Automata based verification over linearly ordered data domains

to be presented at STACS `11

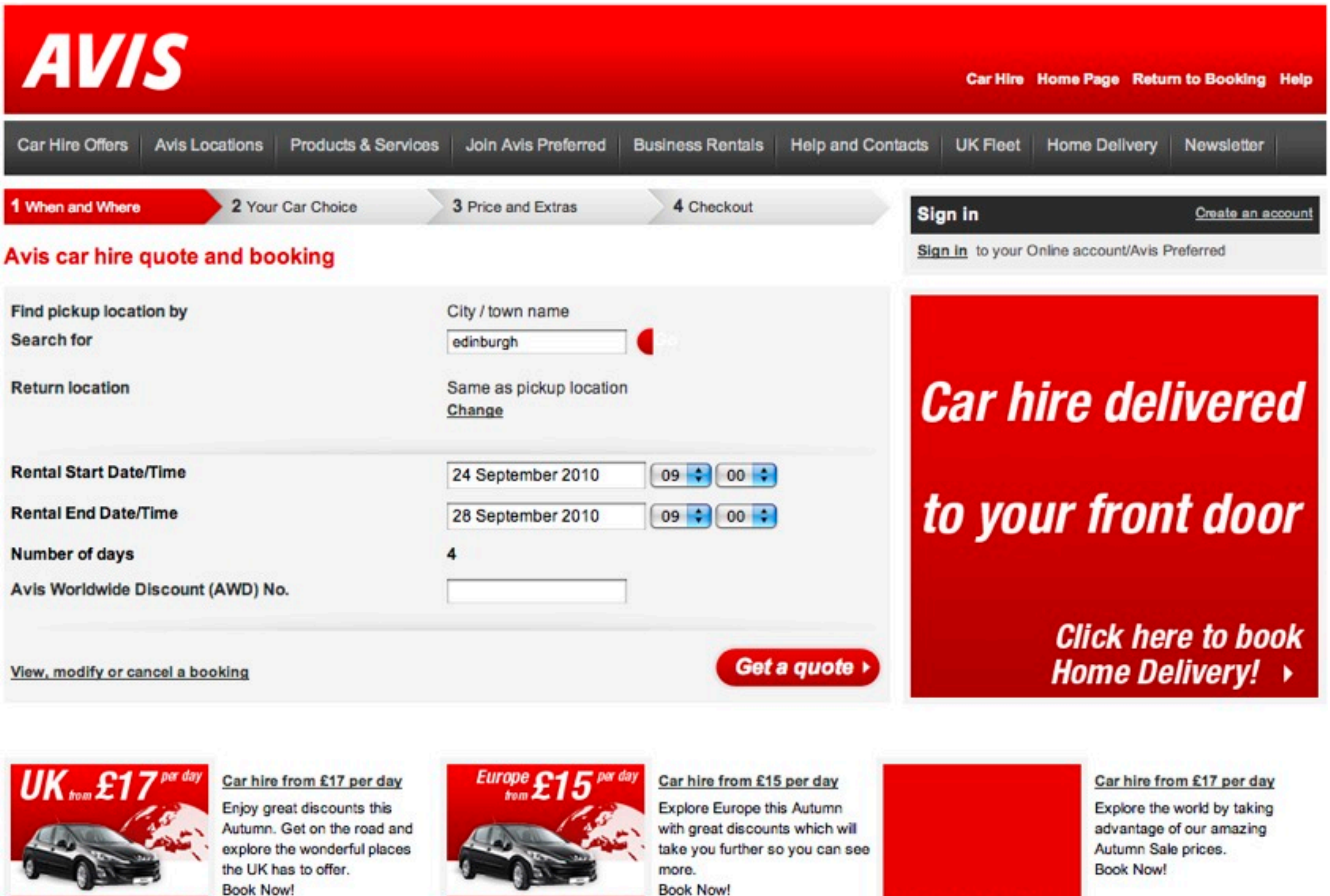
Luc Segoufin
INRIA and ENS Cachan

Szymon Toruńczyk
University of Warsaw

Motivation

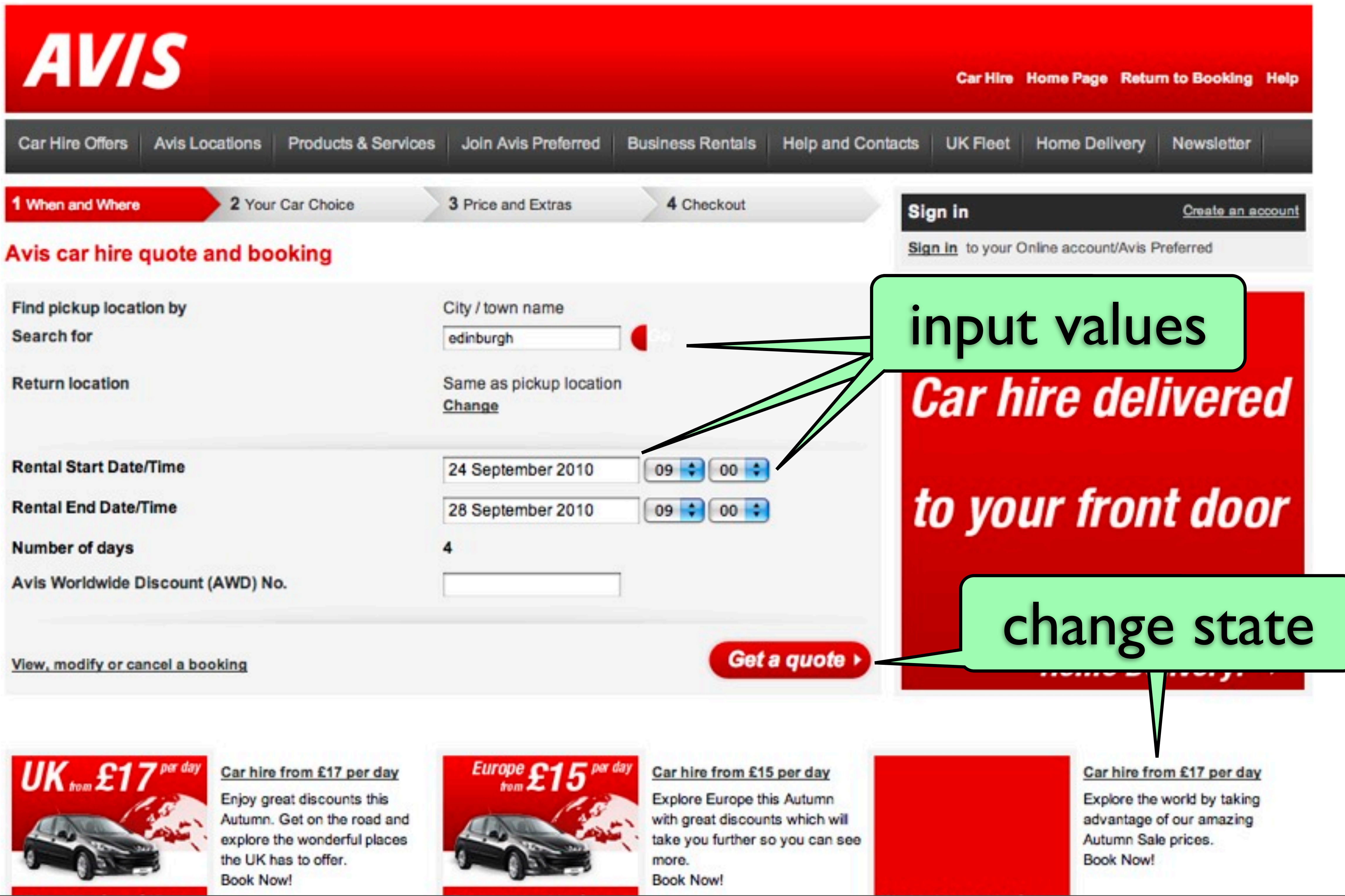
Motivation

Given a system which refers to a database and data values, verify properties of the system



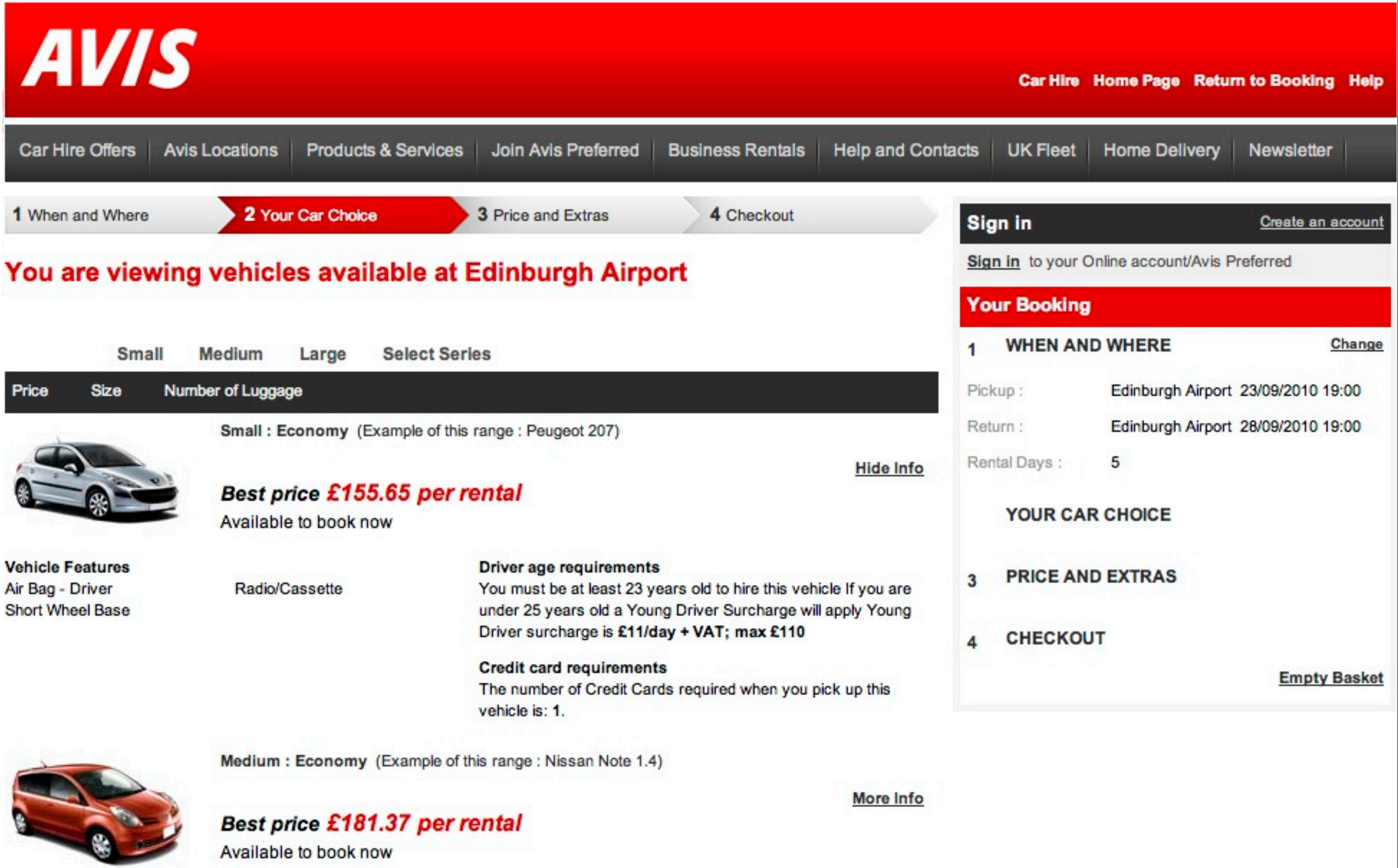
Motivation

Given a system which refers to a database and data values, verify properties of the system



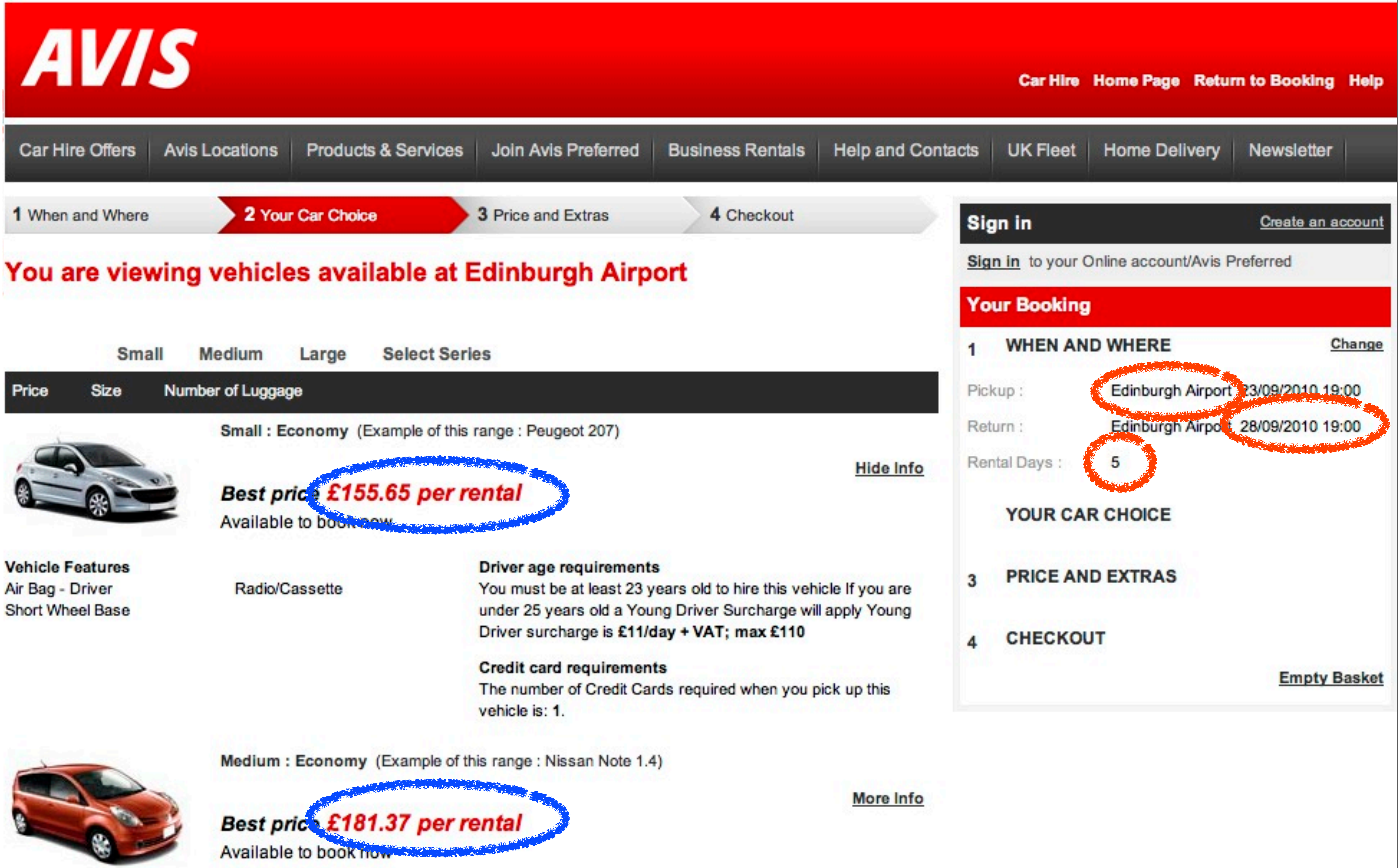
Motivation

Given a system which refers to a database and data values, verify properties of the system



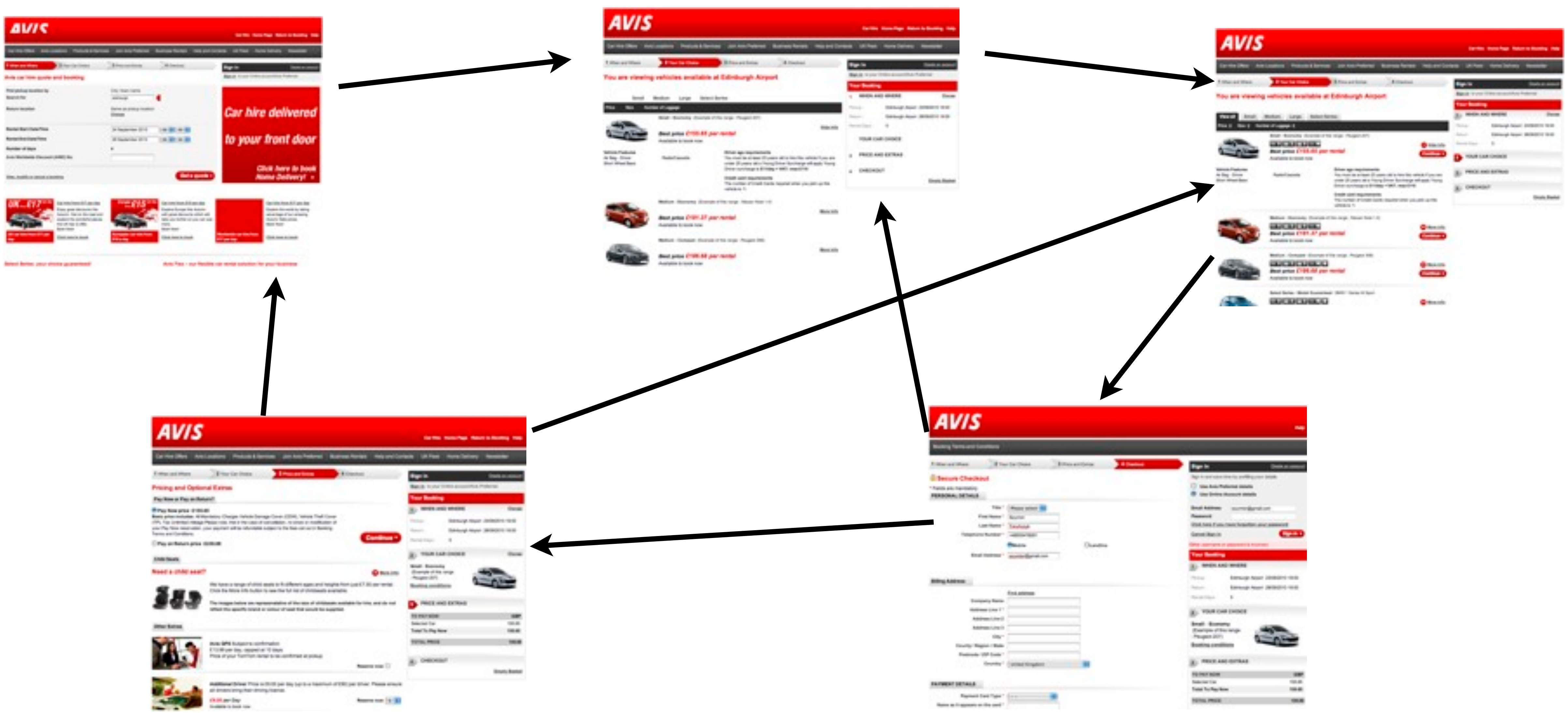
Motivation

Given a system which refers to a database and data values, verify properties of the system



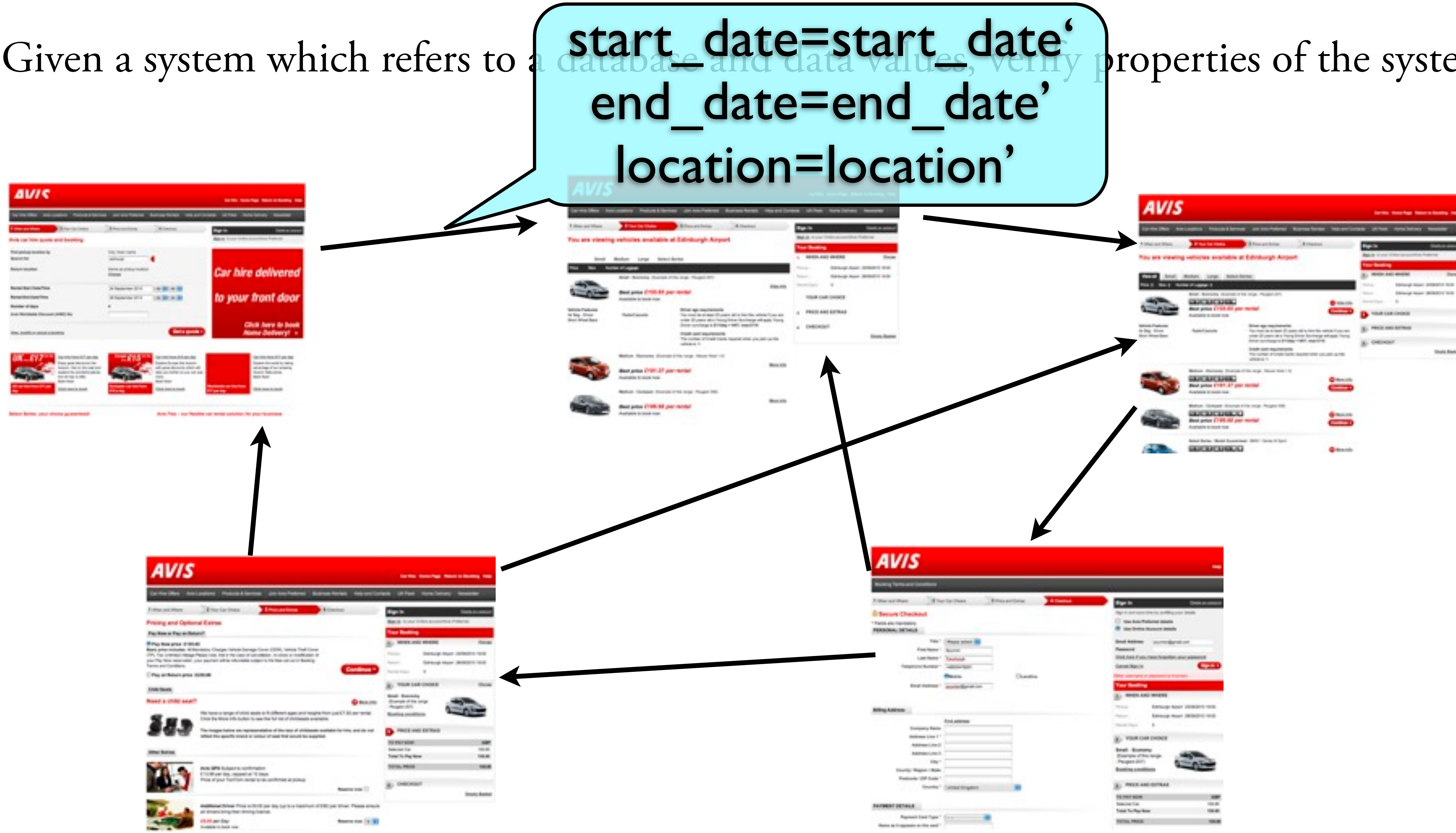
Motivation

Given a system which refers to a database and data values, verify properties of the system



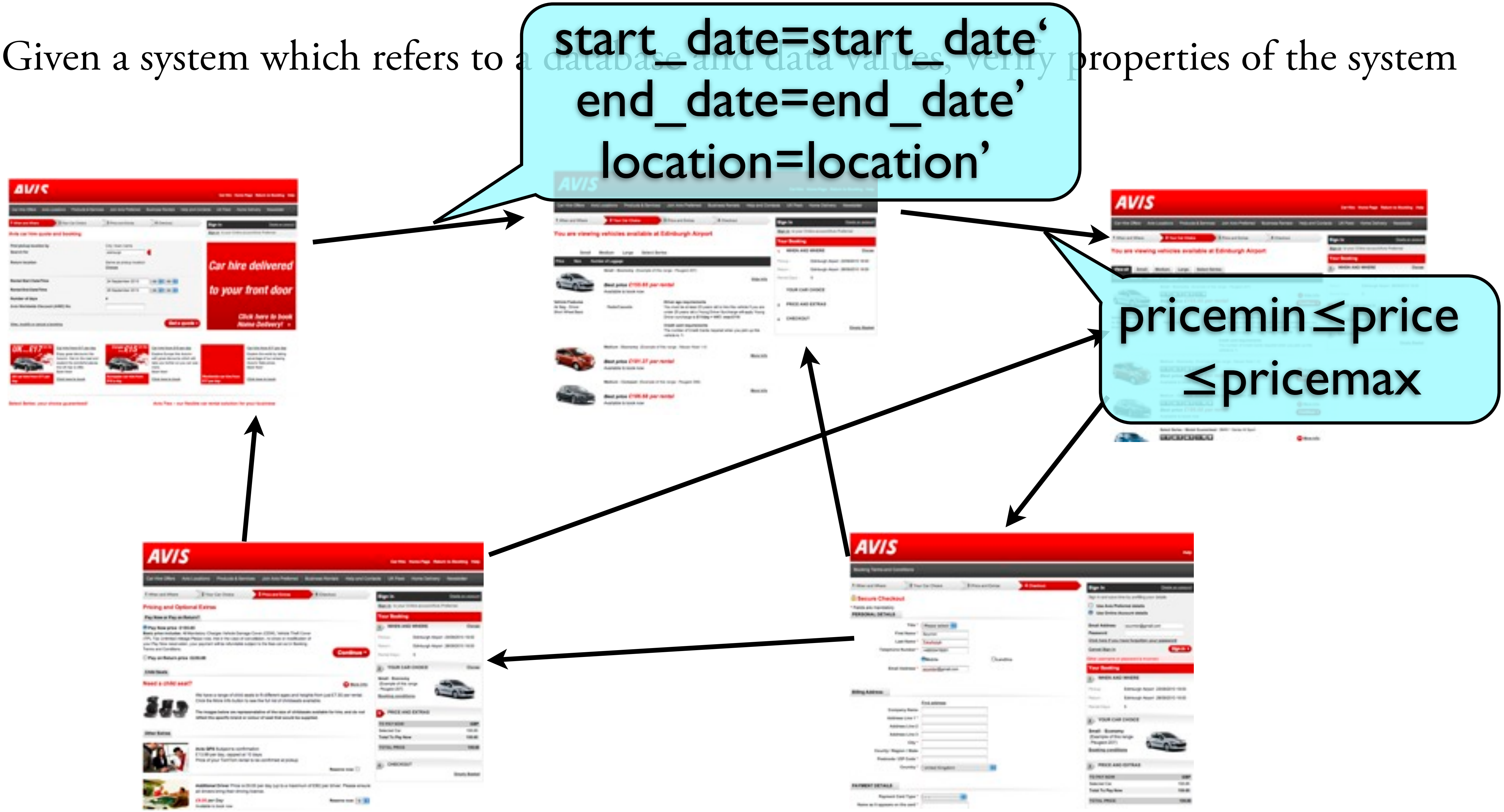
Motivation

Given a system which refers to a database and data values, verify properties of the system



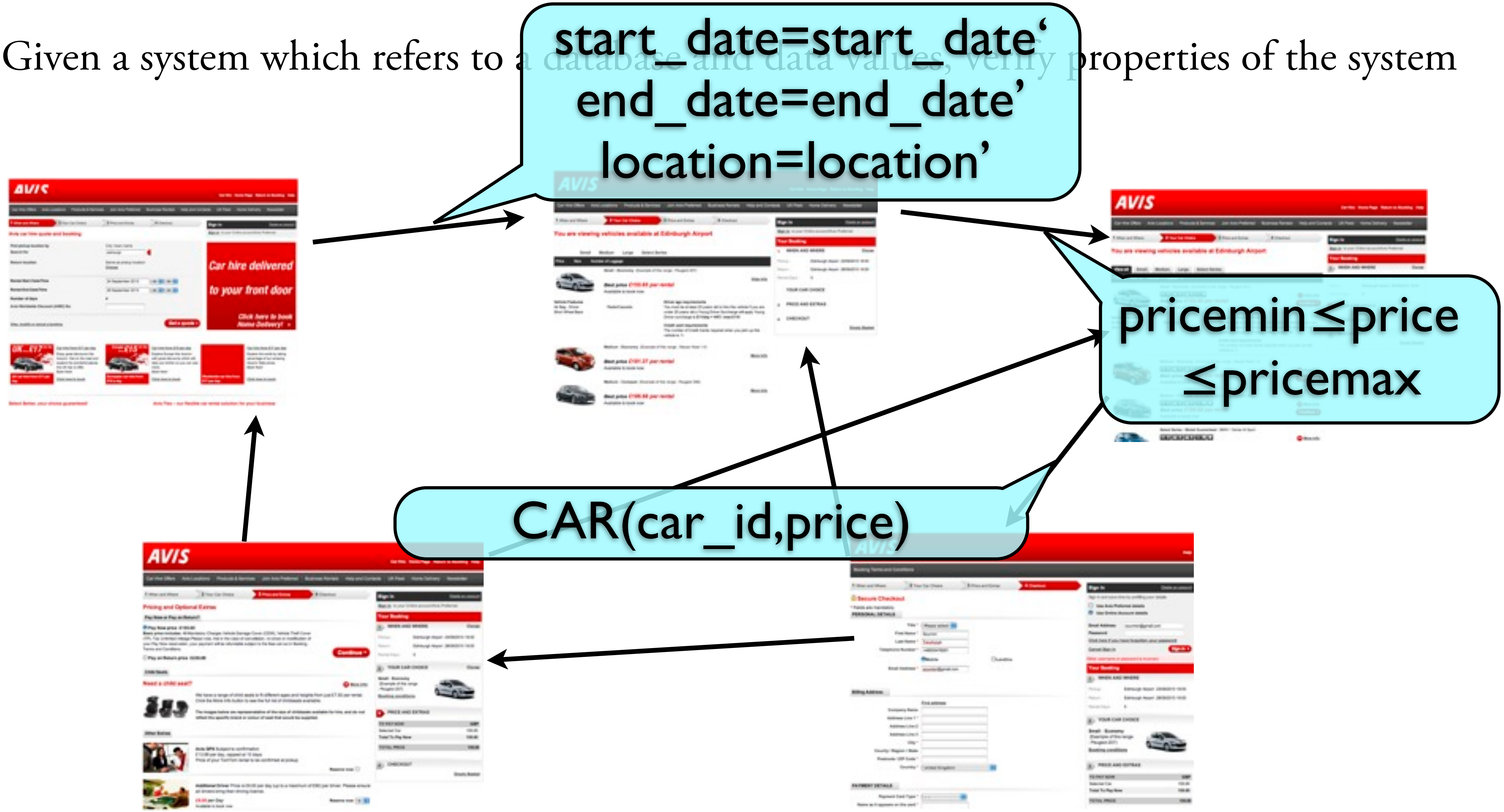
Motivation

Given a system which refers to a database and data values, verify properties of the system



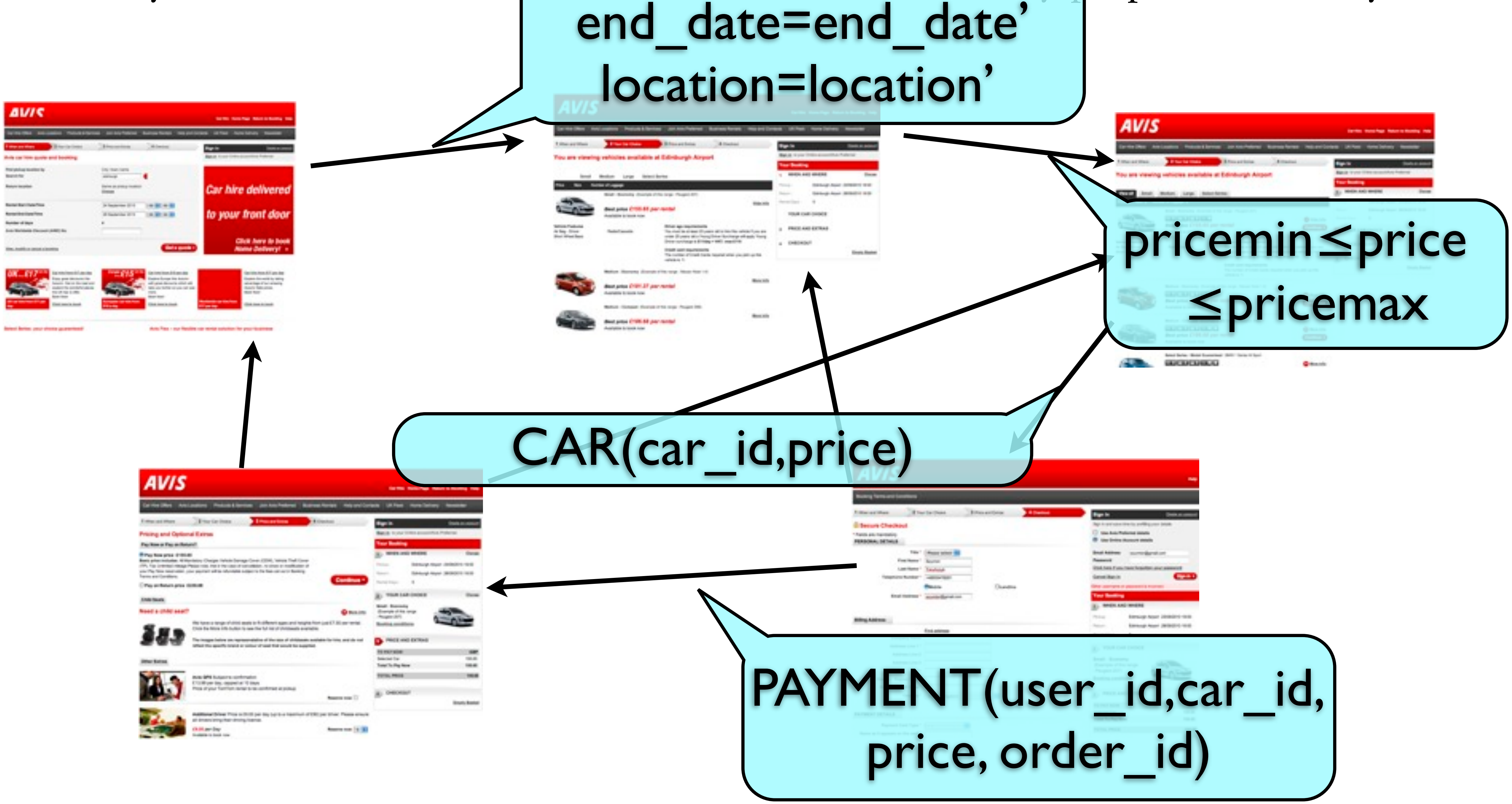
Motivation

Given a system which refers to a database and data values, verify properties of the system



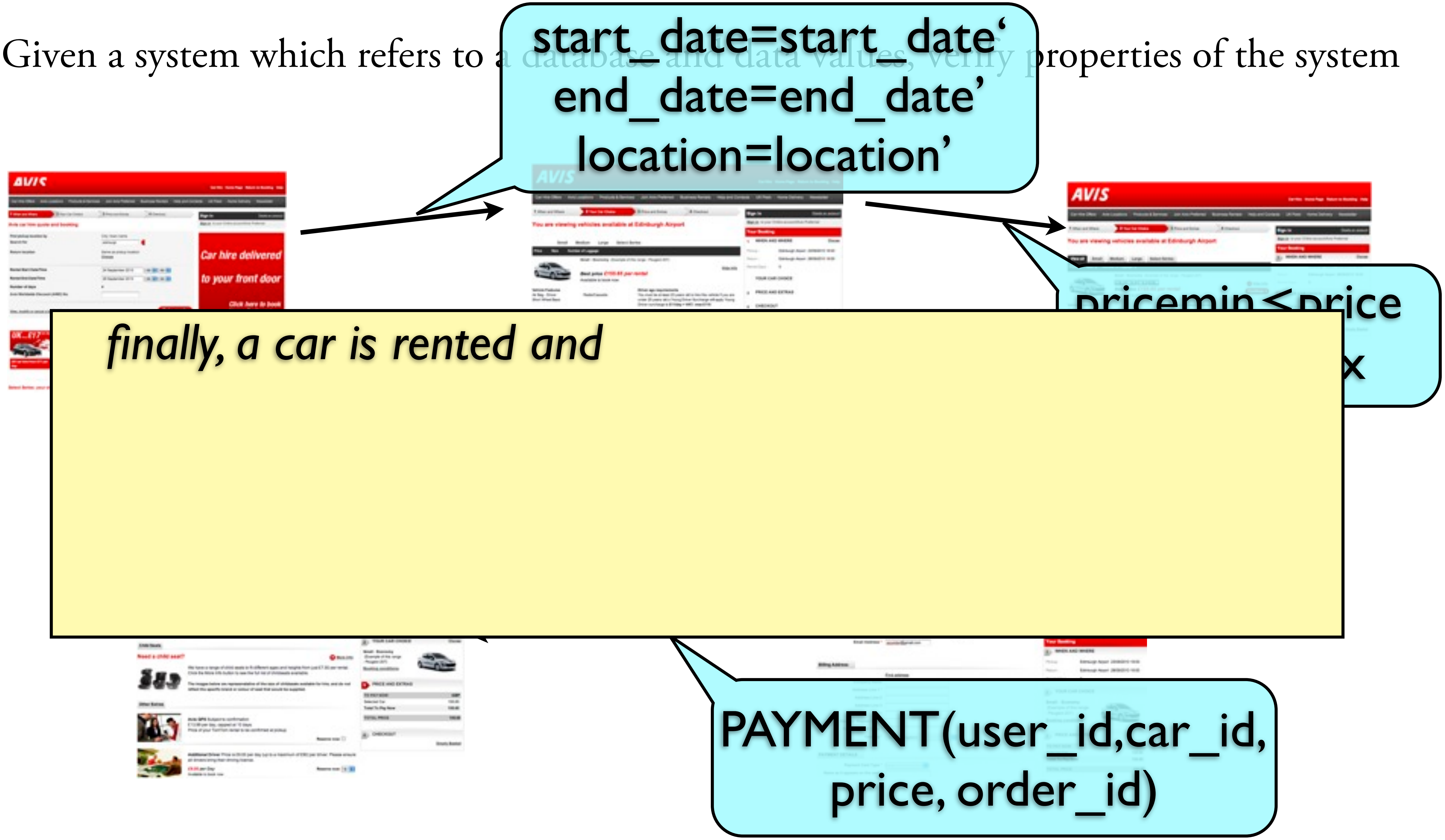
Motivation

Given a system which refers to a database and data values, verify properties of the system



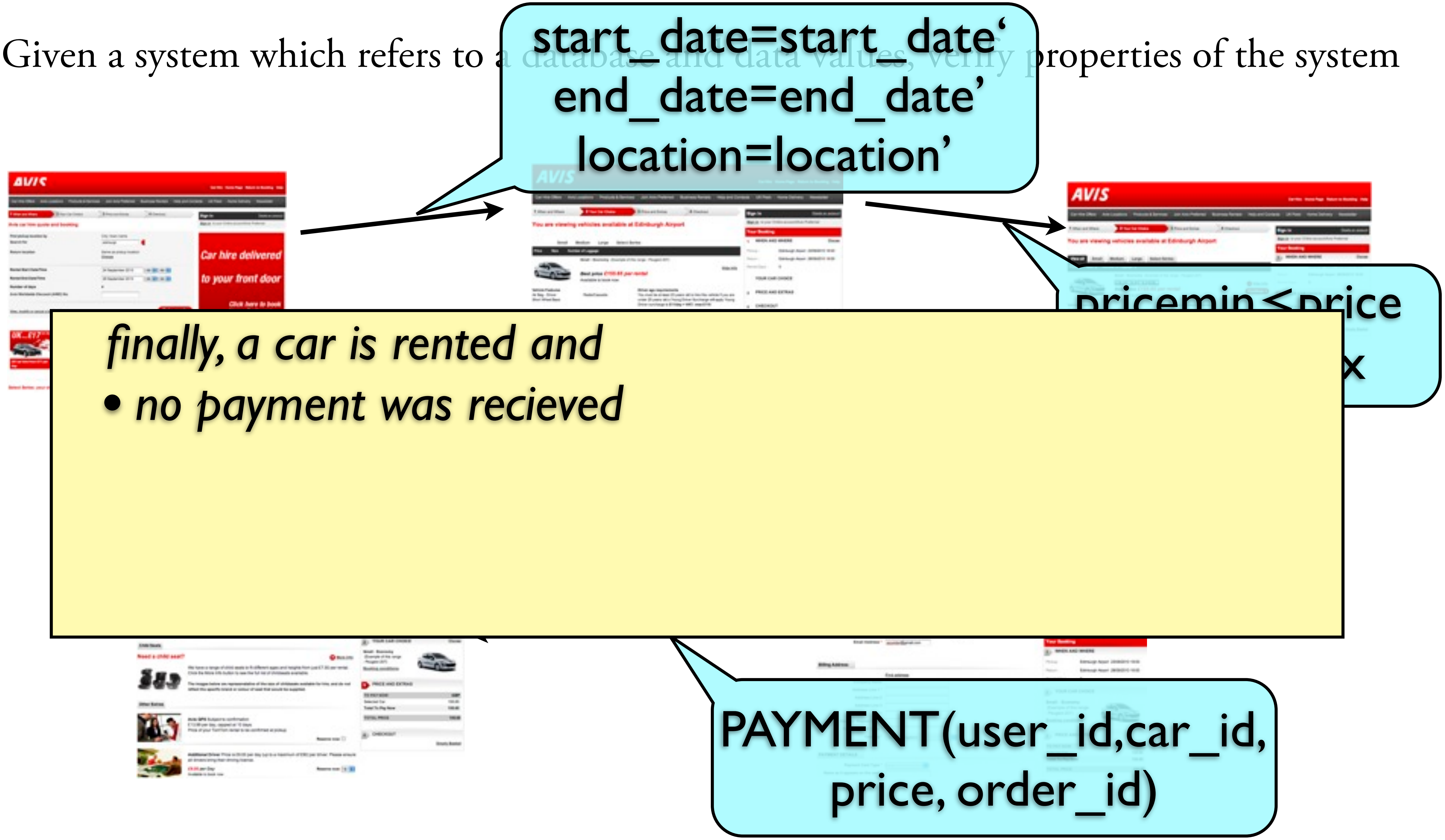
Motivation

Given a system which refers to a database and data values, verify properties of the system



Motivation

Given a system which refers to a database and data values, verify properties of the system

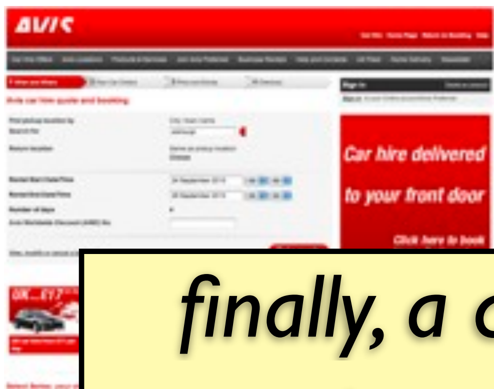


finally, a car is rented and

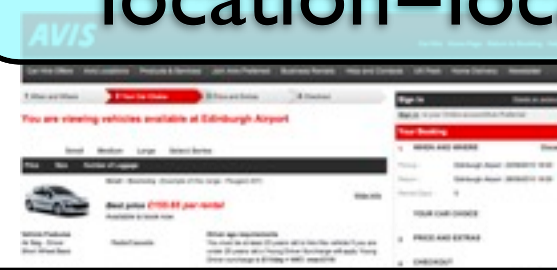
- *no payment was recieved*

Motivation

Given a system which refers to a database and data values, verify properties of the system



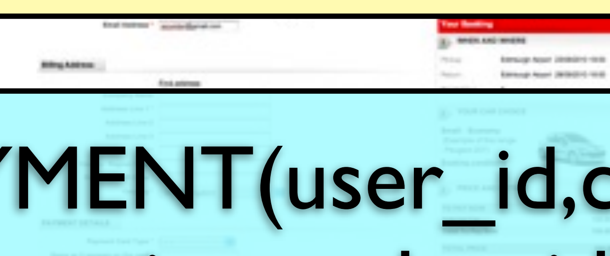
start_date=start_date'
end_date=end_date'
location=location'



pricemin≤price
x

finally, a car is rented and

- *no payment was recieved*
- *a payment was recieved but* payed=0

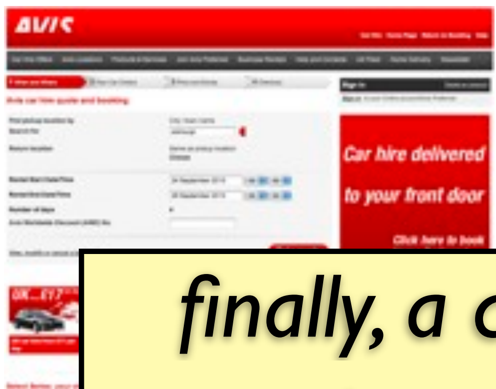


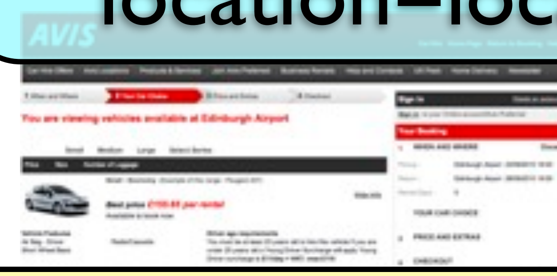
PAYMENT(user_id,car_id,
price, order_id)

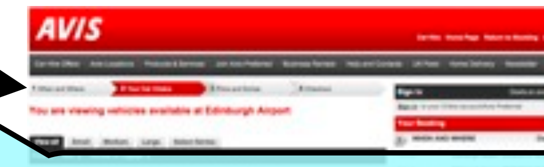
Motivation

Given a system which refers to a database and data values, verify properties of the system

start_date=start_date'
end_date=end_date'
location=location'



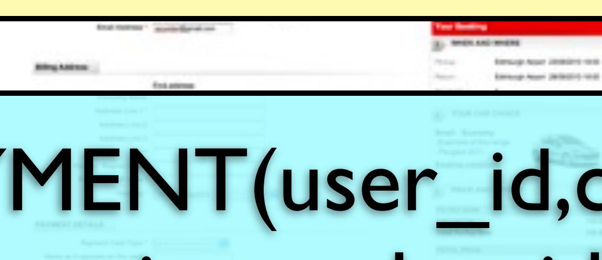




pricemin≤price
x

finally, a car is rented and

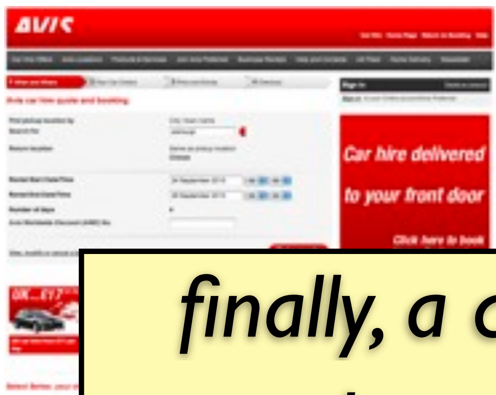
- *no payment was recieved*
- *a payment was recieved but* payed=0
- *there was a payment, but* payed<100 & end_date>2050

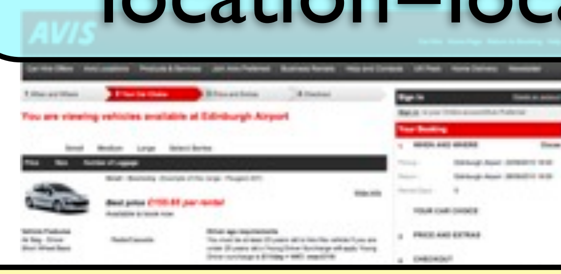


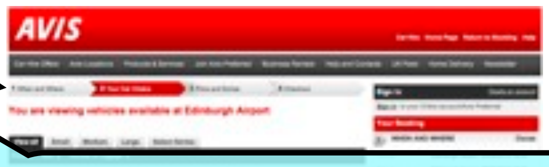
PAYMENT(user_id,car_id,
price, order_id)

Motivation

Given a system which refers to a database and data values, verify properties of the system





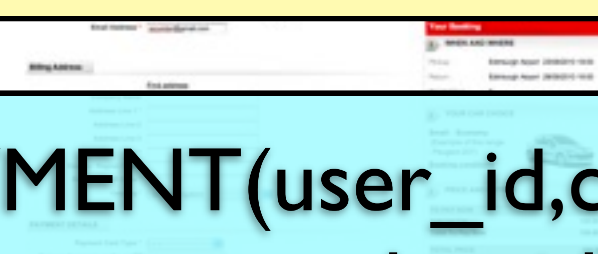


start_date=start_date'
end_date=end_date'
location=location'

pricemin<price

finally, a car is rented and

- *no payment was recieved*
- *a payment was recieved but* payed=0
- *there was a payment, but* payed<100 & end_date>2050
- *there was a payment, but* payed<car_price



PAYMENT(user_id,car_id, price, order_id)

Outline

1. Define automata model
2. Analyze automata over dense orders
3. Analyze automata over discrete orders
4. Add database and infinite runs

Extended automaton

a definition that captures timed automata, or vector addition systems

D – fixed domain

Q – finite set of states

X – finite set of variables

$D^X = \{ v: \; X \to D \}$ – space of variable valuations

$Q \times D^X$ – space of configurations

restricted form

$\delta \subseteq (Q \times D^X) \times (Q \times D^X)$ – set of allowed transitions

$I \subseteq (Q \times D^X)$ – set of initial configurations

$F \subseteq (Q \times D^X)$ – set of final configurations

Examples

- *Vector Addition System*: $D = \mathbb{N}$

$$(q, v) \xrightarrow[v' = v + w]{} (q', v')$$

- *Timed Automata*: $D = \mathbb{R}$

$$(q, v) \xrightarrow[c_2' = 0]{c_1 < 2} (q', v')$$

- (*Lossy*) *channel system*: $D = \{a+b\}^*$

$$(q, v) \xrightarrow[c_1' = c_1 \cdot b]{first_a\,(c_1)} (q', v')$$

Our setting
(without the database)

The domain:

$$\mathscr{D} = \langle\ D, < ,\ P_1,\ P_2,\ P_3 ,\ldots, P_l\ \rangle$$

linearly ordered set

unary predicates
(subsets of D)

Examples

$\langle\ \mathbb{N}, <, 0, 100, P_{even}, P_{prime}\ \rangle$

$\langle\ \mathbb{Q}, <, 0, 100, P_{integer}, P_{<\pi}\ \rangle$

$\langle\ \{a+b\}^*, <_{lex}, P_{(ab)^*}\ \rangle$

Transitions:

$$I, F \subseteq (Q \times D^X)\ , \qquad \delta \subseteq (Q \times D^X) \times (Q \times D^X)$$

are specified by quantifier free formulas over $\mathscr{D}$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states

$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

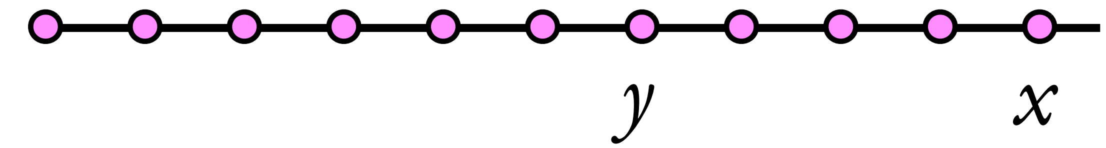
δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

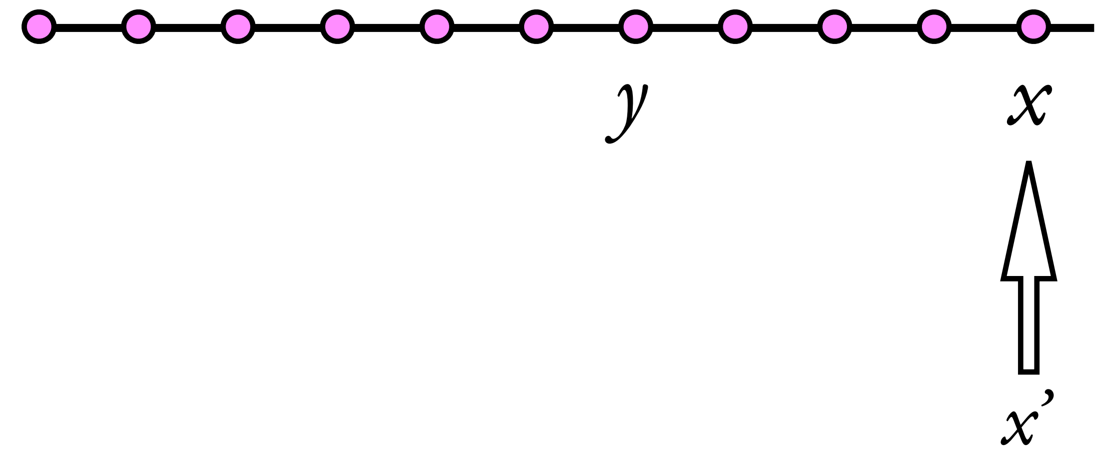
δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

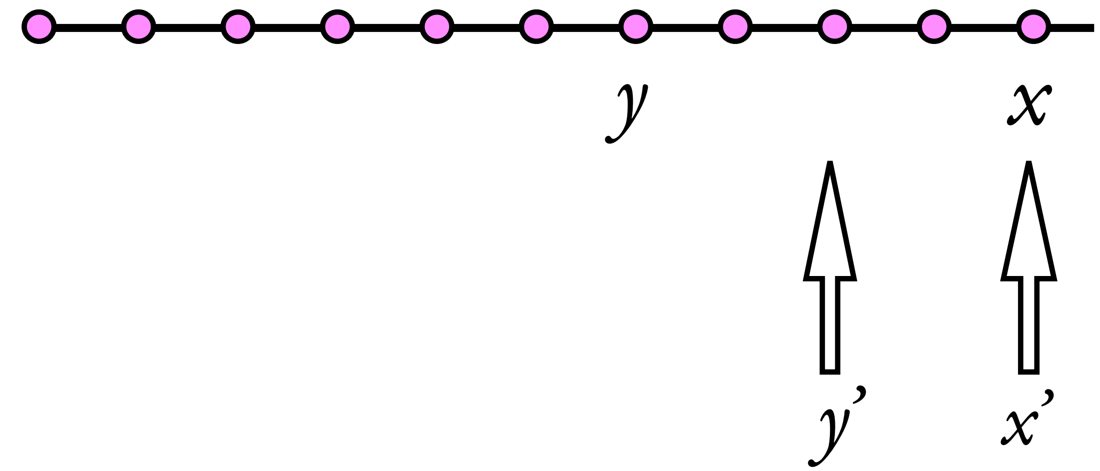
δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

Example D-automaton $\mathscr{A}$

x,y – variables of $\mathscr{A}$

no states



$\mathscr{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

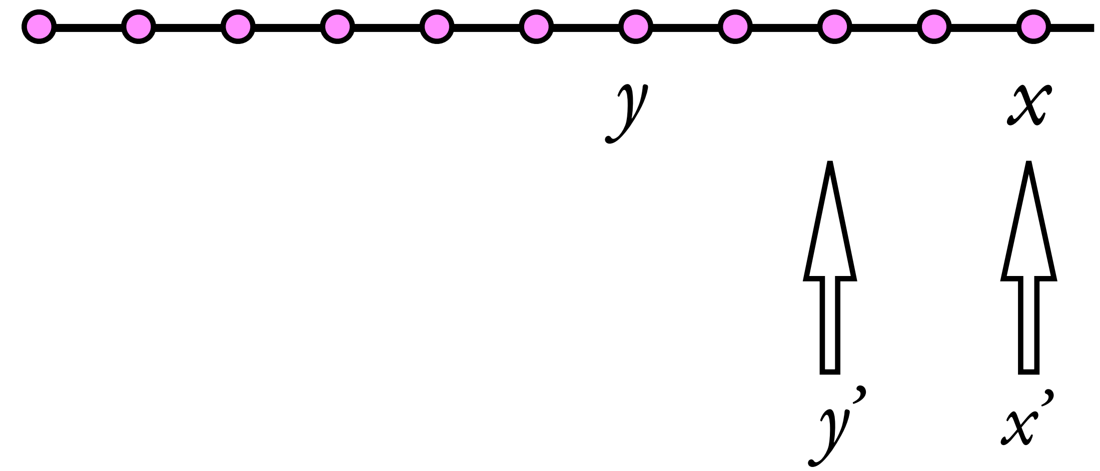
δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

Example D-automaton $\mathcal{A}$

x, y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

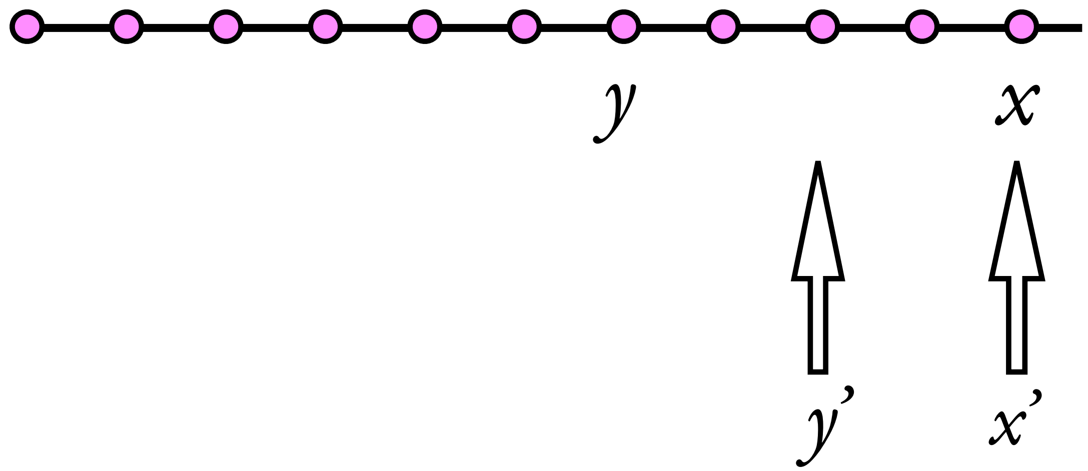
τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$:

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

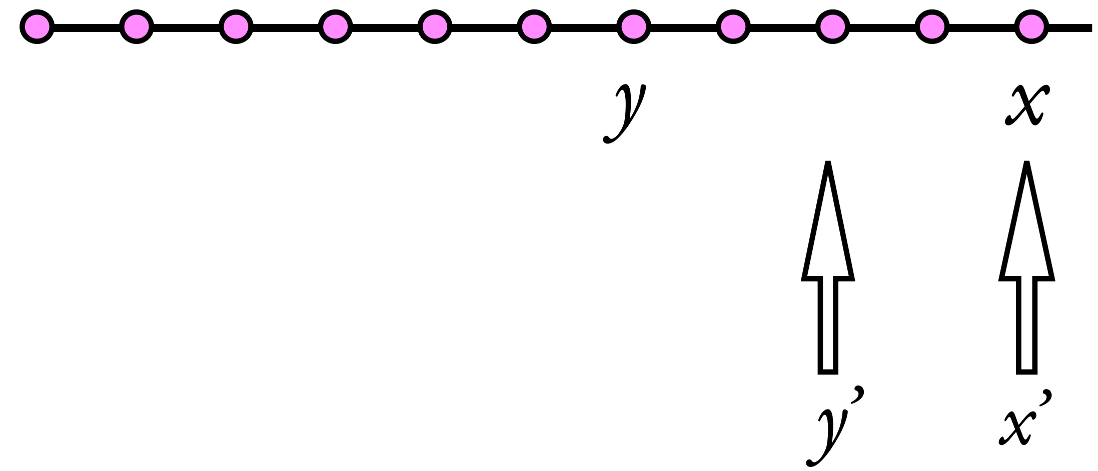
a finite run over $\mathbb{Q}$:

a a a b a b a a a a a

Example D-automaton $\mathscr{A}$

x,y – variables of $\mathscr{A}$

no states



$\mathscr{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

a a a b a b a a a a a

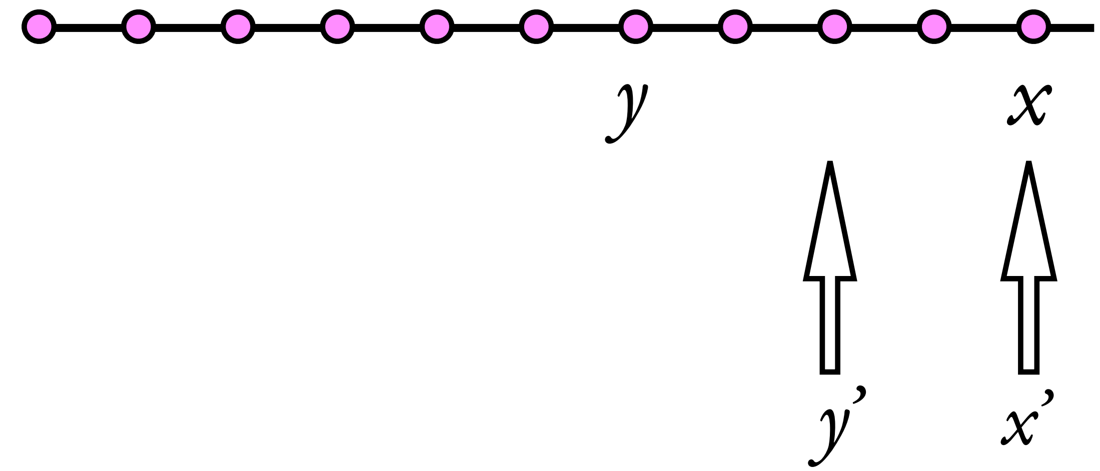
x

y

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

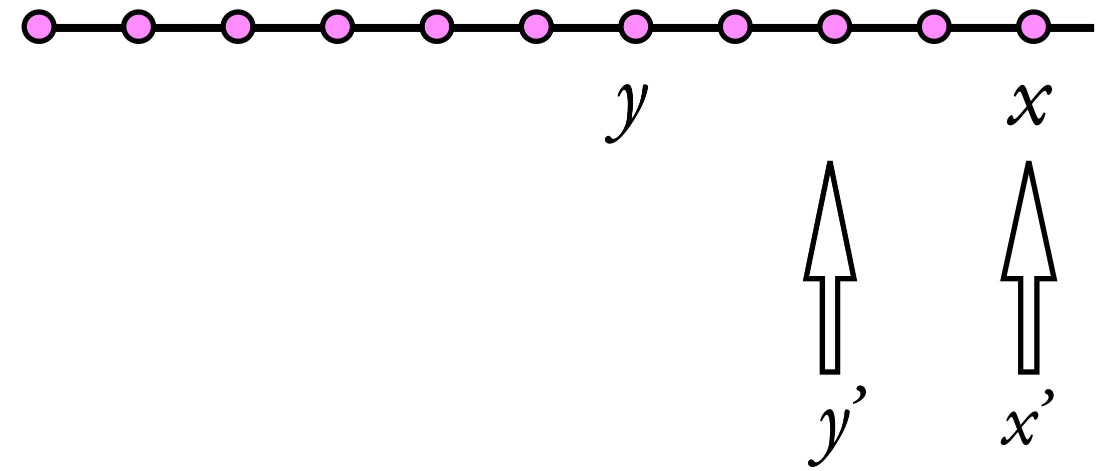
a a a b a b a a a a a

x	5
y	0

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

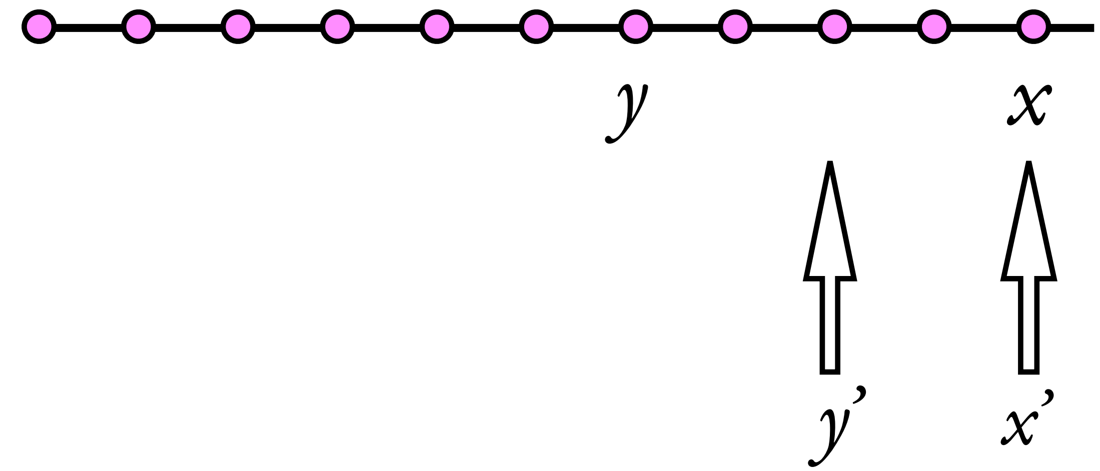
a a a b a b a a a a a

x	5	5
y	0	1

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

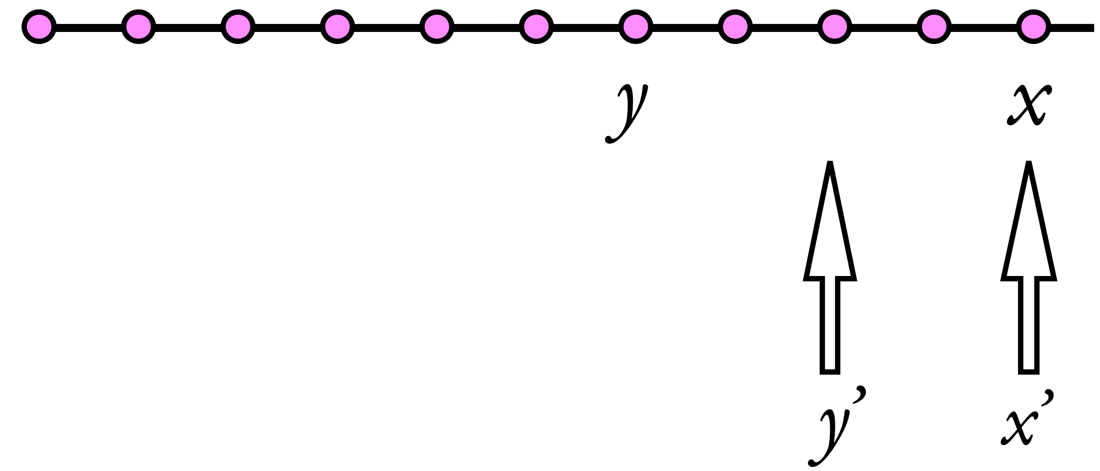
a a a b a b a a a a a

x	5	5	5
y	0	1	2

Example D-automaton $\mathcal{A}$

x, y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

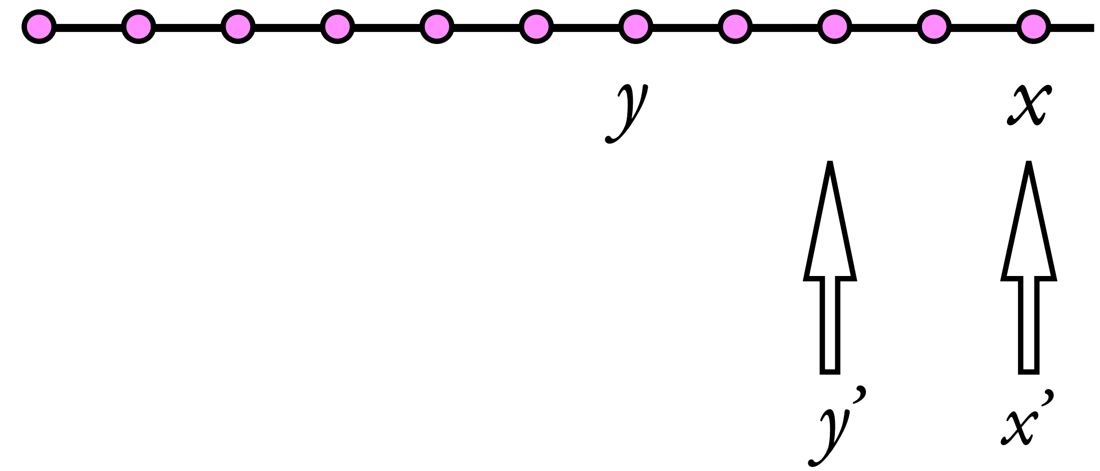
a a a b a b a a a a a

x	5	5	5	5
y	0	1	2	4

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

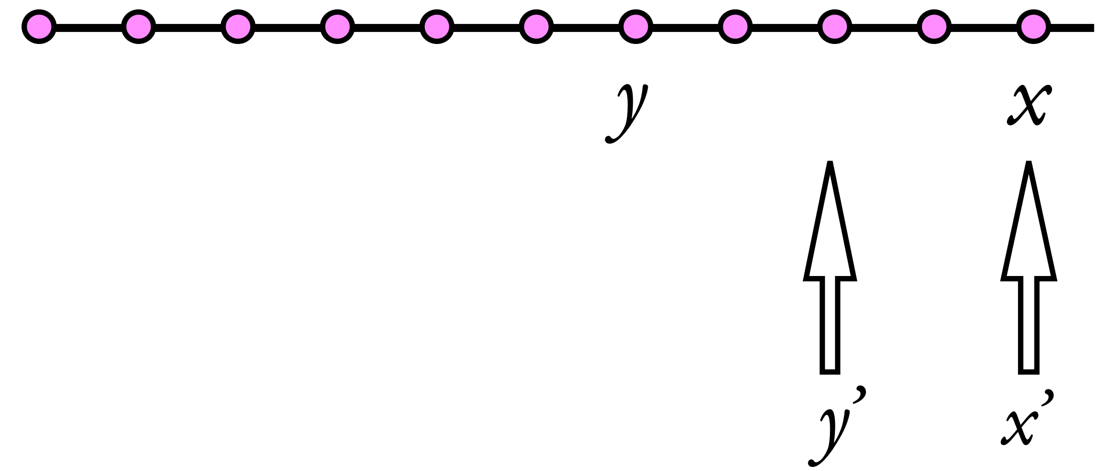
a a a b a b a a a a a

x	5	5	5	5	5
y	0	1	2	4	0

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

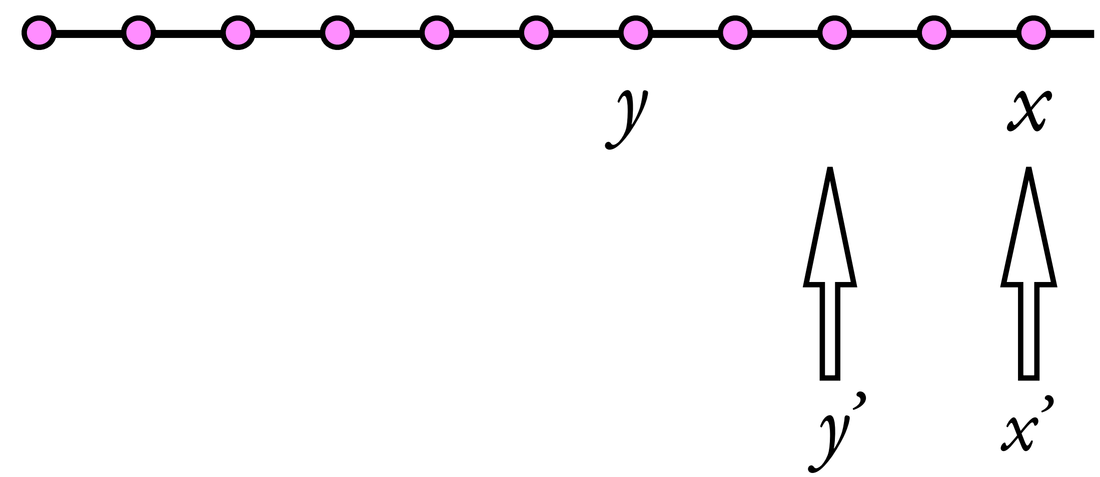
a a a b a b a a a a a

x	5	5	5	5	5	5
y	0	1	2	4	0	3

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

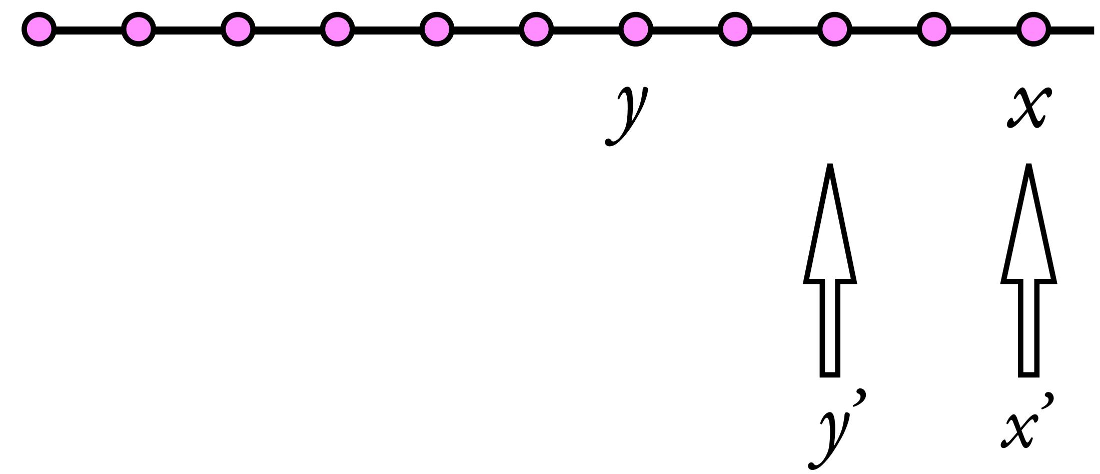
a a a b a b a a a a a

x	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

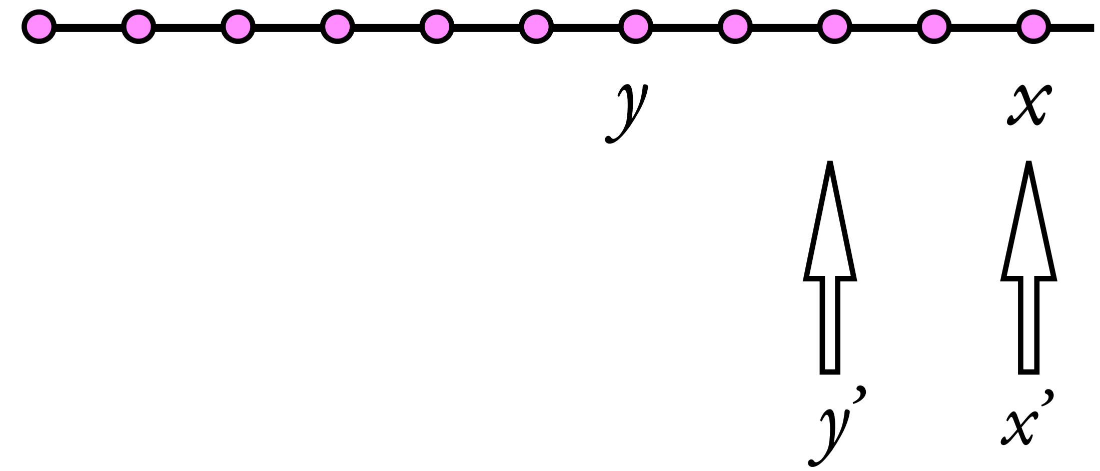
a a a b a b a a a a a

x	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$

Example D-automaton $\mathscr{A}$

x,y – variables of $\mathscr{A}$

no states



$\mathscr{A}$

- τ_I: $(x>0) \land (y=0)$
- δ_a: $(x'=x) \land (y<y'<x)$
- δ_b: $(x'=x) \land (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

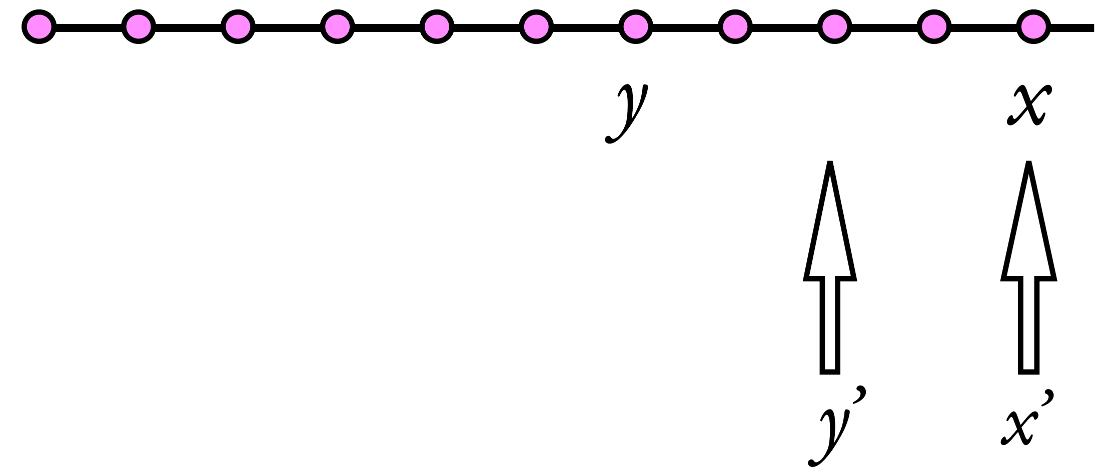
a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

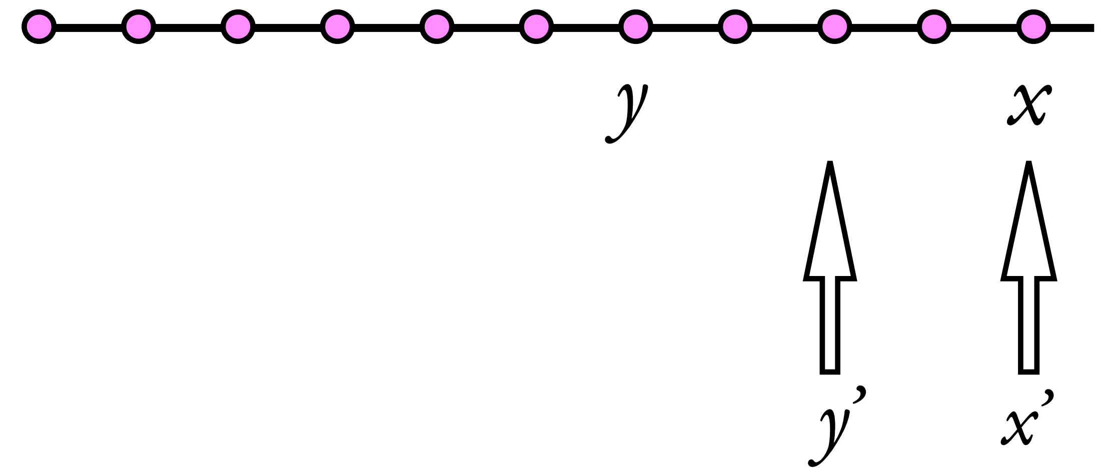
a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

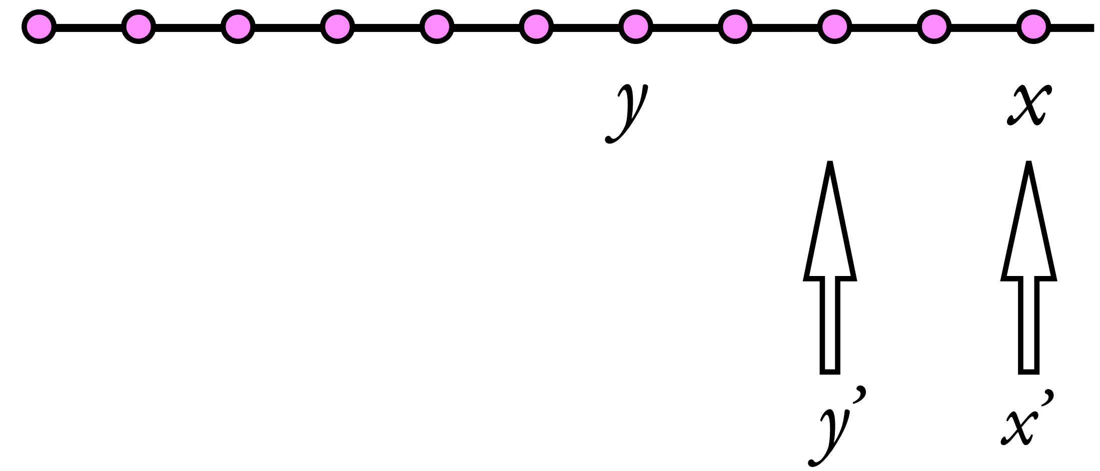
a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

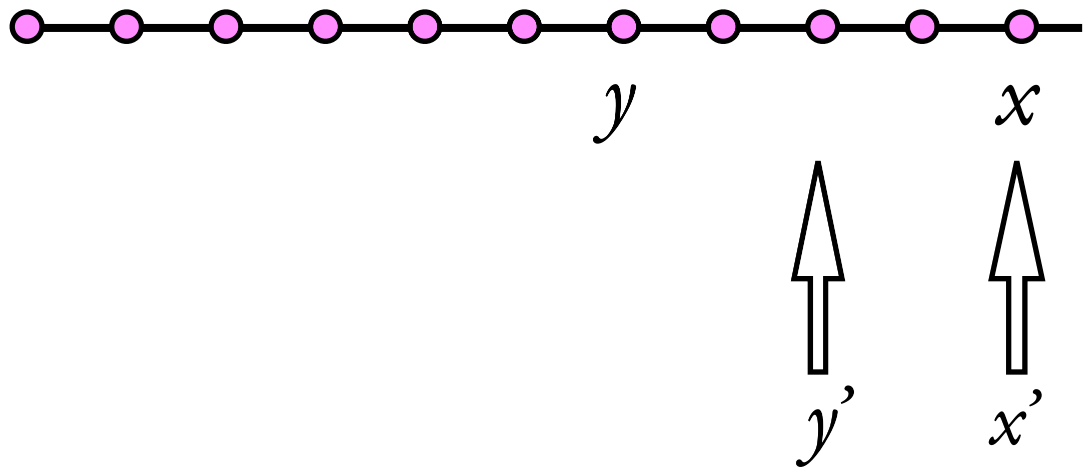
a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

Example D-automaton $\mathcal{A}$

x, y – variables of $\mathcal{A}$

no states



$\mathcal{A}$:

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

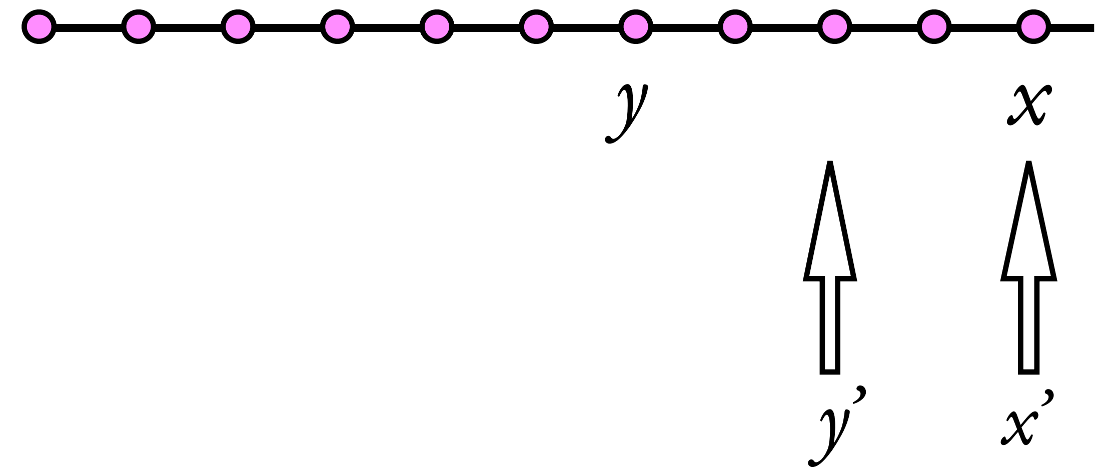
a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$:

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

a a a b a b a a a a a

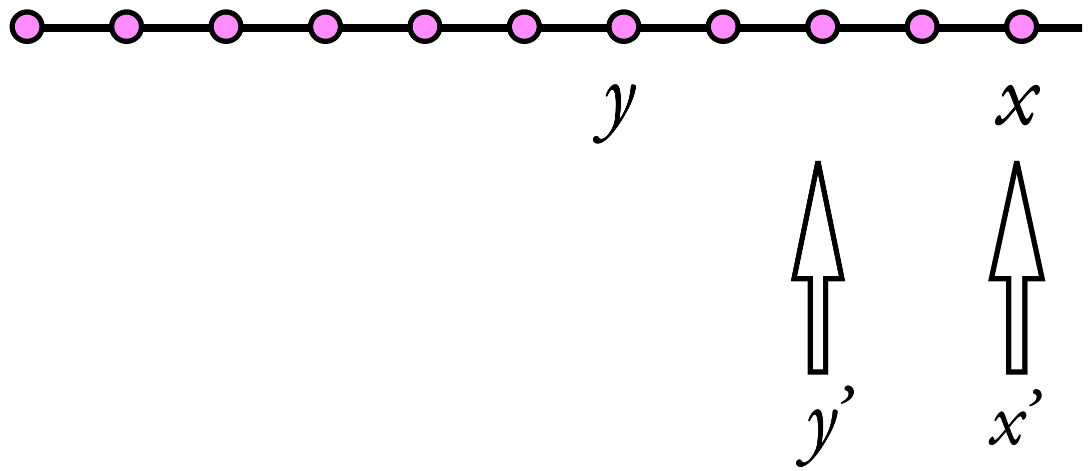
x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$:

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

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

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

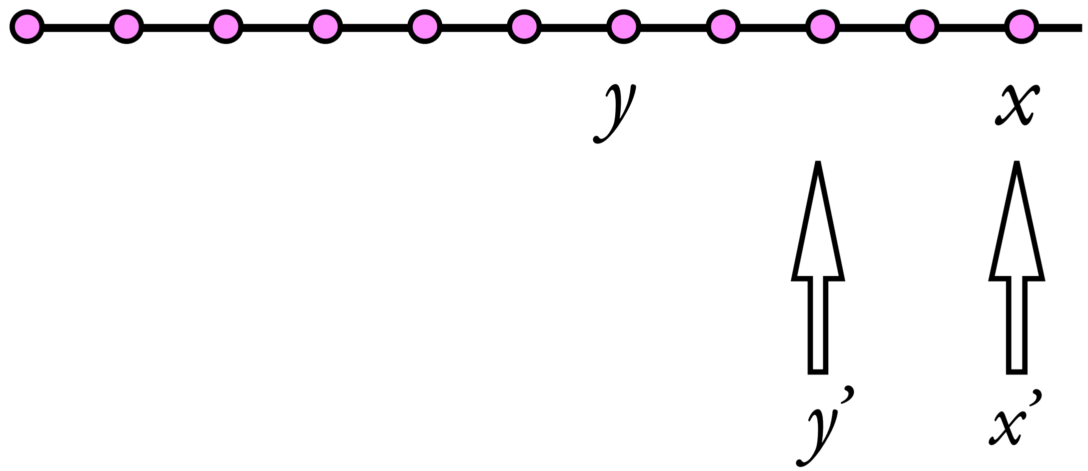
accepted language: $(a+b)^*a$

a finite run over $\mathbb{N}$:

Example D-automaton $\mathcal{A}$

x, y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

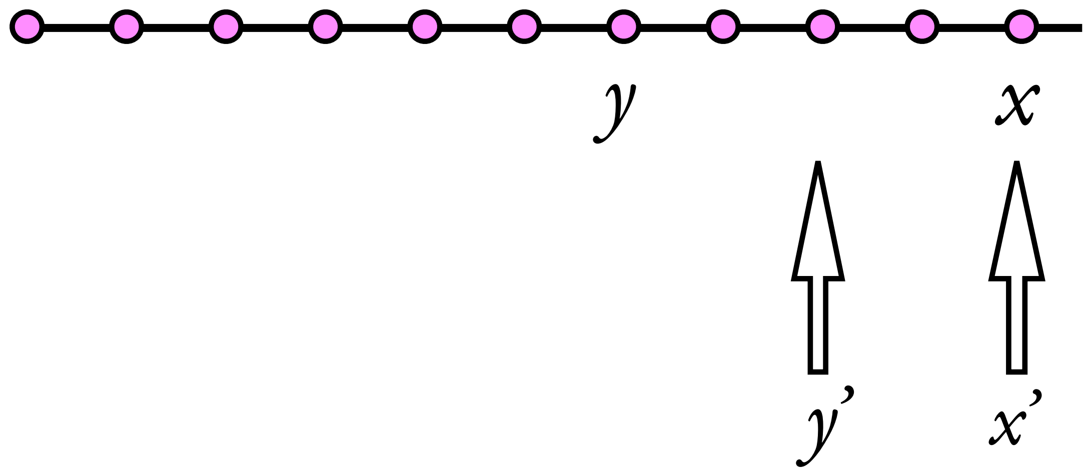
a finite run over $\mathbb{N}$:

a a a b a b a a a a

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$$\mathcal{A} \begin{cases} \tau_I: & (x>0) \wedge (y=0) \\ \delta_a: & (x'=x) \wedge (y<y'<x) \\ \delta_b: & (x'=x) \wedge (y'=0) \\ \tau_F: & (y=1) \end{cases}$$

a finite run over $\mathbb{Q}$:

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

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

a finite run over $\mathbb{N}$:

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

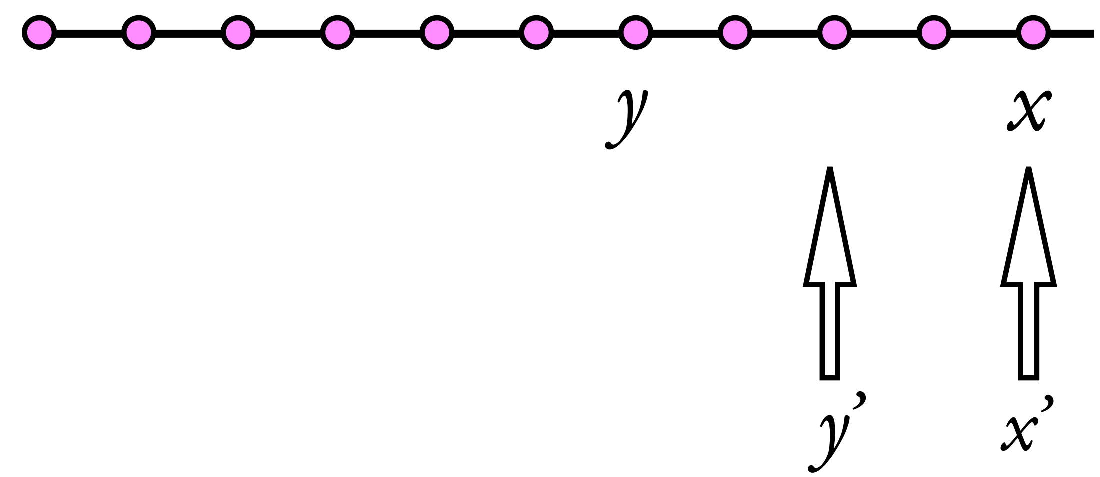
x

y

Example D-automaton $\mathcal{A}$

x, y – variables of $\mathcal{A}$

no states



$\mathcal{A}$:

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

a finite run over $\mathbb{N}$:

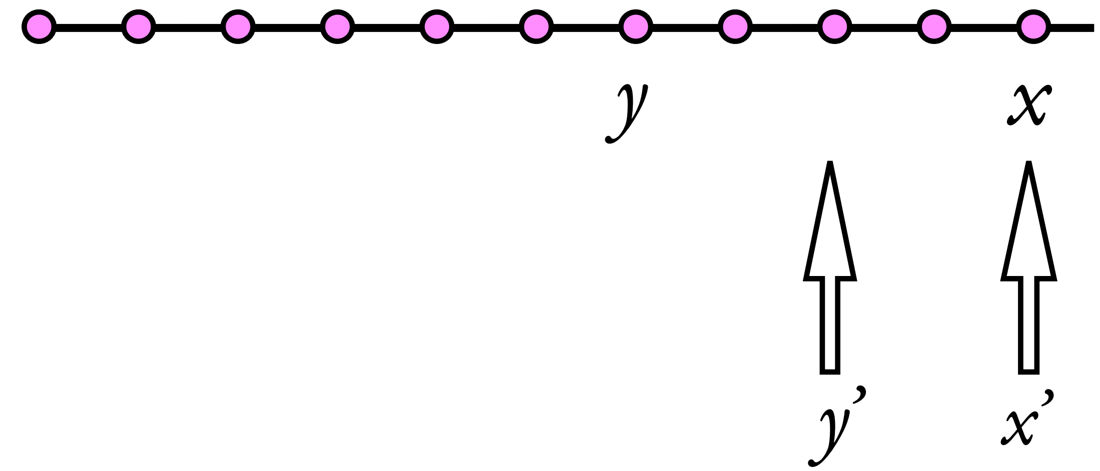
a a a b a b a a a a

x	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	1	2	3	4

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$\mathcal{A}$:

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

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

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

a finite run over $\mathbb{N}$:

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

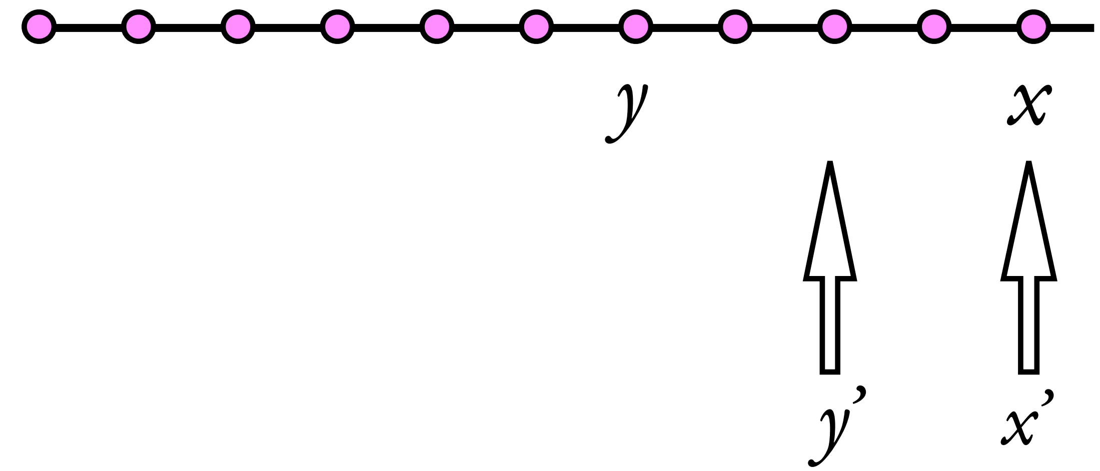
x	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	1	2	3	4

accepted language: $(a^*b)^*a$

Example D-automaton $\mathcal{A}$

x, y – variables of $\mathcal{A}$

no states



$\mathcal{A}$

- τ_I: $(x>0) \wedge (y=0)$
- δ_a: $(x'=x) \wedge (y<y'<x)$
- δ_b: $(x'=x) \wedge (y'=0)$
- τ_F: $(y=1)$

a finite run over $\mathbb{Q}$:

		a	a	a	b	a	b	a	a	a	a	a
x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

a finite run over $\mathbb{N}$:

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

accepted language: $(a^*b)^*\,a$

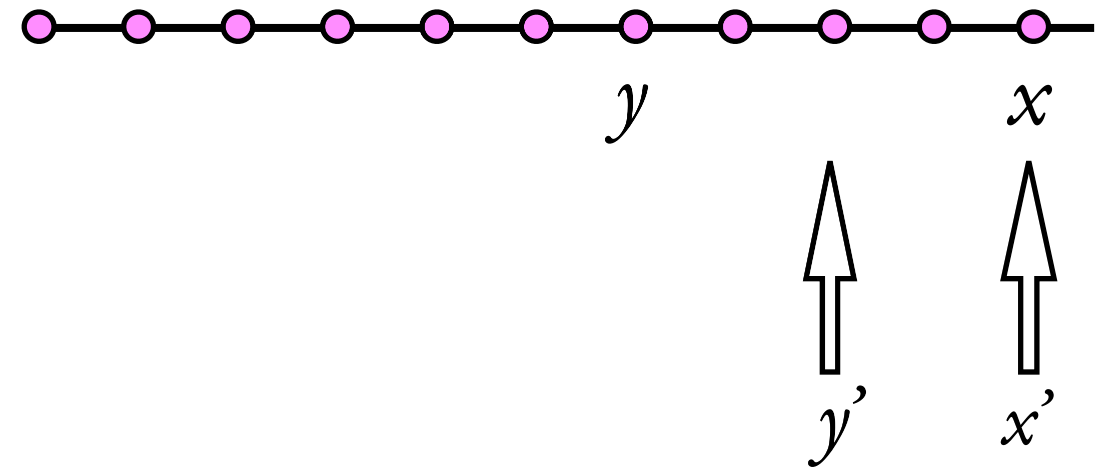
an infinite run over $\mathbb{N}$:

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

Example D-automaton $\mathcal{A}$

x,y – variables of $\mathcal{A}$

no states



$$\mathcal{A}\ \begin{cases} \tau_I: & (x>0)\ \wedge\ (y=0) \\ \delta_a: & (x'=x)\ \wedge\ (y<y'<x) \\ \delta_b: & (x'=x)\ \wedge\ (y'=0) \\ \tau_F: & (y=1) \end{cases}$$

a finite run over $\mathbb{Q}$:

a a a b a b a a a a a

x	5	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{4}{5}$	1

accepted language: $(a+b)^*a$

a finite run over $\mathbb{N}$:

a a a b a b a a a a

x	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	1	2	3	4

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

a a a b a b a a a a

x	5	5	5	5	5	5	5	5	5	5	5
y	0	1	2	4	0	3	0	1	2	3	4

accepted language: $(a^{B}b)^{\omega}$

Adding database constraints

$x,y,z,\ldots$ – a finite set of variables
$R,S,\ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x' > x) \wedge R(x) \wedge \neg S(x, y')$

δ_b: $(x' = 0)$

τ_F: $(x=1)$

Adding database constraints

$x,y,z,\ldots$ – a finite set of variables
$R,S,\ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x' > x) \wedge R(x) \wedge \neg S(x, y')$

δ_b: $(x' = 0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

Adding database constraints

$x,y,z,\ldots$ – a finite set of variables
$R,S,\ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M?*

a finite run over $\mathbb{N}$:

a a a b a b a

Adding database constraints

$x,y,z,\dots$ – a finite set of variables
$R,S,\dots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

$\}\exists$

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$ $\Big\}\exists$

Adding database constraints

$x,y,z,\ldots$ – a finite set of variables
$R,S,\ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

x 0 2 3 4 0 3 0 1

R={(0),(1),(2),(3),(4)} S={} } $\exists$

Adding database constraints

$x,y,z,\ldots$ – a finite set of variables
$R,S,\ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

		a	*a*	*a*	*b*	*a*	*b*	*a*	
x	0	2	3	4	0	3	0	1

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$ $\Big\}\exists$

accepted language: $(a^*b)^*a$

Adding database constraints

$x,y,z,\ldots$ – a finite set of variables
$R,S,\ldots$ – a finite set of relational symbols

τ_I: $(x{=}0)$

δ_a: $(x'{>}x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'{=}0)$

τ_F: $(x{=}1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

x	0	2	3	4	0	3	0	1

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$ $\}\exists$

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x' > x) \wedge R(x) \wedge \neg S(x, y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

x 0 2 3 4 0 3 0 1

$R=\{(0),(1),(2),(3),(4)\}$ S={} } $\exists$

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

a a a b a b a a b a a…

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x' > x) \wedge R(x) \wedge \neg S(x, y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

x 0 2 3 4 0 3 0 1

$R=\{(0),(1),(2),(3),(4)\}$ S={} $\Big\}\exists$

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

a a a b a b a a b a a…

$R=\{(0),(1),(2),(3),(4)\}$ S={}

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x' > x) \wedge R(x) \wedge \neg S(x, y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database M, a word w, and an accepting run over w consistent with M?

a finite run over $\mathbb{N}$:

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

x 0 2 3 4 0 3 0 1

$R=\{(0),(1),(2),(3),(4)\}$ S={}

$\}\exists$

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

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

x 0 1 3 4 0 3 0 1 4 0 1 3

$R=\{(0),(1),(2),(3),(4)\}$ S={}

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x' > x) \wedge R(x) \wedge \neg S(x, y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

	a	*a*	*a*	*b*	*a*	*b*	*a*	
x	0	2	3	4	0	3	0	1

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$ $\}\exists$

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

	a	*a*	*a*	*b*	*a*	*b*	*a*	*a*	*b*	*a*	*a*…	
x	0	1	3	4	0	3	0	1	4	0	1	3

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$

accepted language: $(a^B b)^\omega$

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

Decide emptiness of *D*-automata over ω-words, with databases

EMPTINESS: is there a *finite* database *M*, a word *w*, and an accepting run over *w* consistent with *M*?

a finite run over $\mathbb{N}$:

a a a b a b a

x	0	2	3	4	0	3	0	1

R={(0),(1),(2),(3),(4)} S={} } ∃

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

a a a b a b a a b a a…

x	0	1	3	4	0	3	0	1	4	0	1	3

R={(0),(1),(2),(3),(4)} S={}

accepted language: $(a^{B}b)^{\omega}$

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

1. infinite, ordered data

Decide emptiness of D-automata over ω-words, with databases

EMPTINESS: is there a *finite* database M, a word w, and an accepting run over w consistent with M?

a finite run over $\mathbb{N}$:

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

x: 0 2 3 4 0 3 0 1

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$ $\}\exists$

accepted language: $(a^*b)^*a$

an infinite run over $\mathbb{N}$:

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

x: 0 1 3 4 0 3 0 1 4 0 1 3

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$

accepted language: $(a^B b)^\omega$

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

1. infinite, ordered data

Decide emptiness of D-automata over ω-words, with databases

2. unbounded database

EMPTINESS: is there a *finite* database M, a word w, and an accepting run over w consistent with M?

a finite run over ℕ:

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

x	0	2	3	4	0	3	0	1

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$ } ∃

accepted language: $(a^*b)^*a$

an infinite run over ℕ:

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

x	0	1	3	4	0	3	0	1	4	0	1	3

$R=\{(0),(1),(2),(3),(4)\}$ $S=\{\}$

accepted language: $(a^B b)^\omega$

Adding database constraints

$x, y, z, \ldots$ – a finite set of variables
$R, S, \ldots$ – a finite set of relational symbols

τ_I: $(x=0)$

δ_a: $(x'>x) \wedge R(x) \wedge \neg S(x,y')$

δ_b: $(x'=0)$

τ_F: $(x=1)$

Decide emptiness of D-automata over ω-words, with databases

1. infinite, ordered data

2. unbounded database

3. infinite runs

EMPTINESS: is there a *finite* database M, a word w, and an accepting run over w consistent with M?

a finite run over ℕ:

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

x 0 2 3 4 0 3 0 1

$R=\{(0),(1),(2),(3),(4)\}$ S={} $\}\exists$

accepted language: $(a^*b)^*a$

an infinite run over ℕ:

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

x 0 1 3 4 0 3 0 1 4 0 1 3

$R=\{(0),(1),(2),(3),(4)\}$ S={}

accepted language: $(a^B b)^\omega$

Known results
deciding emptiness (for ω-runs)

	no database	with database
$\mathbb{N}$,=	Kaminski, Francez (94)	Deutsch, Sui, Vianu, Zhou (06) PSPACE
$\mathbb{Q}$,<	Čerans (94) PSPACE	Deutsch, Hull, Patrizi, Vianu (09) PSPACE
$\mathbb{N}$,<	Čerans (94) NONPRIMITIVE	?

Known results

deciding emptiness (for ω-runs)

	no database	with database
$\mathbb{N}$,=	Kaminski, Francez (94)	Deutsch, Sui, Vianu, Zhou (06) PSPACE
$\mathbb{Q}$,<	Čerans (94) PSPACE	Deutsch, Hull, Patrizi, Vianu (09) PSPACE
$\mathbb{N}$,<	Čerans (94) NONPRIMITIVE	?
$\mathscr{D}$,<	PSPACE	PSPACE

Known results

deciding emptiness (for ω-runs)

	no database	with database
$\mathbb{N}$,=	Kaminski, Francez (94)	Deutsch, Sui, Vianu, Zhou (06) PSPACE
$\mathbb{Q}$,<	Čerans (94) PSPACE	Deutsch, Hull, Patrizi, Vianu (09) PSPACE
$\mathbb{N}$,<	Čerans (94) NONPRIMITIVE	PSPACE
$\mathscr{D}$,<	PSPACE	PSPACE

Known results

deciding emptiness (for ω-runs)

	no database	with database
$\mathbb{N}$,=	Kaminski, Francez (94)	Deutsch, Sui, Vianu, Zhou (06) PSPACE
$\mathbb{Q}$,<	Čerans (94) PSPACE	Deutsch, Hull, Patrizi, Vianu (09) PSPACE
$\mathbb{N}$,<	PSPACE	PSPACE
$\mathscr{D}$,<	PSPACE	PSPACE

Outline

1. Define automata model
2. Analyze automata over dense orders
3. Analyze automata over discrete orders
4. Add database and infinite runs

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0=1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, ...

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0<1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0=1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, ...

a

$\Cup$

$\exists(x,y)$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0<1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …

a

$\Cup$ $\exists(x',y')$ $\Cup$ $\exists(x,y)$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0<1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …

a

$\Cup \qquad \Cup$

$$\exists(x',y') \quad \exists(x,y): \quad \delta_a(x, y, x', y')$$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0=1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …

a

$$\exists(x',y') \quad \exists(x,y): \quad \delta_a(x, y, x', y')$$

$(5,0)$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0=1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …

a

$$\exists(x',y') \quad \exists(x,y): \quad \delta_a(x, y, x', y')$$

$\Cup$ $(5,1)$ $\quad$ $\Cup$ $(5,0)$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0=1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …

a

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

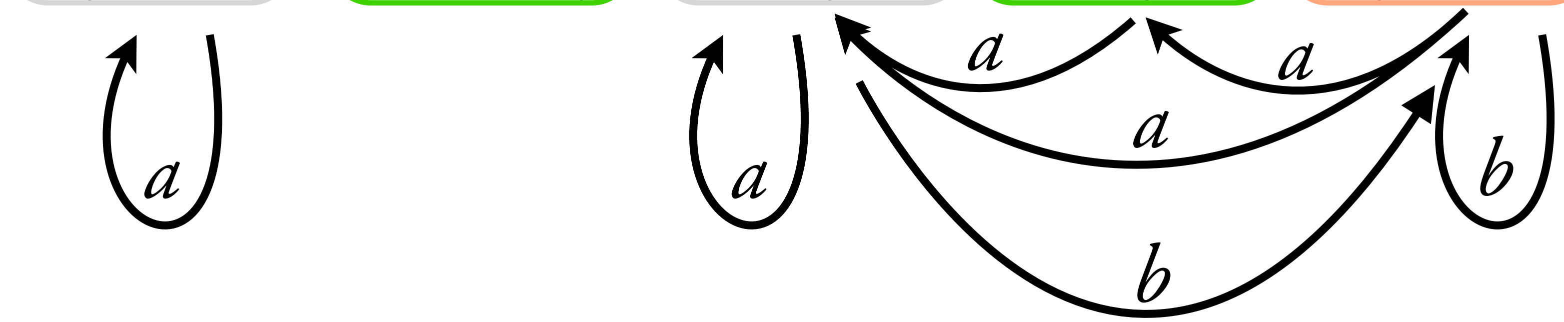
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction

$0<y<1<x$, $0<x<1=y$, $0=1<y<x$, $0<1=y<x$, $0=y<1<x$, $0=y<1=x$, …



Dense case

- $D = \mathbb{Q}$
- finite words
- no database

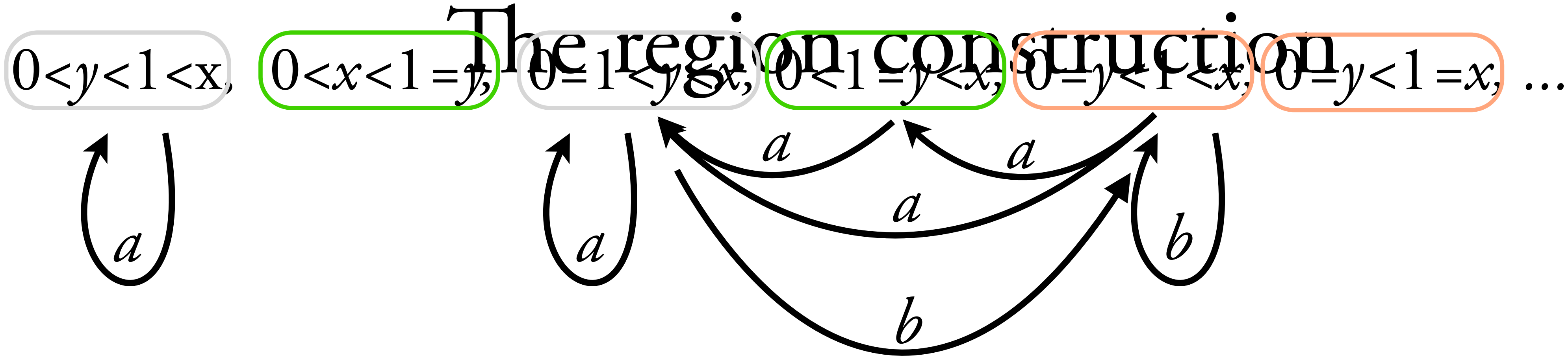
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction



bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

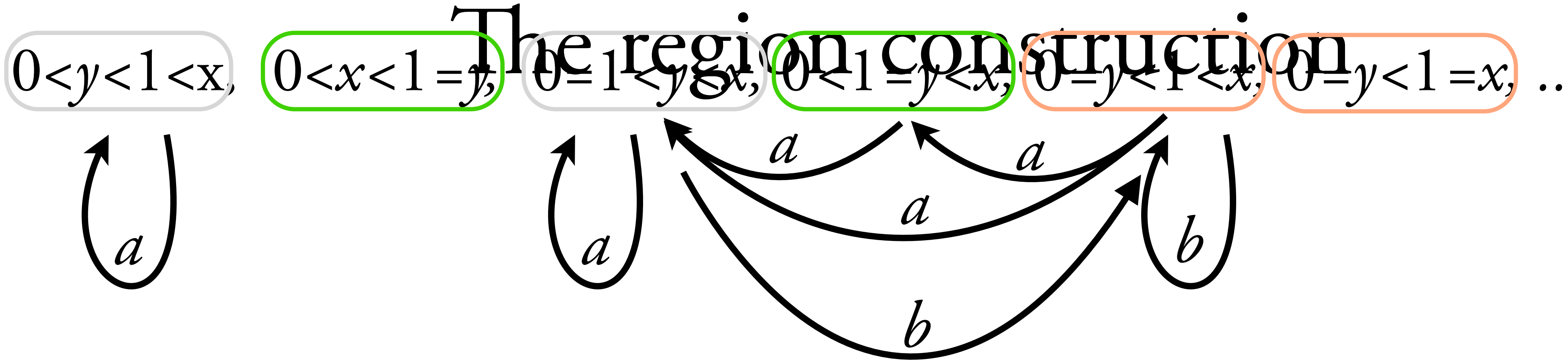
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction



bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

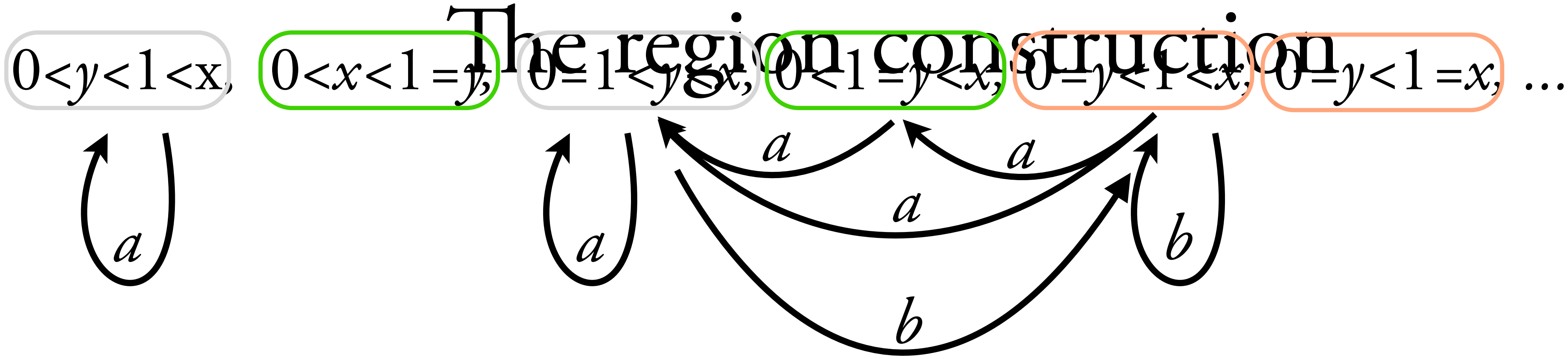
τ_I: $(x>0) \wedge (y=0)$

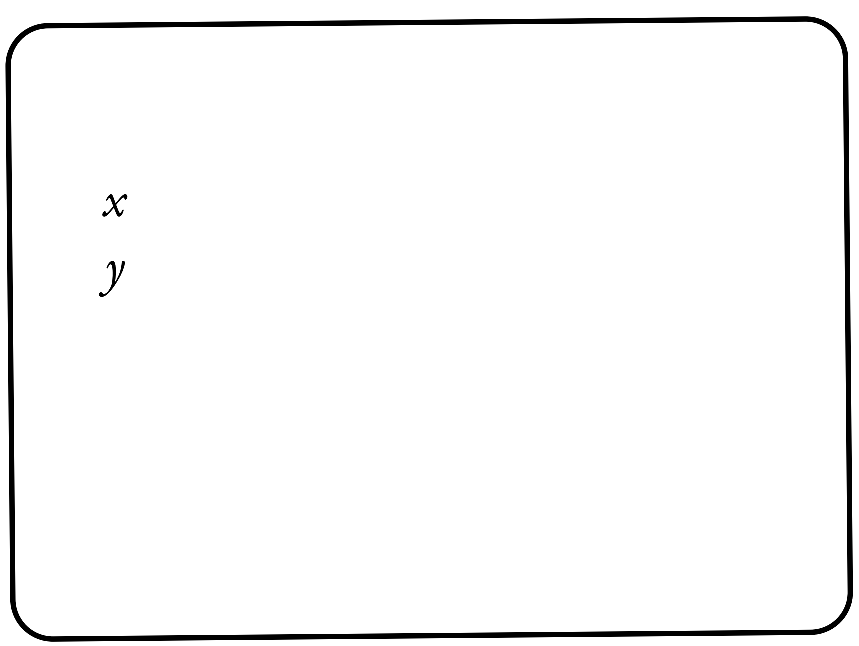
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

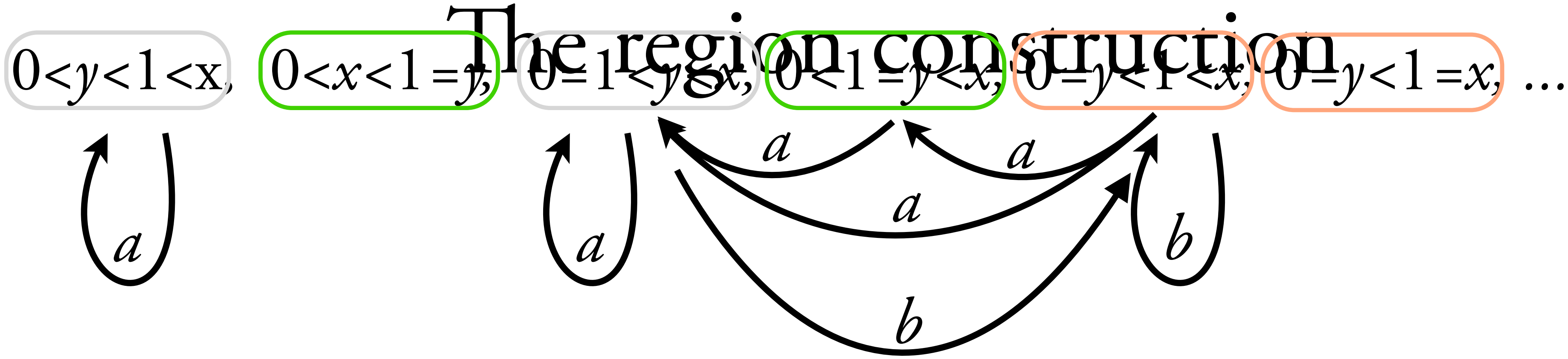
τ_I: $(x>0) \wedge (y=0)$

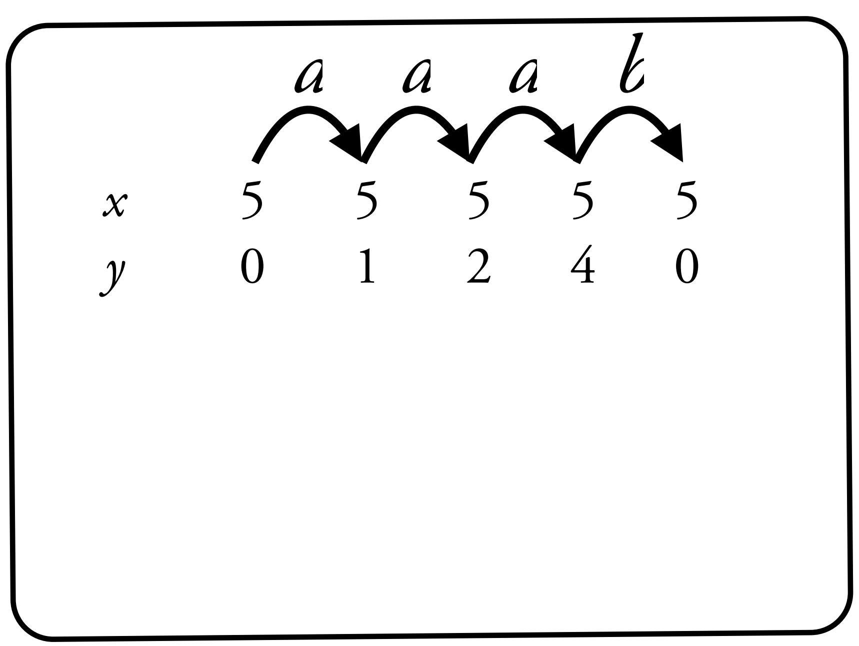
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

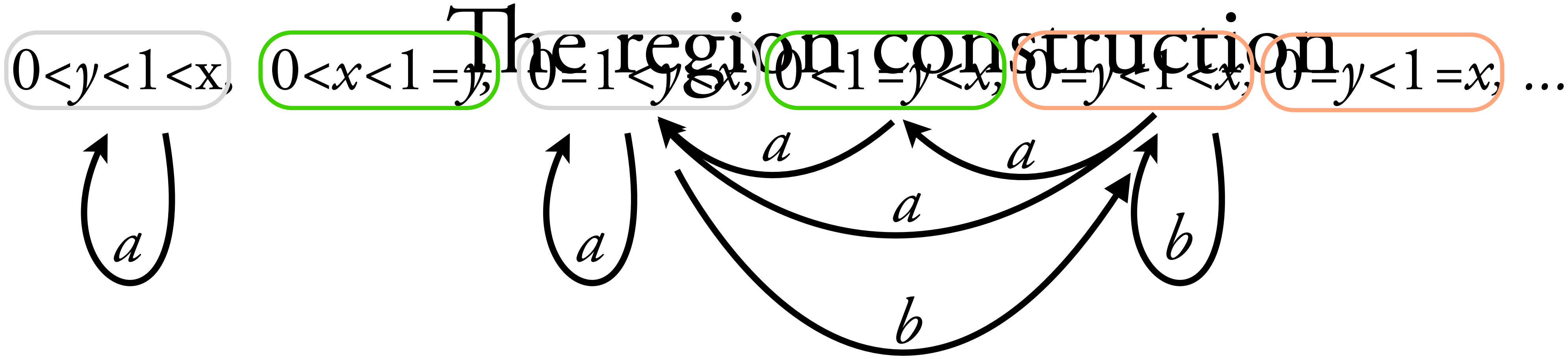
τ_I: $(x>0) \wedge (y=0)$

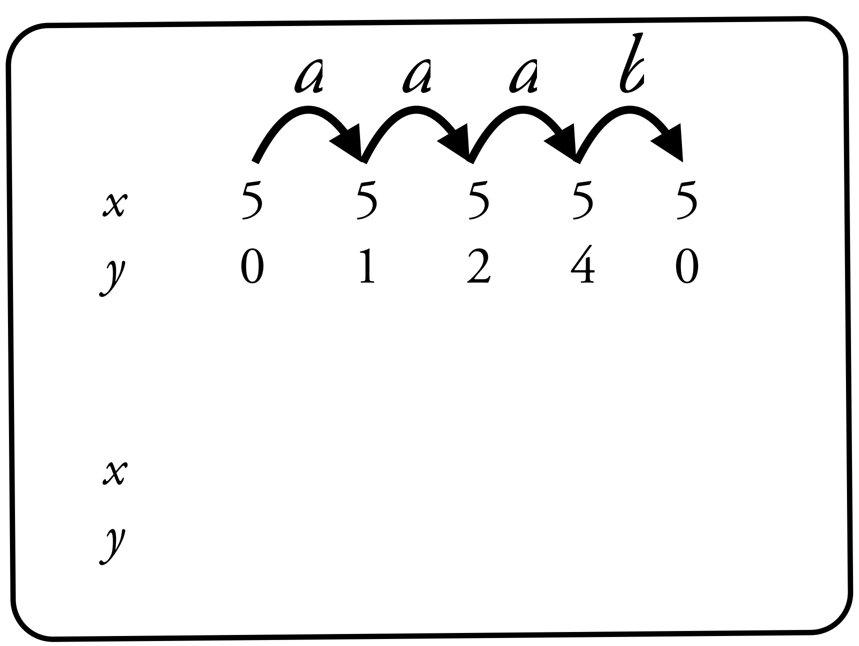
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

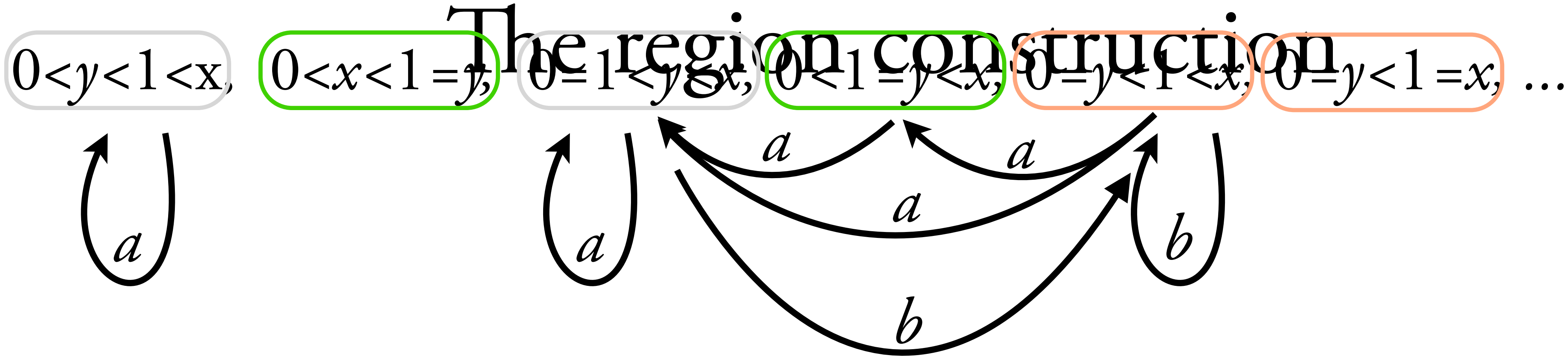
τ_I: $(x>0) \wedge (y=0)$

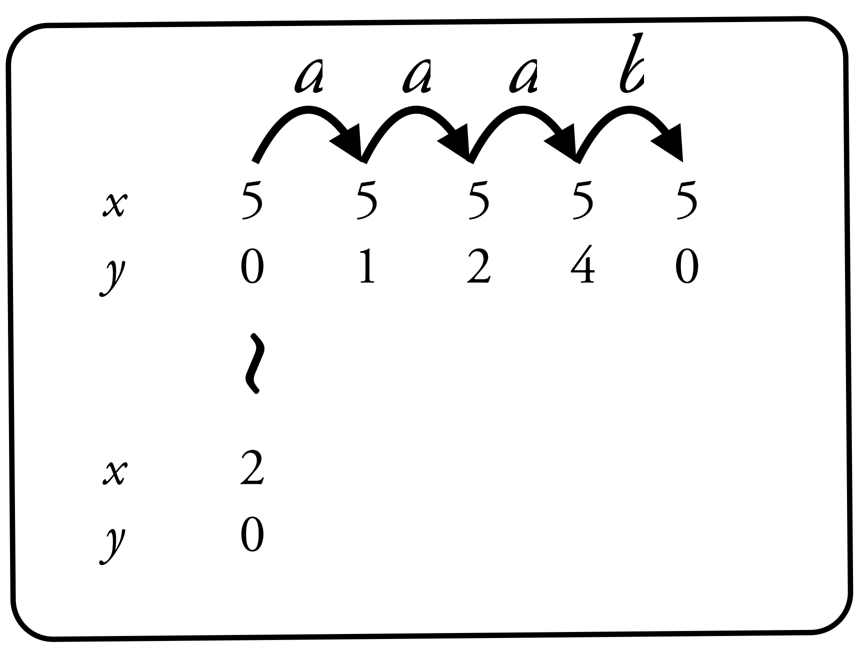
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation

Dense case

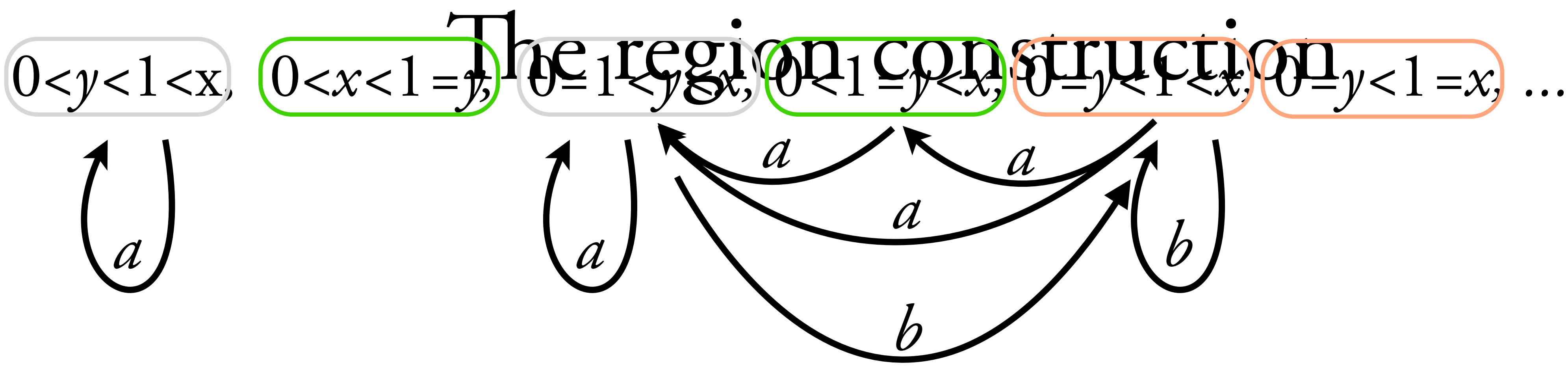
- $D = \mathbb{Q}$
- finite words
- no database

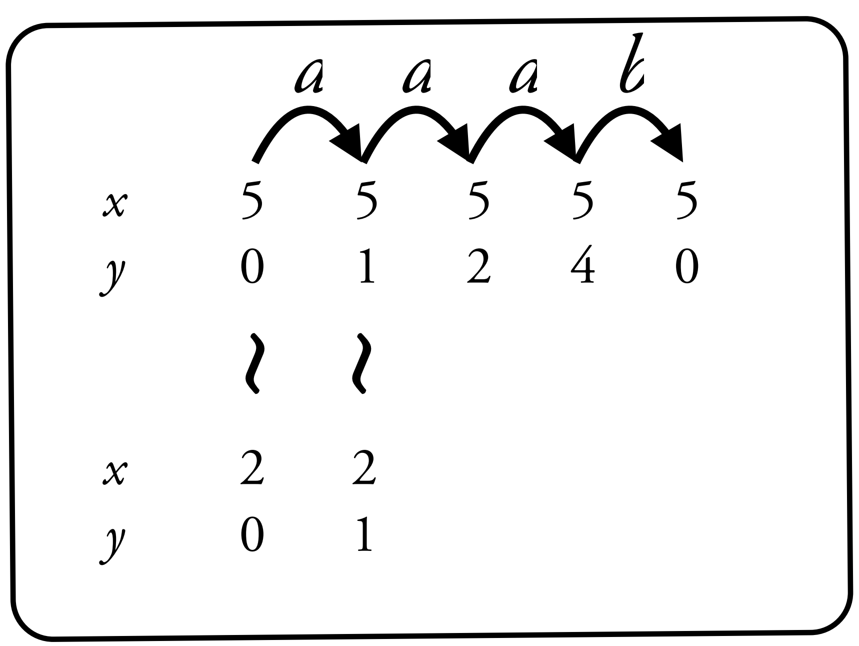
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

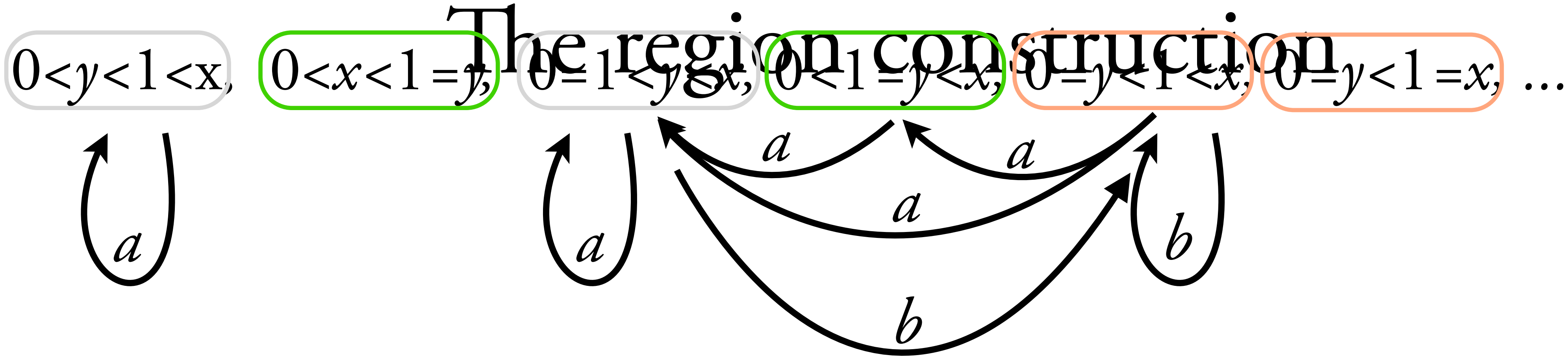
τ_I: $(x>0) \wedge (y=0)$

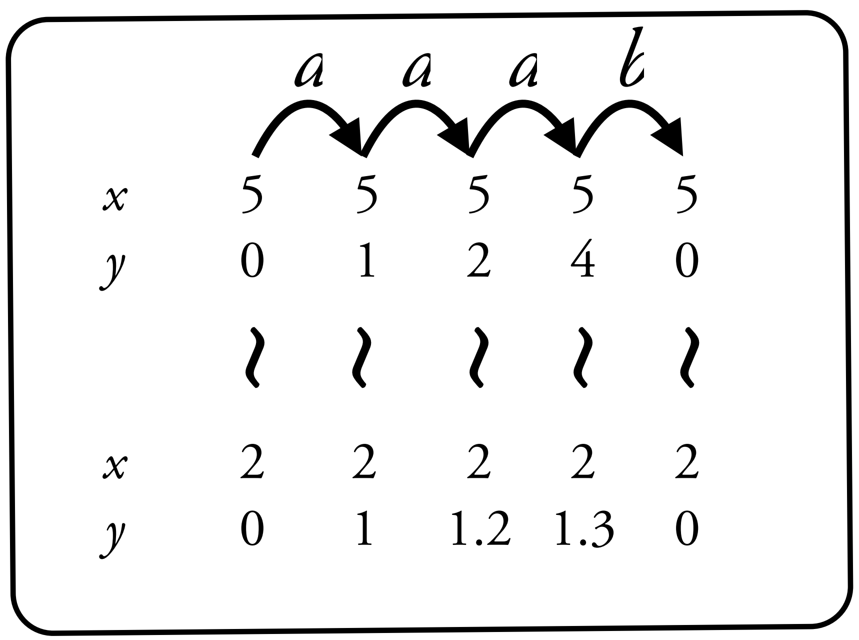
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

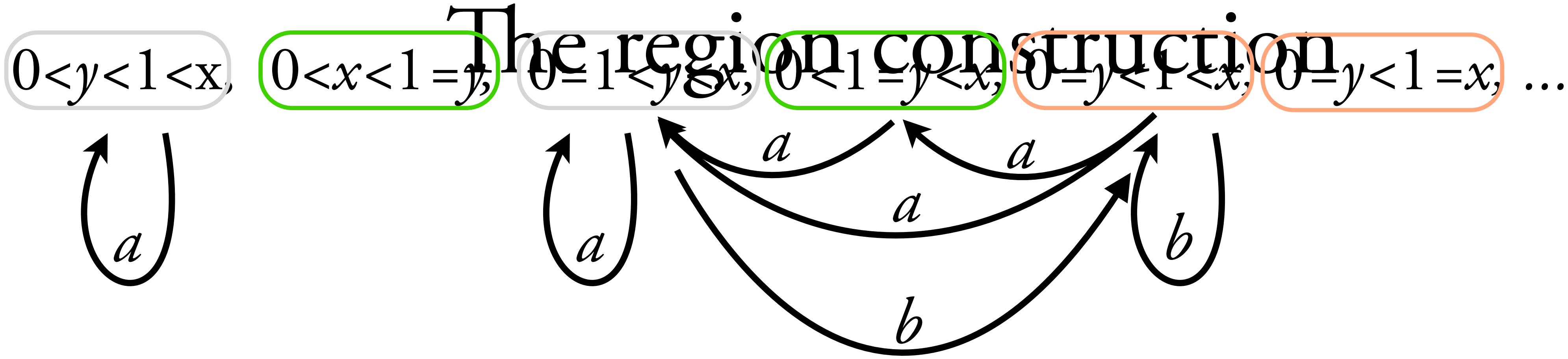
τ_I: $(x>0) \wedge (y=0)$

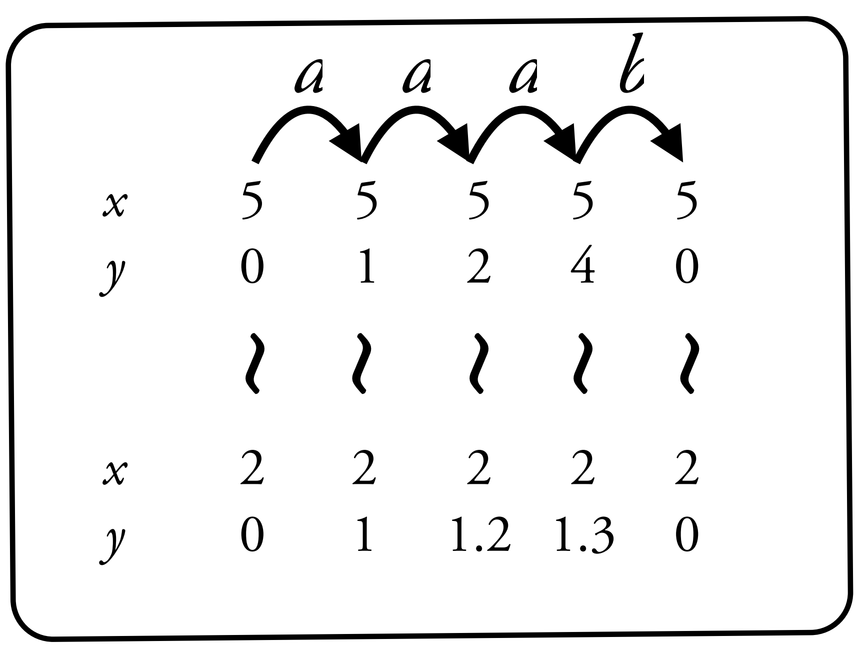
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation → runs of the region automaton correspond to runs of $\mathcal{A}$

Dense case

- $D = \mathbb{Q}$
- finite words
- no database

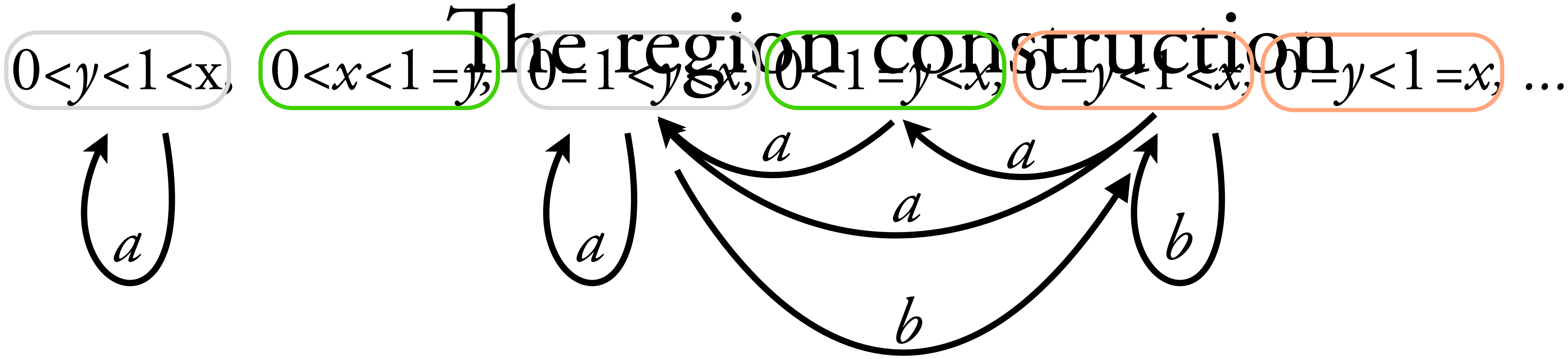
τ_I: $(x>0) \wedge (y=0)$

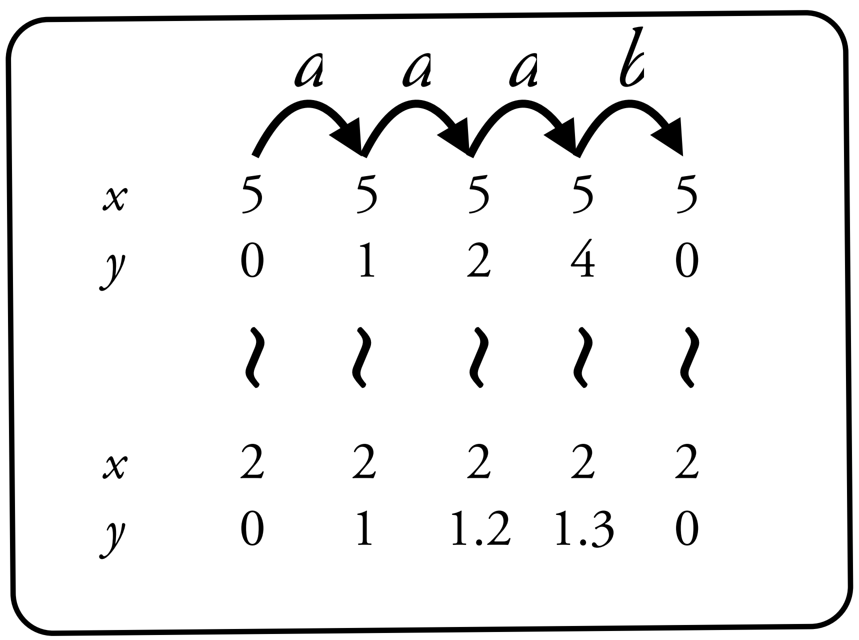
δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$

The region construction





bisimulation → runs of the region automaton correspond to runs of $\mathcal{A}$

accepted language: $(a+b)^*a$

The region construction

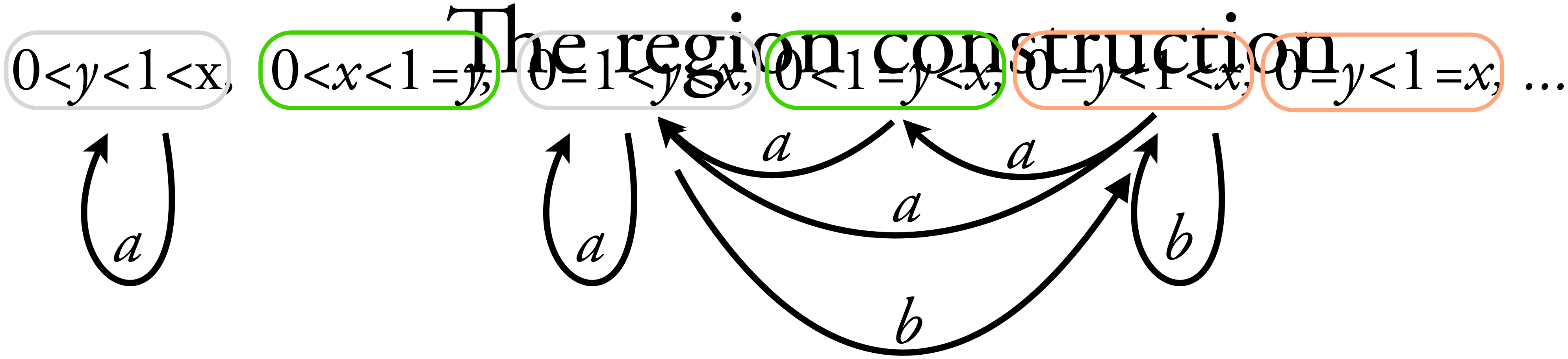
- $D = \mathbb{Q}$
- finite words
- no database

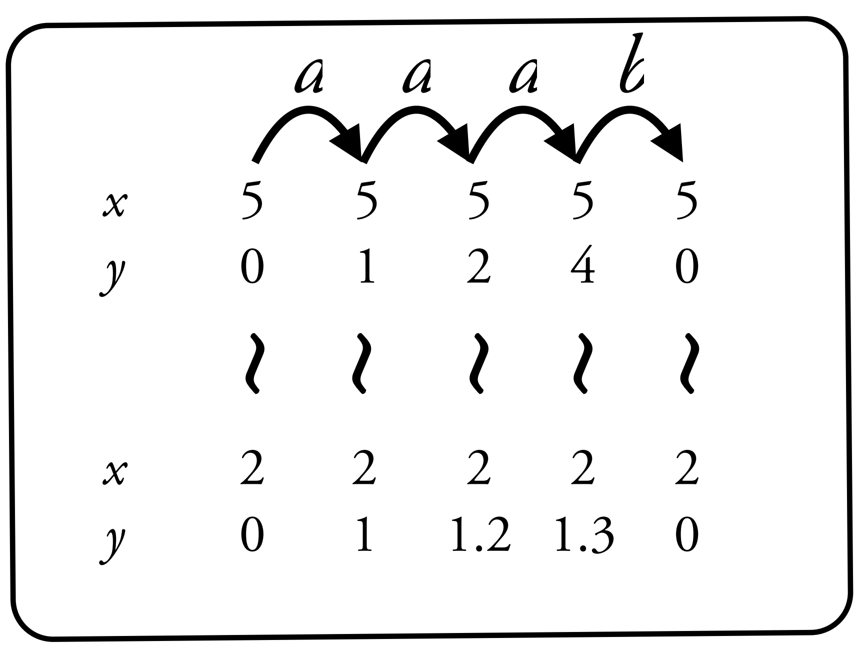
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





bisimulation → runs of the region automaton correspond to runs of $\mathcal{A}$

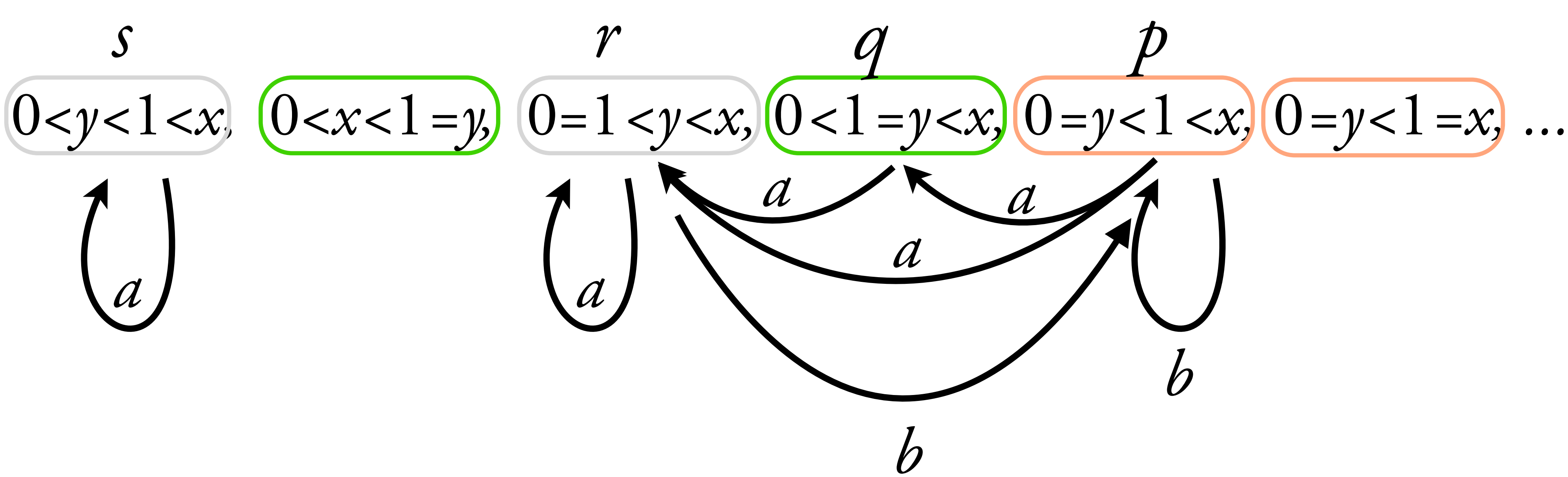
- $D = \mathbb{N}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$



bisimulation → runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

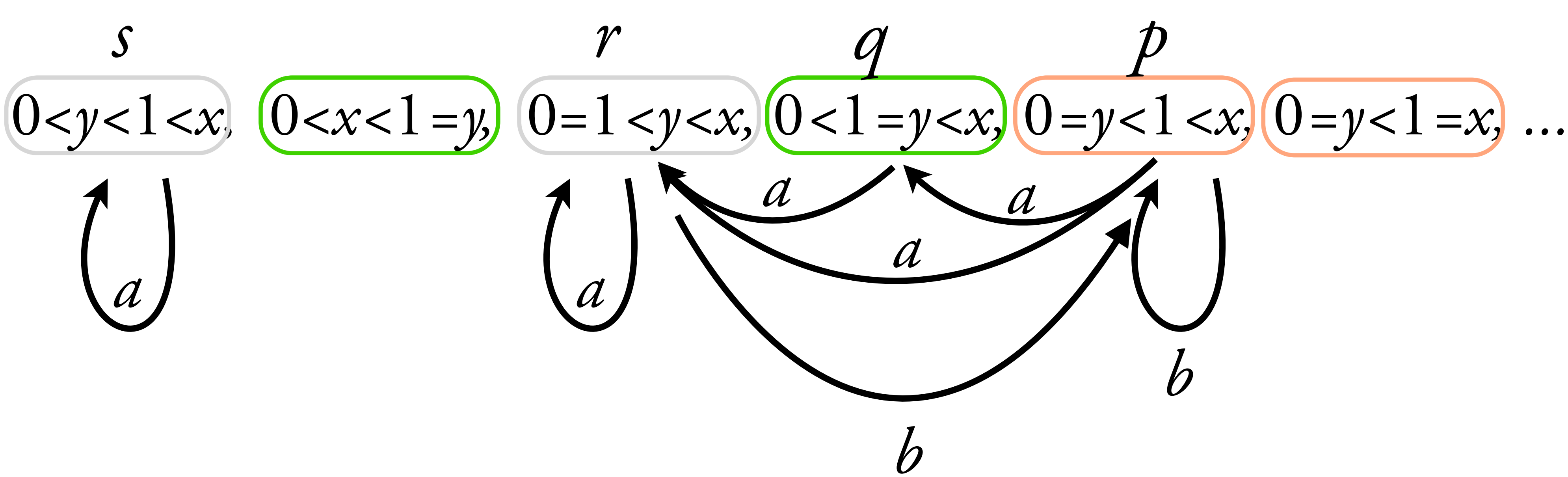
- $D = \mathbb{N}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$



bisimulation → runs of the region automaton correspond to runs of $\mathscr{A}$

Discrete case

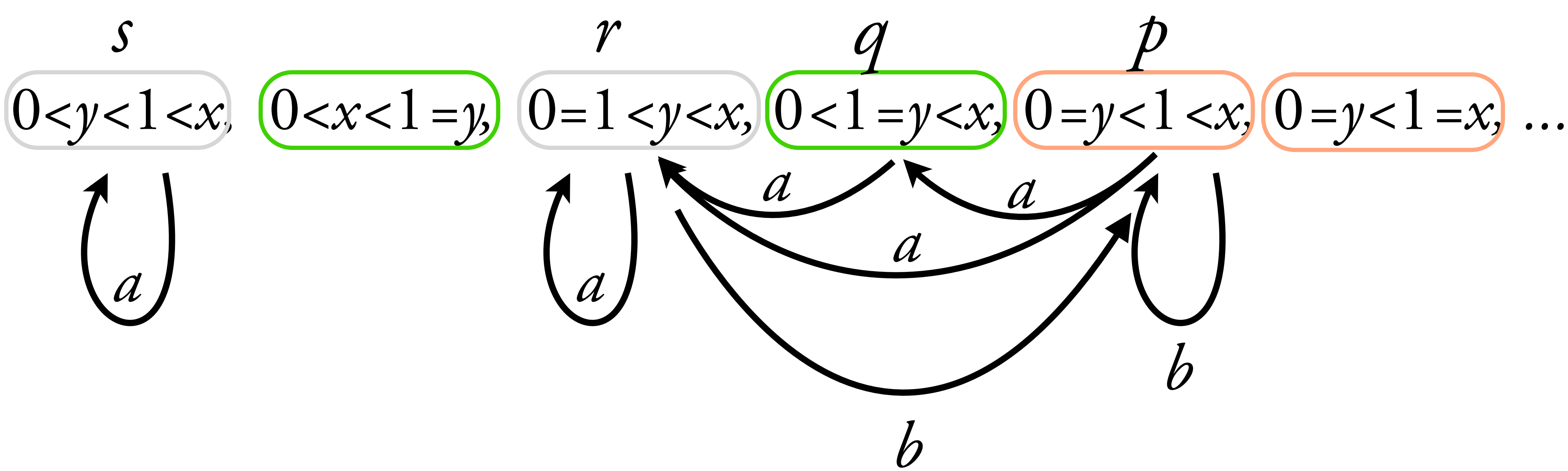
- $D = \mathbb{N}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$



~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathscr{A}$

Discrete case

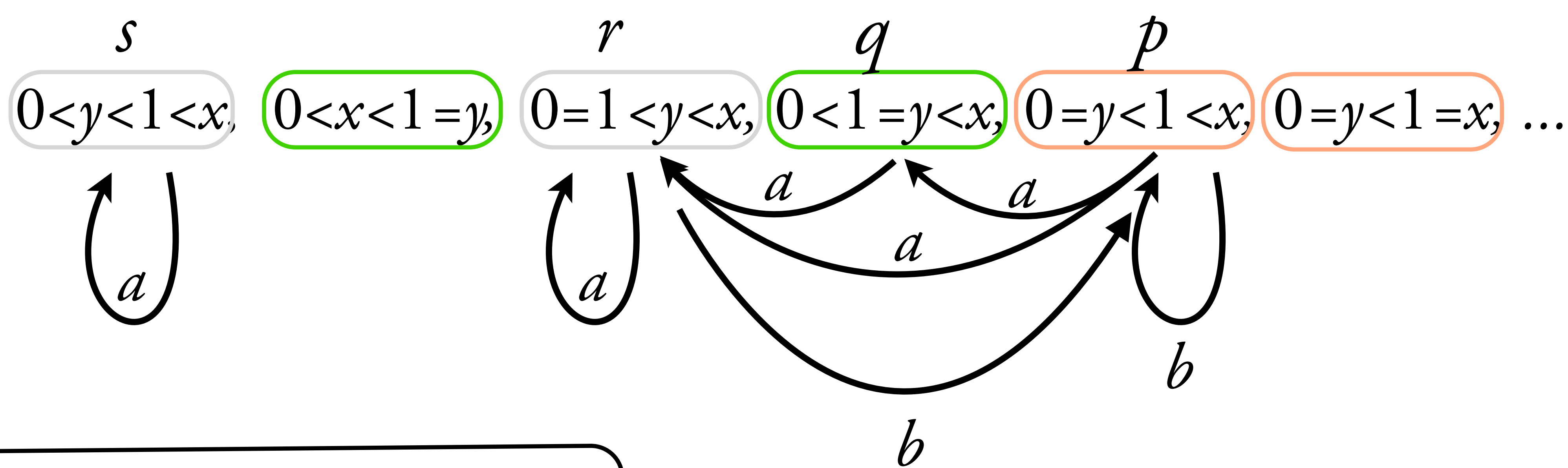
- $D = \mathbb{N}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$



~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathscr{A}$

Discrete case

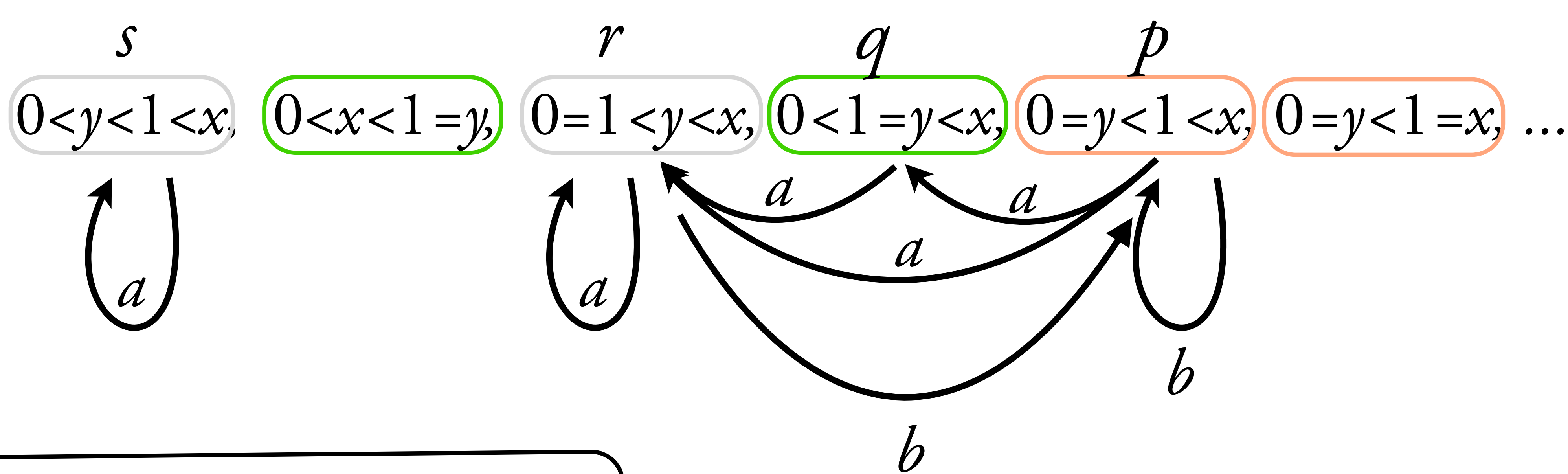
- $D = \mathbb{N}$
- finite words
- no database

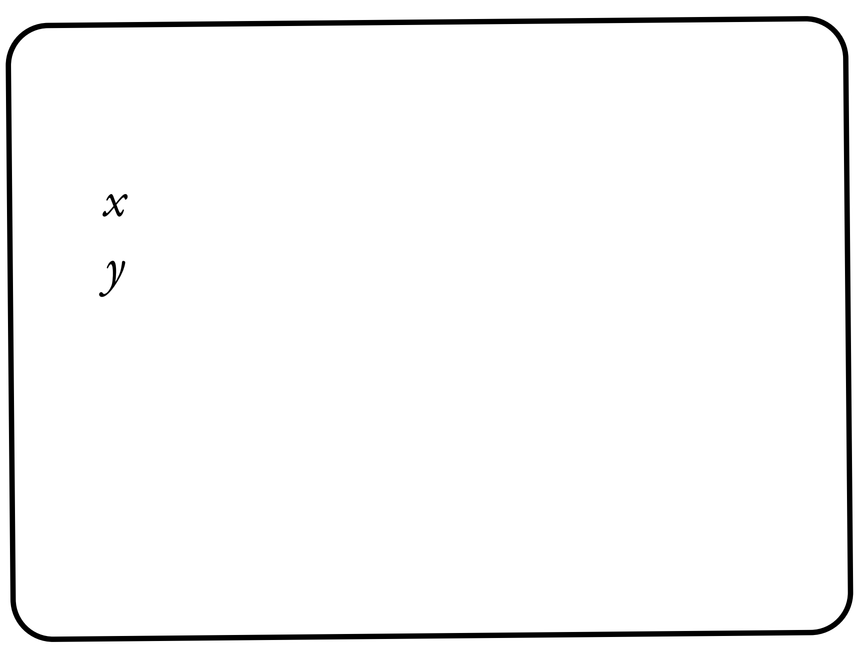
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathscr{A}$

Discrete case

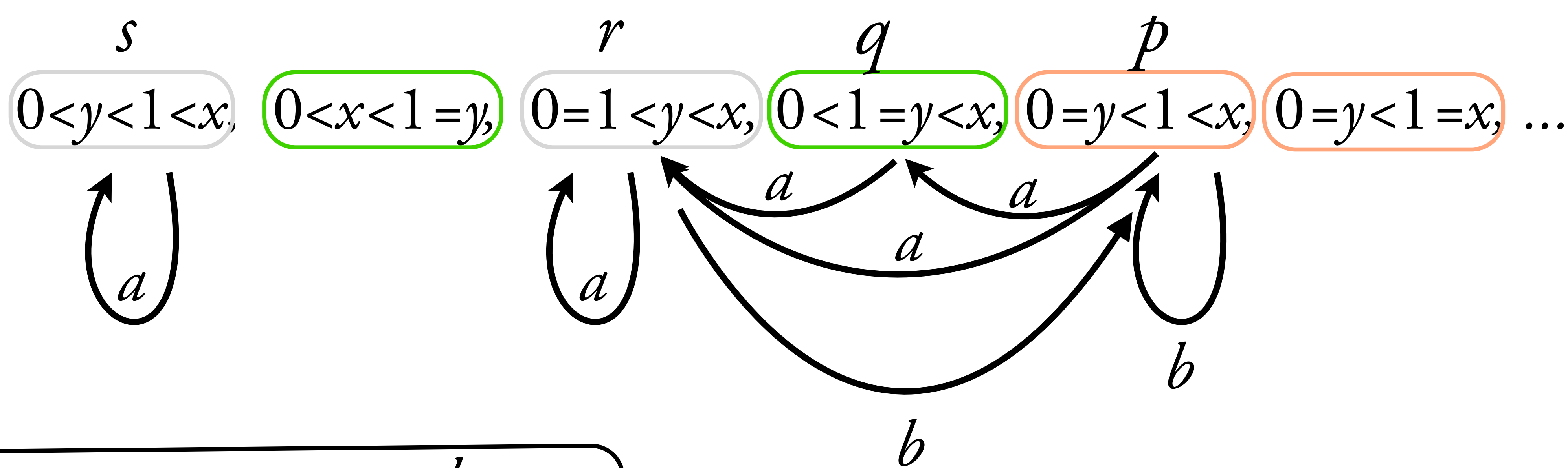
- $D = \mathbb{N}$
- finite words
- no database

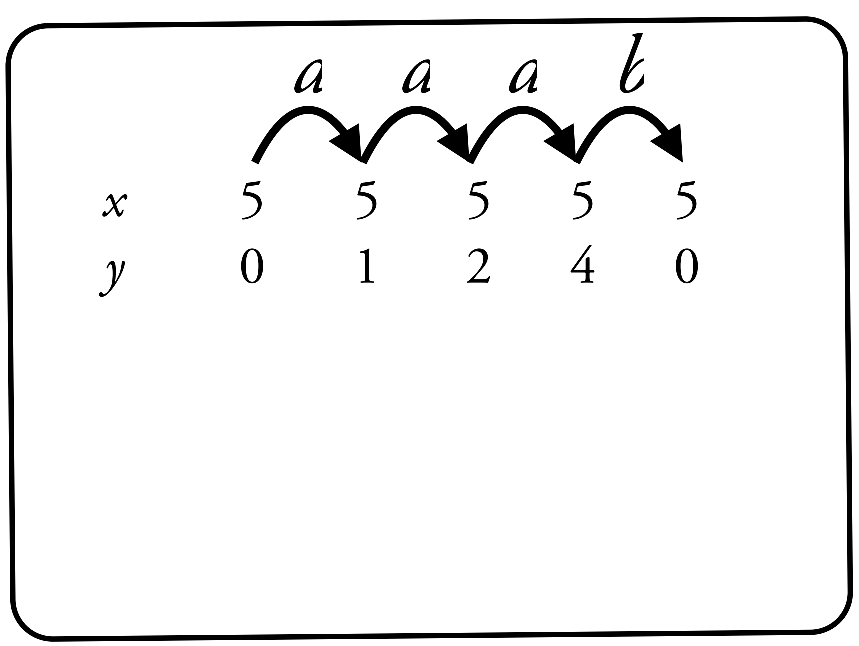
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

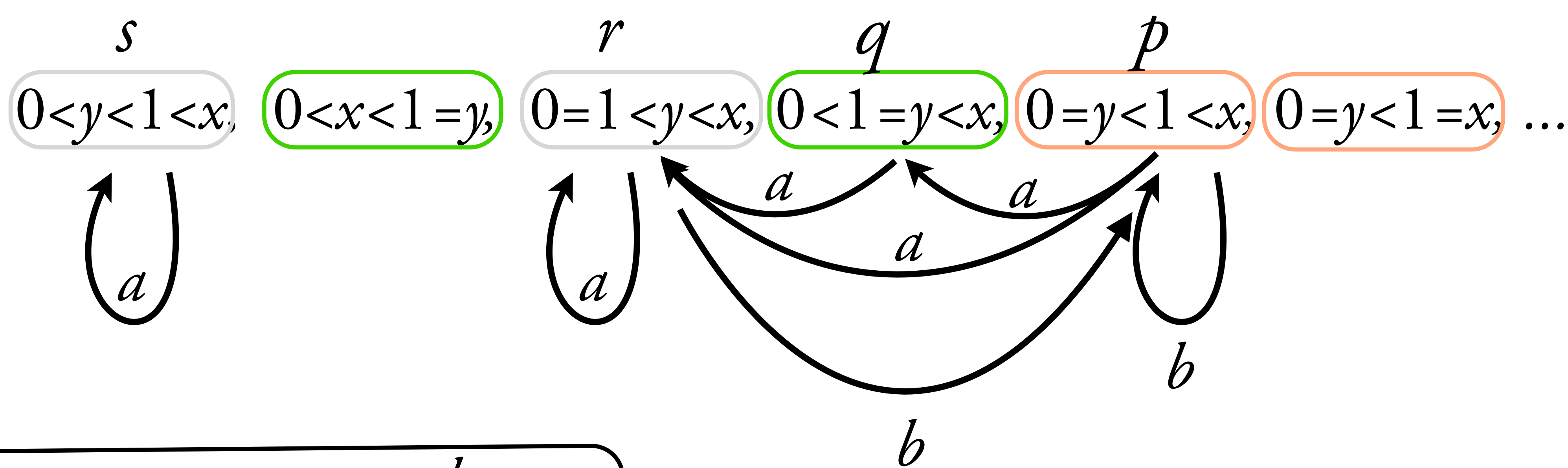
- $D = \mathbb{N}$
- finite words
- no database

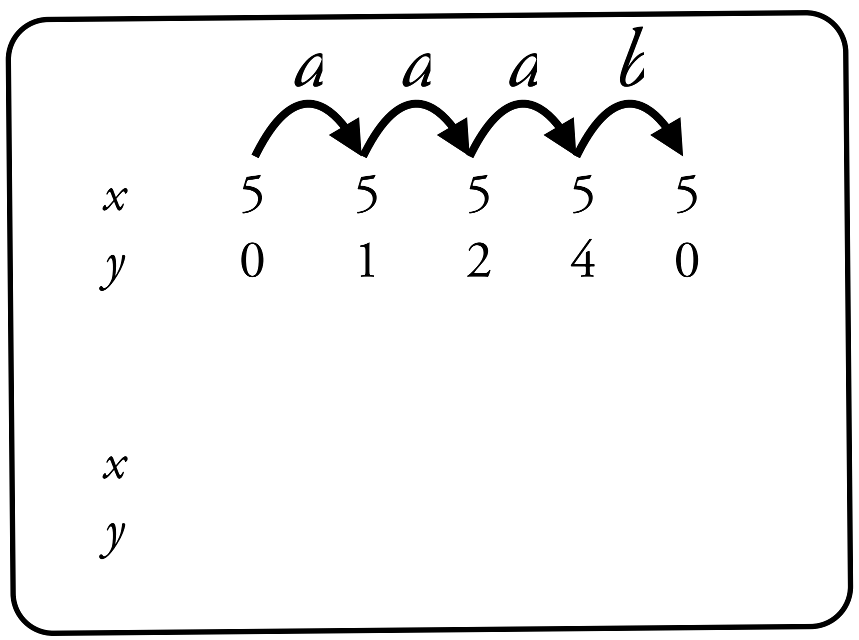
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

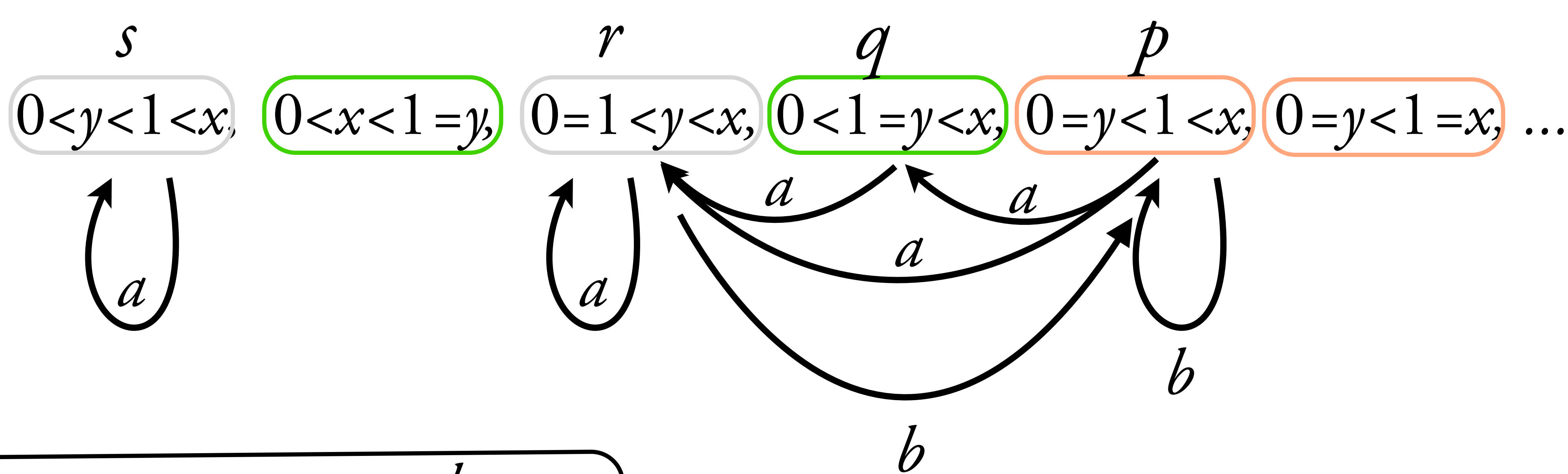
- $D = \mathbb{N}$
- finite words
- no database

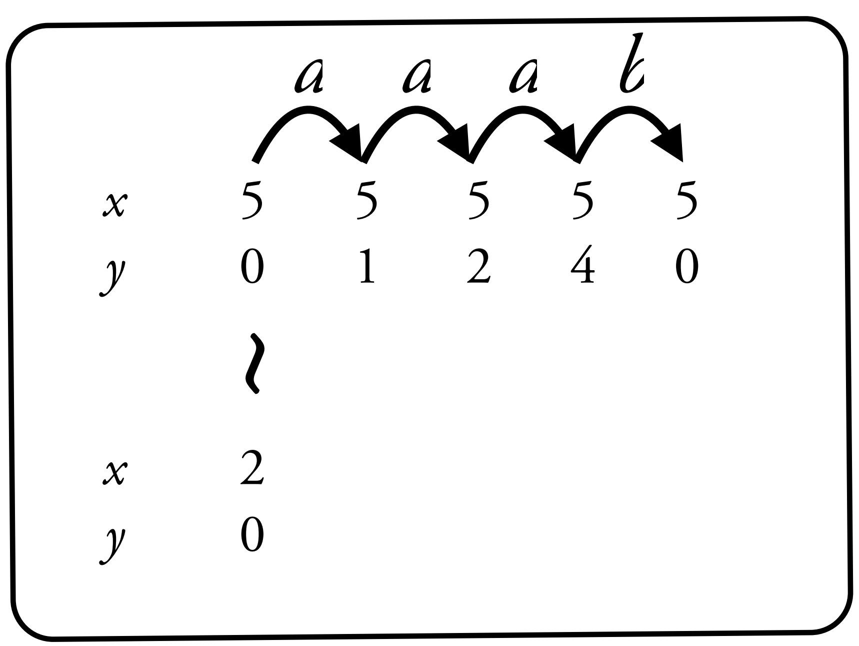
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathscr{A}$

Discrete case

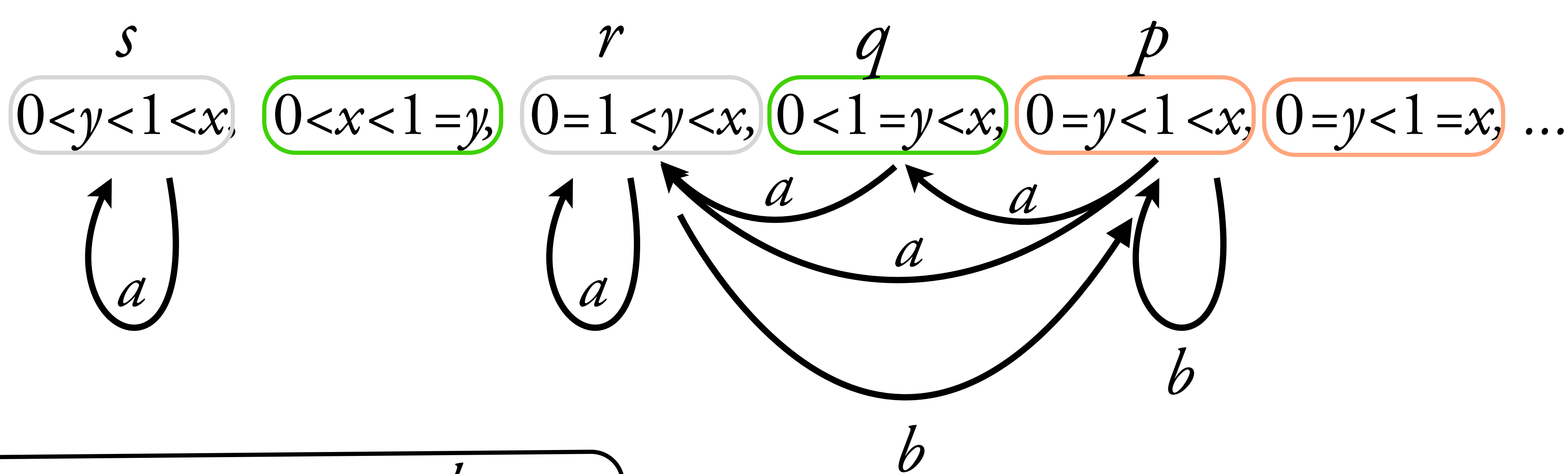
- $D = \mathbb{N}$
- finite words
- no database

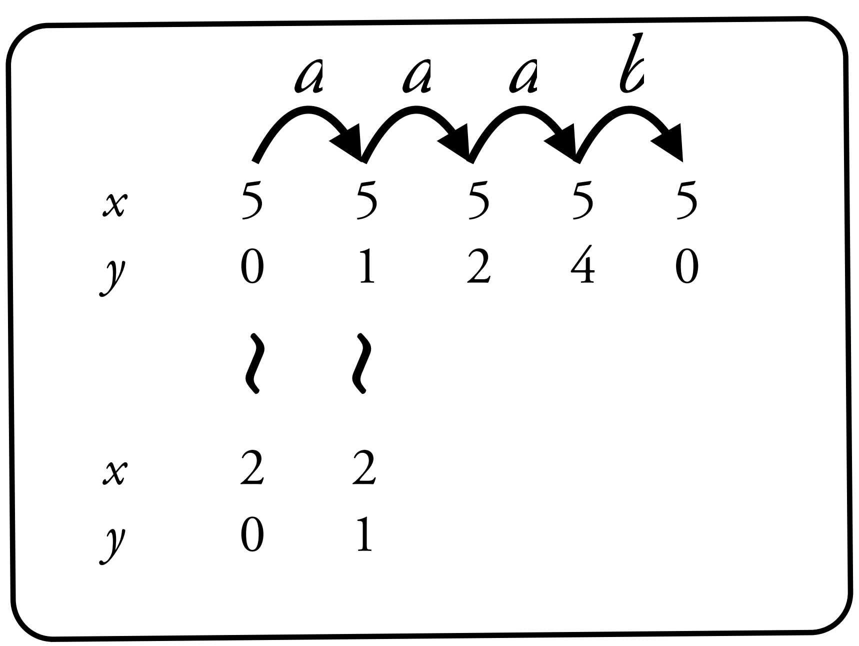
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

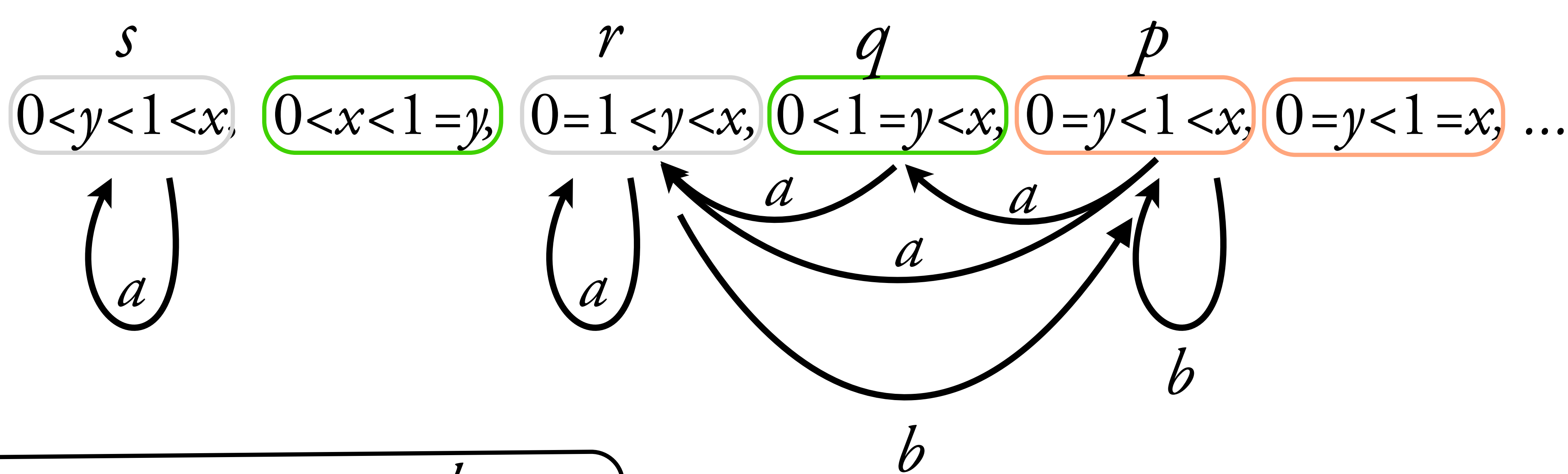
- $D = \mathbb{N}$
- finite words
- no database

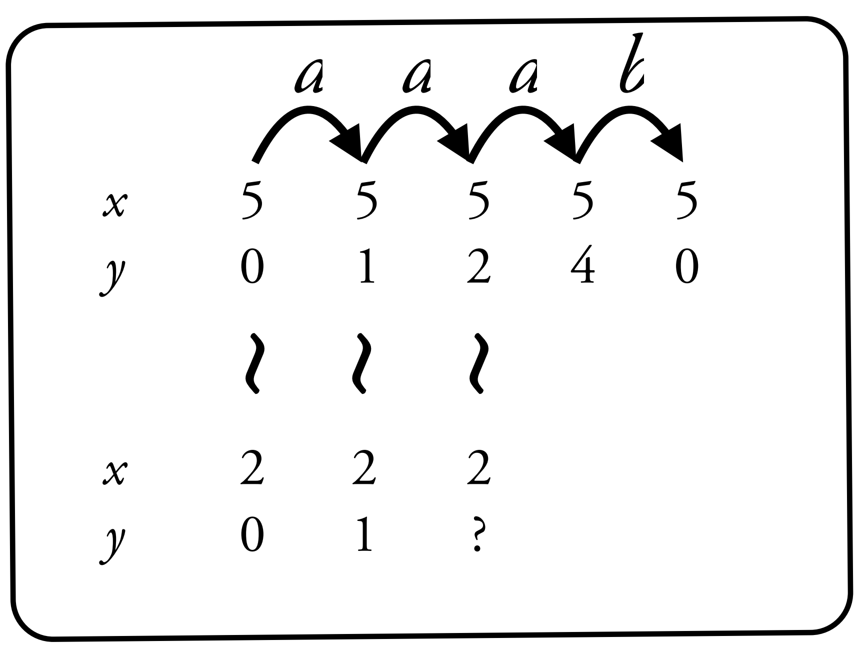
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





~~bisimulation~~ → runs of the region automaton correspond to runs of $\mathscr{A}$

Discrete case

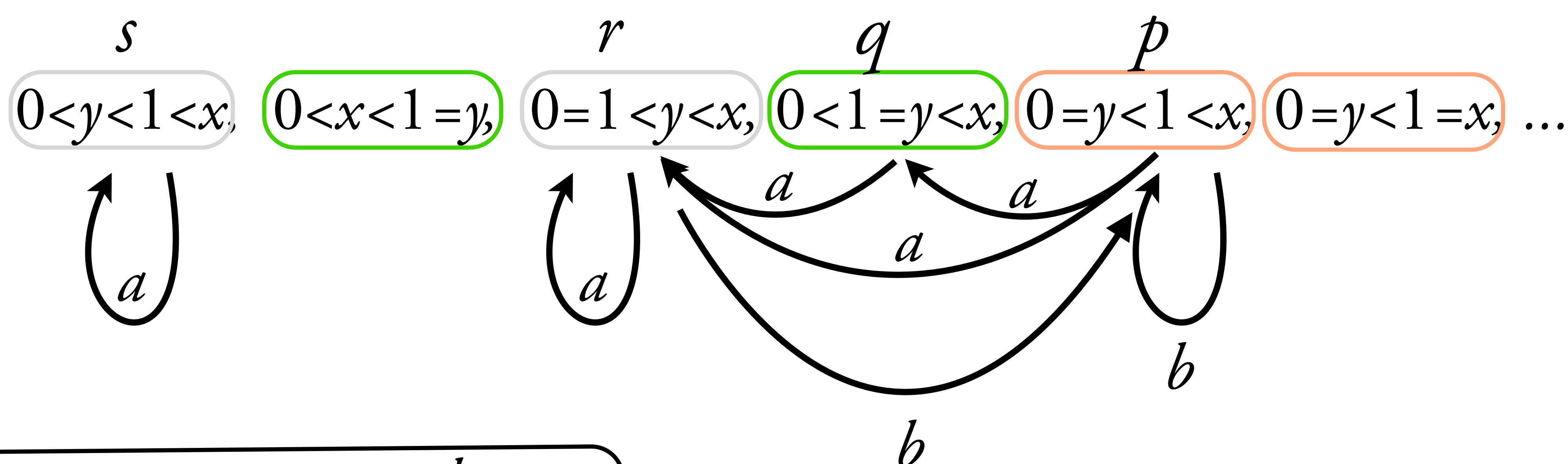
- $D = \mathbb{N}$
- finite words
- no database

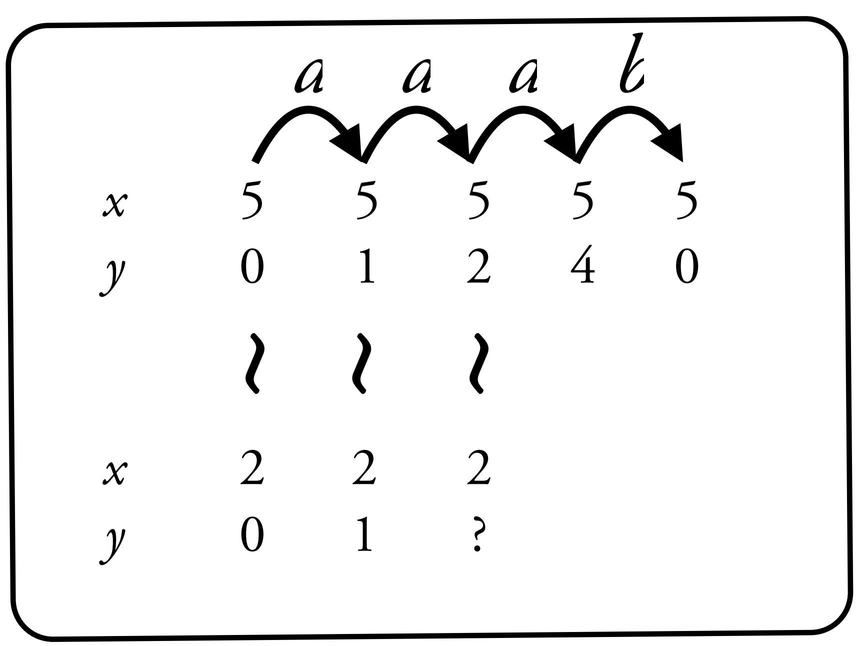
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=1)$





but: in any cell and any n we can find configurations which are *sufficiently good* for all runs of length $\leq n$

→ *finite* runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

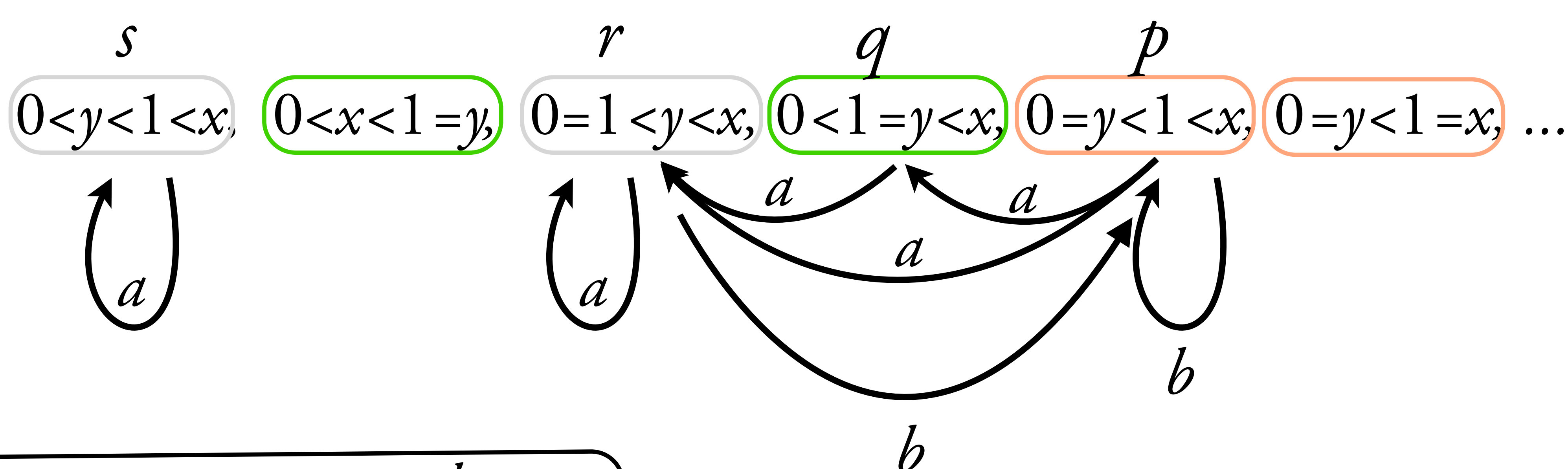
- $D = \mathbb{N}$
- finite words
- no database

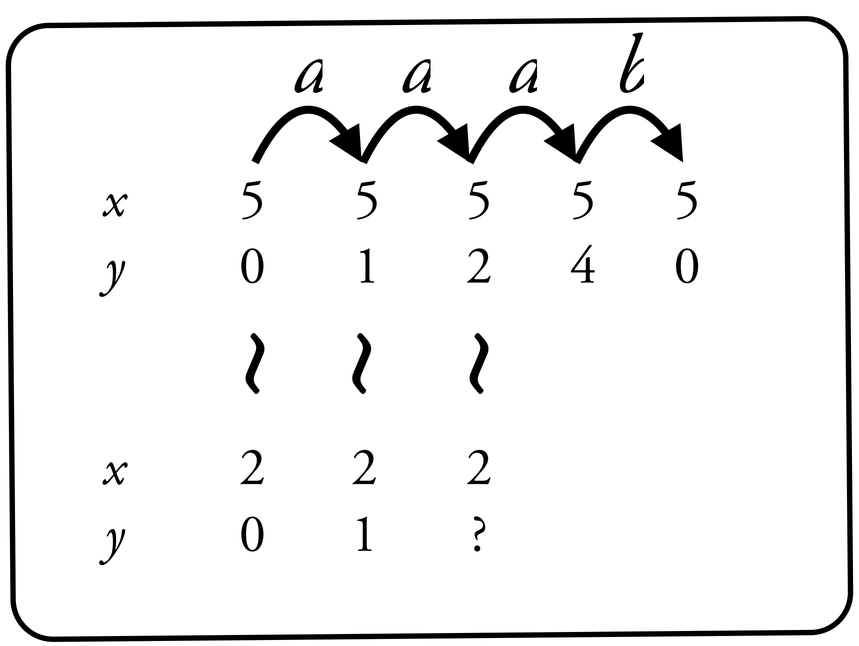
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$





but: in any cell and any n we can find configurations which are *sufficiently good* for all runs of length $\leq n$

→ *finite* runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

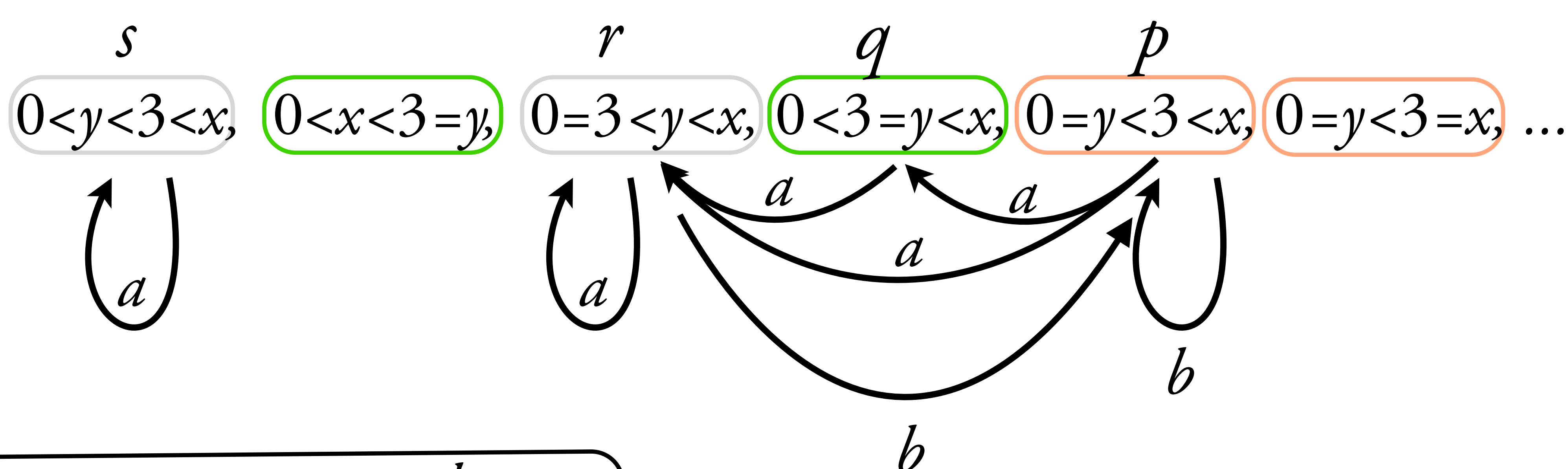
- $D = \mathbb{N}$
- finite words
- no database

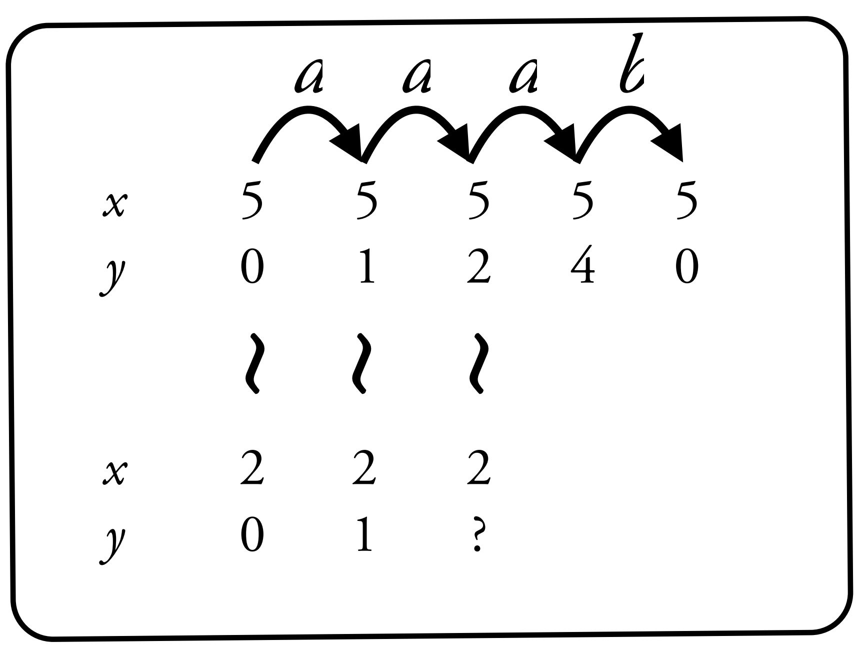
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$





but: in any cell and any n we can find configurations which are *sufficiently good* for all runs of length $\leq n$

→ *finite* runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

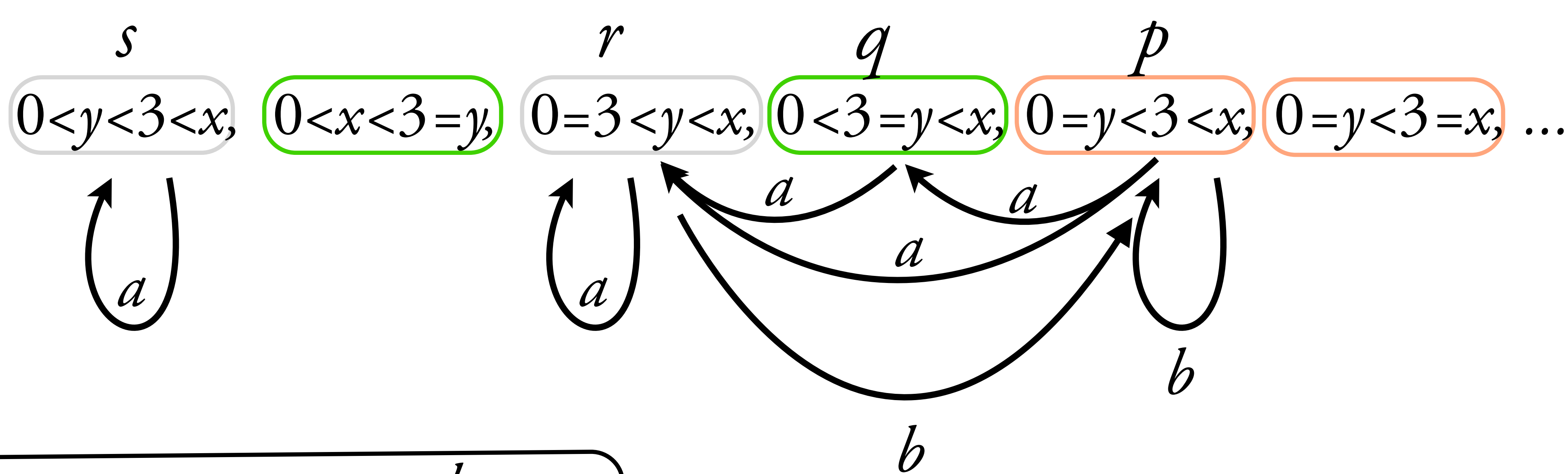
- $D = \mathbb{N}$
- finite words
- no database

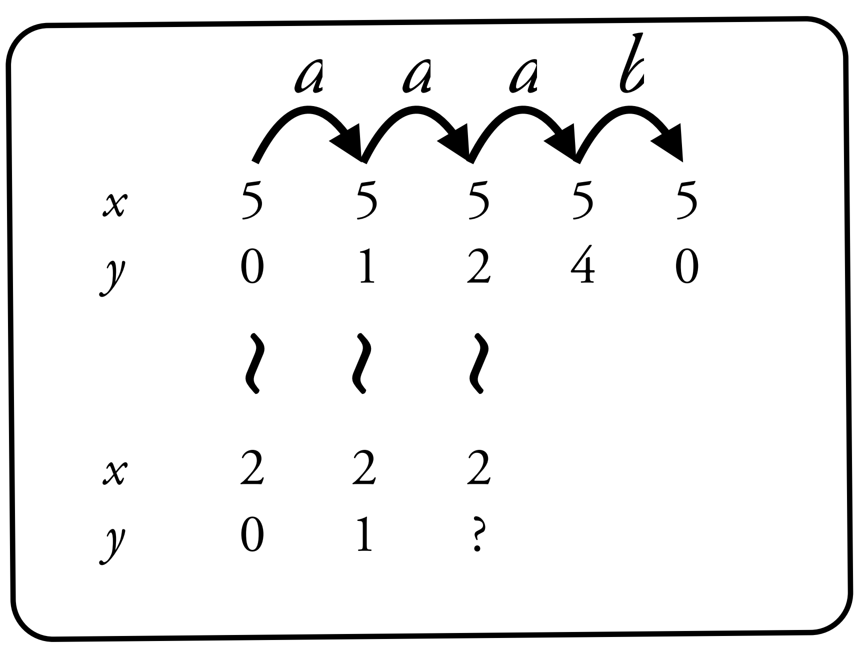
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$





but: in any cell and any n we can find configurations which are *sufficiently good* for all runs of length $\leq n$

→ *finite* runs of the region automaton correspond to runs of $\mathcal{A}$

Discrete case

- $D = \mathbb{N}$
- finite words
- no database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$

Solution: consider 0,1,2,3 as special values (constants)

Discrete case

- $D = \mathbb{N}$
- finite words
- no database

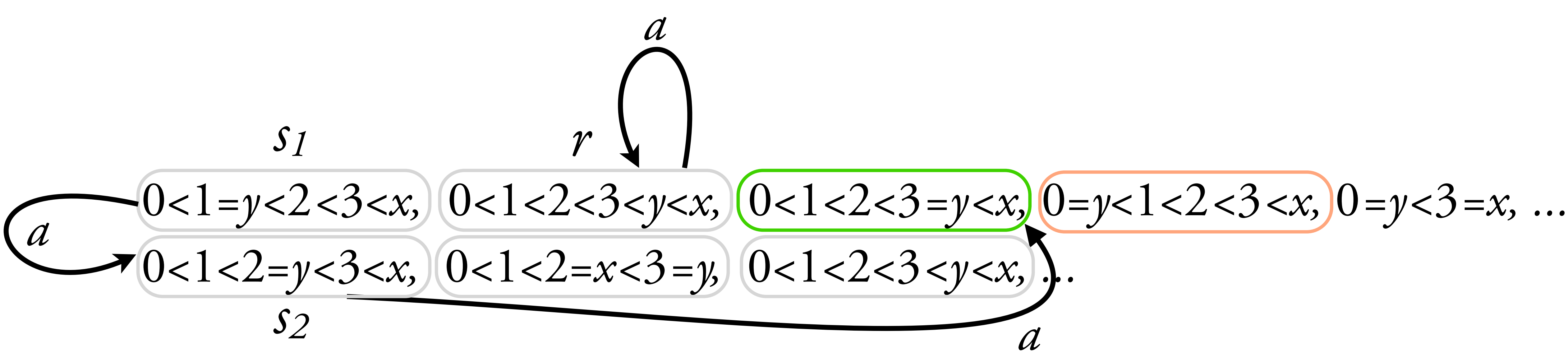
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$

Solution: consider 0,1,2,3 as special values (constants)



Discrete case

- $D = \mathbb{N}$
- finite words
- no database

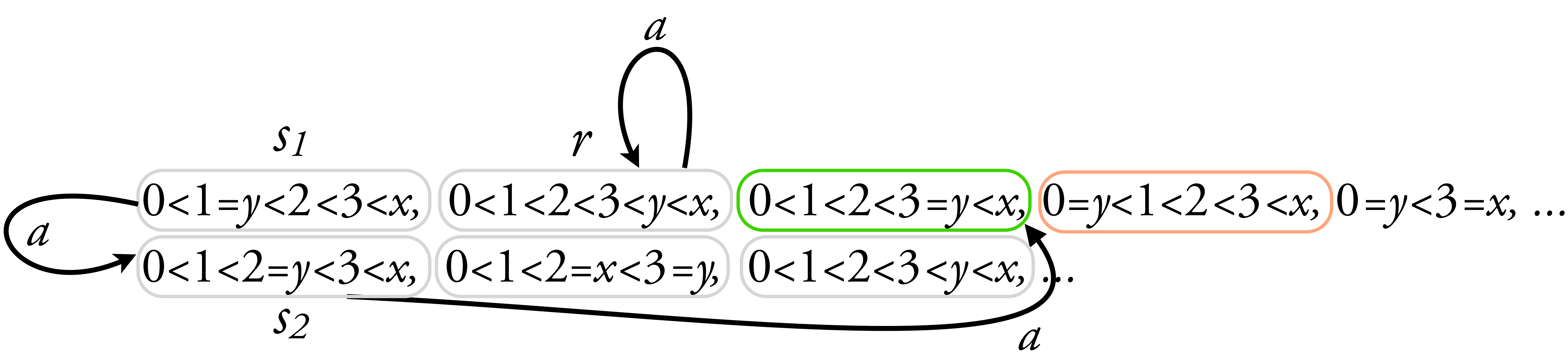
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$

Solution: consider 0,1,2,3 as special values (constants)



Then the obtained cell automaton simulates $\mathcal{A}$ fatefully

Discrete case

- $D = \mathbb{N}$
- finite words
- no database

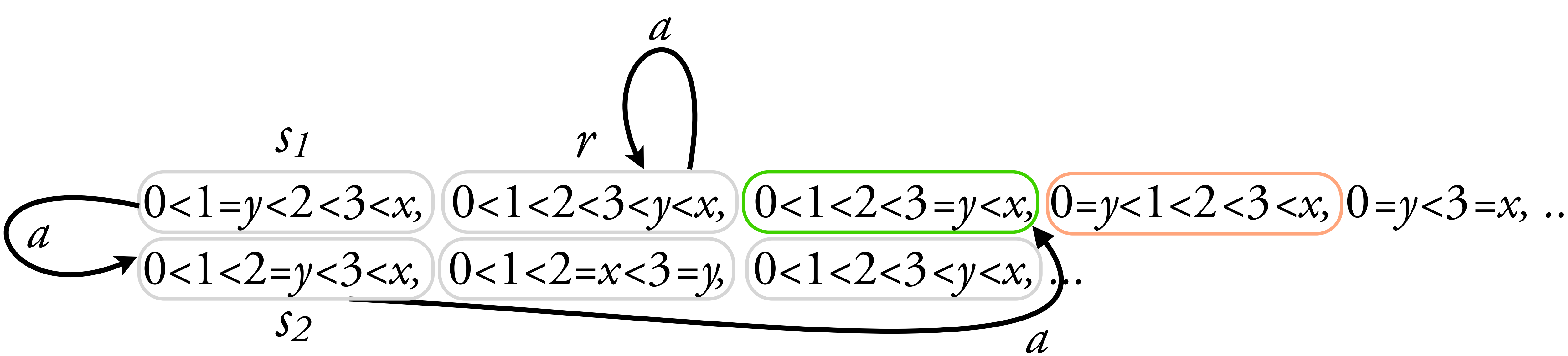
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x)$

δ_b: $(x'=x) \wedge (y'=0)$

τ_F: $(y=3)$

Solution: consider 0,1,2,3 as special values (constants)



Then the obtained cell automaton simulates $\mathcal{A}$ fatefully

Theorem. For any linearly ordered structure there is a finite number of special values

This solves the case of finite runs, with no database.

This solves the case of finite runs, with no database.

Theorem. For all $\mathscr{D}$, $\mathscr{D}$-automata over finite words accept regular languages.

Theorem. Emptiness of $\mathscr{D}$-automata is decidable* in PSPACE

**For any reasonable linearly ordered structure* - there should be an efficient algorithm which can compute the special values, given some initial constants

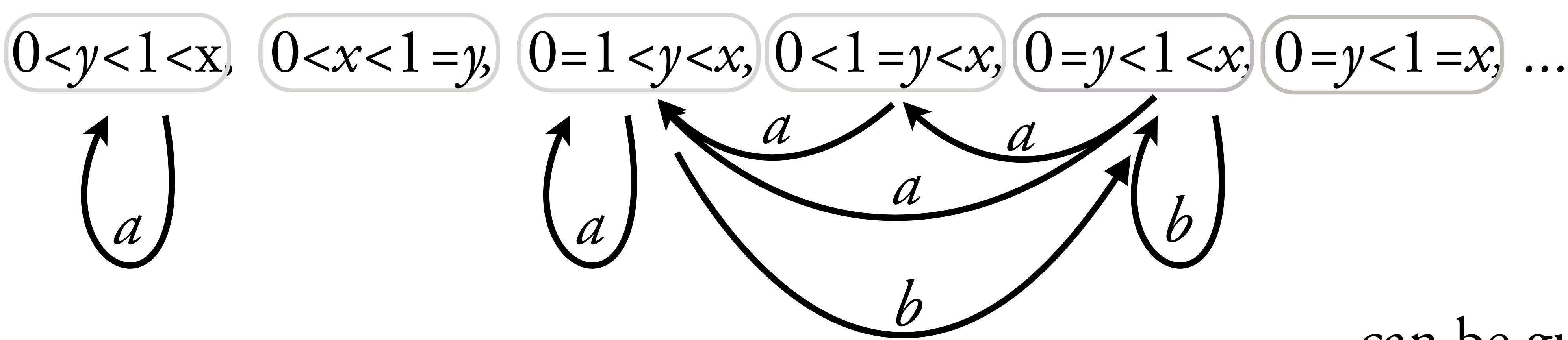
Adding a database

τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$



can be guaranteed to be new

is forced to be zero

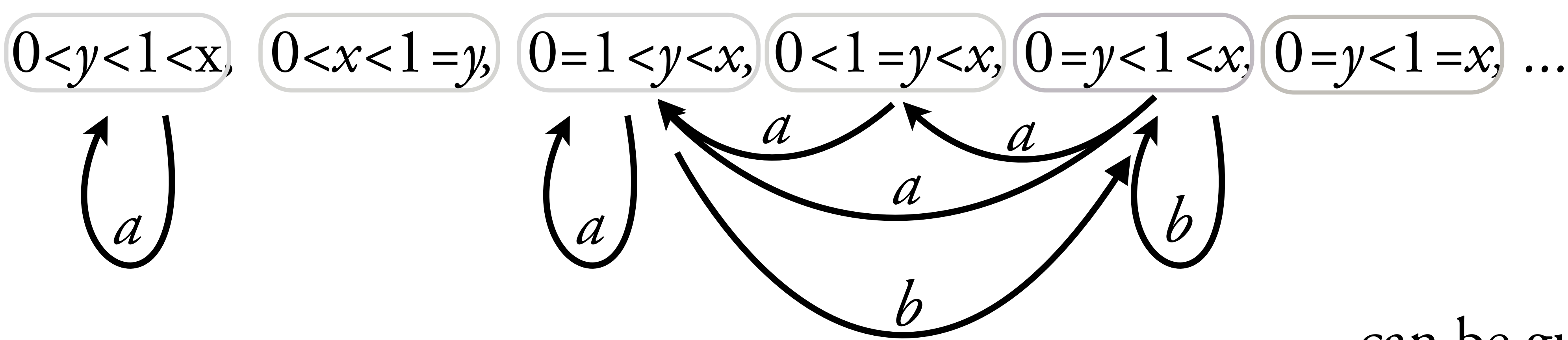
Adding a database

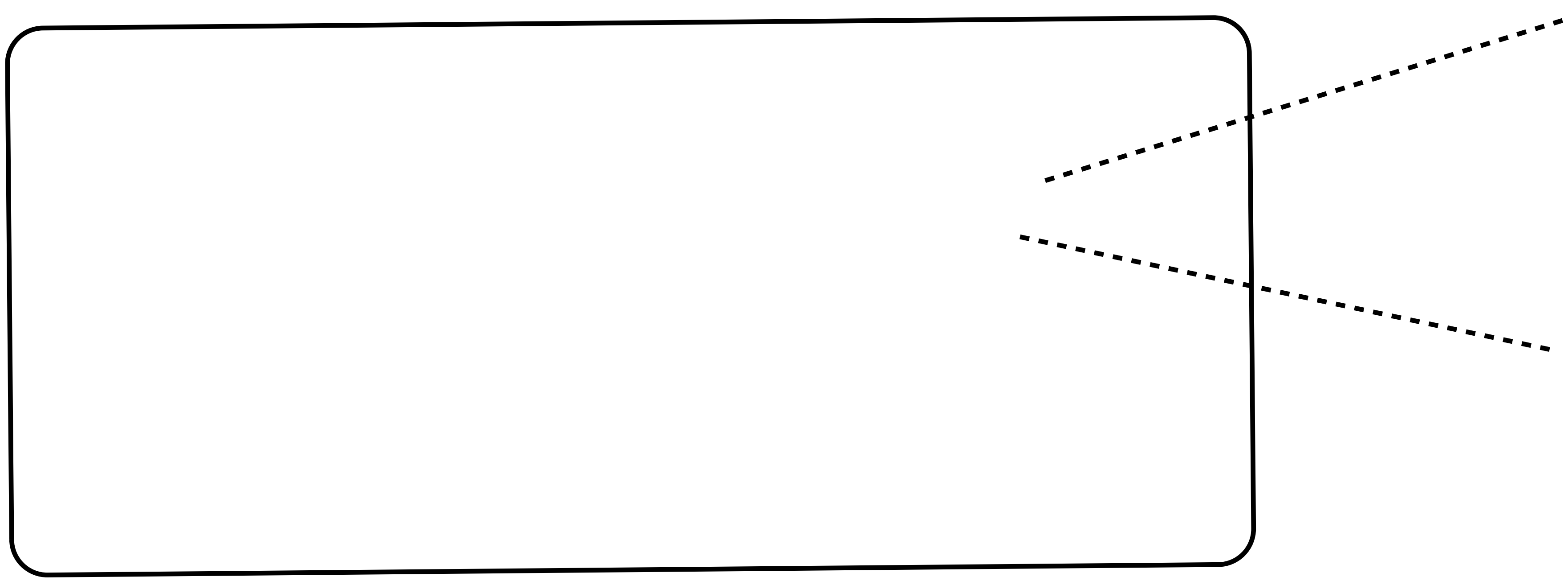
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$





can be guaranteed to be new

is forced to be zero

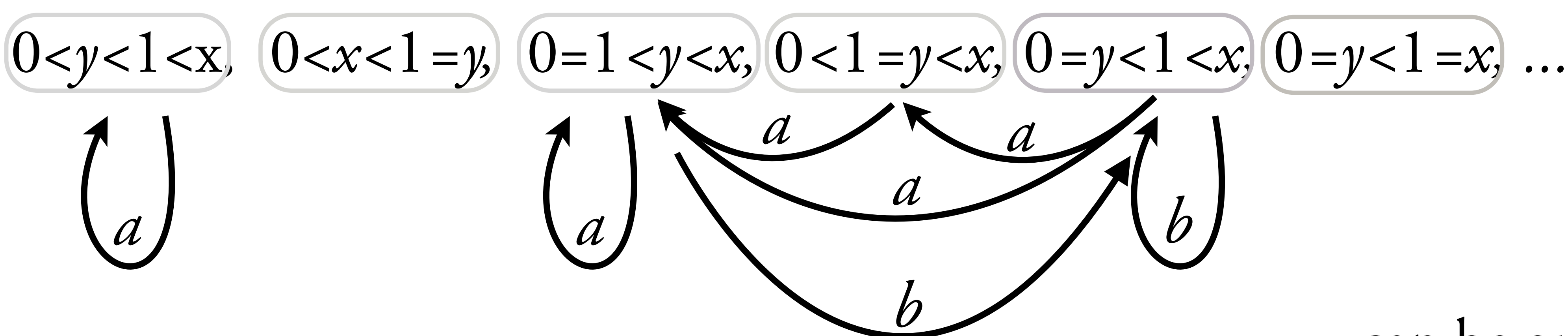
Adding a database

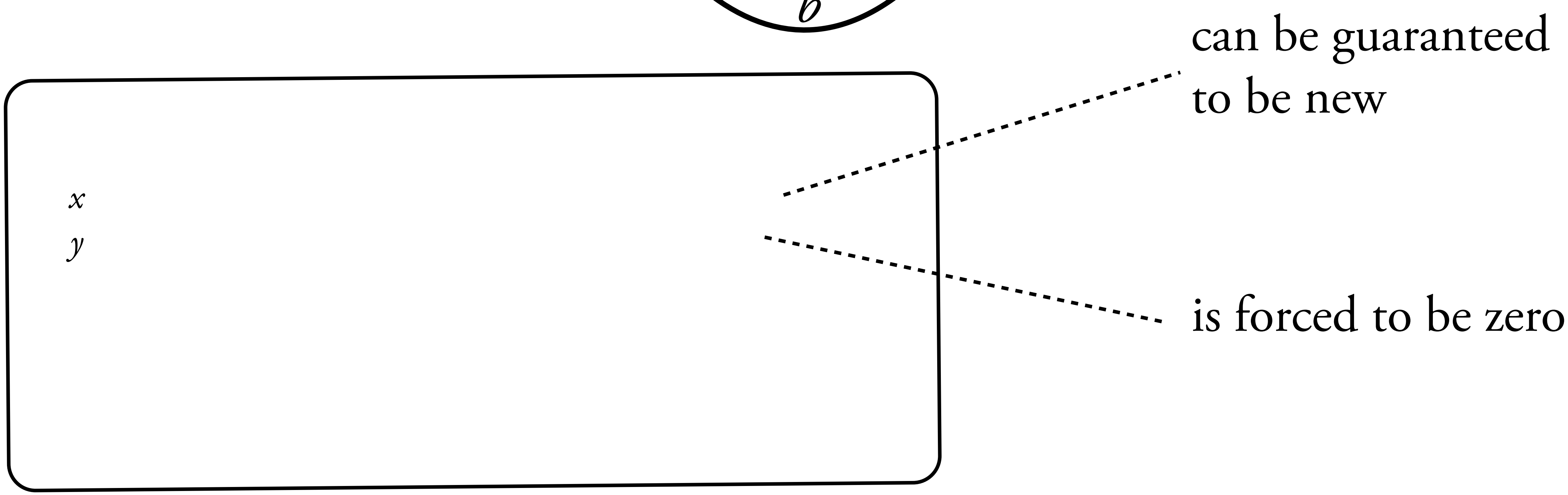
τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$



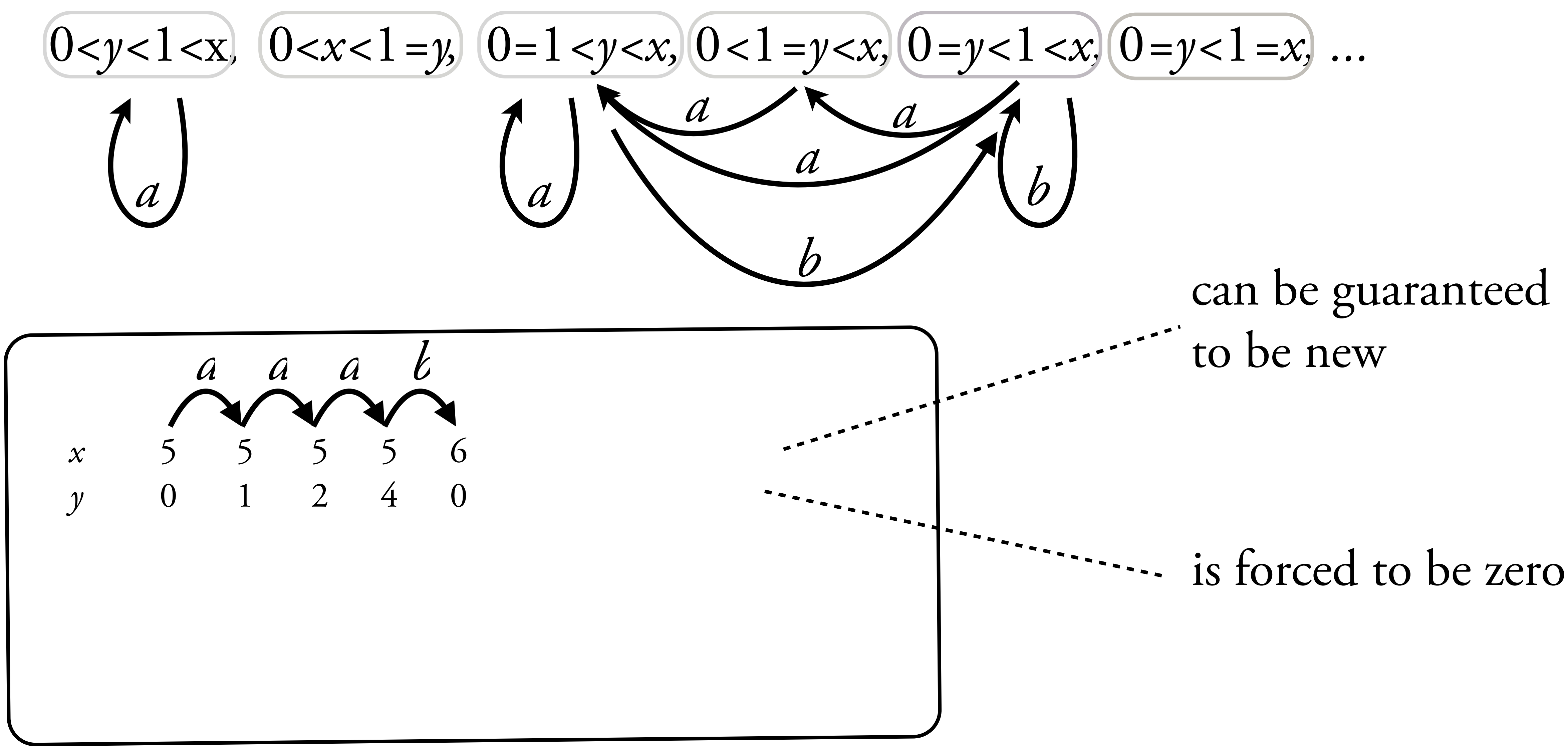


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

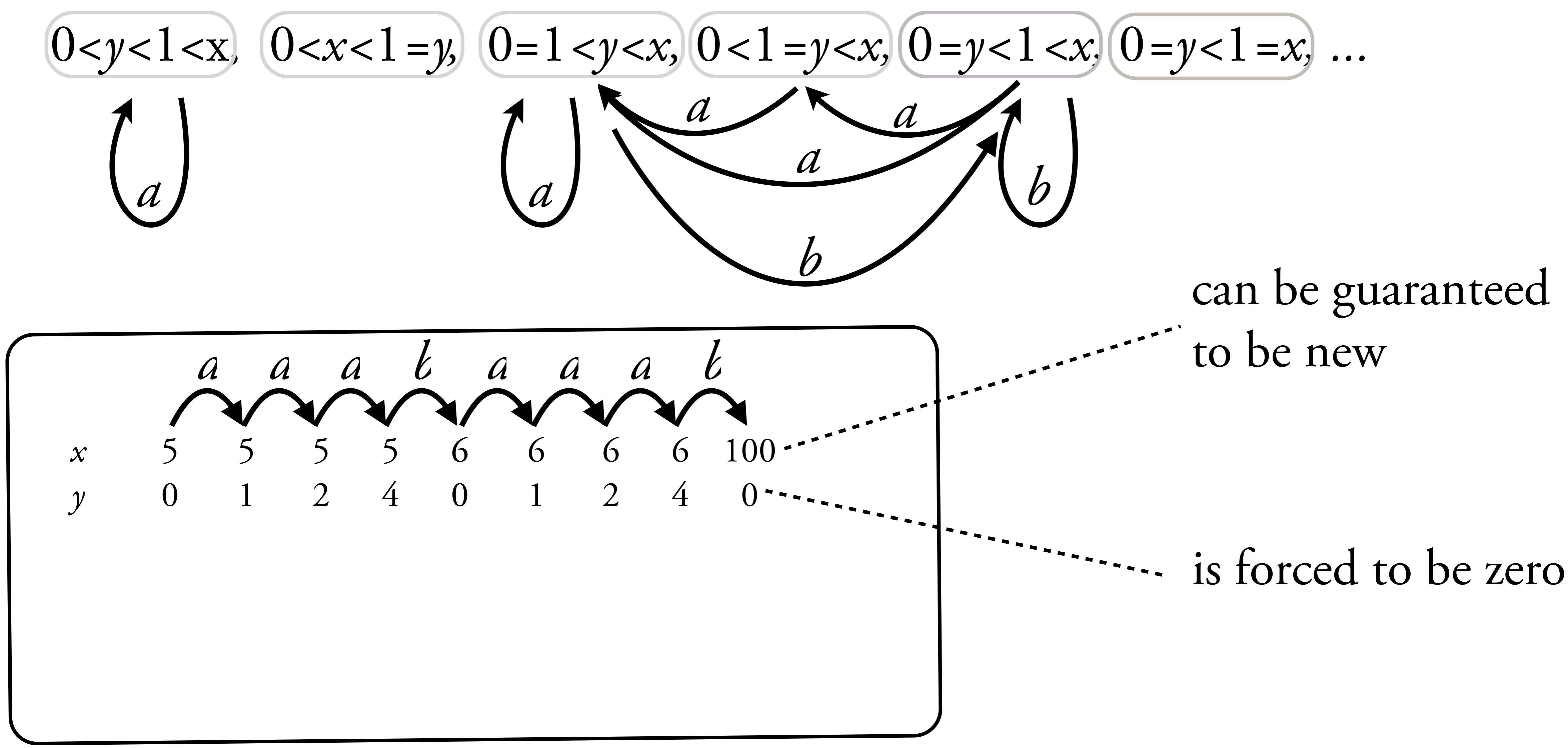


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

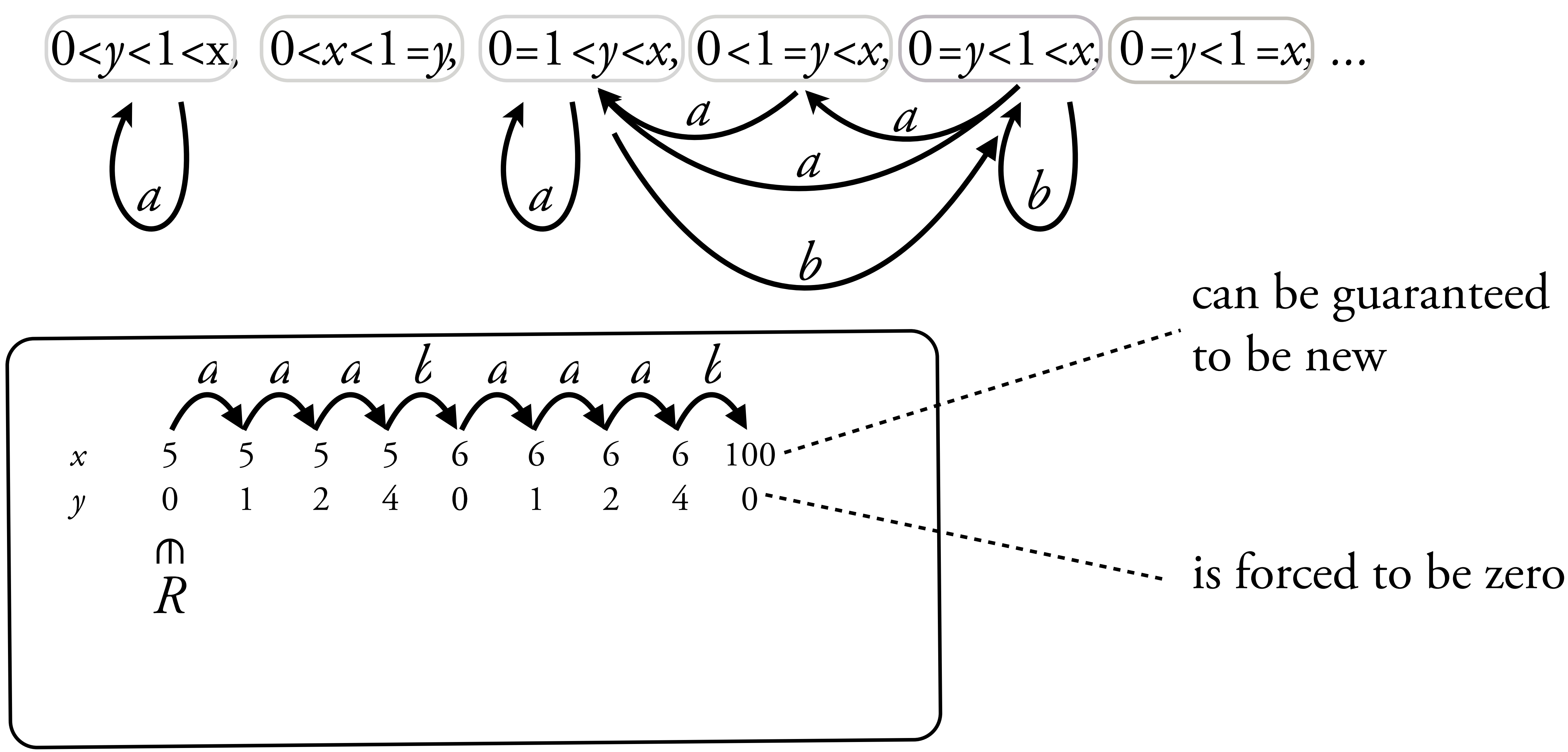


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

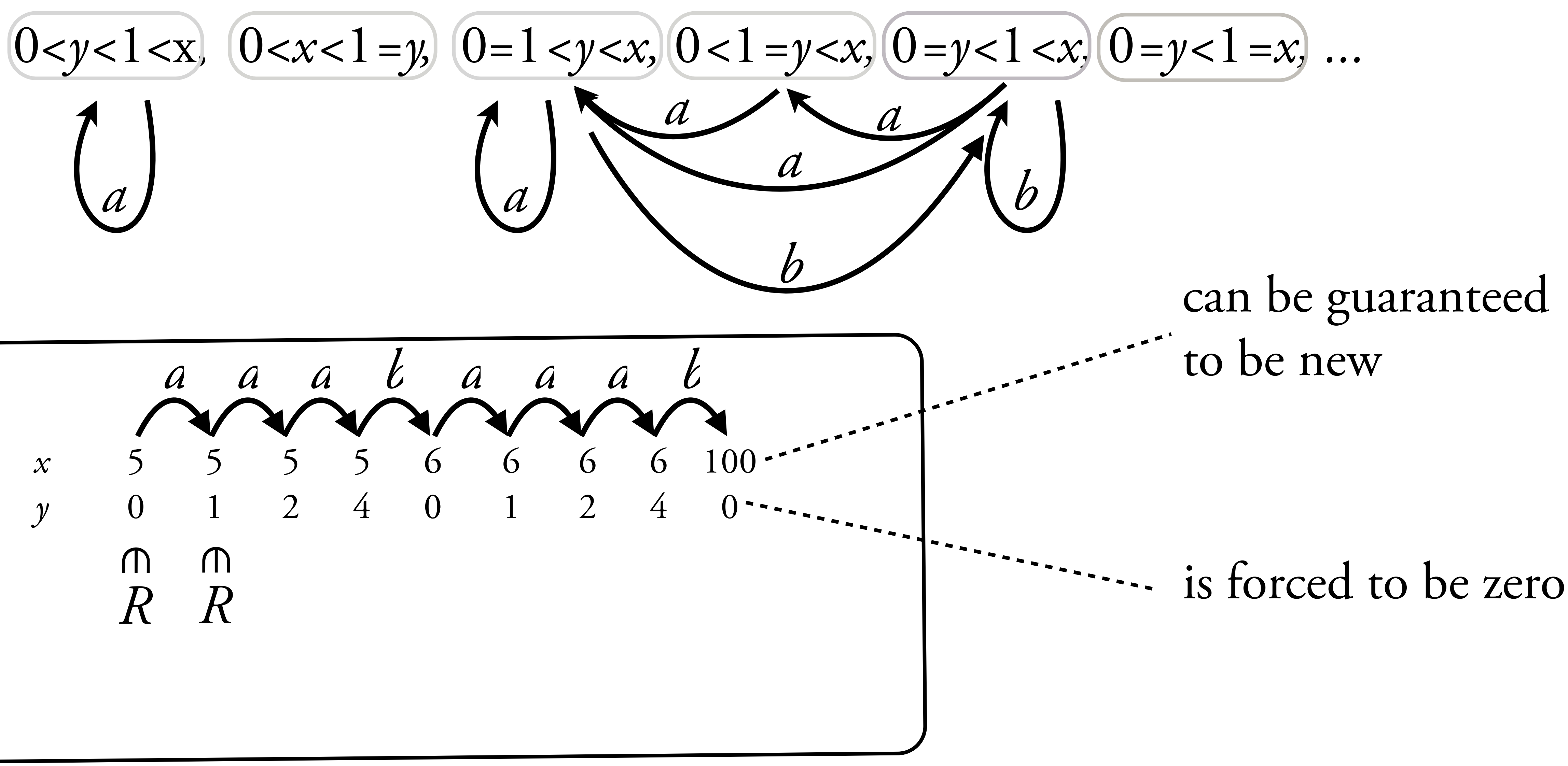


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

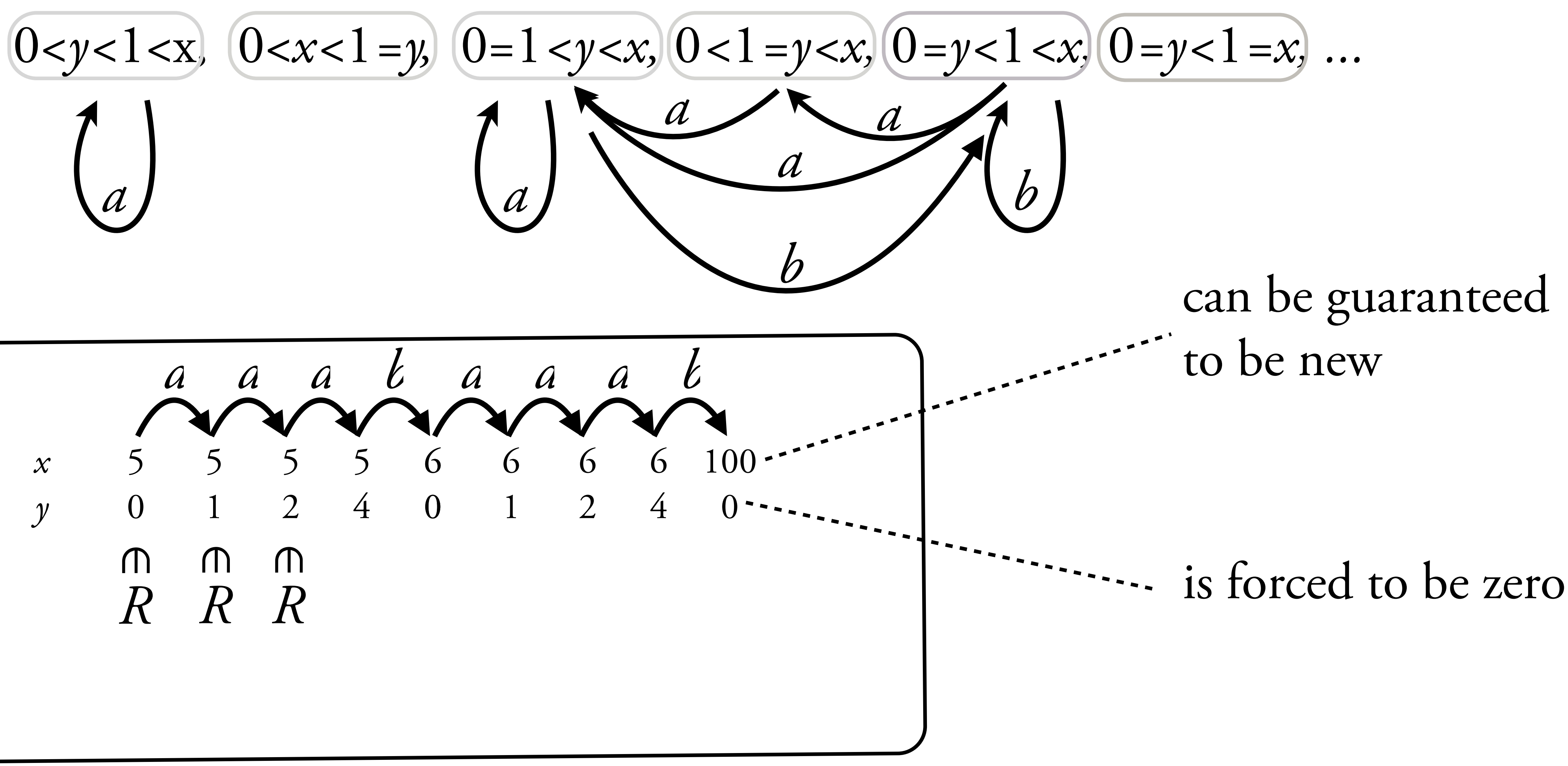


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

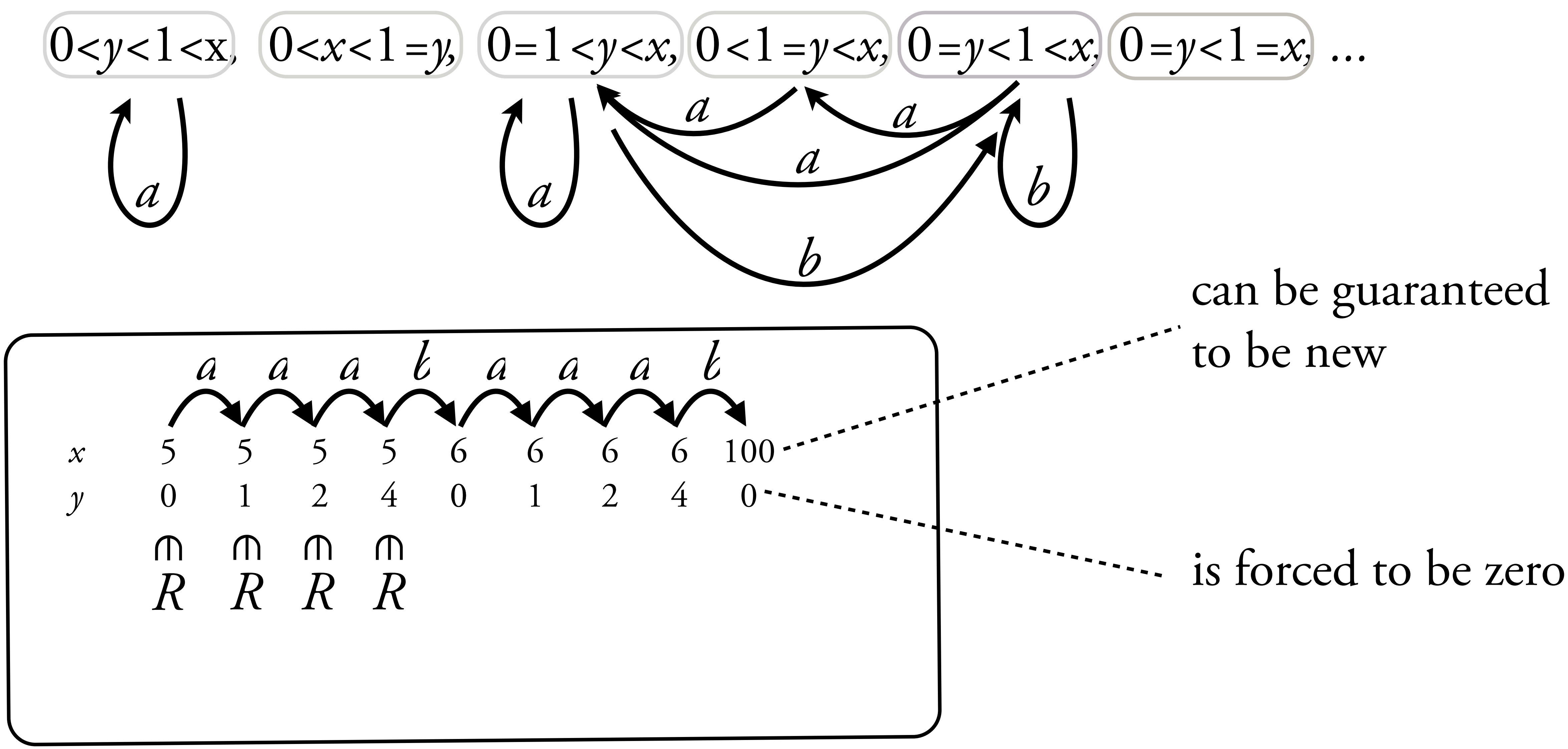


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

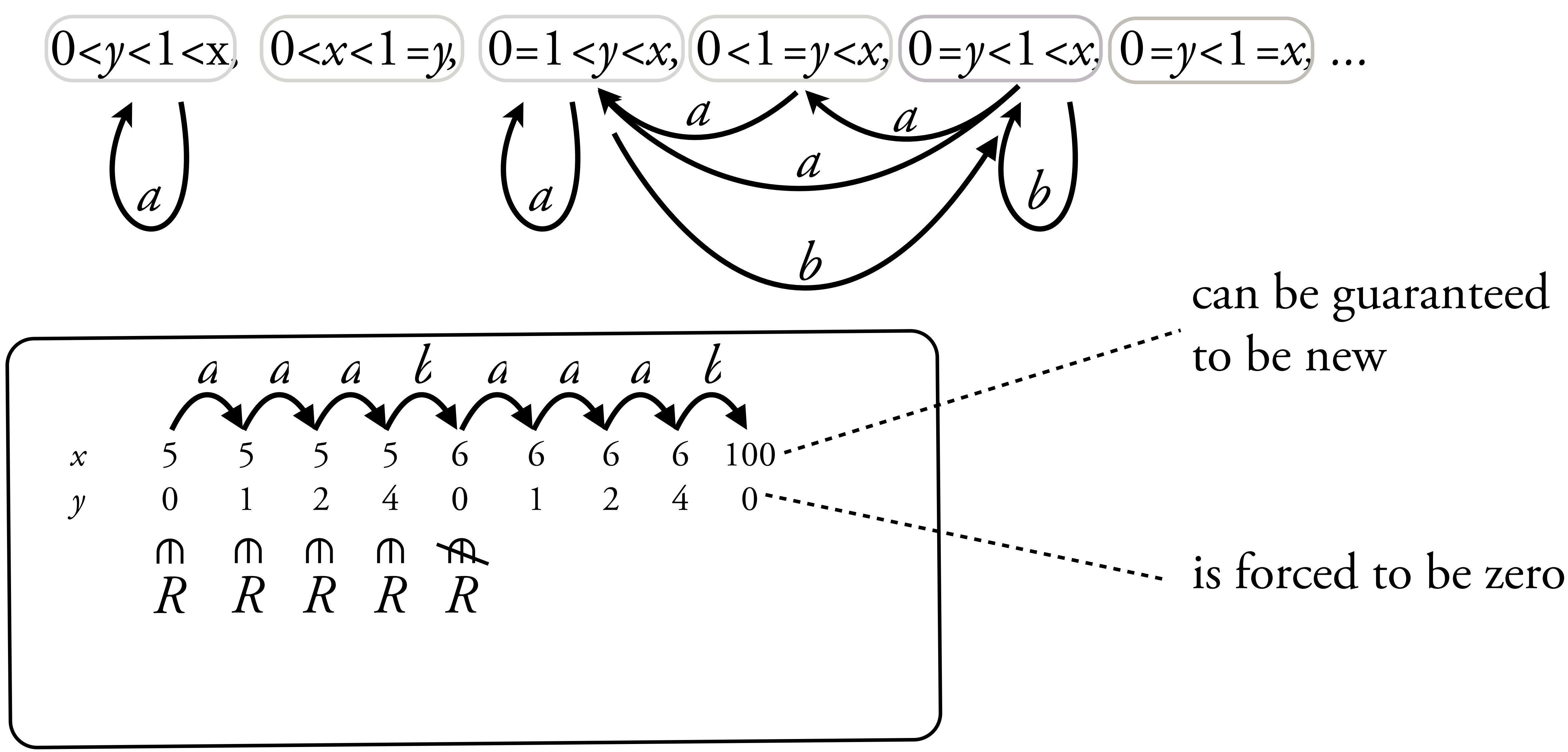


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

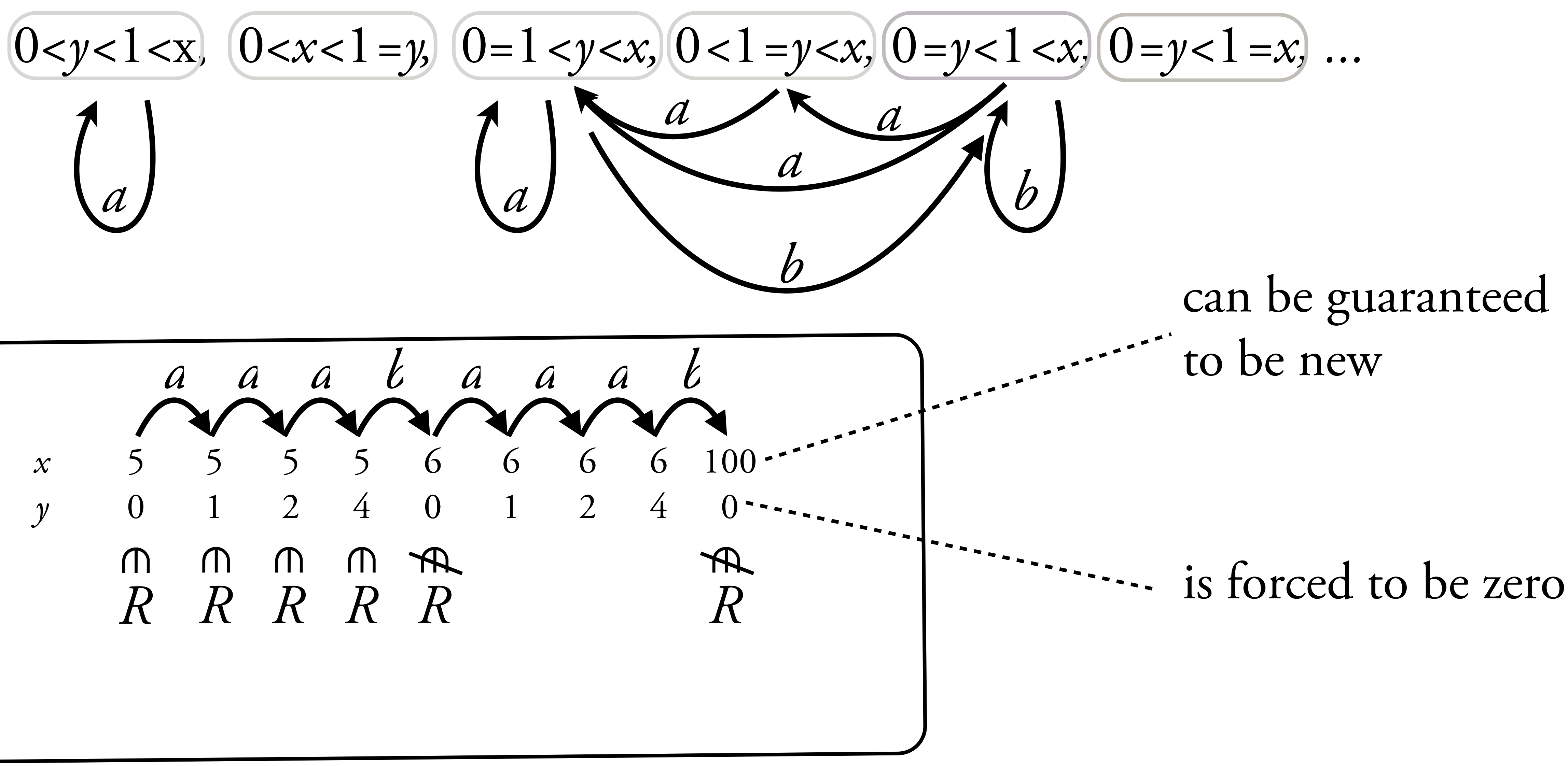


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

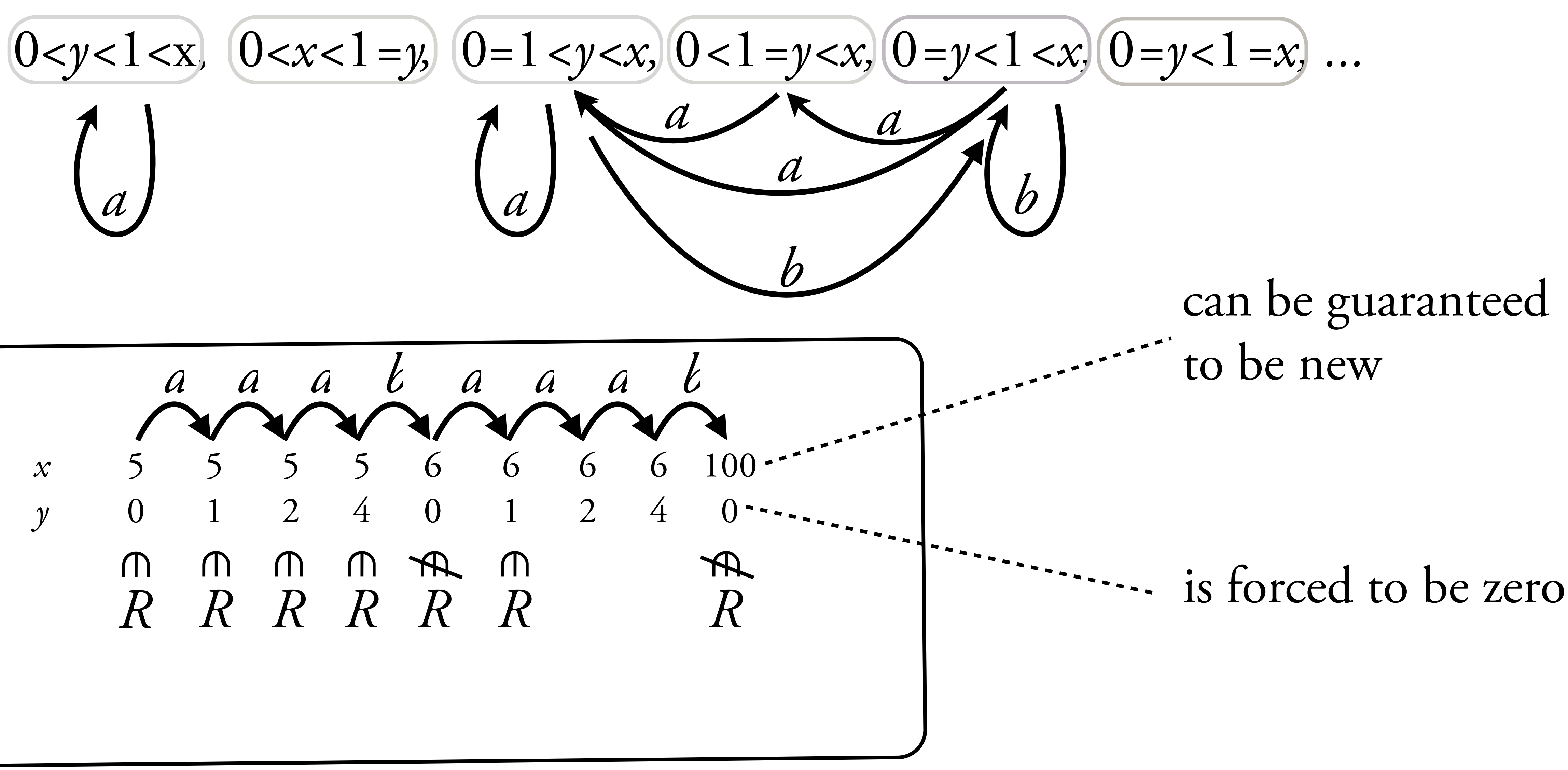


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

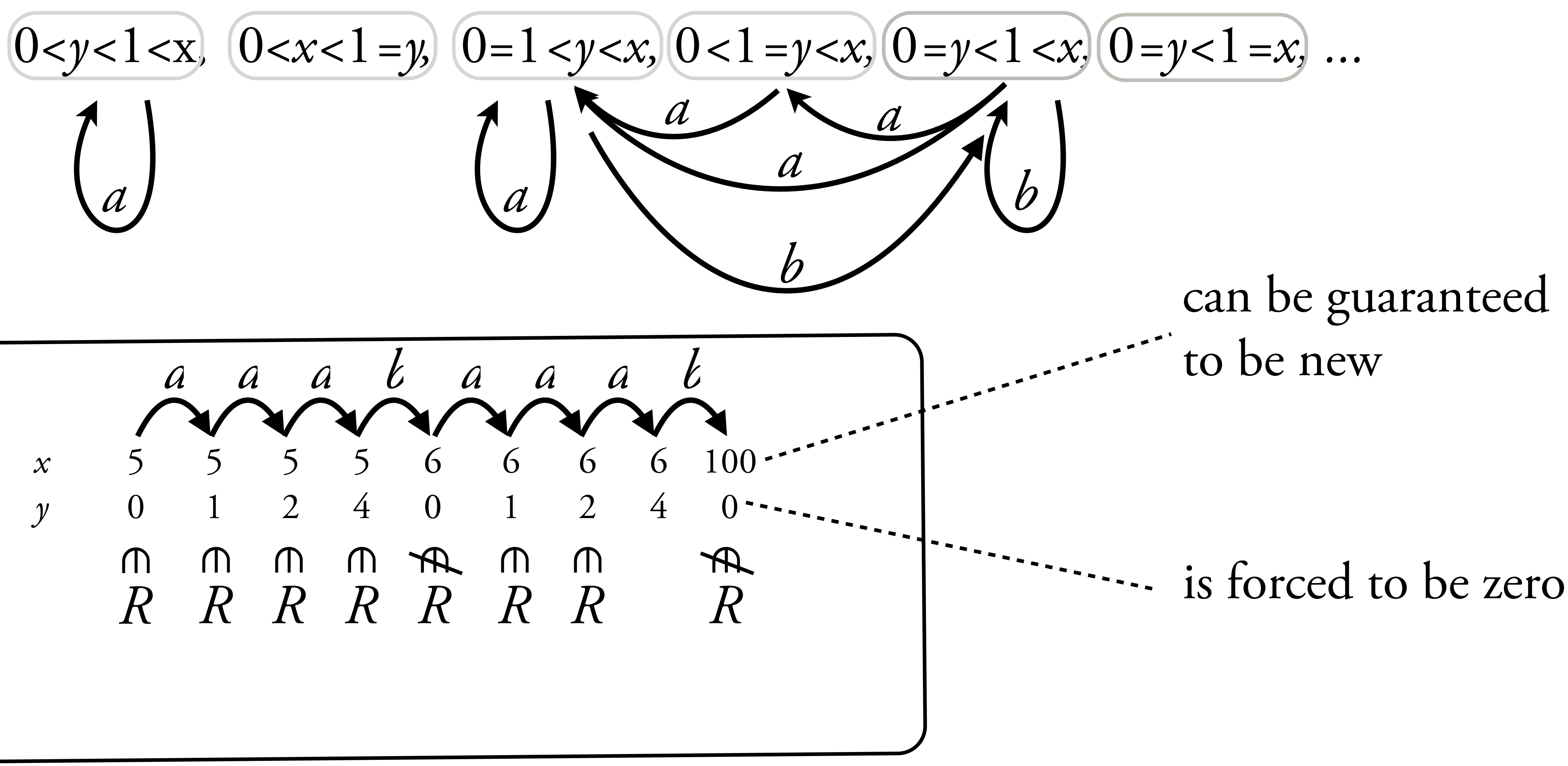


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

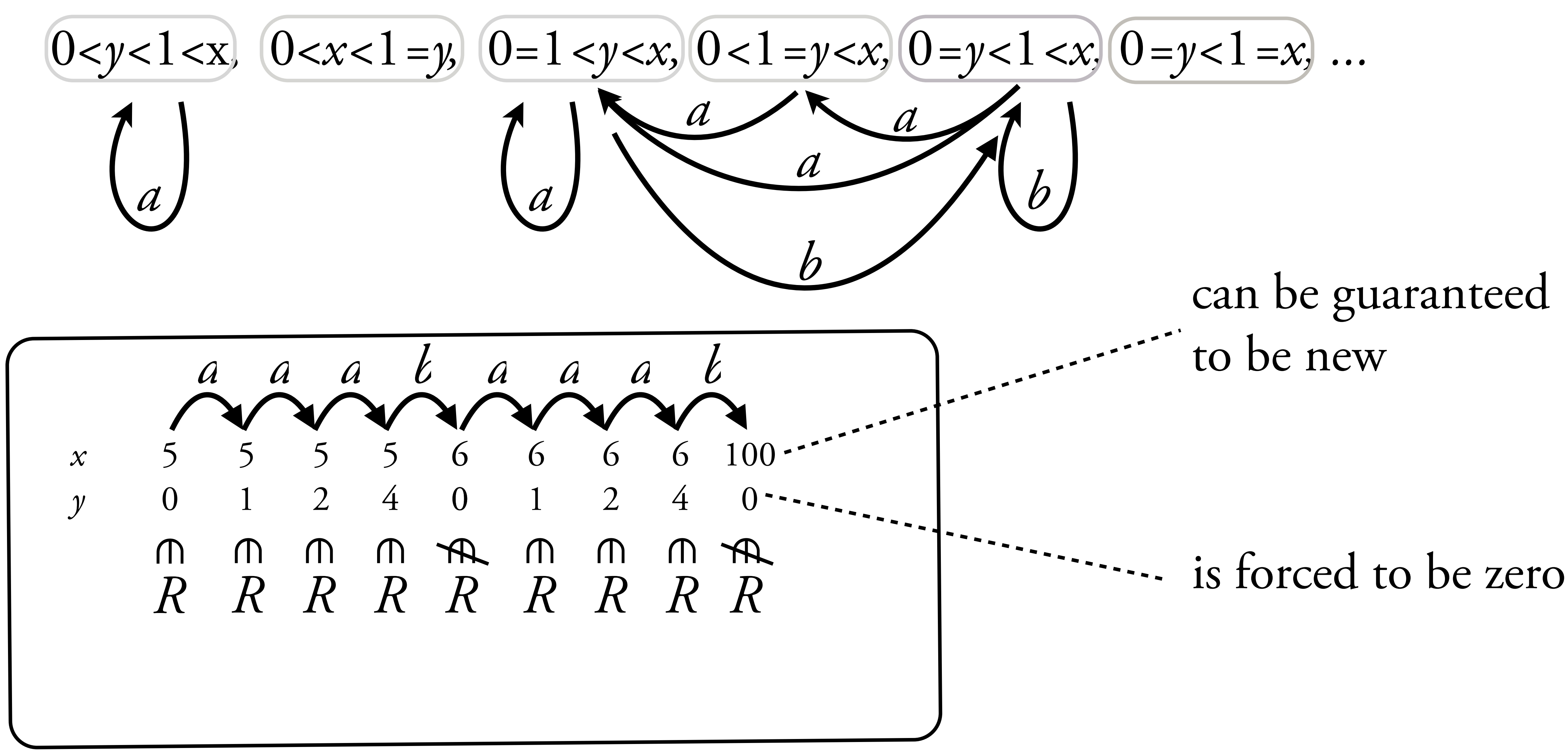


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$

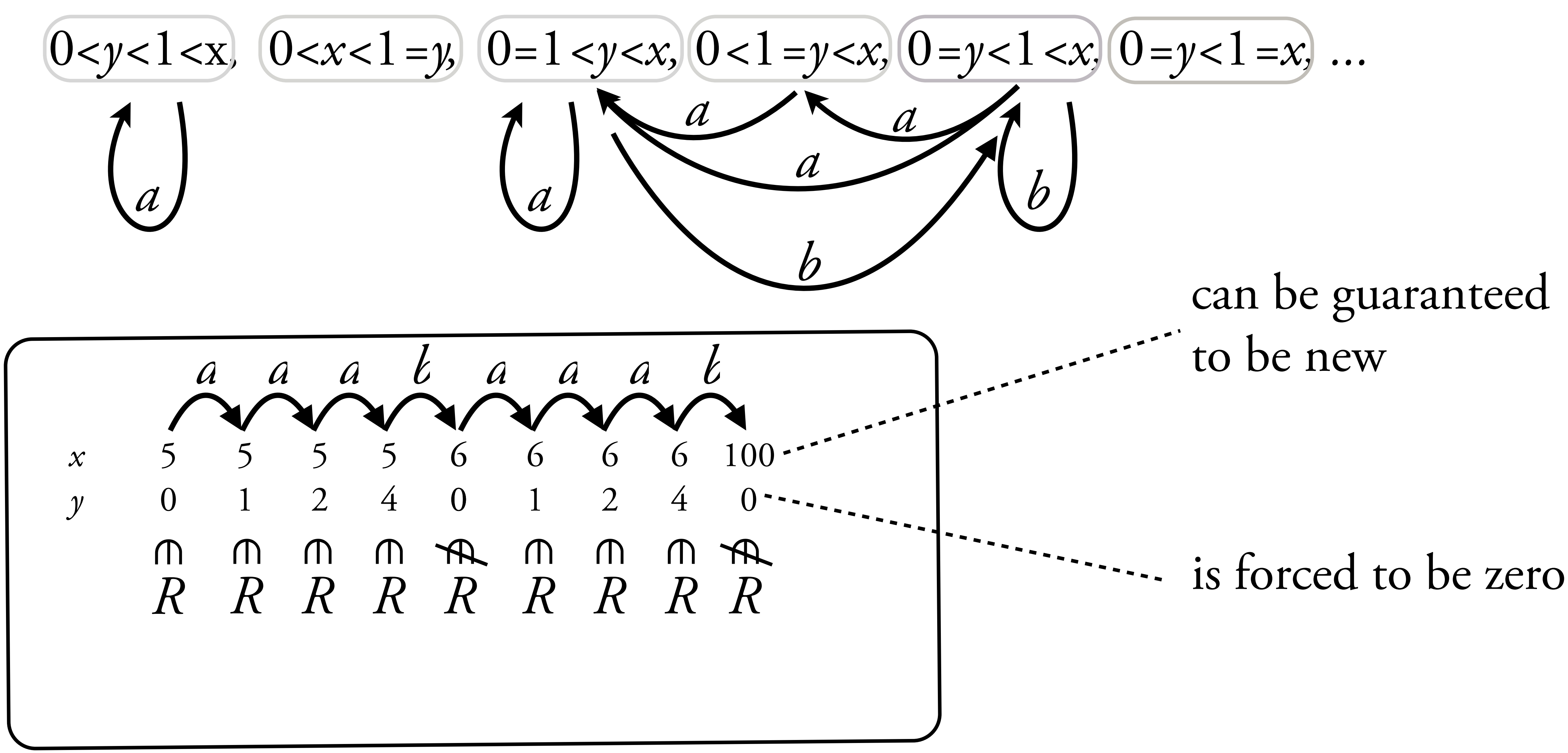


$$\tau_I: \quad (x>0) \wedge (y=0)$$

$$\delta_a: \quad (x'=x) \wedge (y<y'<x) \wedge R(x,y)$$

$$\delta_b: \quad (y'=0) \wedge \neg R(x,y)$$

$$\tau_F: \quad (y=1)$$

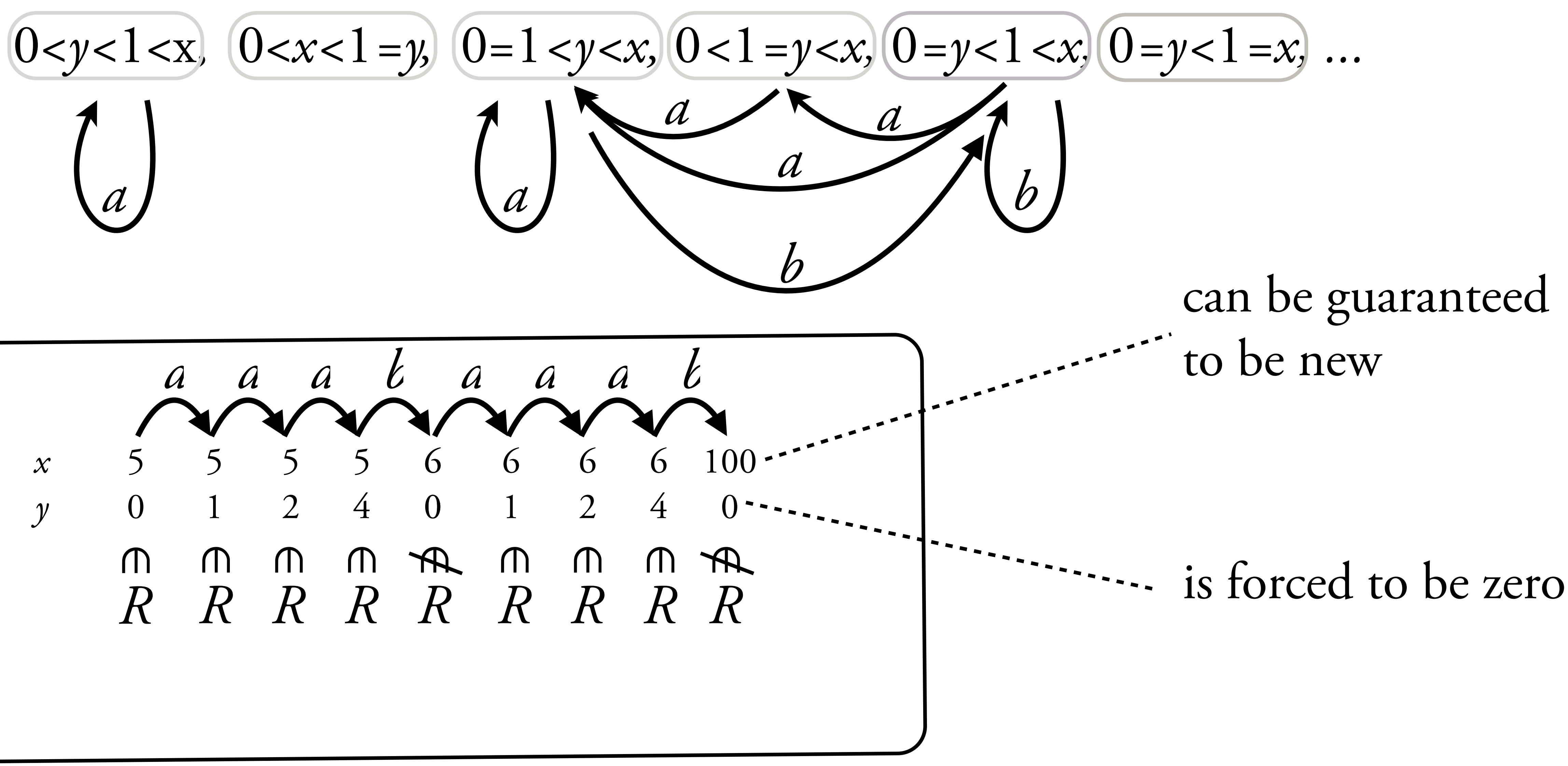


τ_I: $(x>0) \wedge (y=0)$

δ_a: $(x'=x) \wedge (y<y'<x) \wedge R(x,y)$

δ_b: $(y'=0) \wedge \neg R(x,y)$

τ_F: $(y=1)$



En elaboration of these ideas solves the problem for automata with database, over infinite words:

En elaboration of these ideas solves the problem for automata with database, over infinite words:

Theorem. For all $\mathscr{D}$, $\mathscr{D}$-automata over infinite words accept ωB-regular languages.

En elaboration of these ideas solves the problem for automata with database, over infinite words:

Theorem. For all $\mathscr{D}$, $\mathscr{D}$-automata over infinite words accept ωB-regular languages.

Theorem. Emptiness of $\mathscr{D}$-automata is decidable* in PSPACE

*For any reasonable linearly ordered structure $\mathscr{D}$

En elaboration of these ideas solves the problem for automata with database, over infinite words:

Theorem. For all $\mathscr{D}$, $\mathscr{D}$-automata over infinite words accept ωB-regular languages.

Theorem. Emptiness of $\mathscr{D}$-automata is decidable* in PSPACE

*For any reasonable linearly ordered structure $\mathscr{D}$

Corollary. Deciding LTL+(data tests) properties of $\mathscr{D}$-automata is PSPACE-complete.

Thank you!