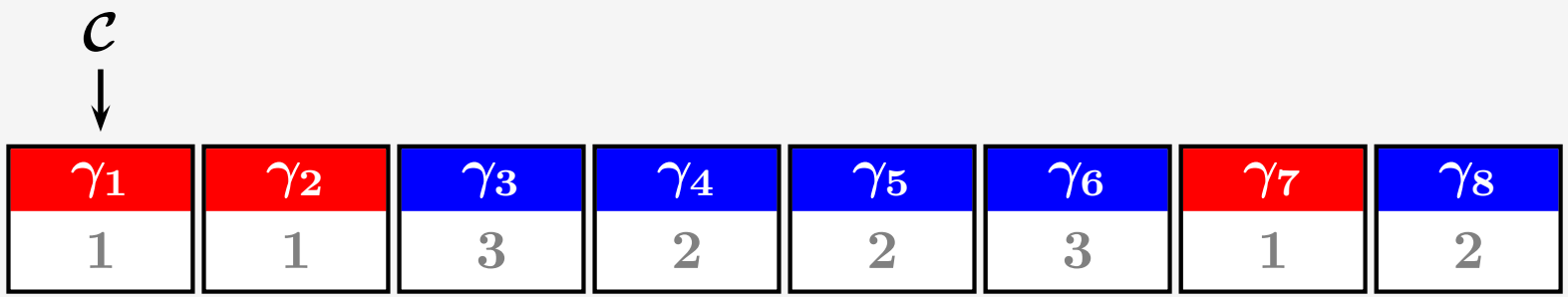
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



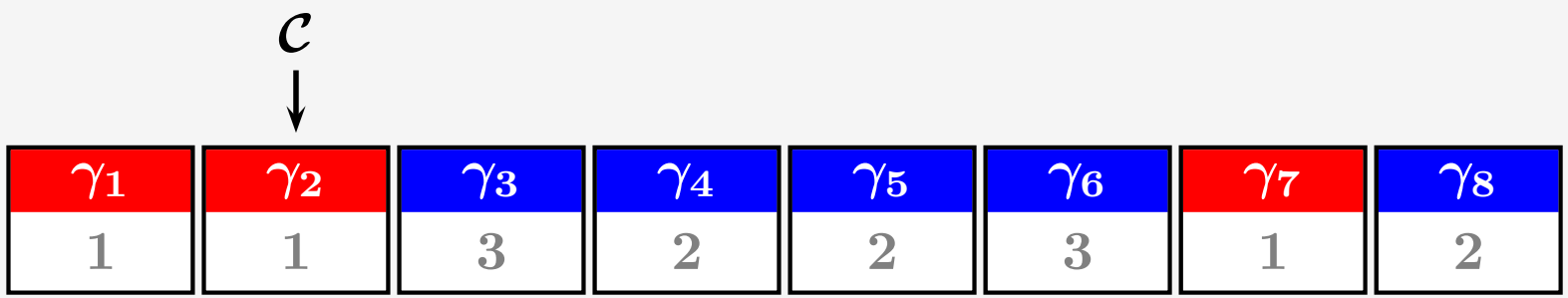
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



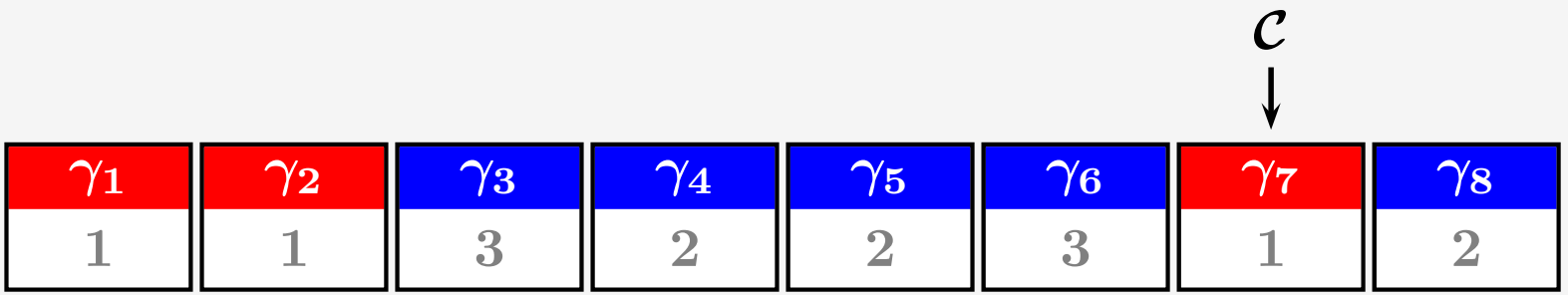
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
1	1	3	2	2	3	1	2

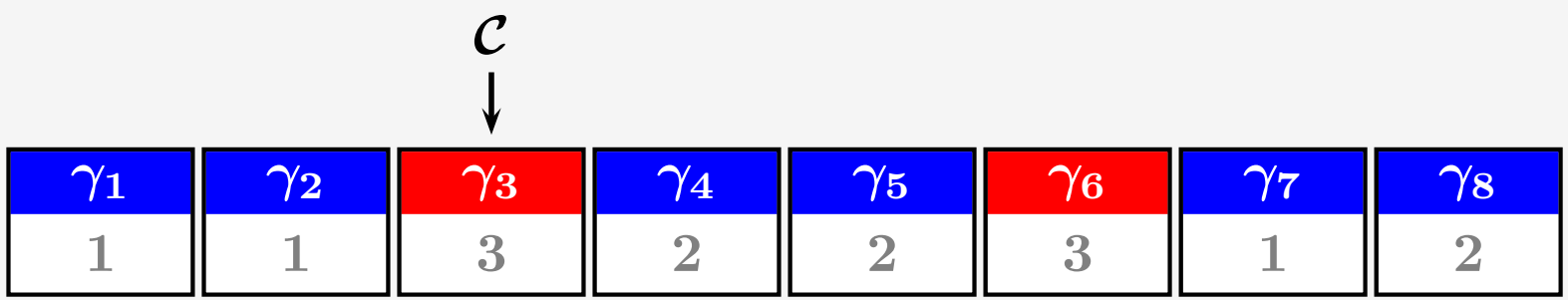
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



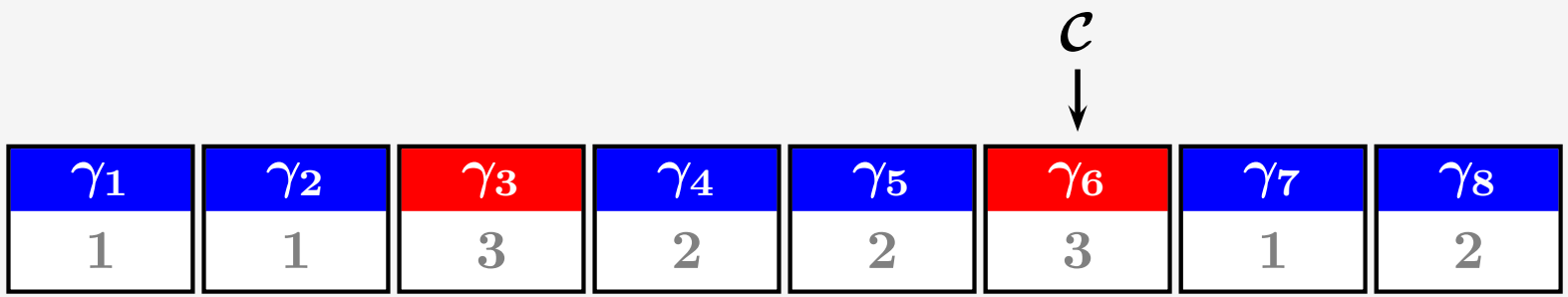
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
1	1	3	2	2	3	1	2

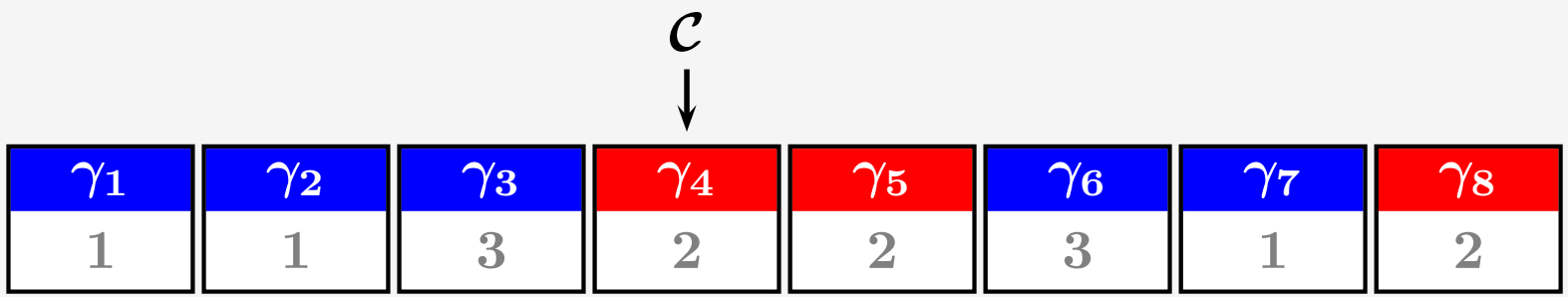
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



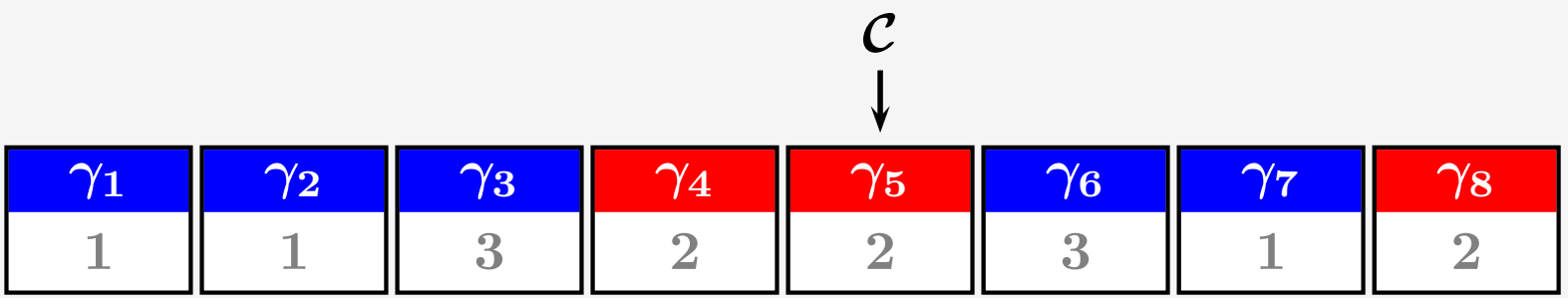
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



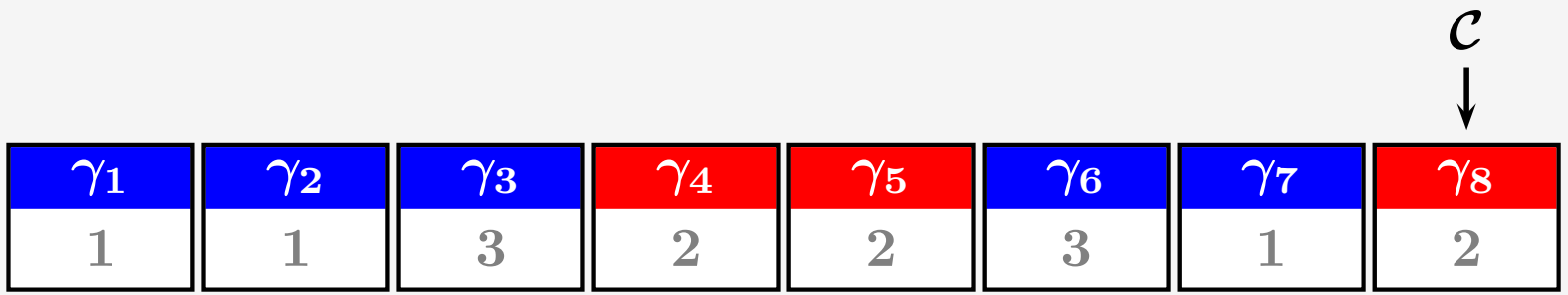
Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

$(\mathcal{B}, \mathcal{C})$

γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

$-$	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata ($\mathbb{DA}$) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

$-$	$-$	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

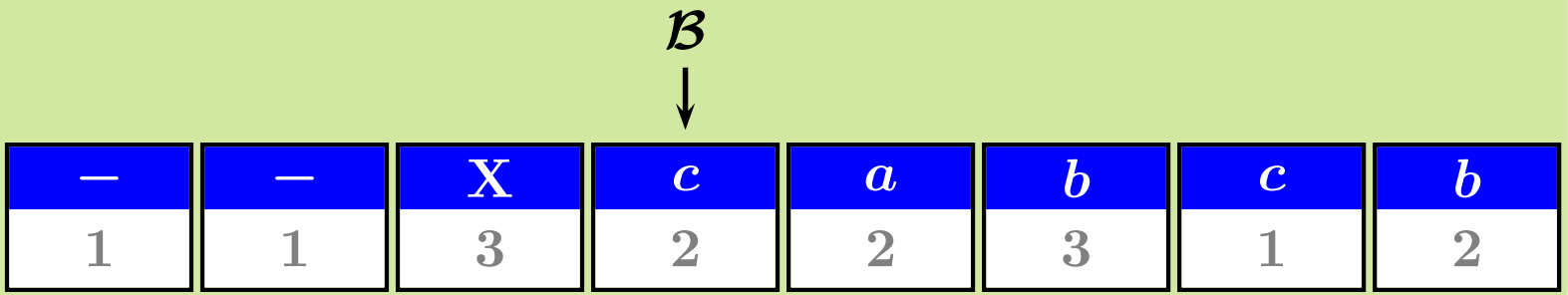
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

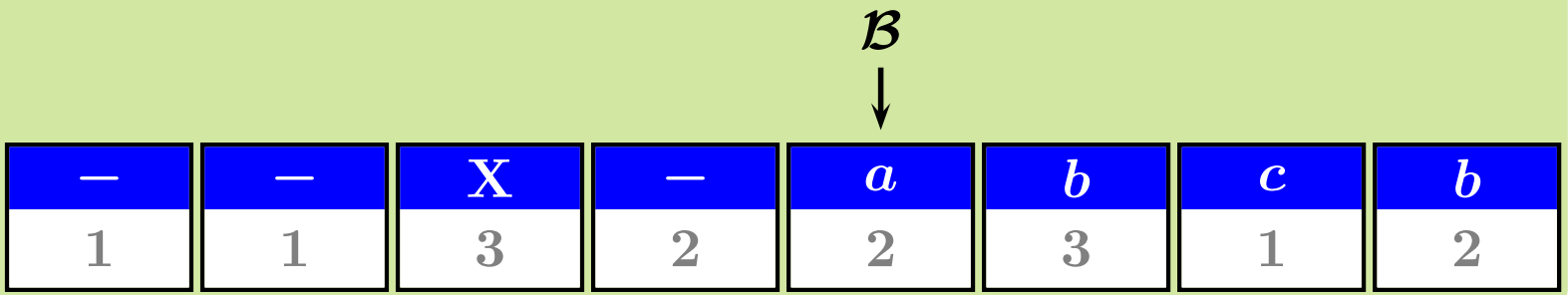
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

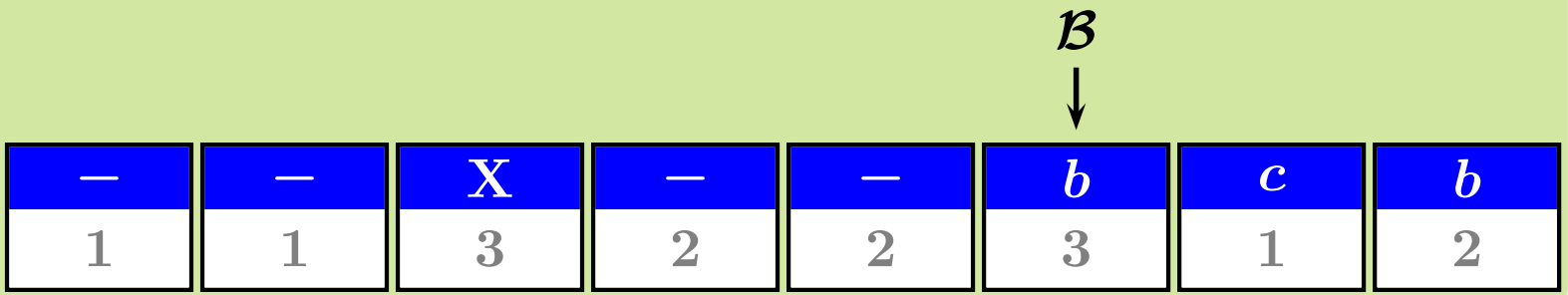
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

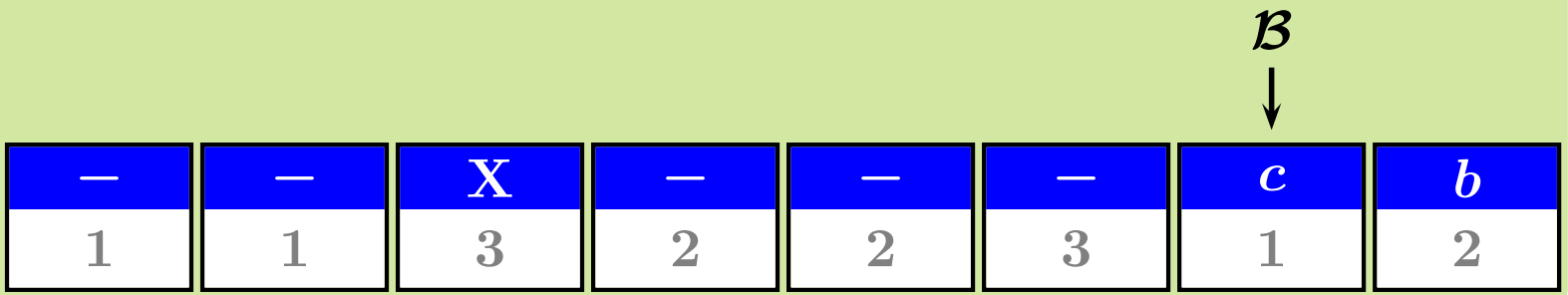
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

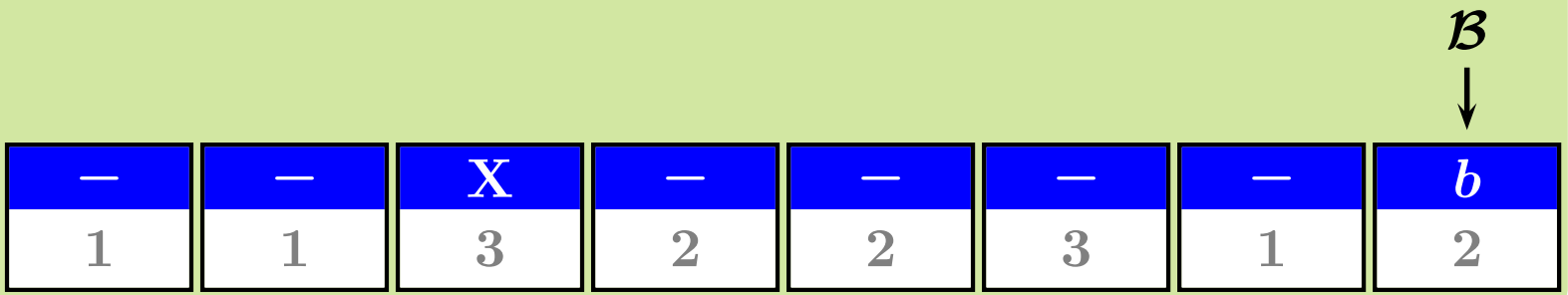
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

–	–	X	–	–	–	–	X
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

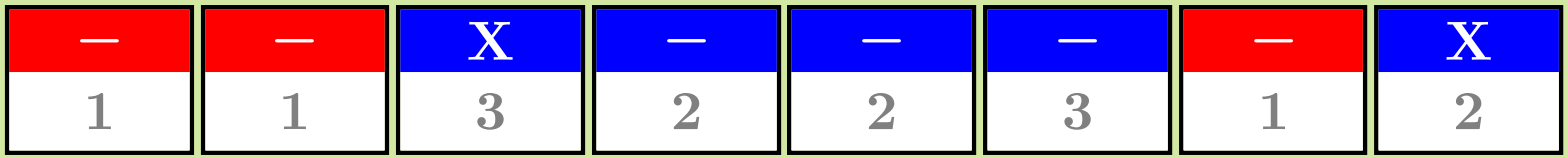
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

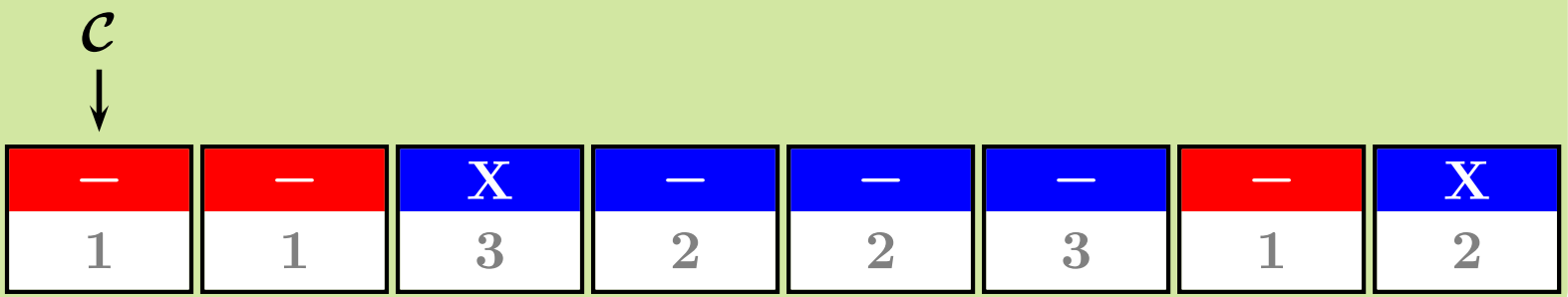
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

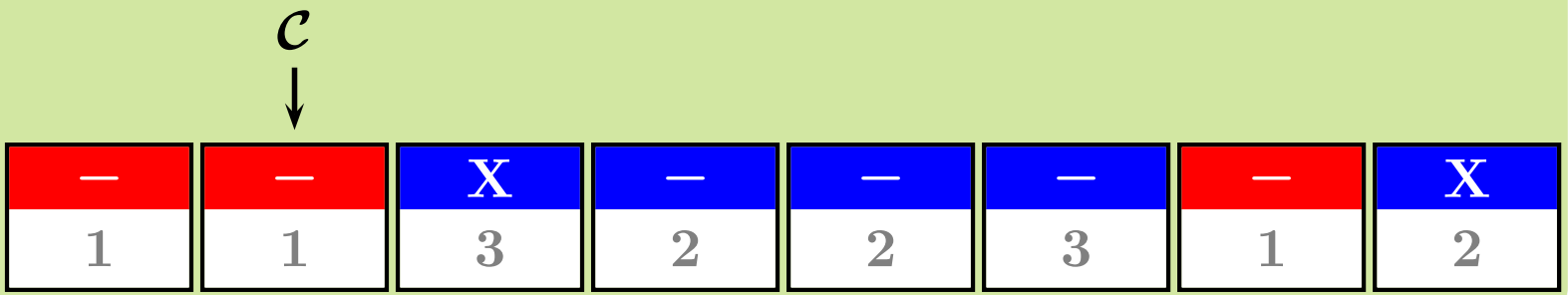
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

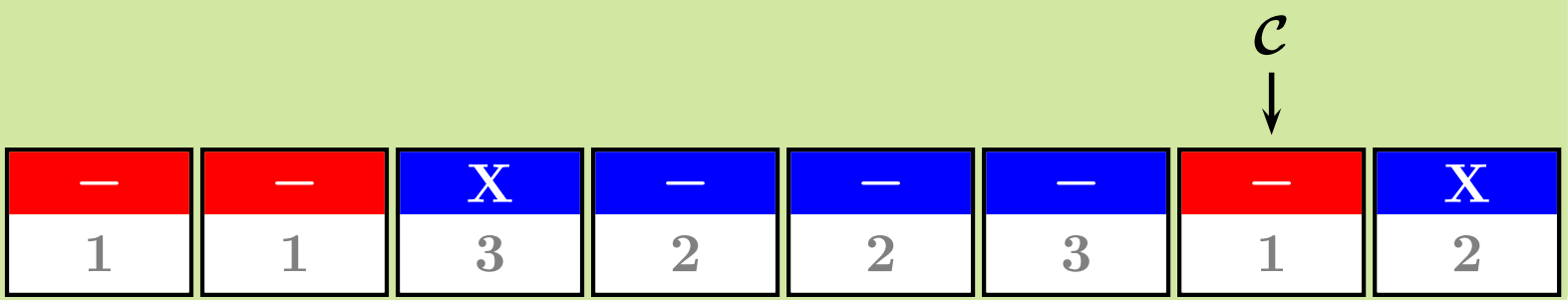
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

–	–	X	–	–	–	–	X
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

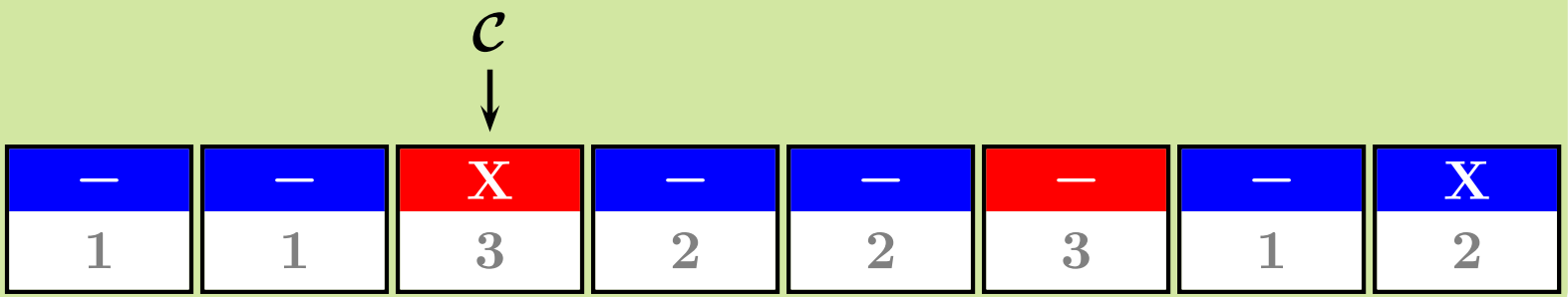
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

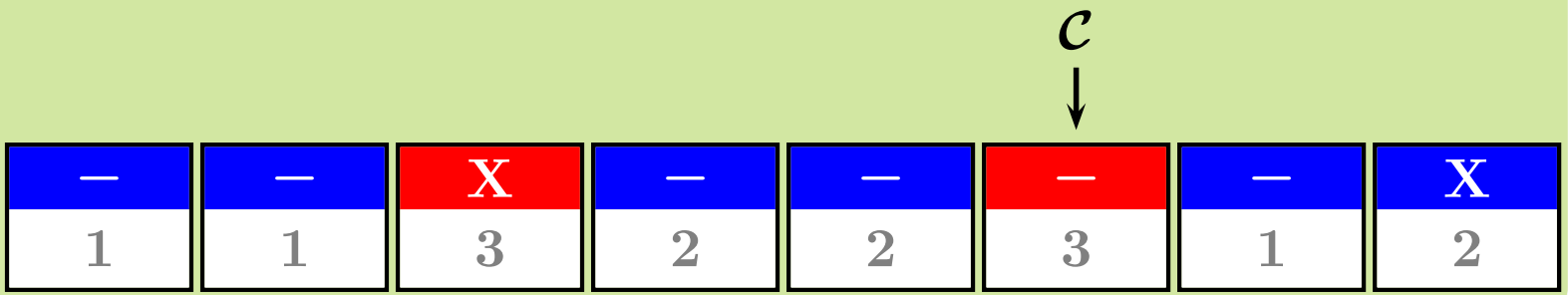
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

–	–	X	–	–	–	–	X
1	1	3	2	2	3	1	2

Data Automata

Data Automata ($\mathbb{DA}$) [Bojańczyk et al. 06]

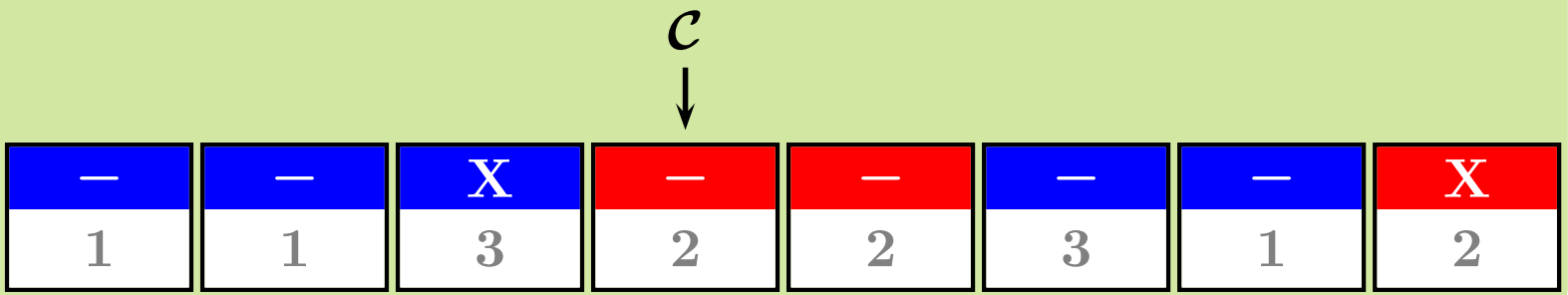
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value."

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

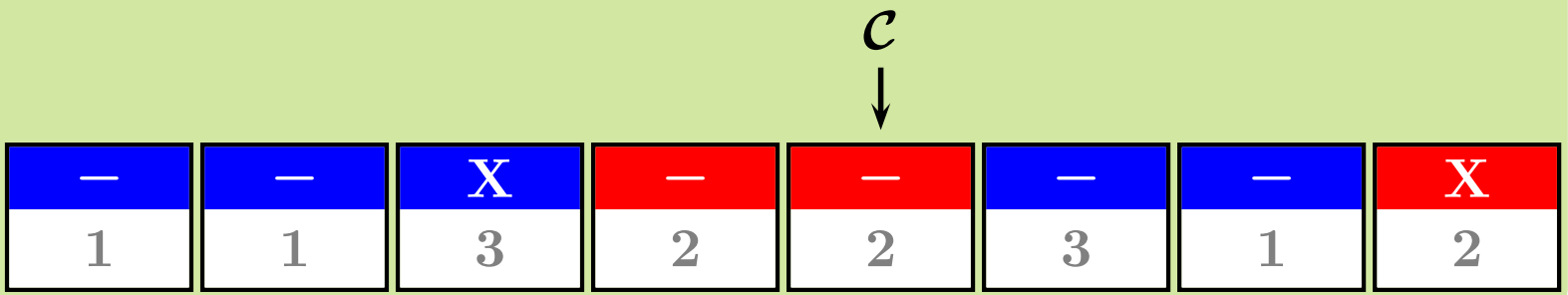
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

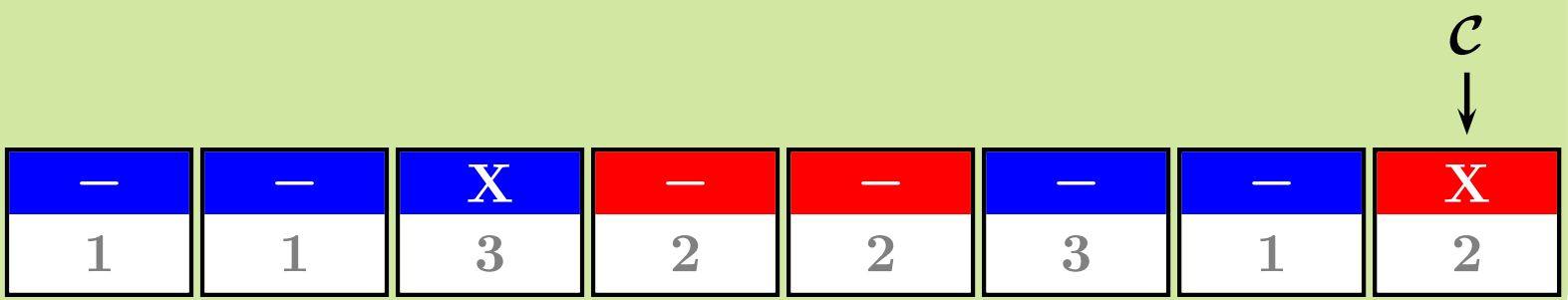
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value."

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„There is an a which is followed by some b with a different data value.”

$(\mathcal{B}, \mathcal{C})$

–	–	X	–	–	–	–	X
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

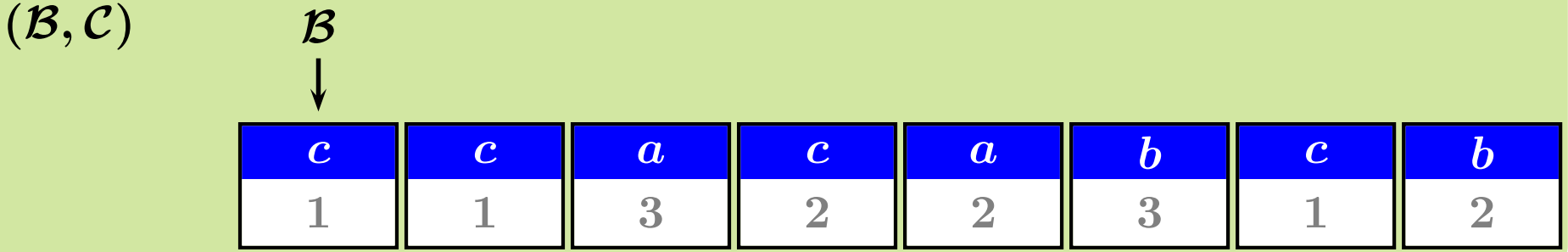
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

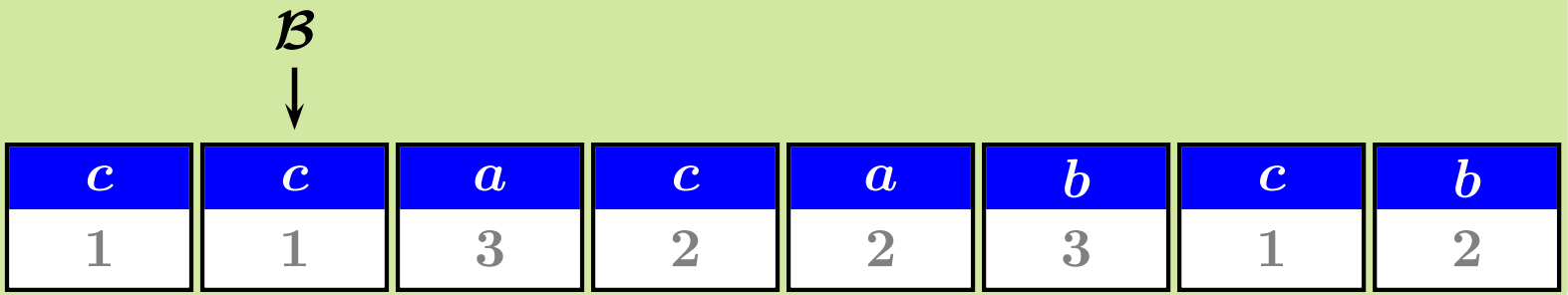
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

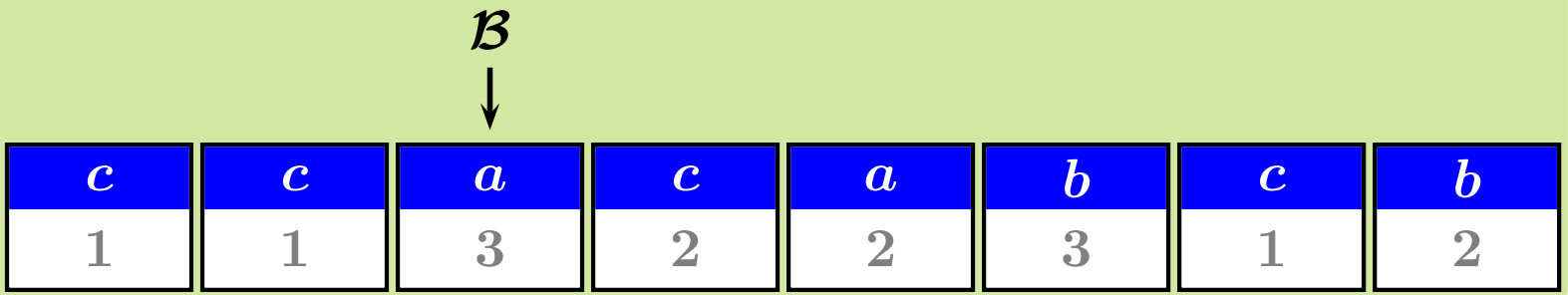
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

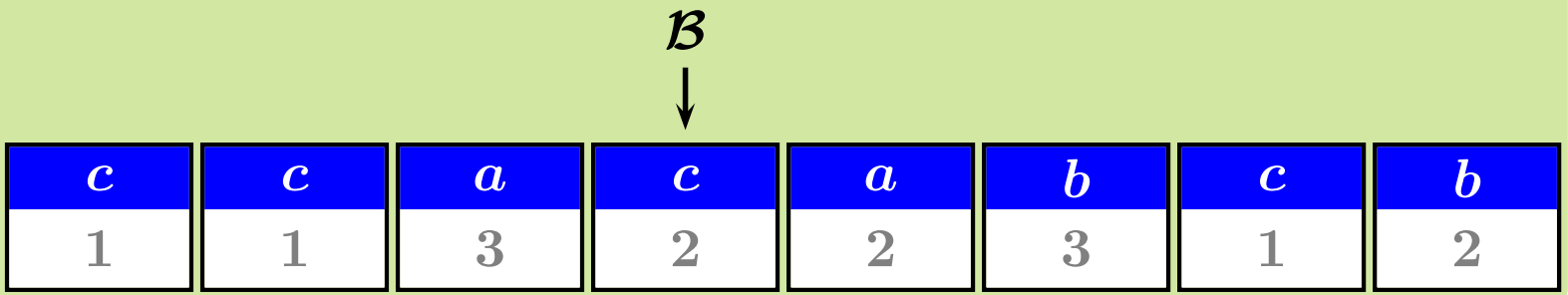
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

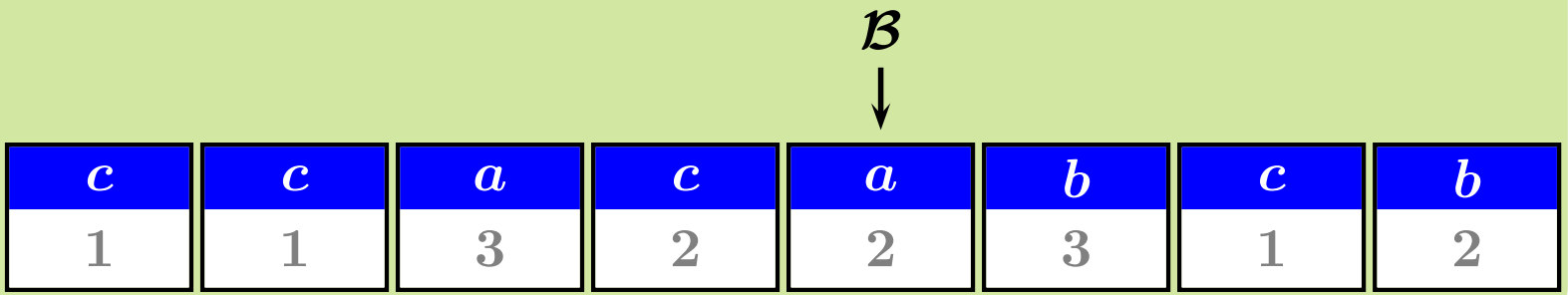
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

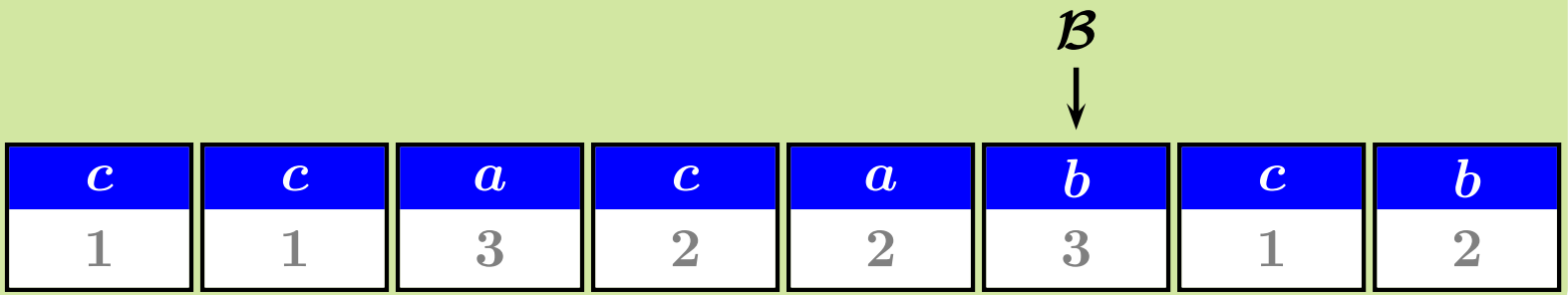
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

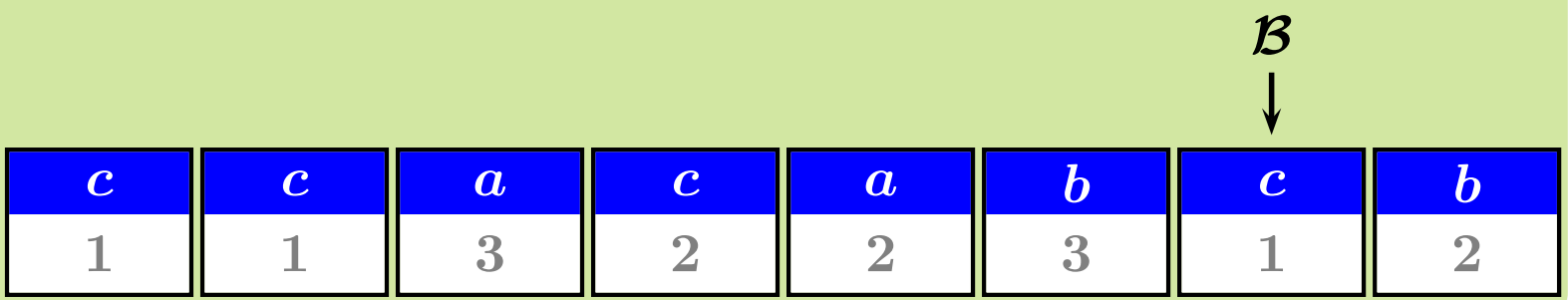
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

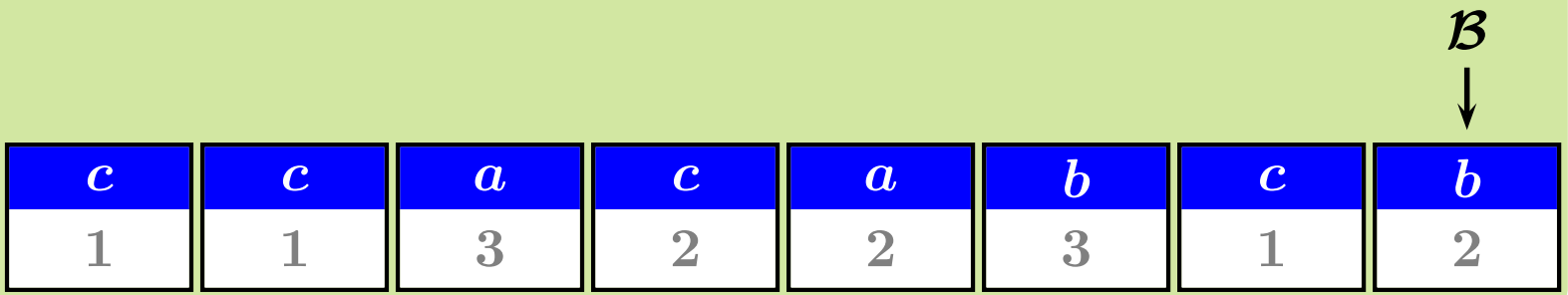
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

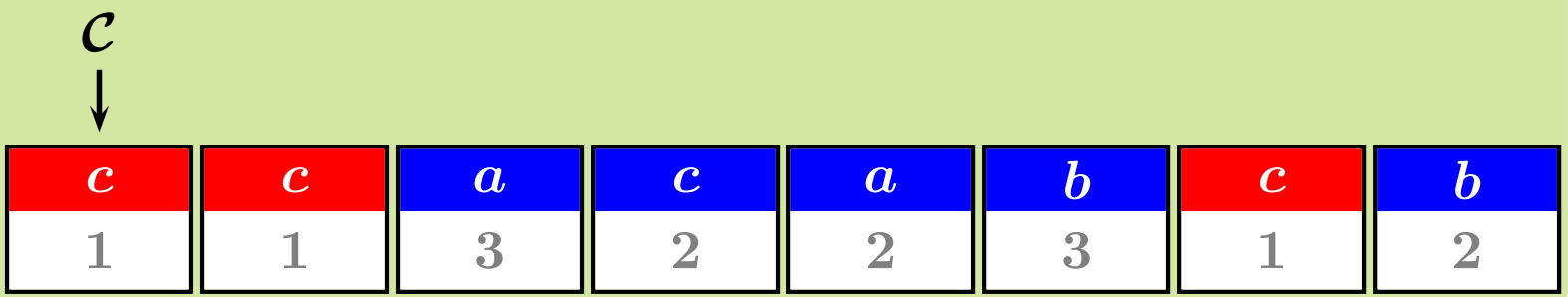
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

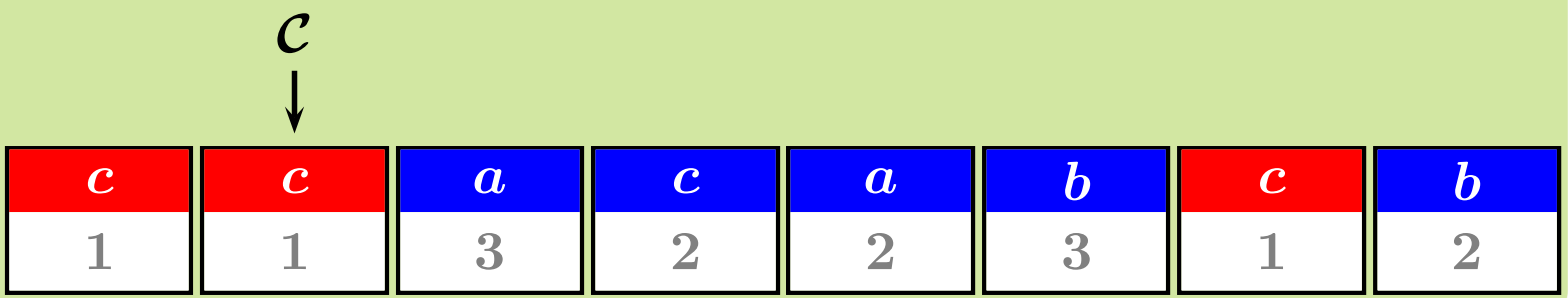
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

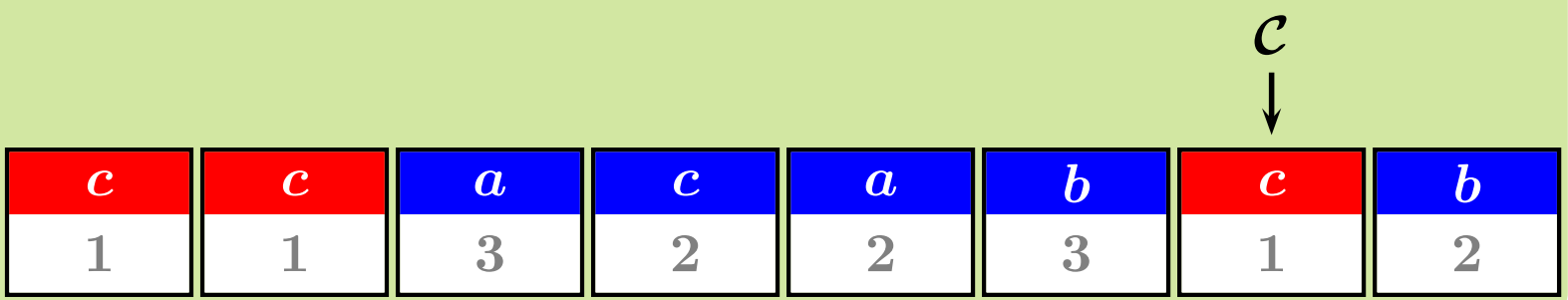
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

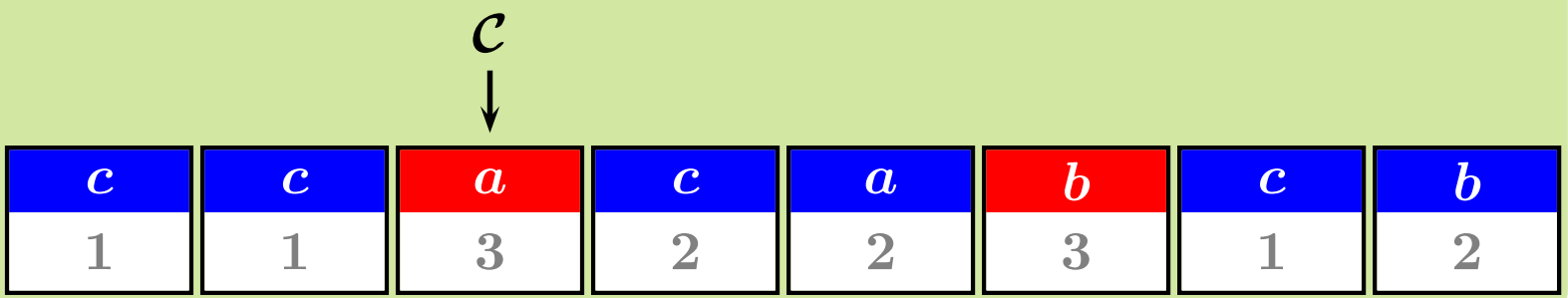
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata ($\mathbb{DA}$) [Bojańczyk et al. 06]

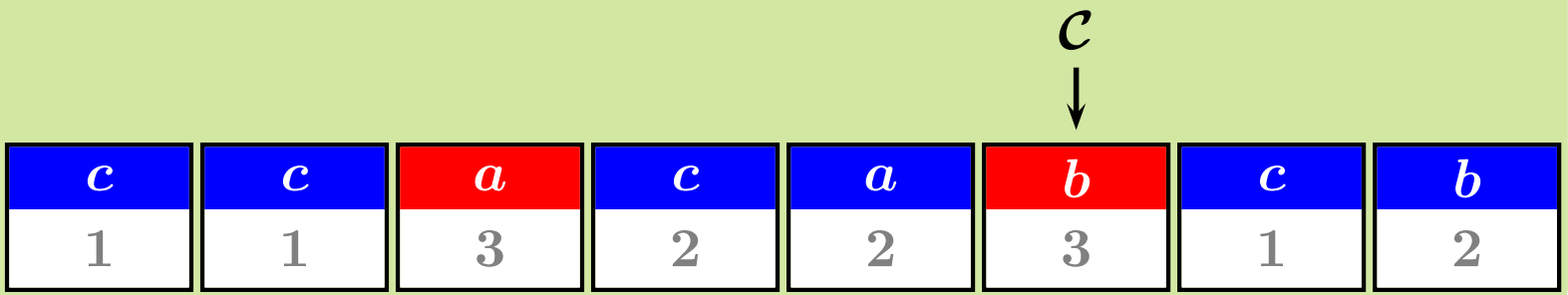
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

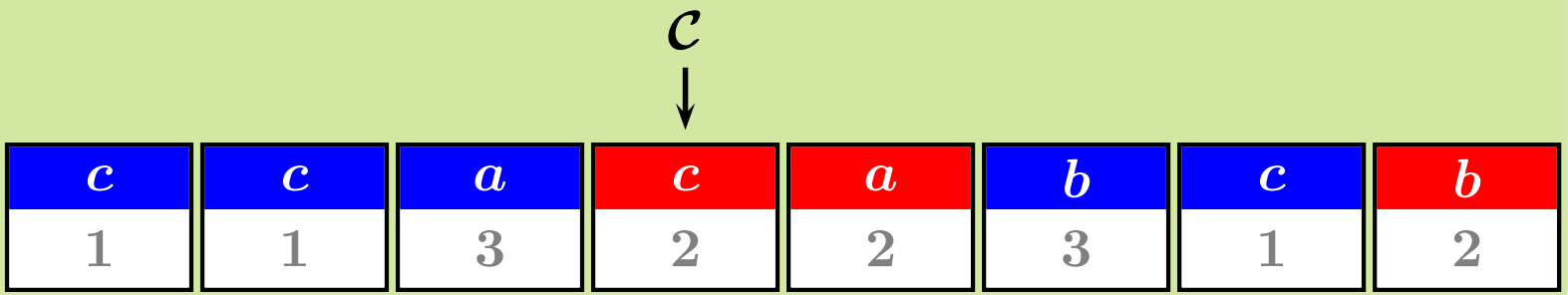
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value."

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

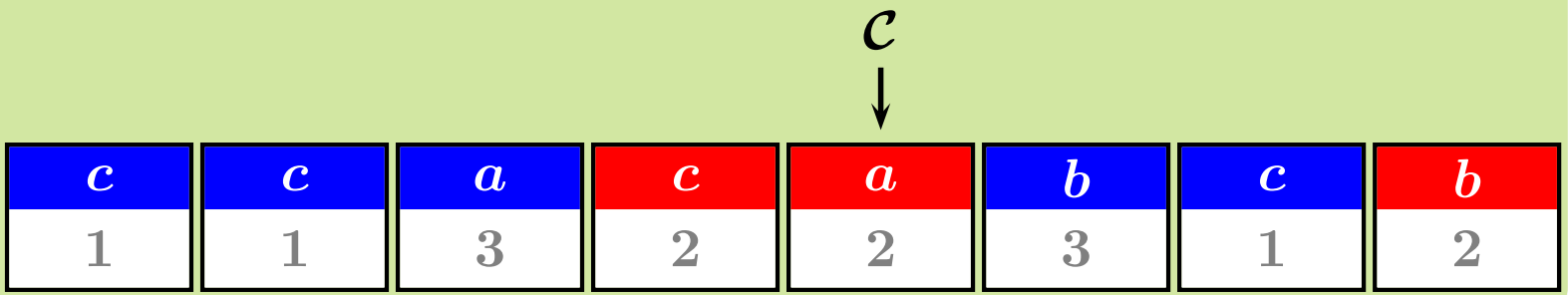
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata ($\mathbb{DA}$) [Bojańczyk et al. 06]

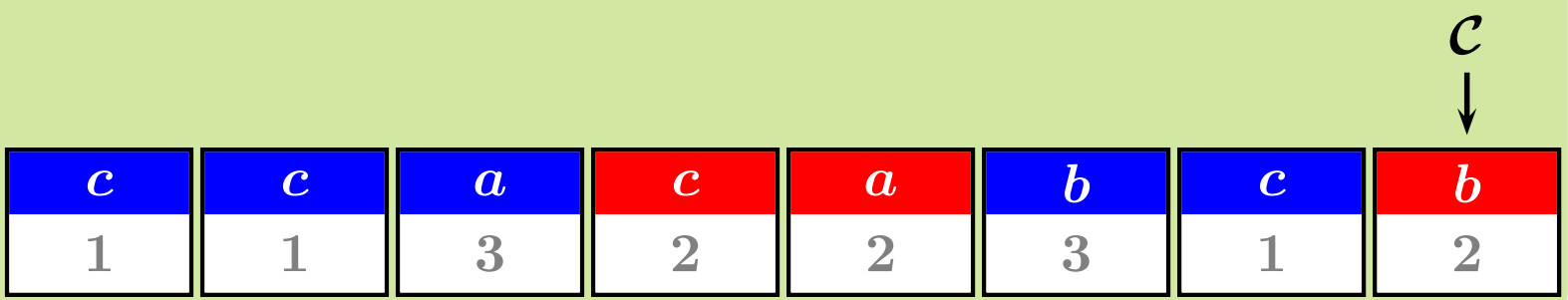
A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Data Automata

Data Automata (DA) [Bojańczyk et al. 06]

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

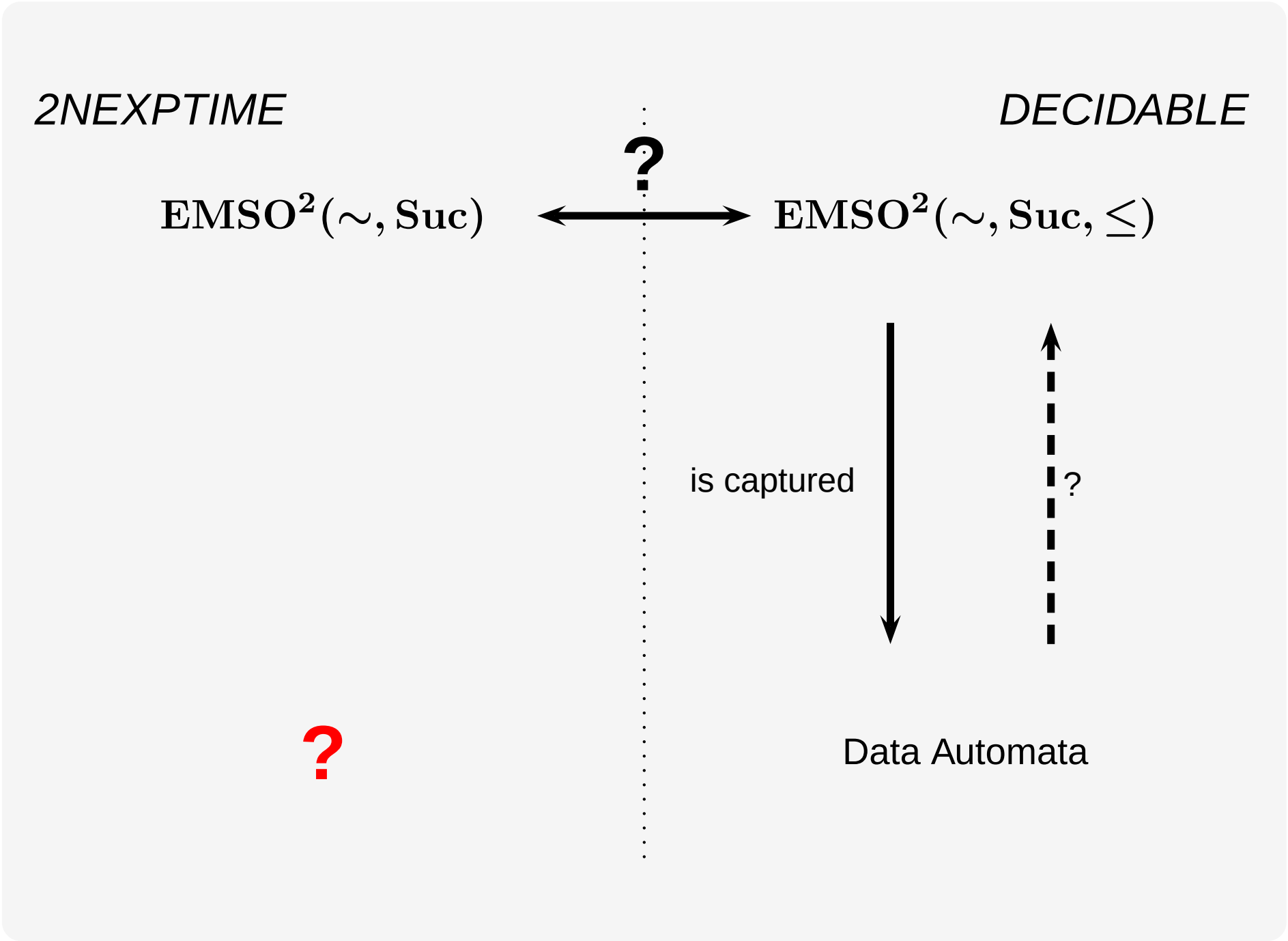
Example

„Every a is followed by some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

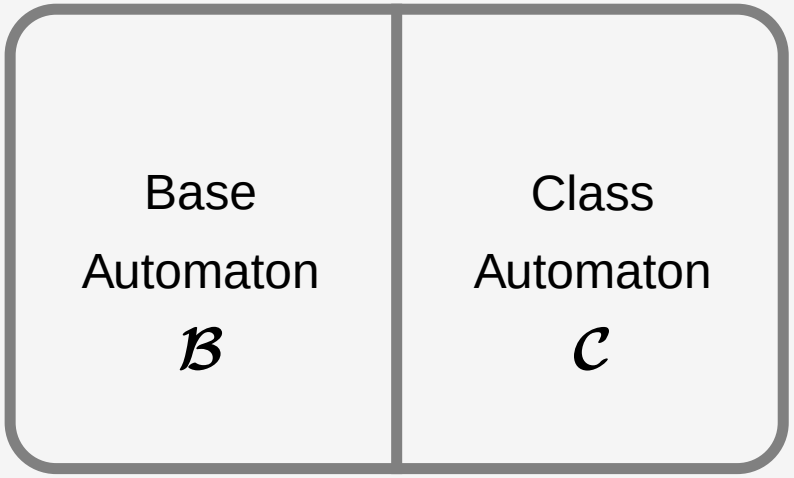
c	c	a	c	a	b	c	b
1	1	3	2	2	3	1	2

Known Results



Weak Data Automata

Data Automata



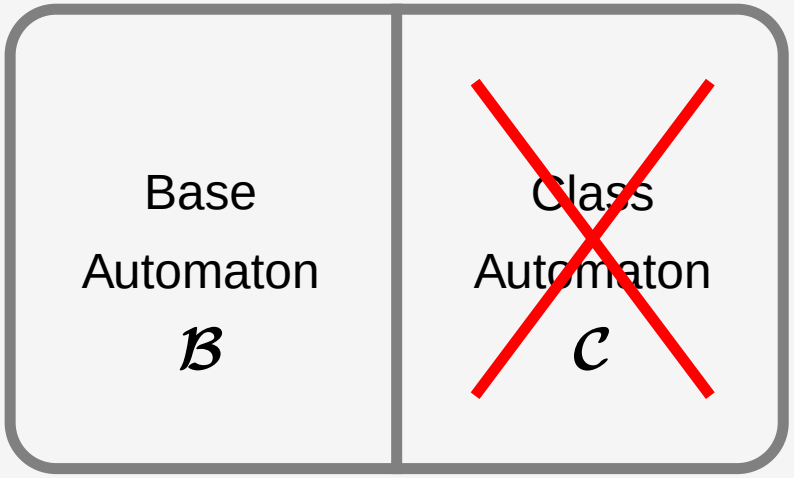
Definition

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Weak Data Automata

Data Automata



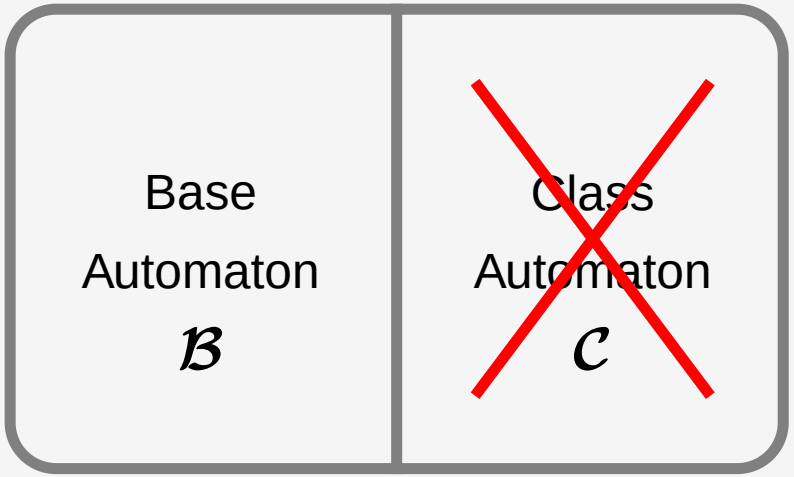
Definition

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Weak Data Automata

Data Automata

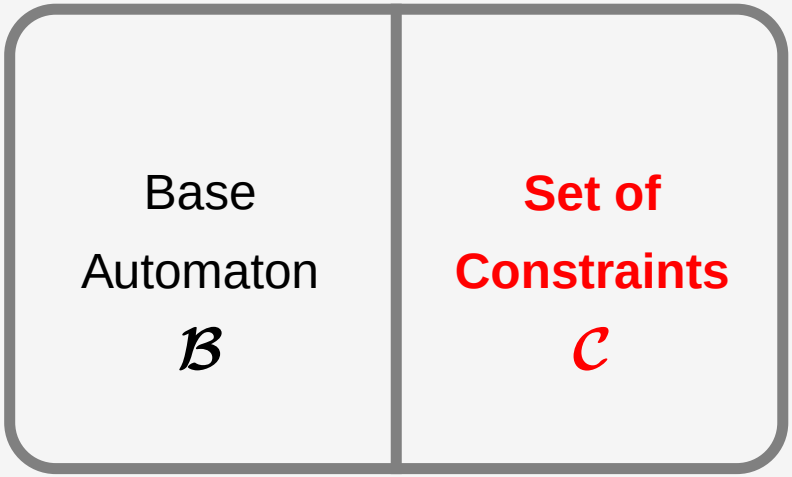


Definition

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

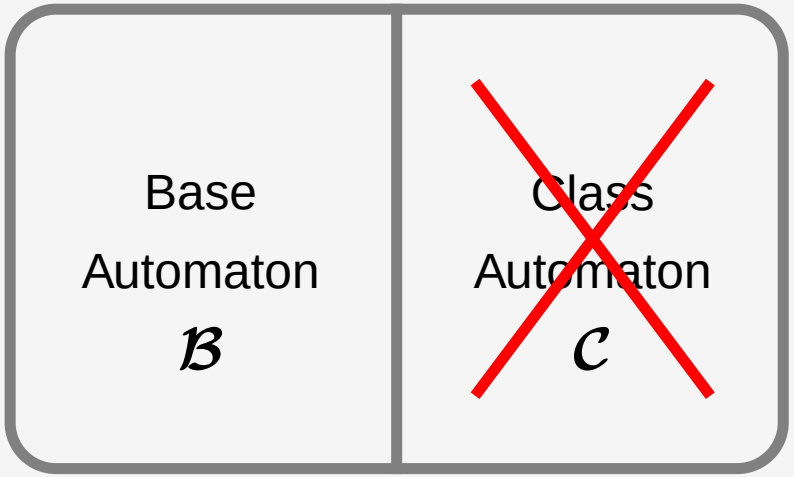
- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Weak Data Automata



Weak Data Automata

Data Automata

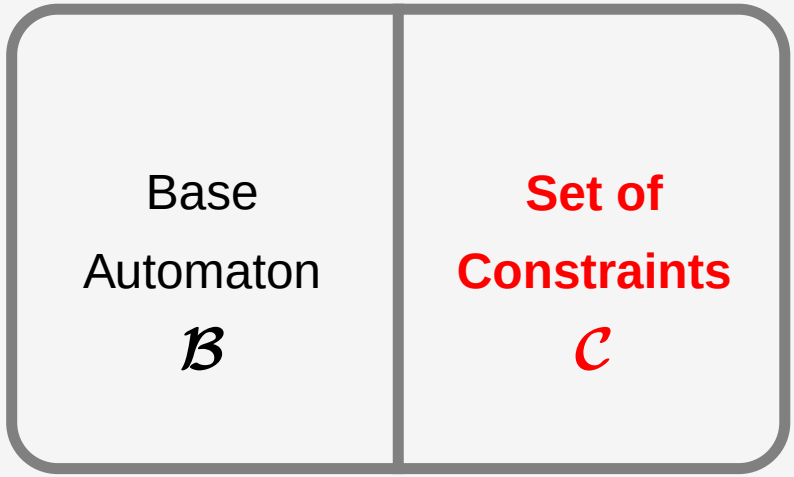


Definition

A data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a class automaton $\mathcal{C}$: a usual string automaton over Γ

Weak Data Automata



Definition

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Constraints on Data Words

Constraints on Data Words

- *key constraints*
  key(σ): all σ-positions have different data values

Example

$w =$

c	c	b	c	a	a	c	b
1	1	4	2	2	3	3	1

Constraints on Data Words

Constraints on Data Words

- *key constraints*
  key(σ): all σ-positions have different data values

Example

$w =$

c	c	b	c	a	a	c	b
1	1	4	2	2	3	3	1

- $w \models \text{key}(a)$

Constraints on Data Words

Constraints on Data Words

- *key constraints*
  key(σ): all σ-positions have different data values
- *inclusion constraints*
  $V(\sigma_1) \subseteq V(\sigma_2)$: all data values on σ_1-positions occur on σ_2-positions

Example

$w =$

c	c	b	c	a	a	c	b
1	1	4	2	2	3	3	1

- $w \models$ key(a)

Constraints on Data Words

Constraints on Data Words

- *key constraints*
  key(σ): all σ-positions have different data values
- *inclusion constraints*
  $V(\sigma_1) \subseteq V(\sigma_2)$: all data values on σ_1-positions occur on σ_2-positions

Example

$w =$

c	c	b	c	a	a	c	b
1	1	4	2	2	3	3	1

- $w \models$ key(a)
- $w \models V(a) \subseteq V(c)$

Constraints on Data Words

Constraints on Data Words

- *key constraints*
  key(σ): all σ-positions have different data values
- *inclusion constraints*
  $V(\sigma_1) \subseteq V(\sigma_2)$: all data values on σ_1-positions occur on σ_2-positions
- *denial constraints*
  $V(\sigma_1) \cap V(\sigma_2) = \emptyset$: σ_1-positions and σ_2-positions do not share any data value

Example

$w =$

c	c	b	c	a	a	c	b
1	1	4	2	2	3	3	1

- $w \models \text{key}(a)$
- $w \models V(a) \subseteq V(c)$

Constraints on Data Words

Constraints on Data Words

- *key constraints*
  key(σ): all σ-positions have different data values
- *inclusion constraints*
  $V(\sigma_1) \subseteq V(\sigma_2)$: all data values on σ_1-positions occur on σ_2-positions
- *denial constraints*
  $V(\sigma_1) \cap V(\sigma_2) = \emptyset$: σ_1-positions and σ_2-positions do not share any data value

Example

$w =$

c	c	b	c	a	a	c	b
1	1	4	2	2	3	3	1

- $w \models \text{key}(a)$
- $w \models V(a) \subseteq V(c)$
- $w \models V(a) \cap V(b) = \emptyset$

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values.”

$(\mathcal{B}, \mathcal{C})$

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

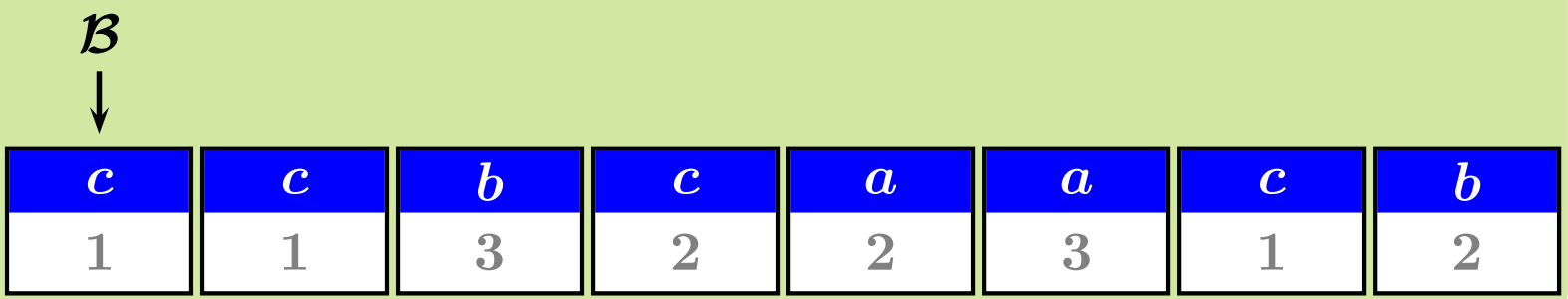
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values."

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

$-$	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values."

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

$-$	$-$	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values."

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

$-$	$-$	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

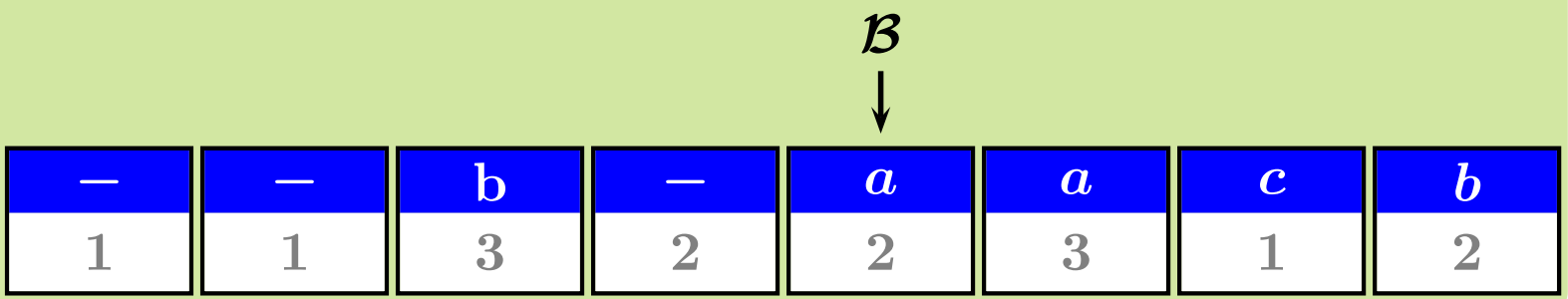
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values."

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

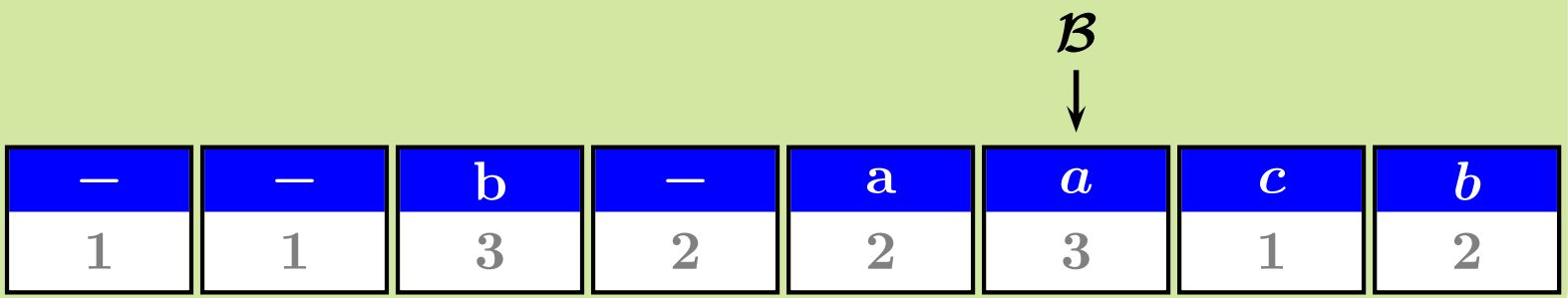
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values.”

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

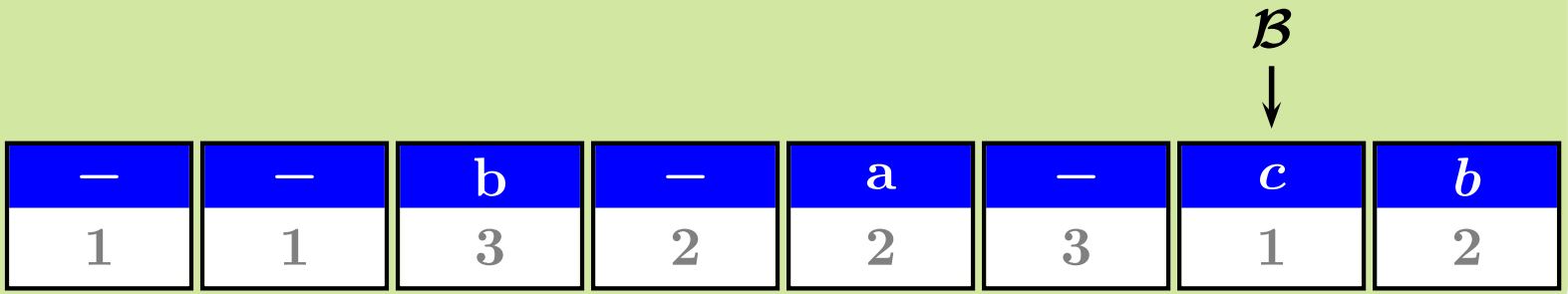
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values."

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values."

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$

–	–	b	–	a	–	–	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values.”

$(\mathcal{B}, \mathcal{C})$

–	–	b	–	a	–	–	–
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„There is an a and a b with different data values.”

$(\mathcal{B}, \mathcal{C})$

–	–	b	–	a	–	–	–
1	1	3	2	2	3	1	2

- $\mathcal{C} = \{V(a) \cap V(b) = \emptyset\}$

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value."

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

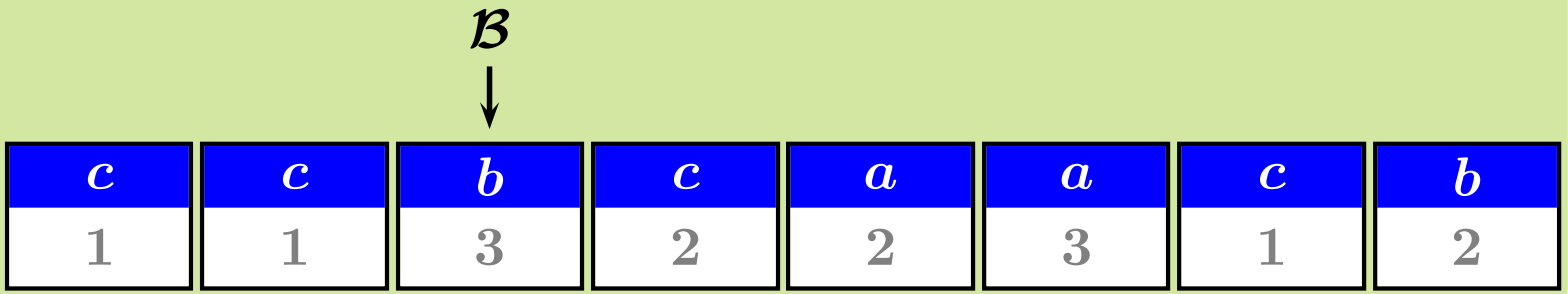
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value."

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

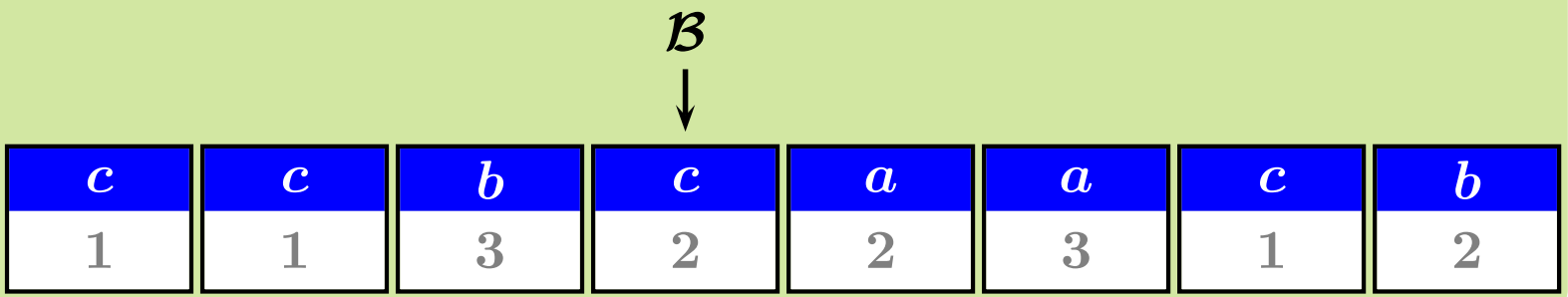
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

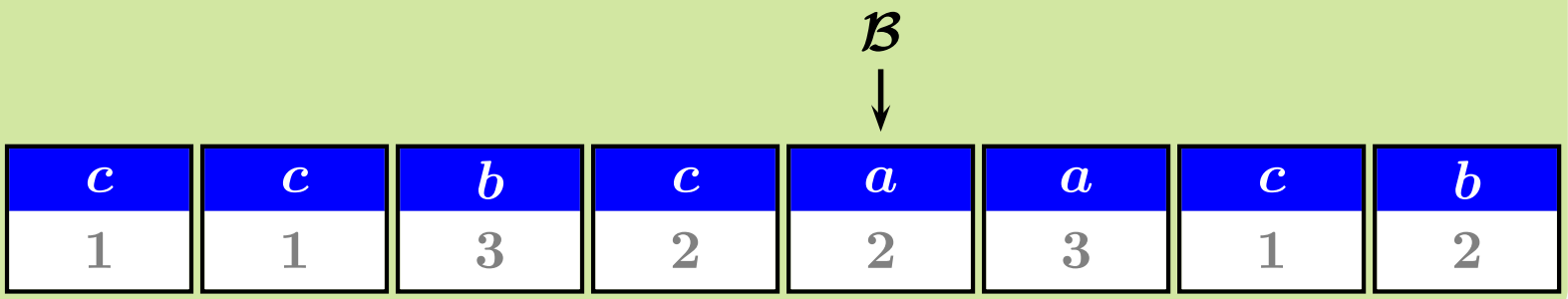
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

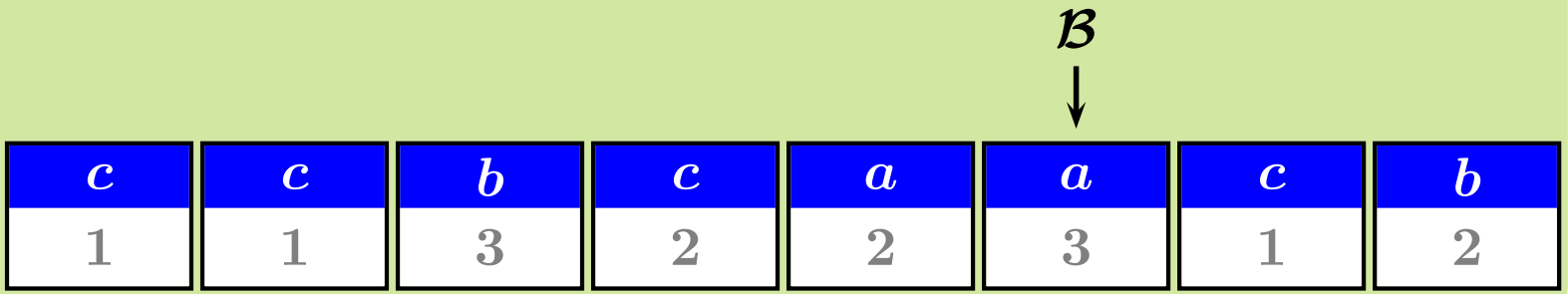
A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$



Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

$\mathcal{B}$ ↓

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value."

$(\mathcal{B}, \mathcal{C})$

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

$(\mathcal{B}, \mathcal{C})$

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

- $\mathcal{C} = \{V(a) \subseteq V(b)\}$

Weak Data Automata

Weak Data Automata ($\mathbf{WDA}$)

A weak data automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base automaton $\mathcal{B}$: a nondeterministic letter-to-letter string transducer with input alphabet Σ and output alphabet Γ
- a set $\mathcal{C}$ of constraints over Γ

Example

„For every a there exists some b with the same data value.”

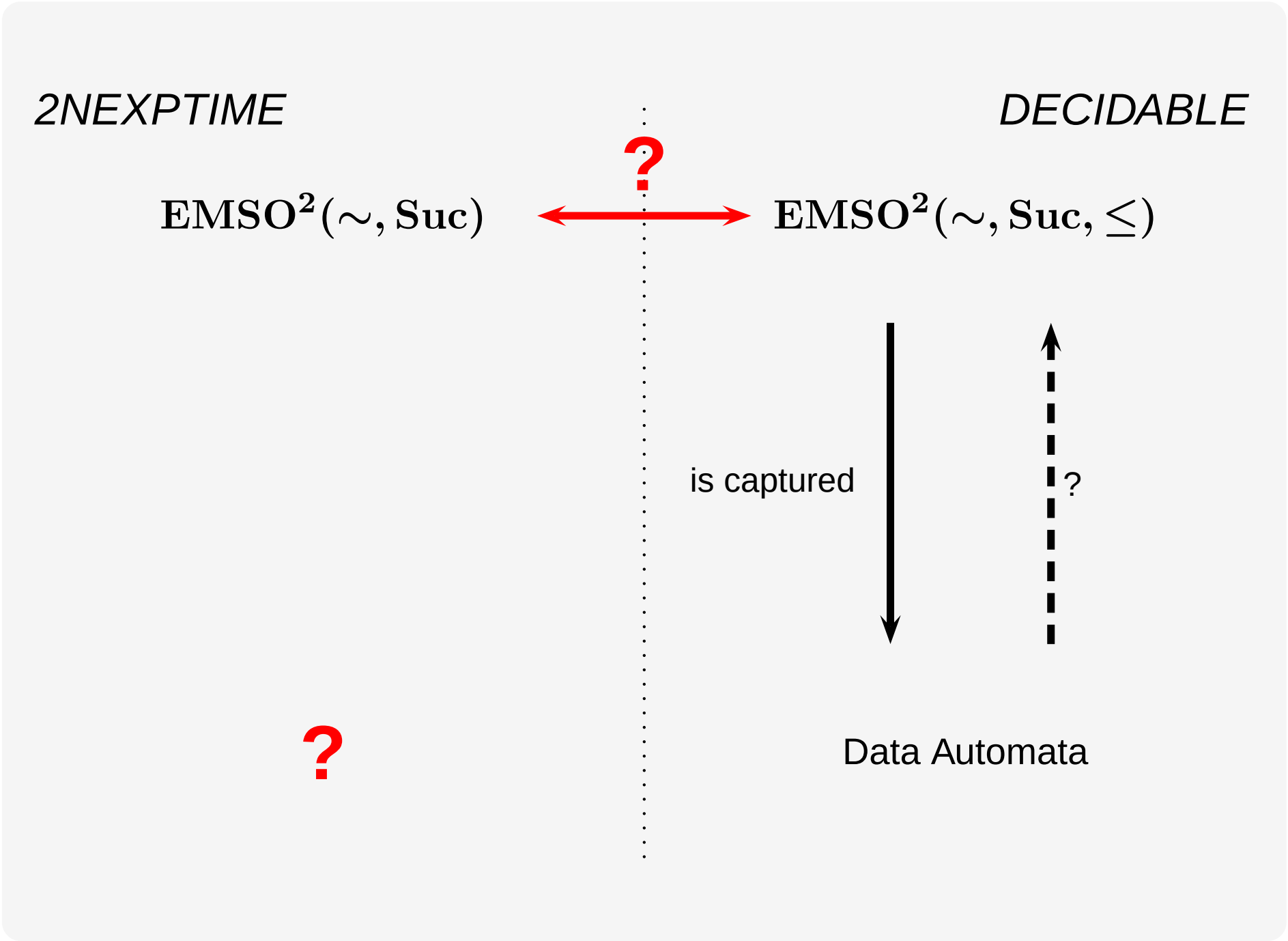
$(\mathcal{B}, \mathcal{C})$

c	c	b	c	a	a	c	b
1	1	3	2	2	3	1	2

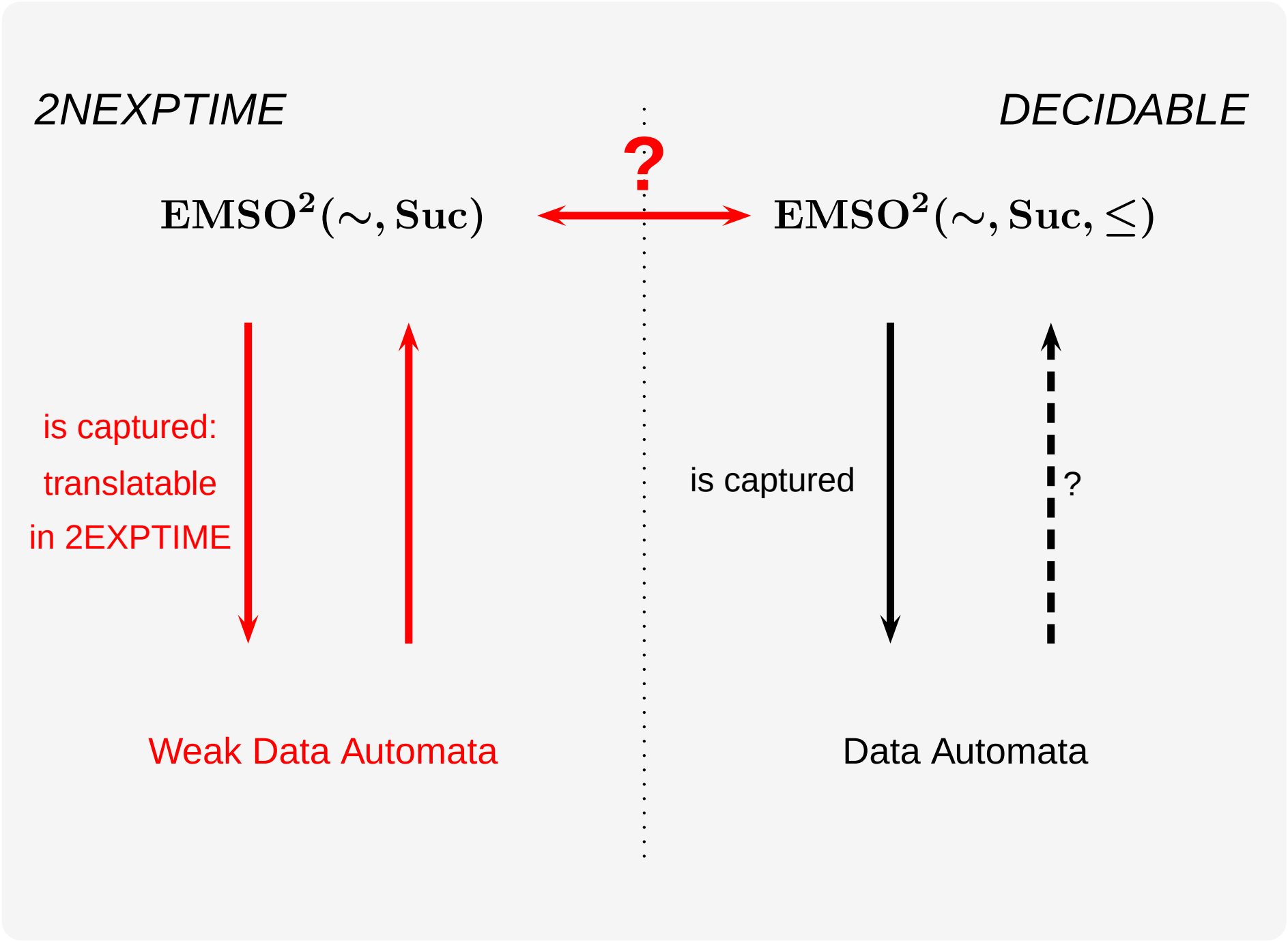
- $\mathcal{C} = \{V(a) \subseteq V(b)\}$

- Main difference between class automata and constraints: *constraints cannot access the order of positions*

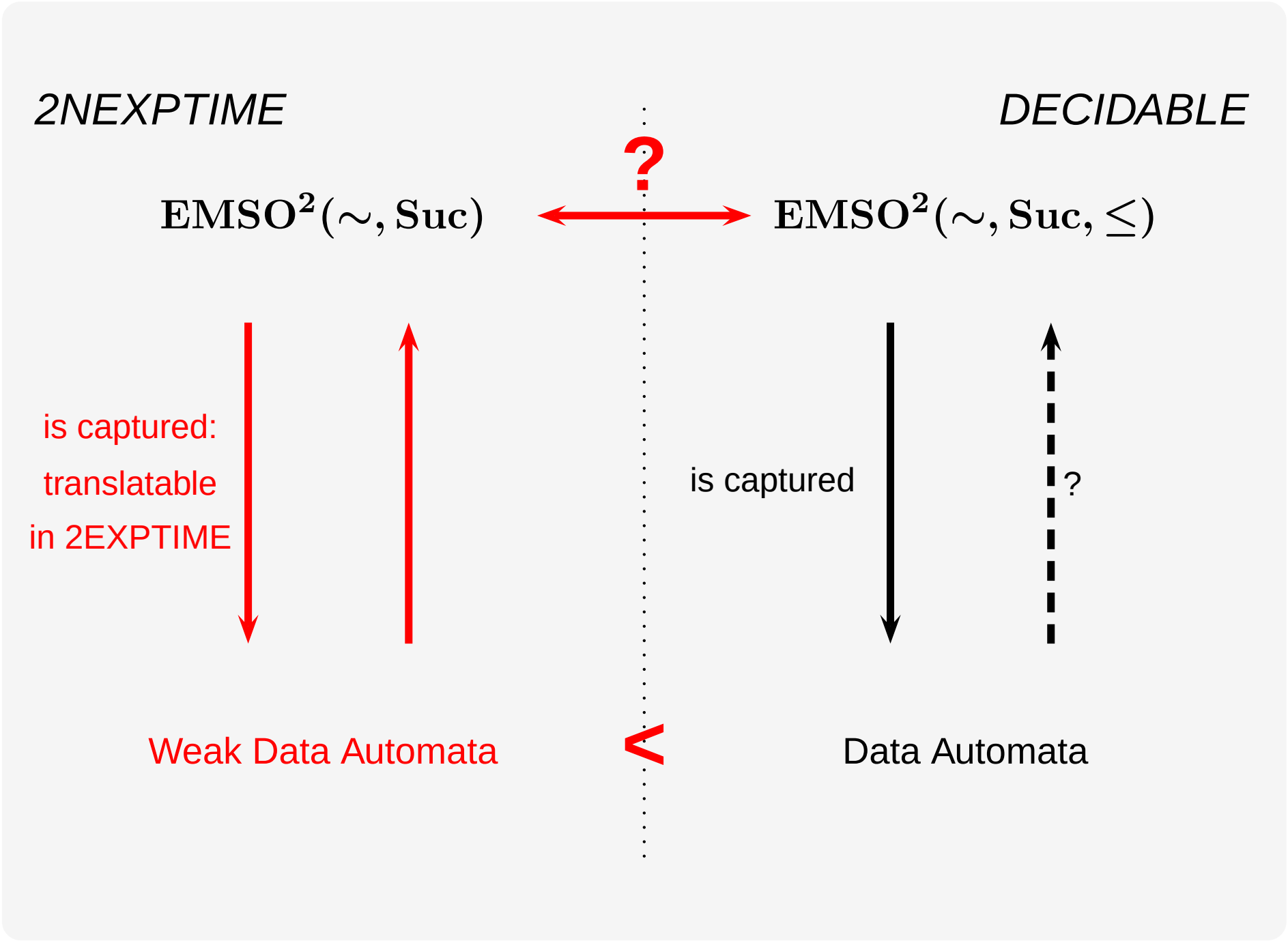
Our Results


2NEXPTIME
DECIDABLE
?
$\mathbf{EMSO^2}(\sim, \mathbf{Suc})$
$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$
is captured
?
?
Data Automata

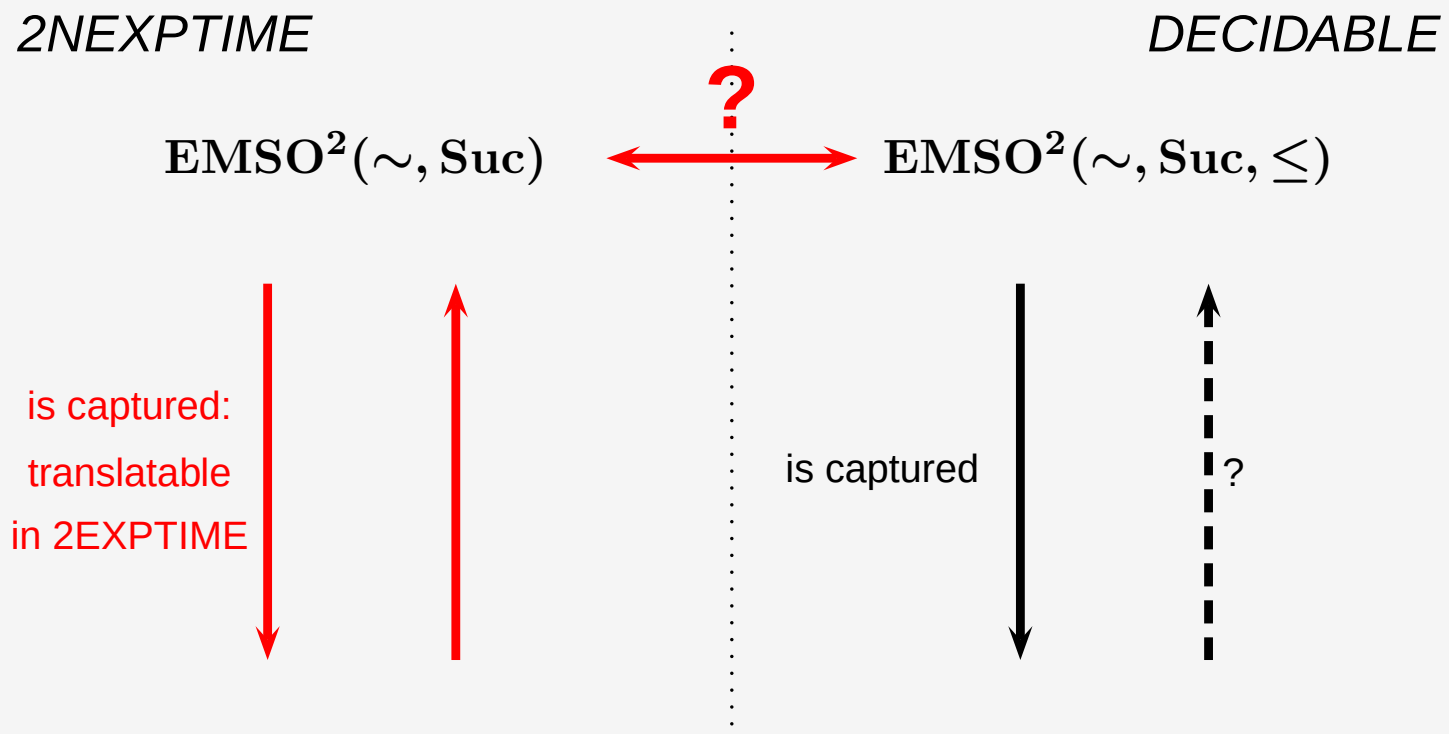
Our Results


2NEXPTIME
DECIDABLE
?
$\mathbf{EMSO^2(\sim, Suc)}$
$\mathbf{EMSO^2(\sim, Suc, \leq)}$
is captured:
translatable
in 2EXPTIME
is captured
?
Weak Data Automata
Data Automata

Our Results


2NEXPTIME
DECIDABLE
?
$\mathbf{EMSO^2}(\sim, \mathbf{Suc})$
$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$
is captured:
translatable
in 2EXPTIME
is captured
?
Weak Data Automata
<
Data Automata

Our Results

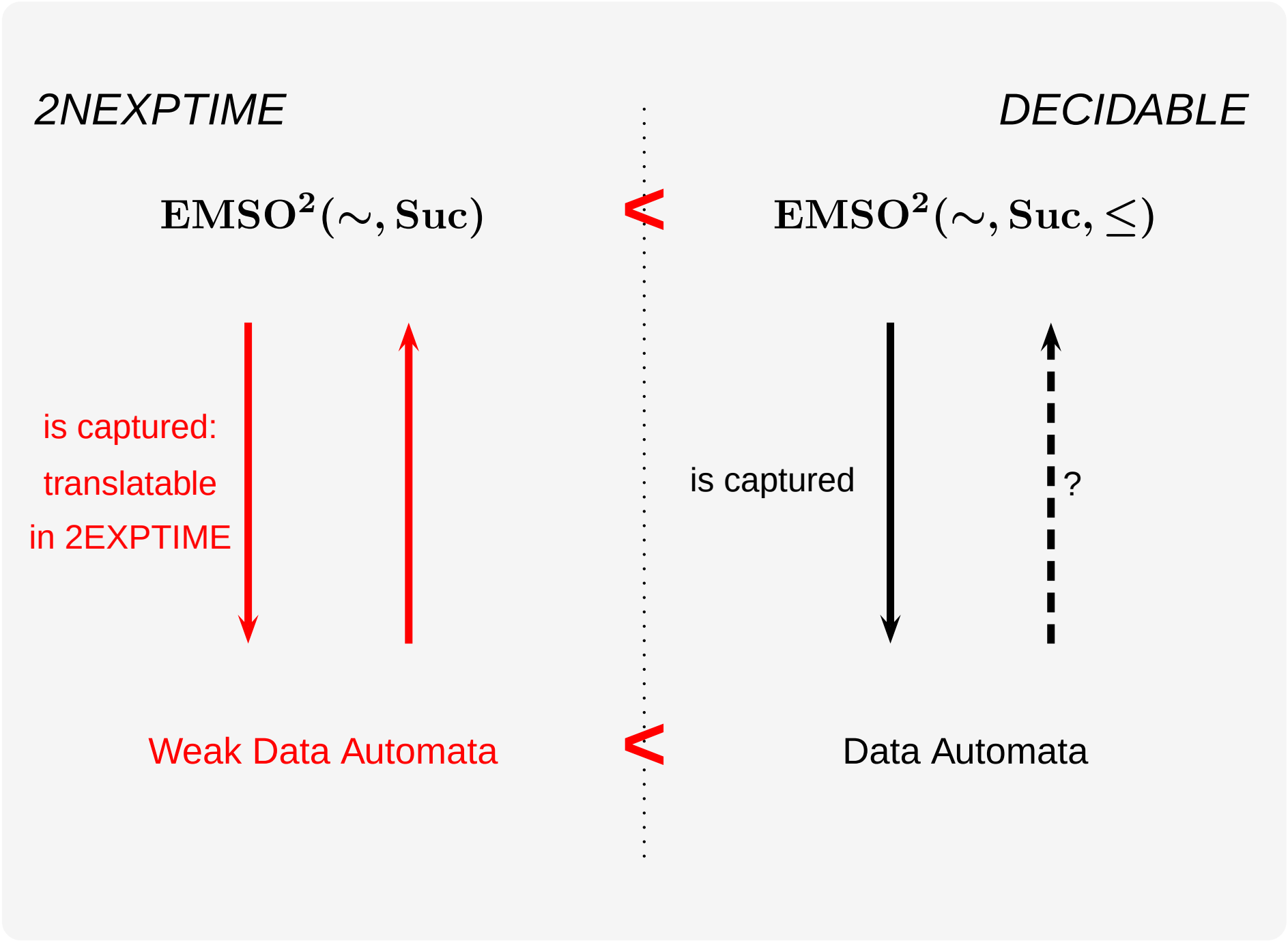


Theorem

$$\mathbf{WDA} < \mathbf{DA}$$

- inclusion: the class automaton can check all constraints
  - e.g. key(a): class automaton checks that all class strings contain at most one a
- separating property: „Every a is followed by some b with the same data value.”

Our Results


2NEXPTIME
DECIDABLE
$\mathbf{EMSO^2}(\sim, \mathbf{Suc})$
<
$\mathbf{EMSO^2}(\sim, \mathbf{Suc}, \leq)$
is captured:
translatable
in 2EXPTIME
is captured
?
Weak Data Automata
<
Data Automata

Weak ω-Data Automata

Data ω-words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times D)^\omega$ is a data ω-word over Σ

Weak ω-Data Automata

Data ω-words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times D)^\omega$ is a data ω-word over Σ

Data ω-Automata [Bojańczyk et al. 06]

A data ω-automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C}_{\text{fin}}, \mathcal{C}_{\text{inf}})$ over an alphabet Σ consists of

- a base Büchi automaton $\mathcal{B}$
- a class automaton $\mathcal{C}_{\text{fin}}$
- a class Büchi automaton $\mathcal{C}_{\text{inf}}$

Weak ω-Data Automata

Data ω-words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times D)^\omega$ is a data ω-word over Σ

Data ω-Automata [Bojańczyk et al. 06]

A data ω-automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C}_{\text{fin}}, \mathcal{C}_{\text{inf}})$ over an alphabet Σ consists of

- a base Büchi automaton $\mathcal{B}$
- a class automaton $\mathcal{C}_{\text{fin}}$
- a class Büchi automaton $\mathcal{C}_{\text{inf}}$

Weak Data ω-Automata

A weak data ω-automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

- a base Büchi automaton $\mathcal{B}$
- a set $\mathcal{C}$ of constraints

Weak ω-Data Automata

Data ω-words

- Let
  - Σ be a finite alphabet
  - $\mathcal{D}$ be an infinite set of data values
- $w \in (\Sigma \times D)^\omega$ is a data ω-word over Σ

Data ω-Automata [Bojańczyk et al. 06]

A data ω-automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C}_{\text{fin}}, \mathcal{C}_{\text{inf}})$ over an alphabet Σ consists of

- a base Büchi automaton $\mathcal{B}$
- a class automaton $\mathcal{C}_{\text{fin}}$
- a class Büchi automaton $\mathcal{C}_{\text{inf}}$

Weak Data ω-Automata

A weak data ω-automaton $\mathcal{D} = (\mathcal{B}, \mathcal{C})$ over an alphabet Σ consists of

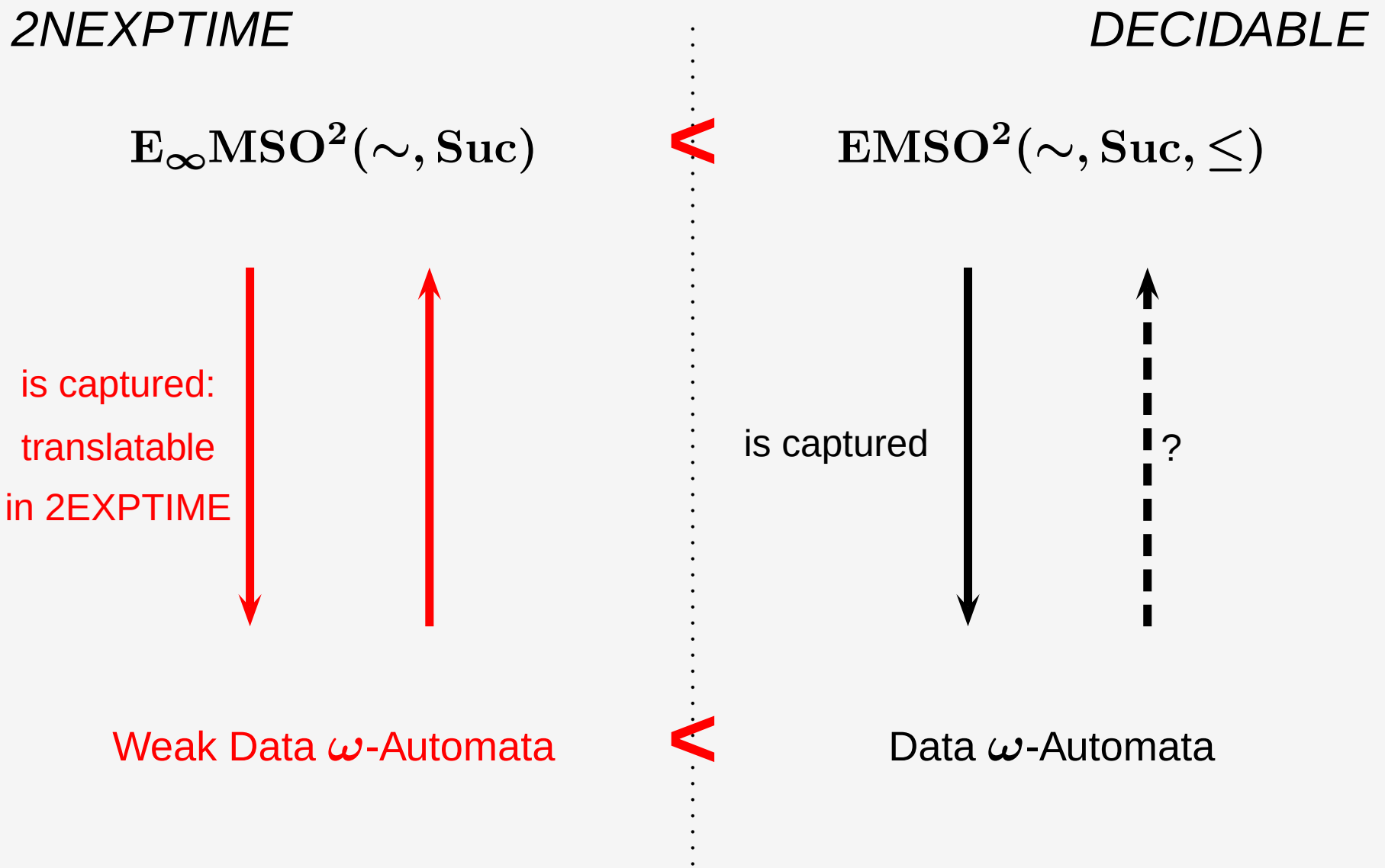
- a base Büchi automaton $\mathcal{B}$
- a set $\mathcal{C}$ of constraints

$\mathrm{E}_\infty\mathrm{MSO}^2(\sim, \mathrm{Suc})$

Extension of $\mathrm{EMSO}^2(\sim, \mathrm{Suc})$:

- quantified sets can be restricted to be bound to infinite sets only

Our Results on Data ω-Words



Conclusion

- $\mathbf{WDA}$: automata model for $\mathbf{EMSO^2}(\sim, \mathbf{Suc})$ (and $\mathbf{E}_\infty\mathbf{MSO^2}(\sim, \mathbf{Suc})$) with reasonable complexity

Conclusion

- $\mathbf{WDA}$: automata model for $\mathbf{EMSO^2}(\sim, \mathbf{Suc})$ (and $\mathbf{E_\infty MSO^2}(\sim, \mathbf{Suc})$) with reasonable complexity
- Open Questions:
  - Are $\mathbf{WDA}$ the best automata model for these logics?

Conclusion

- $\mathbf{WDA}$: automata model for $\mathbf{EMSO^2}(\sim, \mathbf{Suc})$ (and $\mathbf{E}_\infty\mathbf{MSO^2}(\sim, \mathbf{Suc})$) with reasonable complexity
- Open Questions:
  - Are $\mathbf{WDA}$ the best automata model for these logics?
  - What is the exact complexity of the emptiness problem for weak data automata?
    - The 2NEXPTIME upper bound yields only a 3NEXPTIME upper bound for the satisfiability of $\mathbf{EMSO^2}(\sim, \mathbf{Suc})$
    - which is in 2NEXPTIME [Niewerth et al. 09]