Dynamic Partial Order Reduction under the Release-Acquire Semantics

joint work with
Mohamed Faouzi Atig
Bengt Jonsson
Tuan Phong Ngo

Outline

- **Memory Models**
  - **SC (Sequential Consistency**
  - **RA (Release-Acquire)**
- **Verification**
  - **Dynamic Partial-Order Reduction (DPOR)**
  - **Experiments**
- **Conclusions**

Semantics:

- **Lahav, Giannarakis, Vafeiadis. Taming release-acquire consistency. POPL, 2016.**
- **Kaiser, Dang, Dreyer, Lahav, Vafeiadis. Strong Logic for Weak Memory: Reasoning About Release-Acquire Consistency ECOOP 2017.**

Results:

- **Parosh Aziz Abdulla, Jatin Arora, Mohamed Faouzi Atig, Shankara Narayanan Krishna. Verification of programs under the release-acquire semantics. PLDI 2019.**
- **Parosh Aziz Abdulla, Mohamed Faouzi Atig, Bengt Jonsson, Magnus Lång, Tuan Phong NgoKonstantinos Sagonas. Optimal Stateless Model Checking for Read-from Equivalence under Sequential Consistency. OOPSLA 2019.**
- Parosh Aziz Abdulla, Mohamed Faouzi Atig, Bengt Jonsson, Tuan Phong Ngo. Optimal stateless model checking under the release-acquire semantics. OOPSLA 2018.
- Parosh Aziz Abdulla, Stavros Aronis, Bengt Jonsson, Konstantinos Sagonas. Source Sets: A Foundation for Optimal Dynamic Partial Order Reduction. JACM 2017.
- **Parosh Aziz Abdulla, Stavros Aronis, Bengt Jonsson, Konstantinos Sagonas. Optimal dynamic partial order reduction. POPL 2014.**

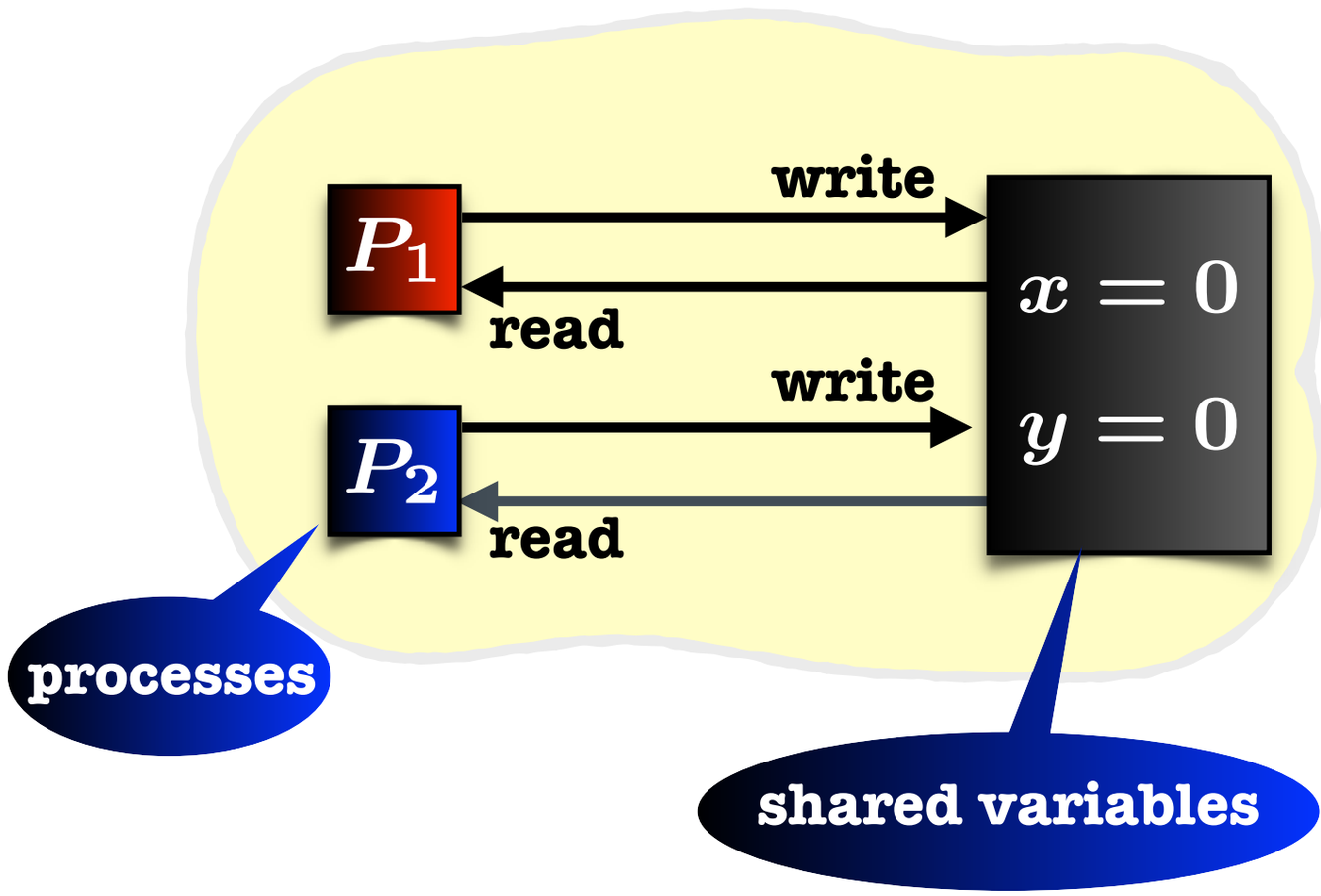
Related Work: check the reference lists in the above papers

Outline

- **Memory Models**
  - **SC (Sequential Consistency**
  - **RA (Release-Acquire)**
- **Verification**
  - **Dynamic Partial-Order Reduction (DPOR)**
  - **Experiments**
- **Conclusions**

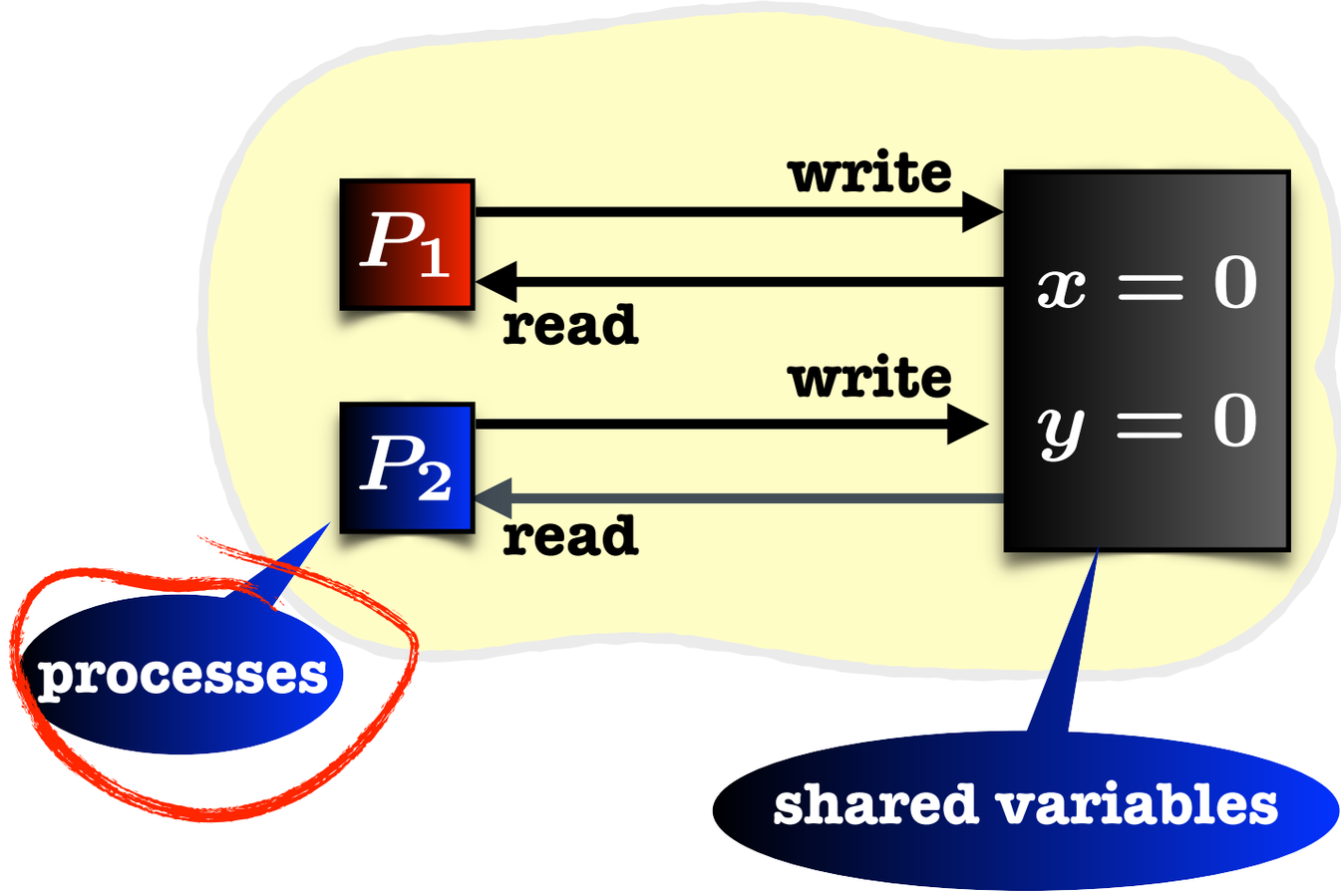
Concurrent Programs

- **Processes**
- **Shared Variables**



Concurrent Programs

- **Processes**
- **Shared Variables**



Concurrent Programs

- **Processes**
- **Shared Variables**

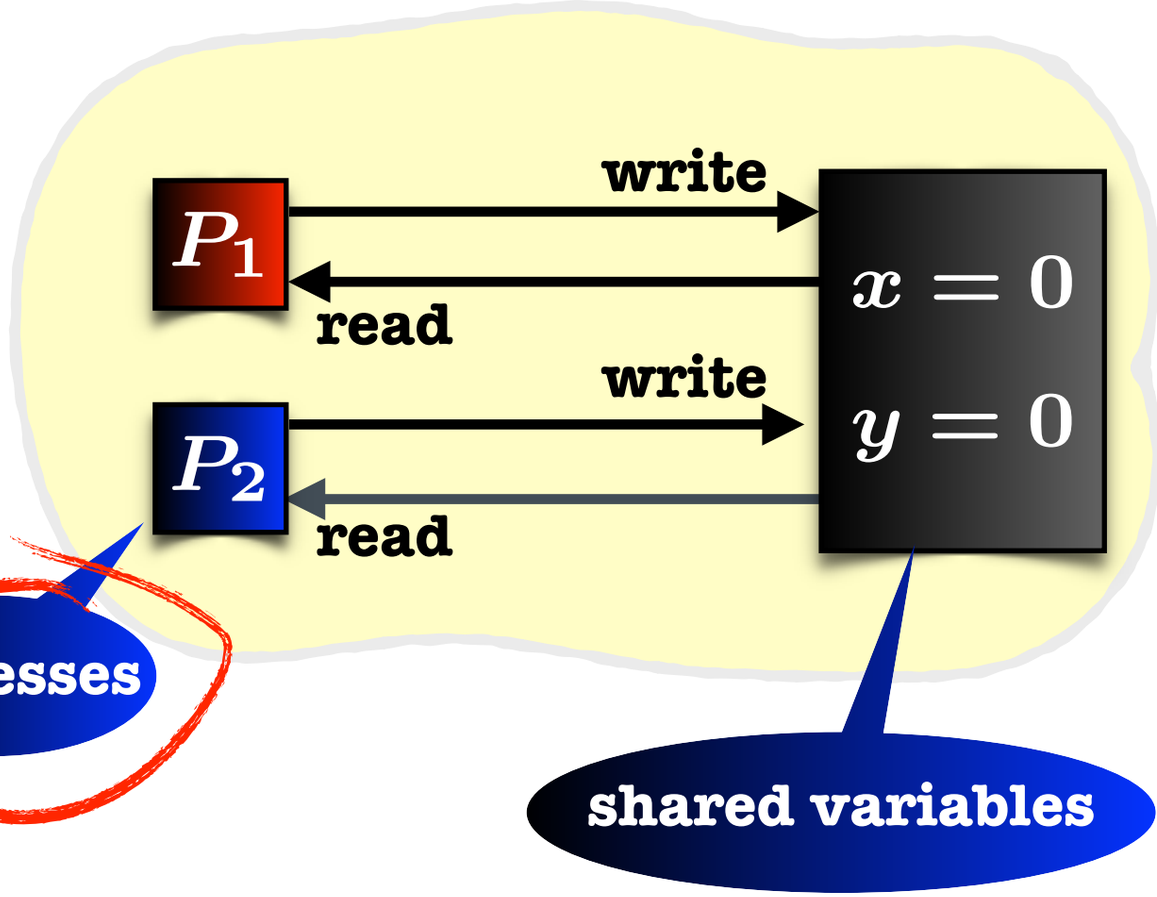
Initially: $x = y = 0$

$$\begin{array}{l|l} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



Concurrent Programs

- **Processes**
- **Shared Variables**

Initially: $x = y = 0$

$$P_1 \quad \| \quad P_2$$

$$x := 1 \quad \| \quad y := 1$$

$$a := y \quad \| \quad b := x$$

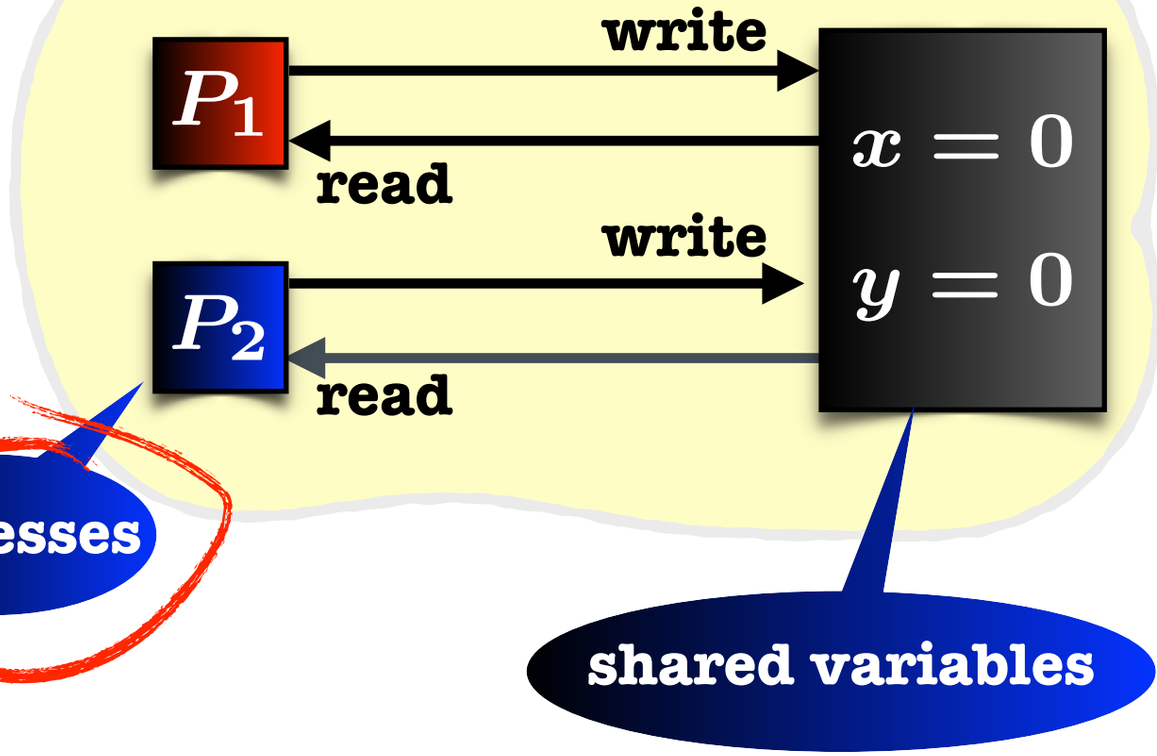
$(a = 0) \wedge (b = 0)$?

assertion

local variables

Store Buffering (SB)

processes



shared variables

Concurrent Programs

- **Processes**
- **Shared Variables**

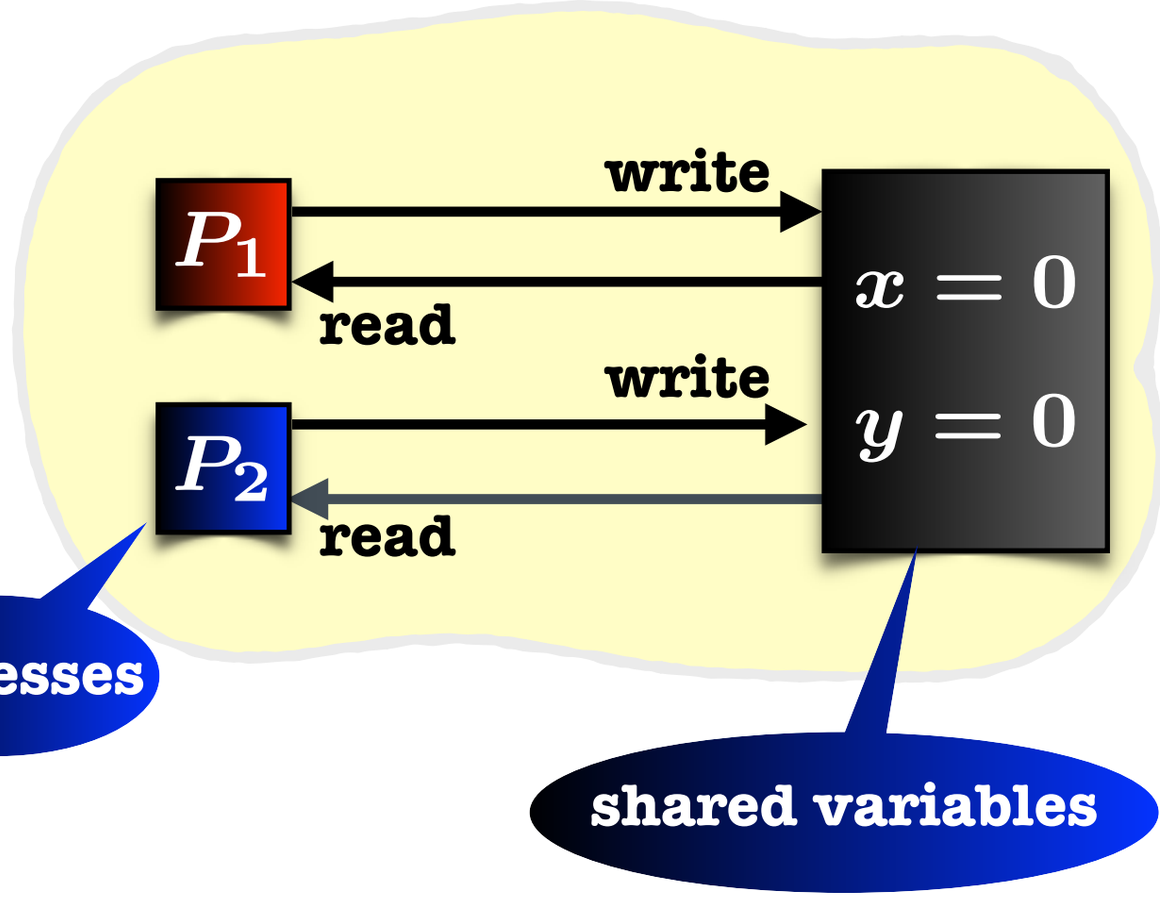
Initially: $x = y = 0$

$$\begin{array}{l|l} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



Concurrent Programs

- **Processes**
- **Shared Variables**

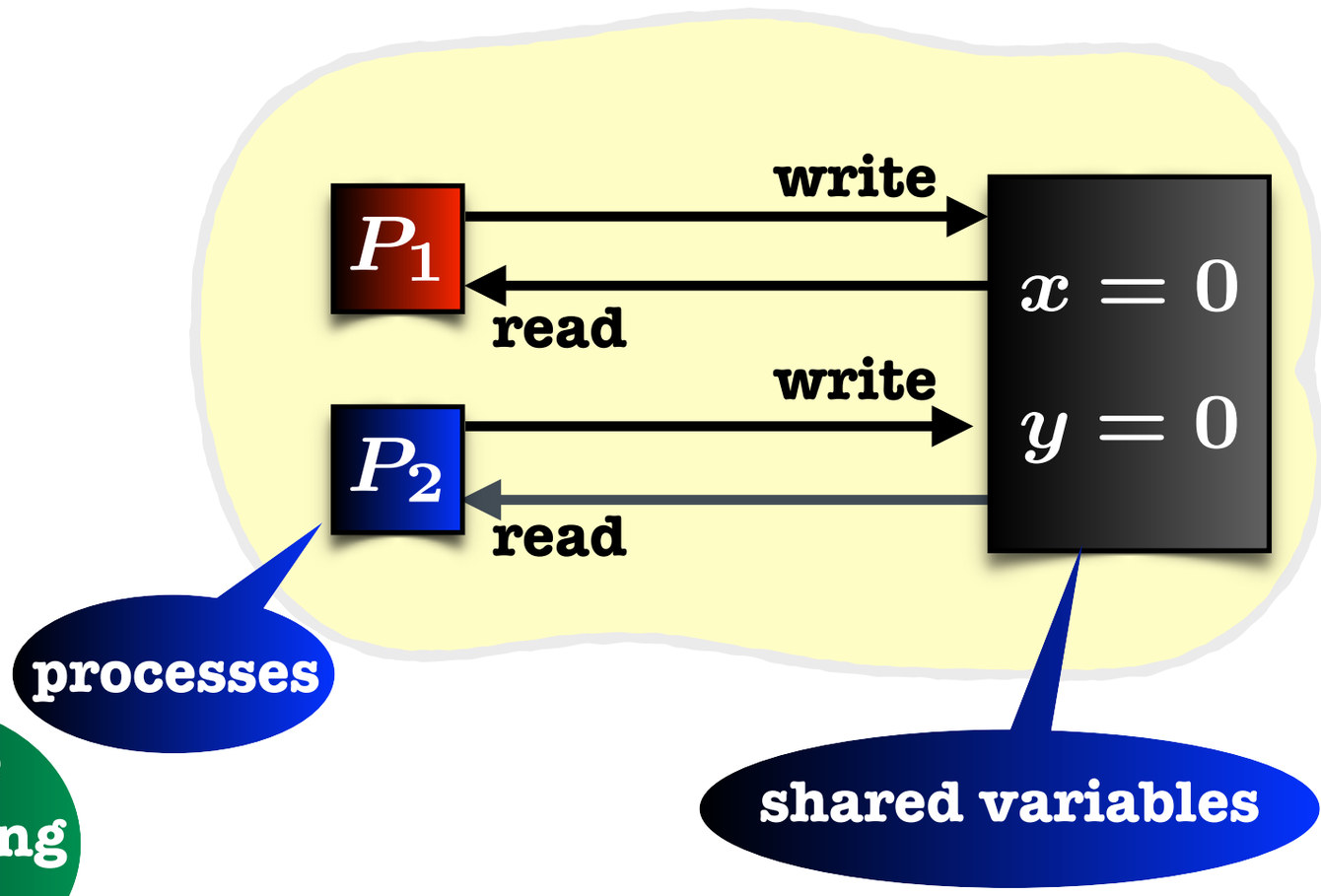
Initially: $x = y = 0$

$$\begin{array}{l|l} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

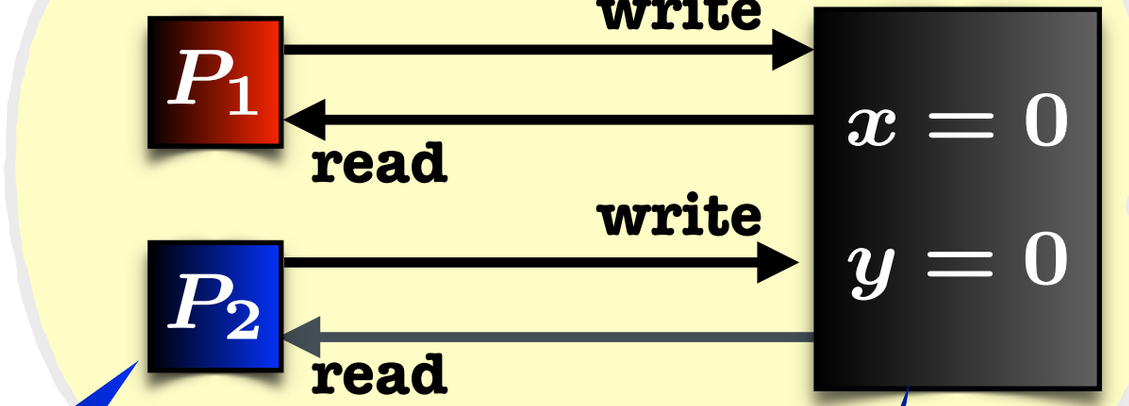
Initially: $x = y = 0$

$$\begin{array}{l|l} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

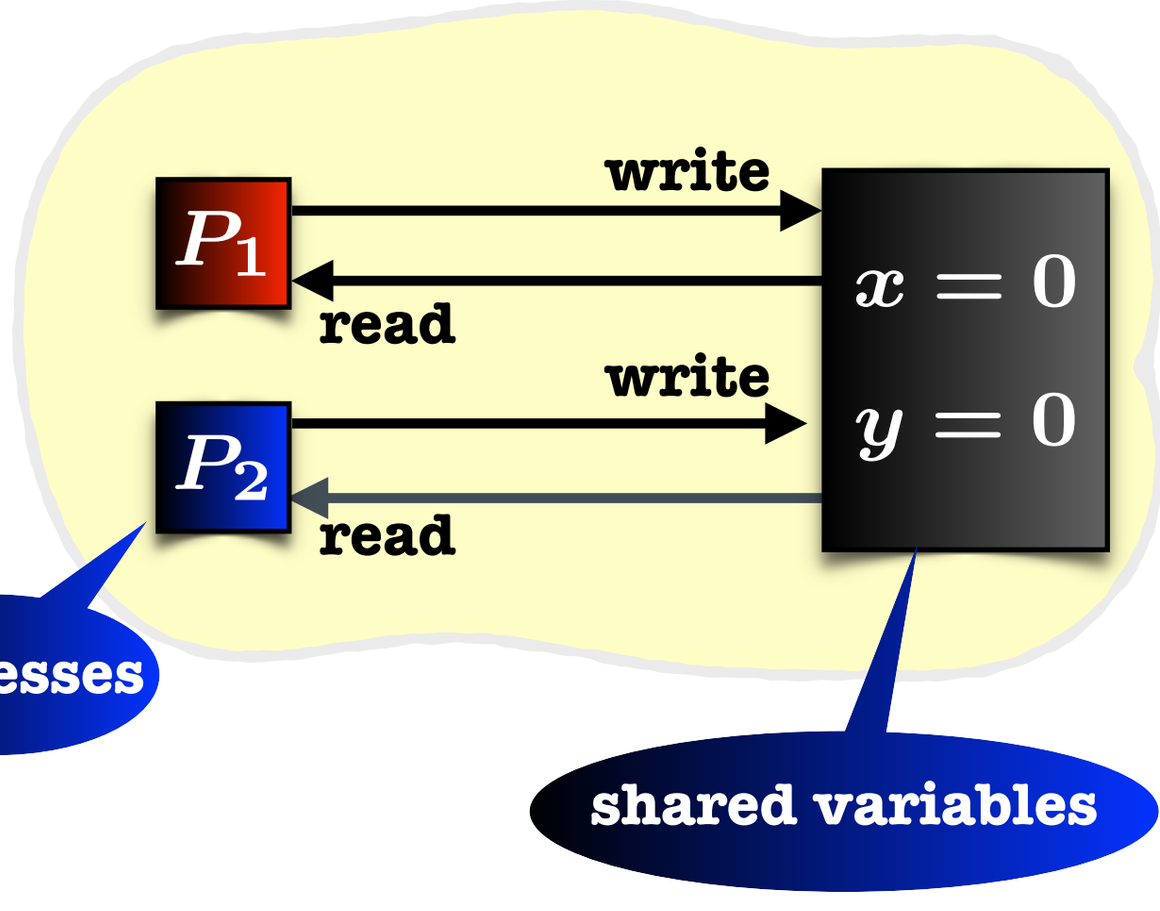
Initially: $x = y = 0$

$$\begin{array}{l|l} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

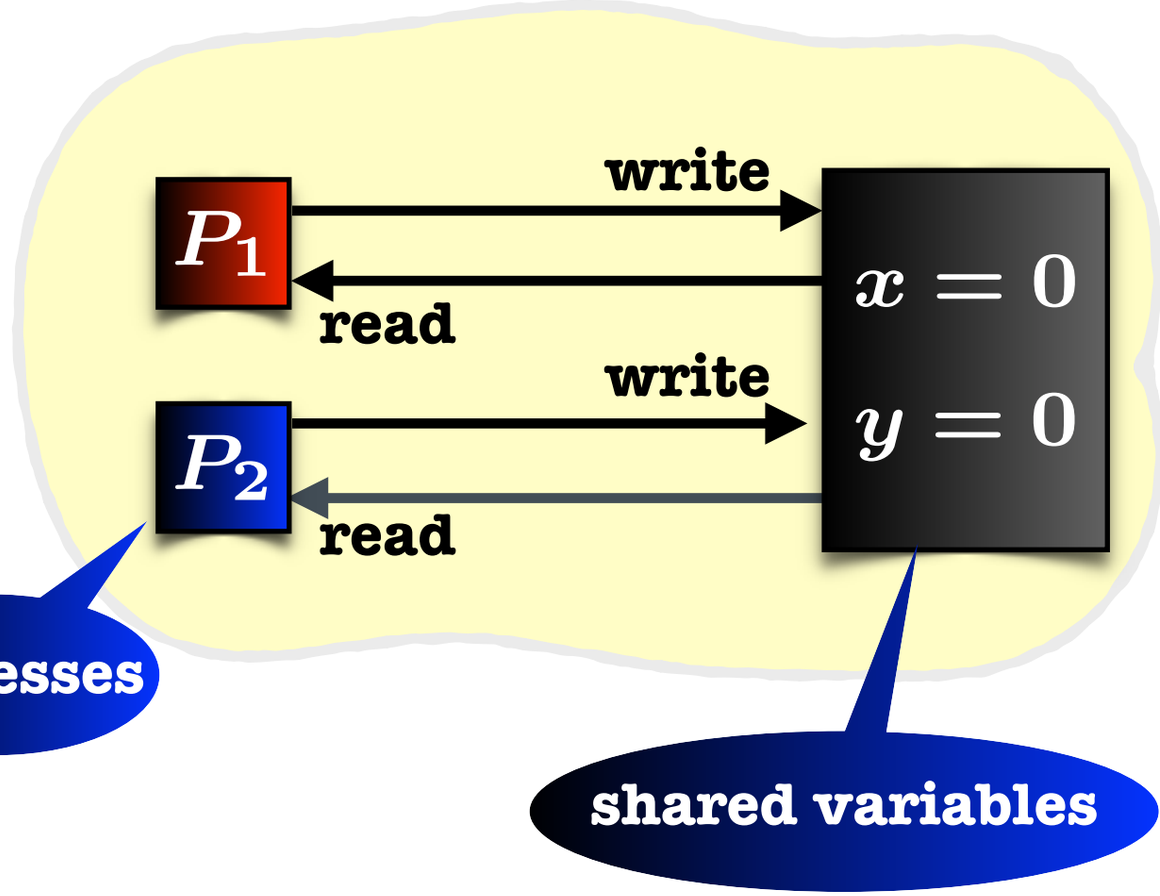
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

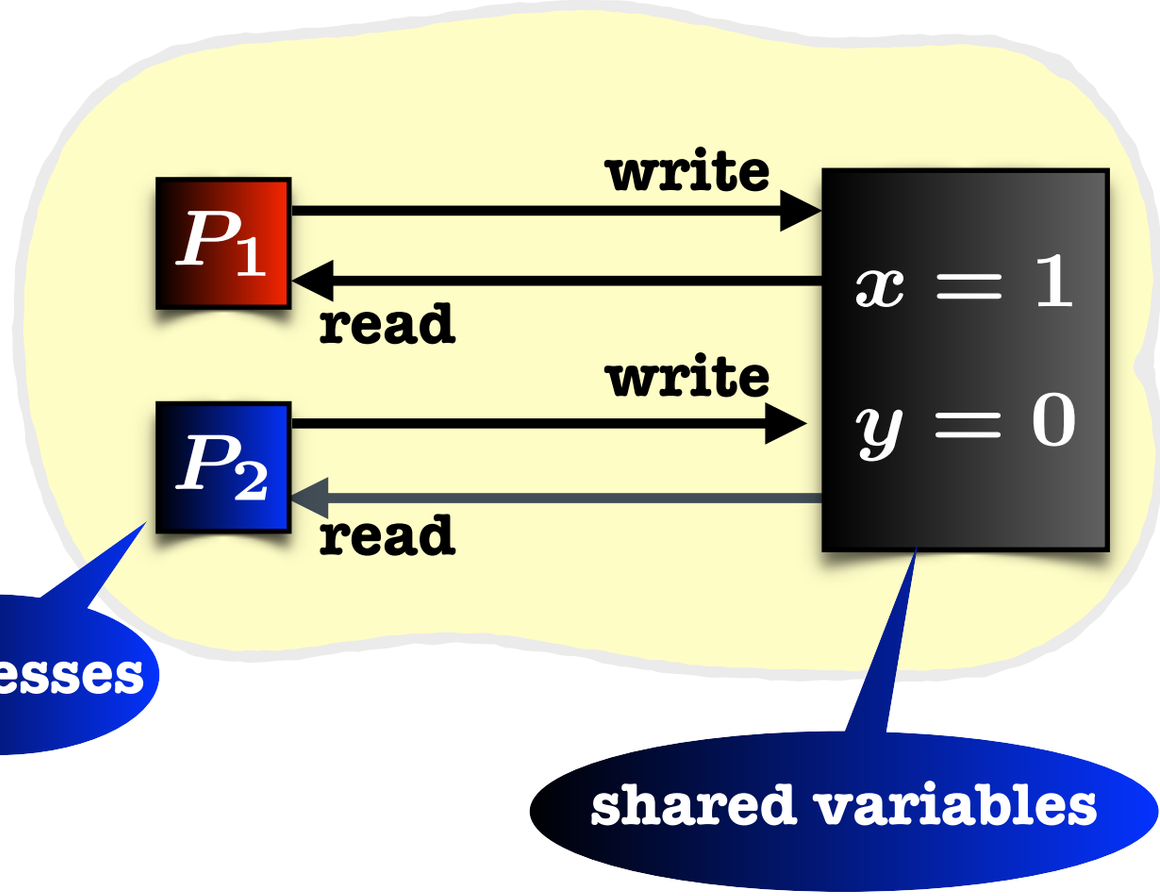
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

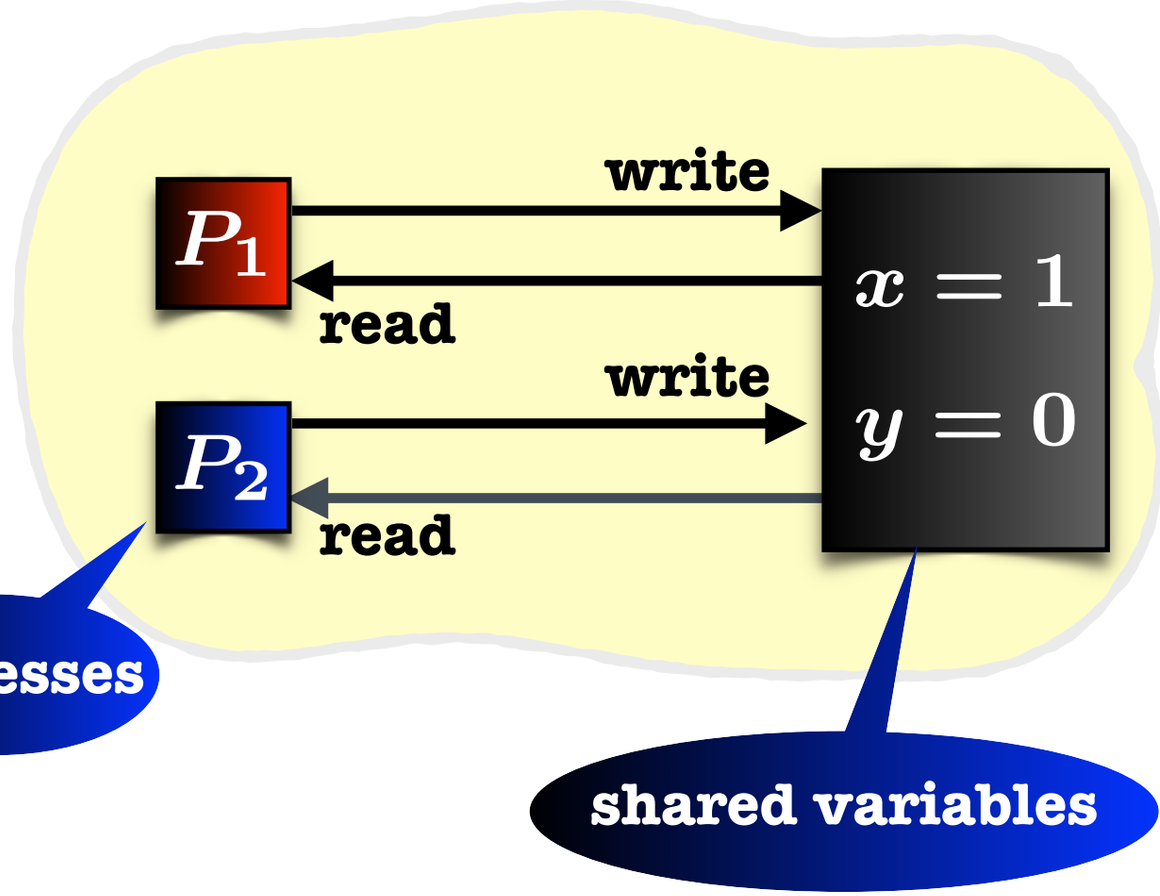
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events e_1 $x := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

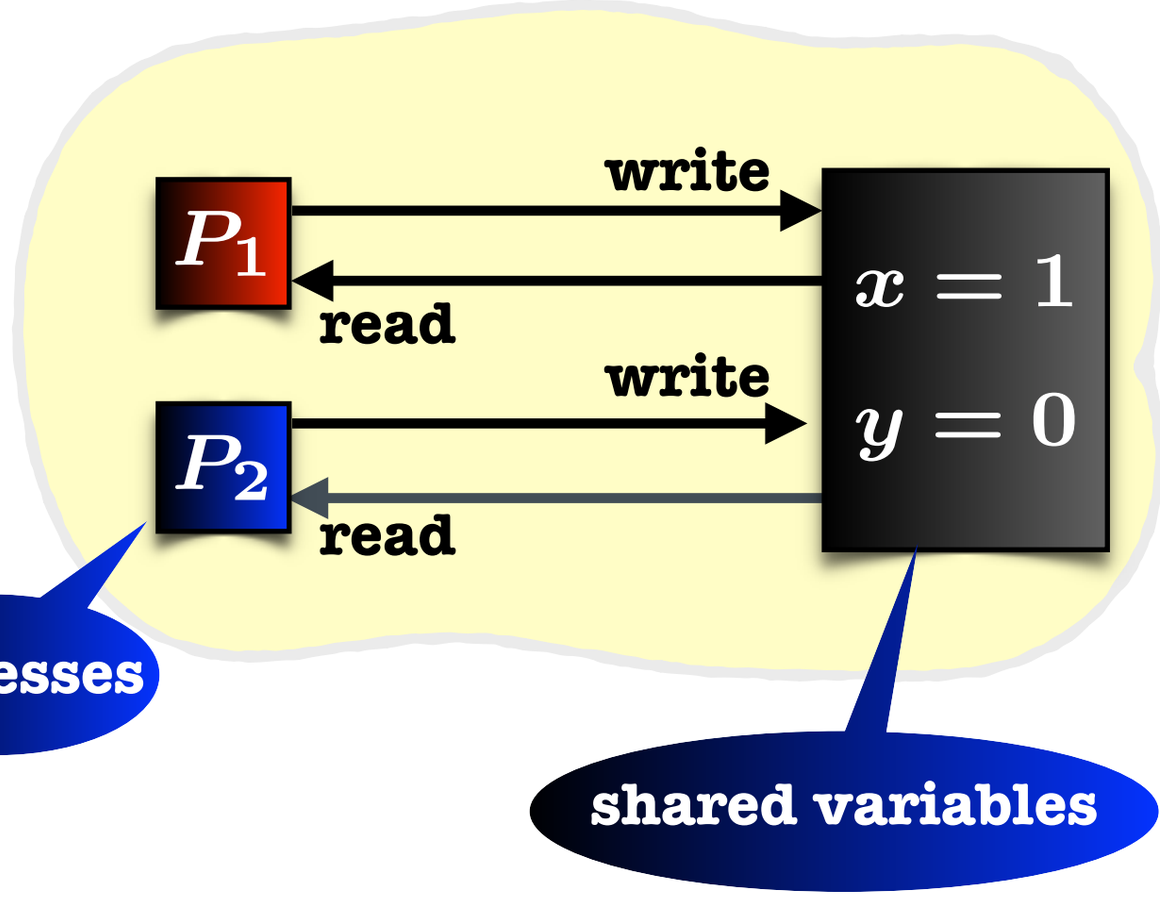
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events e_1 $x := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

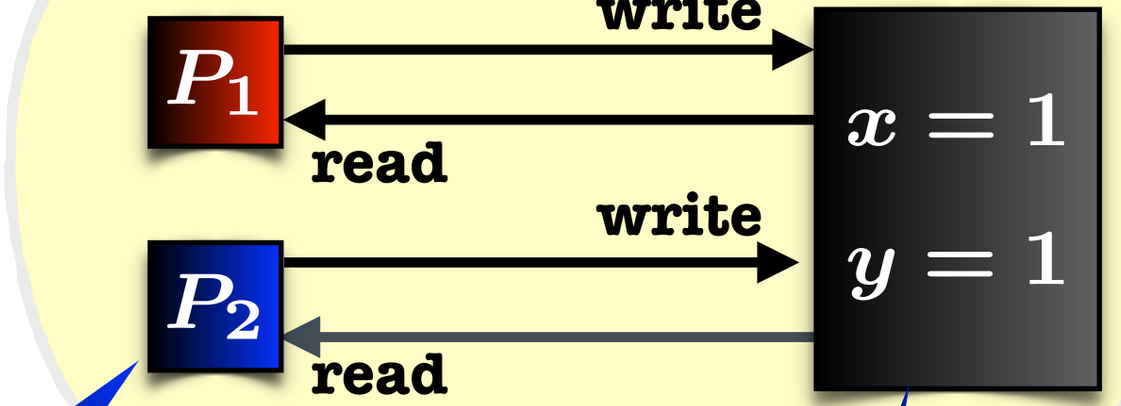
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events e_1 $x := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

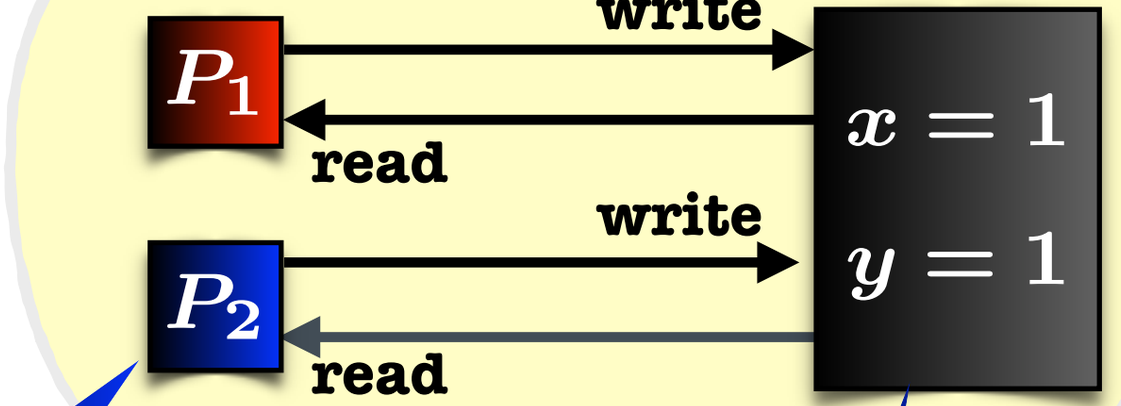
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events

e_1 $x := 1$

e_2 $y := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

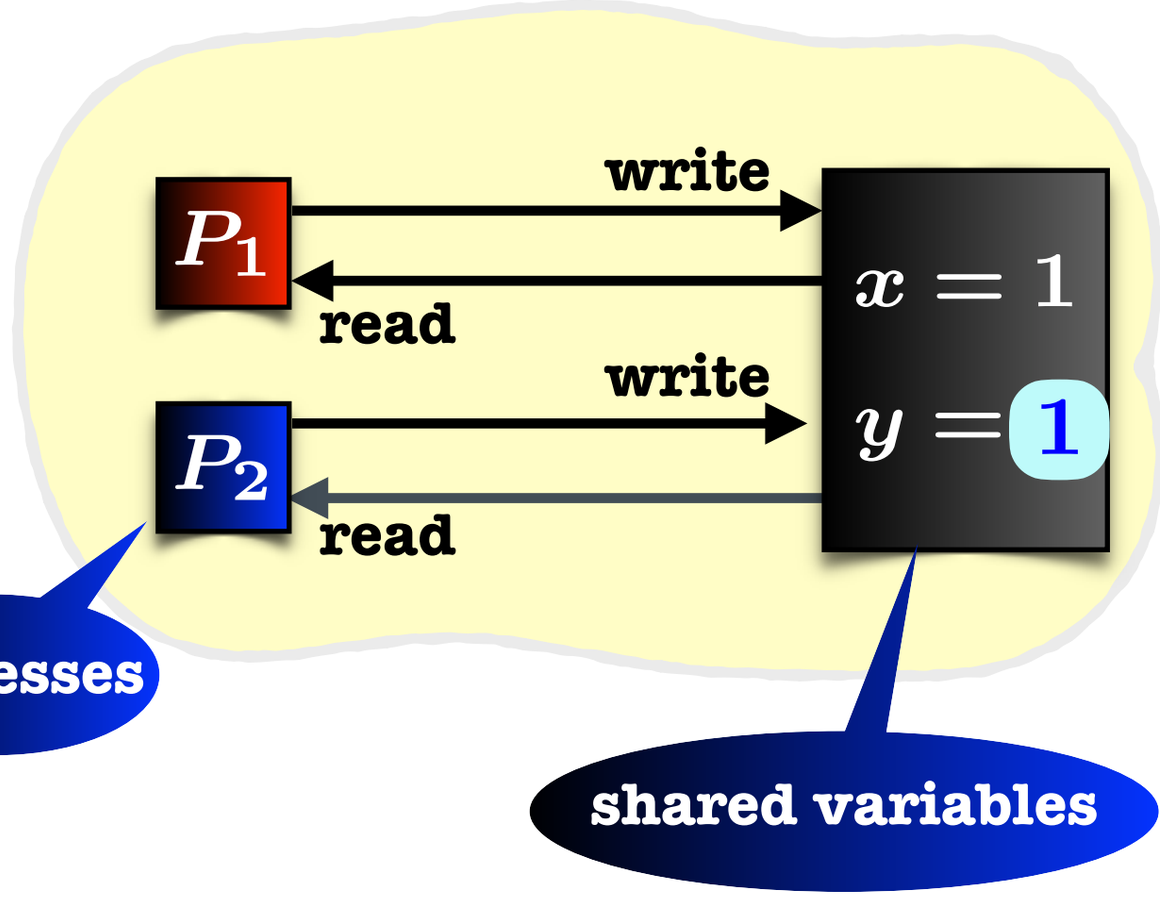
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events

e_1 $x := 1$

e_2 $y := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

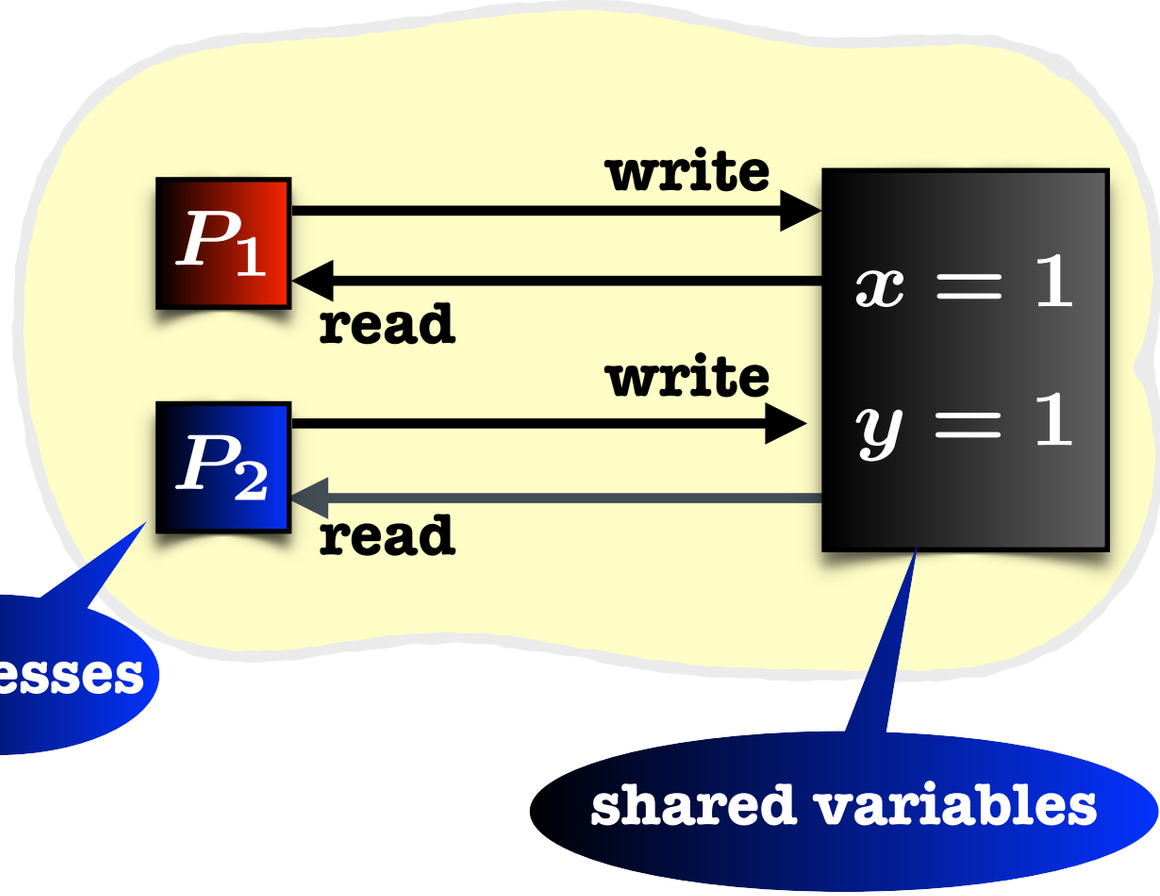
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := 1 & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)?$

assertion



π : run

events

e_1 $x := 1$

e_2 $y := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

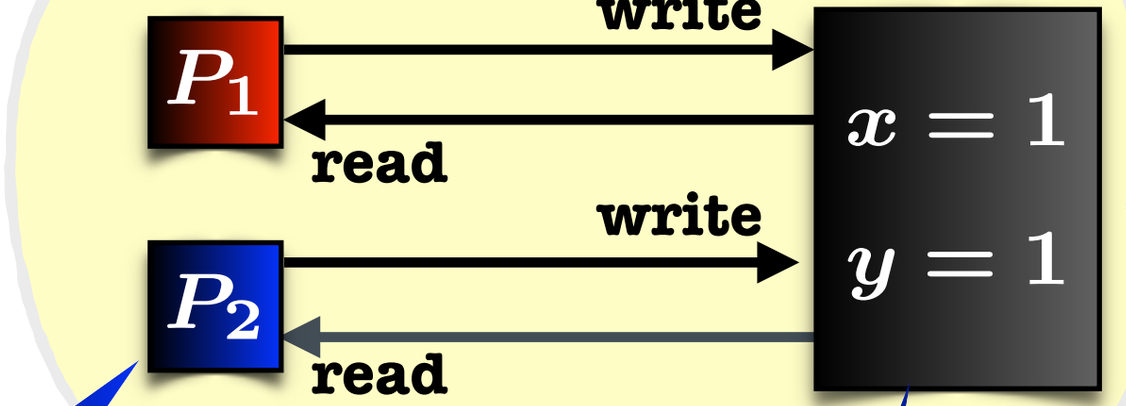
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := 1 & b := x \end{array}$$

Store Buffering (SB)

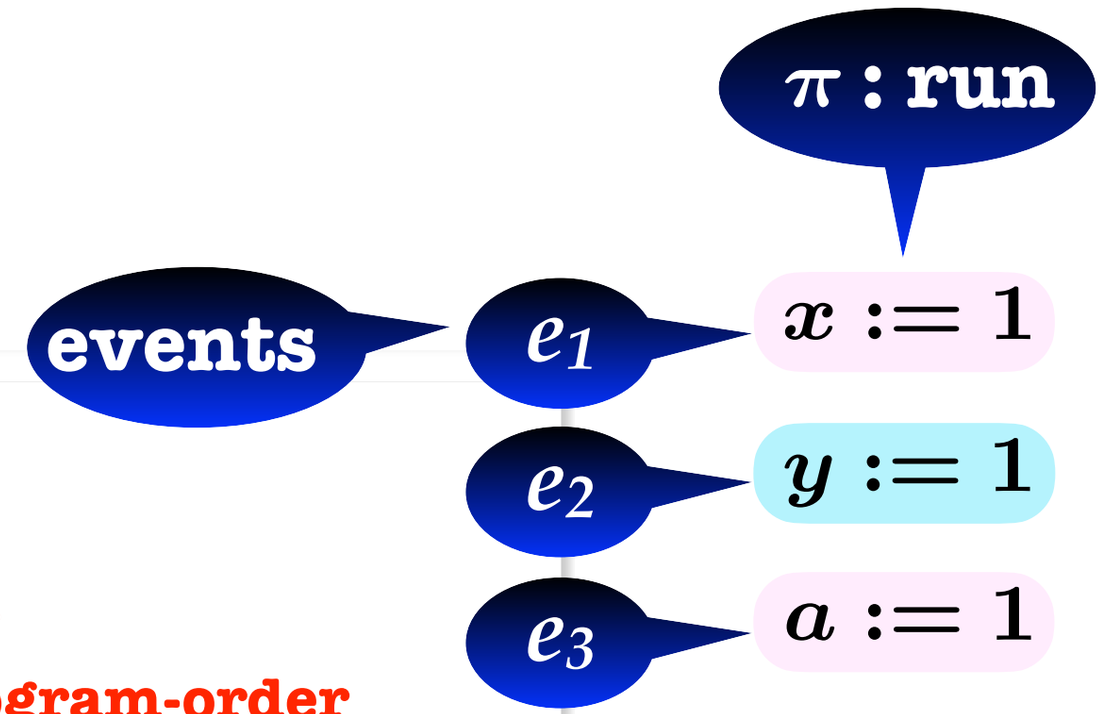
$(a = 0) \wedge (b = 0)?$

assertion



Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**



Concurrent Programs

- **Processes**
- **Shared Variables**

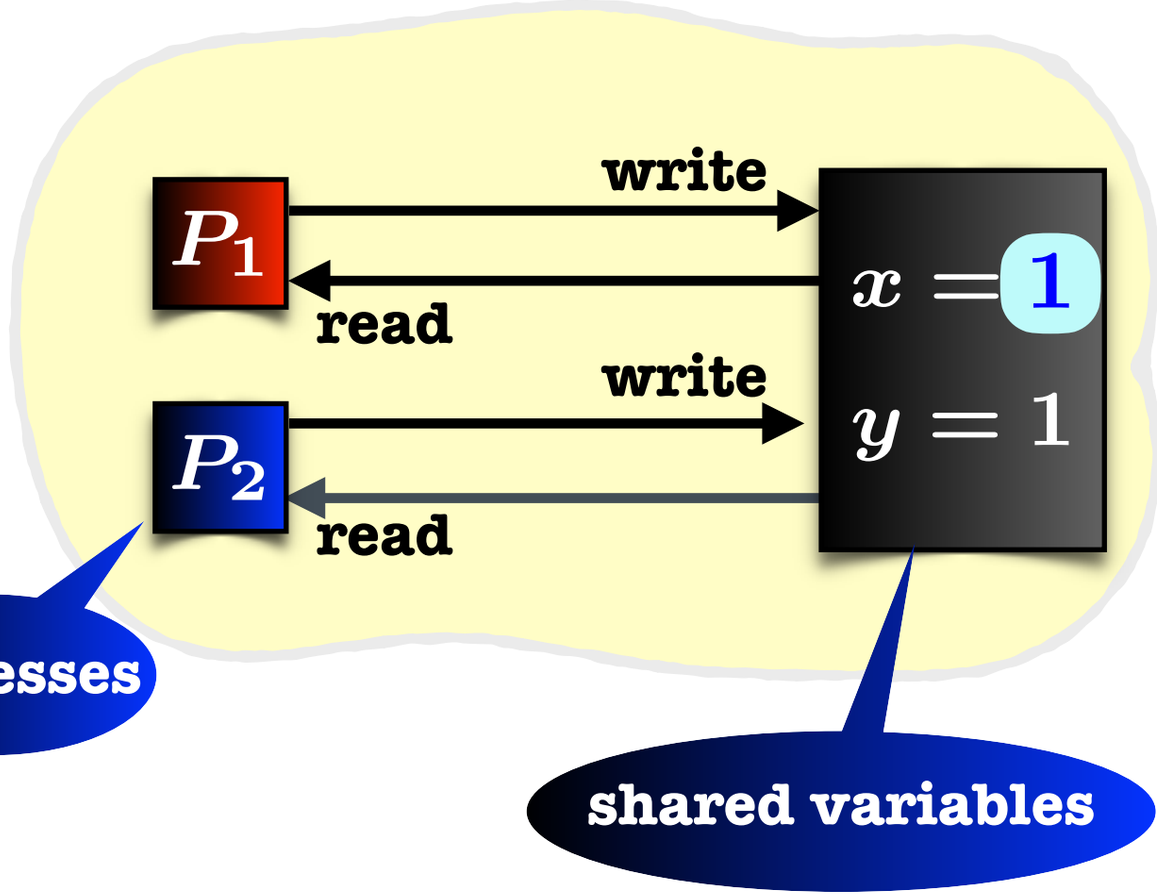
Initially: $x = y = 0$

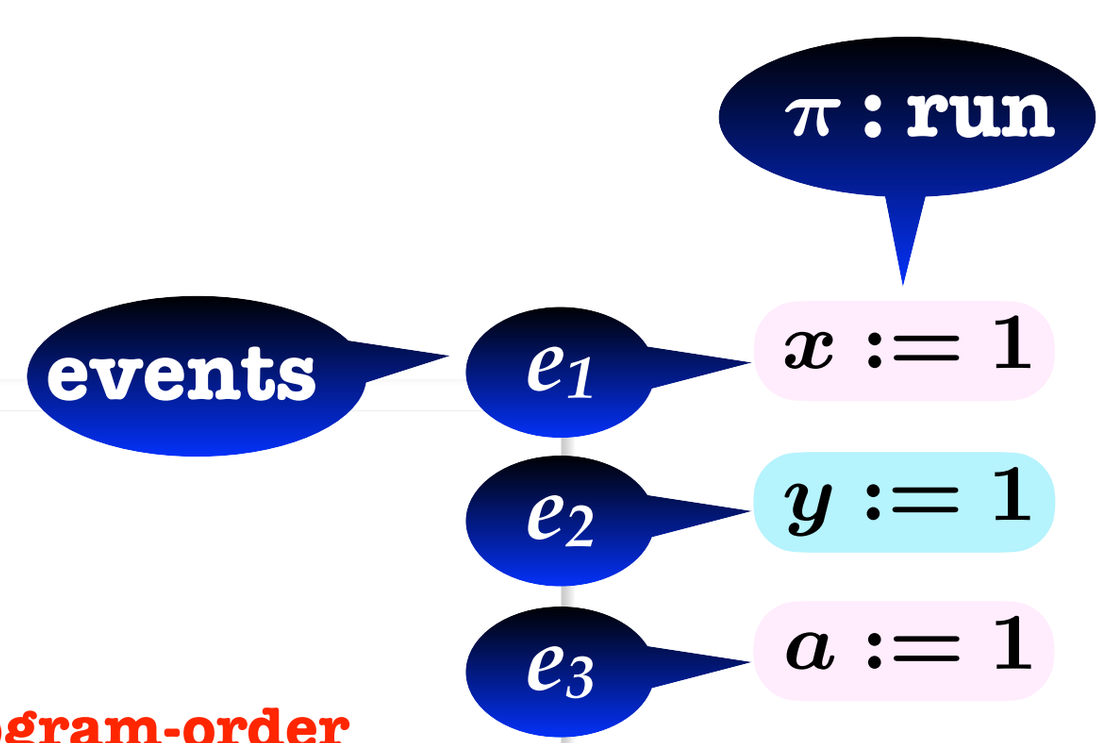
$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := 1 & b := x \end{array}$$

$(a = 0) \wedge (b = 0)?$

assertion

Store Buffering (SB)





Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

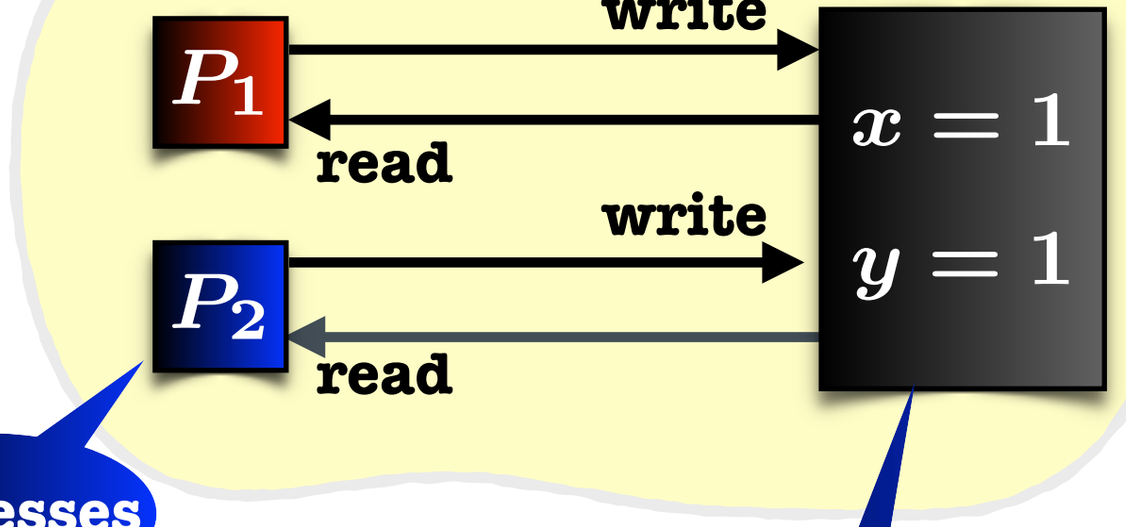
- **Processes**
- **Shared Variables**

Initially: $x = y = 0$

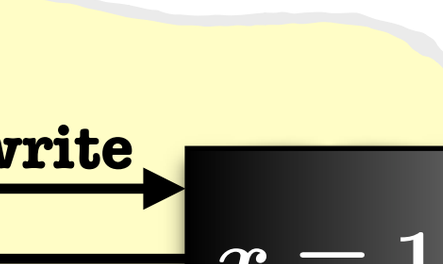
$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := 1 & b := 1 \end{array}$$

$(a = 0) \wedge (b = 0)?$

assertion



Store Buffering (SB)



events

e_1	$x := 1$
e_2	$y := 1$
e_3	$a := 1$

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Concurrent Programs

- **Processes**
- **Shared Variables**

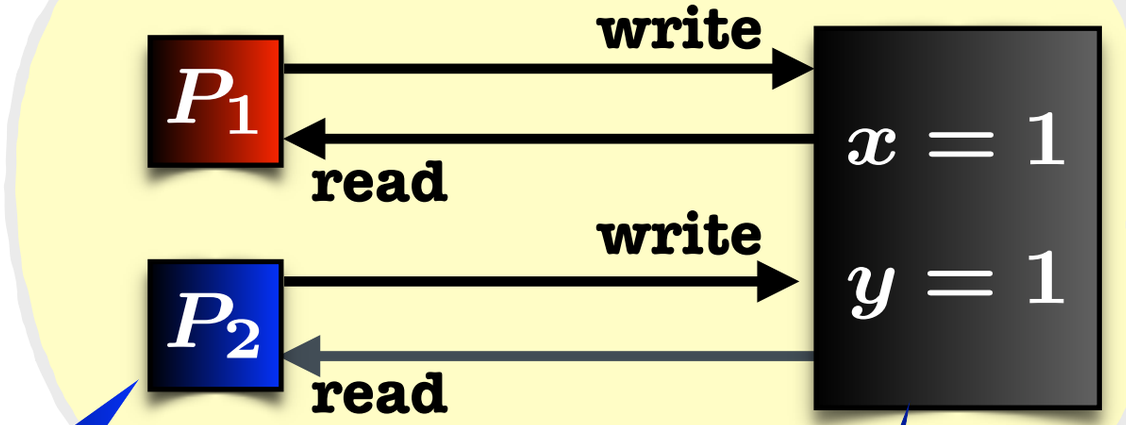
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := 1 & b := 1 \end{array}$$

$(a = 0) \wedge (b = 0)?$

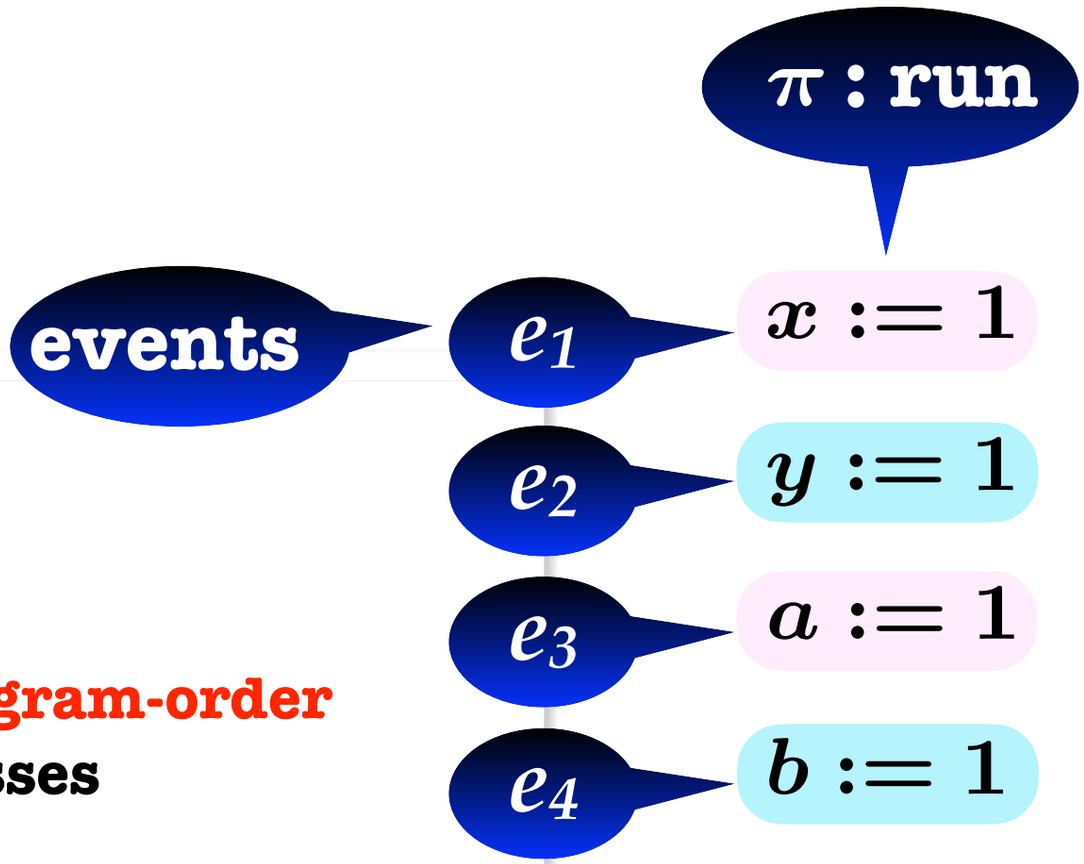
assertion

Store Buffering (SB)



Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**



Concurrent Programs

- **Processes**
- **Shared Variables**

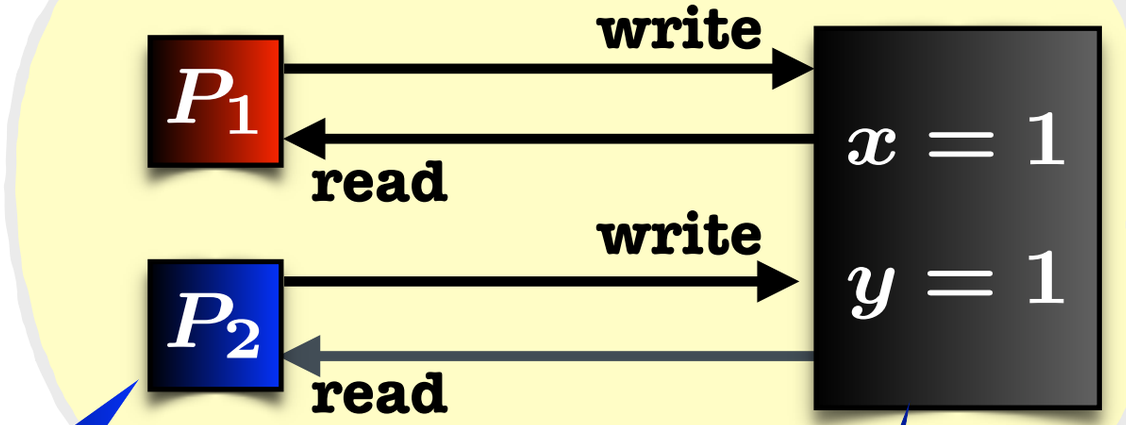
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

$(a = 0) \wedge (b = 0)?$

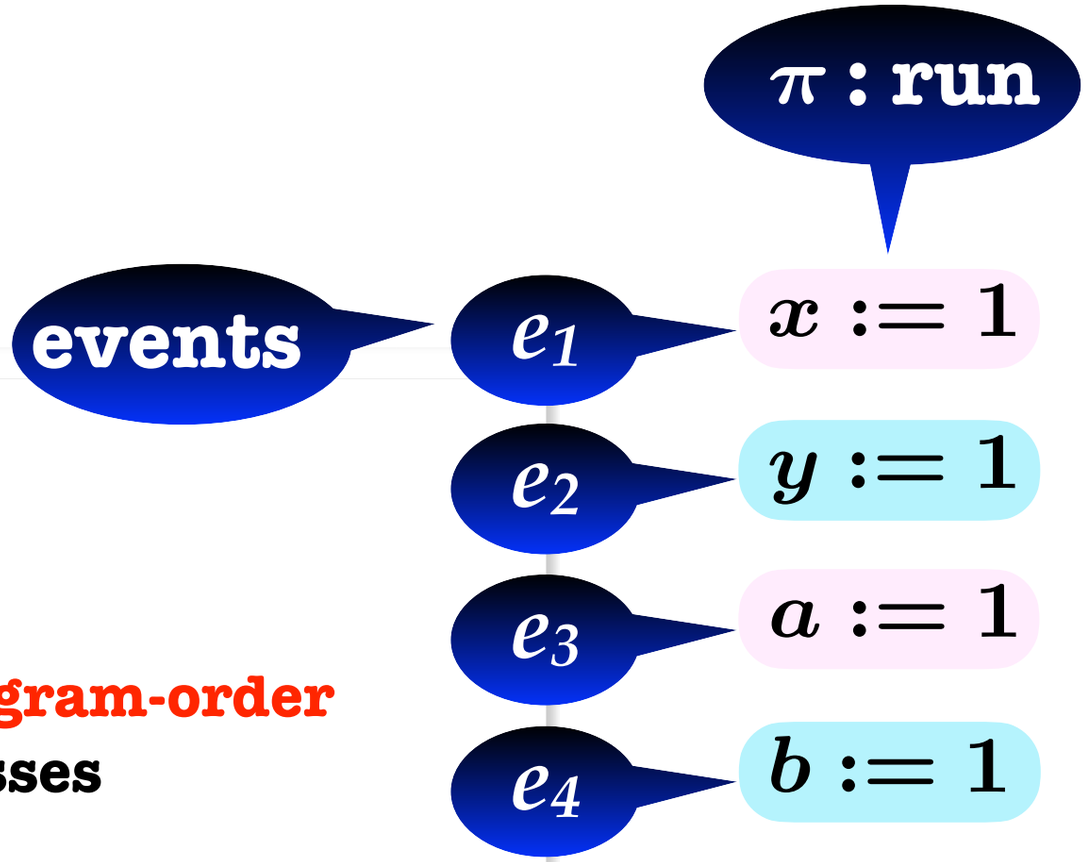
assertion

Store Buffering (SB)



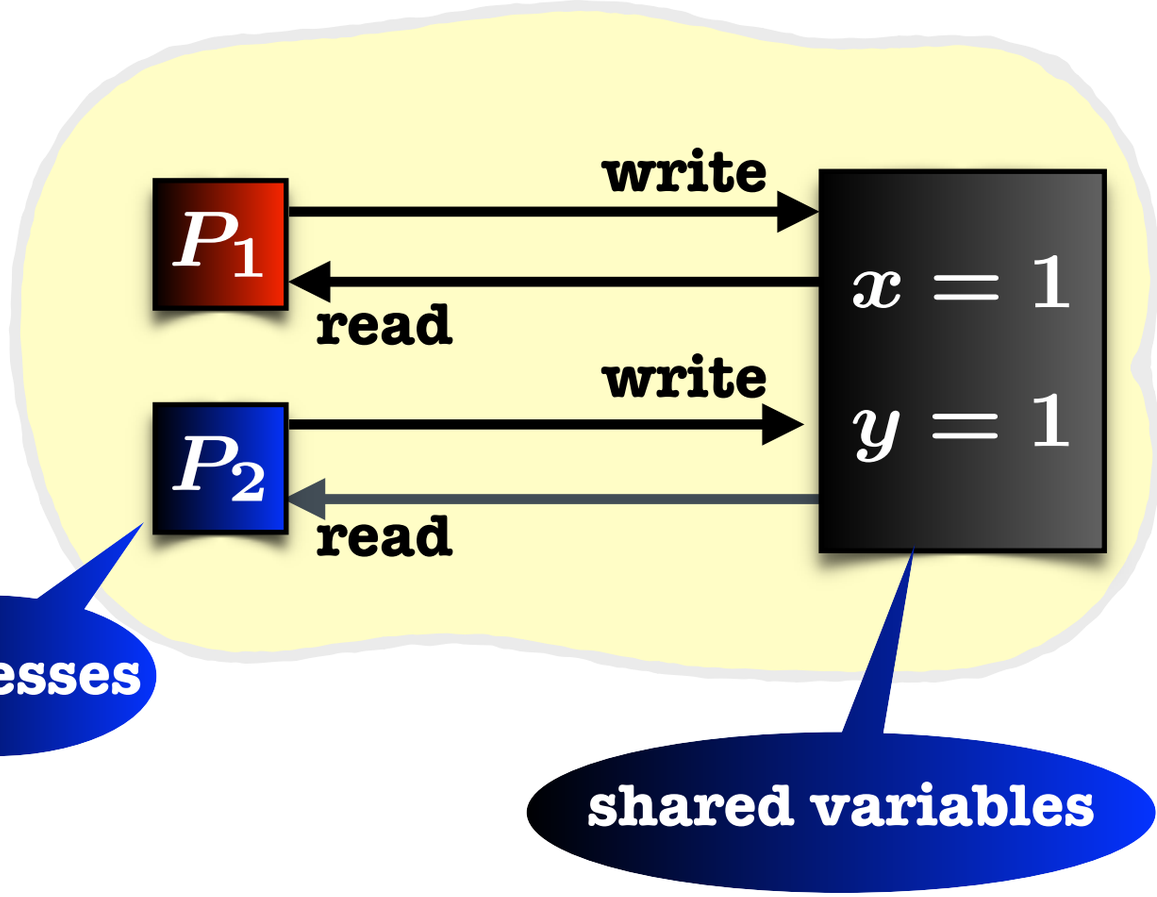
Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**



Concurrent Programs

- **Processes**
- **Shared Variables**



Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

Store Buffering (SB)

$(a = 0) \wedge (b = 0)$?

assertion

unreachable

Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

π : run

events

e_1 $x := 1$

e_2 $y := 1$

e_3 $a := 1$

e_4 $b := 1$

Concurrent Programs

- **Processes**
- **Shared Variables**

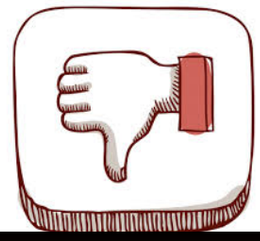
Initially: $x = y = 0$

$$\begin{array}{c|c} P_1 & P_2 \\ x := 1 & y := 1 \\ a := y & b := x \end{array}$$

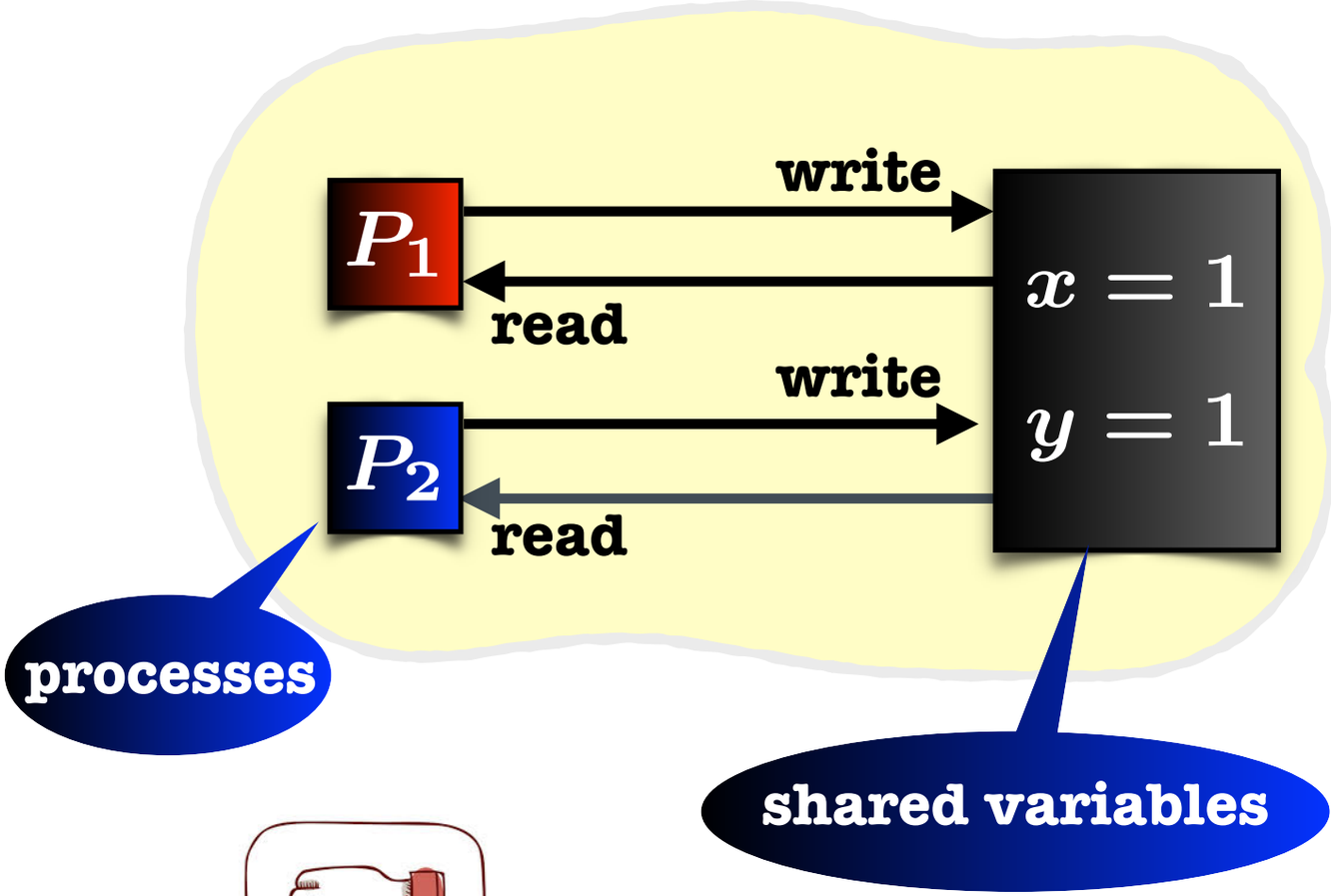
$(a = 0) \wedge (b = 0)?$

assertion

- simple
- intuitive



- costly
- unrealistic



Sequential Consistency (SC)

- **Shared Memory**
- **Processes perform read/write atomically**
- **execute instructions of one process in program-order**
- **interleave instructions of different processes**
- **read-from the latest write**

Weakly Consistent Systems

- **Microprocessors:**
  - **TSO, POWER, ARM, …**
- **Weak Cache Protocols:**
  - **TSO-CC, Racer, SISD, …**
- **Programming Languages:**
  - **C11, Java, …**
- **Distributed Data Stores:**
  - **Amazon, Facebook, Google, …**

Weakly Consistent Systems

- Microprocessors:
  - TSO, POWER, ARM, …
- Weak Cache Protocols:
  - TSO-CC, Racer, SISD, …
- Programming Languages:
  - C11, Java, …
- Distributed Data Stores:
  - Amazon, Facebook, Google, …

Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire

Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire

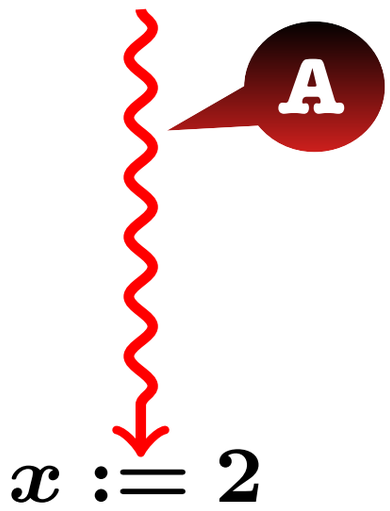
Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire



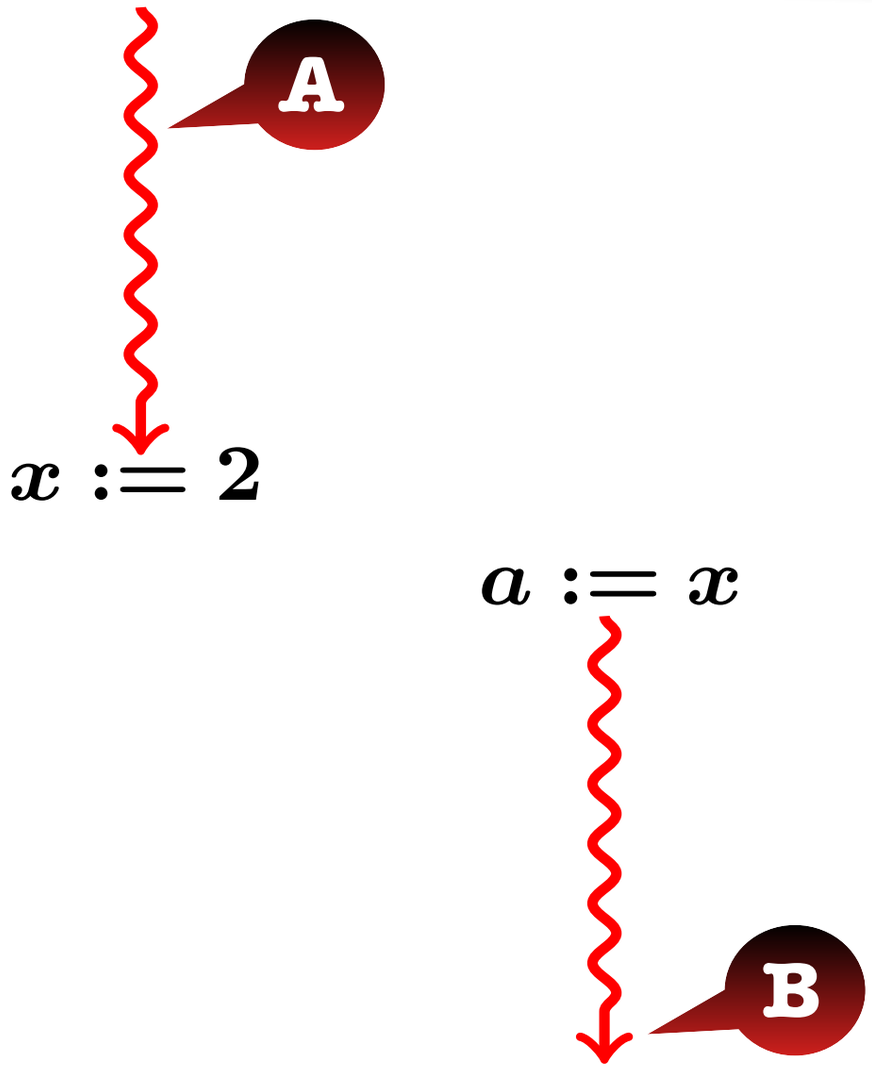
Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire



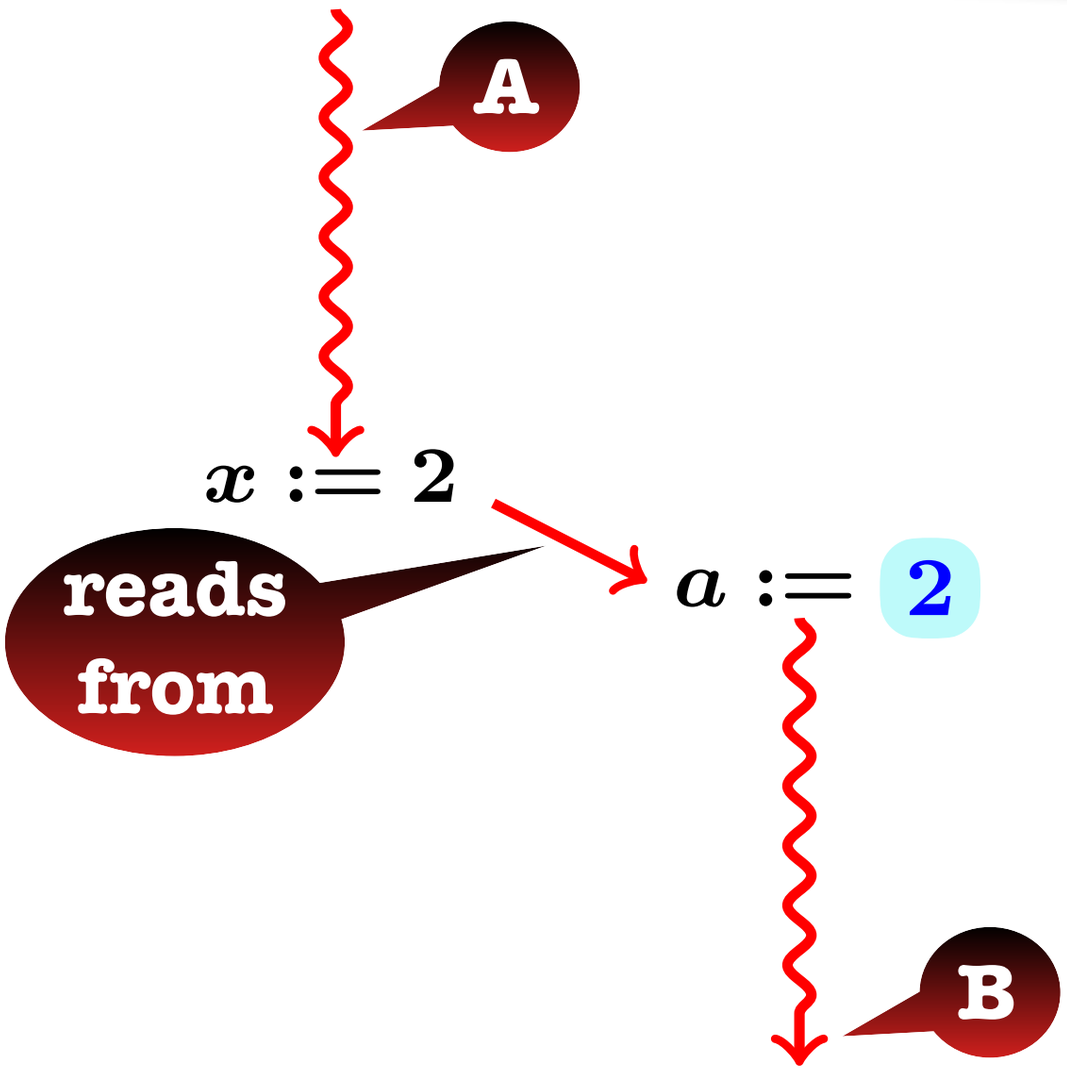
Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire



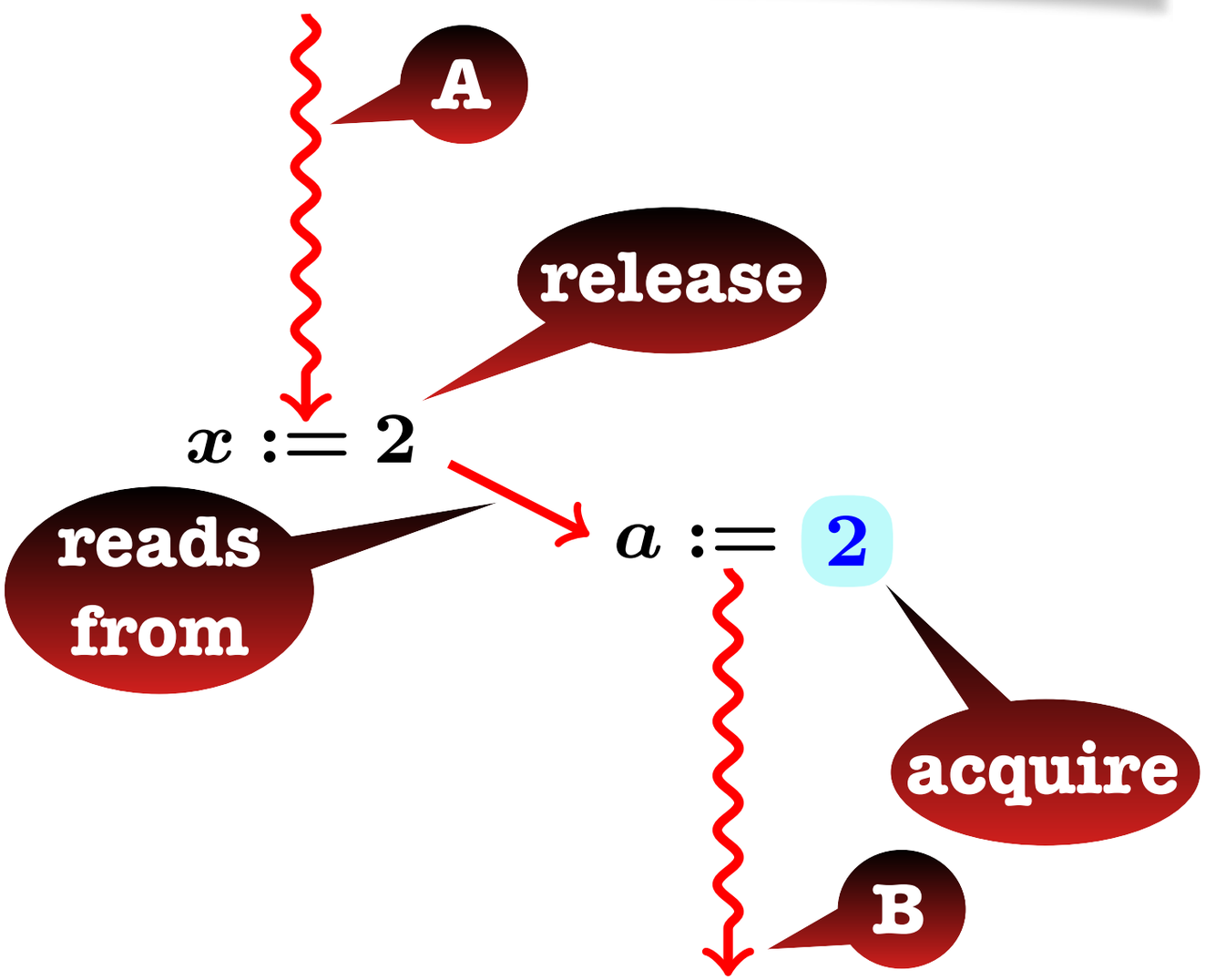
Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire



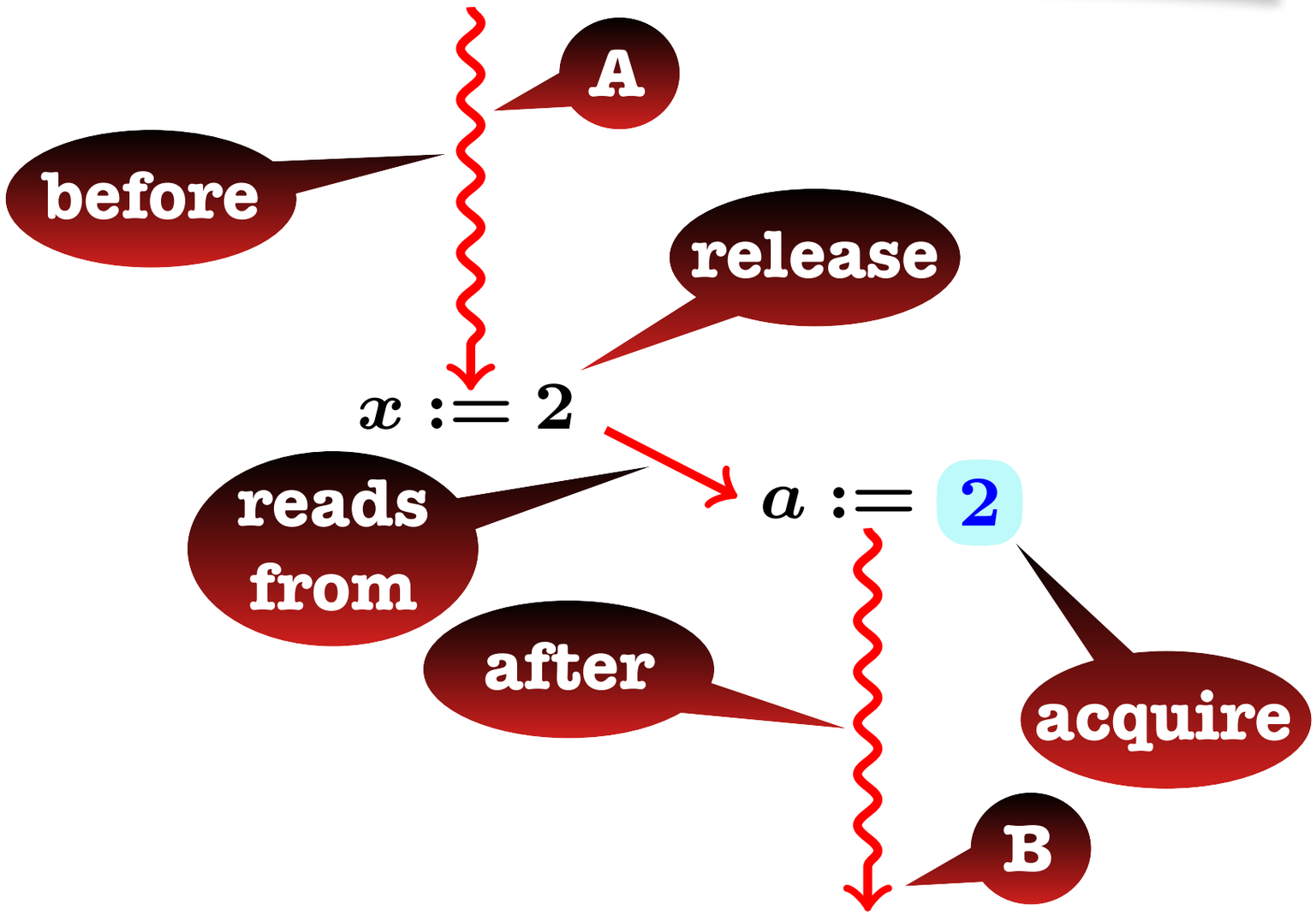
Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire



Release-Acquire (RA)

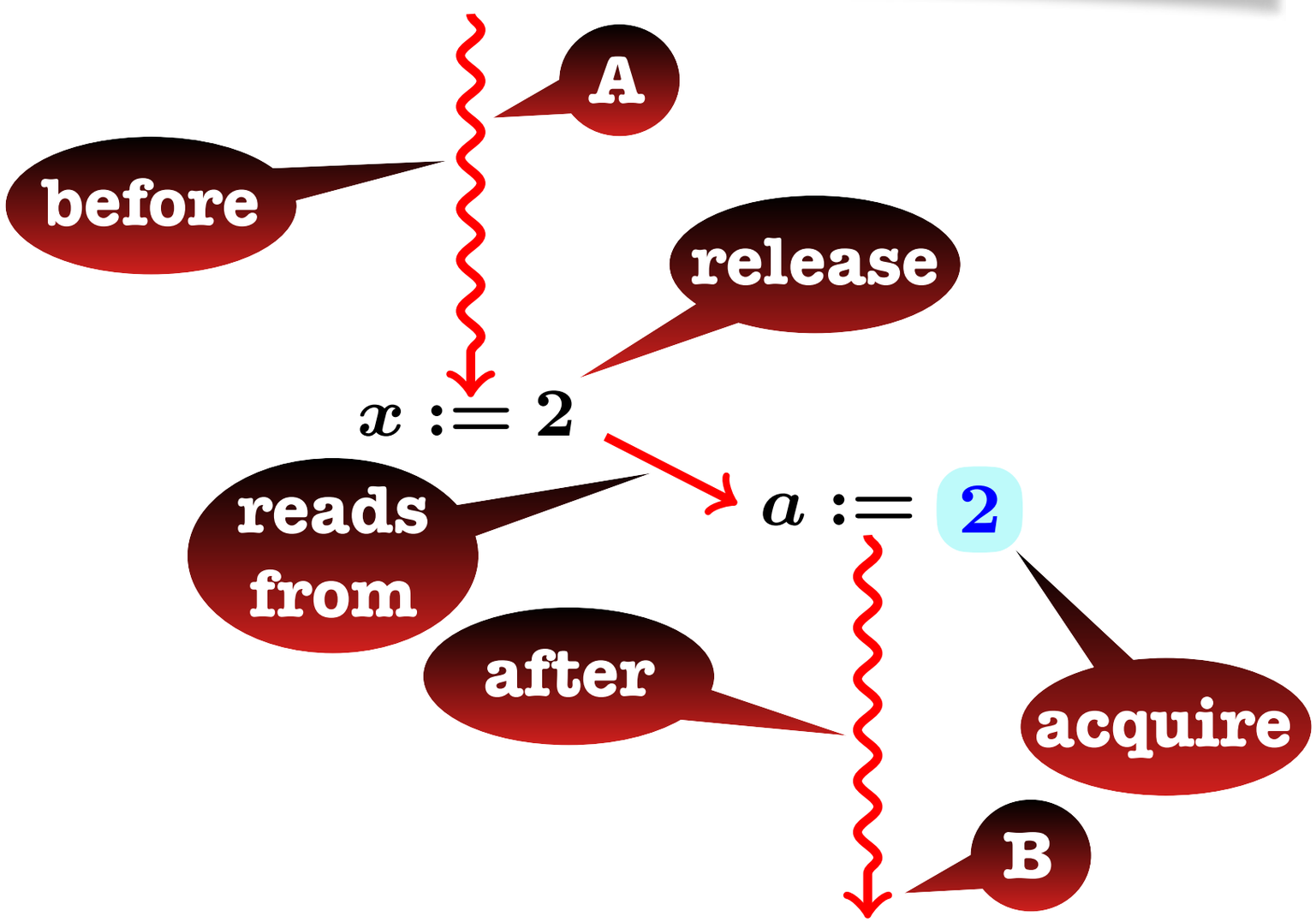
Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

- Program order
- Synchronization
  - write: release
  - read: acquire

A happens before B



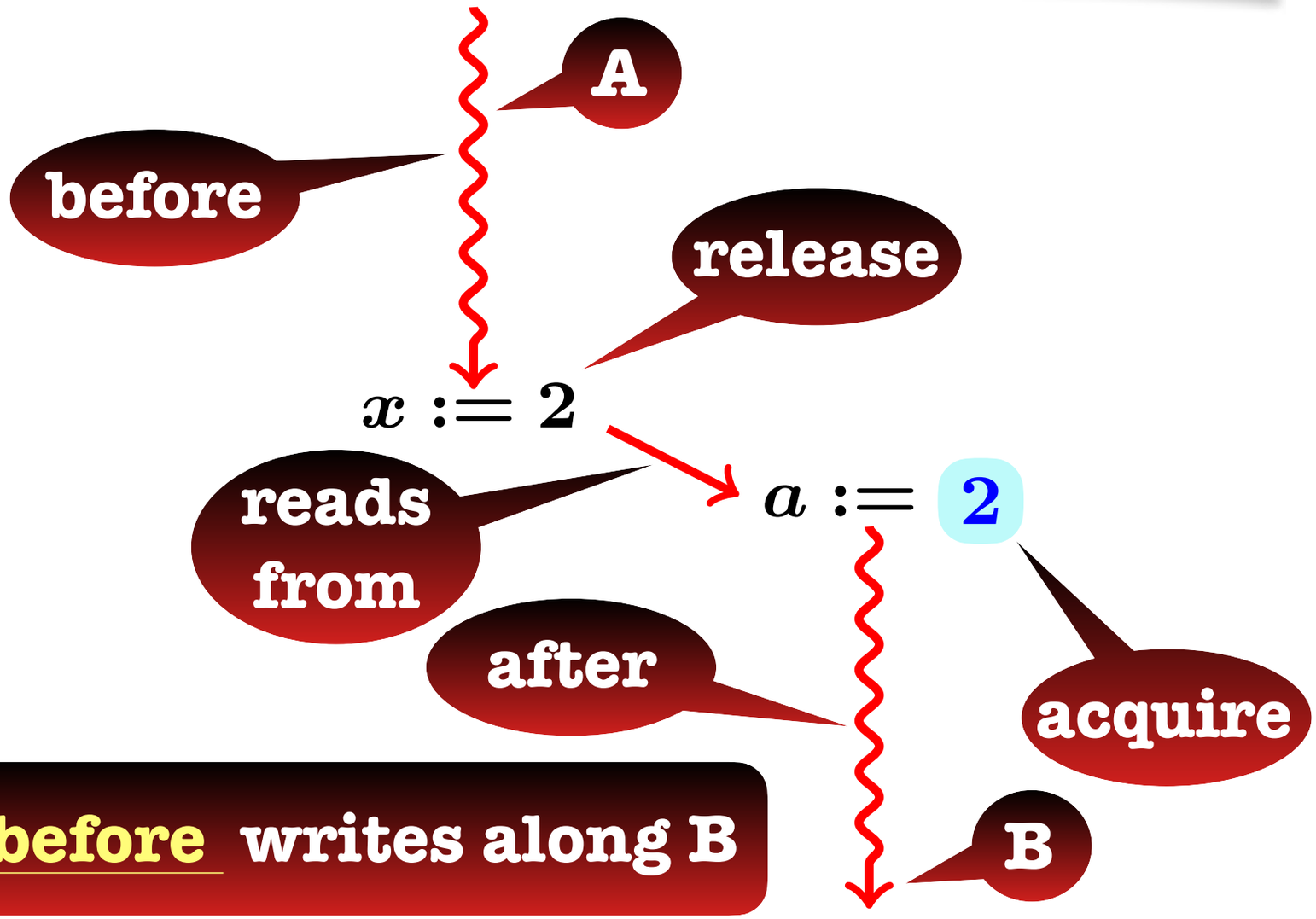
Release-Acquire (RA)

Motivation

- Part of C11 semantics
- Good balance between performance and programmability [Lahav et al , POPL 2016]
  - (much) weaker than SC
  - sufficient for many concurrent algorithms
  - has verified compilation schemes to TSO and Power, ARM, etc.

Semantics

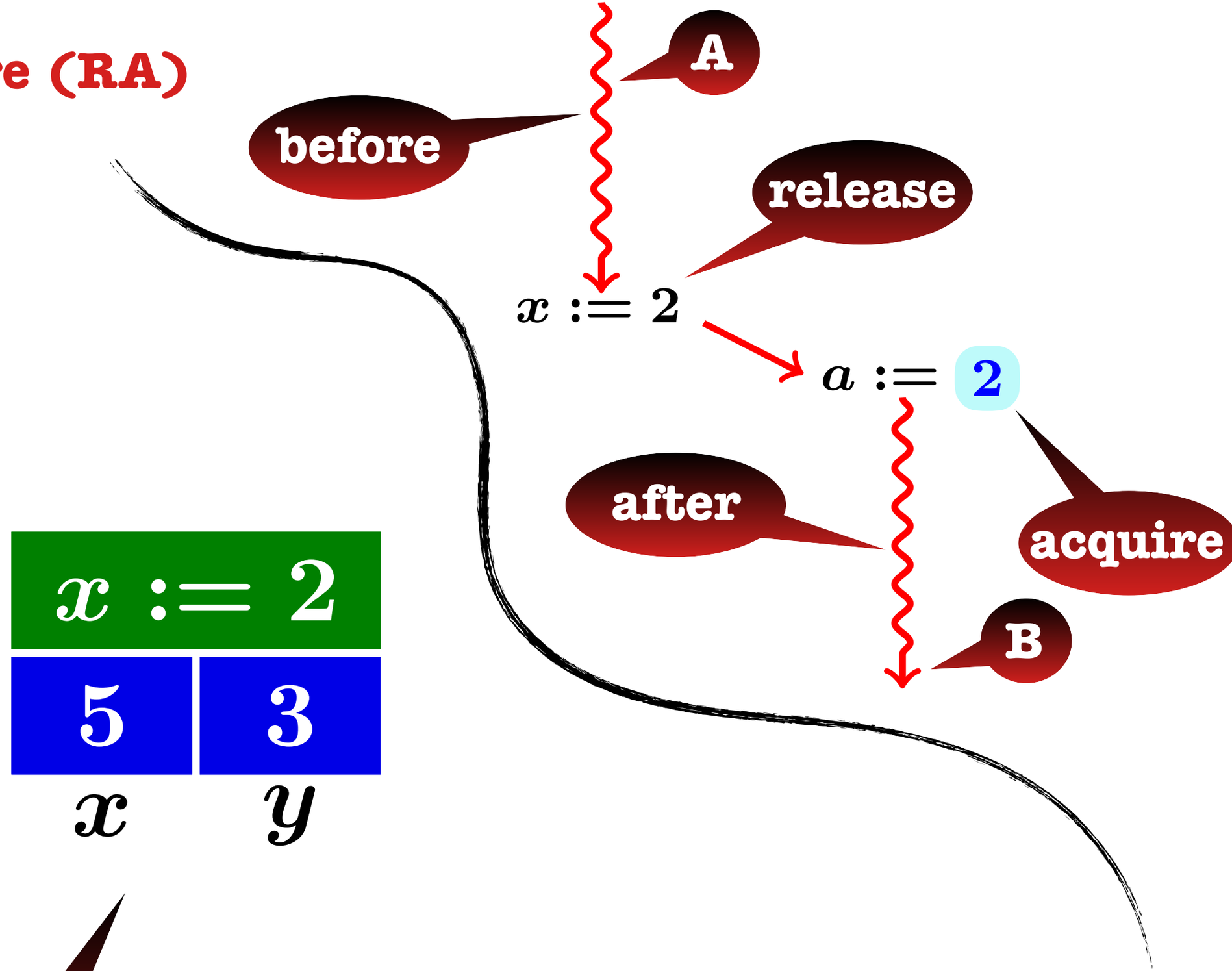
- Program order
- Synchronization
  - write: release
  - read: acquire



A happens before B

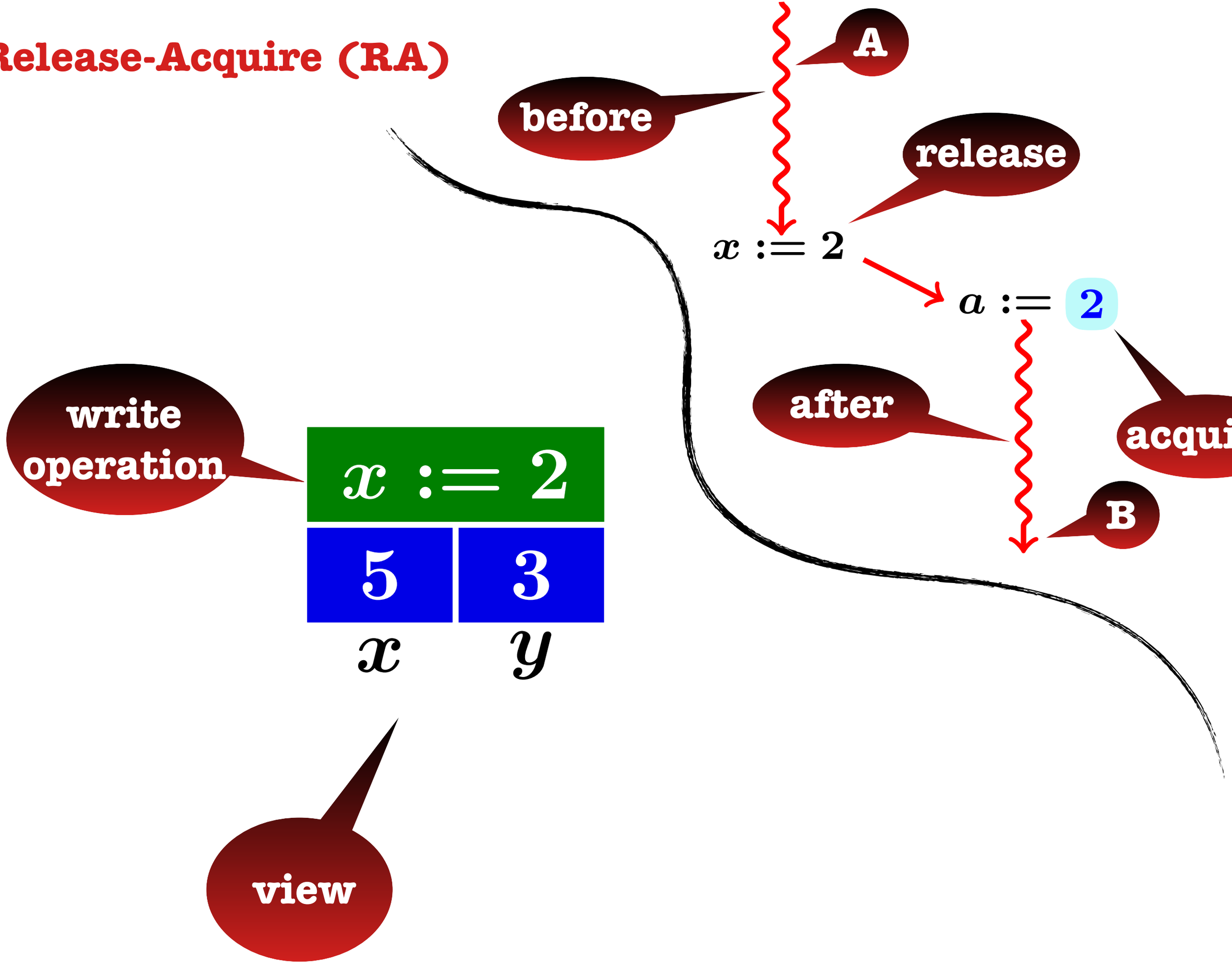
writes along A happen before writes along B

Release-Acquire (RA)

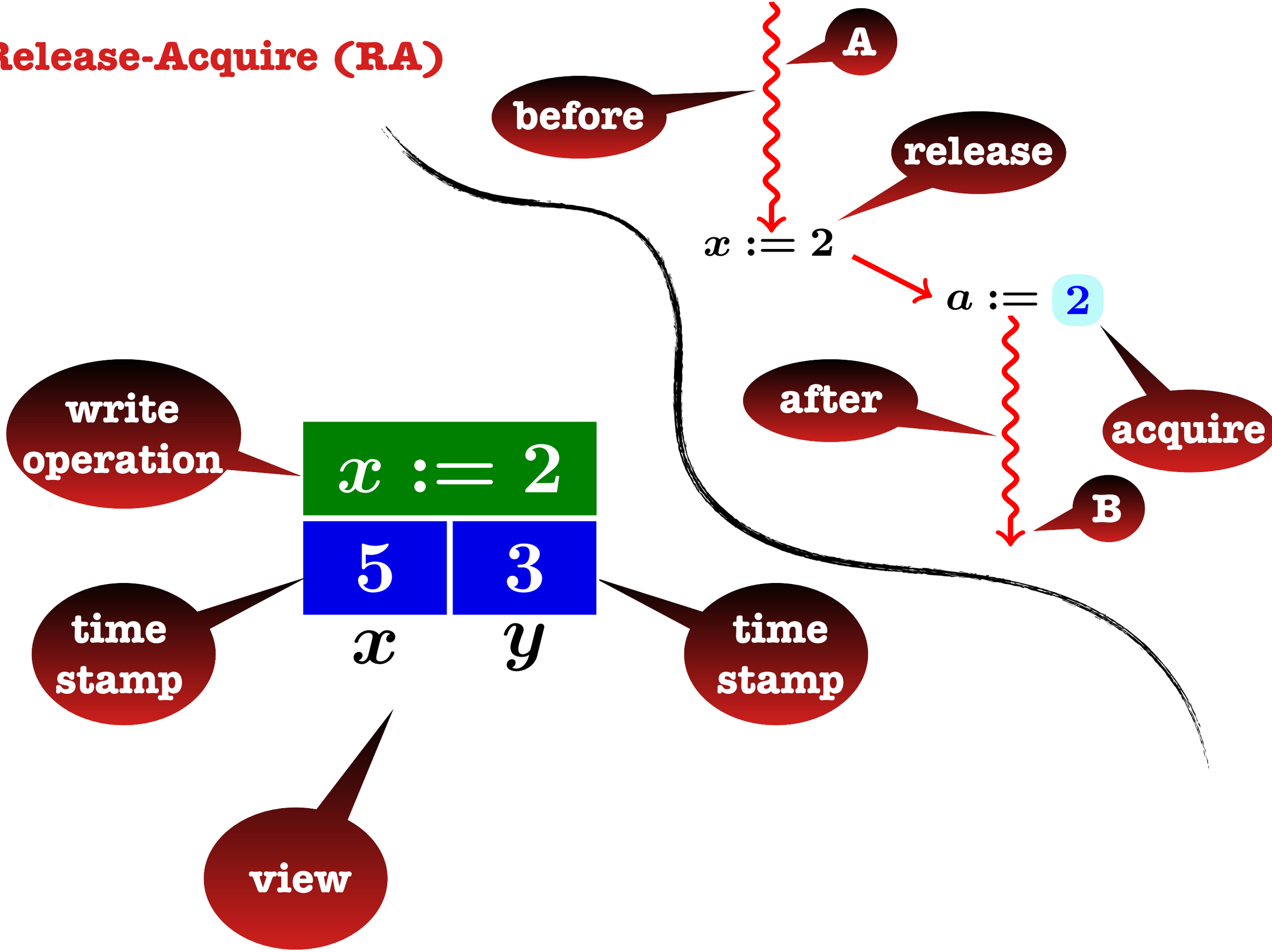


$x := 2$	
5	3
x	y

Release-Acquire (RA)



Release-Acquire (RA)



Release-Acquire (RA)

A

before

release

$x := 2$

$a := 2$

after

acquire

B

write operation

$x := 2$

5	3
x	y

time stamp

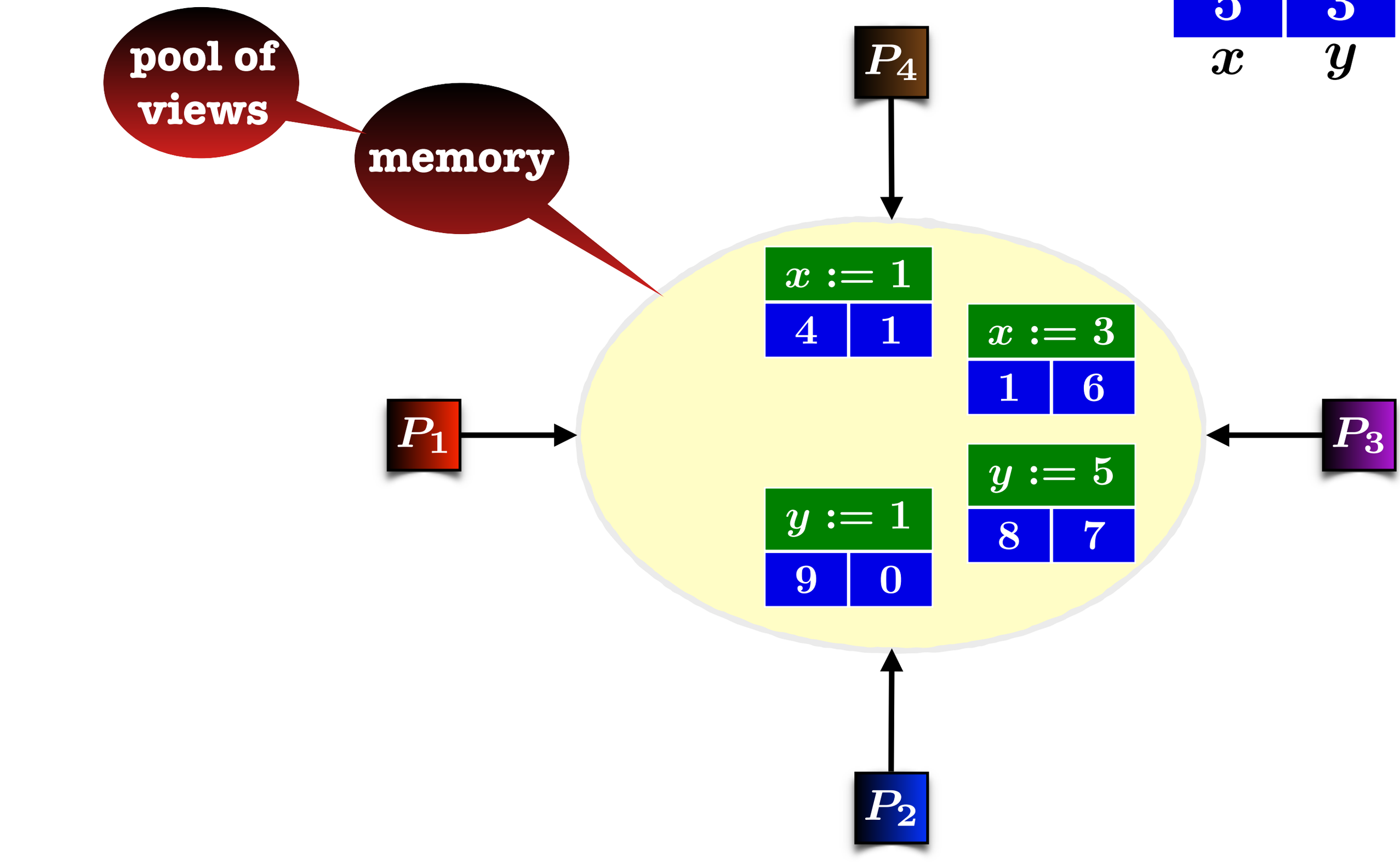
logical clock

view

time stamp

logical clock

Release-Acquire (RA)



Release-Acquire (RA)

pool of views

memory

view

$x := 2$	
5	3
x	y

P_4

$x := 1$	
4	1

$x := 3$	
1	6

$y := 5$	
8	7

$y := 1$	
9	0

P_1

P_3

P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

view

$x := 2$

5	3
x	y

P_4

$x := 1$

4	1

$x := 3$

1	6

$y := 5$

8	7

$y := 1$

9	0

P_1

P_3

P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

view

$x := 2$	
5	3
x	y

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$	
4	1

$x := 3$	
1	6

$y := 5$	
8	7

$y := 1$	
9	0

P_1 P_2 P_3

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

view

$x := 2$	
5	3
x	y

$x := 1$	
4	1

$x := 3$	
1	6

$y := 1$	
9	0

$y := 5$	
8	7

P_1 P_2 P_3

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3	2

local view

Release-Acquire (RA)

view

$x := 2$	
5	3
x	y

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$	
4	1

$x := 3$	
1	6

$y := 1$	
9	0

$y := 5$	
8	7

P_1 P_3 P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3	2

local view

Release-Acquire (RA)

view

$x := 2$	
5	3
x	y

pool of views

memory

Read

1. ~~select view in memory~~
2. variable time stamp $\geq$ yours
3. ~~update local view~~

$x := 1$	
4	1

$x := 3$	
1	6

$y := 1$	
9	0

$y := 5$	
8	7

P_1 P_3 P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3	2

local view

Release-Acquire (RA)

view

$x := 2$	
5	3
x	y

pool of views

memory

Read

1. ~~select view in memory~~
2. variable time stamp $\geq$ yours
3. ~~update local view~~

$x := 1$	
4	1

$x := 3$	
1	6

$y := 1$	
9	0

$y := 5$	
8	7

P_1 P_3 P_2

P_1 : $a := x$

P_1 : $x := 2$

P_1 : $b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

view

$x := 2$	
5	3
x	y

✔

$x := 1$	
4	1

$x := 3$	
1	6

✘

$y := 1$	
9	0

$y := 5$	
8	7

P_1

P_3

P_2

P_1 : $a := x$

P_1 : $x := 2$

P_1 : $b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$	
4	1

$x := 3$	
1	6

$y := 5$	
8	7

$y := 1$	
9	0

P_1 P_2 P_3

P_1 : $a := x$

P_1 : $x := 2$

P_1 : $b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$	
4	1

$x := 3$	
1	6

$y := 5$	
8	7

$y := 1$	
9	0

P_1 P_2 P_3

P_1 : $a := x$

P_1 : $x := 2$

P_1 : $b := y$

3	2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1

P_3

P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3 | 2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

max

P_1

P_3

P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

3 | 2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 4 | 2

P_1 | 3 | 2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

local view

P_1

P_2

P_3

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

max

P_1

P_3

P_2

$P_1 : \quad a := x$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

4 | 2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1

P_2

P_3

P_1 : $a := x$

P_1 : $x := 2$

P_1 : $b := y$

4 | 2

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1

P_2

P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

| 4 | 2 |

local view

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1

P_3

P_2

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

| 4 | 2 |

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

P_4

$x := 1$

4 | 1

$x := 3$

1 | 6

P_3

$y := 5$

8 | 7

$y := 1$

9 | 0

P_1

$P_1 : \quad a := 1$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

4 | 2

local view

P_2

e_1 $\quad a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1

P_3

P_2

$P_1 : \quad a := 1$

$P_1 : \quad x := 2$

$P_1 : \quad b := y$

4 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1

P_3

P_2

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

4 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

4 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

4 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

$x := 2$ | 5 | 2

4 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

$x := 2$ | 5 | 2

5 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

$x := 2$ | 5 | 2

5 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 2$ | 5 | 2

$y := 1$ | 9 | 0

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

P_1

P_2

P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

5 | 2

local view

e_1 $a := 1$

Release-Acquire (RA)

pool of views

memory

Write

1. create new local view
2. variable time stamp:
   i. newer than yours
   ii. not in memory
3. copy new view to memory

$x := 1$ | 4 | 1

$x := 2$ | 5 | 2

$y := 1$ | 9 | 0

$x := 3$ | 1 | 6

$y := 5$ | 8 | 7

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

5 | 2

local view

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

P_4

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

P_1

$x := 2$ | 5 | 2

P_3

$y := 5$ | 8 | 7

$y := 1$ | 9 | 0

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

5 | 2

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$x := 2$ | 5 | 2

$y := 1$ | 9 | 0

$y := 5$ | 8 | 7

P_1 P_2 P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

5 | 2

local view

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

P_1

P_3

$y := 1$	
9	0

$y := 5$	
8	7

5	2

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

P_1

P_3

$y := 1$ | 9 | 0

$y := 5$ | 8 | 7

5 | 2

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

P_1

P_3

$y := 1$ ✗

9	0

$y := 5$ ✓

8	7

5	2

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

P_1

P_3

$y := 5$

8 | 7

5 | 2

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

P_1

P_3

$y := 5$

8 | 7

✔

8 | 7

5 | 2

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

P_1

P_3

$y := 5$

8 | 7

✓

8 | 7

local view

P_2

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$x := 2$ | 5 | 2

$y := 1$ | 9 | 0

$y := 5$ | 8 | 7

P_1

P_2

P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := y$

8 | 7

local view

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$x := 2$ | 5 | 2

$y := 1$ | 9 | 0

$y := 5$ | 8 | 7

P_1

P_2

P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := 5$

8 | 7

local view

e_1 $a := 1$

e_2 $x := 2$

Release-Acquire (RA)

pool of views

memory

Read

1. select view in memory
2. variable time stamp $\geq$ yours
3. update local view

$x := 1$ | 4 | 1

$x := 3$ | 1 | 6

$x := 2$ | 5 | 2

$y := 1$ | 9 | 0

$y := 5$ | 8 | 7

P_1

P_2

P_3

P_1 : $a := 1$

P_1 : $x := 2$

P_1 : $b := 5$

8 | 7

local view

e_1 $a := 1$

e_2 $x := 2$

e_3 $b := 5$

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := y$

$\|$

P_2

$y := 1$

$b := x$

0	0

0	0

assertion $(a = 0) \wedge (b = 0)?$

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1	P_2	
$x := 1$	$y := 1$	0 0
$a := y$	$b := x$	

assertion $(a = 0) \wedge (b = 0)?$

$x := 0$	$y := 0$
0 0	0 0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

$x := 1$
2	0
0	0

P_1

$x := 1$

$a := y$

$\parallel$

P_2

$y := 1$

$b := x$

0	0

assertion $(a = 0) \wedge (b = 0)$?

$x := 0$
0	0

$y := 0$
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

$x := 1$
2	0
2	0

P_1

$x := 1$

$a := y$

$\parallel$

P_2

$y := 1$

$b := x$

0	0

assertion $(a = 0) \wedge (b = 0)$?

$x := 0$
0	0

$y := 0$
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$y := 1$
$a := y$	$b := x$

a	b
2	0
0	0

assertion $(a = 0) \wedge (b = 0)$?

$x := 1$	$x := 0$	$y := 0$
2	0	0
0	0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$y := 1$
$a := y$	$b := x$

x	y	a	b
2	0		
		0	0

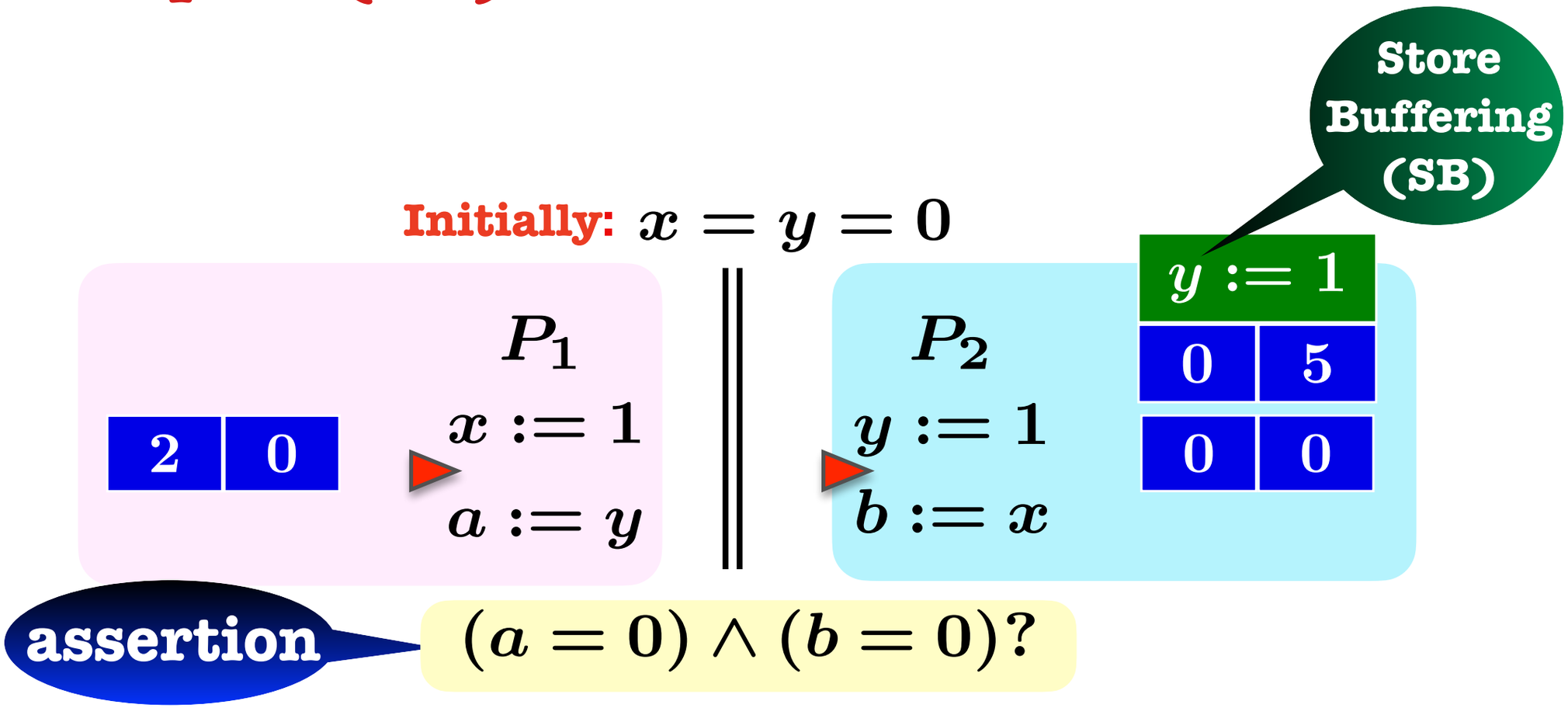
assertion $(a = 0) \wedge (b = 0)$?

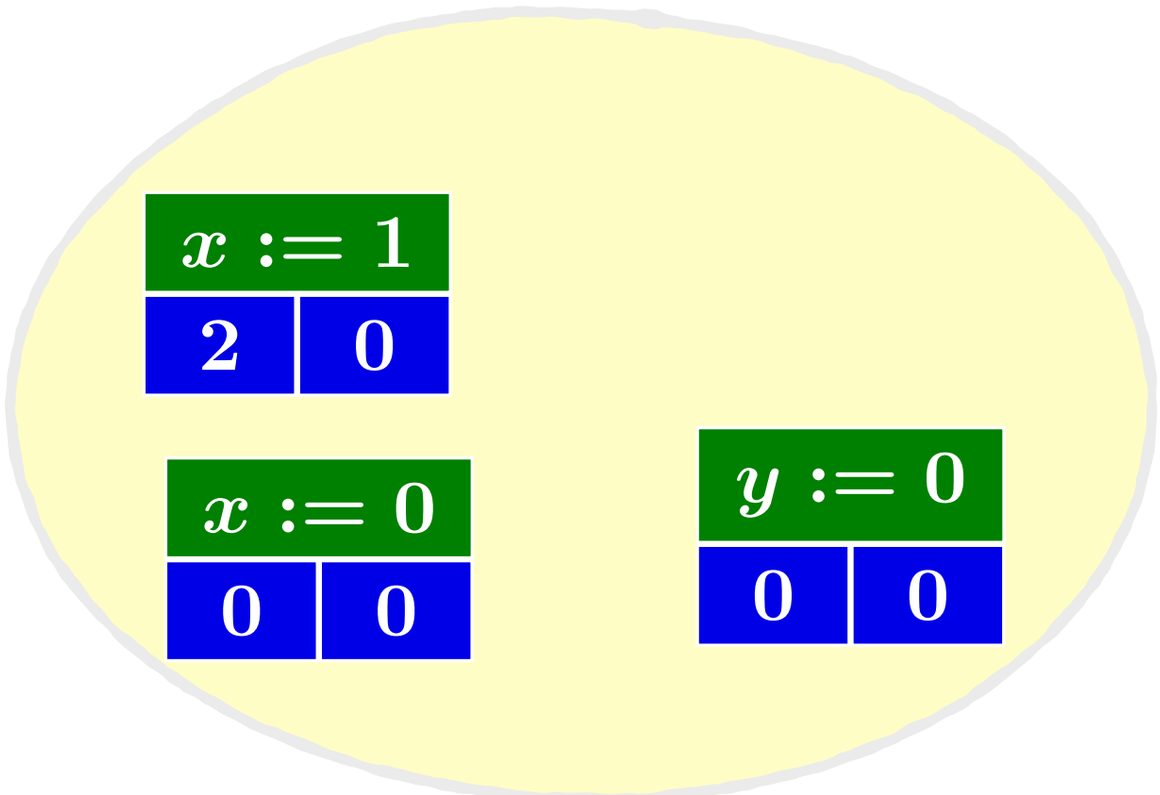
$x := 1$	
2	0

$x := 0$	
0	0

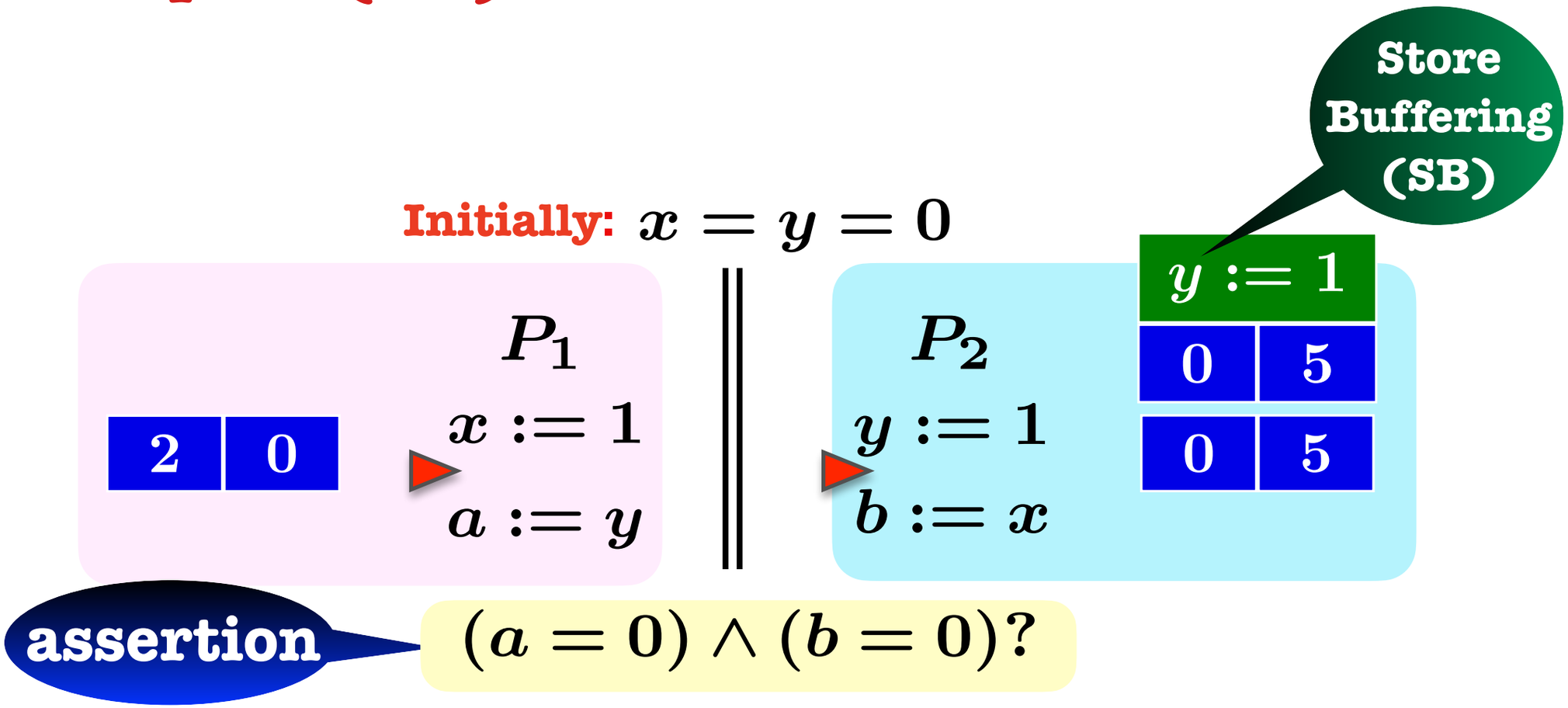
$y := 0$	
0	0

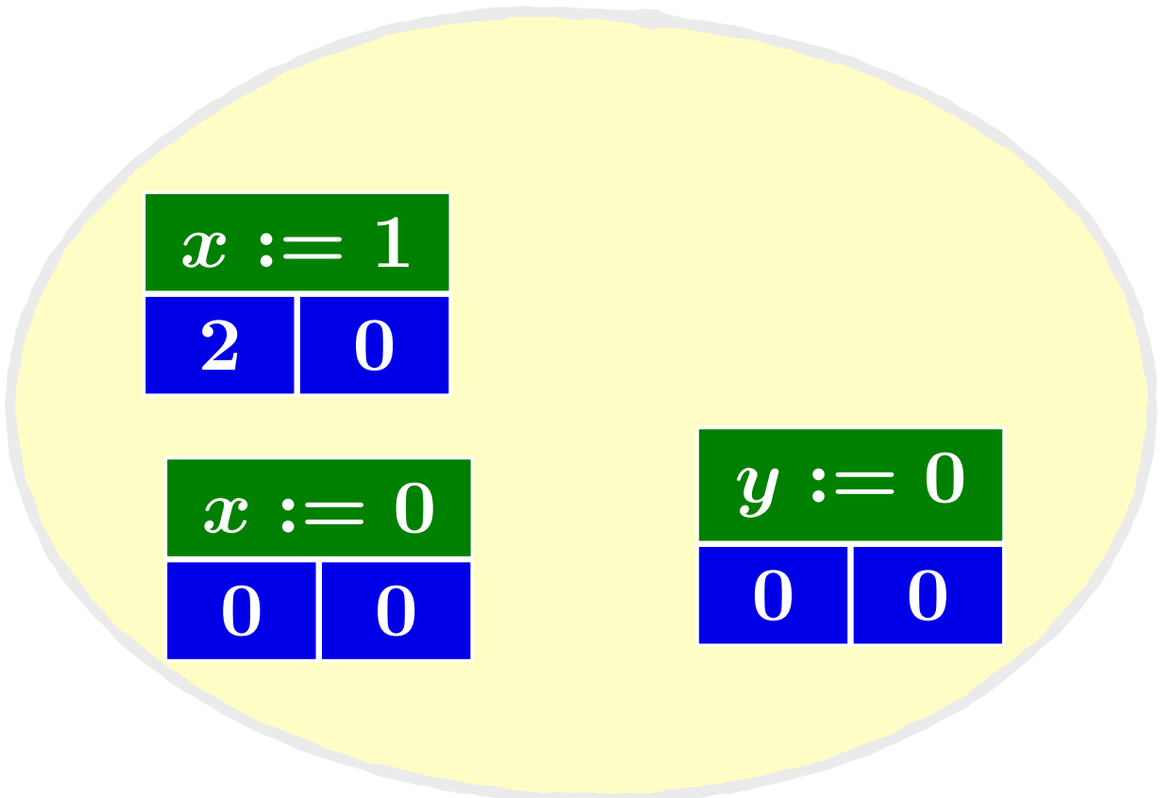
Release-Acquire (RA)





Release-Acquire (RA)





Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$y := 1$
$a := y$	$b := x$

2	0

0	5

assertion $(a = 0) \wedge (b = 0)$?

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := y$

2	0

$\parallel$

P_2

$y := 1$

$b := x$

0	5

assertion $(a = 0) \wedge (b = 0)$?

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := y$

2	0

$\parallel$

P_2

$y := 1$

$b := x$

0	5

assertion $(a = 0) \wedge (b = 0)$?

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y :=$ 0	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := 0$

2	0

$\parallel$

P_2

$y := 1$

$b := x$

0	5

assertion $(a = 0) \wedge (b = 0)$?

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := 0$

2	0

$\parallel$

P_2

$y := 1$

$b := x$

0	5

assertion $(a = 0) \wedge (b = 0)?$

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := 0$

2	0

$\parallel$

P_2

$y := 1$

$b := x$

0	5

assertion $(a = 0) \wedge (b = 0)?$

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := 0$

2	0

$\parallel$

P_2

$y := 1$

$b := 0$

0	5

assertion $(a = 0) \wedge (b = 0)$?

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$y := 1$
$a := 0$	$b := 0$

2	0

0	5

assertion $(a = 0) \wedge (b = 0)?$ **reachable**

$x := 1$	
2	0

$y := 1$	
0	5

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Store Buffering (SB)

Initially: $x = y = 0$

P_1

$x := 1$

$a := 0$

2	0

$\parallel$

P_2

$y := 1$

$b := 0$

0	5

assertion

$(a = 0) \wedge (b = 0)?$

reachable

"Dekker's Protocol faulty under RA"

$x := 1$
2 | 0

$y := 1$
0 | 5

$x := 0$
0 | 0

$y := 0$
0 | 0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

0	0

P_2

$y := 1$

0	0

P_3

$a_1 := x$

$a_2 := y$

0	0

P_4

$b_1 := y$

$b_2 := x$

0	0

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := x$

$a_2 := y$

0	0

P_4

$b_1 := y$

$b_2 := x$

0	0

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := x$

$a_2 := y$

0	0

P_4

$b_1 := y$

$b_2 := x$

0	0

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := x$

$a_2 := y$

0	0

P_4

$b_1 := y$

$b_2 := x$

0	0

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

$a_2 := y$

3	0

P_4

$b_1 := y$

$b_2 := x$

0	0

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

3	0

$a_2 := y$

P_4

$b_1 := y$

0	0

$b_2 := x$

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

3	0

$a_2 := y$

P_4

$b_1 := y$

0	0

$b_2 := x$

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

$a_2 := 0$

3	0

P_4

$b_1 := y$

$b_2 := x$

0	0

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

3	0

$a_2 := 0$

P_4

$b_1 := y$

0	0

$b_2 := x$

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

3	0

$a_2 := 0$

P_4

$b_1 := 1$

0	4

$b_2 := x$

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

IRIW

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

$a_2 := 0$

3	0

P_4

$b_1 := 1$

$b_2 := 0$

0	4

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

reachable

$x := 1$	
3	0

$y := 1$	
0	4

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Initially: $x = y = 0$

P_1

$x := 1$

3	0

P_2

$y := 1$

0	4

P_3

$a_1 := 1$

3	0

$a_2 := 0$

P_4

$b_1 := 1$

0	4

$b_2 := 0$

$(a_1 = 1) \wedge (a_2 = 0) \wedge$
$(b_1 = 1) \wedge (b_2 = 0)?$

reachable

IRIW

“write events can be observed in different orders by different processes under RA”

$x := 1$	
3	0

$y := 1$	
0	4

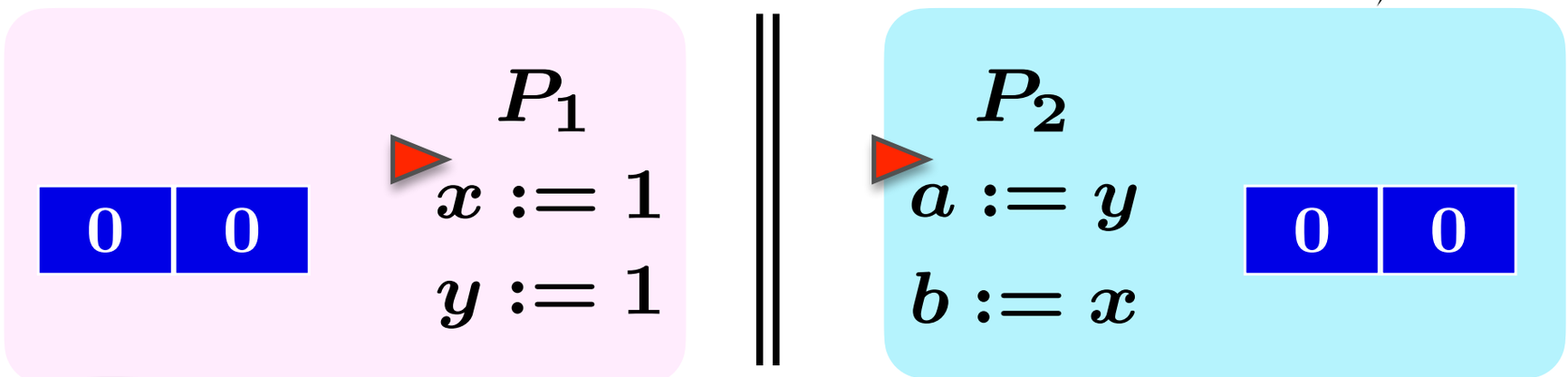
$x := 0$	
0	0

$y := 0$	
0	0

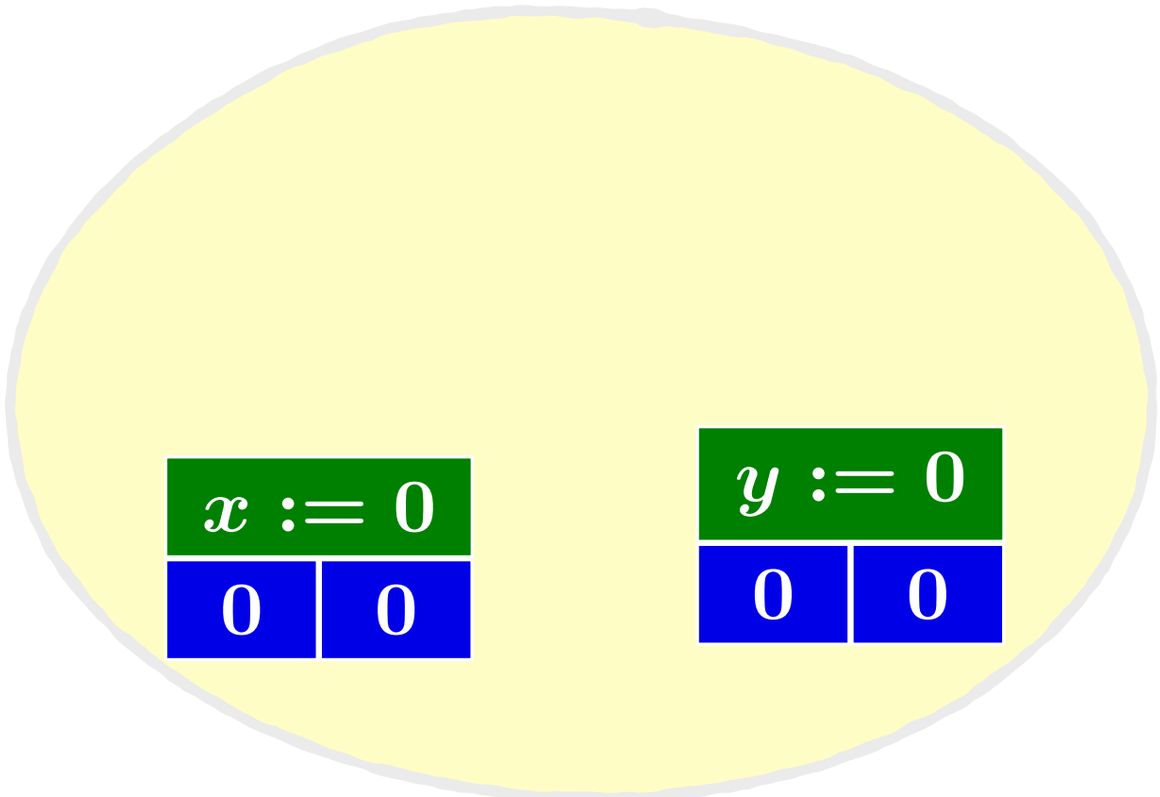
Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$



assertion $(a = 1) \wedge (b = 0)?$



Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

$x := 1$	
3	0
0	0

P_1

$x := 1$

$y := 1$

$\parallel$

P_2

$a := y$

$b := x$

0	0

assertion $(a = 1) \wedge (b = 0)$?

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

$x := 1$	
3	0
3	0

P_1

$x := 1$

$y := 1$

$\parallel$

P_2

$a := y$

$b := x$

0	0

assertion $(a = 1) \wedge (b = 0)$?

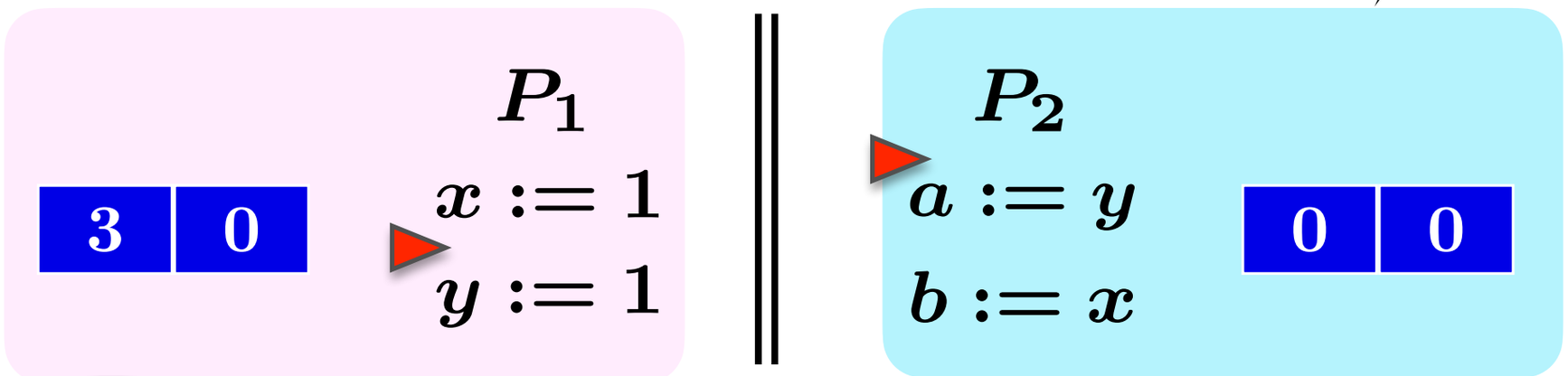
$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

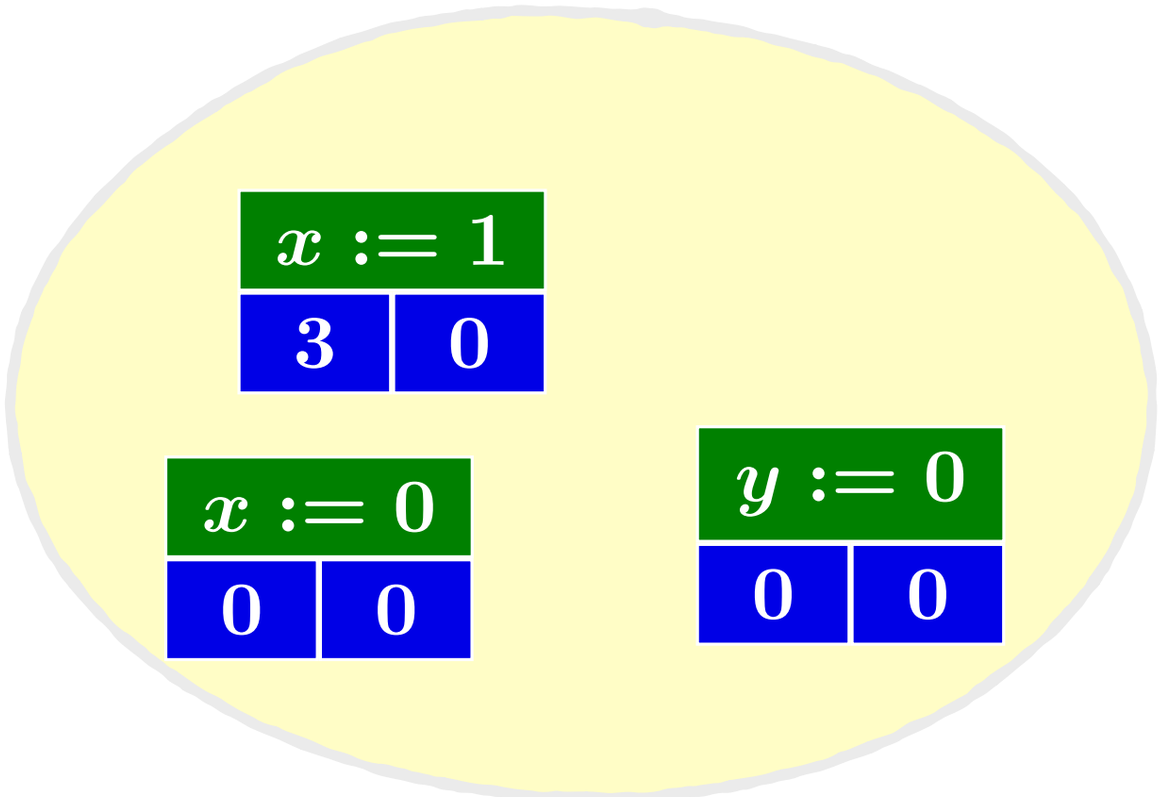
Massage Passing (MP)

Initially: $x = y = 0$



assertion

$(a = 1) \wedge (b = 0)?$



Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 0

$\parallel$

P_2

$a := y$

$b := x$

0 | 0

assertion

$(a = 1) \wedge (b = 0)?$

$x := 1$

3 | 0

$x := 0$

0 | 0

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

$y := 1$	
3	2
3	0

P_1

$x := 1$

$y := 1$

$\parallel$

P_2

$a := y$

$b := x$

0	0

assertion

$(a = 1) \wedge (b = 0)?$

$x := 1$	
3	0

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

$y := 1$	
3	2
3	2

P_1

$x := 1$

$y := 1$

$\|$

P_2

$a := y$

$b := x$

0	0

assertion

$(a = 1) \wedge (b = 0)$?

$x := 1$	
3	0

$x := 0$	
0	0

$y := 0$	
0	0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$a := y$
$y := 1$	$b := x$

3 | 2

0 | 0

assertion $(a = 1) \wedge (b = 0)?$

$x := 1$	$y := 1$
3 | 0	3 | 2

$x := 0$	$y := 0$
0 | 0	0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := y$

$b := x$

0 | 0

assertion $(a = 1) \wedge (b = 0)?$

$x := 1$

3 | 0

$y := 1$

3 | 2

$x := 0$

0 | 0

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$a := y$
$y := 1$	$b := x$

3 2 | 0 0

assertion: $(a = 1) \wedge (b = 0)$?

$x := 1$: 3 0

$y := 1$: 3 2

$x := 0$: 0 0

$y := 0$: 0 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := y$

$b := x$

0 | 0

assertion $(a = 1) \wedge (b = 0)?$

$x := 1$

3 | 0

$y := 1$

3 | 2

$x := 0$

0 | 0

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1	P_2
$x := 1$	$a := 1$
$y := 1$	$b := x$

3 2 | 0 0

assertion $(a = 1) \wedge (b = 0)?$

$x := 1$: 3 0

$y := 1$: 3 2

$x := 0$: 0 0

$y := 0$: 0 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := 1$

$b := x$

3 | 2

assertion

$(a = 1) \wedge (b = 0)?$

$x := 1$

3 | 0

$y := 1$

3 | 2

$x := 0$

0 | 0

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := 1$

$b := x$

3 | 2

assertion

$(a = 1) \wedge (b = 0)?$

$x := 1$

3 | 0

$y := 1$

3 | 2

$x := 0$

0 | 0

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := 1$

$b := x$

3 | 2

assertion

$(a = 1) \wedge (b = 0)?$

$x := 1$: 3 | 0

$y := 1$: 3 | 2

$x := 0$: 0 | 0

$y := 0$: 0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := 1$

$b := x$

3 | 2

assertion

$(a = 1) \wedge (b = 0)?$

$x := 1$

3 | 0

$y := 1$

3 | 2

$x := 0$

0 | 0

✗

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

P_1

$x := 1$

$y := 1$

3 2

$\|$

P_2

$a := 1$

$b := x$

3 2

assertion

$(a = 1) \wedge (b = 0)?$

unreachable

$x := 1$

3 0

$y := 1$

3 2

$x := 0$

0 0

✗

$y := 0$

0 0

Release-Acquire (RA)

Massage Passing (MP)

Initially: $x = y = 0$

acquire

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := 1$

$b := x$

3 | 2

assertion

$(a = 1) \wedge (b = 0)?$

unreachable

release

$x := 1$

3 | 0

$y := 1$

3 | 2

$x := 0$

0 | 0

✗

$y := 0$

0 | 0

Release-Acquire (RA)

Massage Passing (MP)

before

Initially: $x = y = 0$

acquire

P_1

$x := 1$

$y := 1$

3 | 2

$\parallel$

P_2

$a := 1$

$b := x$

3 | 2

assertion

$(a = 1) \wedge (b = 0)$?

release

after

unreachable

$x := 1$ | 3 | 0

$y := 1$ | 3 | 2

$x := 0$ | 0 | 0 ✗

$y := 0$ | 0 | 0

Outline

- **Memory Models**
  - **SC (Sequential Consistency**
  - **RA (Release-Acquire)**
- **Verification**
  - **Dynamic Partial-Order Reduction (DPOR)**
  - **Experiments**
- **Conclusions**

P_1

```
while (1)
        write: x=1
```

P_1

```
while (1)
        write: x=1
```

$$x := 1$$
$$x := 1$$
$$x := 1$$
$$\dots$$
$$x := 1$$

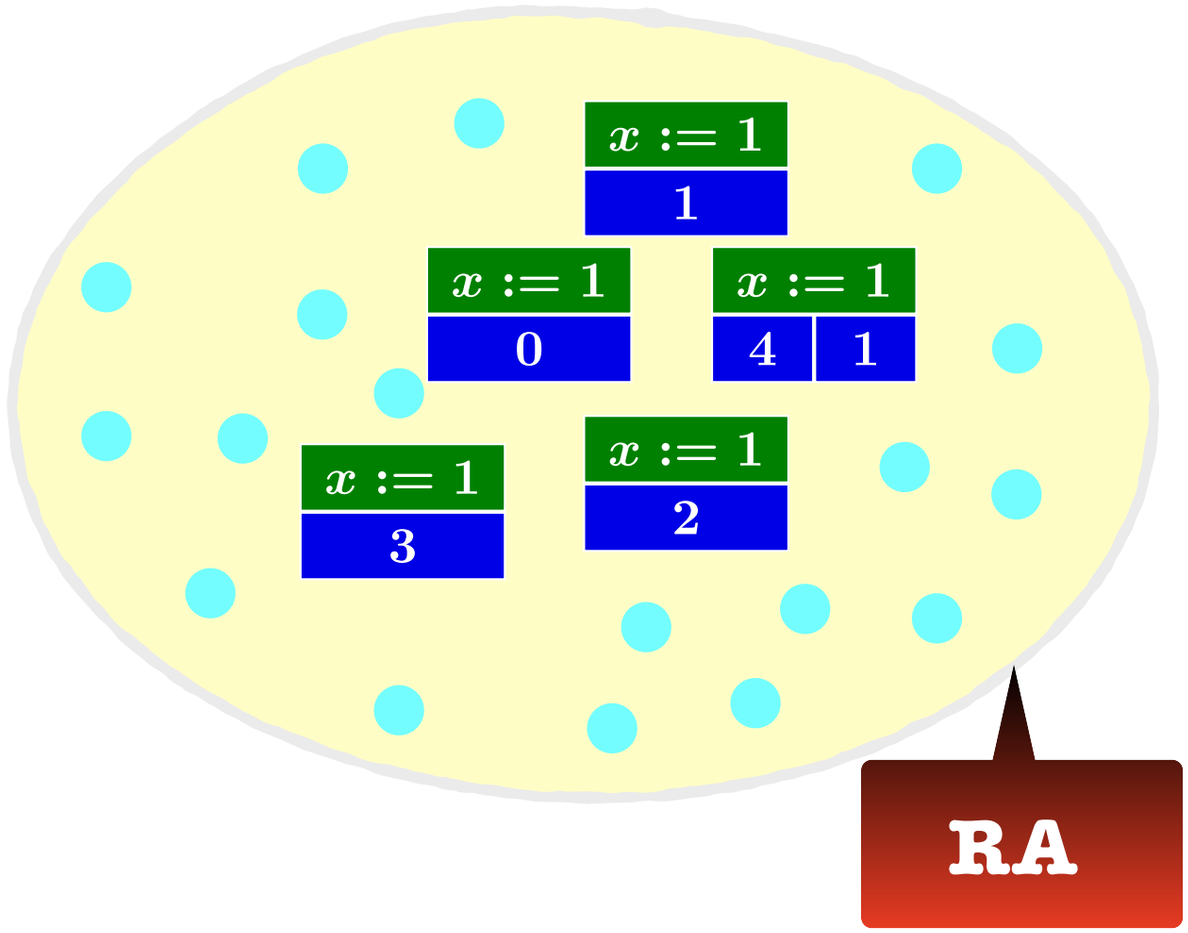
$$\dots$$

P_1

while (1)
write: x=1

$x := 1$
$x := 1$
$x := 1$
...
$x := 1$

...



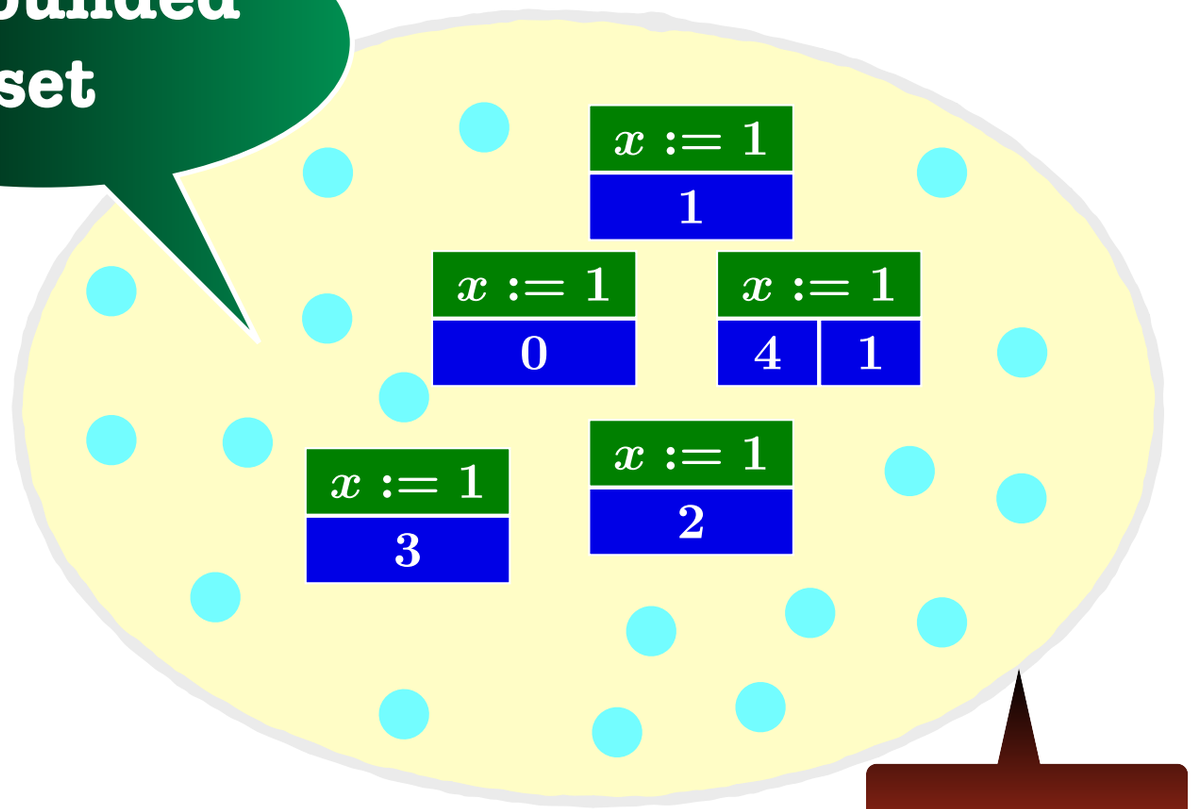
P_1

while (1)
write: x=1

$x := 1$
$x := 1$
$x := 1$
...
$x := 1$

...

unbounded set



RA

P_1

while (1)
write: x=1

$x := 1$
$x := 1$
$x := 1$
...
$x := 1$

...

unbounded set

infinite state space

$x := 1$ | 1

$x := 1$ | 0

$x := 1$ | 4 | 1

$x := 1$ | 3

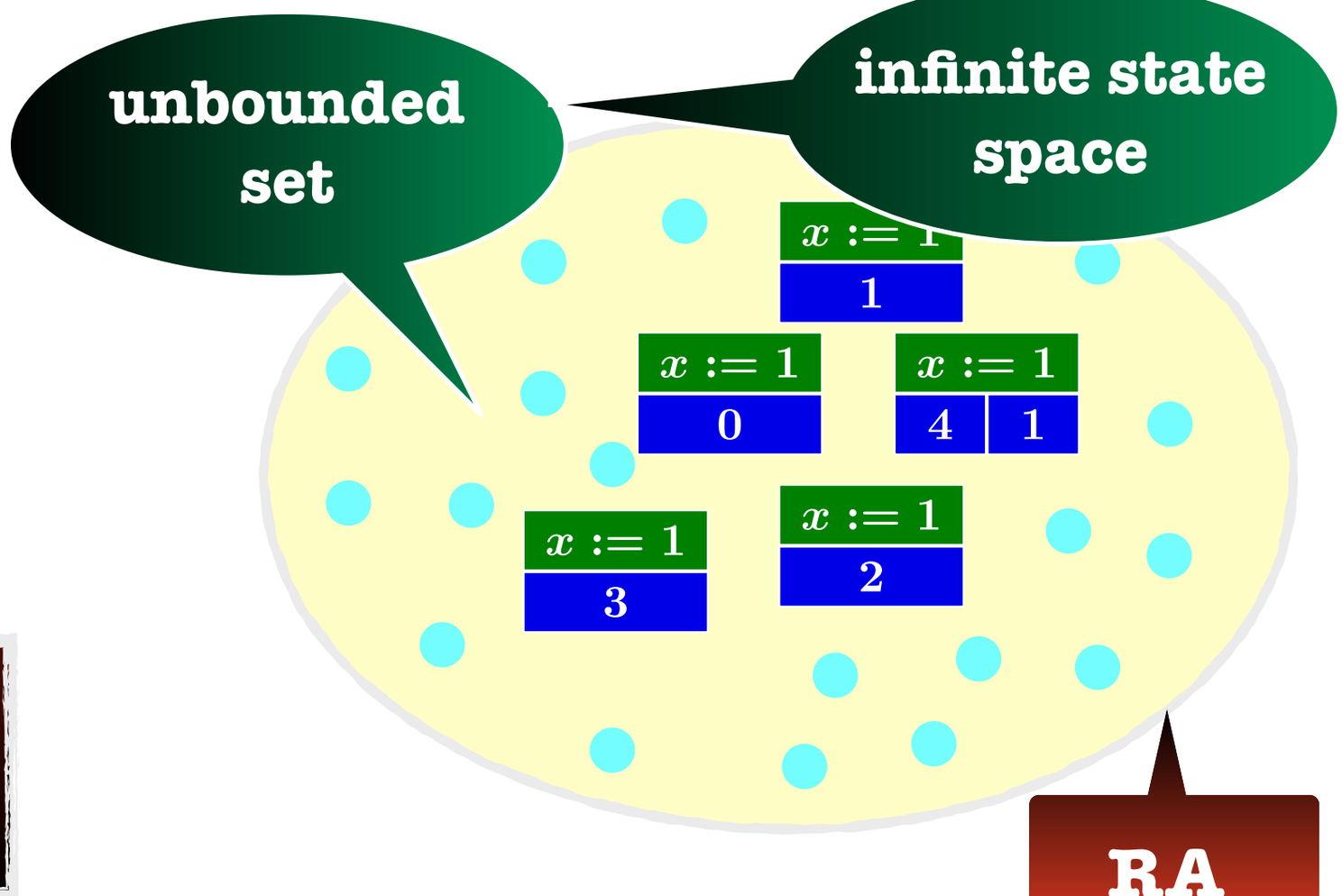
$x := 1$ | 2

RA

Decidability and Complexity

$x := 1$

Reachability Problem



Decidability and Complexity

$x := 1$

Reachability Problem

unbounded set

infinite state space

$x := 1$ | 1

$x := 1$ | 0

$x := 1$ | 4 | 1

Undecidable
(PLDI'19)

RA

Decidability and Comple

Reachal Probl

unbounded set

infinite state space

$x := 1$

1

Approximation Techniques

- Over-Approximation
  - Abstraction
- Under-approximation
  - Bounded Context-Switch (PLDI'19)
  - DPOR (OOPSLA'18)

unbounded set

infinite state space

$x := 1$

1

Decidability and Comple

Reachal Probl

Approximation Techniques

- Over-Approximation
  - Abstraction
- Under-approximation
  - Bounded Context-Switch (PLDI'19)
  - DPOR (OOPSLA'18)

Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$

$(a = 1) \wedge (b = 2)$?

assertion

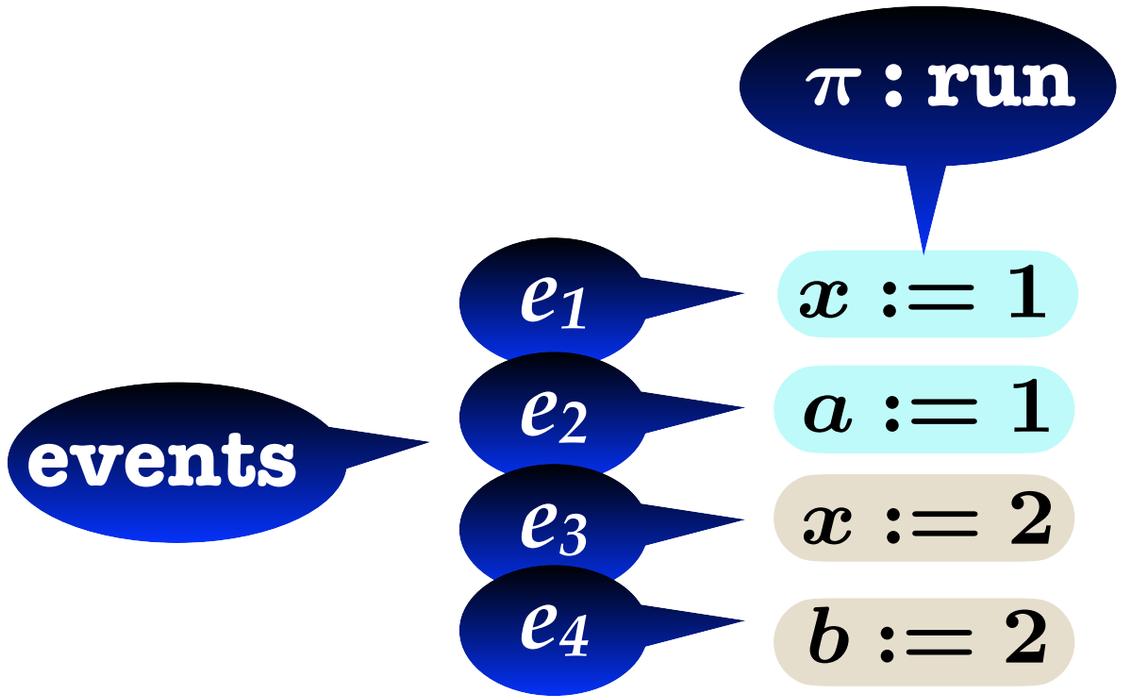
- Threads/Processes:
  - Concurrent
  - Loop-Free
  - Deterministic
- Shared Variables: x,y
- Local Variables: a,b,c,d
- Interleaving Semantics

Concurrent Programs (under SC)

Initially $x = 0$

th_1	th_2
$x := 1$	$x := 2$
$a := x$	$b := x$

$(a = 1) \wedge (b = 2)$?



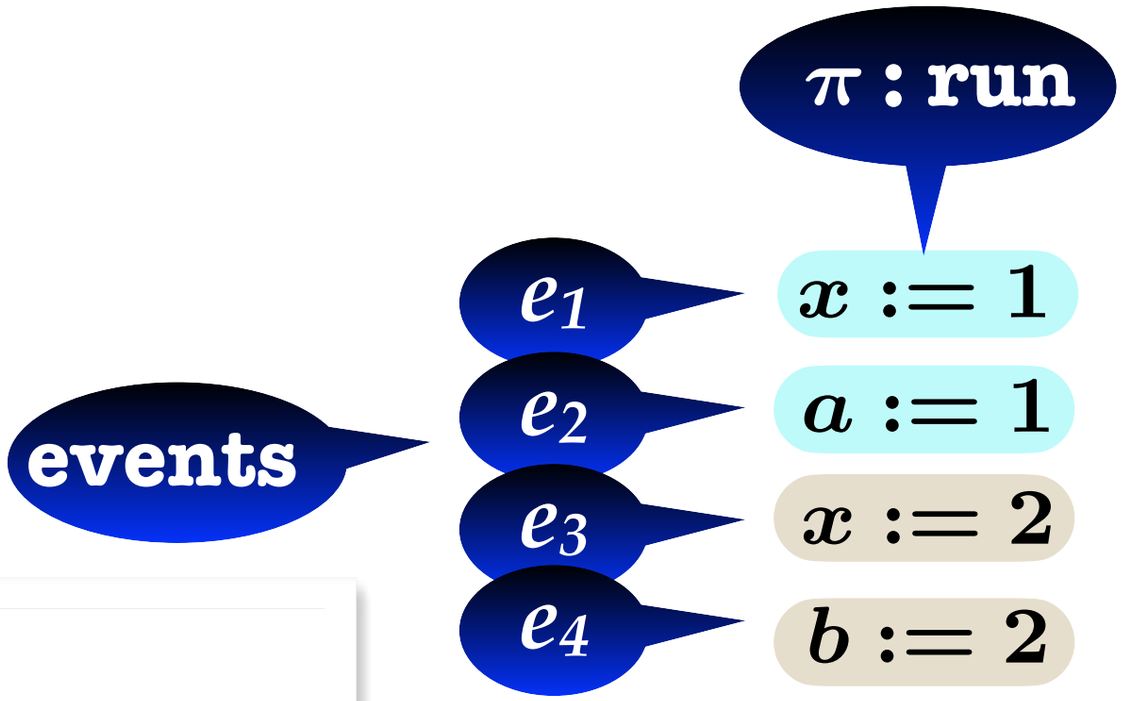
assertion

- Threads/Processes:
  - Concurrent
  - Loop-Free
  - Deterministic
- Shared Variables: x,y
- Local Variables: a,b,c,d
- Interleaving Semantics

Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



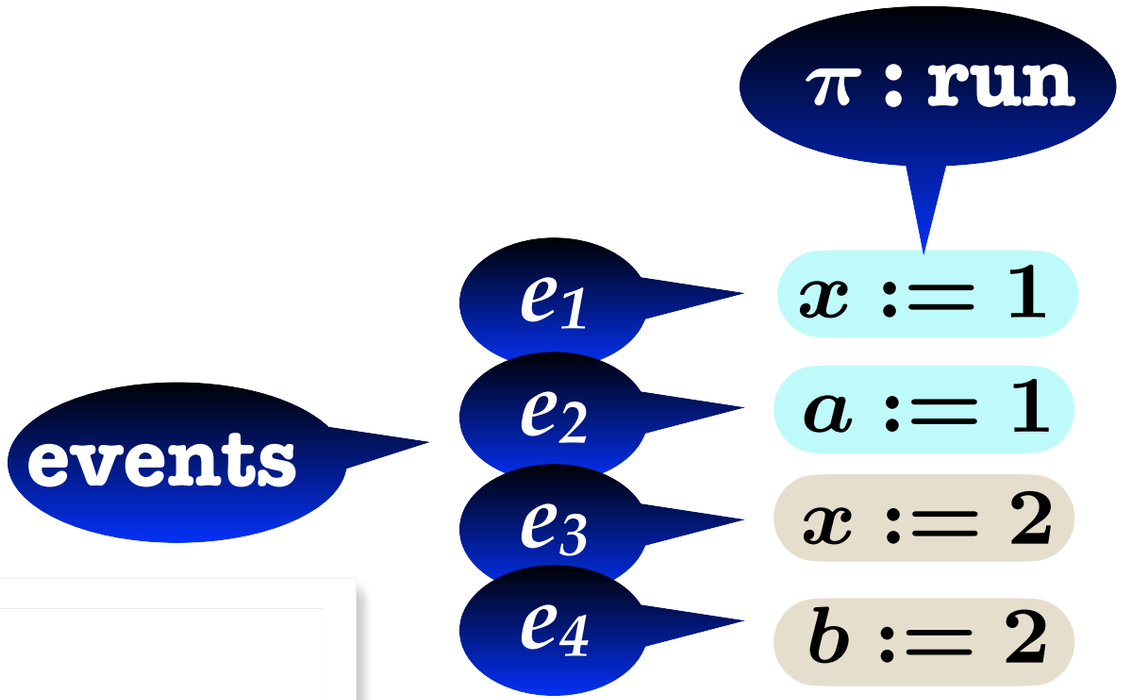
- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

τ : trace

Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

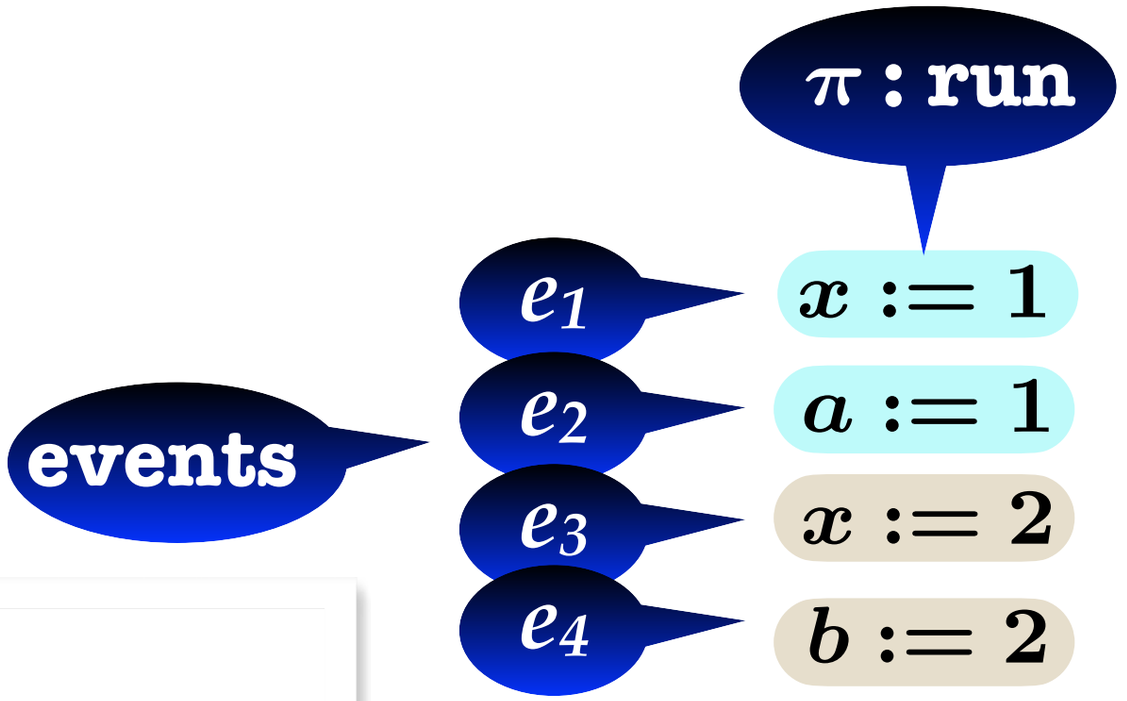
τ : trace

$x := 1$

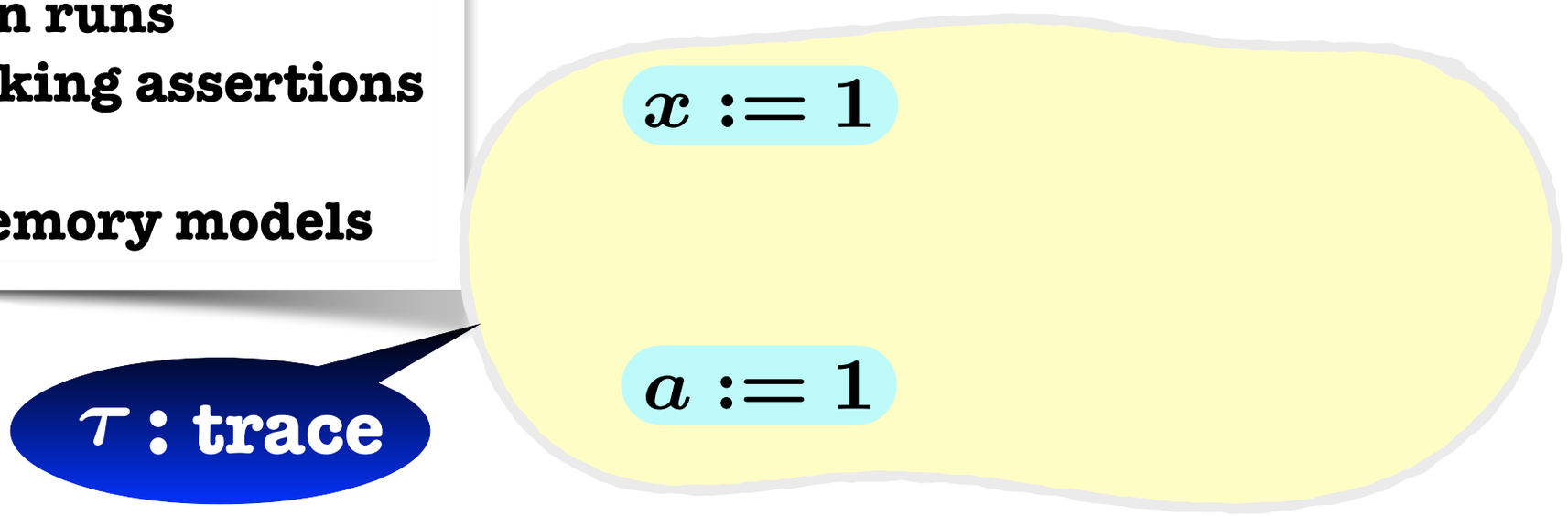
Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



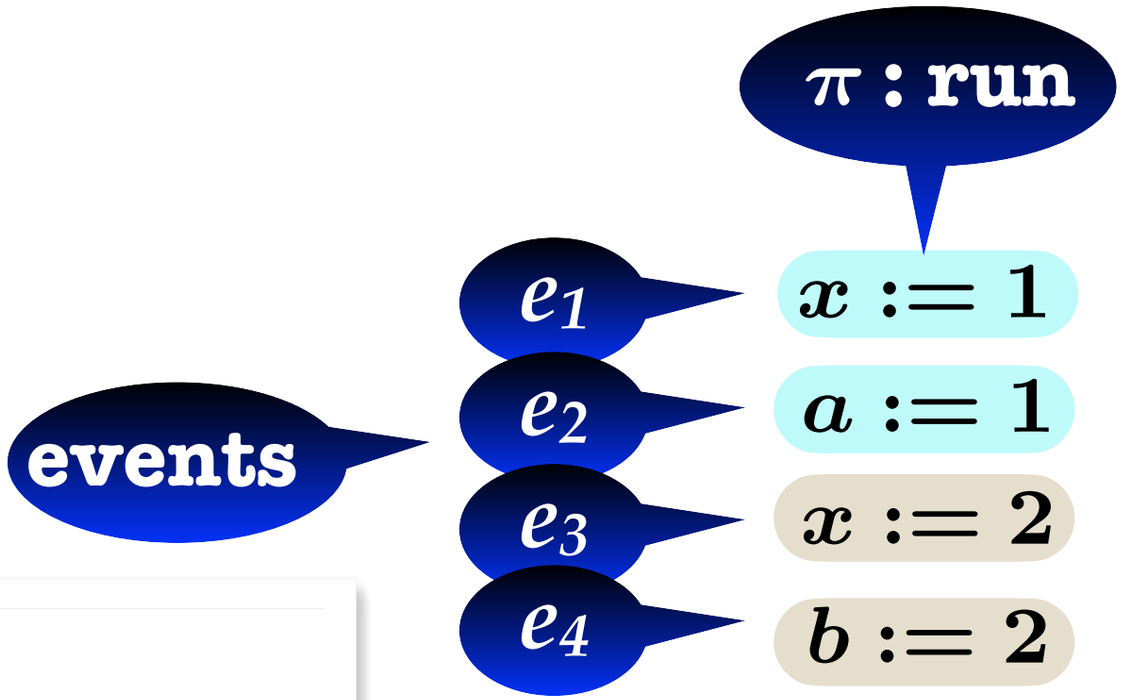
- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**



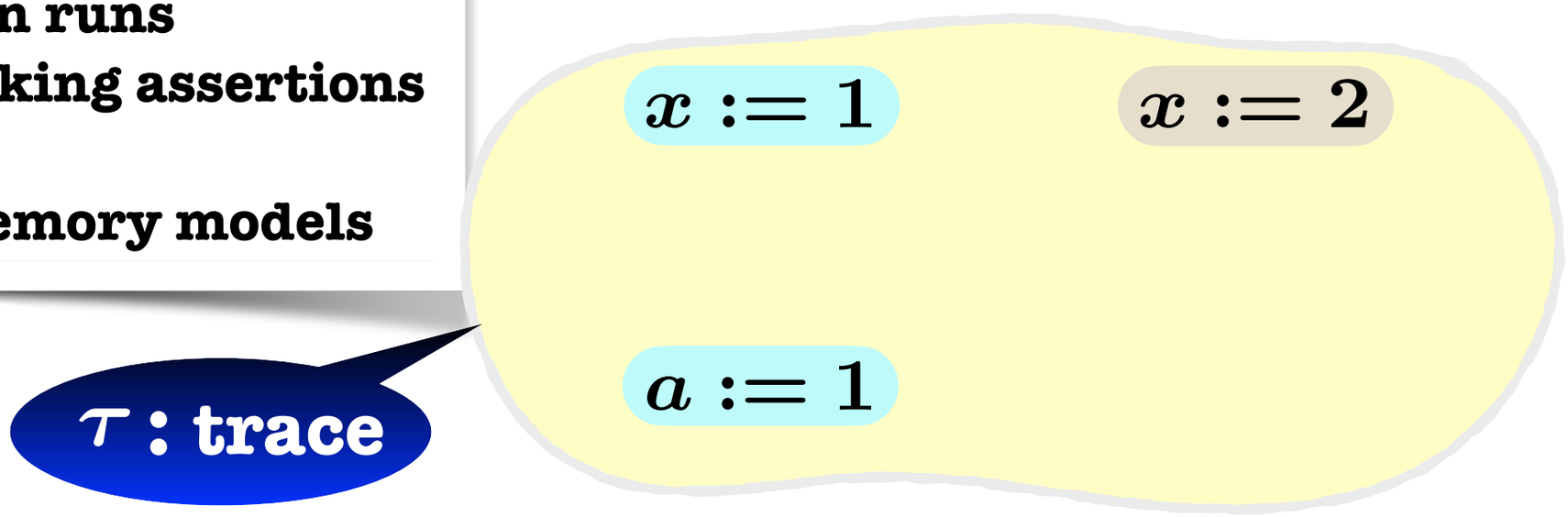
Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



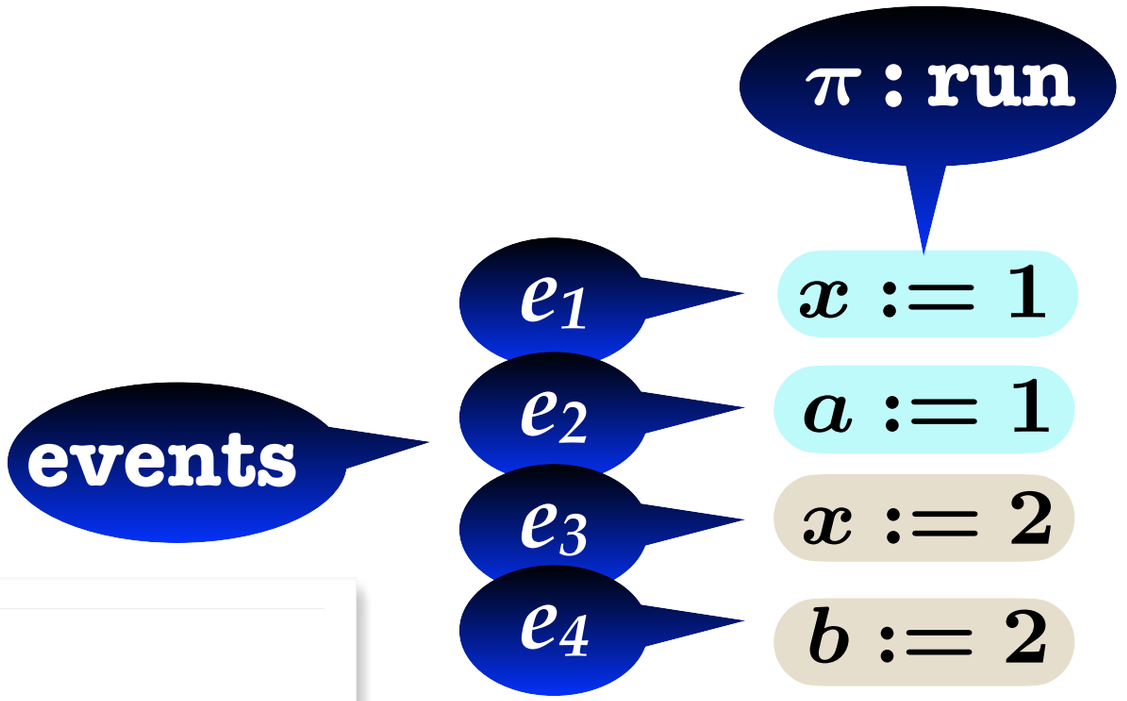
- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
- **abstract:**
  - **allow different memory models**



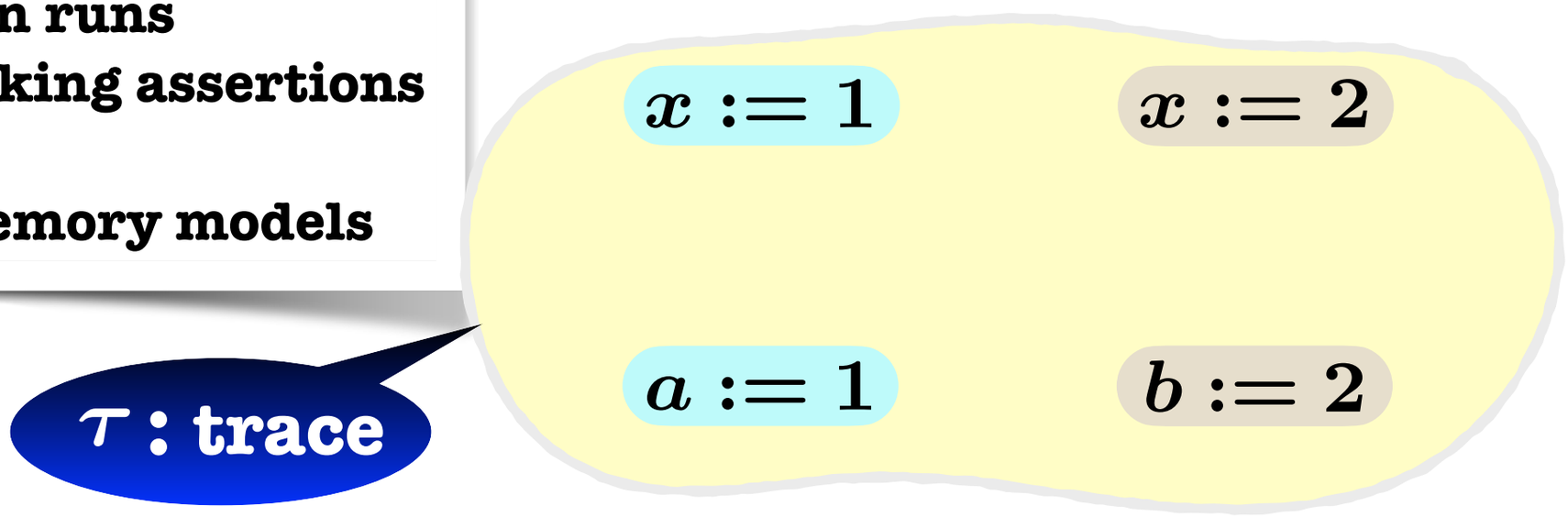
Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



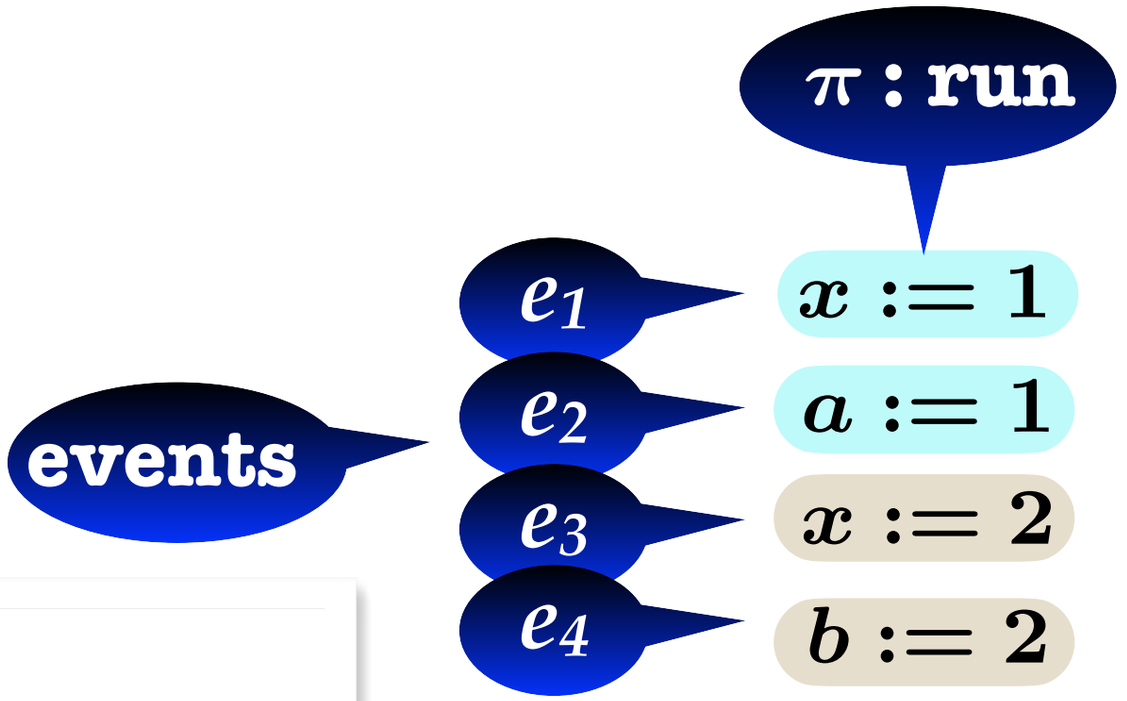
- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
- **abstract:**
  - **allow different memory models**



Concurrent Programs (under SC)

Initially $x = 0$

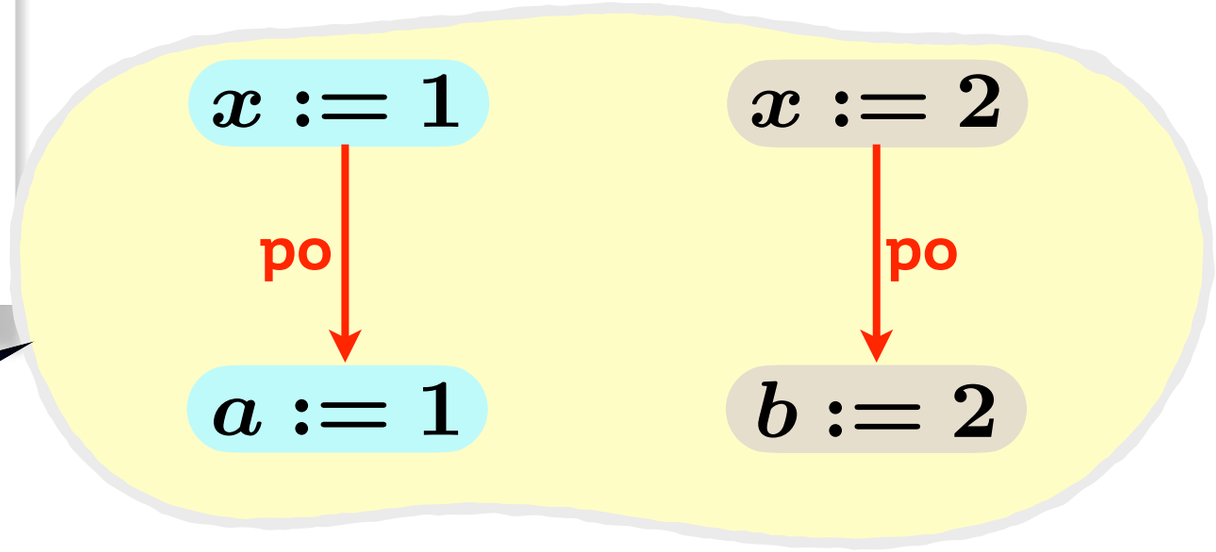
th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order

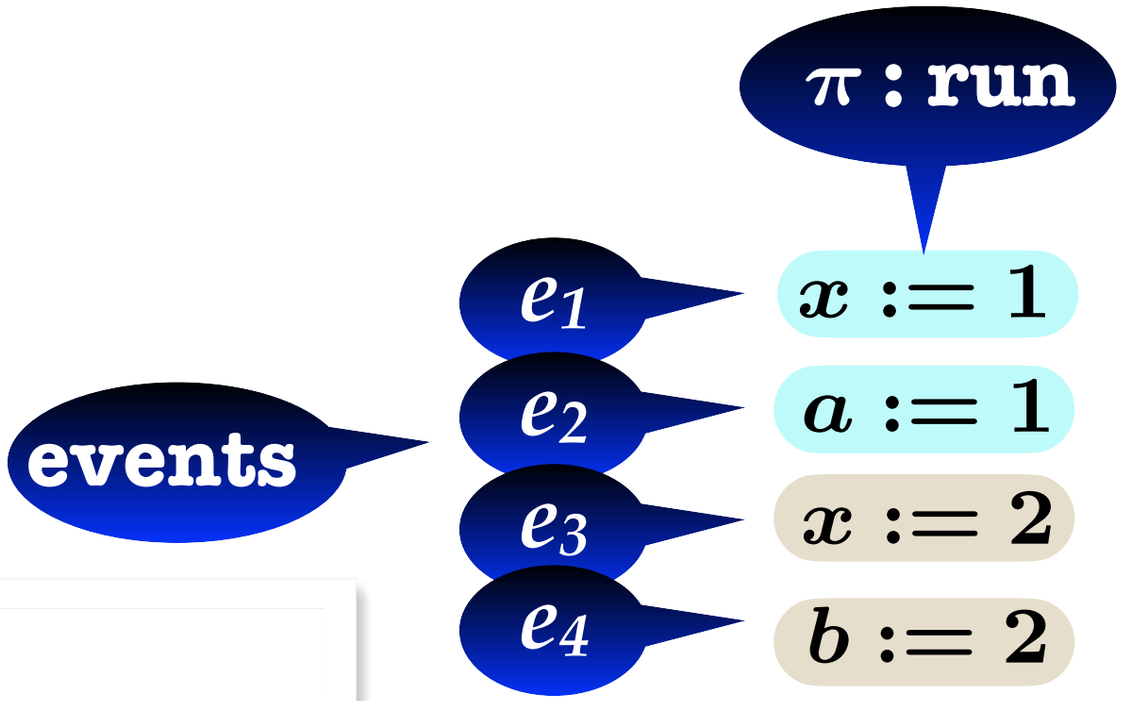
τ : trace



Concurrent Programs (under SC)

Initially $x = 0$

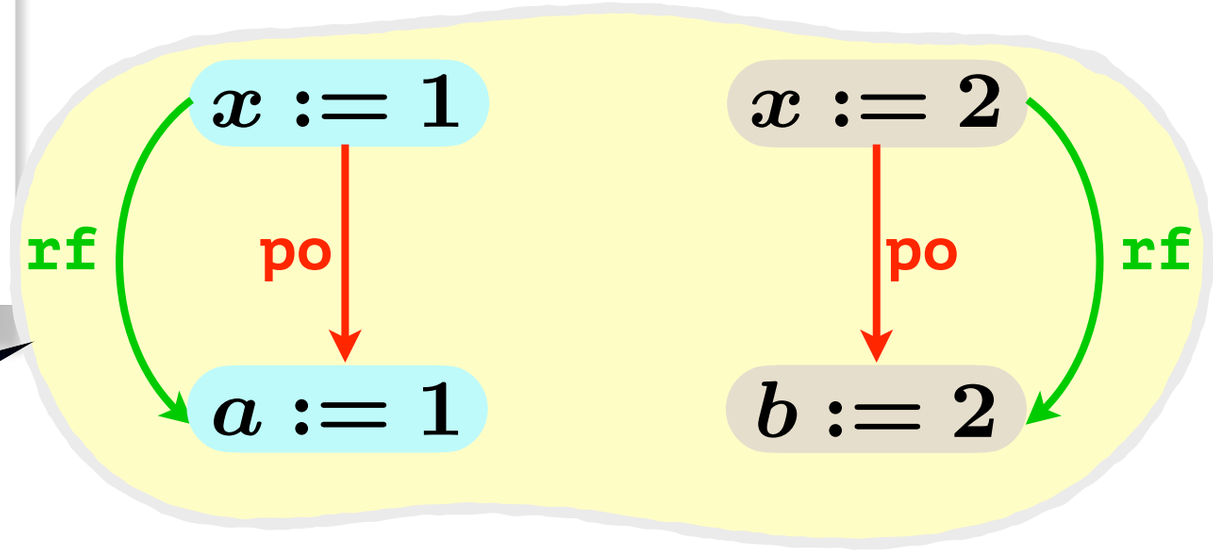
th_1	th_2
$x := 1$	$x := 2$
$a := x$	$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order
- rf: read-from

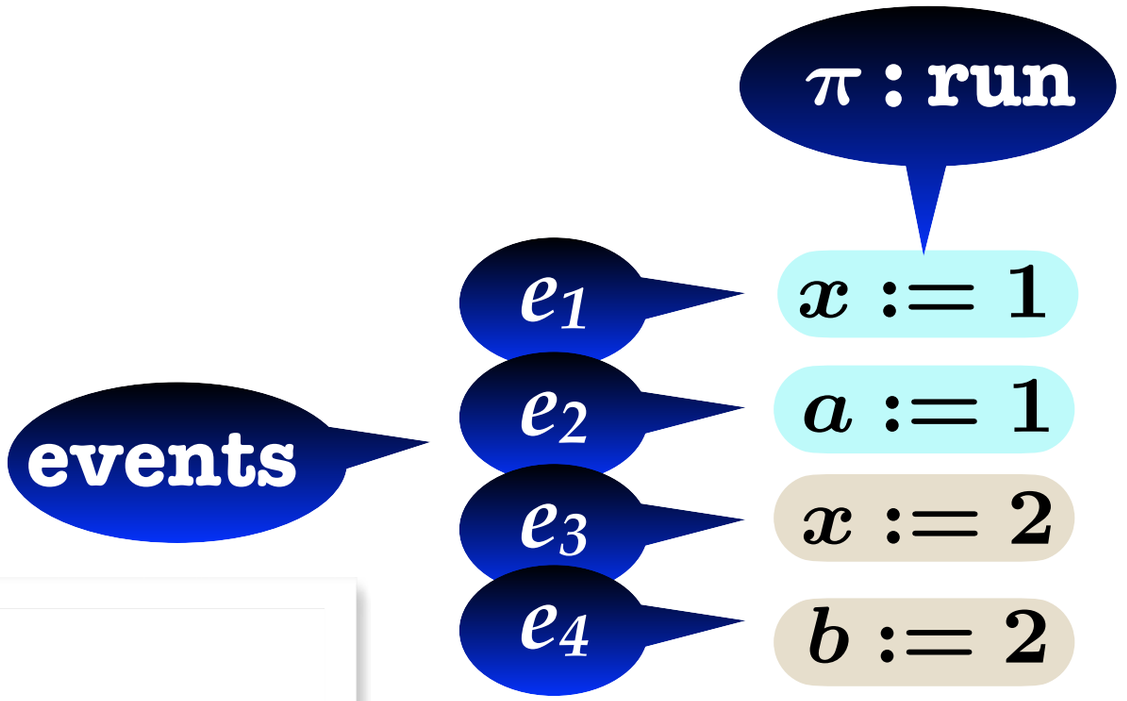
τ : trace



Concurrent Programs (under SC)

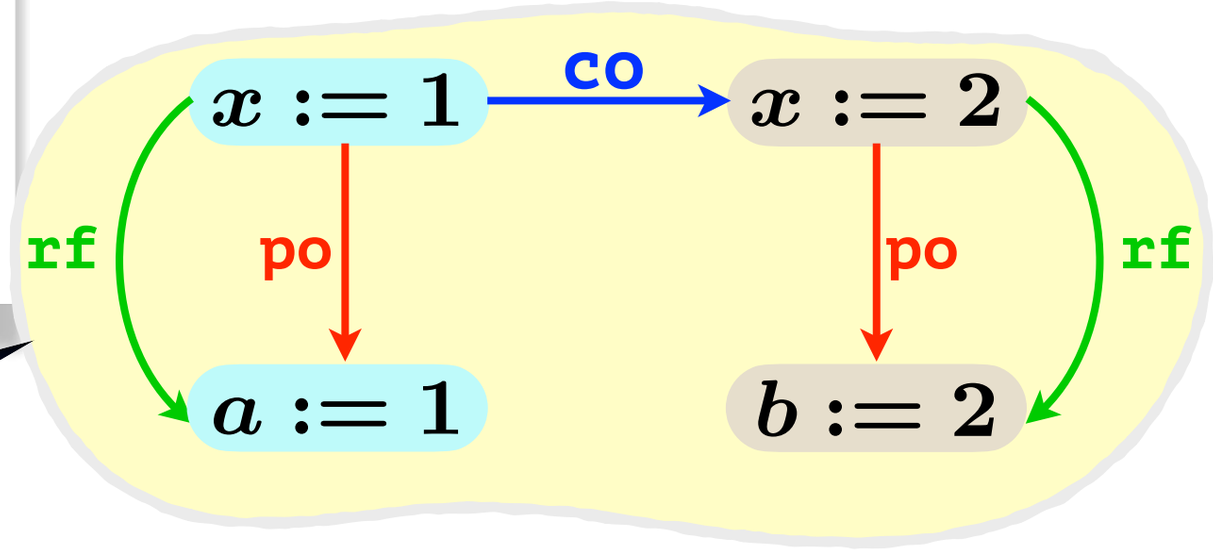
Initially $x = 0$

th_1	$\parallel$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

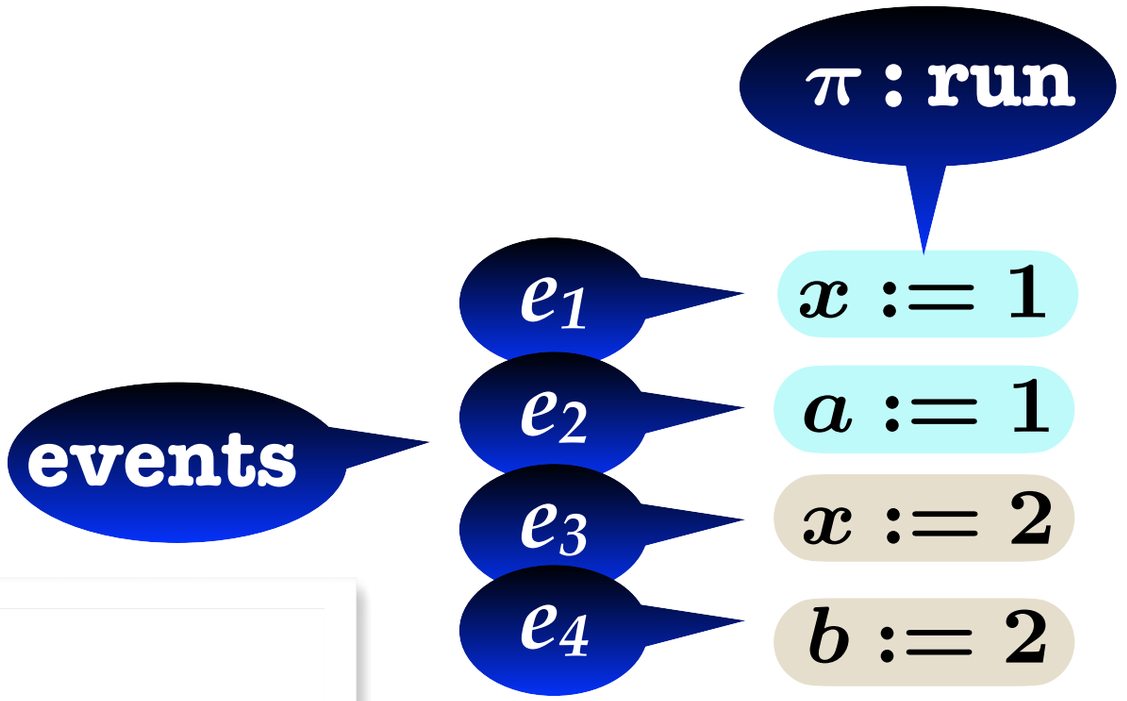
- po: program order
- rf: read-from
- co: coherence order



Concurrent Programs (under SC)

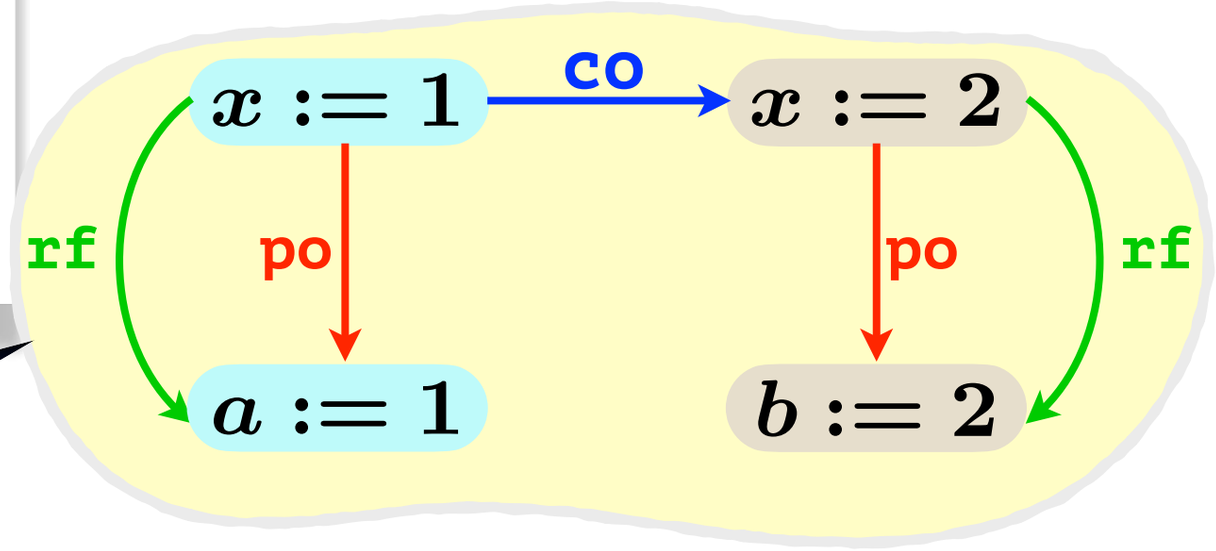
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order
- rf: read-from
- co: coherence order



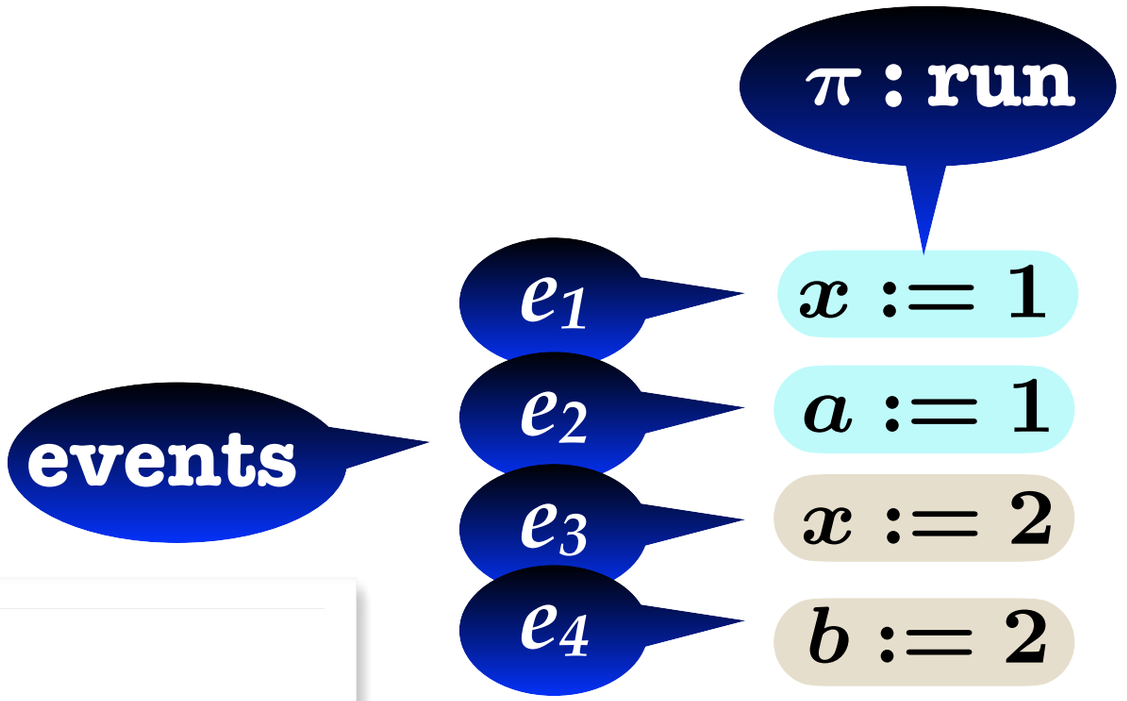
$\mathtt{co} = \cup_{x \in X} \mathtt{co}^x$

total

Concurrent Programs (under SC)

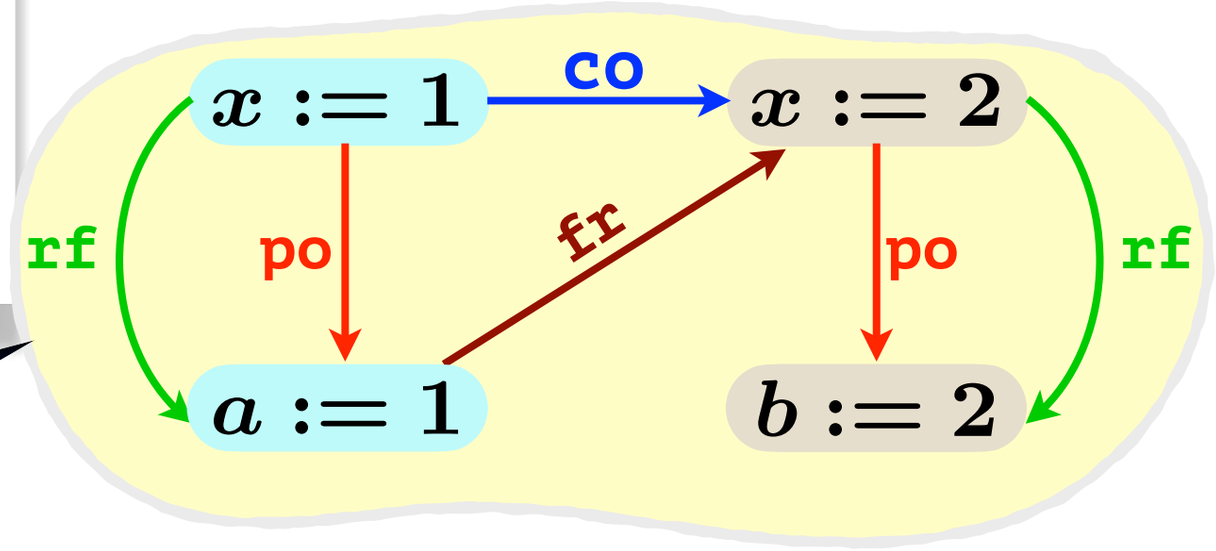
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



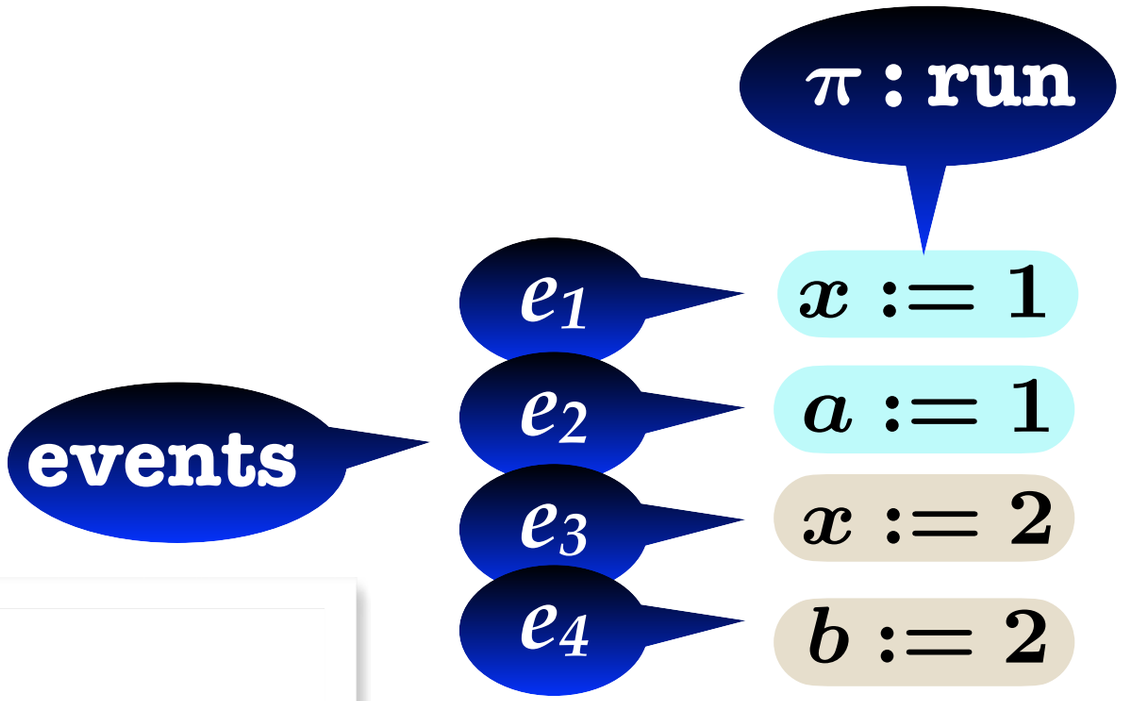
$\mathbf{co} = \cup_{x \in X} \mathbf{co}^x$

total

Concurrent Programs (under SC)

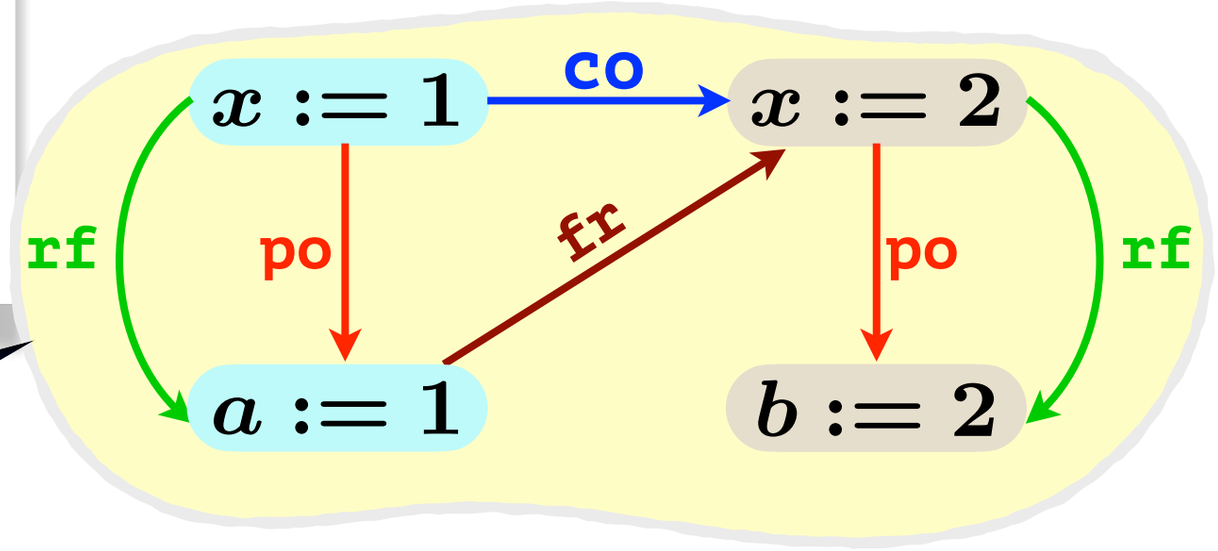
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



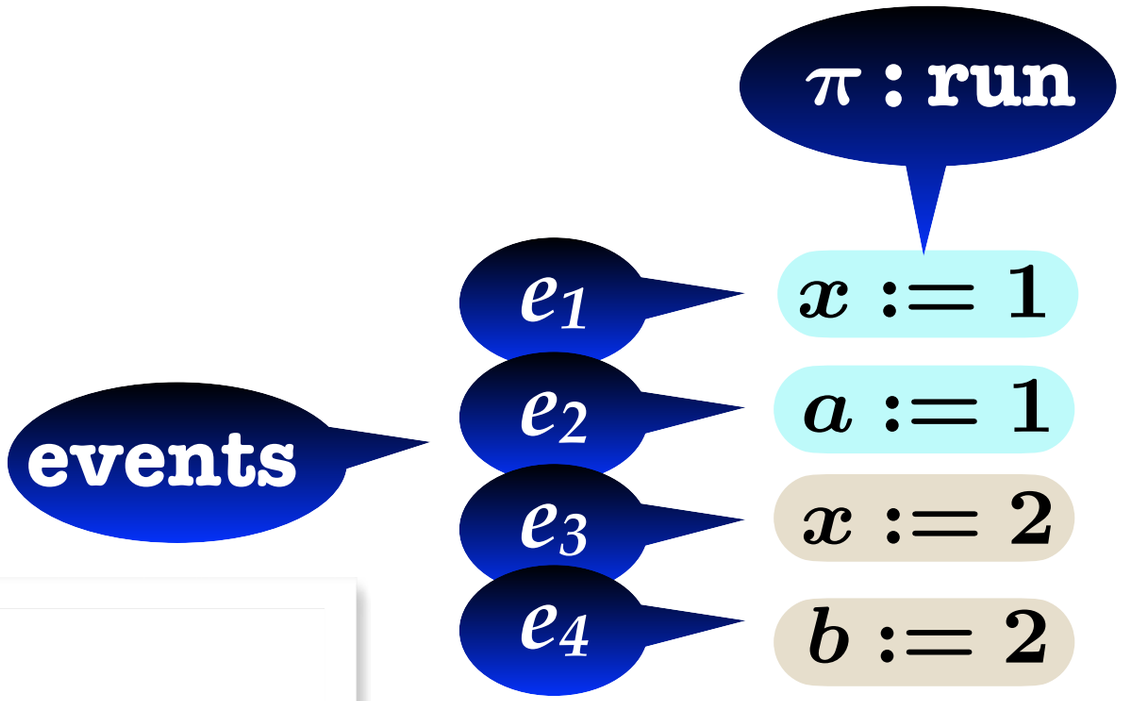
SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$

$\mathtt{co} = \cup_{x \in X} \mathtt{co}^x$ — total

Concurrent Programs (under SC)

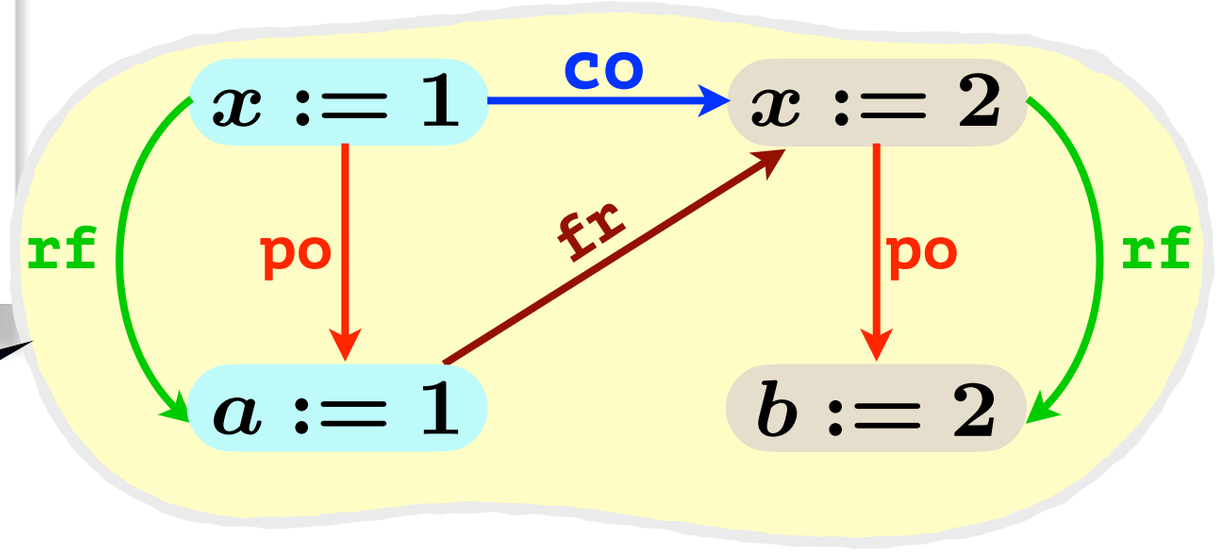
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



$\mathbf{co} = \cup_{x \in X} \mathbf{co}^x$ — **total**

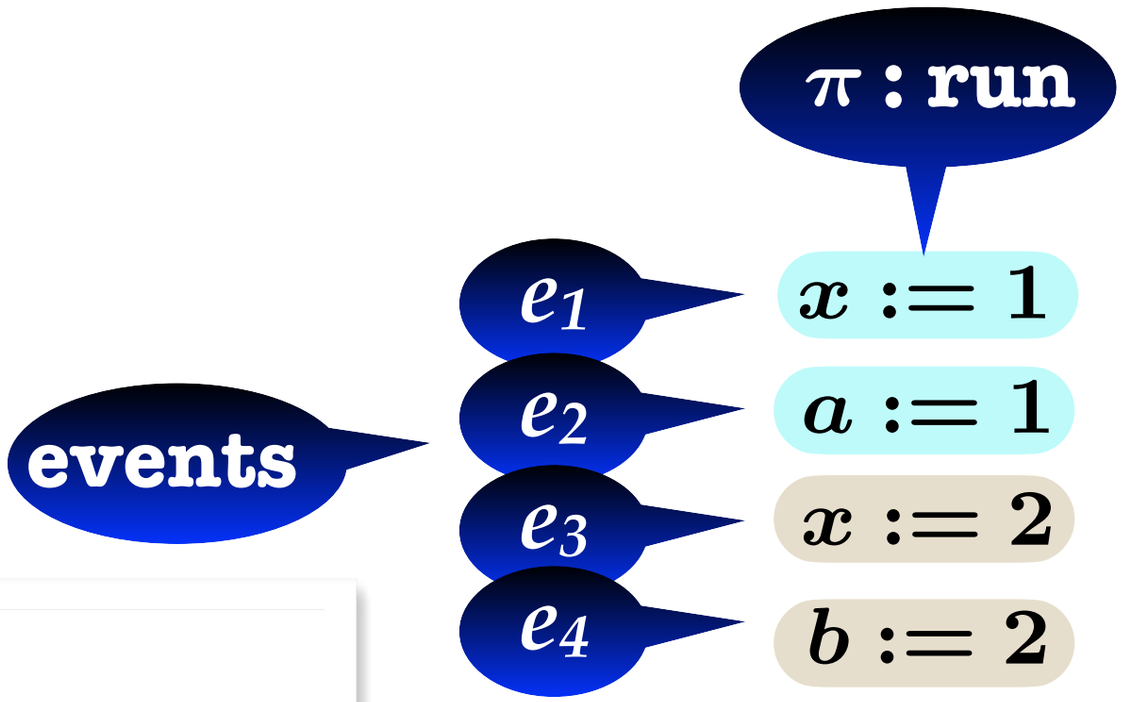
SC: $acyclic(\mathbf{po} \cup \mathbf{rf} \cup \mathbf{co} \cup \mathbf{fr})$

$\tau \models \mathrm{SC}$

Concurrent Programs (under SC)

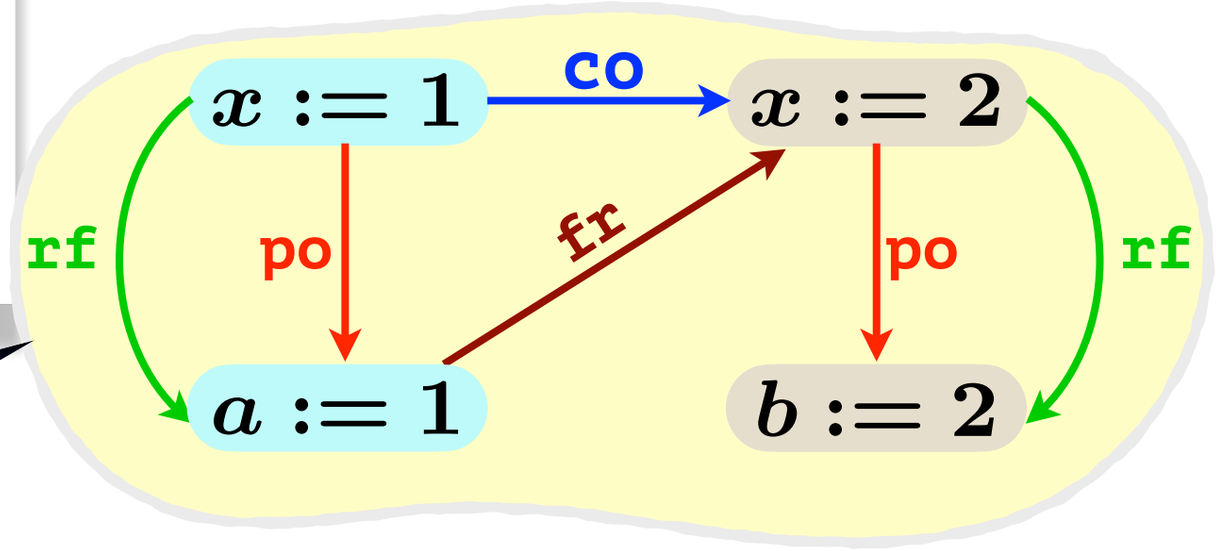
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**

- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



$\mathbf{co} = \cup_{x \in X} \mathbf{co}^x$ — **total**

SC: $acyclic(\mathbf{po} \cup \mathbf{rf} \cup \mathbf{co} \cup \mathbf{fr})$

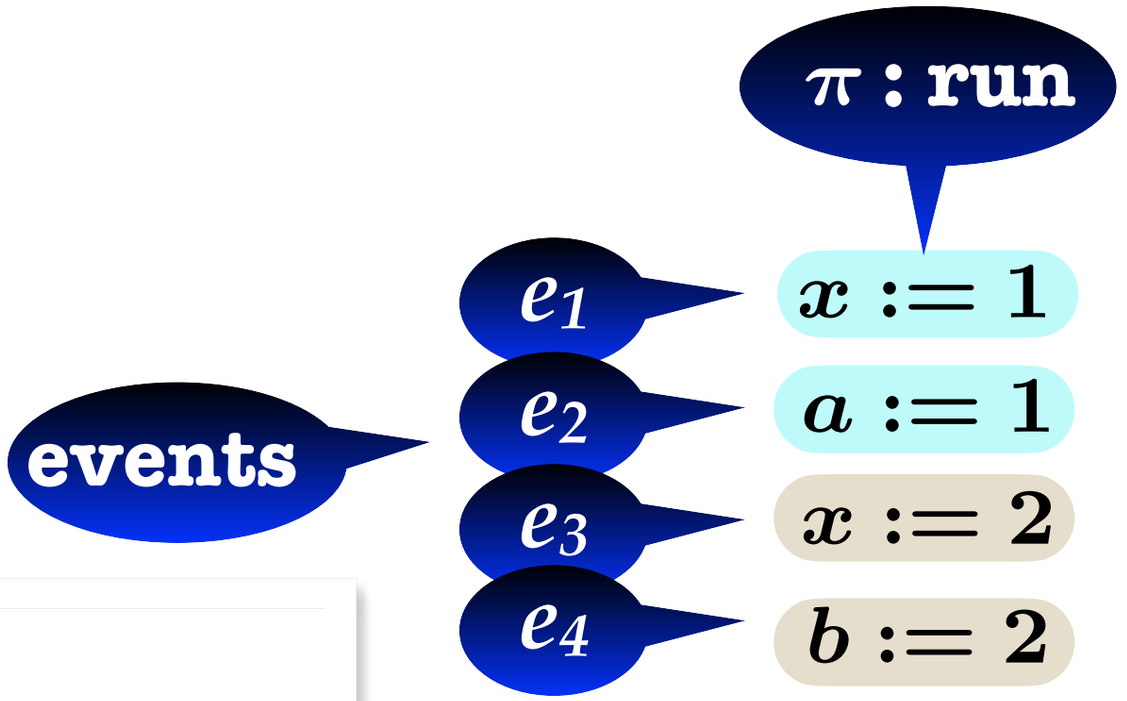
$\tau \models \mathbf{SC}$

$\pi \models \tau$

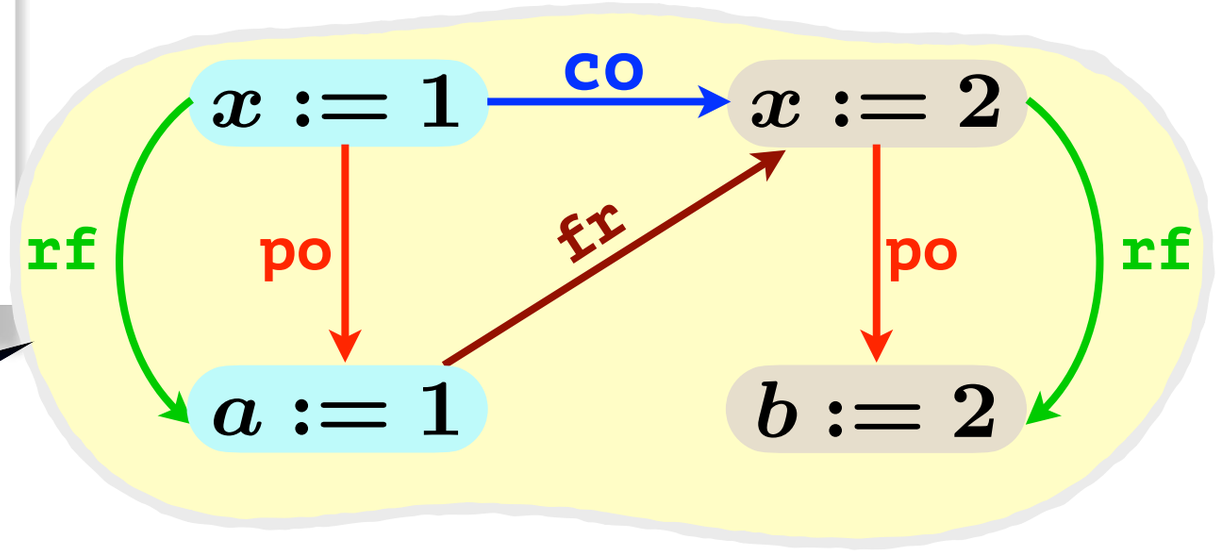
Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allow different memory models**



- po: program order
- rf: read-from
- co: coherence order
- fr: from-read

τ : trace

$\mathtt{co} = \cup_{x \in X} \mathtt{co}^x$

total

SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$

$\tau \models \mathrm{SC}$

$\pi \models \tau$

Concurrent Programs (under SC)

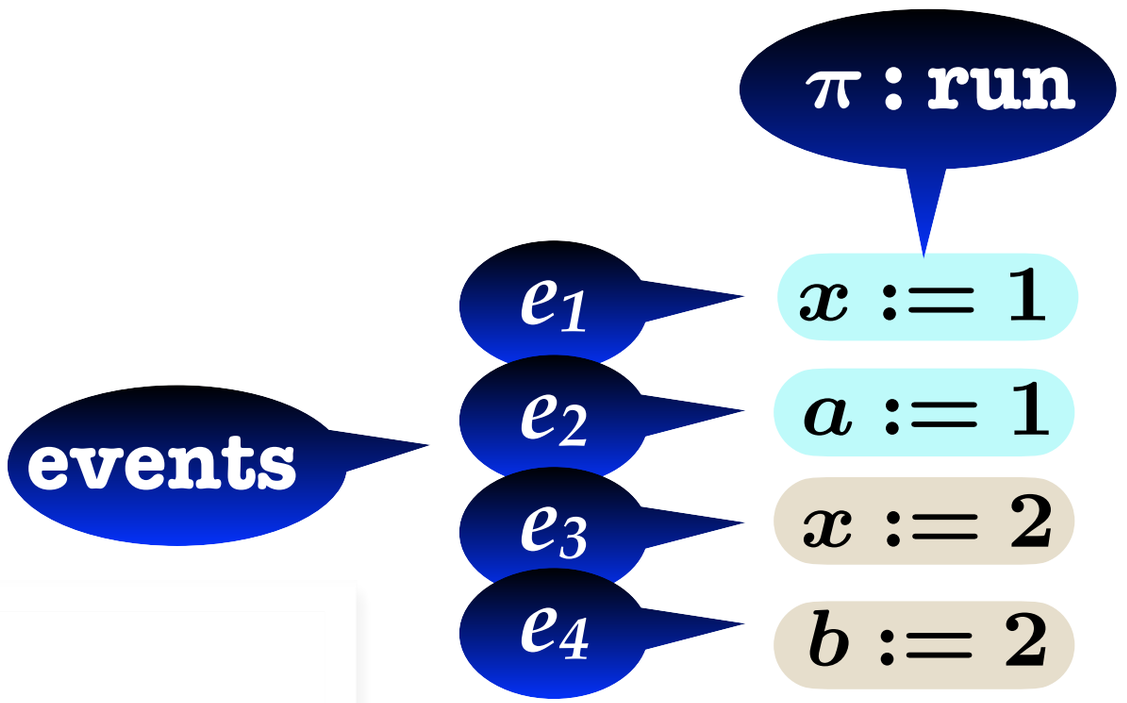
Initially $x = 0$

th_1	$\parallel$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$

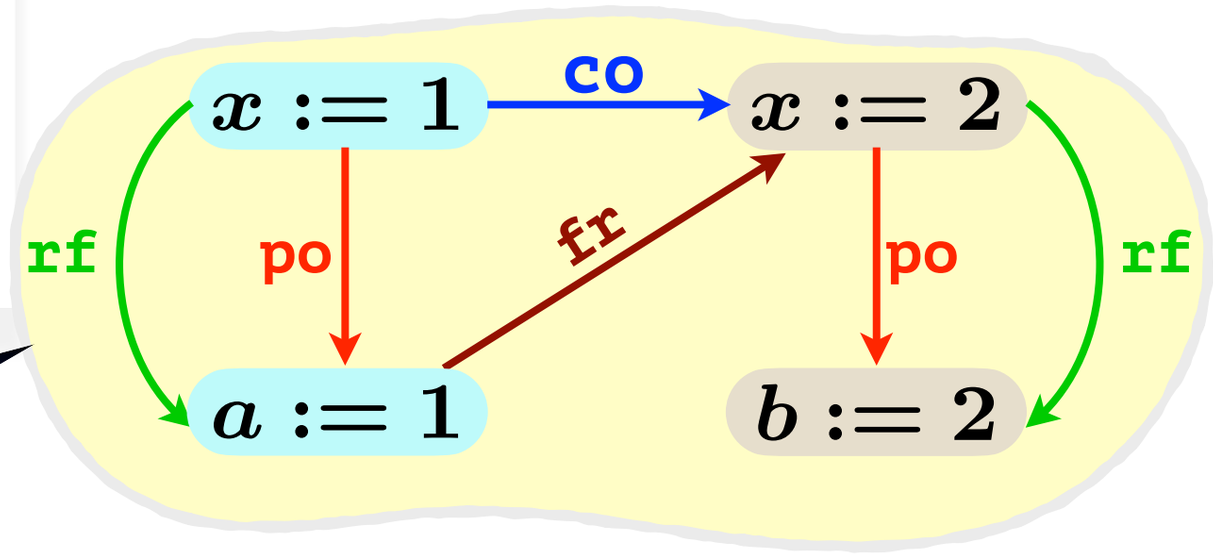
- $(a = 1) \wedge (b = 2)$?

assertion

- efficient:
  - more compact than runs
  - convenient for checking assertions
- abstract:
  - allows different memory models



- **po**: program order
- **rf**: read-from
- **co**: coherence order
- **fr**: from-read



τ : trace

$\mathbf{co} = \cup_{x \in X} \mathbf{co}^x$

total

SC: $acyclic(\mathbf{po} \cup \mathbf{rf} \cup \mathbf{co} \cup \mathbf{fr})$

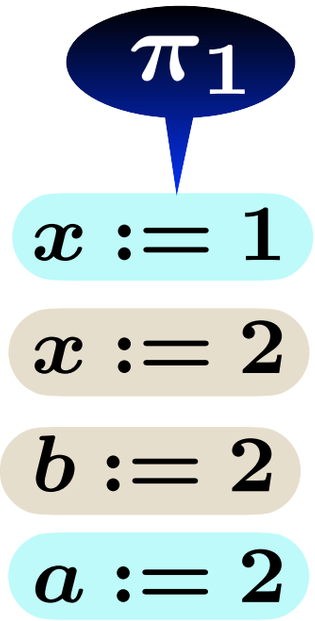
$\tau \models \mathrm{SC}$

$\pi \models \tau$

Concurrent Programs (under SC)

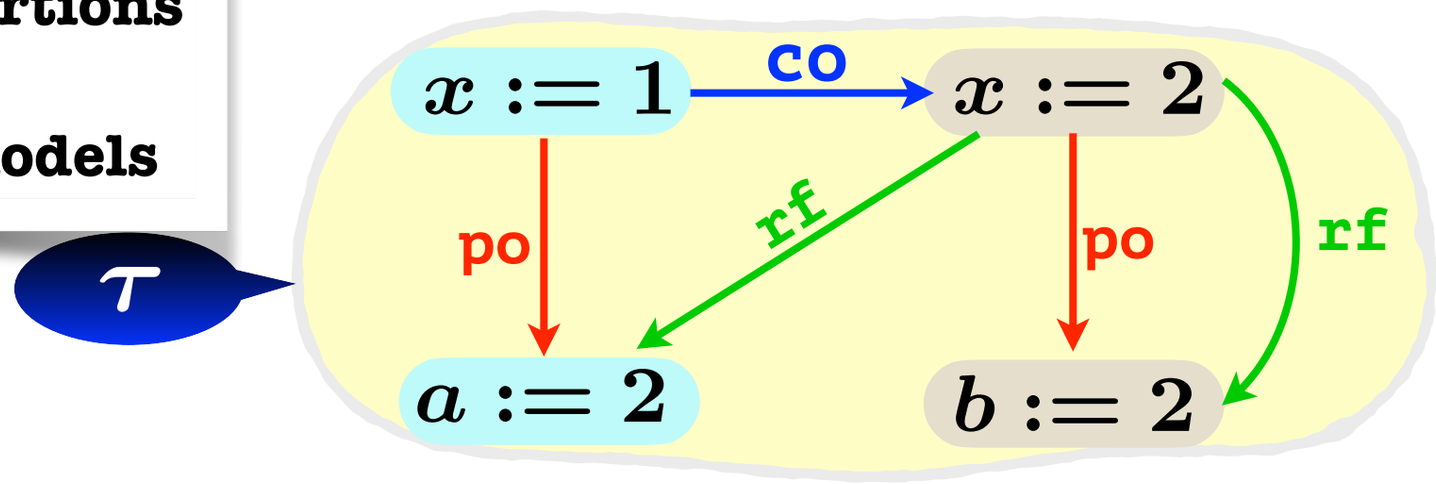
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allows different memory models**

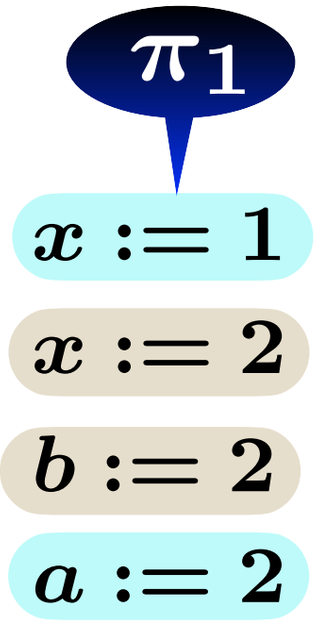
- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



Concurrent Programs (under SC)

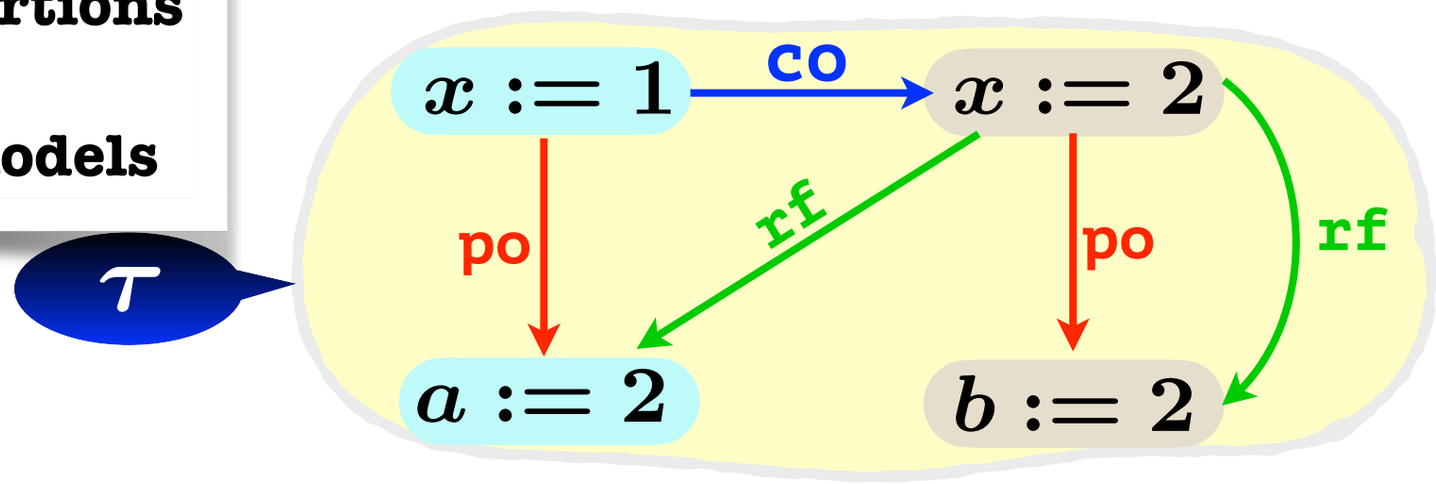
Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allows different memory models**

- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



Concurrent Programs (under SC)

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$

π_1

$x := 1$

$x := 2$

$b := 2$

$a := 2$

π_2

$x := 1$

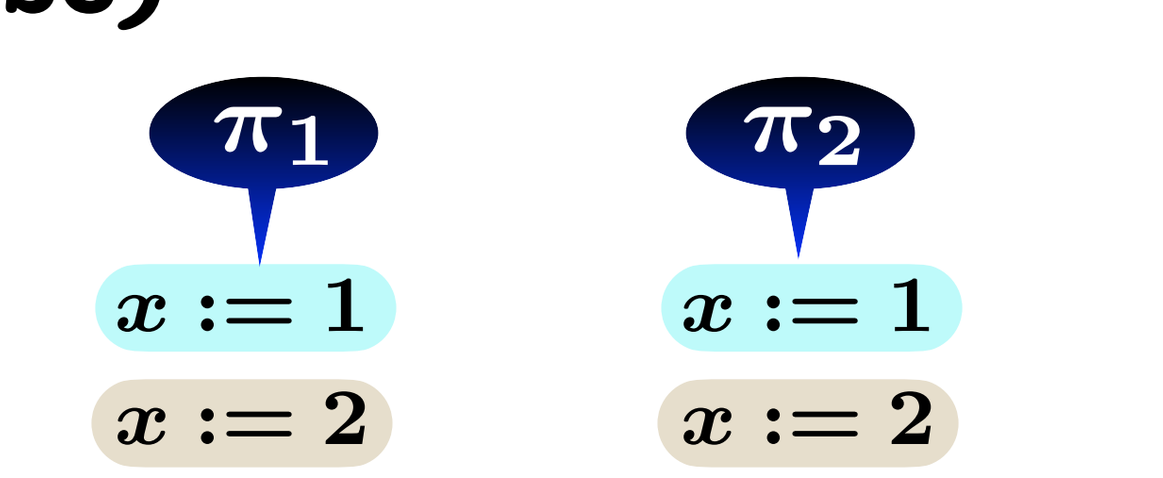
$x := 2$

$a := 2$

$b := 2$

- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allows different memory models**

τ

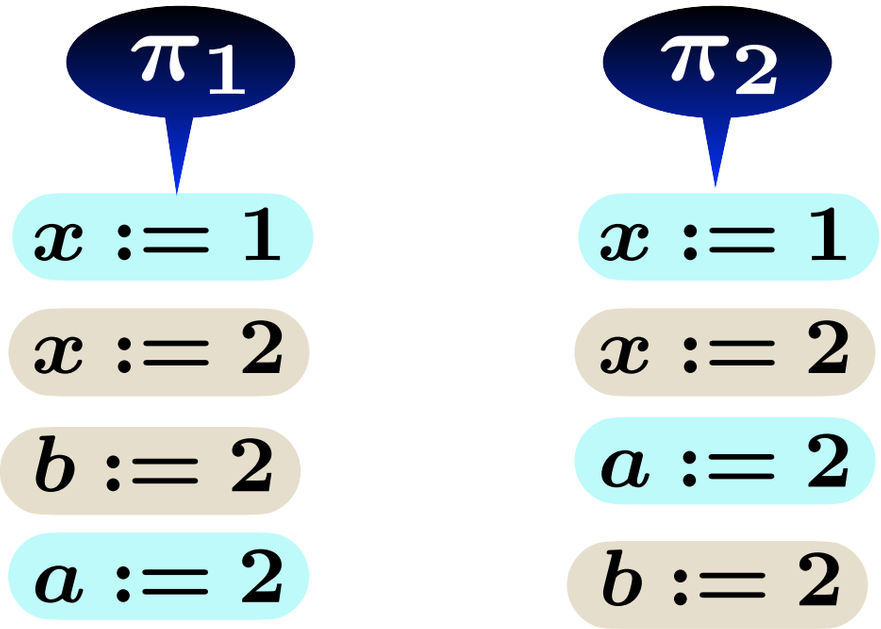


- po: program order
- rf: read-from
- co: coherence order
- fr: from-read

Concurrent Programs (under SC)

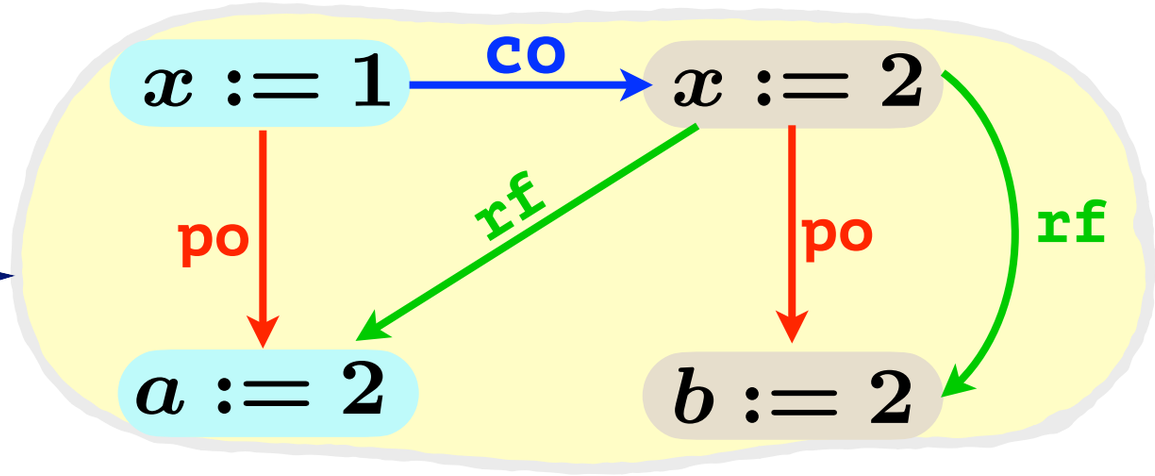
Initially $x = 0$

th_1	$\parallel$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



- **Traces**
  - **efficient:**
    - **more compact than runs**
    - **sufficient for checking assertions**
  - **abstract:**
    - **allows different memory models**

- po: program order
- rf: read-from
- co: coherence order
- fr: from-read



$\pi_1 \models \tau$

$\pi_2 \models \tau$

DPOR: Dynamic Partial Order Reduction

Initially $x = 0$

th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$

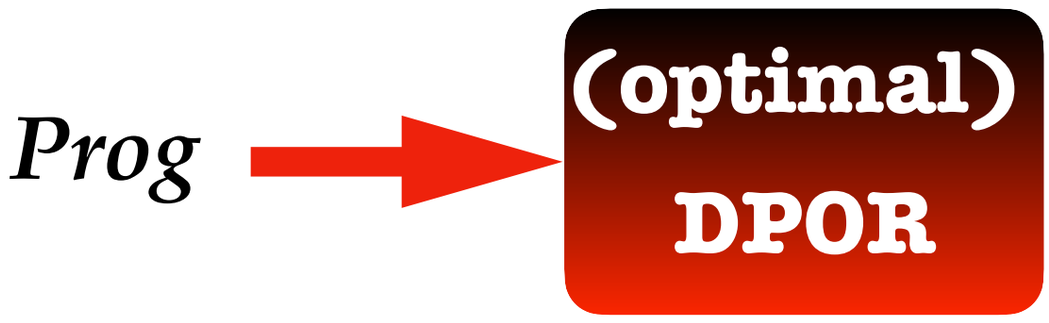
(optimal) DPOR

(POPL'14)

DPOR: Dynamic Partial Order Reduction

Initially $x = 0$

th_1	th_2
$x := 1$	$x := 2$
$a := x$	$b := x$

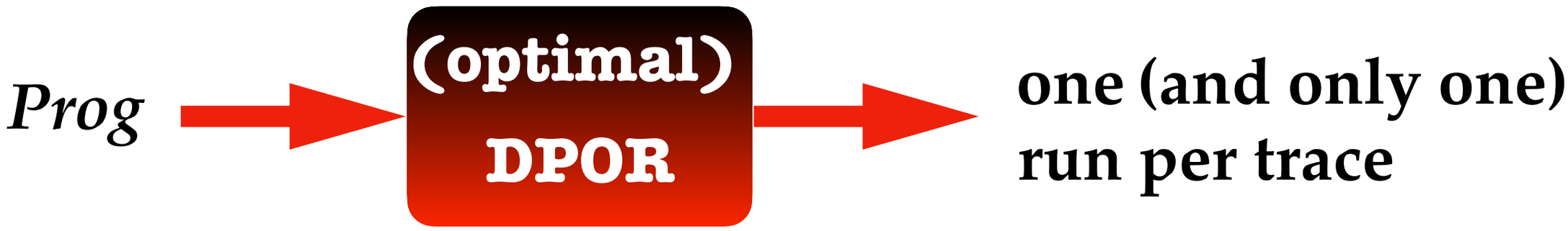


(POPL'14)

DPOR: Dynamic Partial Order Reduction

Initially $x = 0$

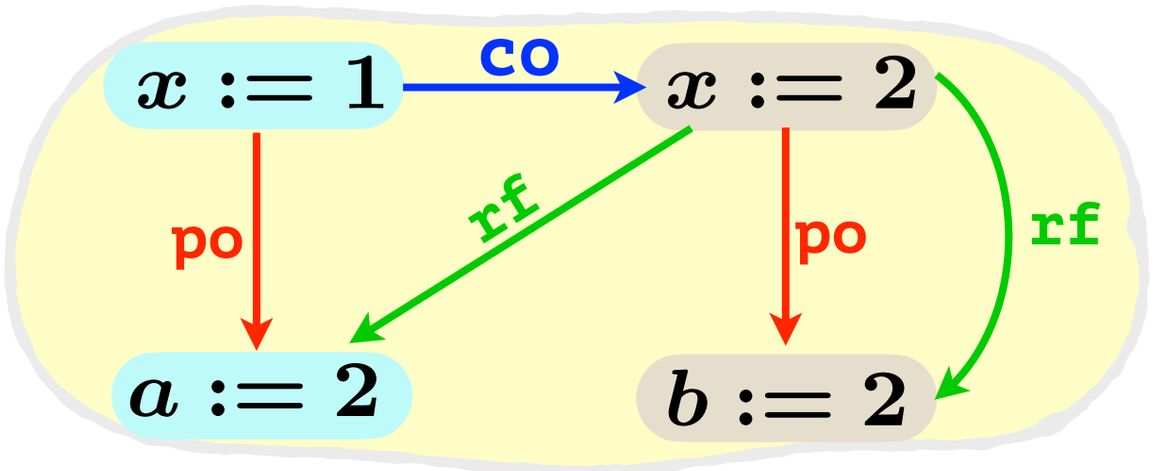
th_1	$\|\|$	th_2
$x := 1$		$x := 2$
$a := x$		$b := x$



(POPL'14)

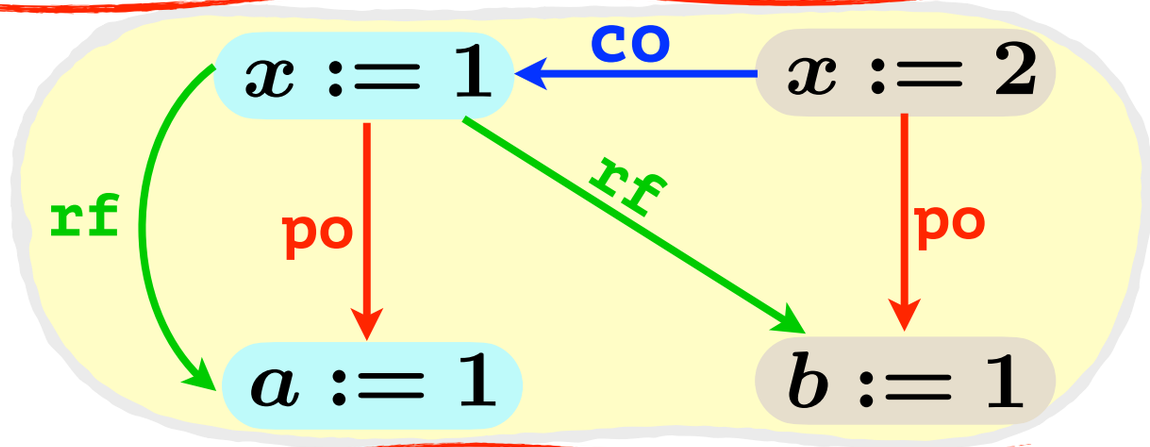
$x := 1$
$x := 2$
$a := 2$
$b := 2$

$x := 1$
$x := 2$
$b := 2$
$a := 2$

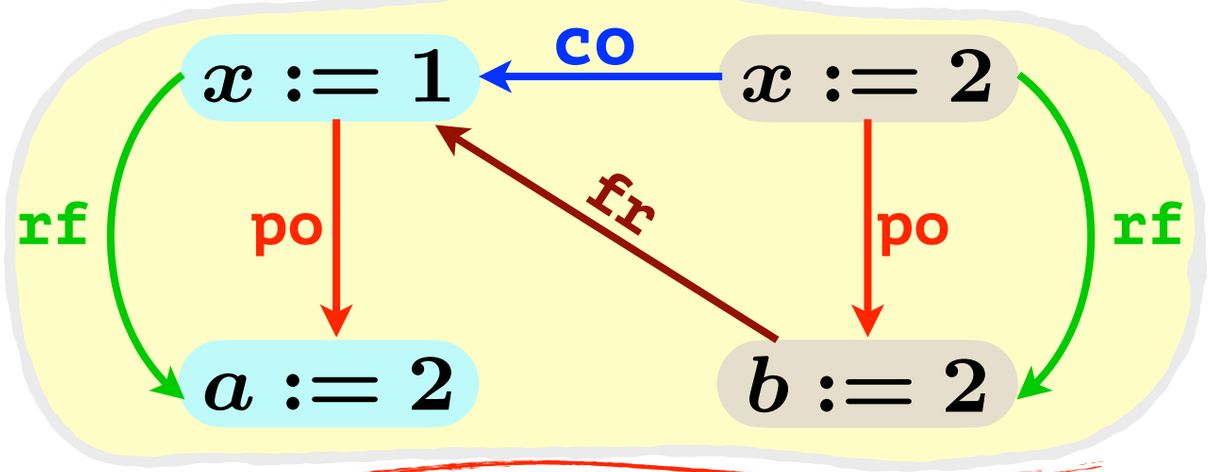


$x := 2$
$x := 1$
$b := 1$
$a := 1$

$x := 2$
$x := 1$
$a := 1$
$b := 1$

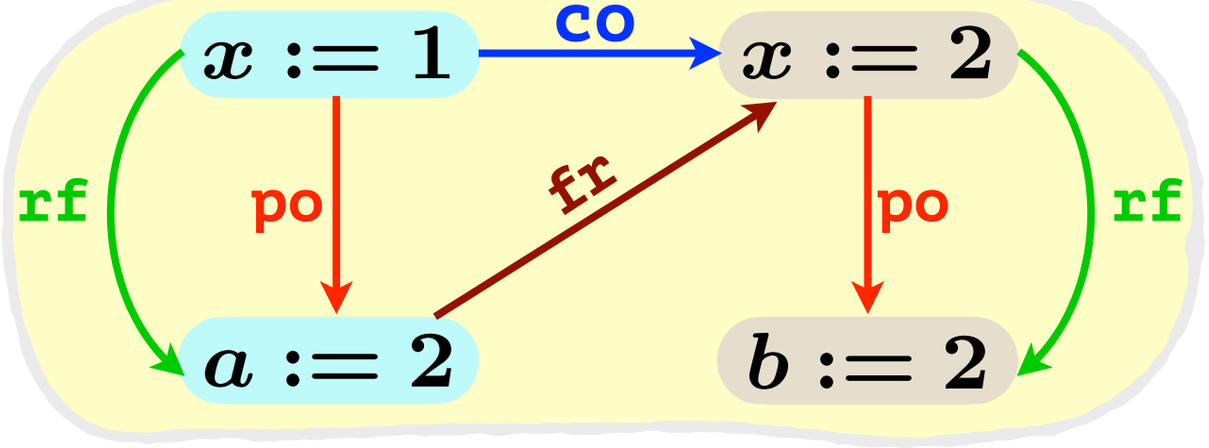


$x := 1$
$a := 1$
$x := 2$
$b := 2$

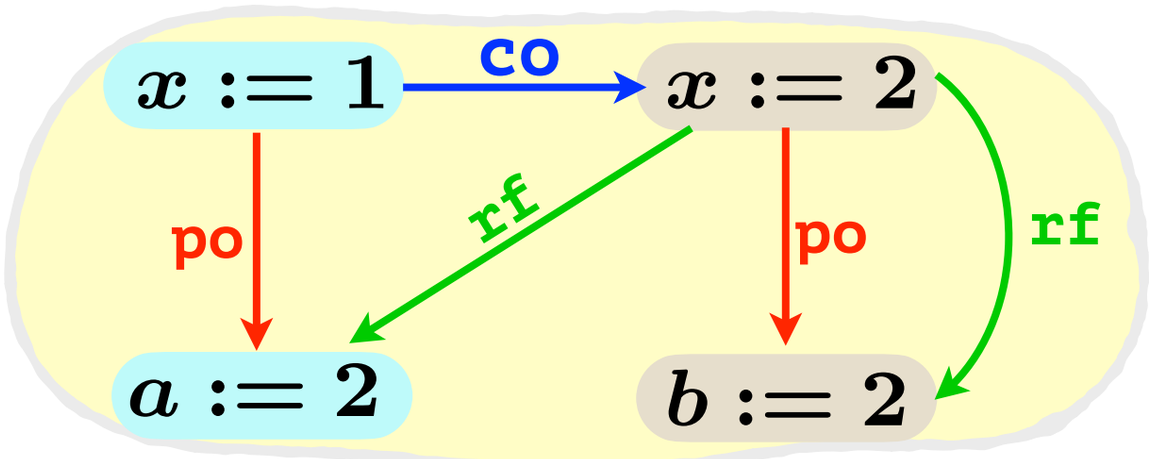


(optimal) DPOR

$x := 2$
$b := 2$
$x := 1$
$a := 1$

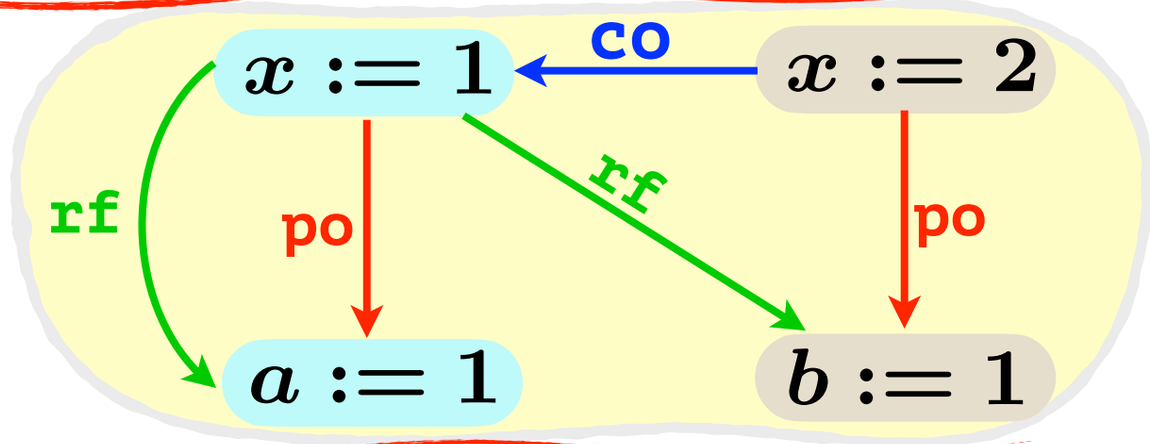


$x := 1$
$x := 2$
$a := 2$
$b := 2$

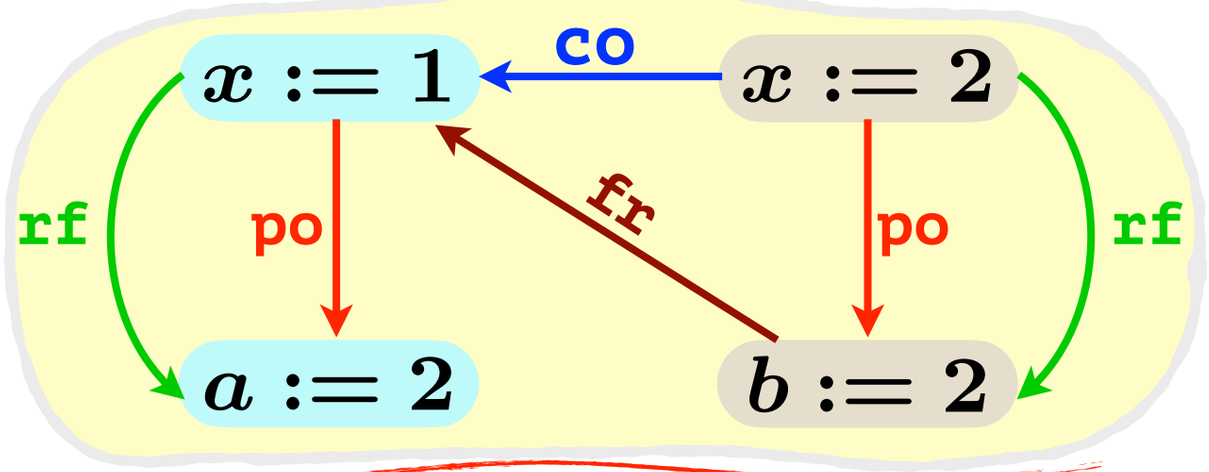


$x := 2$
$x := 1$
$b := 1$
$a := 1$

$x := 2$
$x := 1$
$a := 1$
$b := 1$

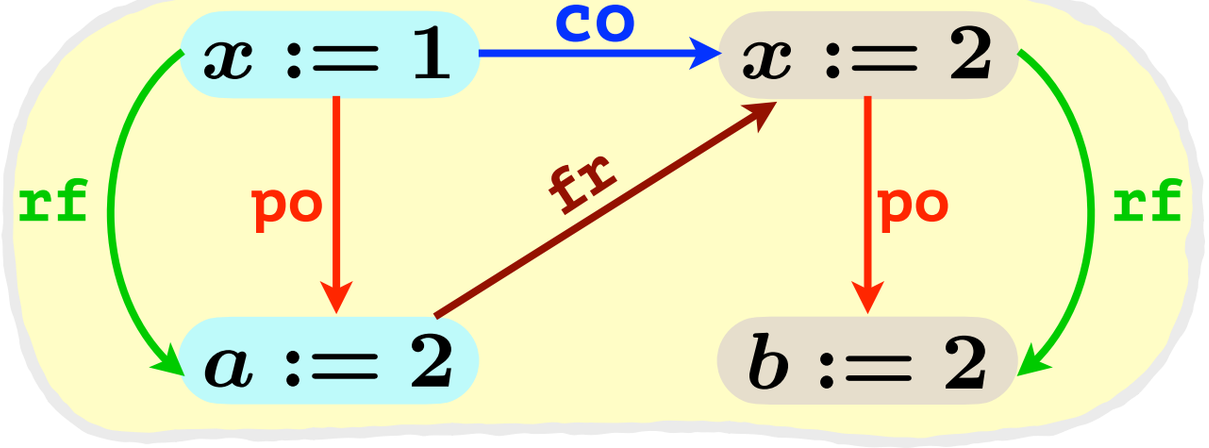


$x := 1$
$a := 1$
$x := 2$
$b := 2$

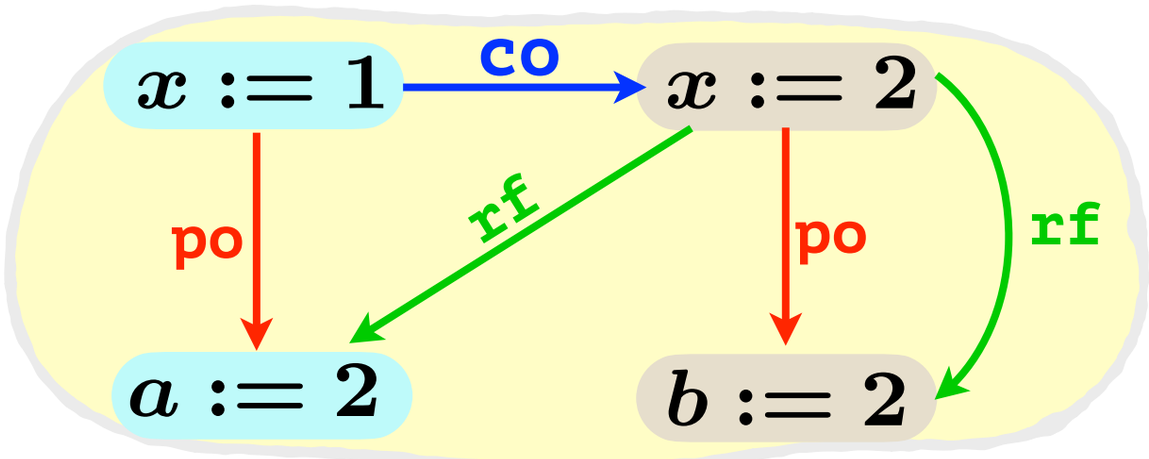


(optimal) DPOR

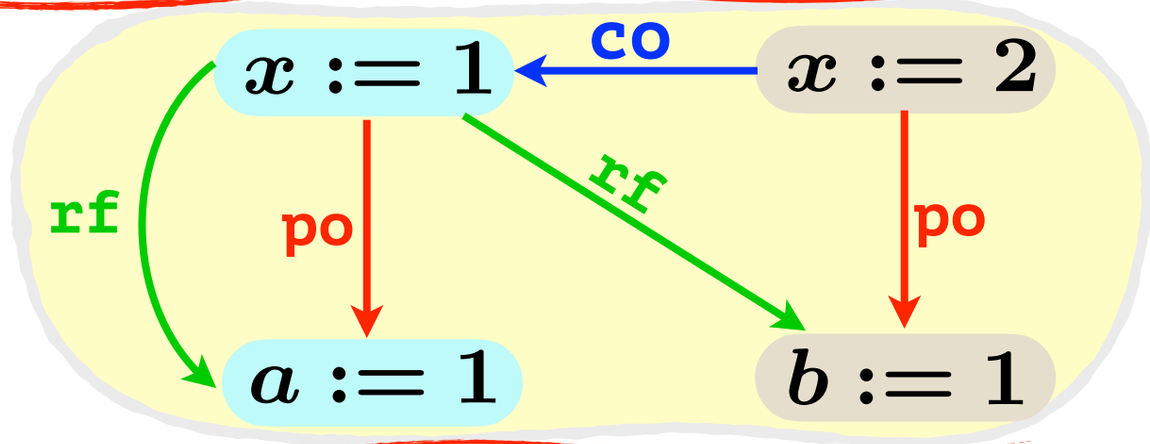
$x := 2$
$b := 2$
$x := 1$
$a := 1$



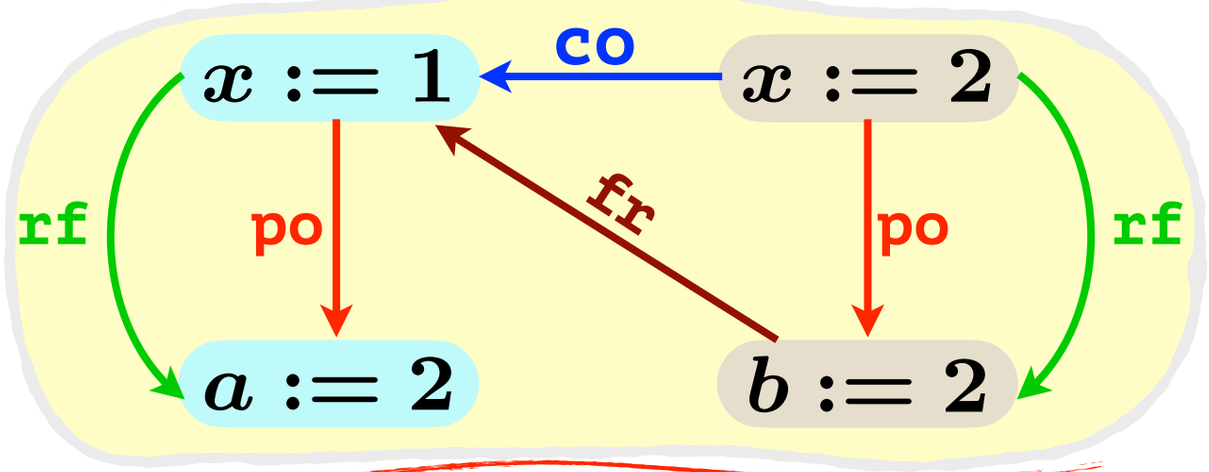
$x := 1$
$x := 2$
$a := 2$
$b := 2$



$x := 2$
$x := 1$
$b := 1$
$a := 1$

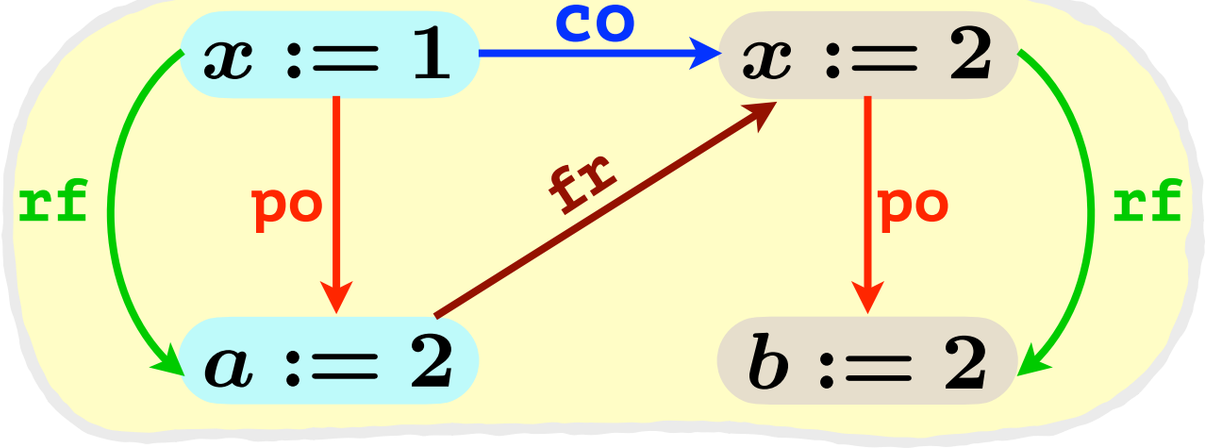


$x := 1$
$a := 1$
$x := 2$
$b := 2$



(optimal) DPOR

$x := 2$
$b := 2$
$x := 1$
$a := 1$

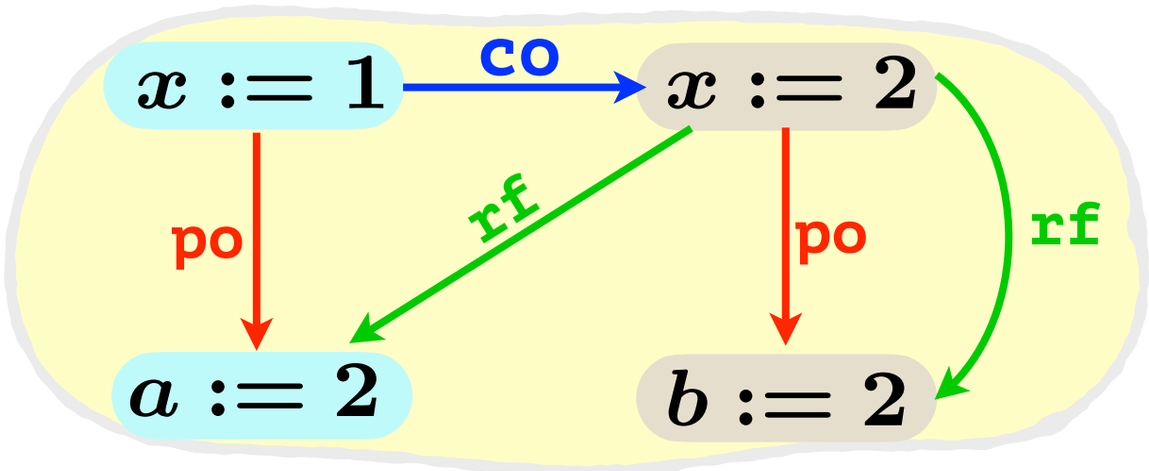


$x := 1$

$x := 2$

$a := 2$

$b := 2$

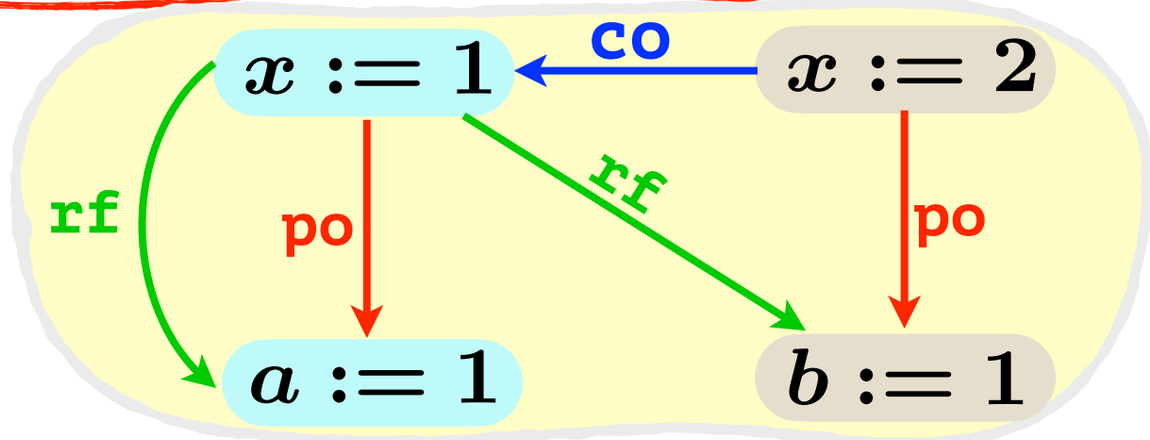


$x := 2$

$x := 1$

$b := 1$

$a := 1$

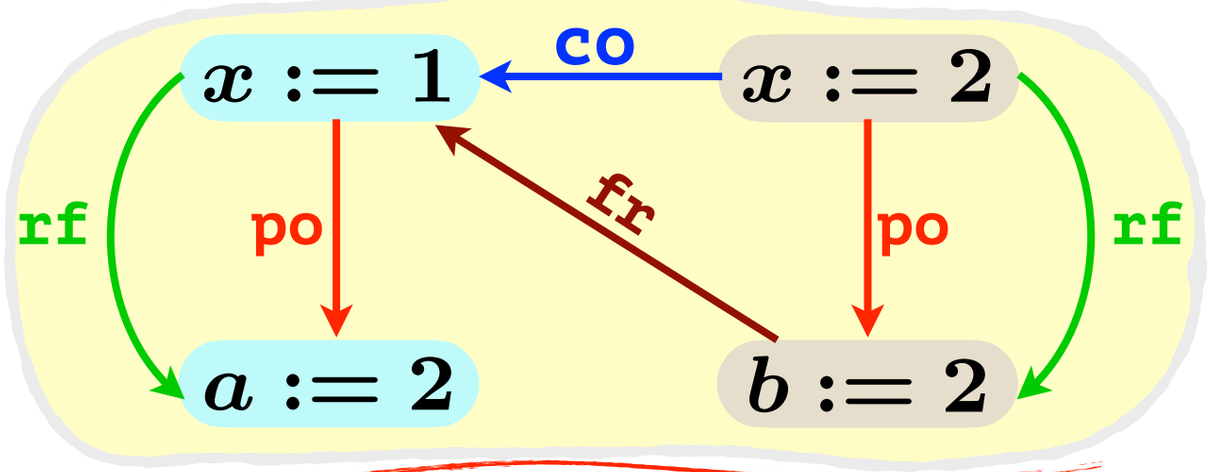


$x := 1$

$a := 1$

$x := 2$

$b := 2$



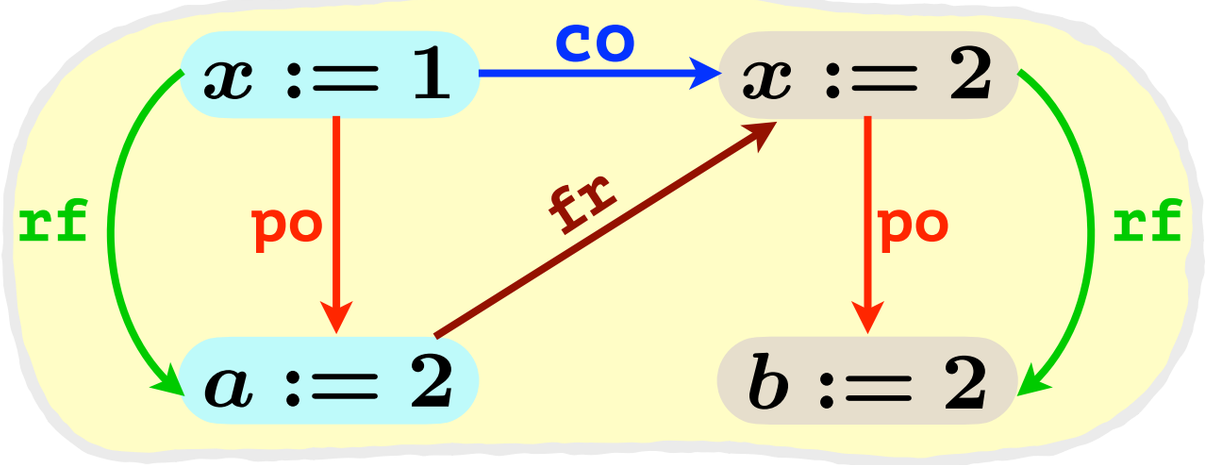
(super-optimal) DPOR

$x := 2$

$b := 2$

$x := 1$

$a := 1$

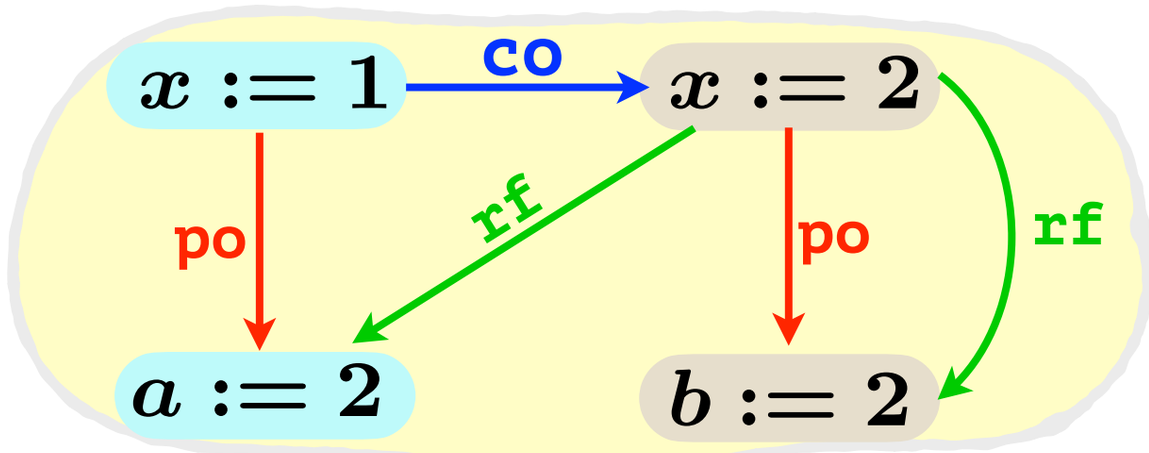


$x := 1$

$x := 2$

$a := 2$

$b := 2$

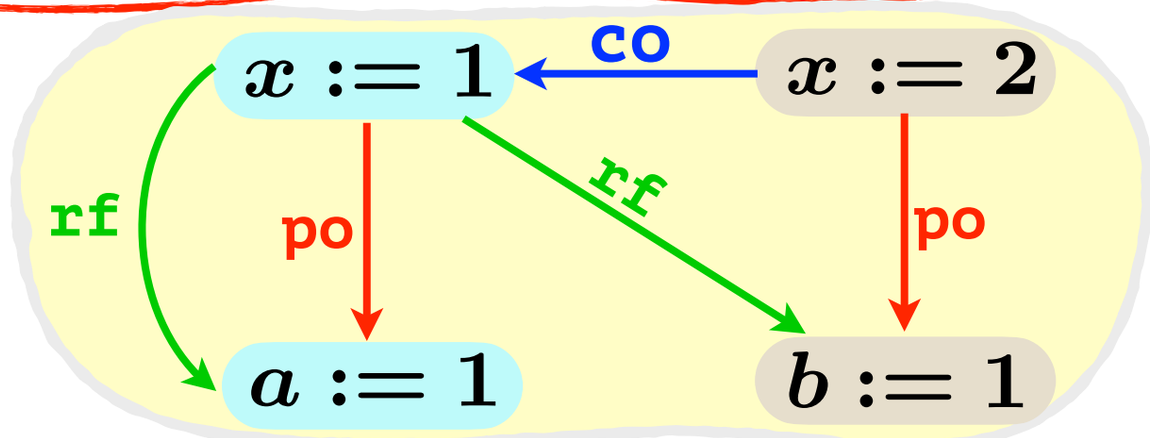


$x := 2$

$x := 1$

$b := 1$

$a := 1$

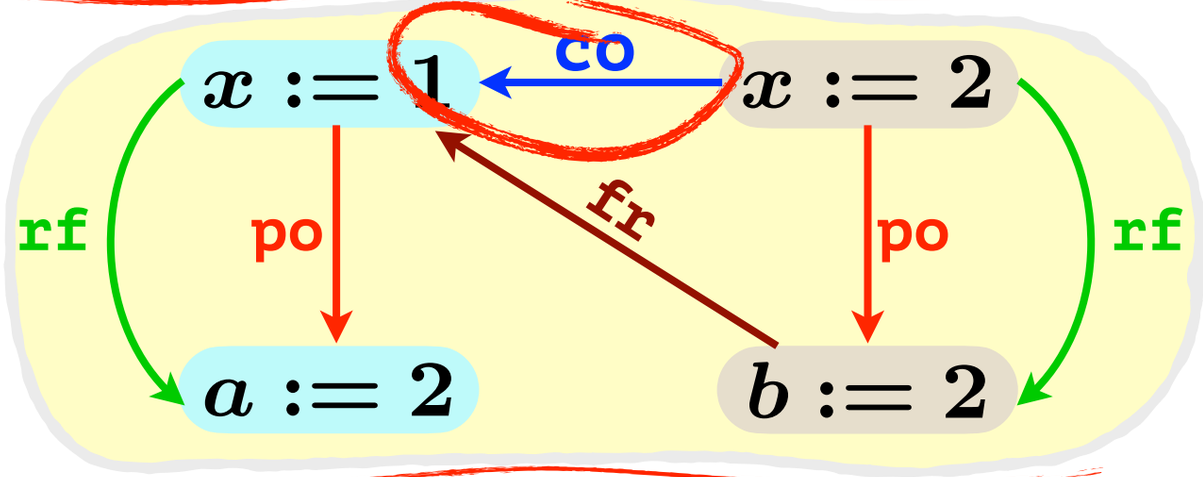


$x := 1$

$a := 1$

$x := 2$

$b := 2$



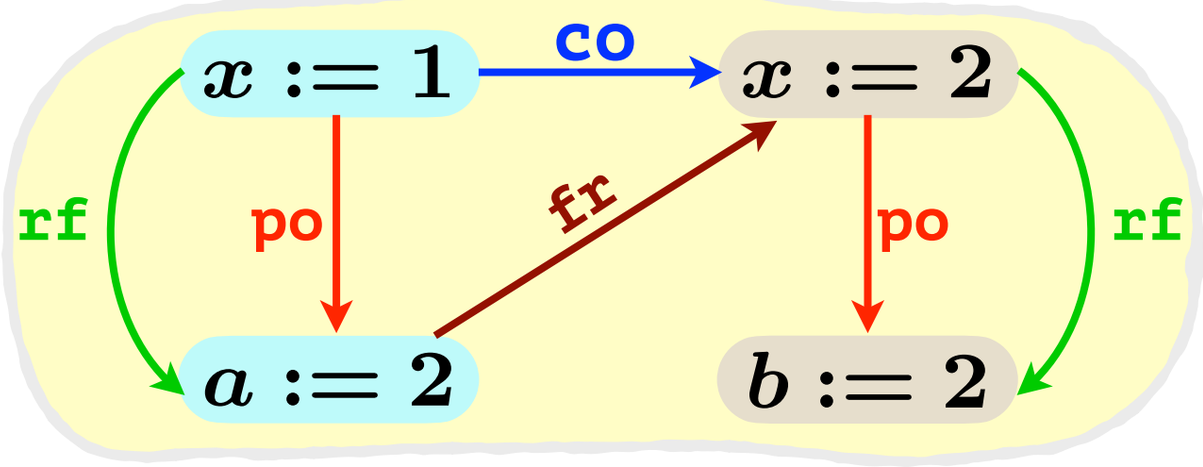
(super-optimal) DPOR

$x := 2$

$b := 2$

$x := 1$

$a := 1$

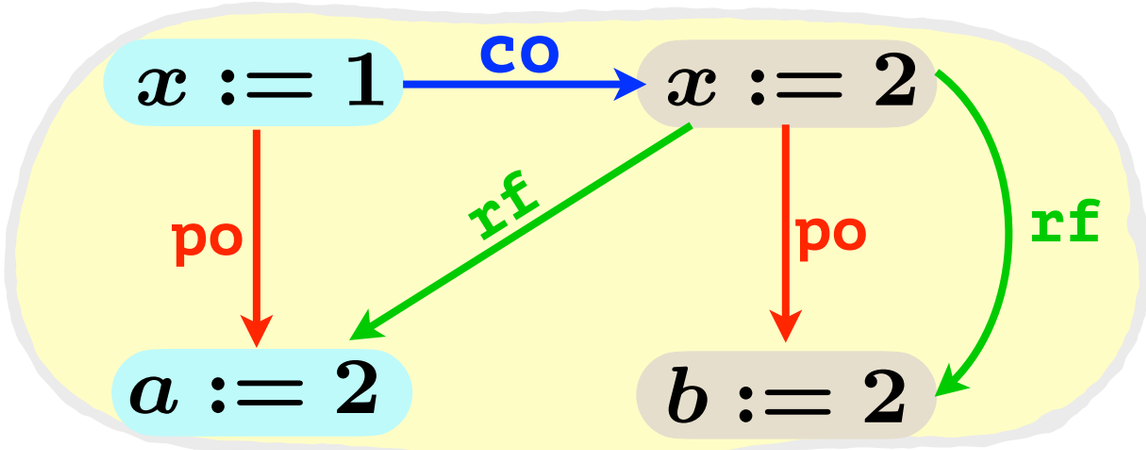


$x := 1$

$x := 2$

$a := 2$

$b := 2$

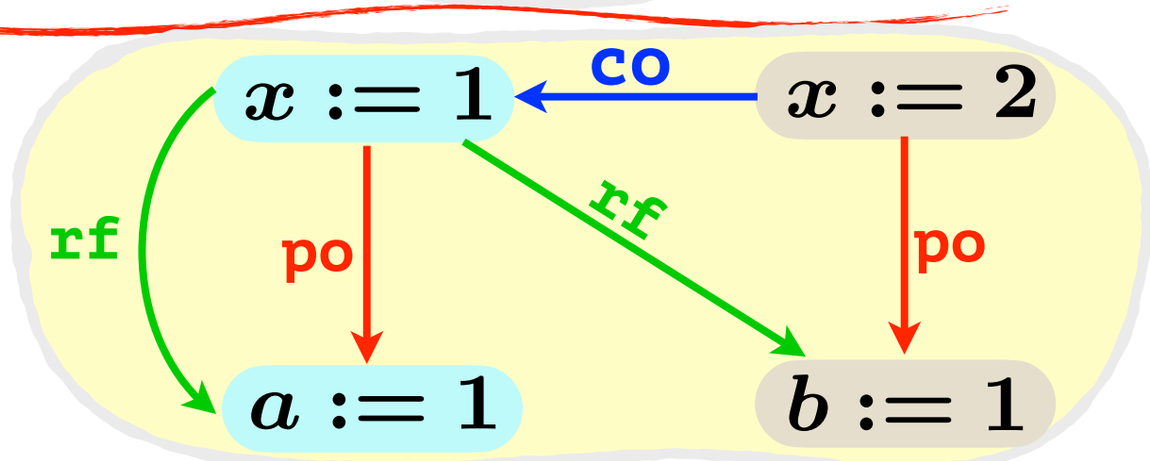


$x := 2$

$x := 1$

$b := 1$

$a := 1$

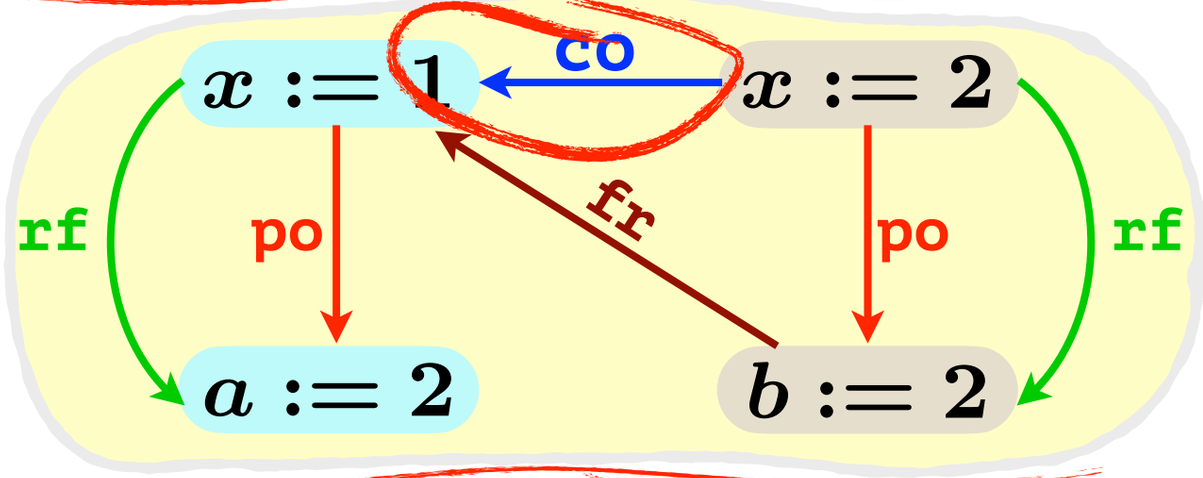


$x := 1$

$a := 1$

$x := 2$

$b := 2$



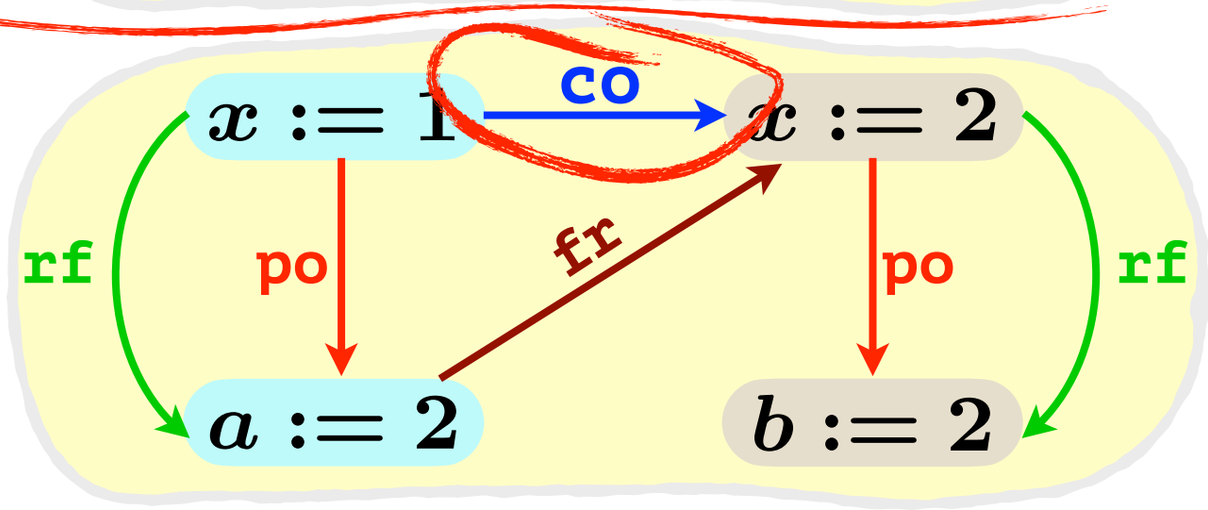
(super-optimal) DPOR

$x := 2$

$b := 2$

$x := 1$

$a := 1$

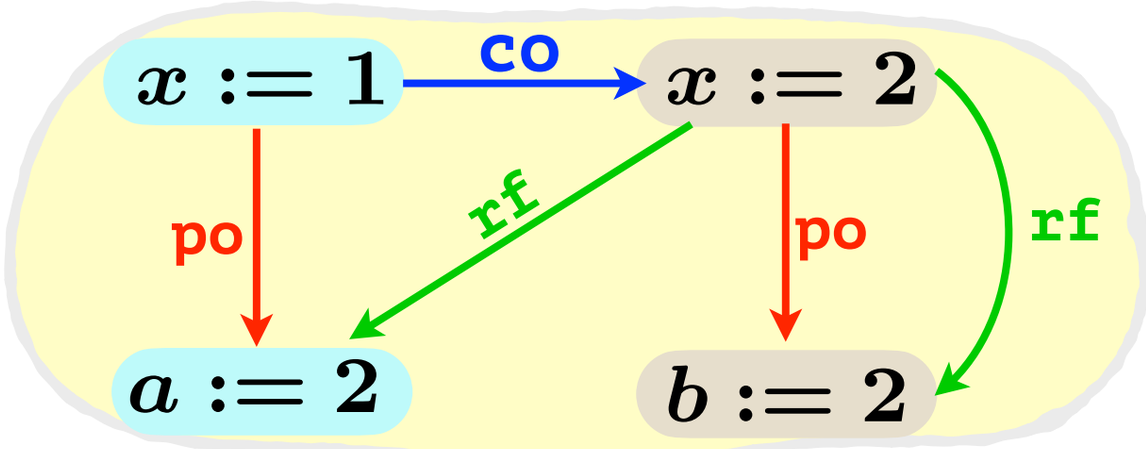


$x := 1$

$x := 2$

$a := 2$

$b := 2$

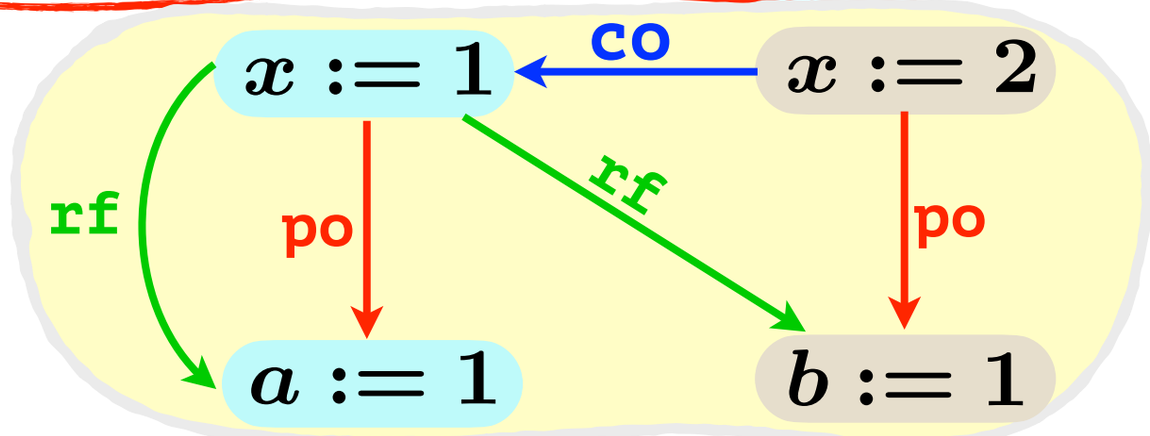


$x := 2$

$x := 1$

$b := 1$

$a := 1$

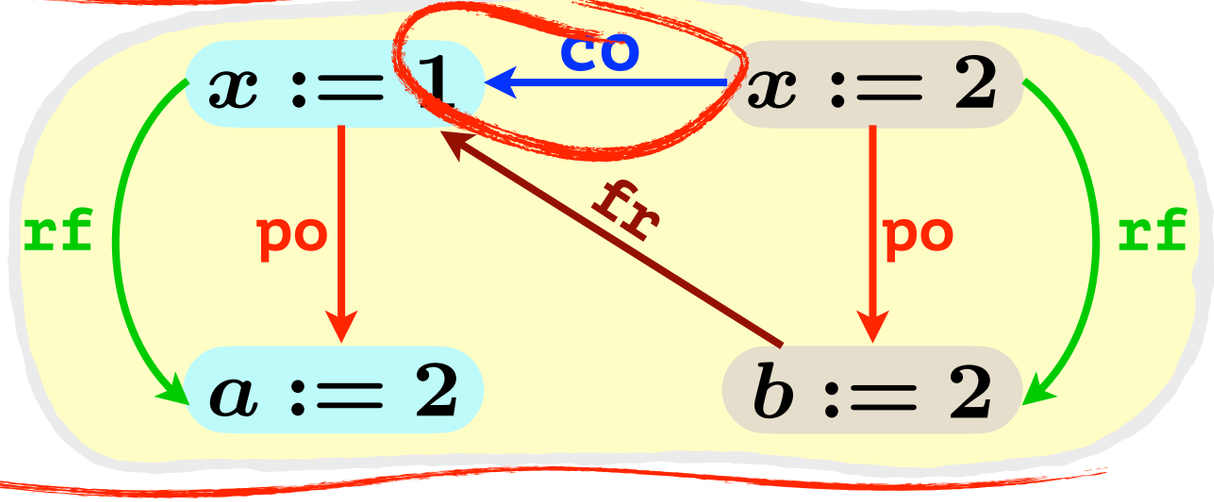


$x := 1$

$a := 1$

$x := 2$

$b := 2$



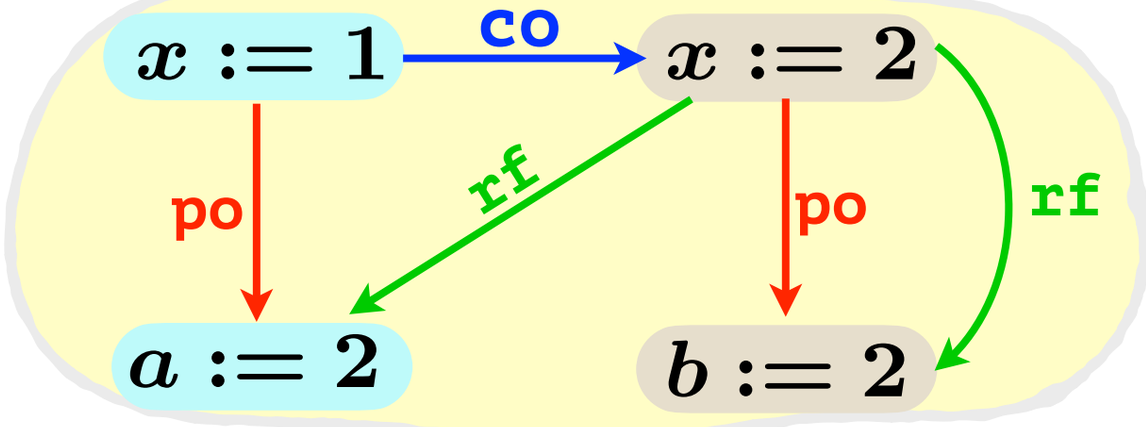
(super-optimal) DPOR

$x := 1$

$x := 2$

$a := 2$

$b := 2$

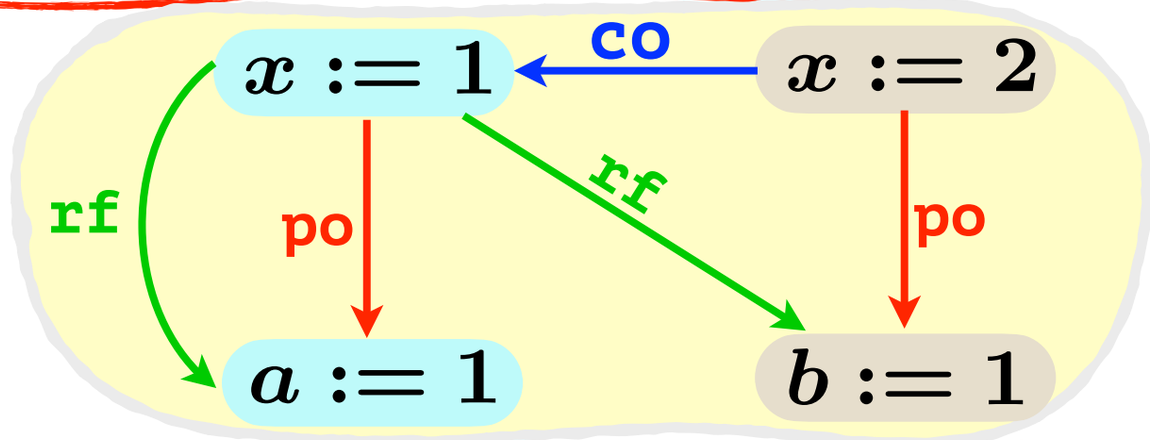


$x := 2$

$x := 1$

$b := 1$

$a := 1$

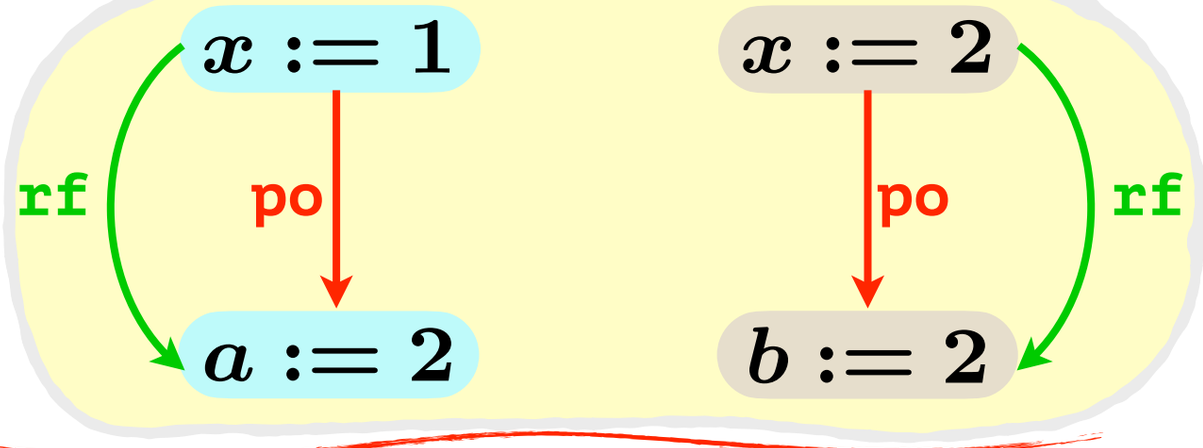


$x := 1$

$a := 1$

$x := 2$

$b := 2$



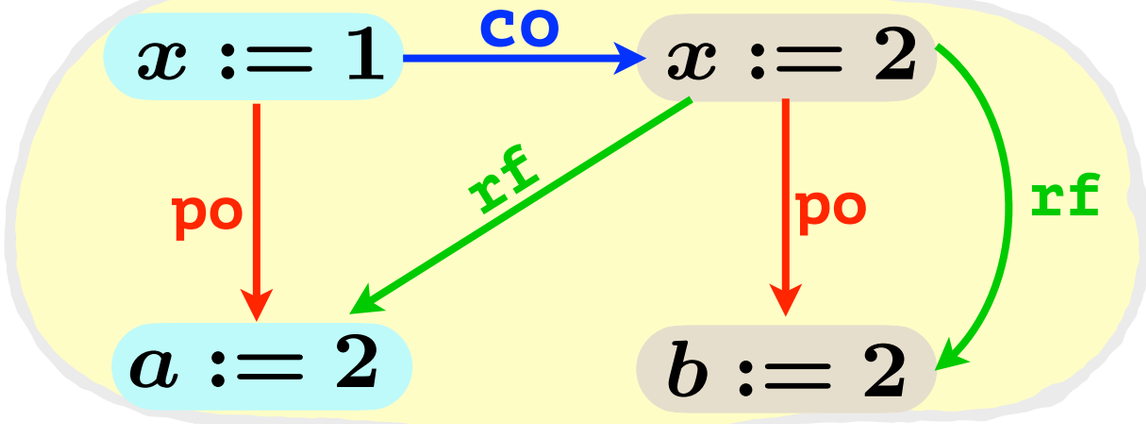
(super-optimal) DPOR

$x := 1$

$x := 2$

$a := 2$

$b := 2$

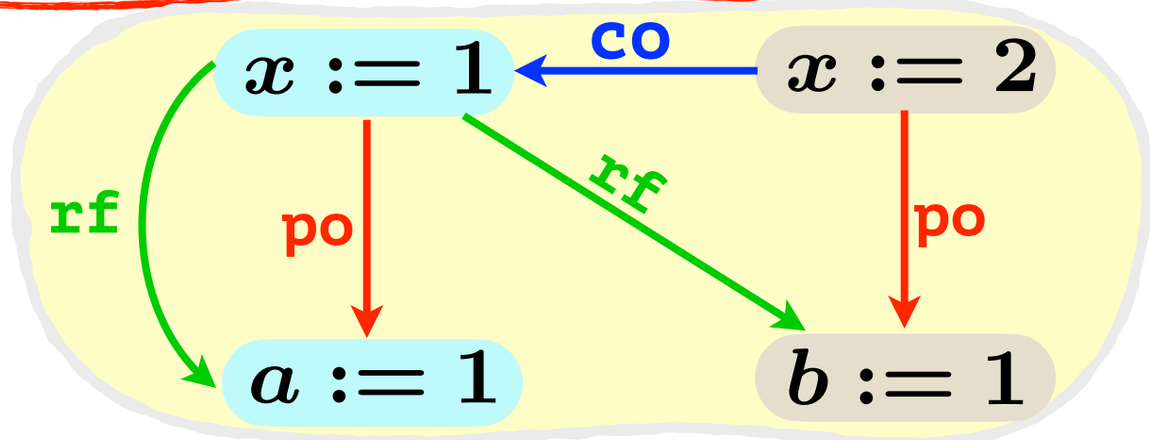


$x := 2$

$x := 1$

$b := 1$

$a := 1$

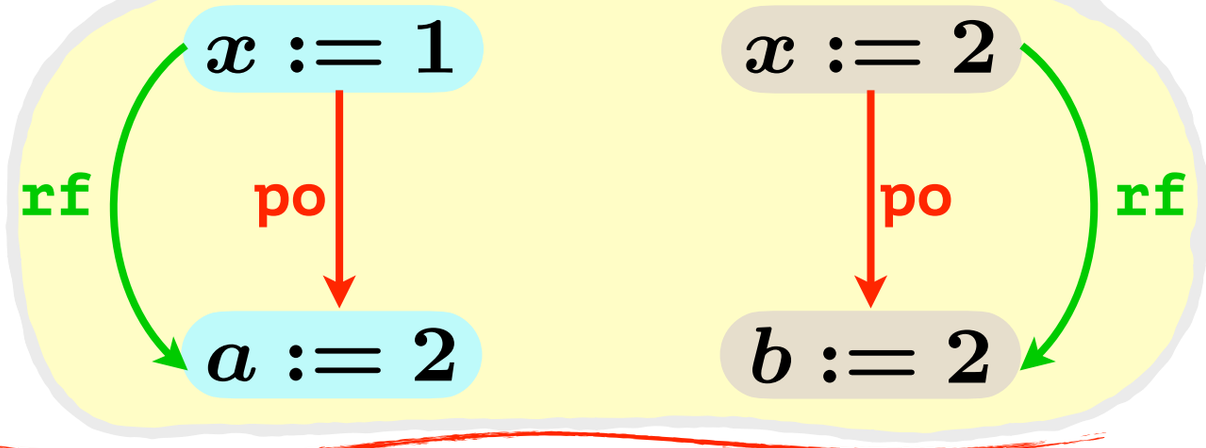


$x := 1$

$a := 1$

$x := 2$

$b := 2$

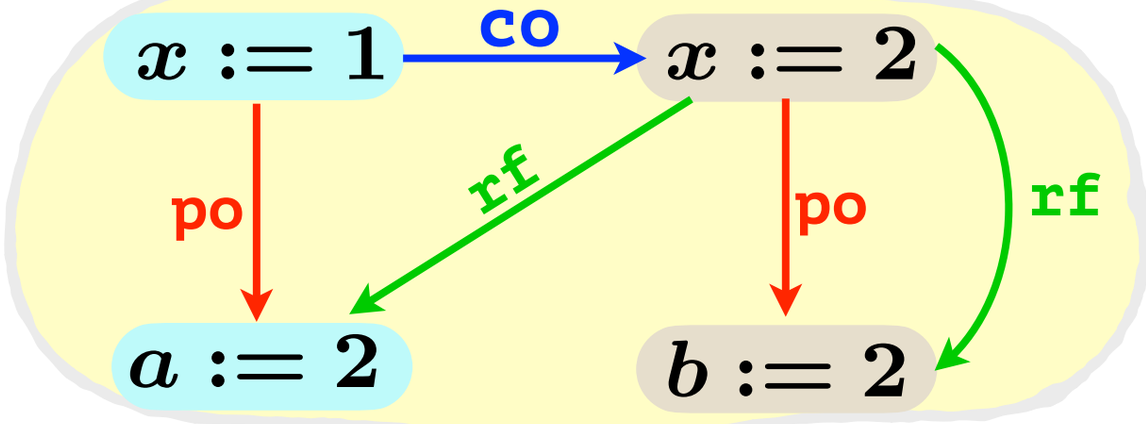


$x := 1$

$x := 2$

$a := 2$

$b := 2$

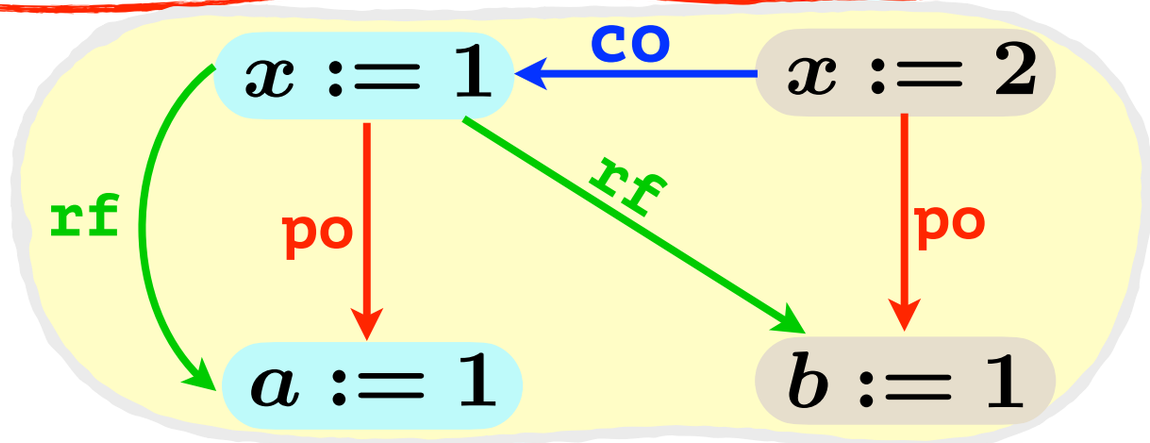


$x := 2$

$x := 1$

$b := 1$

$a := 1$

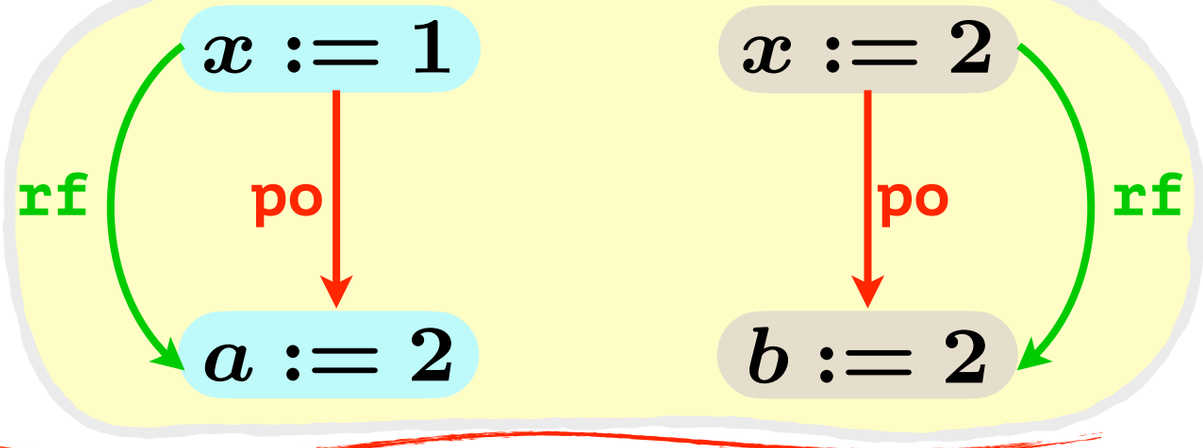


$x := 1$

$a := 1$

$x := 2$

$b := 2$



Prog → (super-optimal) DPOR → **only traces with different po and rf**

Release-Acquire (RA)

$$a := x \;\|\; x := 1 \;\|\; y := 1 \;\|\; c := y$$
$$b := x \;\|\; y := 2 \;\|\; x := 2 \;\|\; d := y$$

τ

RA: $\forall x.\ acyclic(\ \mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x\ )$

Release-Acquire (RA)

$$a := x \;\|\; b := x \;\|\; x := 1 \;\|\; y := 2 \;\|\; y := 1 \;\|\; x := 2 \;\|\; c := y \;\|\; d := y$$

$x := 1$

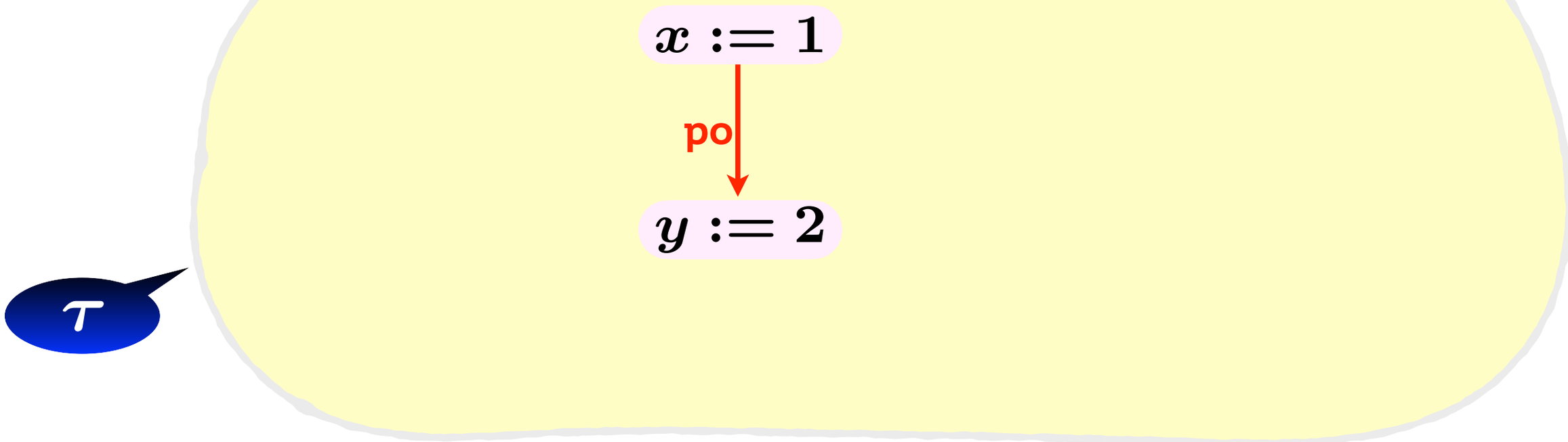
$y := 2$

τ

RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)

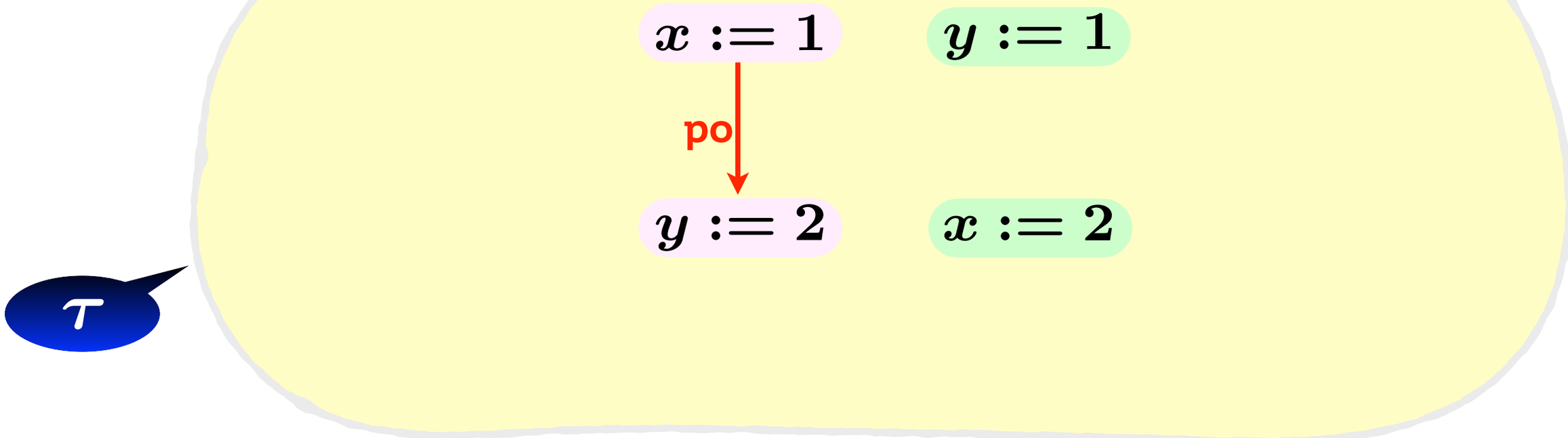
$$a := x \;\|\; b := x \;\|\; x := 1 \;\|\; y := 2 \;\|\; y := 1 \;\|\; x := 2 \;\|\; c := y \;\|\; d := y$$



RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

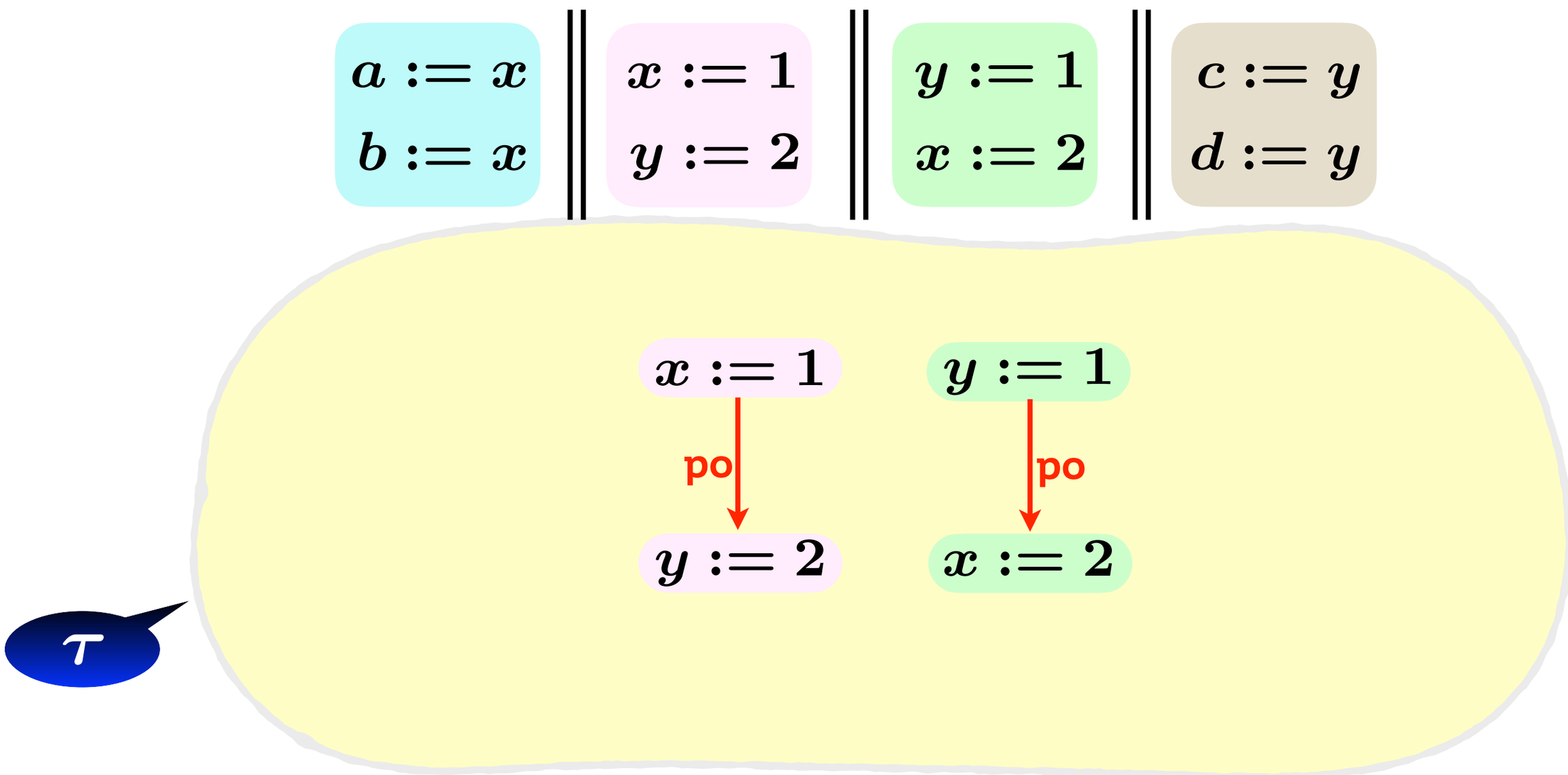
Release-Acquire (RA)

$$a := x \;\;\; b := x \quad \| \quad x := 1 \;\;\; y := 2 \quad \| \quad y := 1 \;\;\; x := 2 \quad \| \quad c := y \;\;\; d := y$$



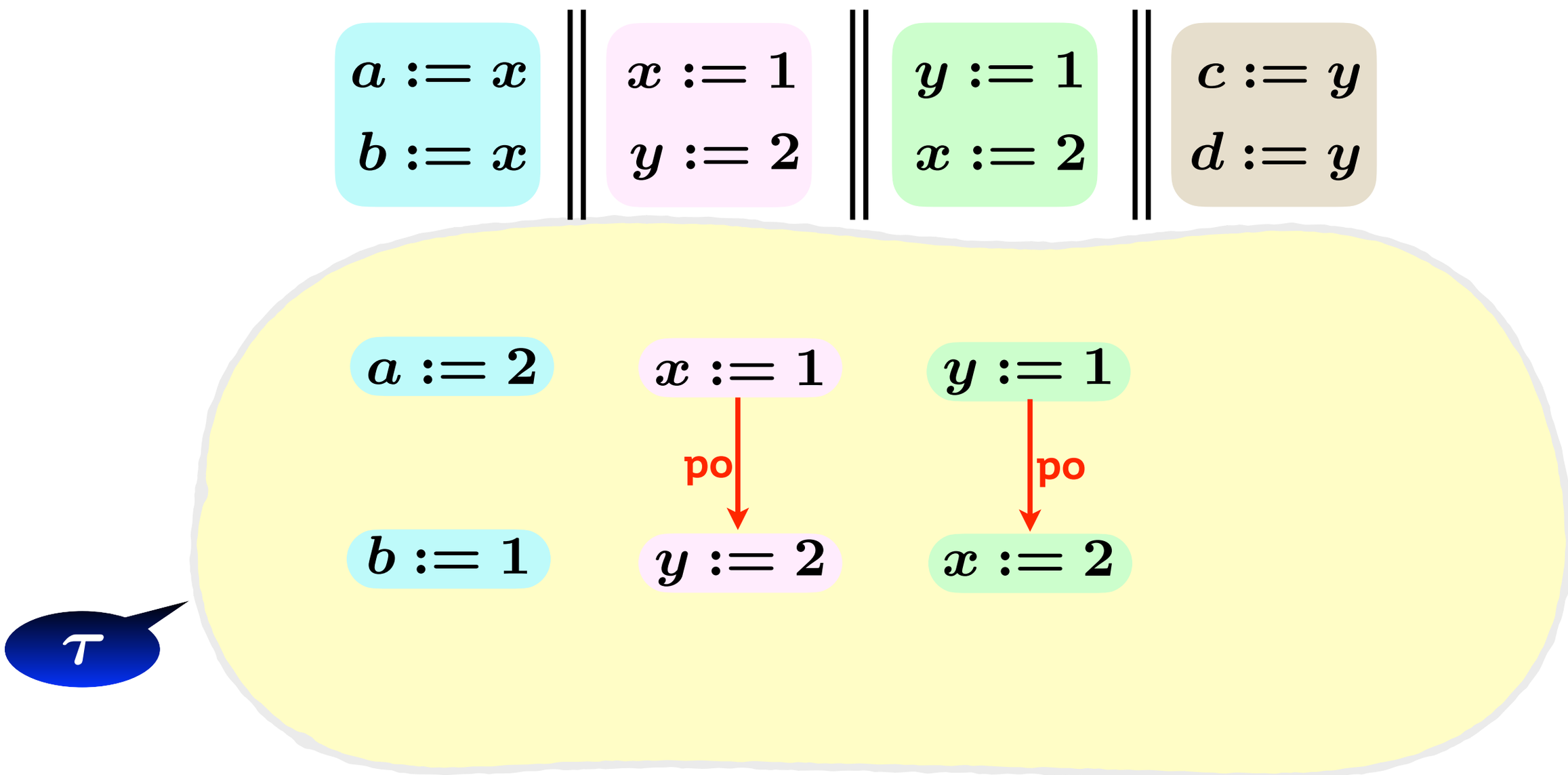
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



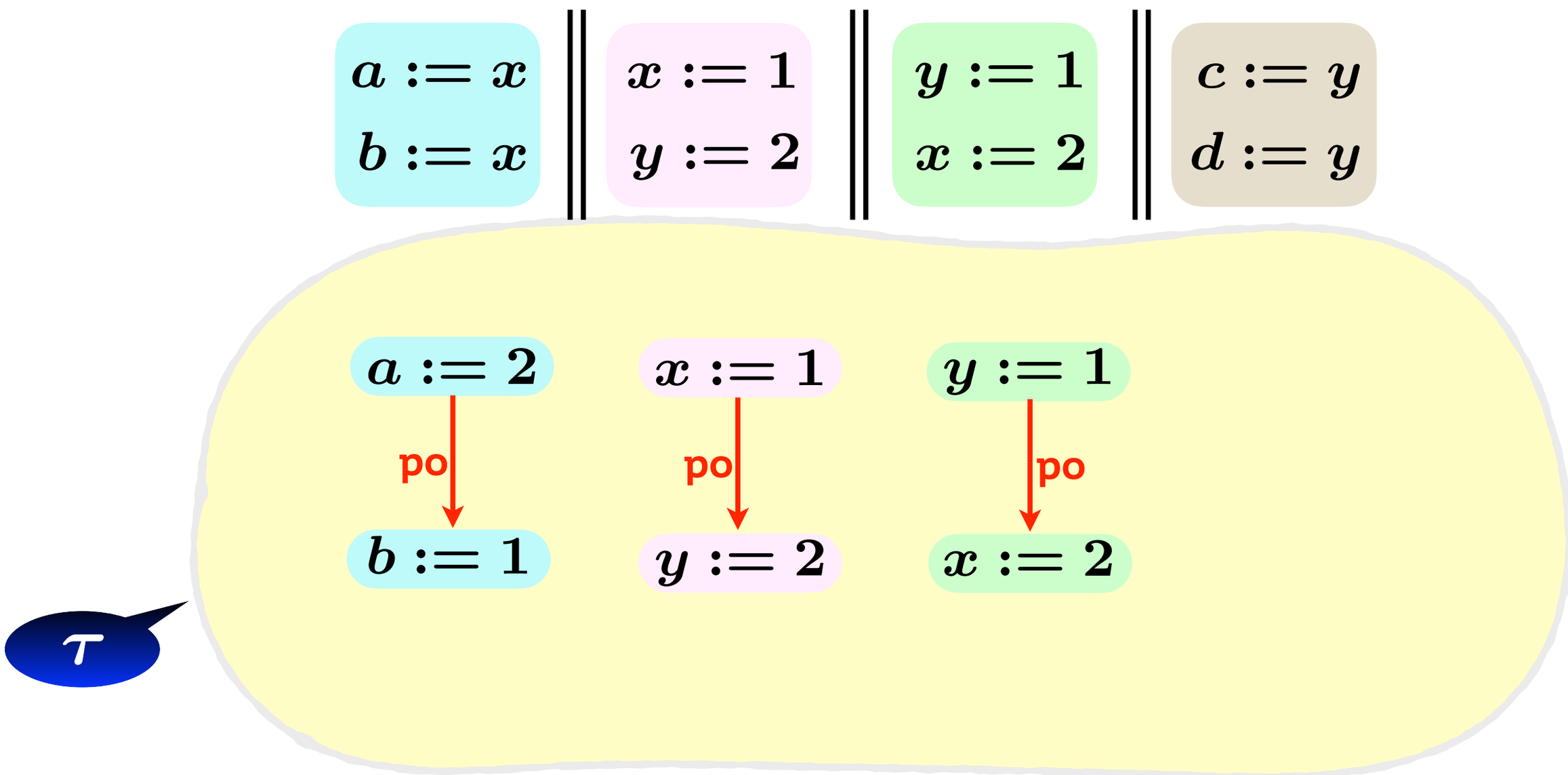
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



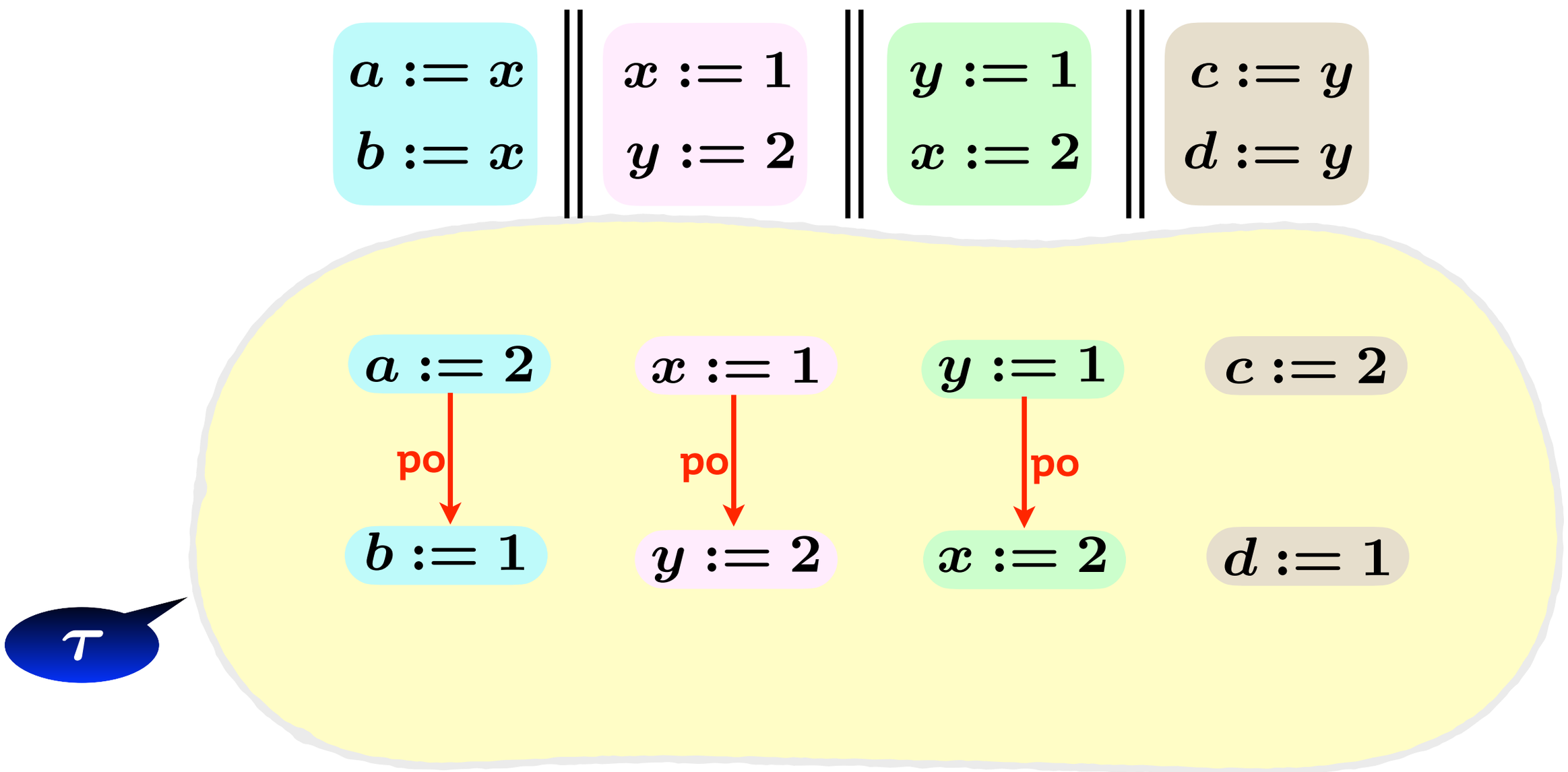
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



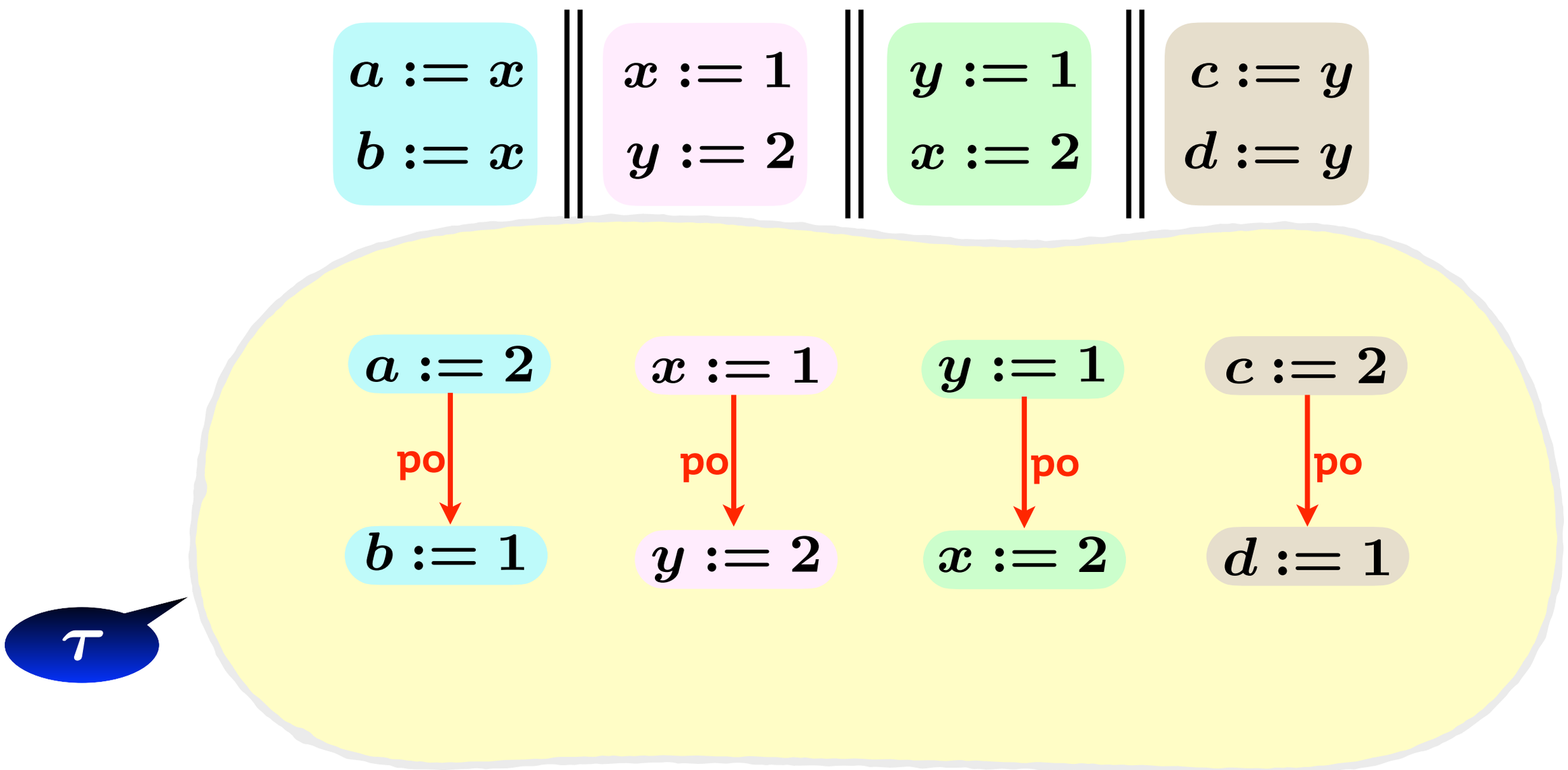
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



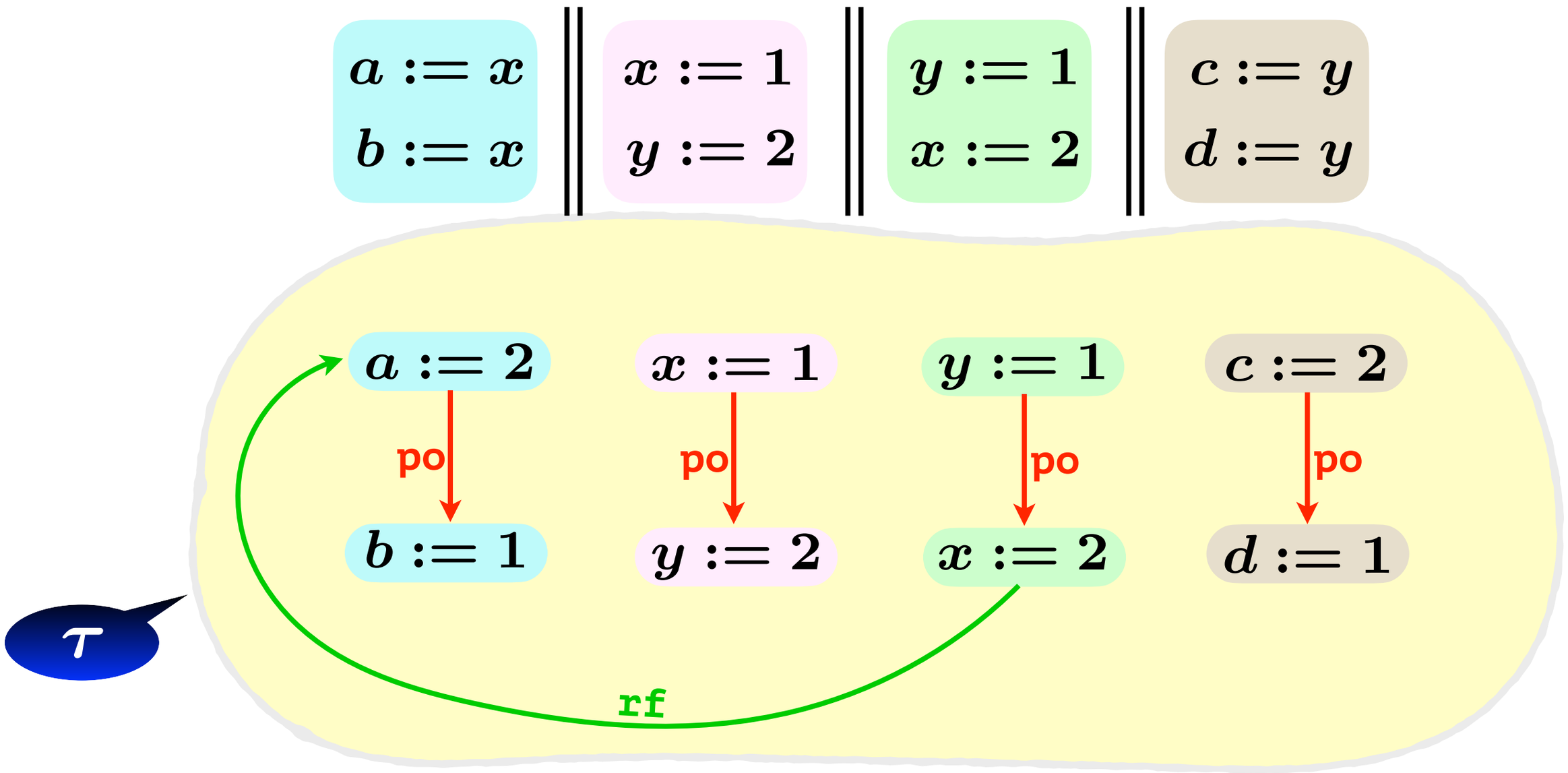
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



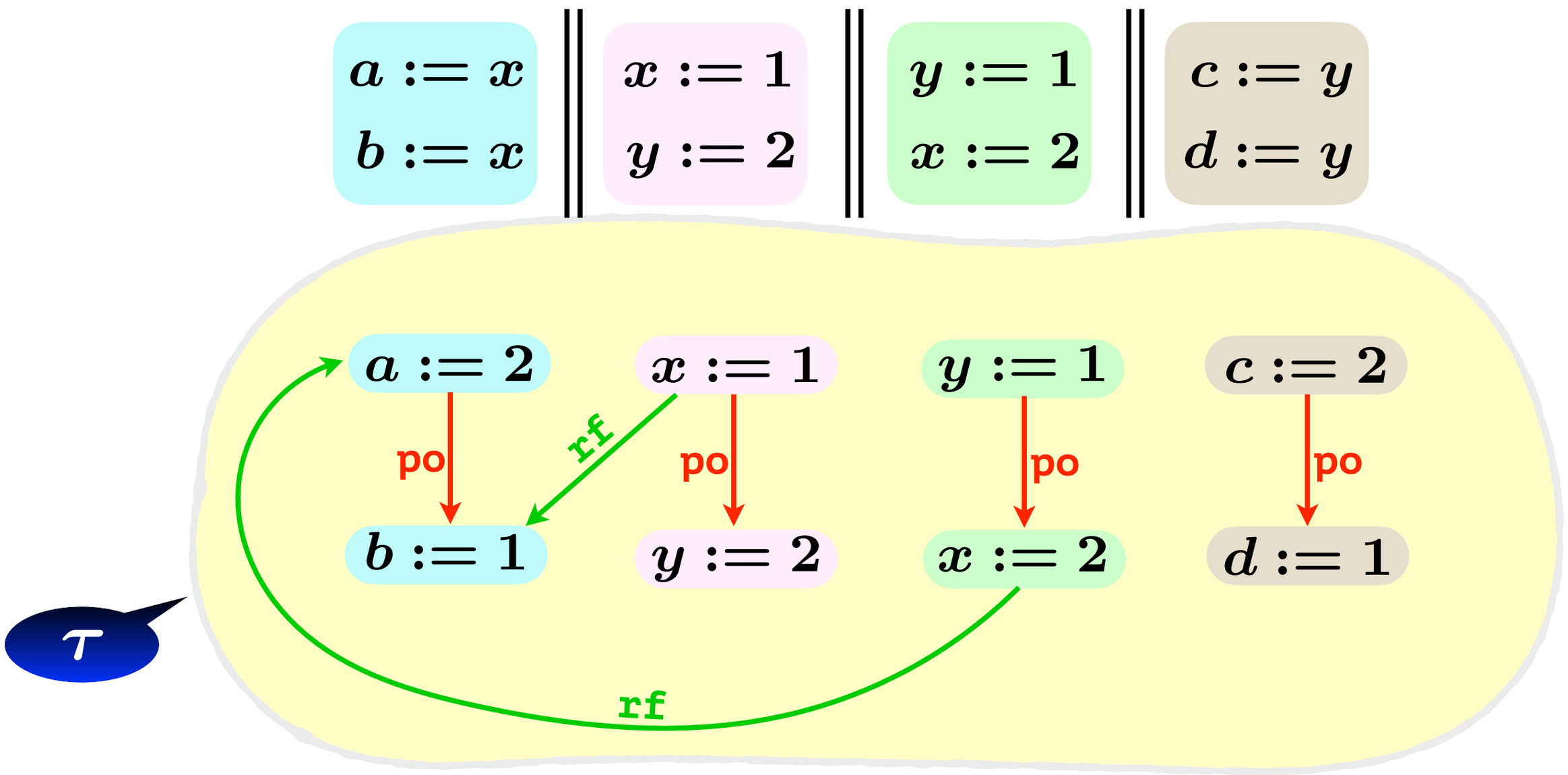
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



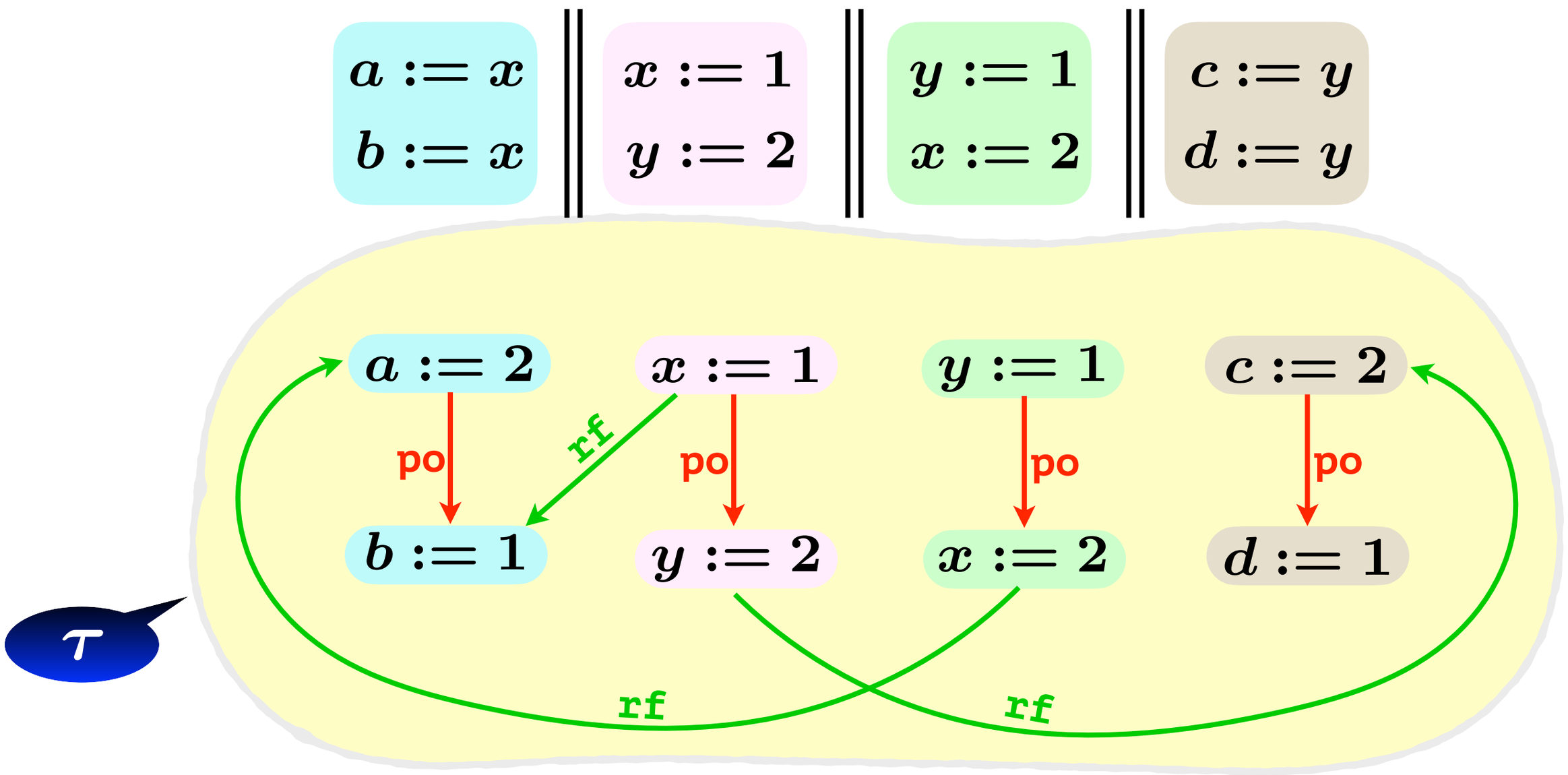
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



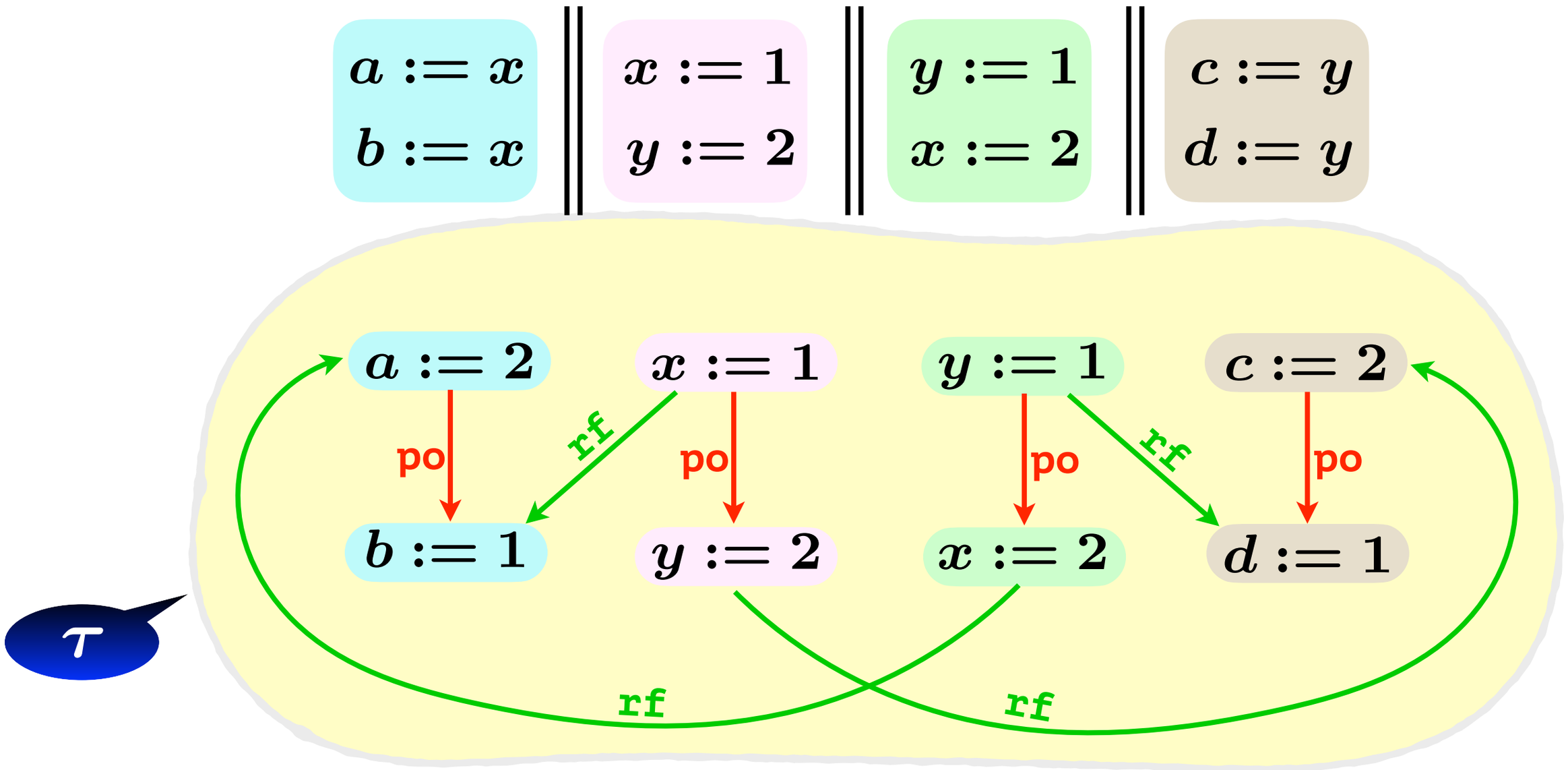
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



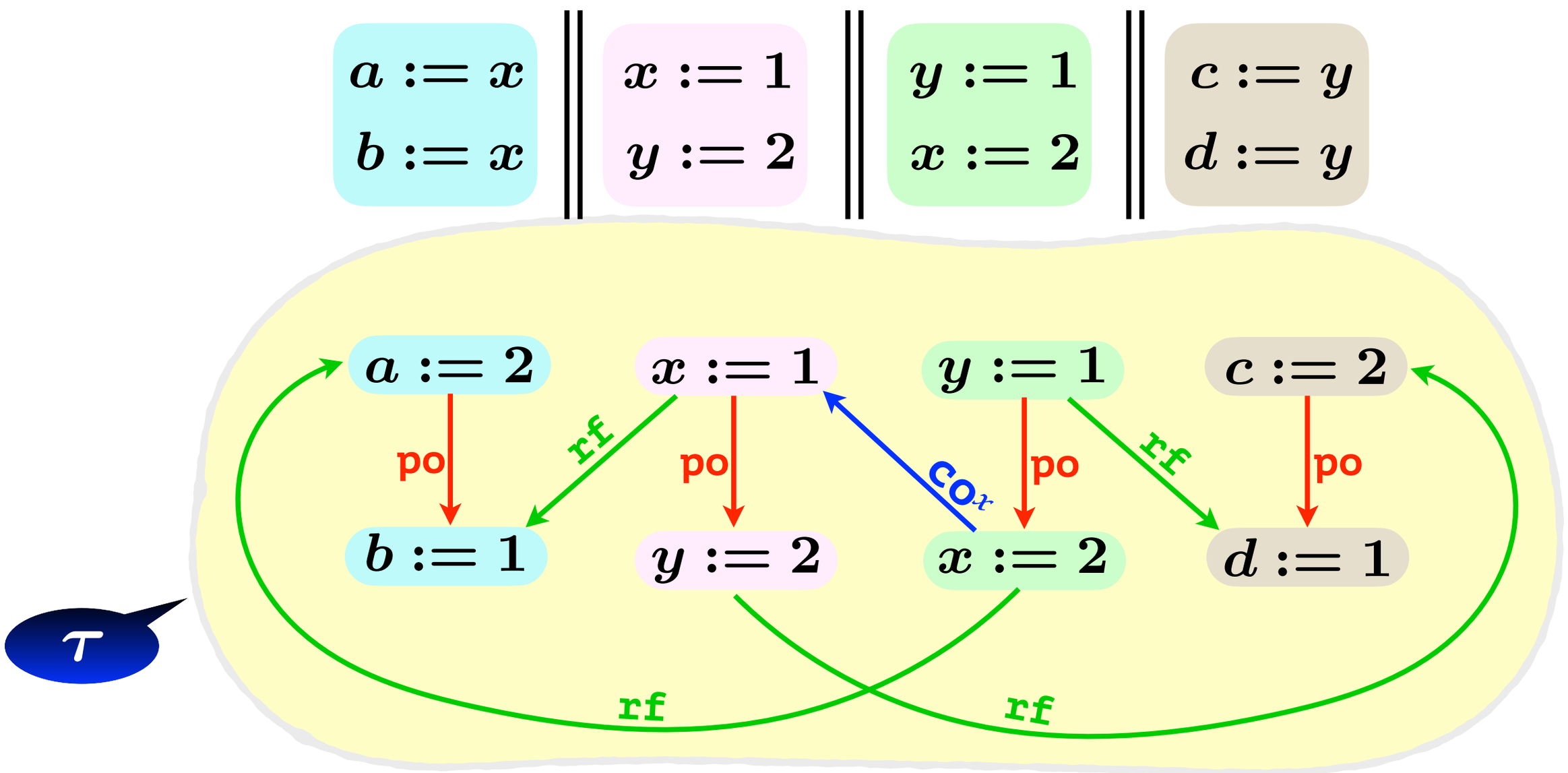
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



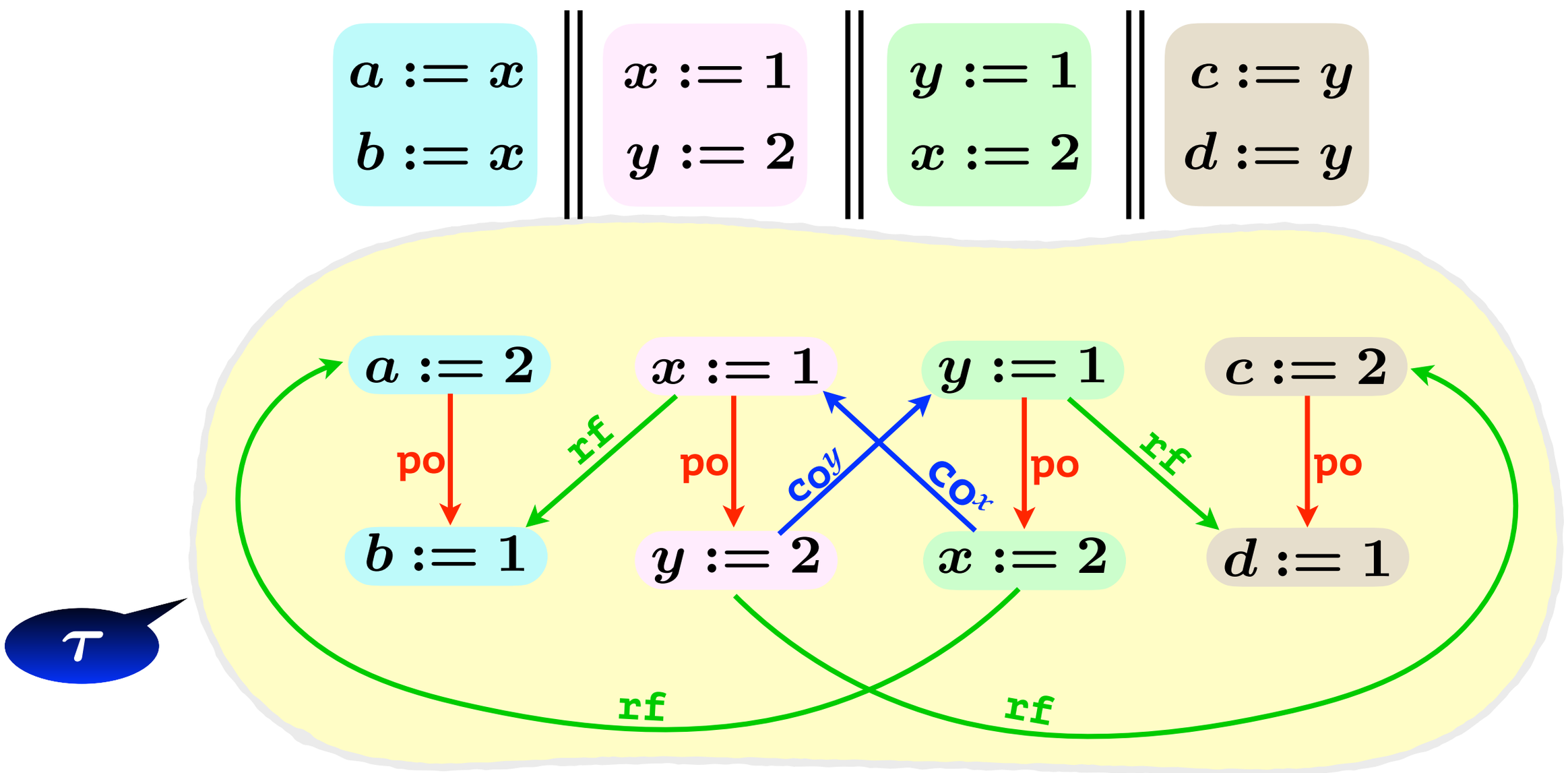
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



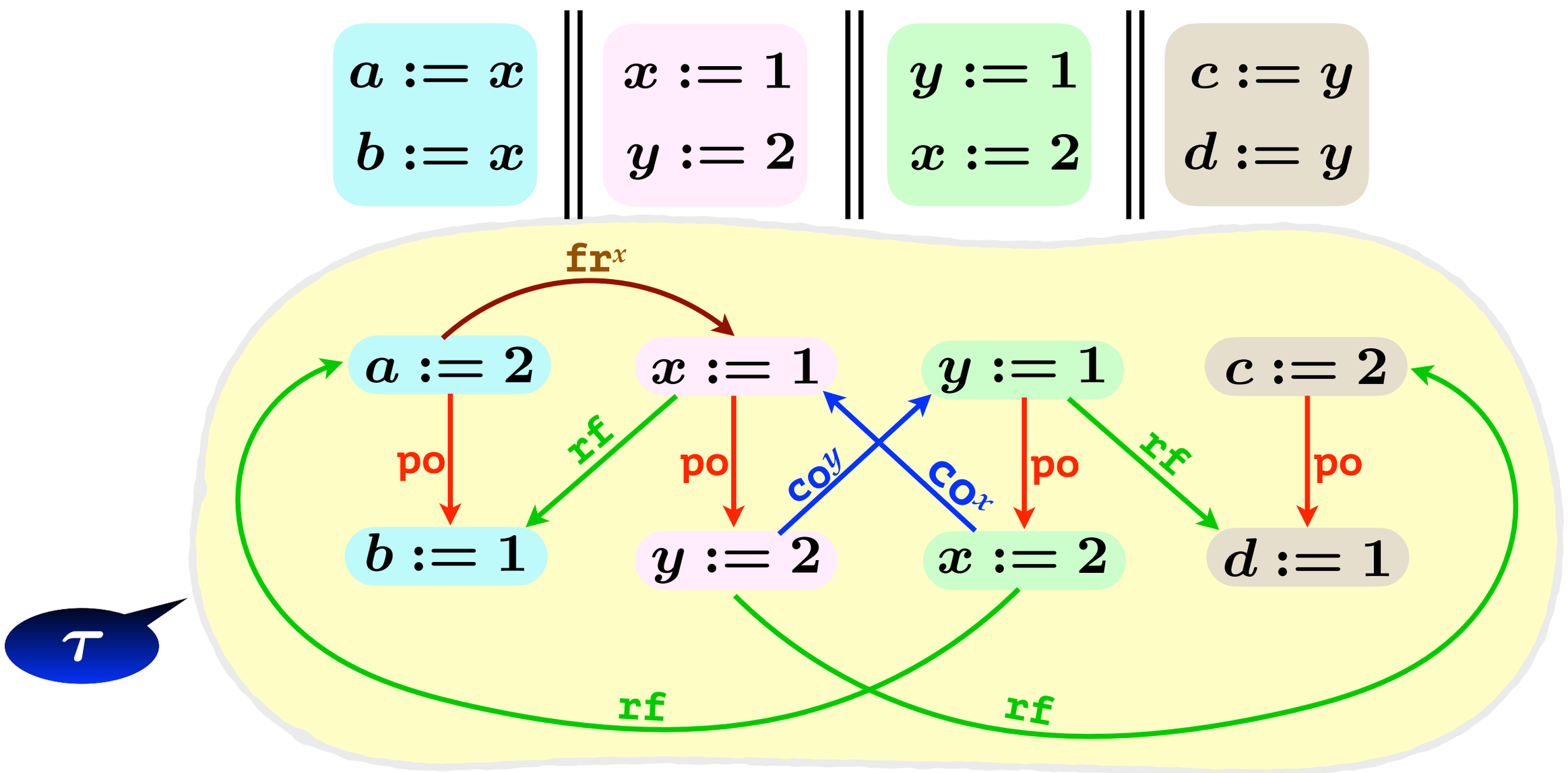
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



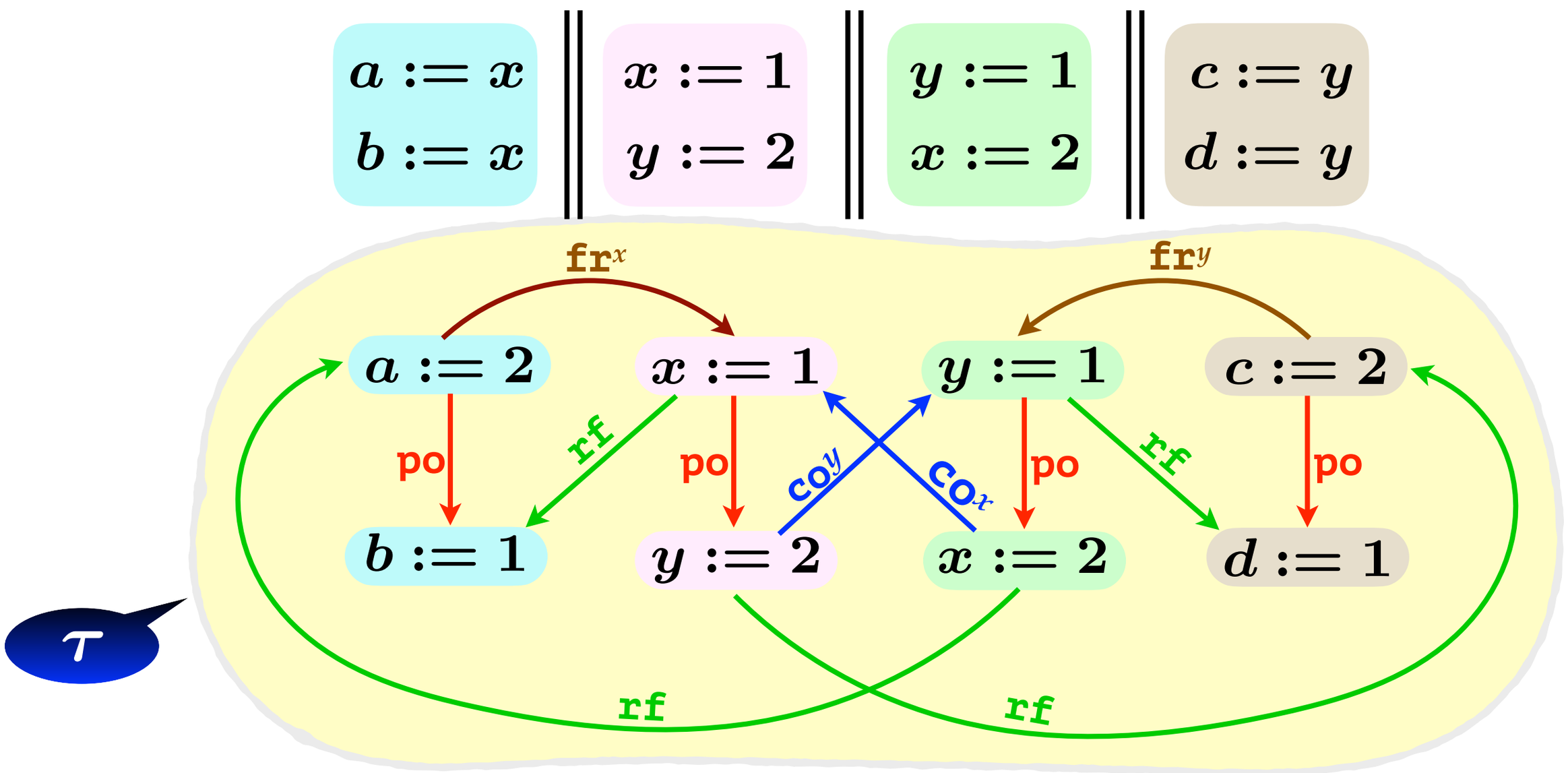
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



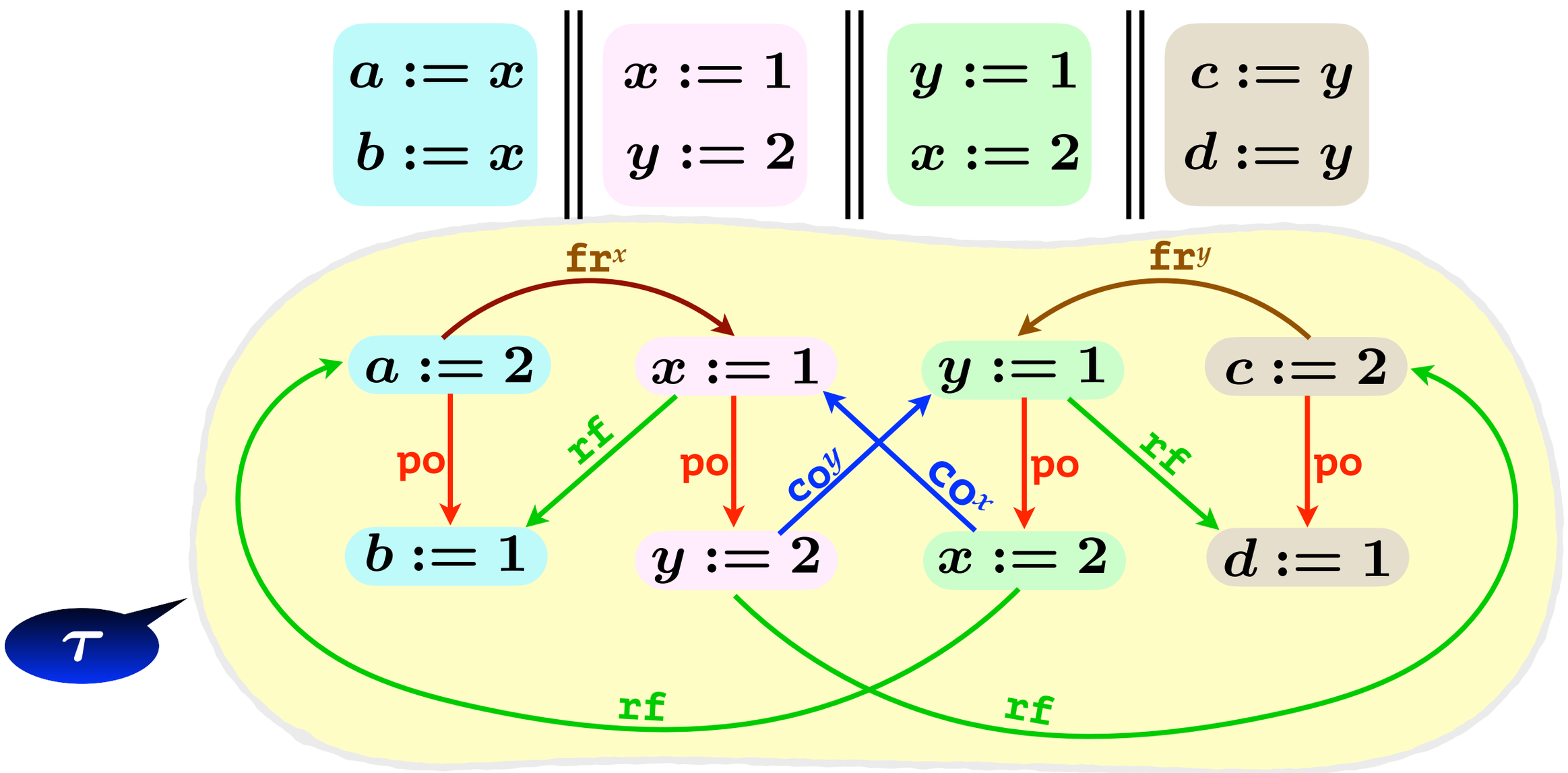
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



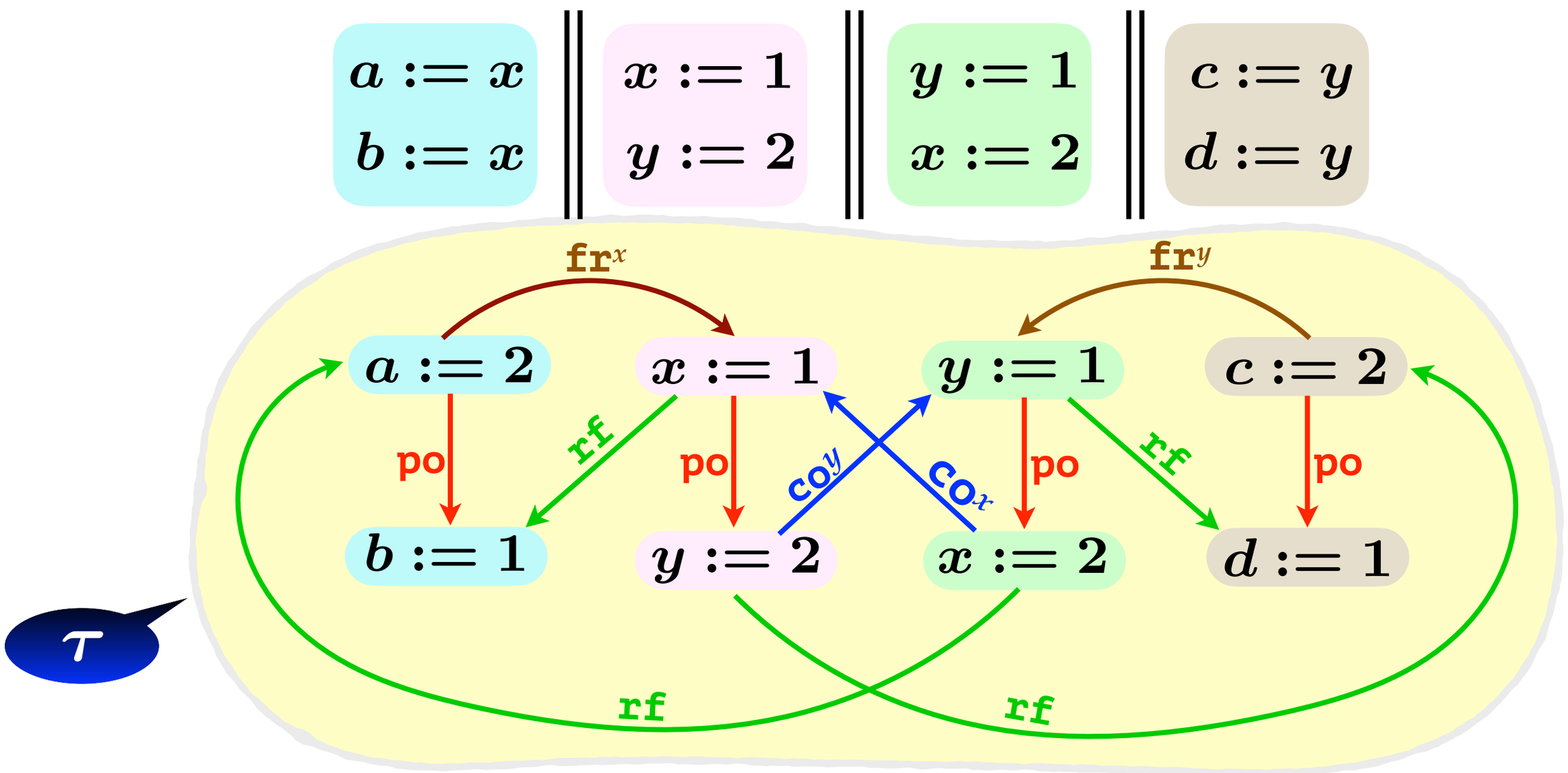
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Release-Acquire (RA)



RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$ $\quad \tau \models \mathrm{RA}$

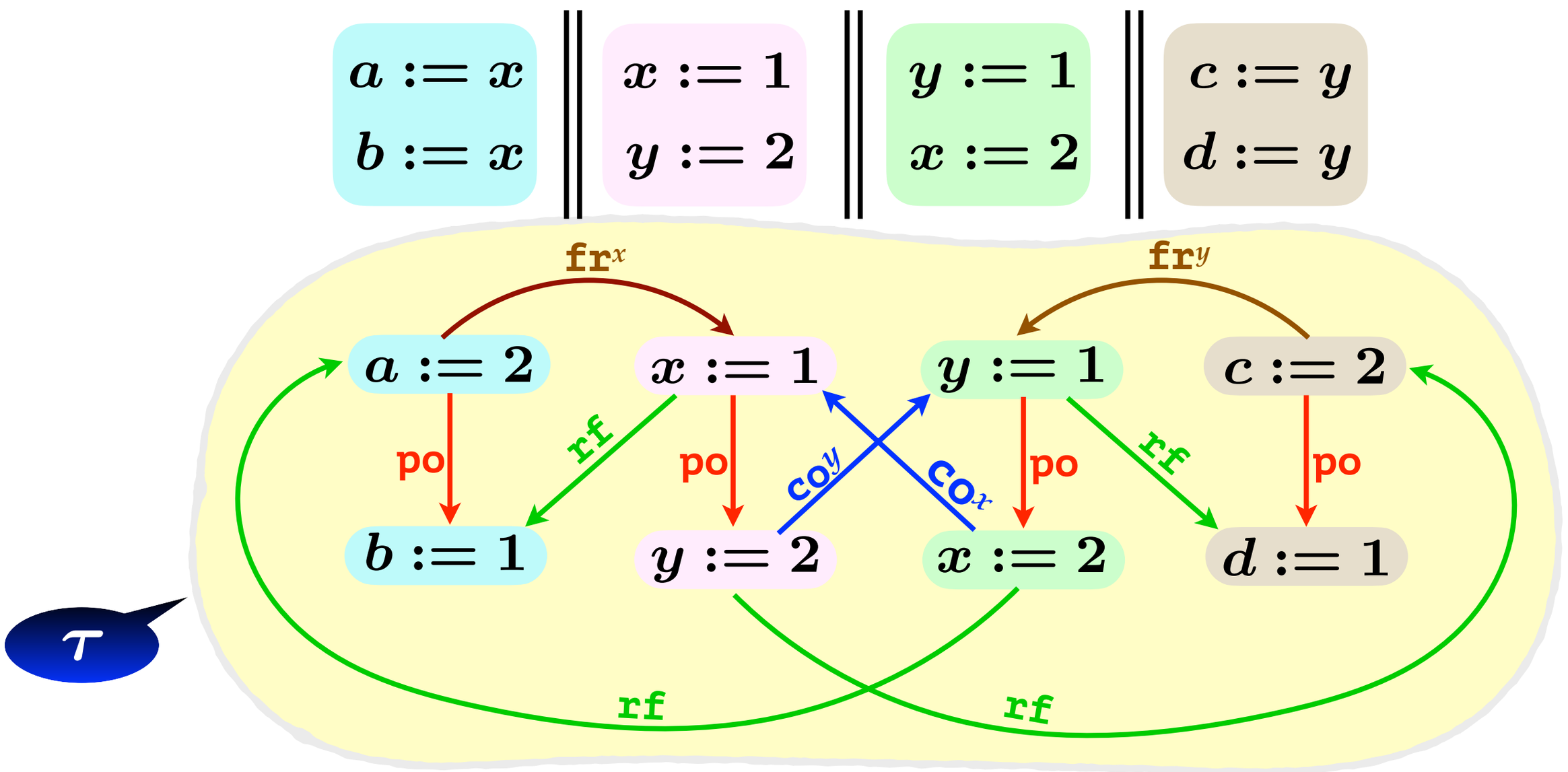
Release-Acquire (RA)



RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$ $\quad \tau \models \mathbf{RA}$

SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$

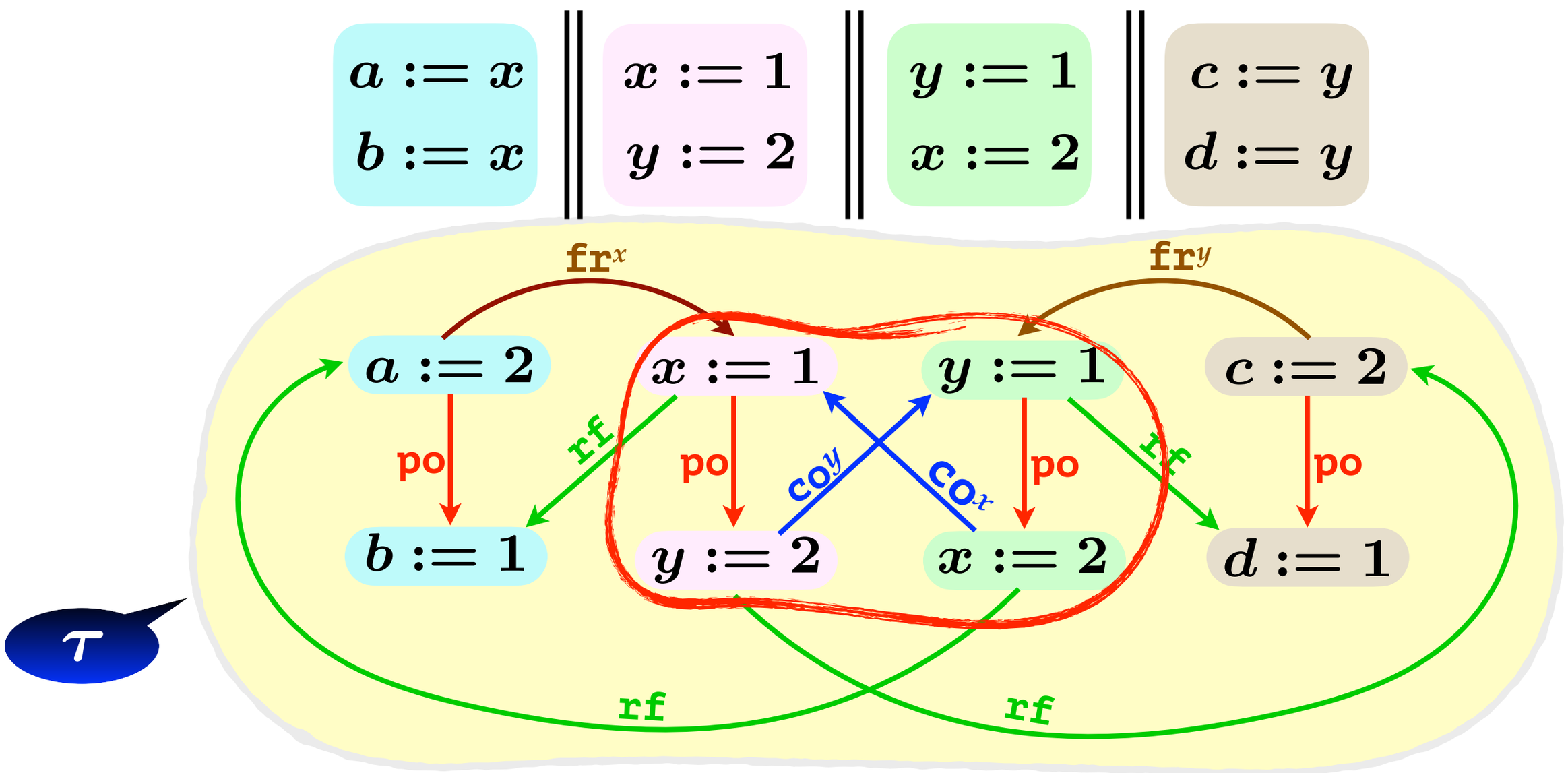
Release-Acquire (RA)



RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$ $\quad \tau \models \mathbf{RA}$

SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$ $\quad \tau \not\models \mathbf{SC}$

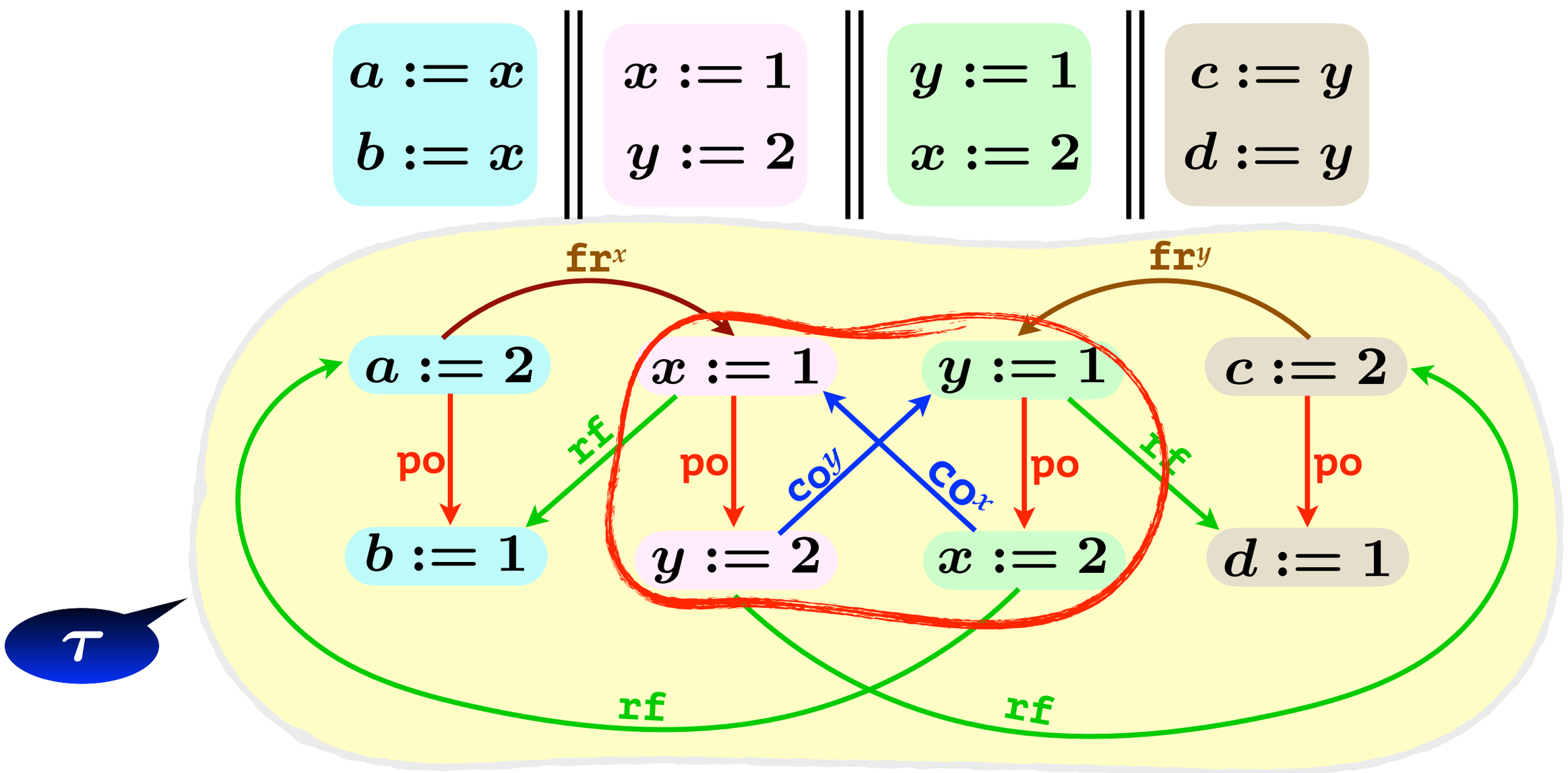
Release-Acquire (RA)



RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$ $\quad \tau \models \mathbf{RA}$

SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$ $\quad \tau \not\models \mathbf{SC}$

Release-Acquire (RA)

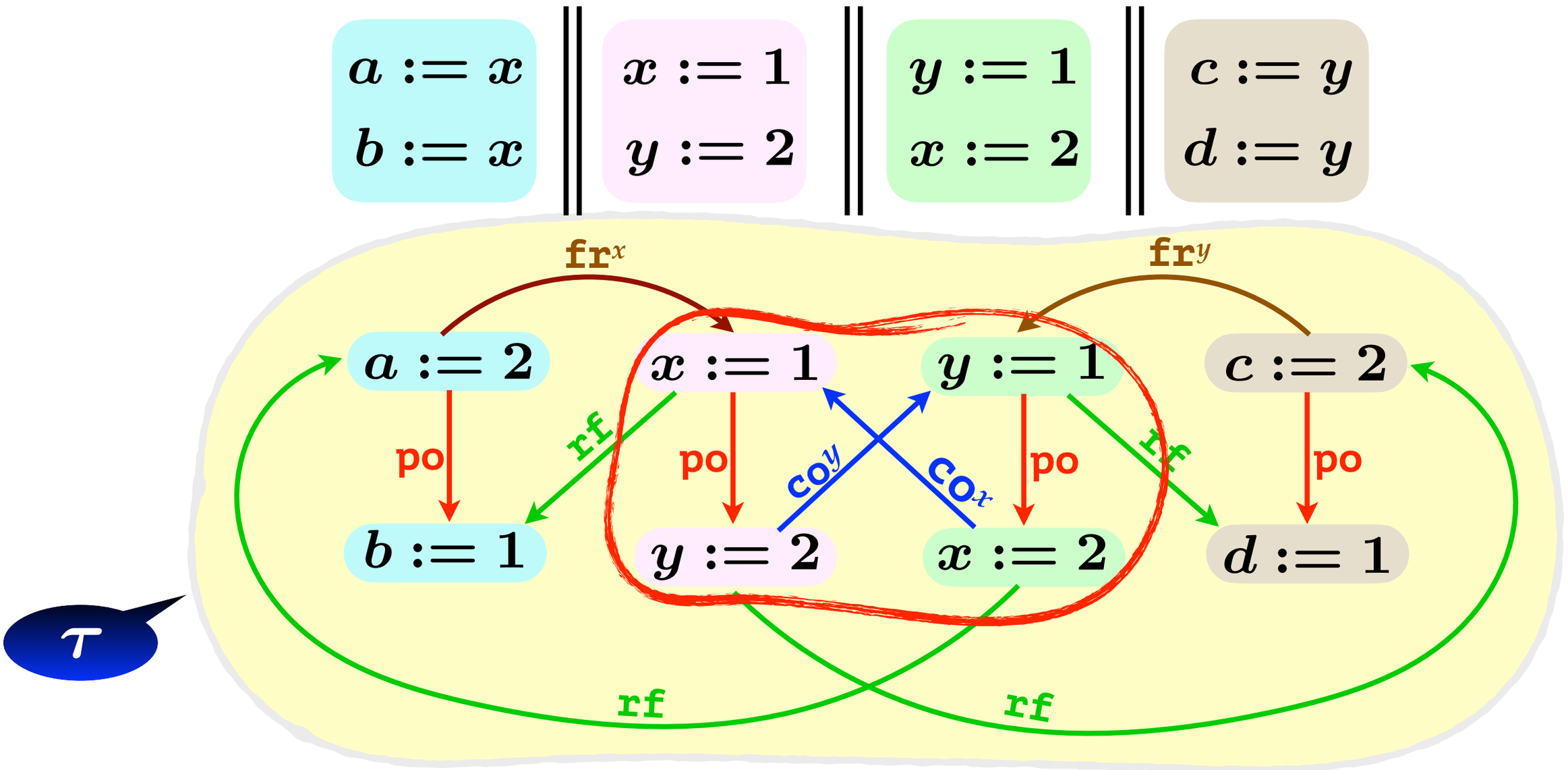


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$ $\quad \tau \models \text{RA}$

SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$ $\quad \tau \not\models \text{SC}$

$(a = 2) \wedge (b = 1) \wedge (c = 2) \wedge (d = 1)?$

Release-Acquire (RA)



RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$ $\quad \tau \models \text{RA}$

SC: $acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co} \cup \mathtt{fr})$ $\quad \tau \not\models \text{SC}$

$(a = 2) \wedge (b = 1) \wedge (c = 2) \wedge (d = 1)?$

DPOR under the RA-semantics

(OOPSLA'2018)

- Sound
  - only valid traces
- Complete
  - all valid traces
- Optimal
  - different po+rf
- Deadlock-free

DPOR under the RA-semantics

- Build traces one after one
- For a given trace τ :
  - Build τ incrementally
  - extend τ by one event e at a time

- Sound
  - only valid traces
- Complete
  - all valid traces
- Optimal
  - different po+rf
- Deadlock-free

DPOR under the RA-semantics

- Build traces one after one
- For a given trace τ :
  - Build τ incrementally
  - extend τ by one event e at a time

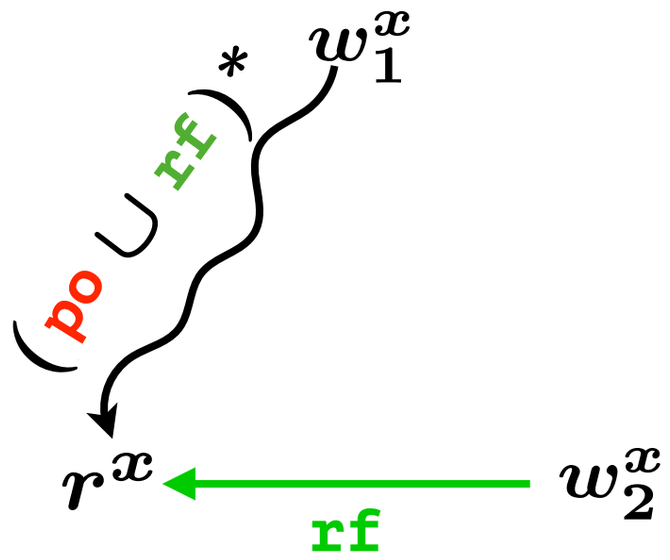
DPOR under the RA-semantics

- Build traces one after one
- For a given trace τ :
  - Build τ incrementally
  - extend τ by one event e at a time

DPOR under the RA-semantics

- Build traces one after one
- For a given trace τ :
  - Build τ incrementally
  - extend τ by one event e at a time

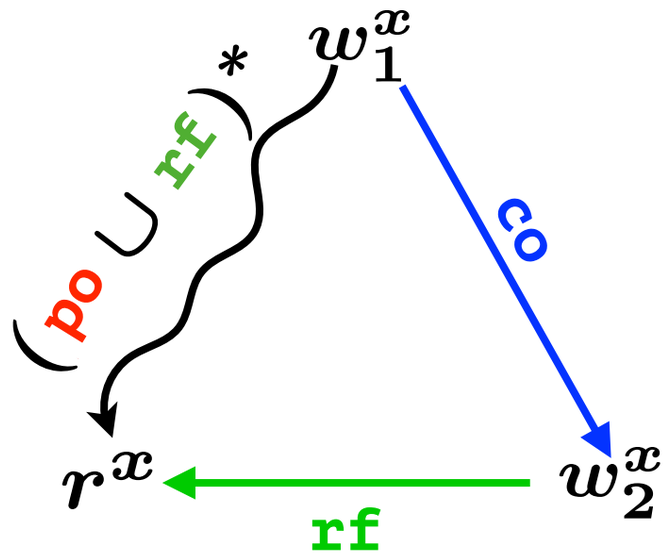
Saturation



- Add co-edges by demand
- never generate two traces with identical po and rf

$$\text{RA: } \forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$$

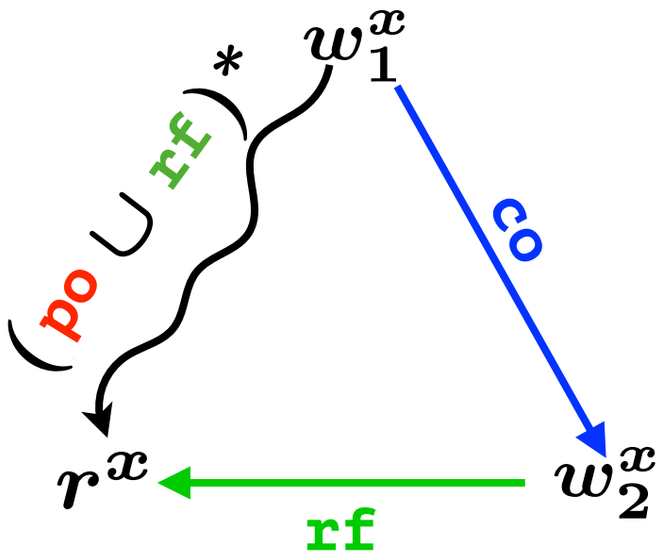
Saturation

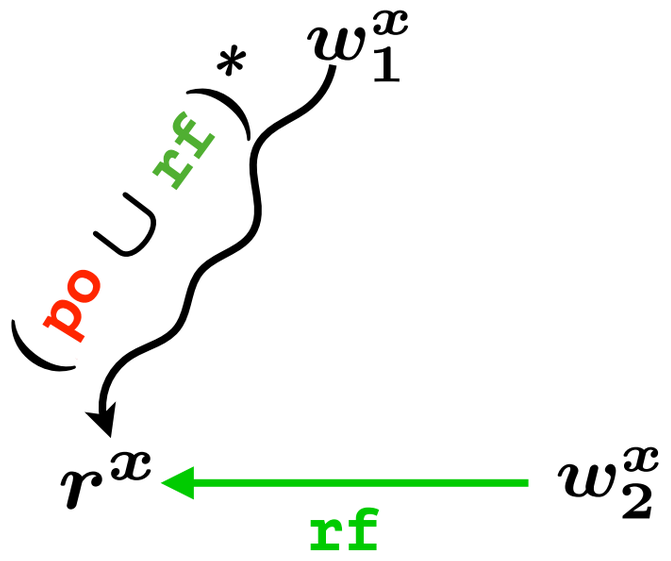


- Add co-edges by demand
- never generate two traces with identical po and rf

$$\text{RA: } \forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$$

Saturation

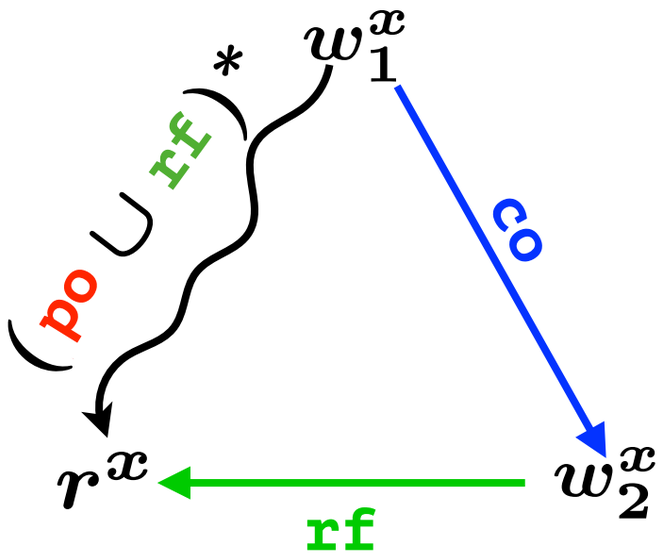


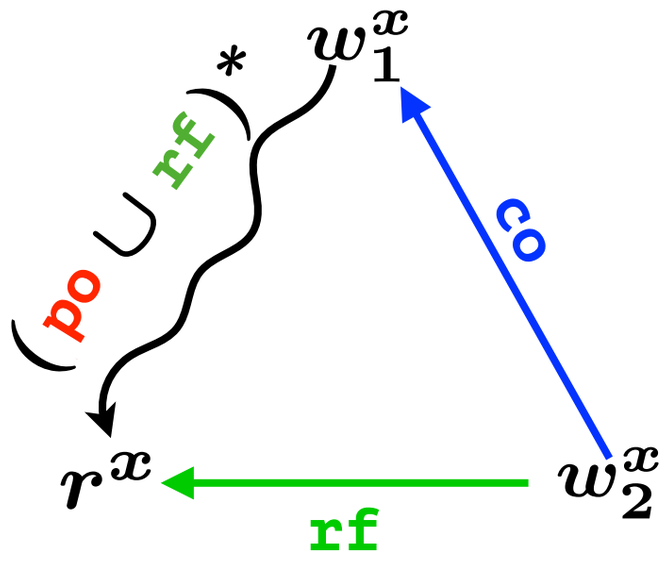


- **Add co-edges by demand**
- **never generate two traces with identical po and rf**

$$\text{RA: } \forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$$

Saturation

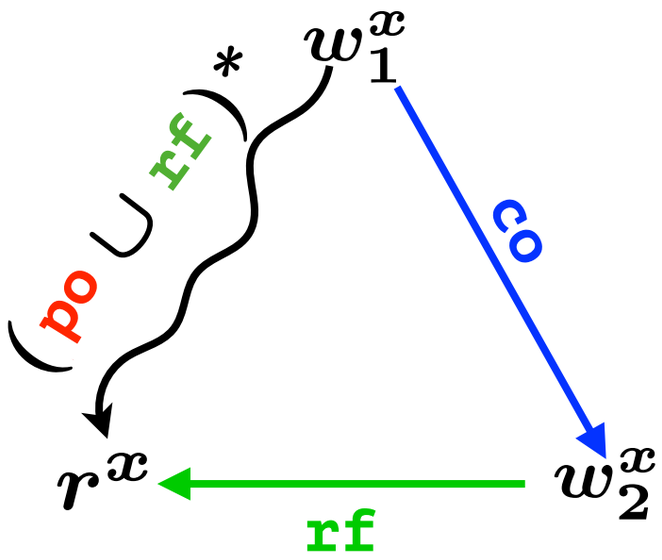


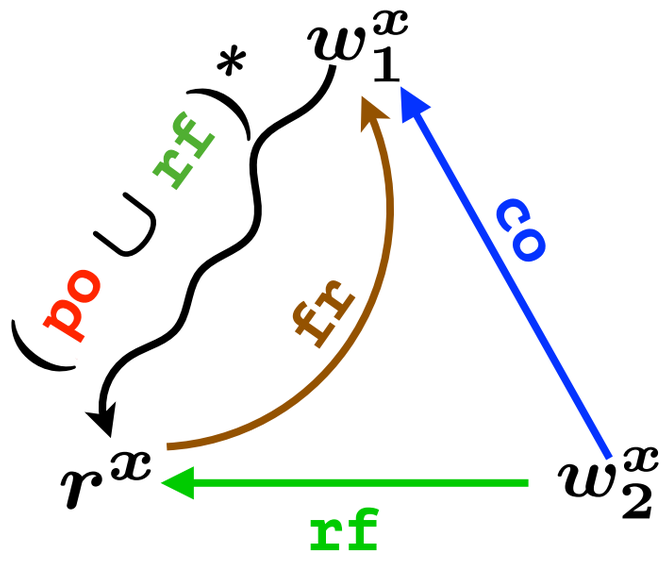


- Add co-edges by demand
- never generate two traces with identical po and rf

RA: $\forall x.\ acyclic(\mathbf{po} \cup \mathbf{rf} \cup \mathbf{co}^x \cup \mathbf{fr}^x)$

Saturation

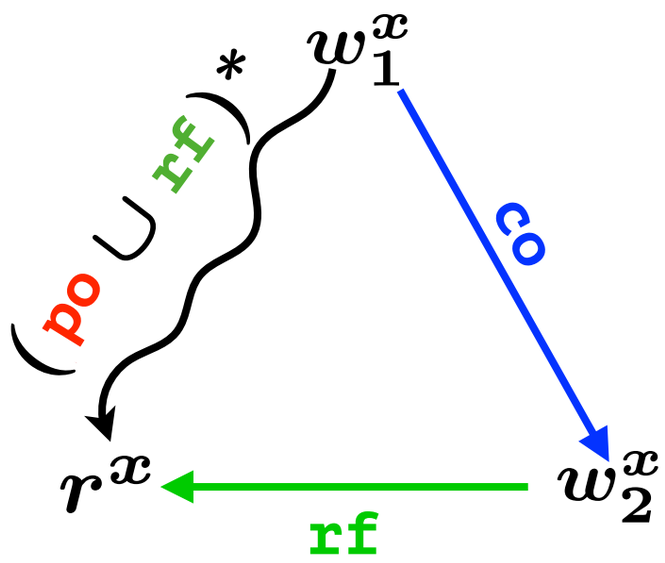


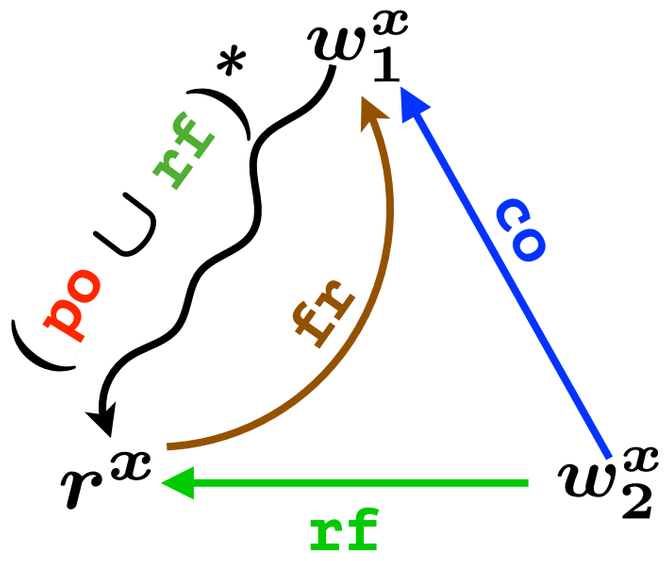


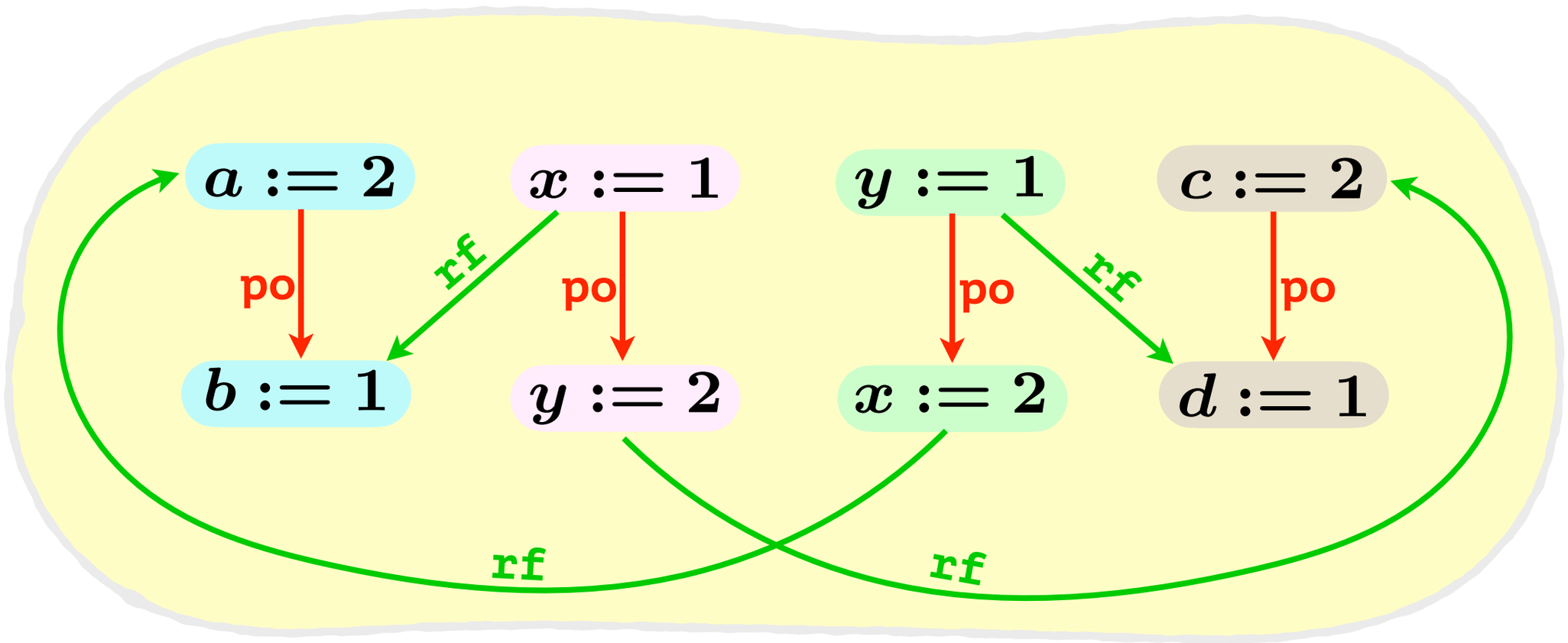
- Add co-edges by demand
- never generate two traces with identical po and rf

RA: $\forall x.\ acyclic(\mathbf{po} \cup \mathbf{rf} \cup \mathbf{co}^x \cup \mathbf{fr}^x)$

Saturation

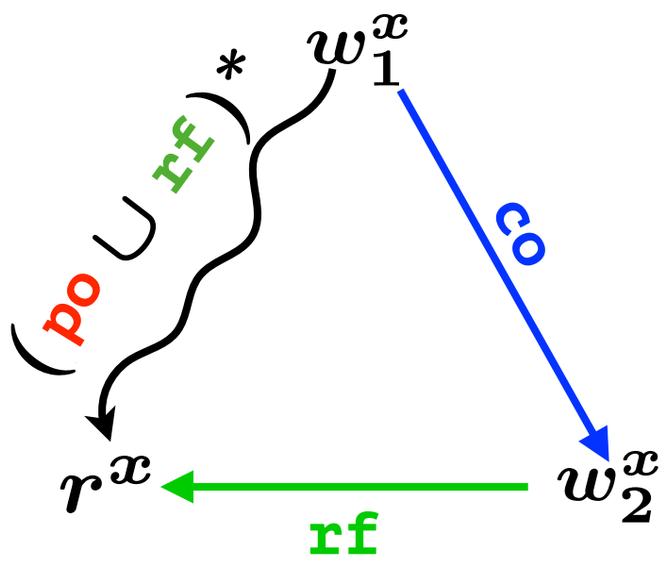


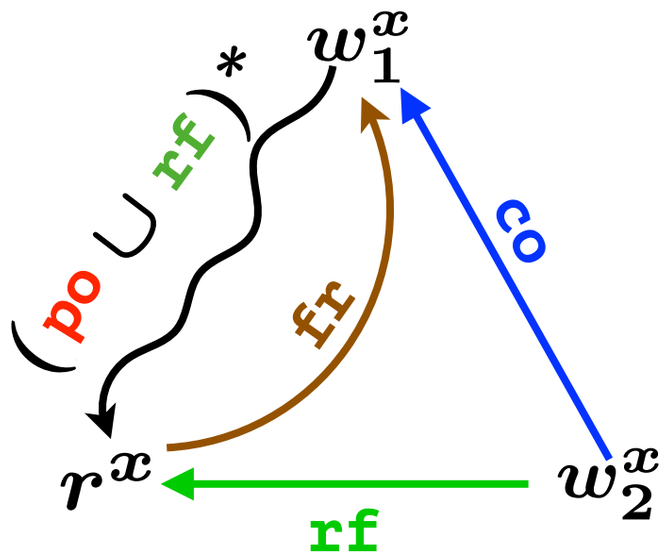


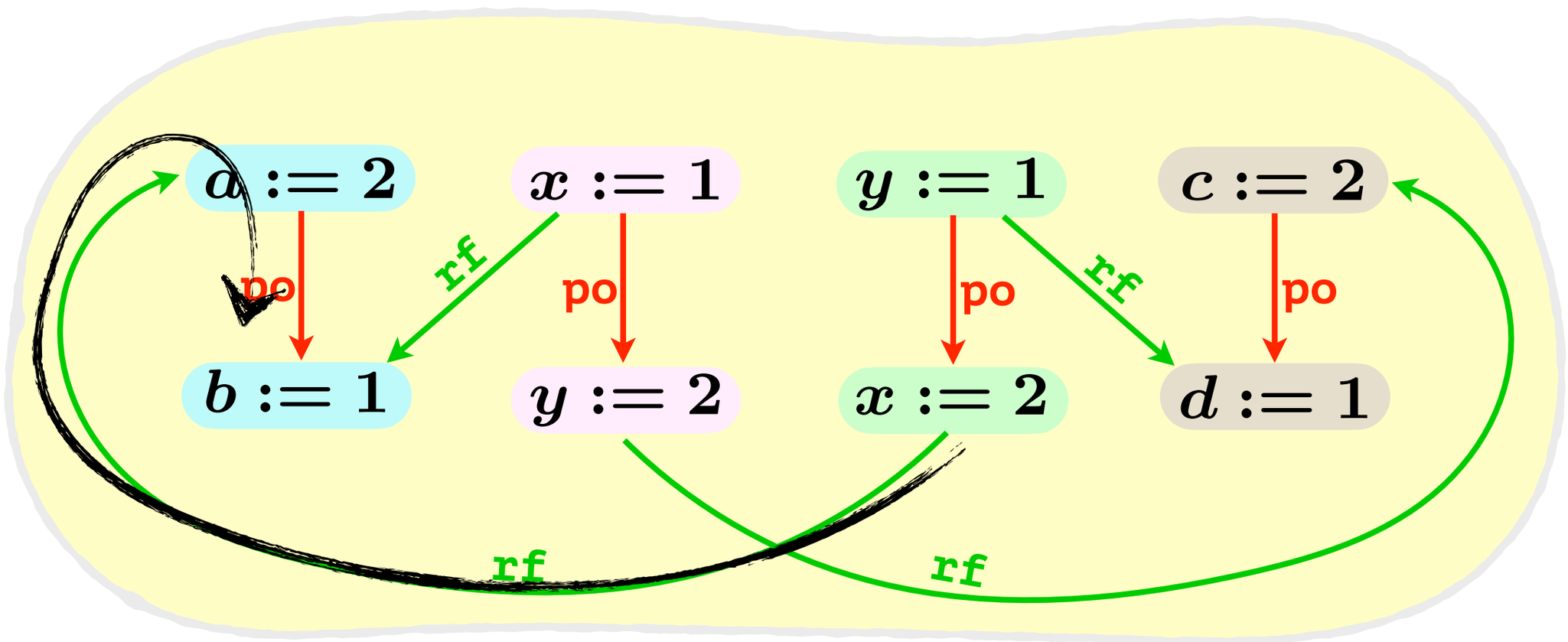


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

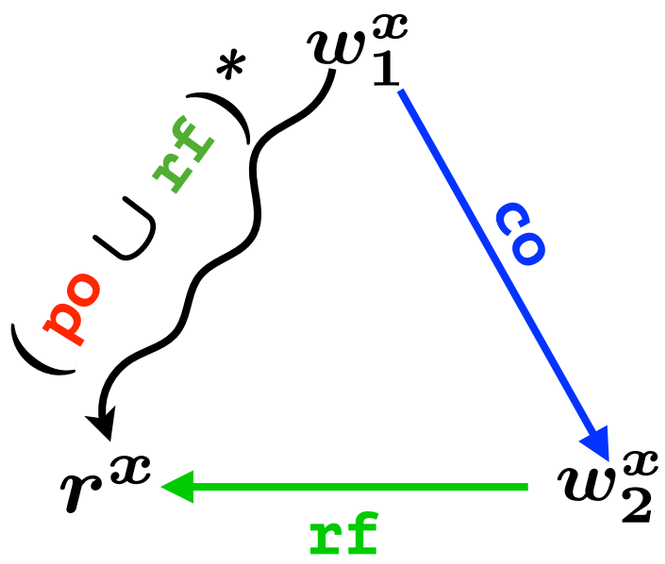


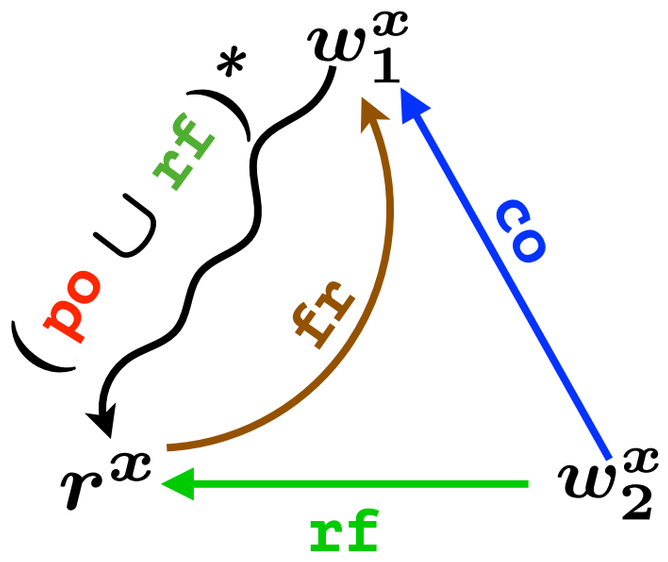


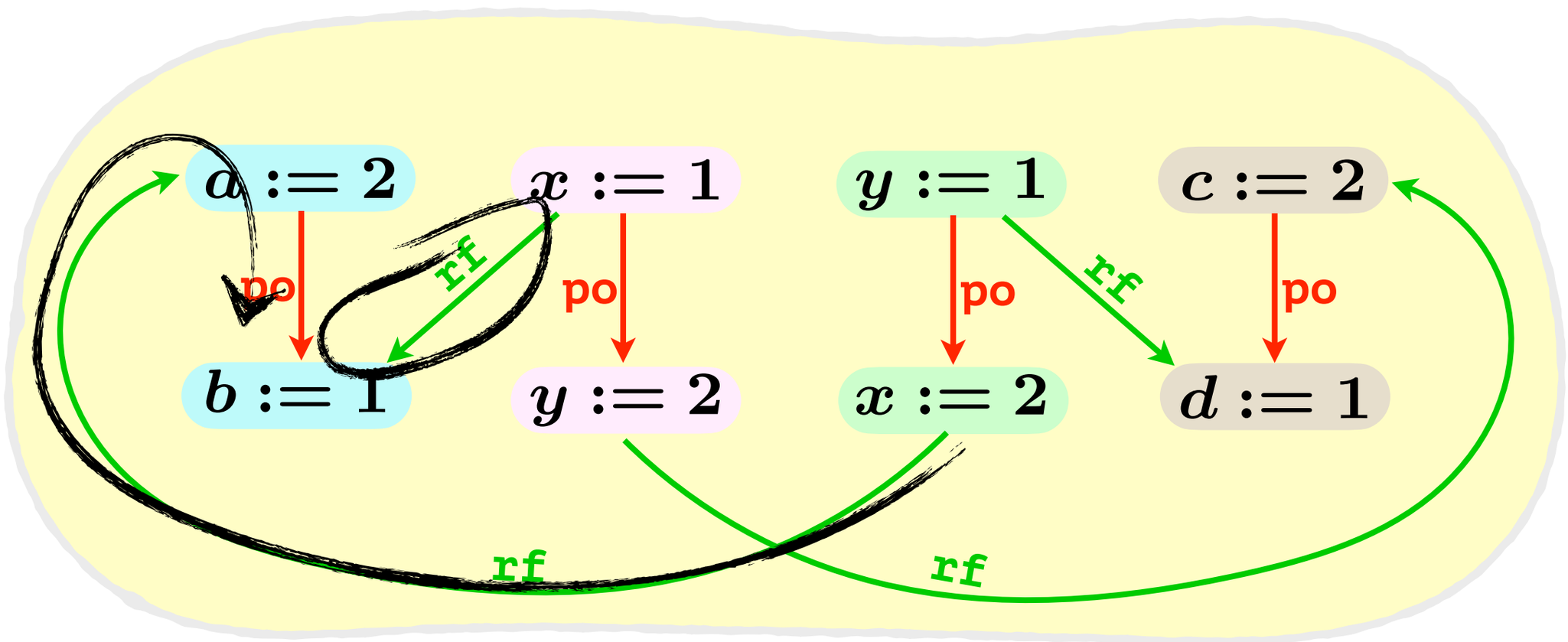


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

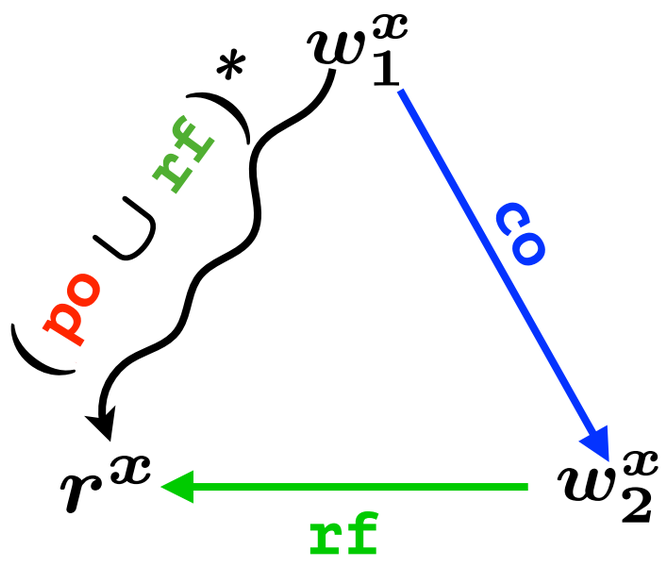


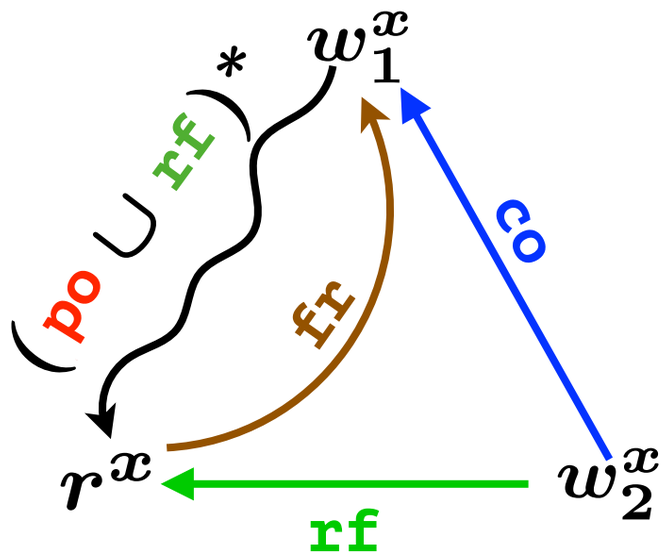


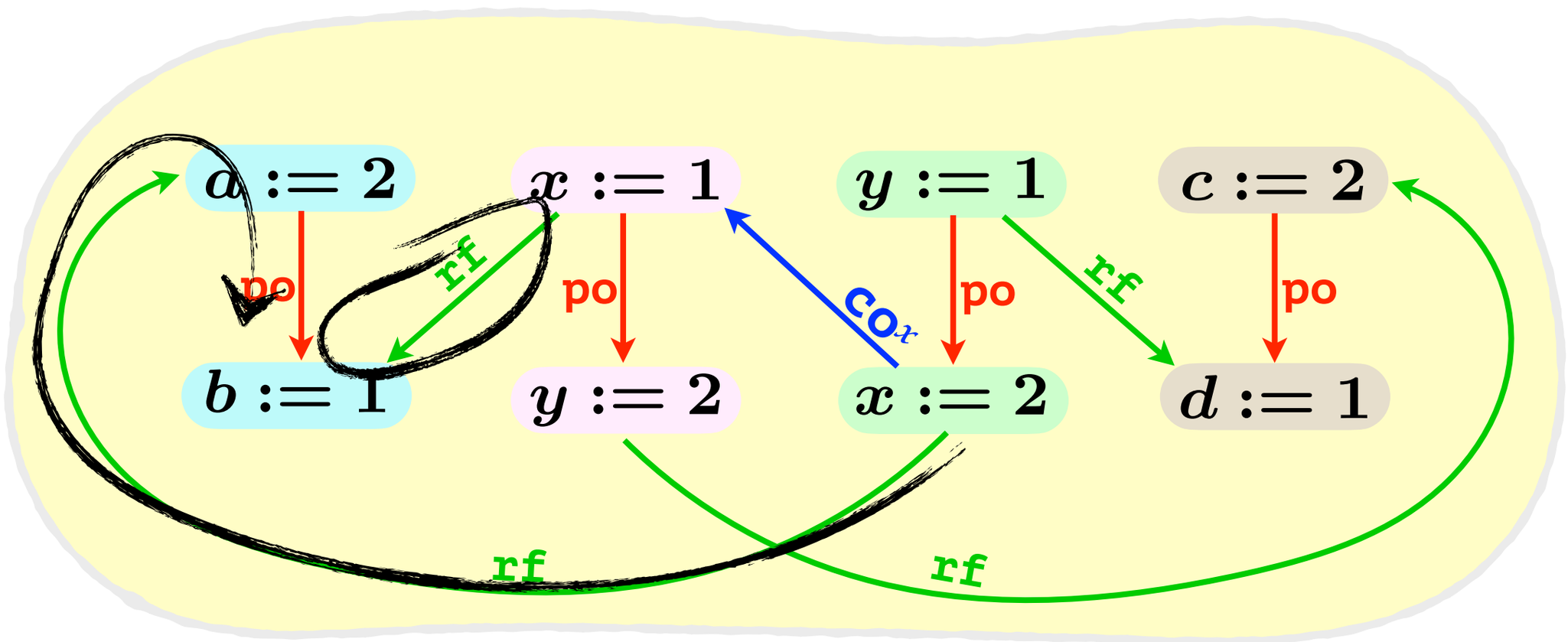


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

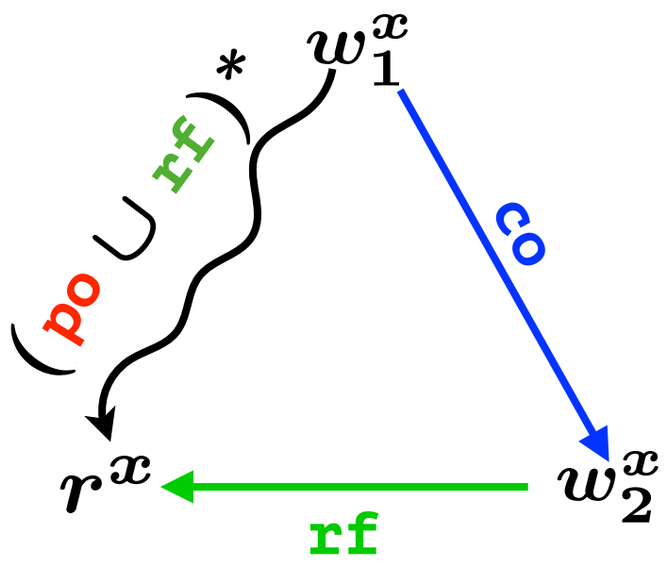


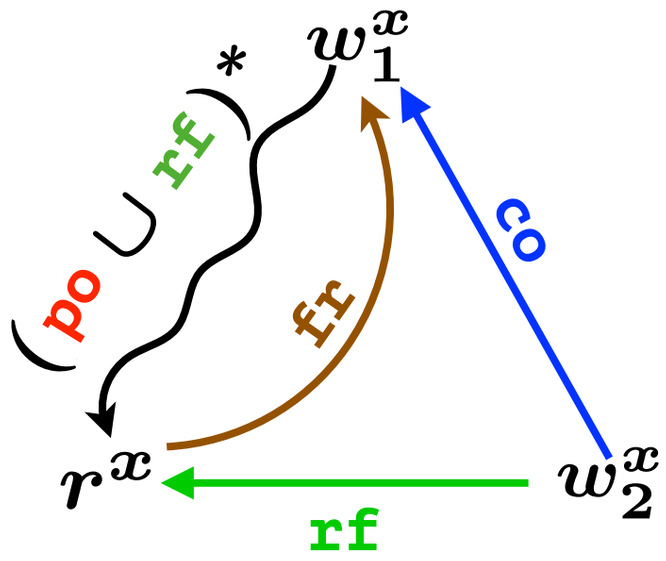


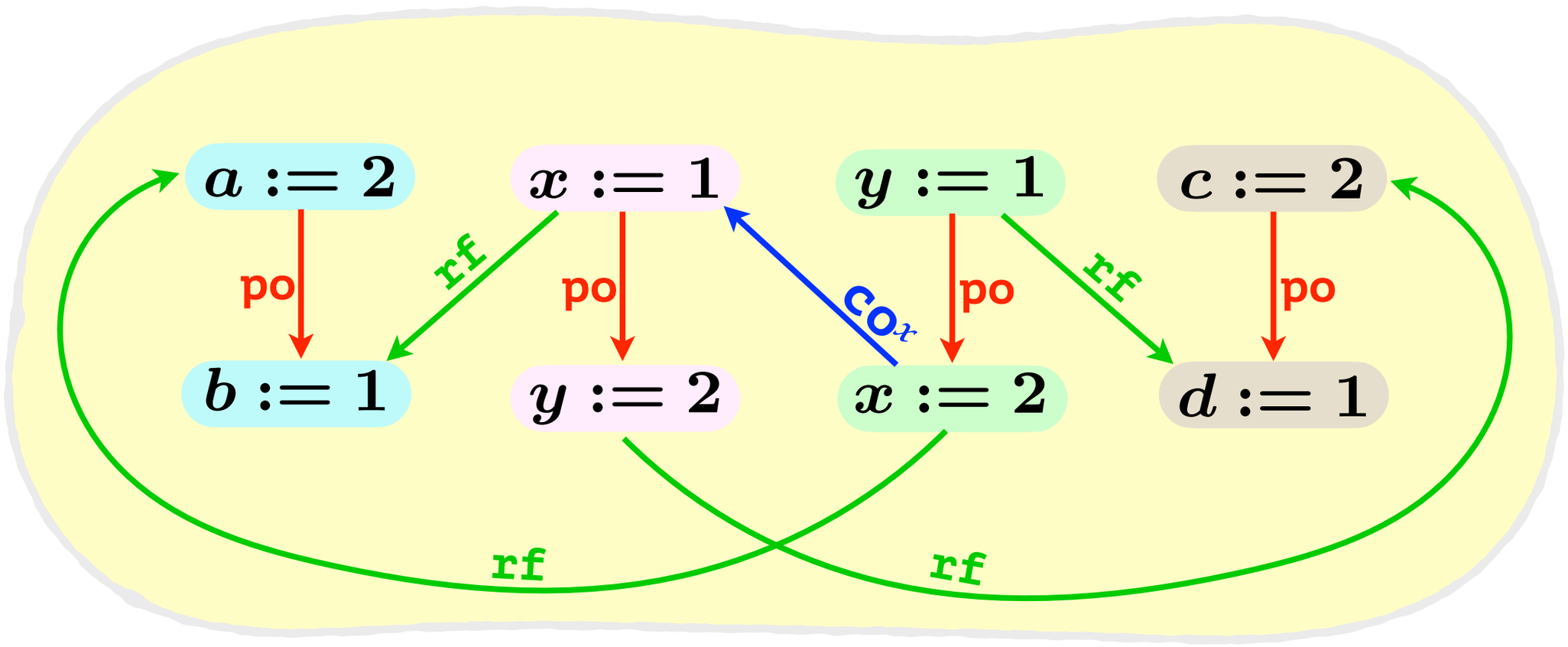


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

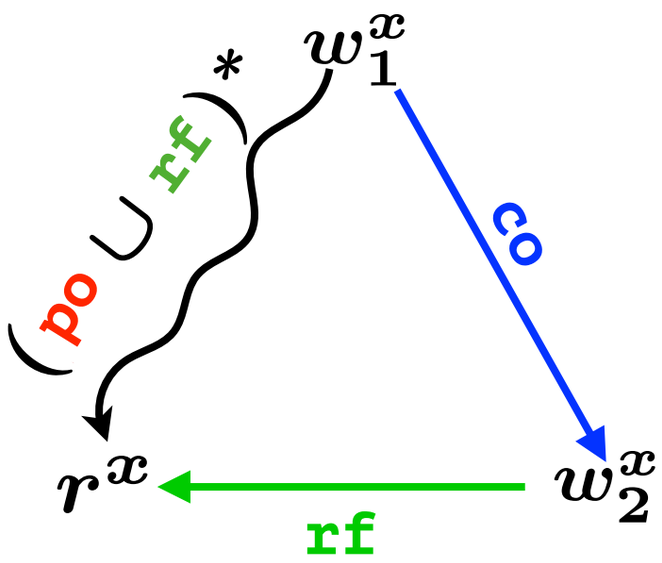


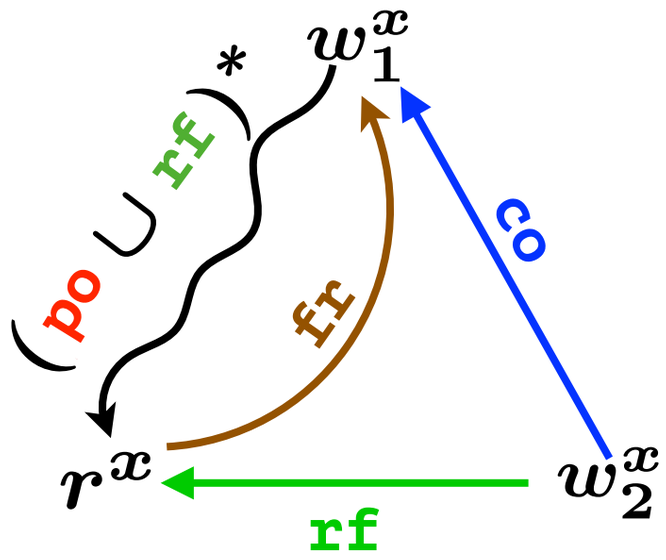


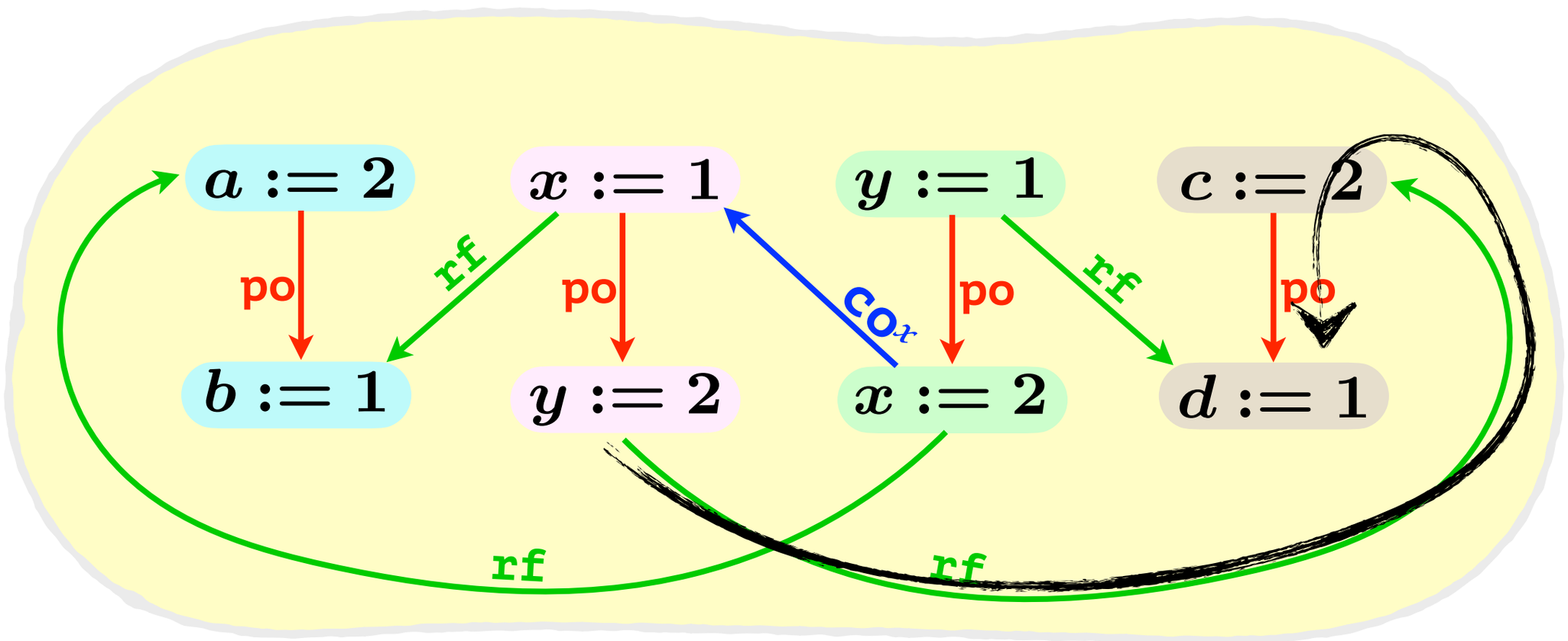


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

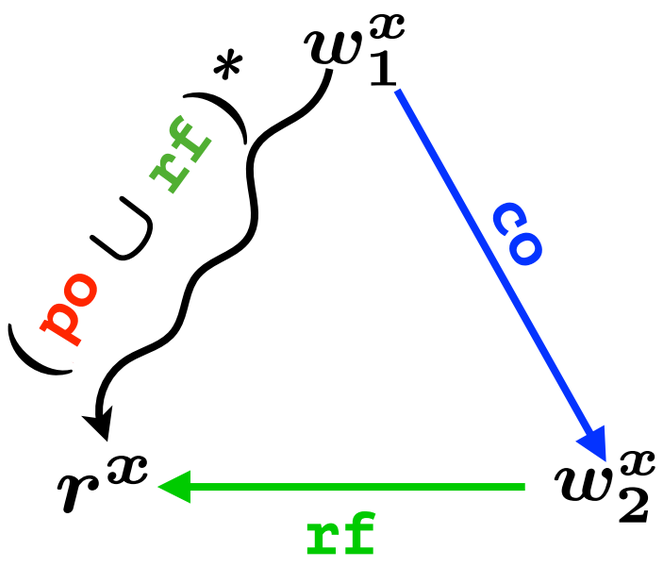


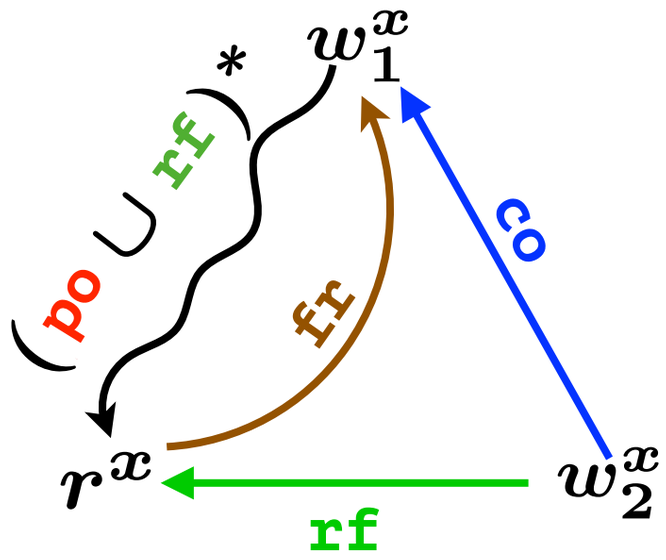


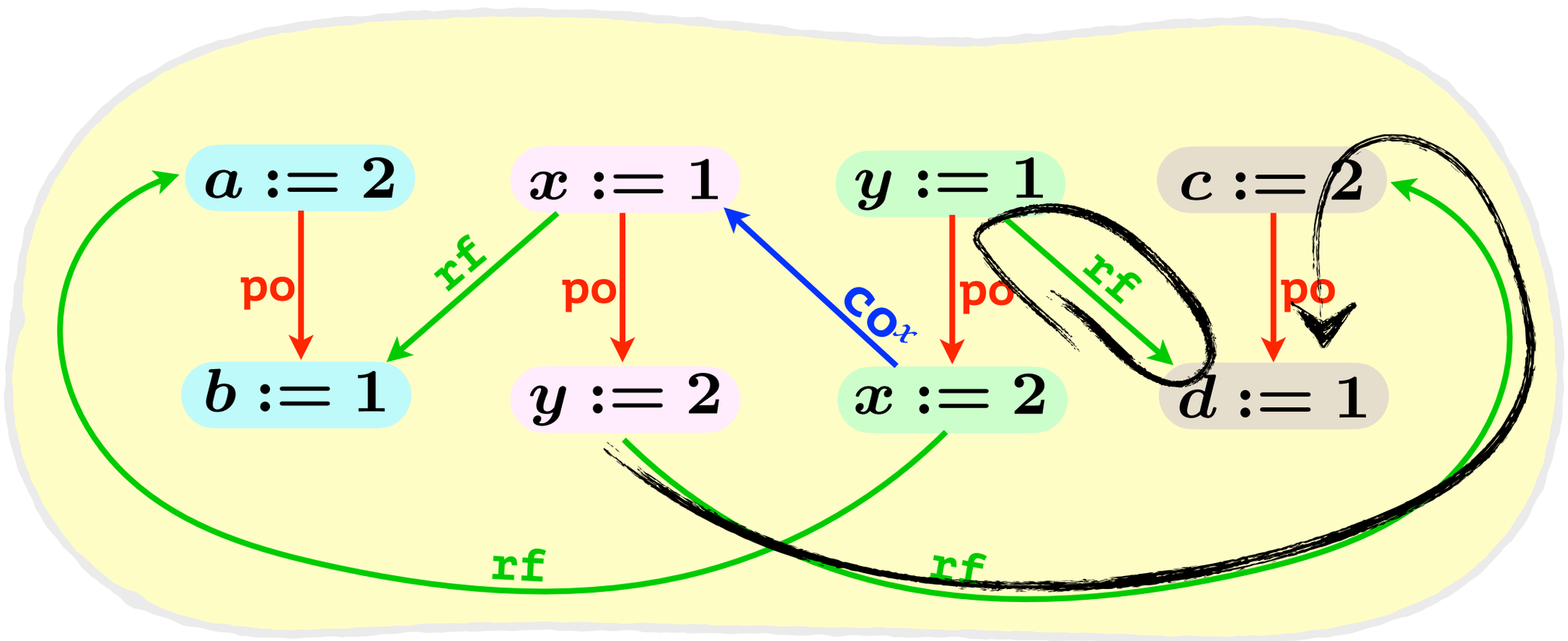


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

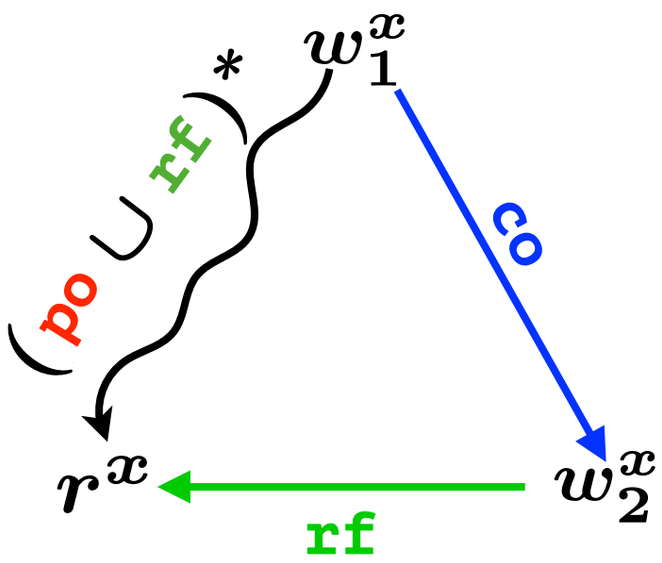


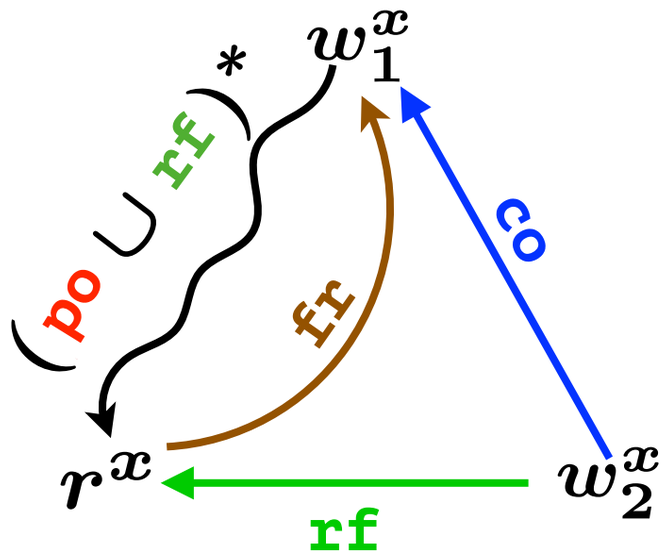


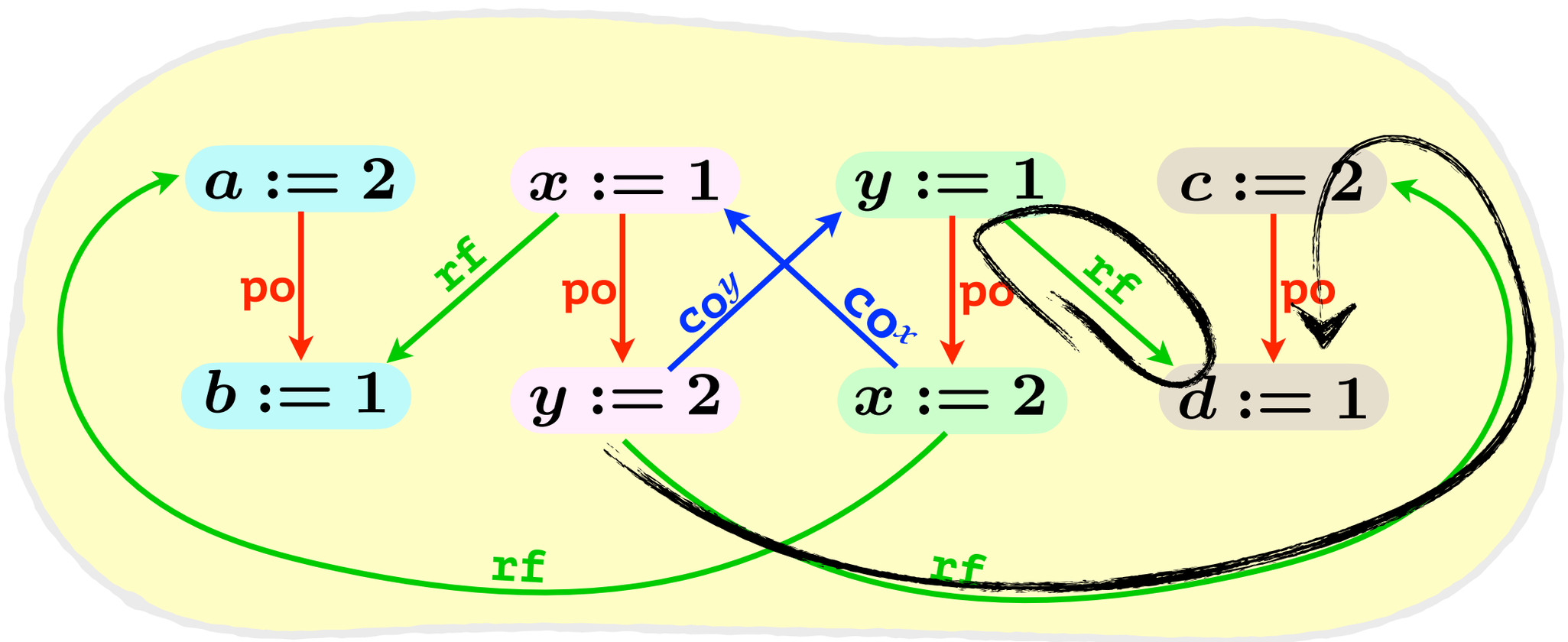


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

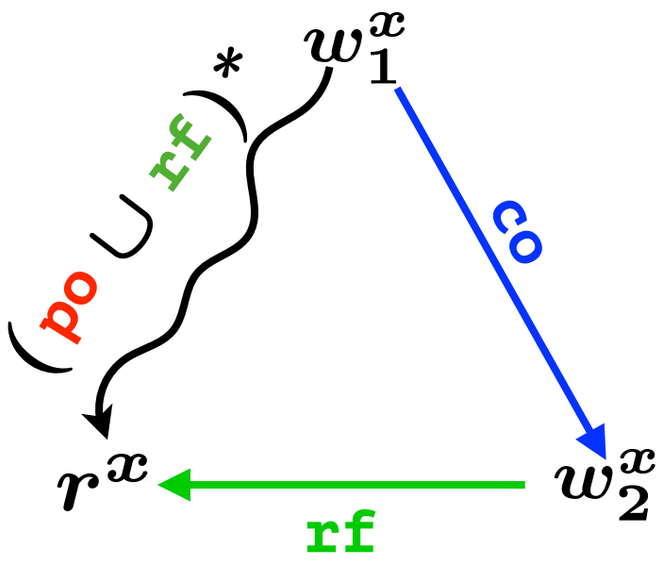


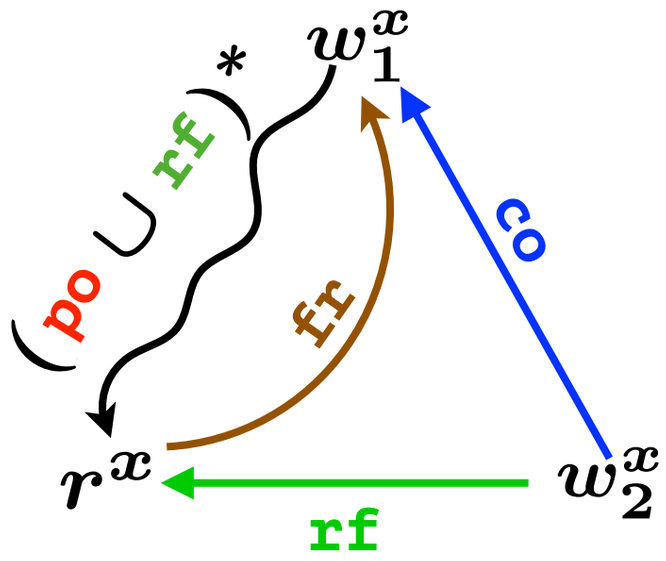


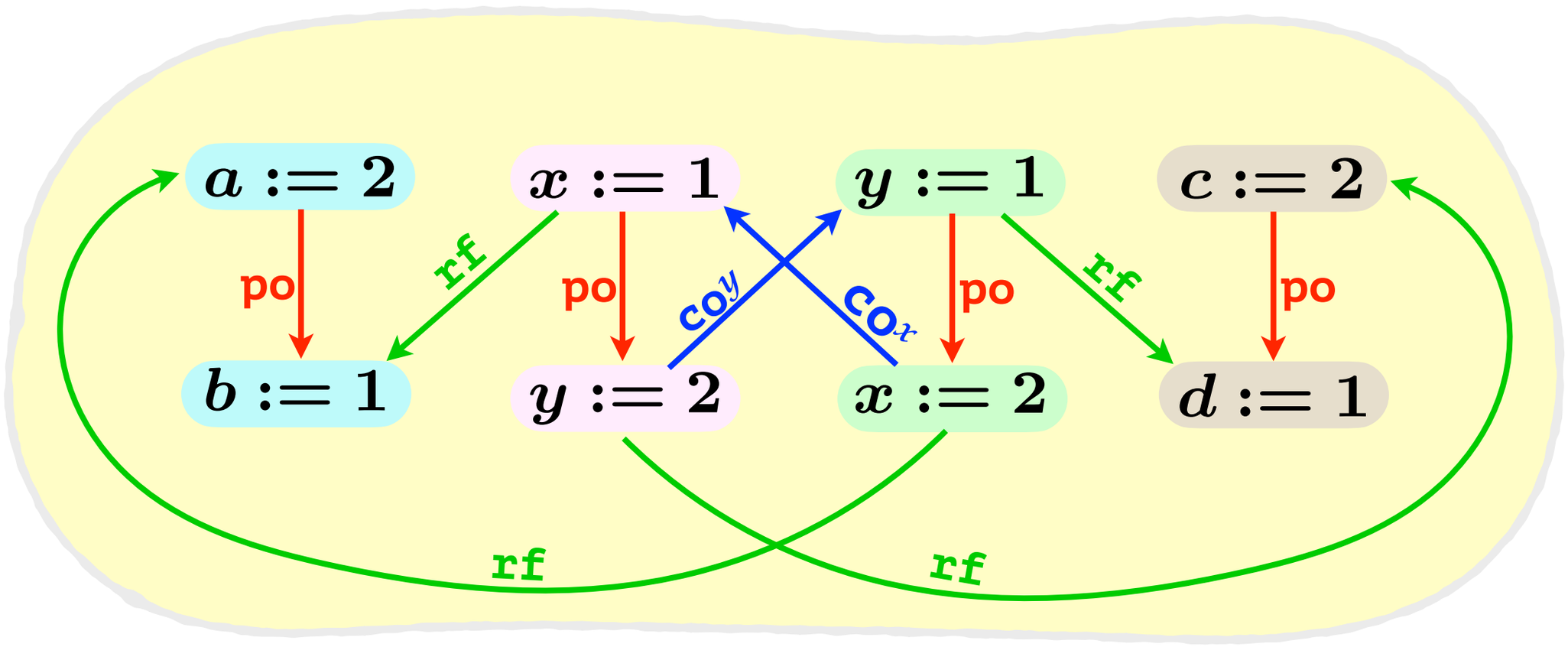


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

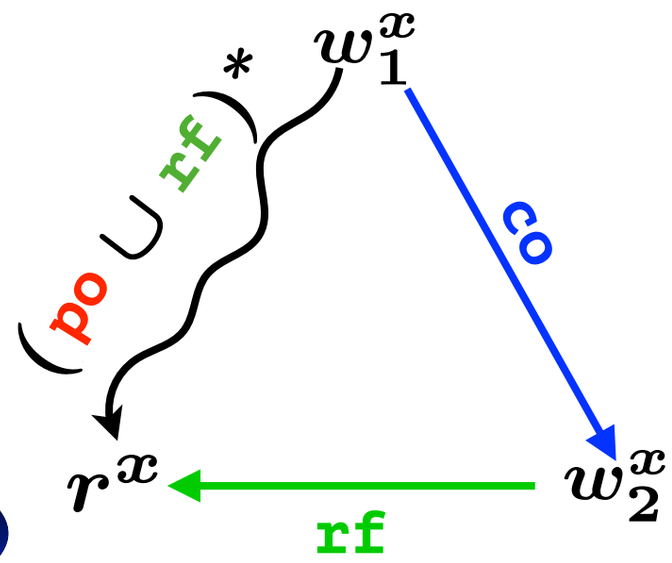


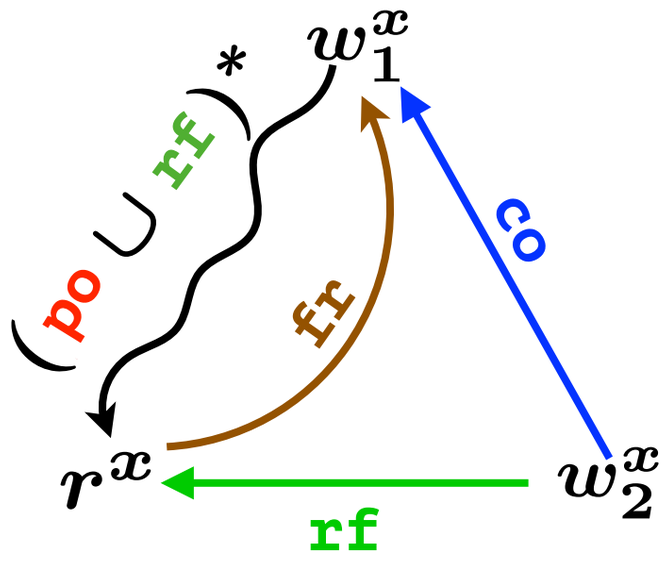




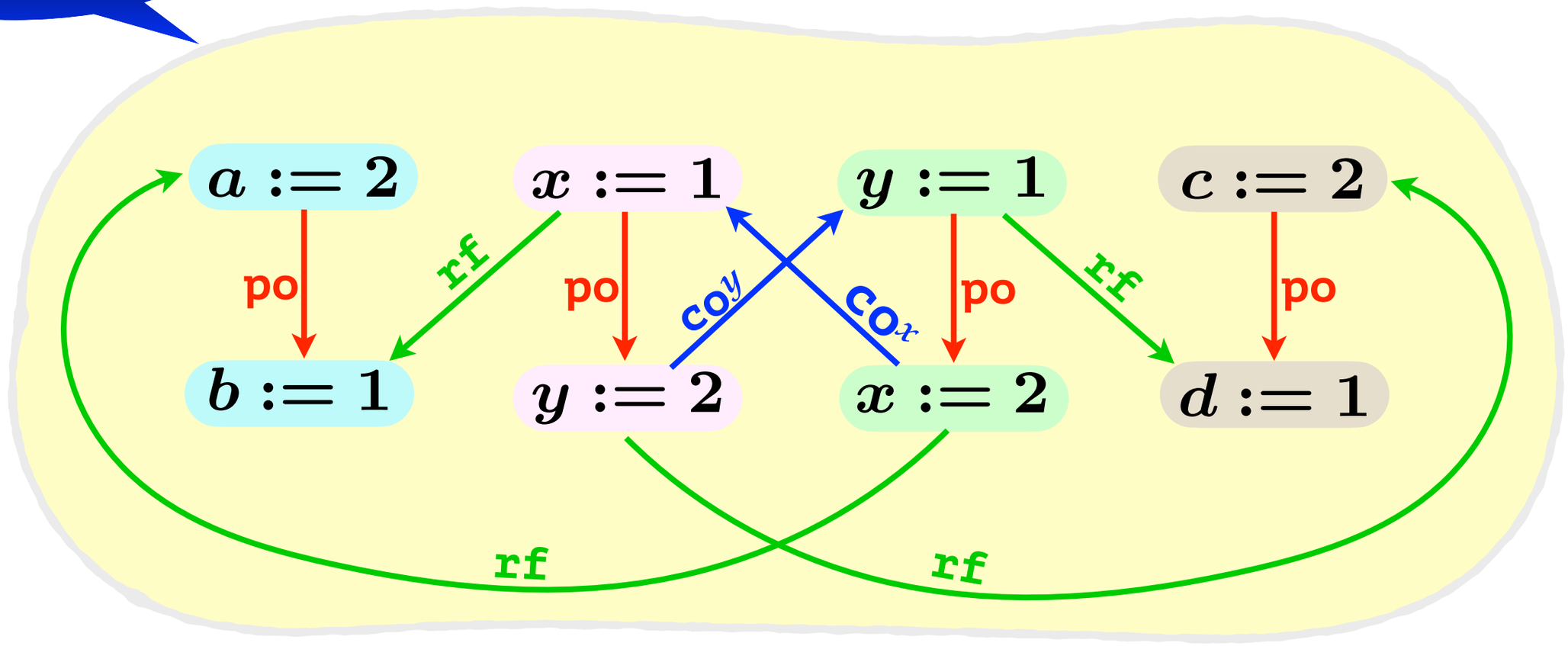
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation



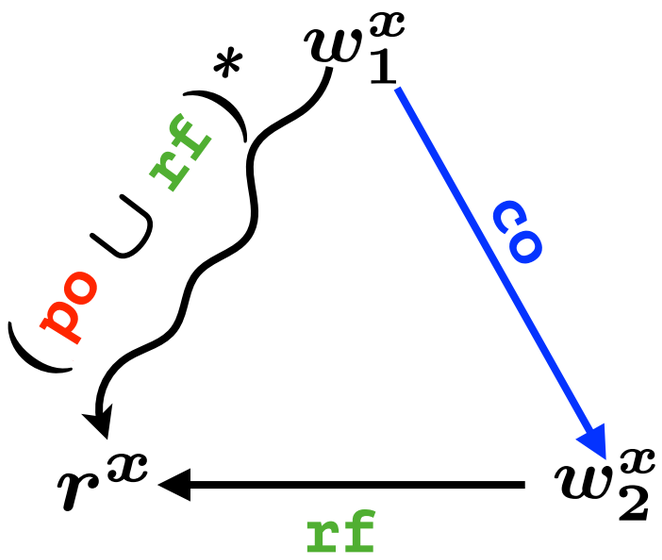


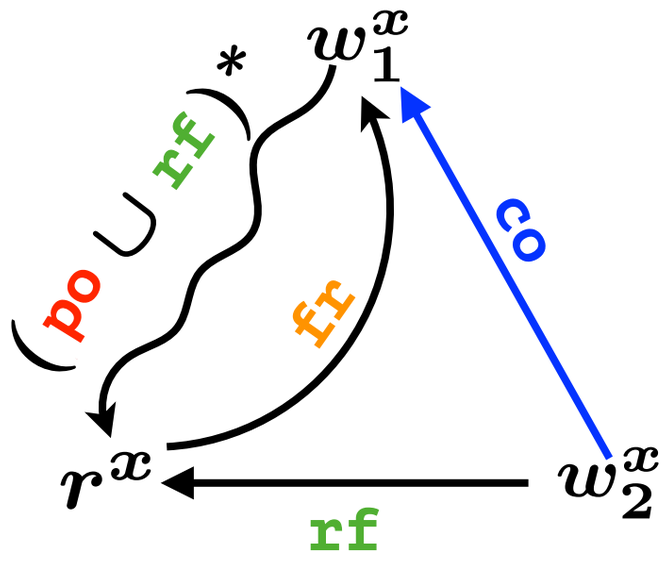
saturated

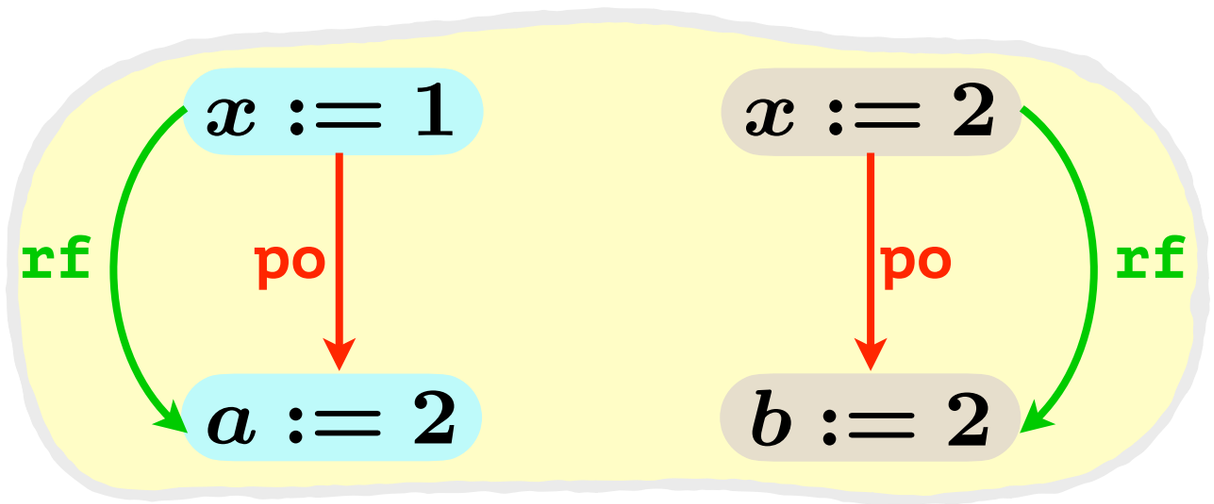


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Saturation

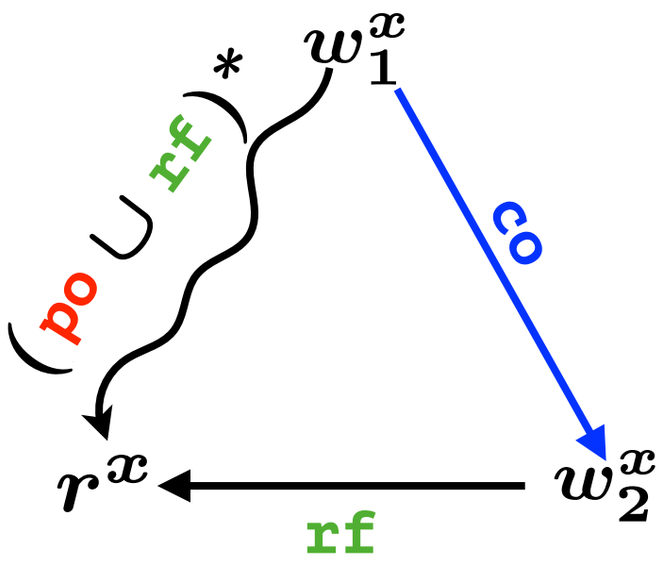


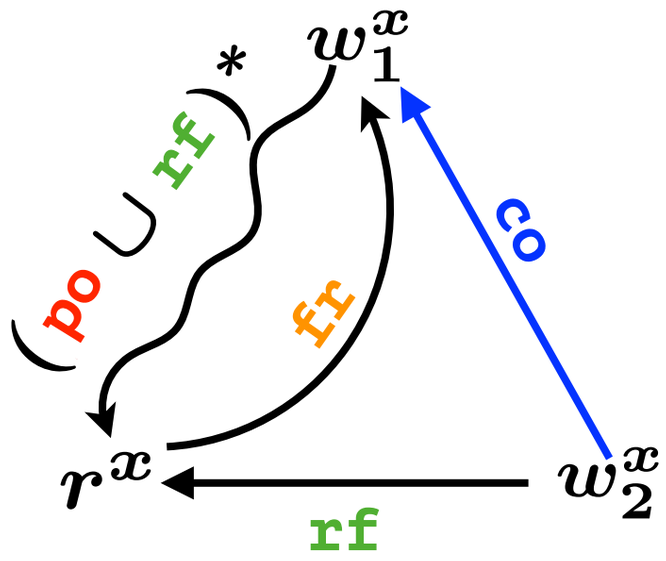




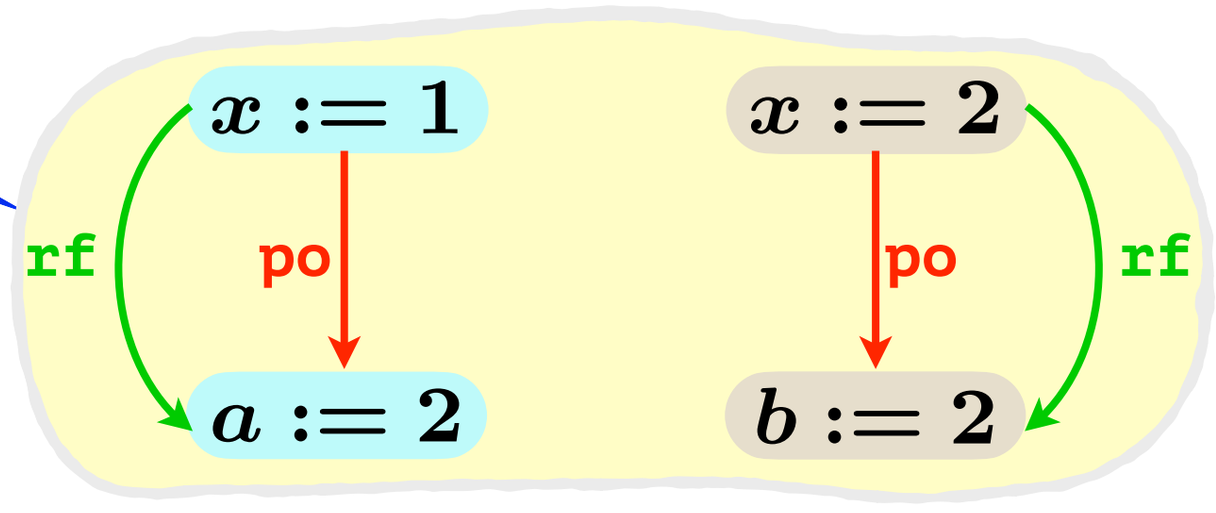
RA: $\forall x.\ acyclic(\texttt{po} \cup \texttt{rf} \cup \texttt{co}^x \cup \texttt{fr}^x)$

Saturation



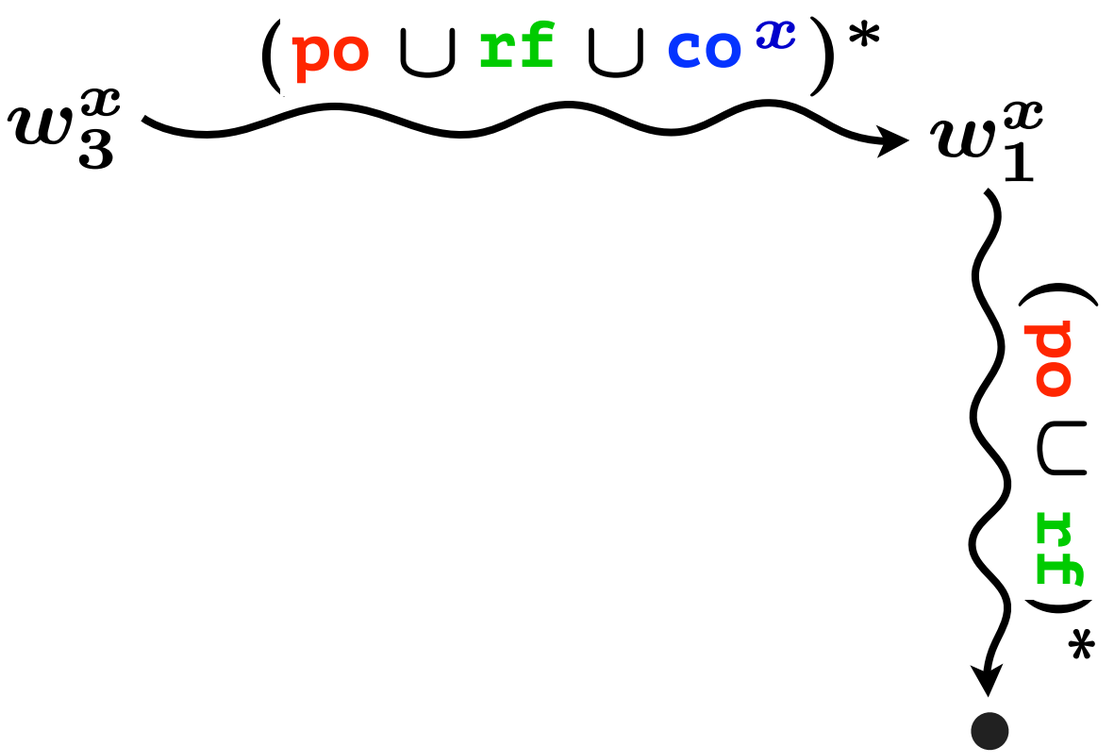


saturated

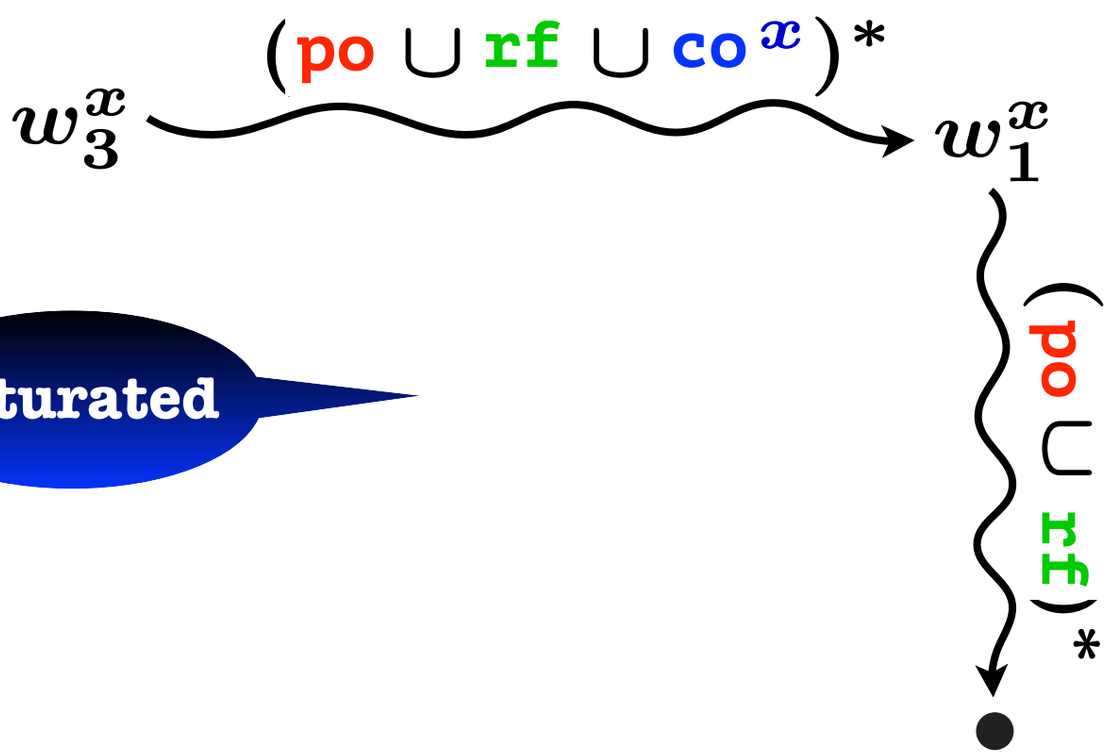


RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

Adding Events to Traces

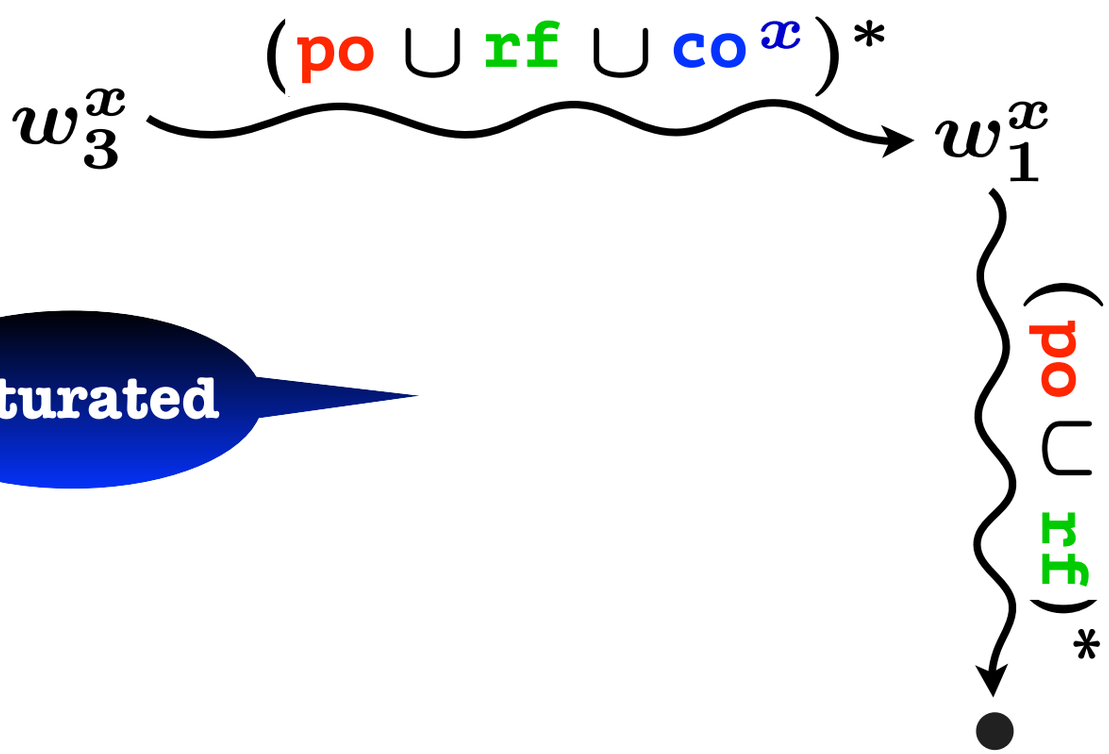


Adding Events to Traces

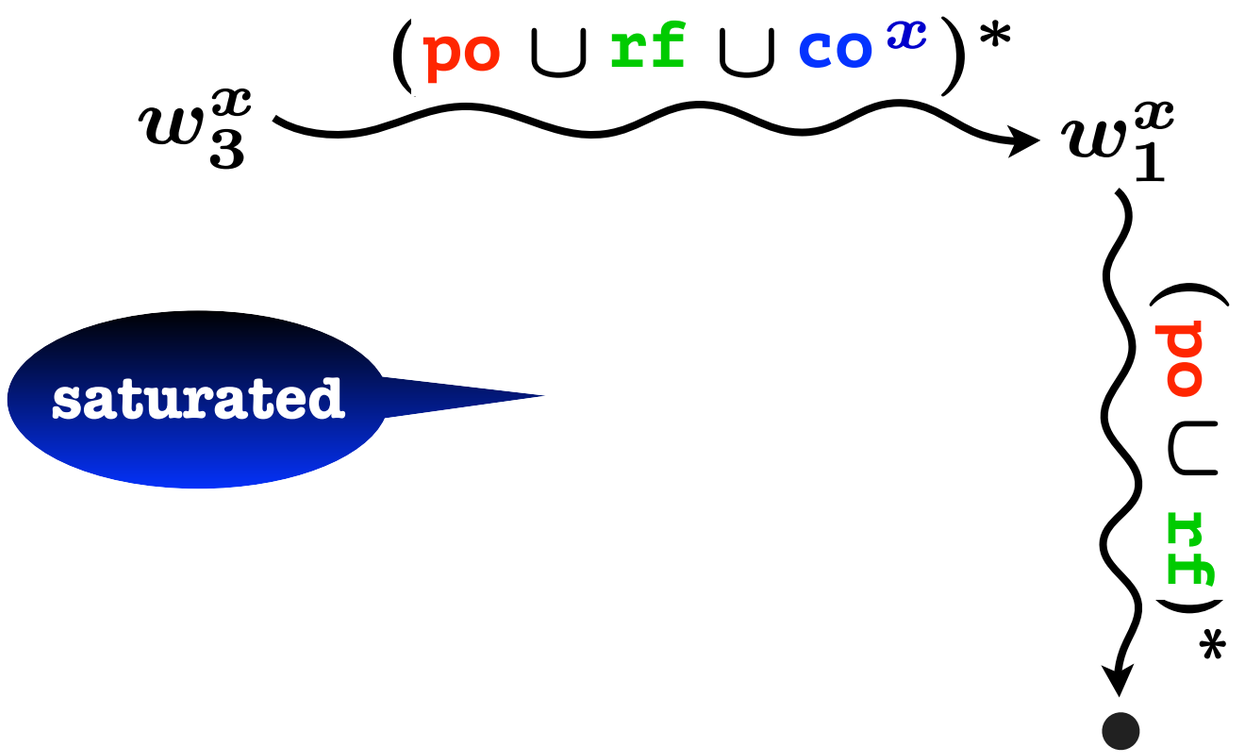


Adding Events to Traces

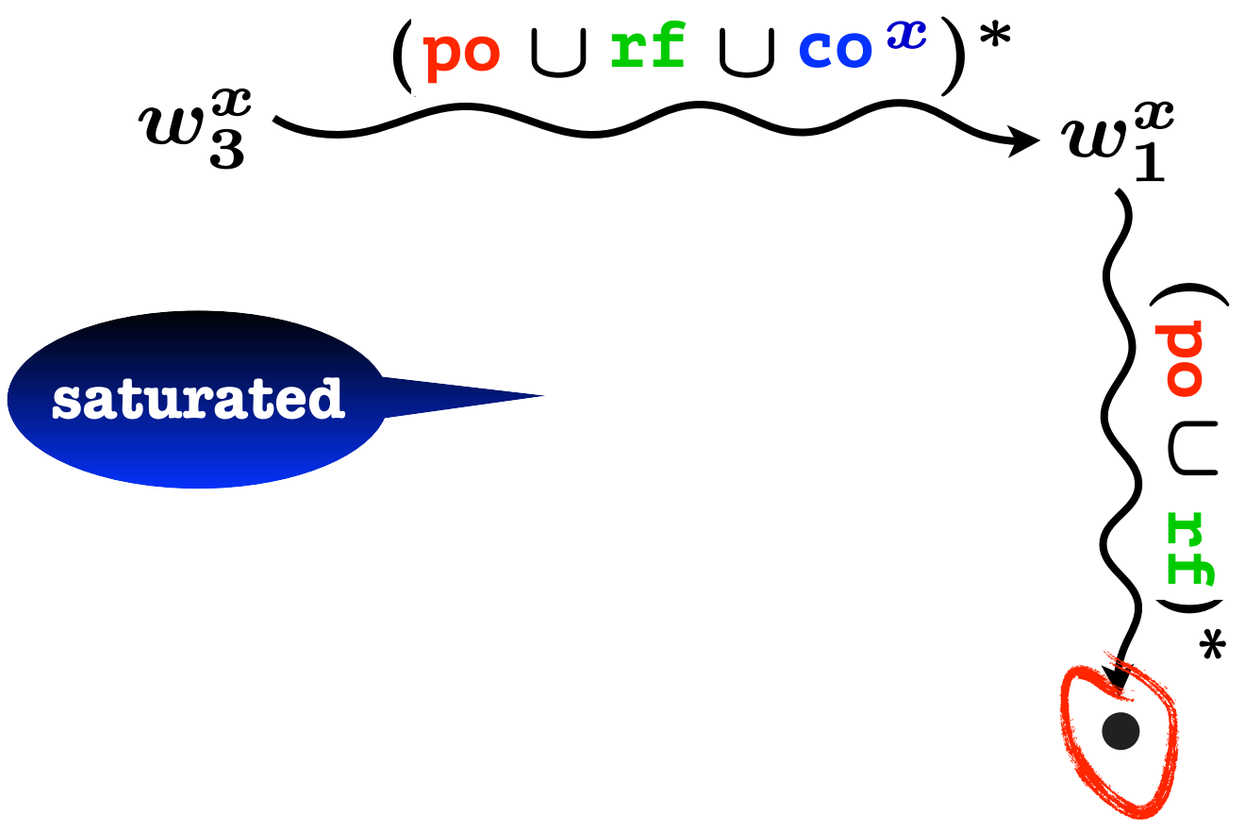
$\tau \oplus e$



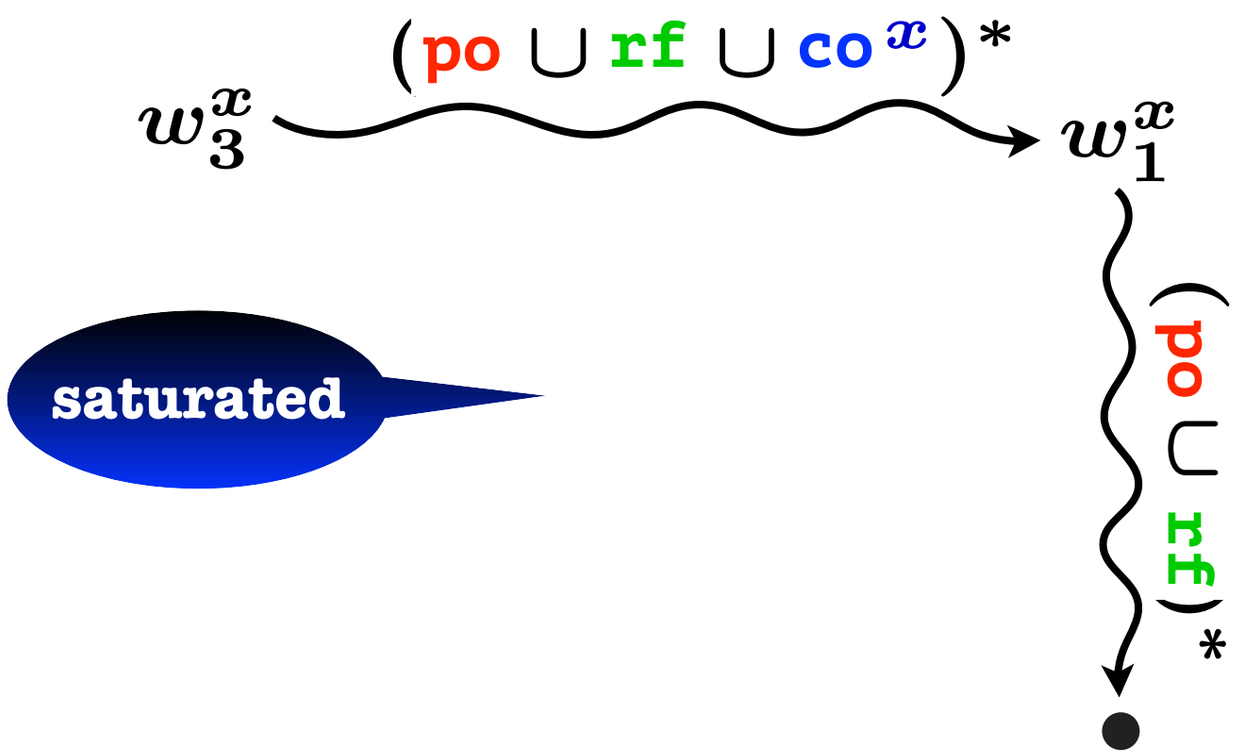
Adding Events to Traces



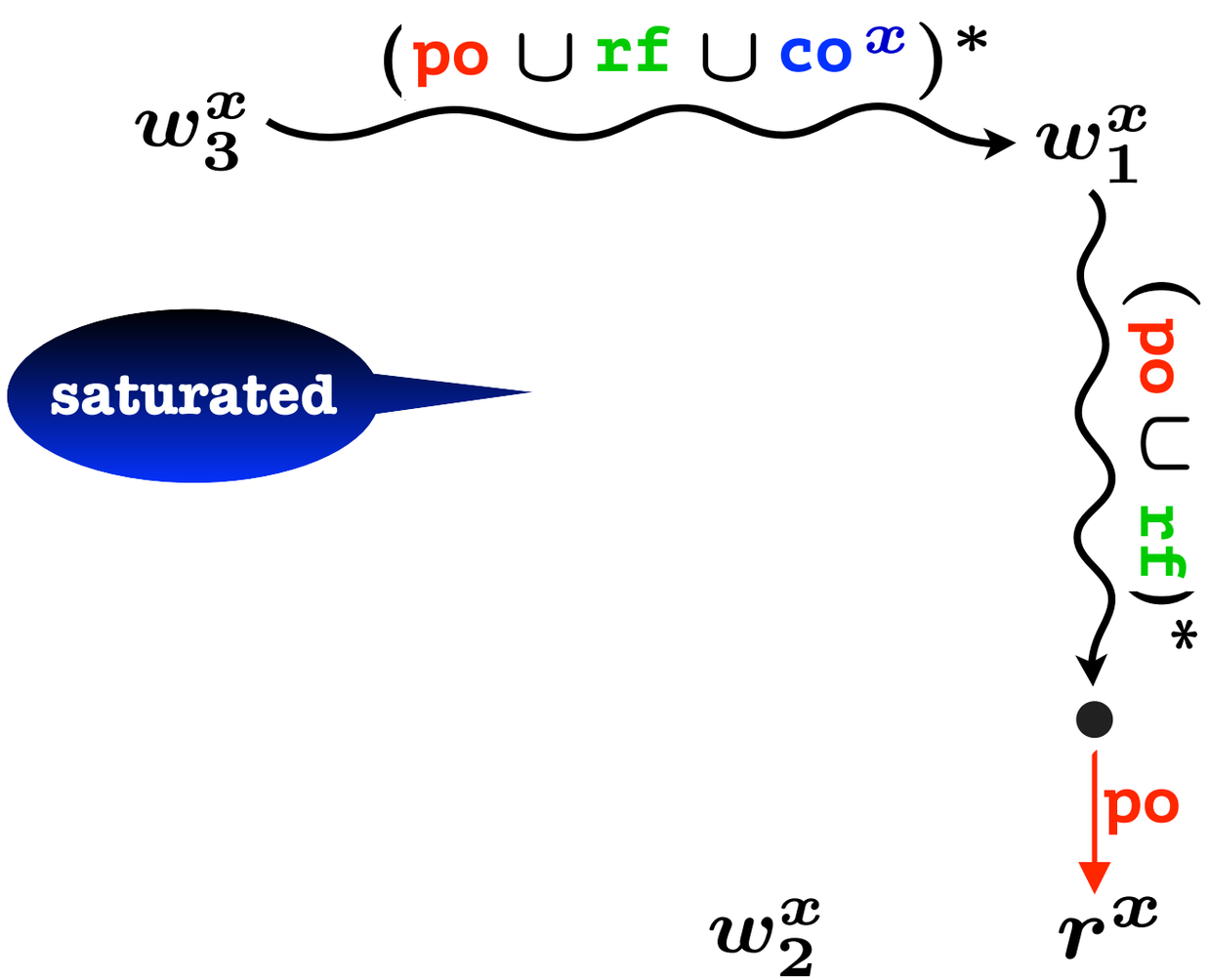
Adding Events to Traces



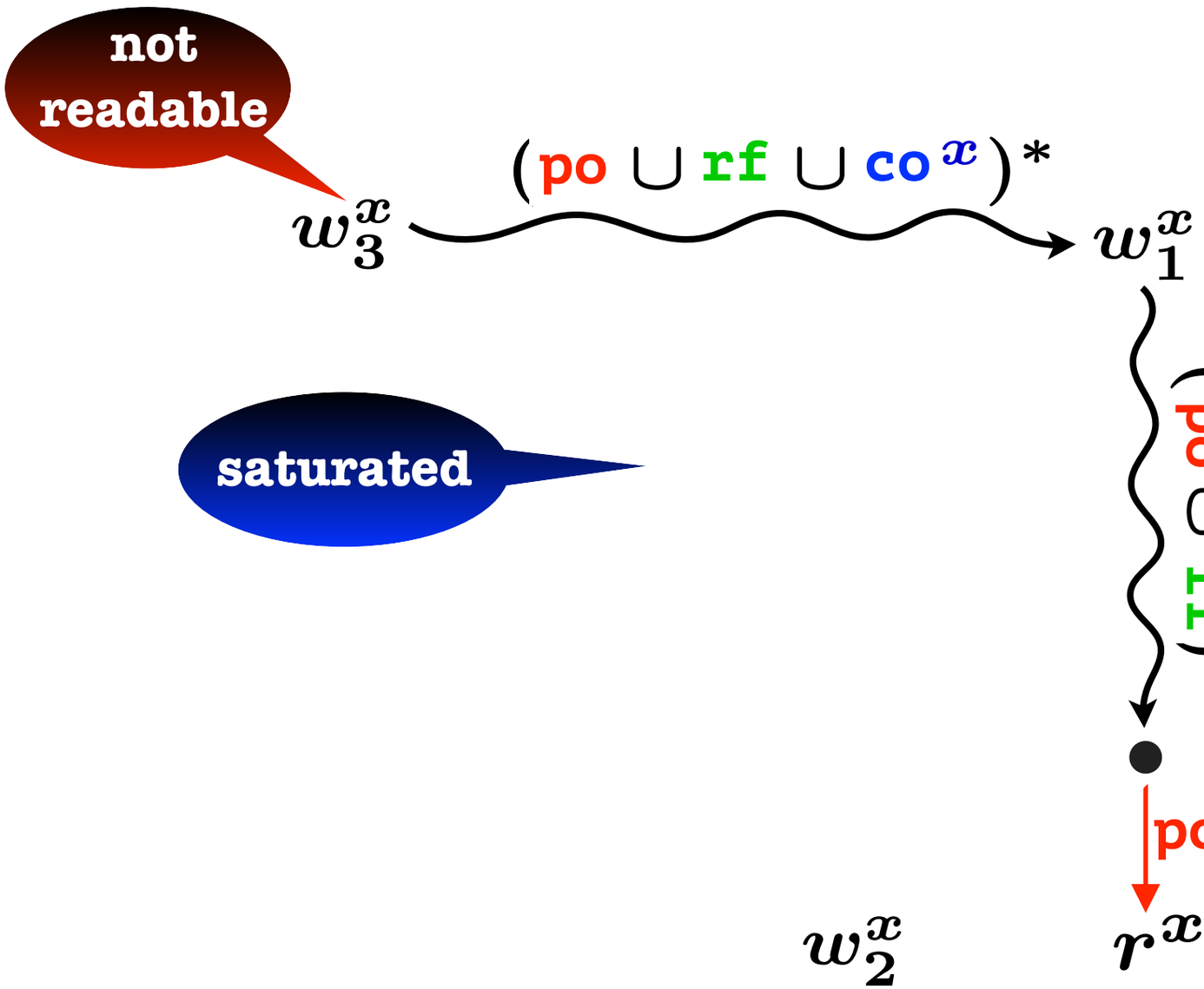
Adding Events to Traces



Adding Events to Traces

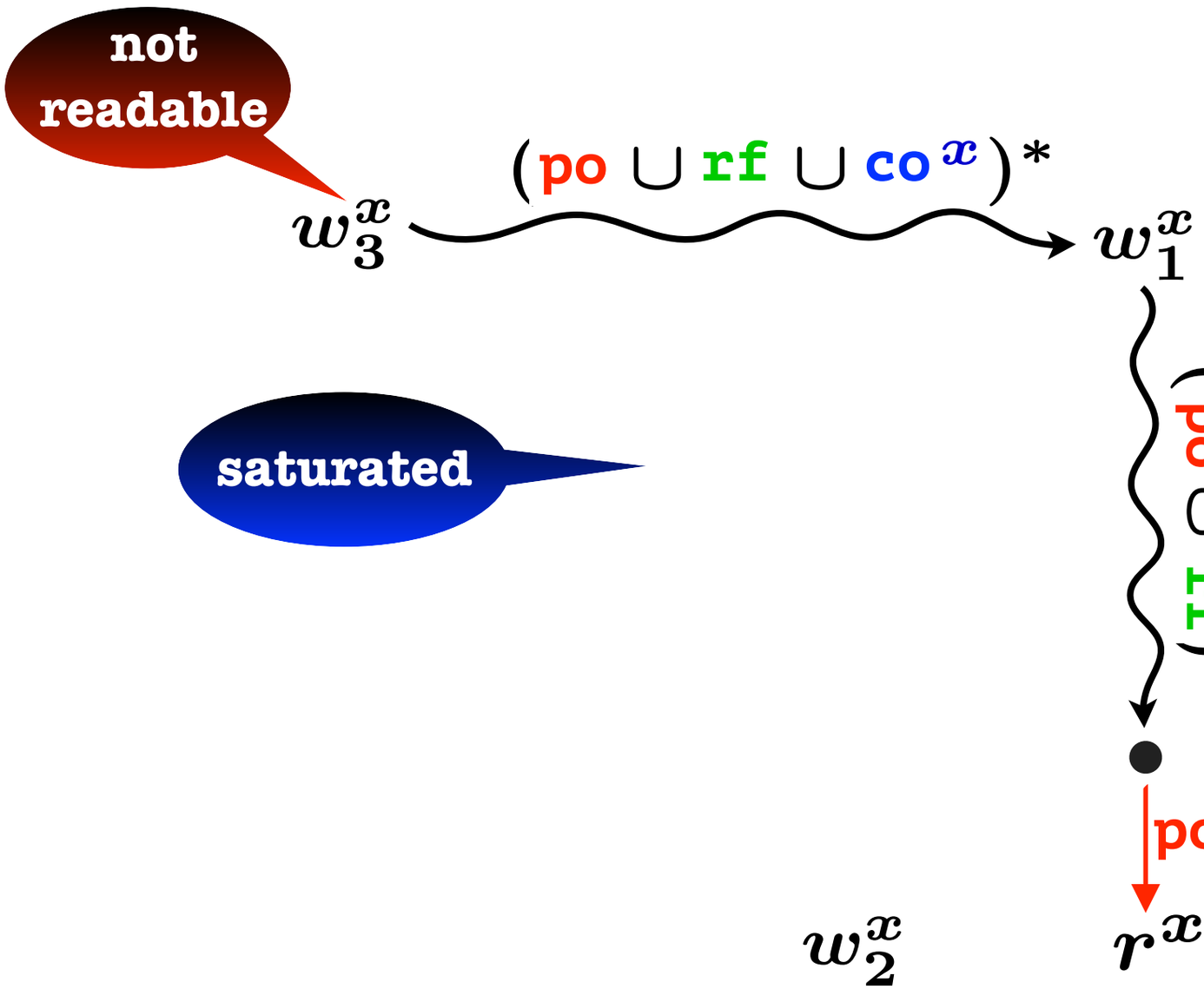


Adding Events to Traces



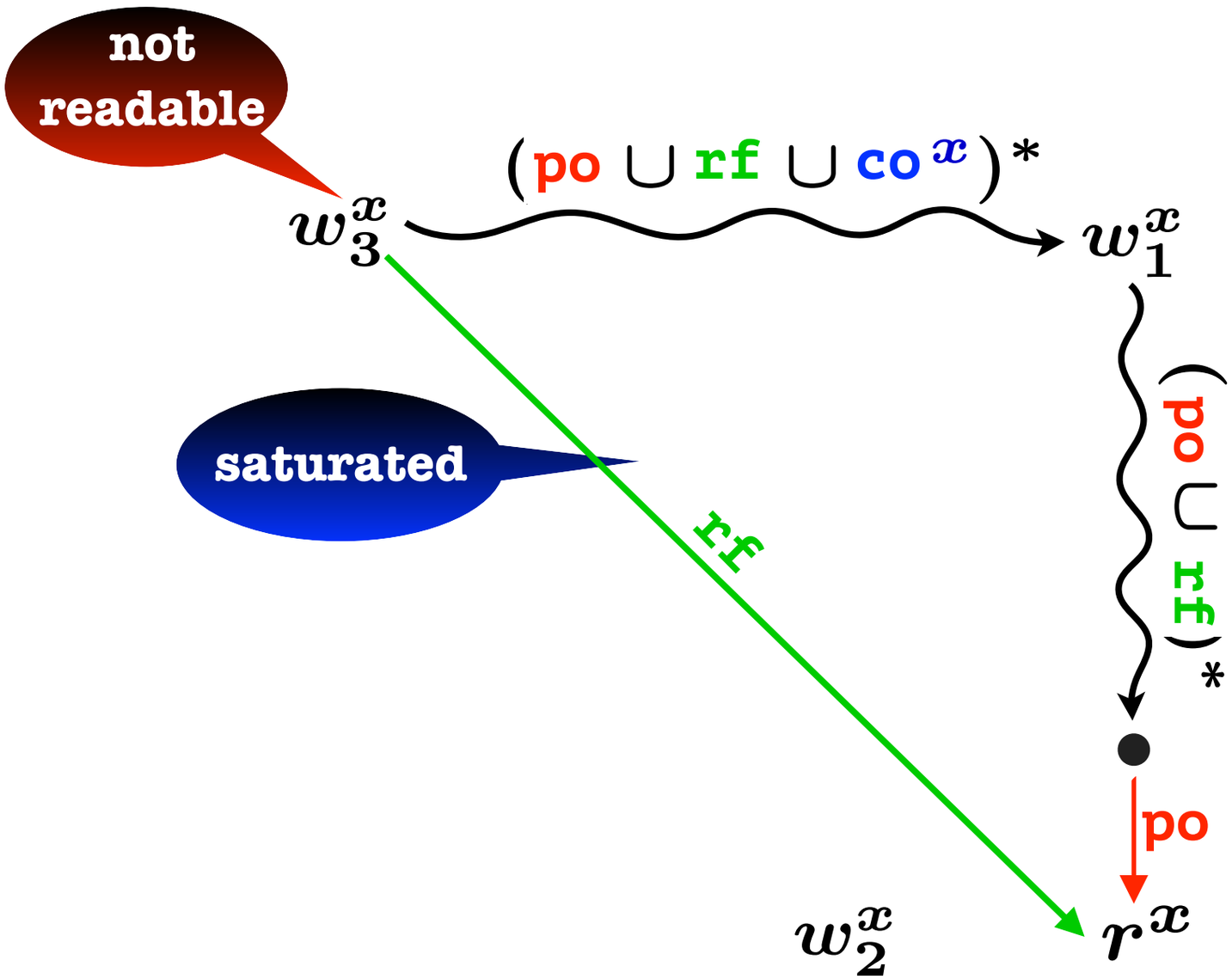
Adding Events to Traces

RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$



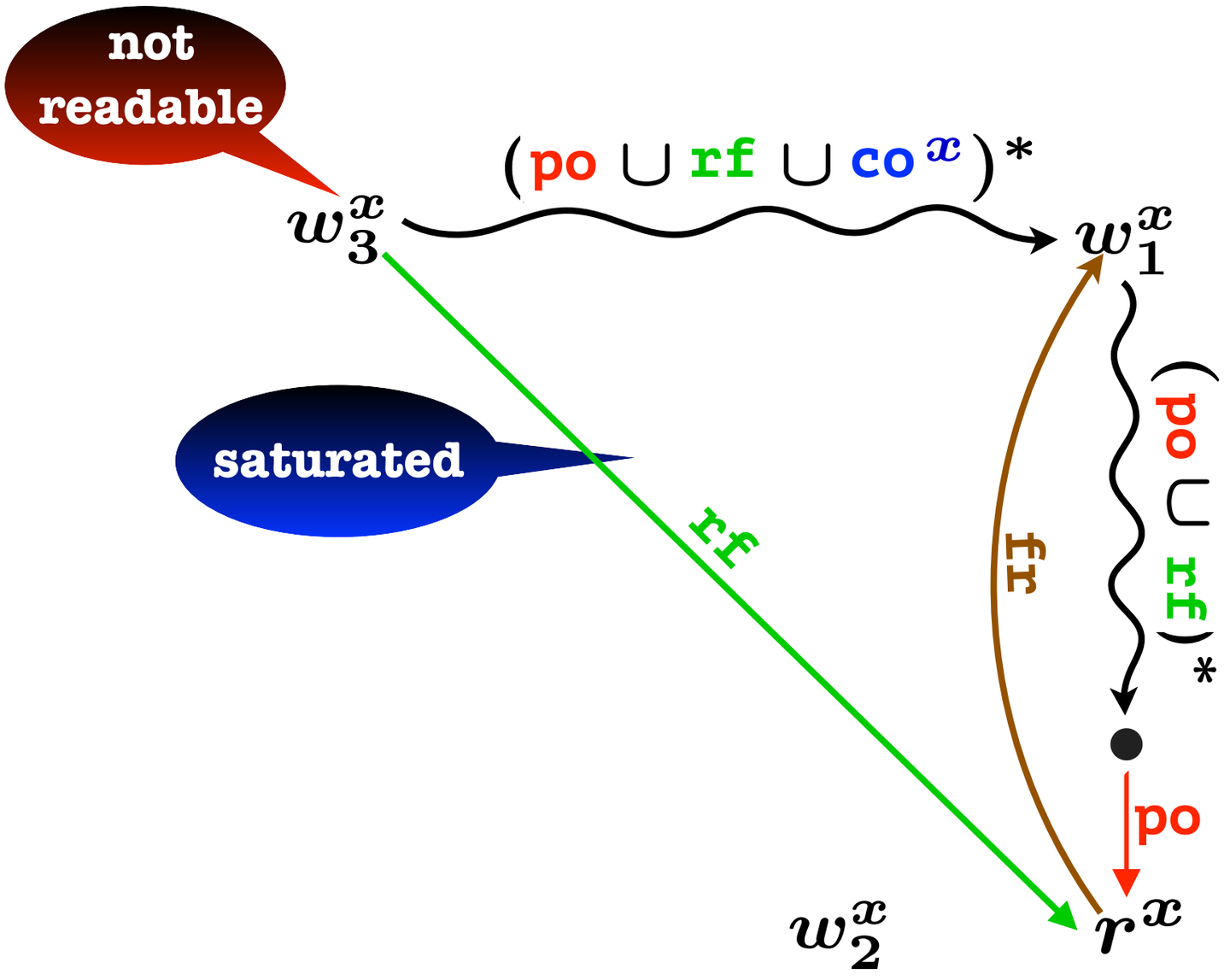
Adding Events to Traces

RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$

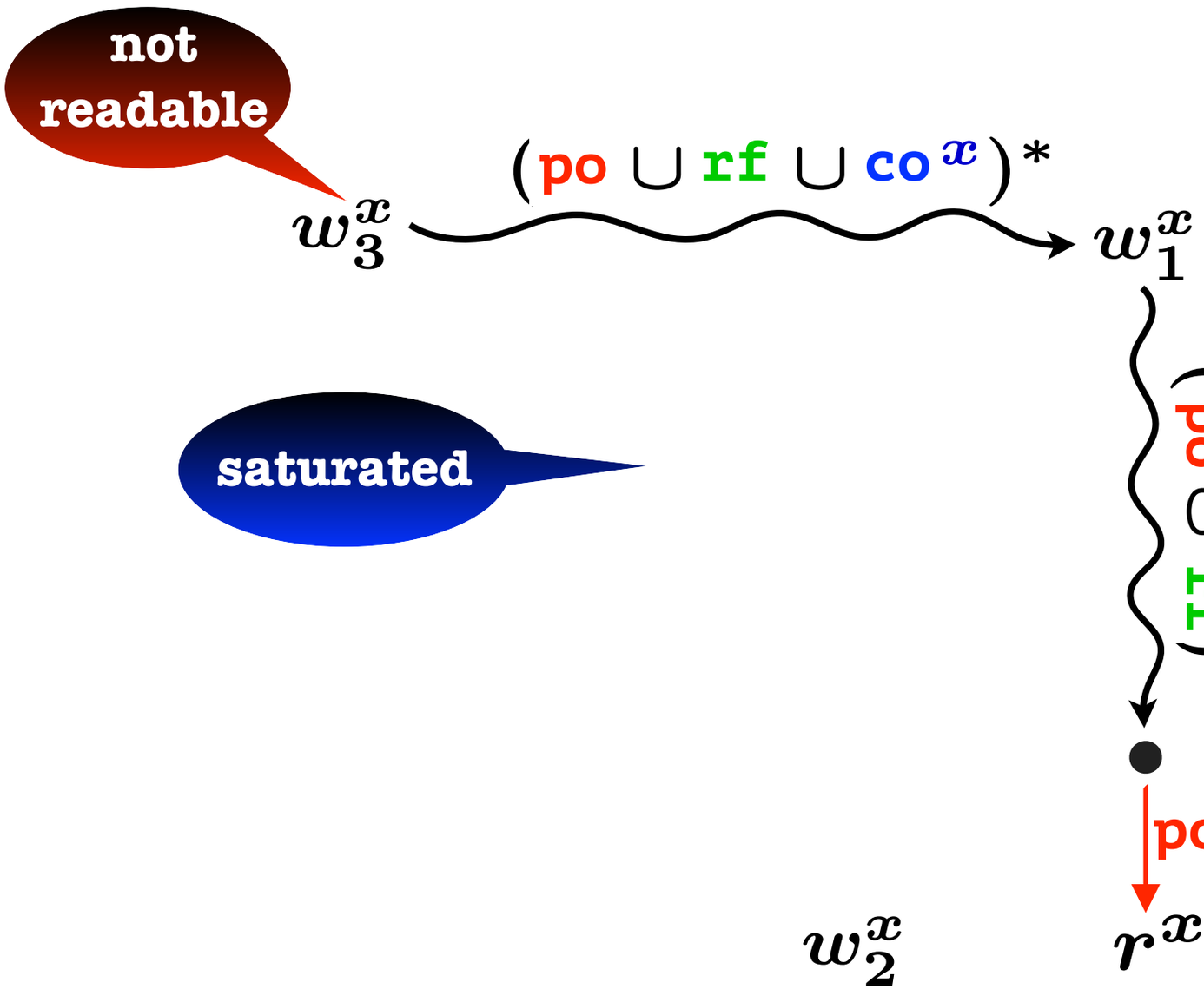


Adding Events to Traces

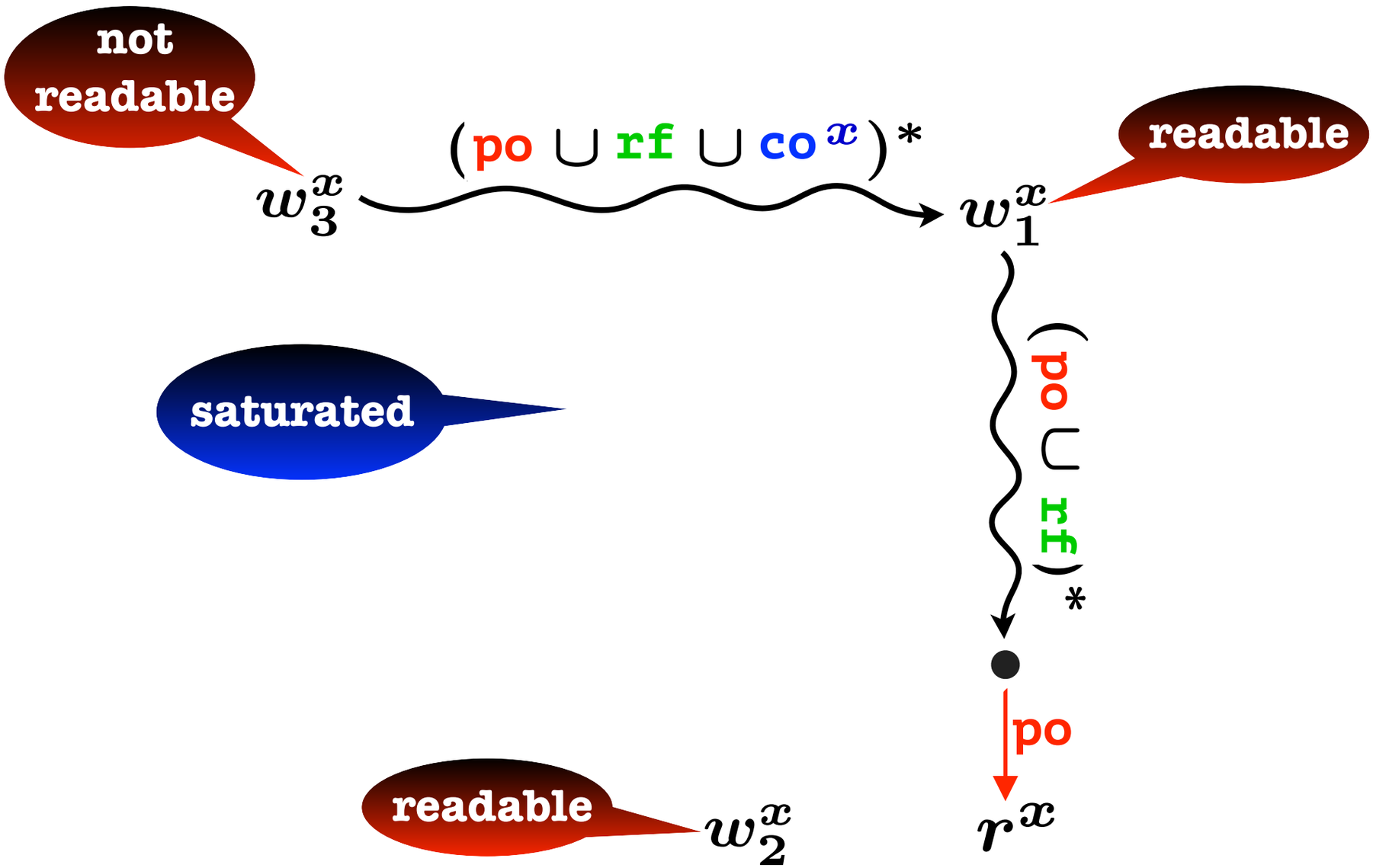
RA: $\forall x.\ acyclic(\mathtt{po} \cup \mathtt{rf} \cup \mathtt{co}^x \cup \mathtt{fr}^x)$



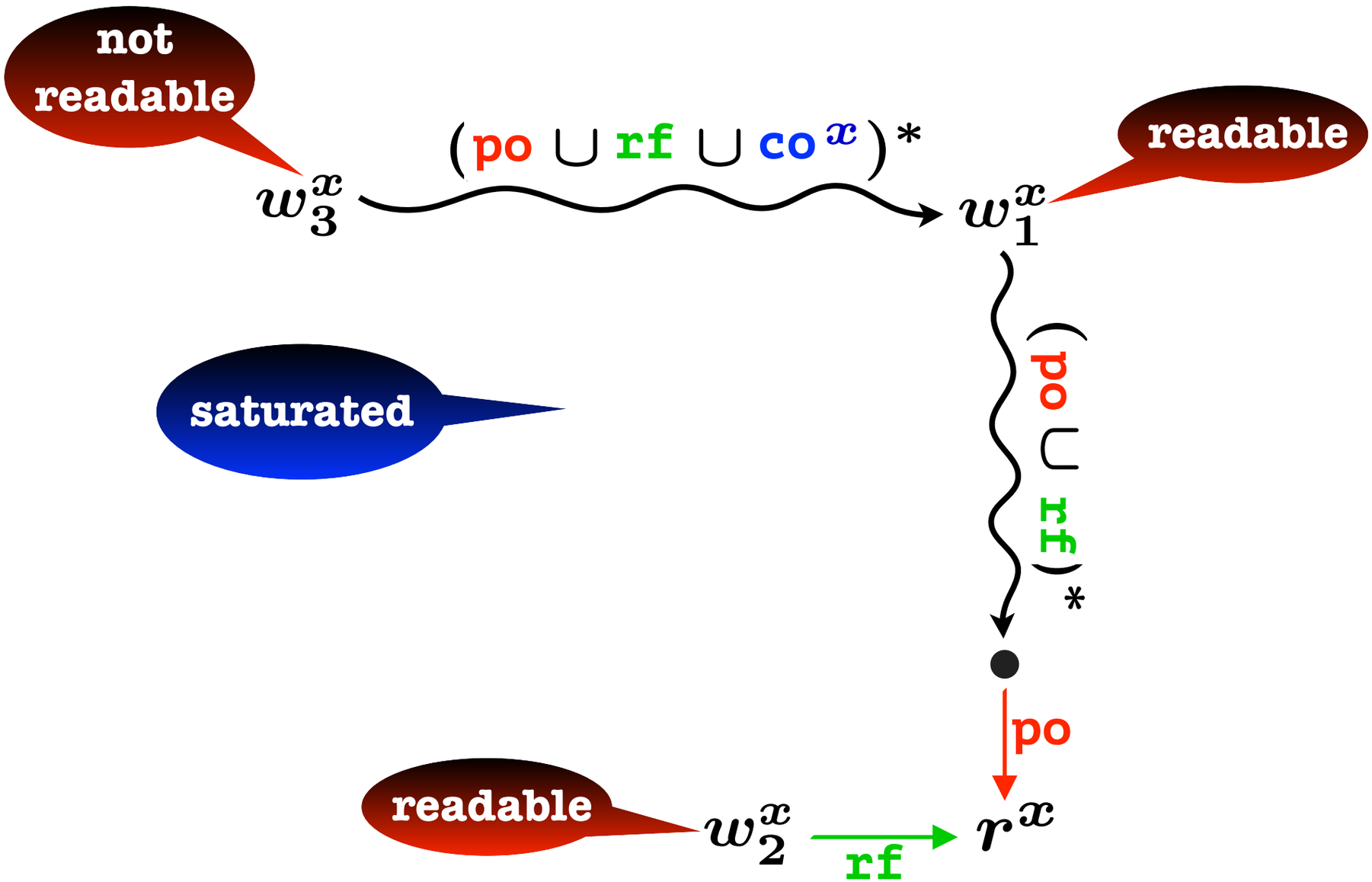
Adding Events to Traces



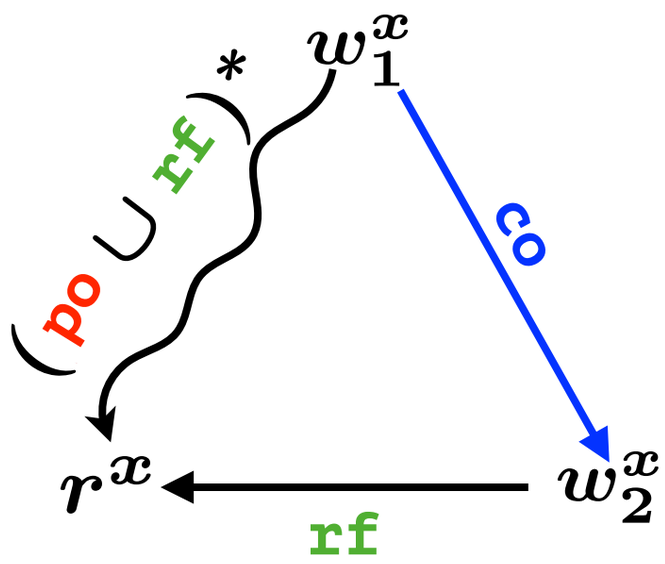
Adding Events to Traces

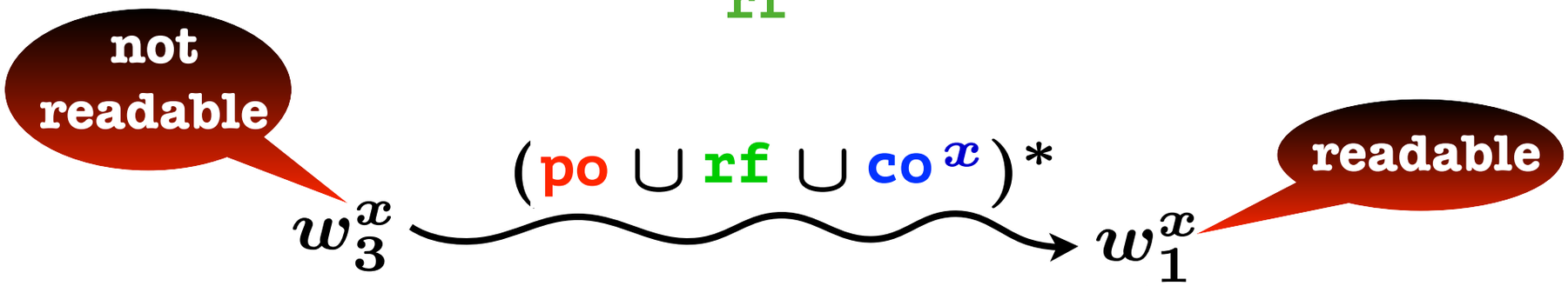


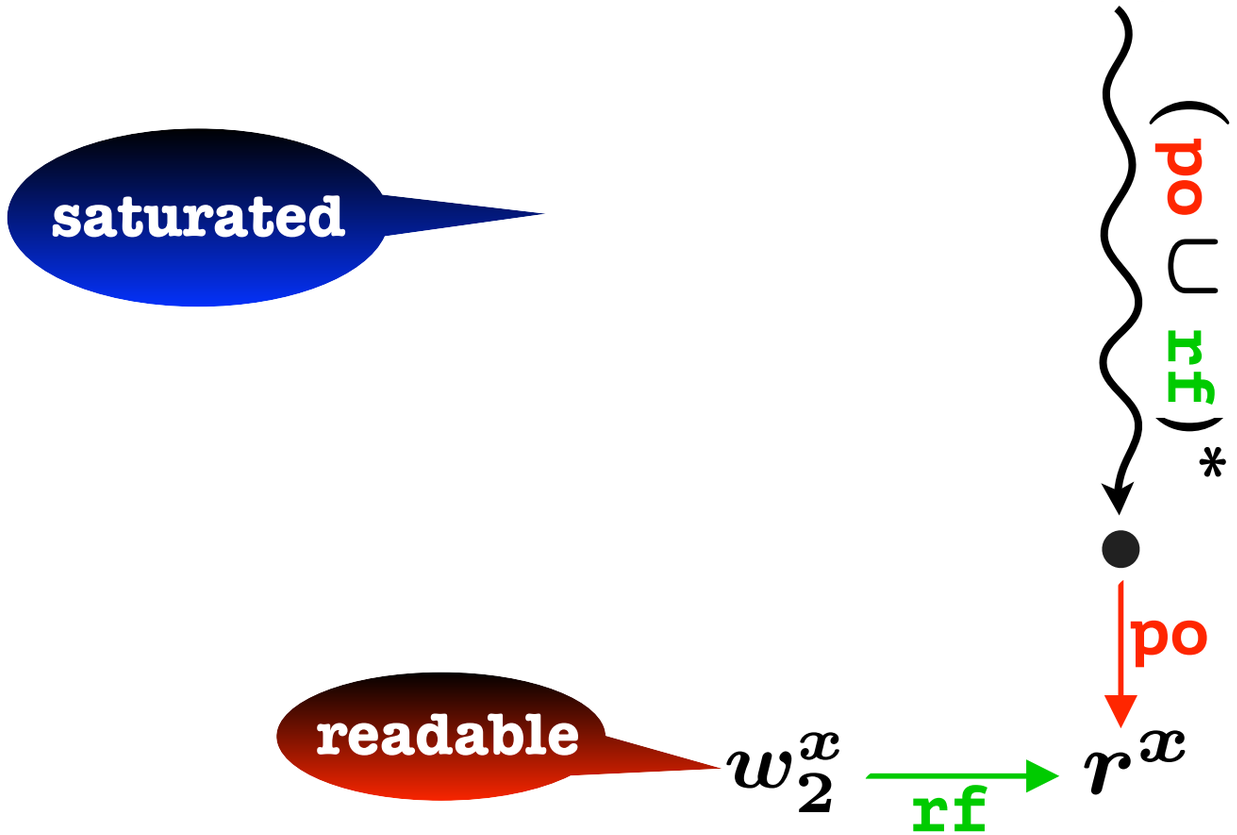
Adding Events to Traces



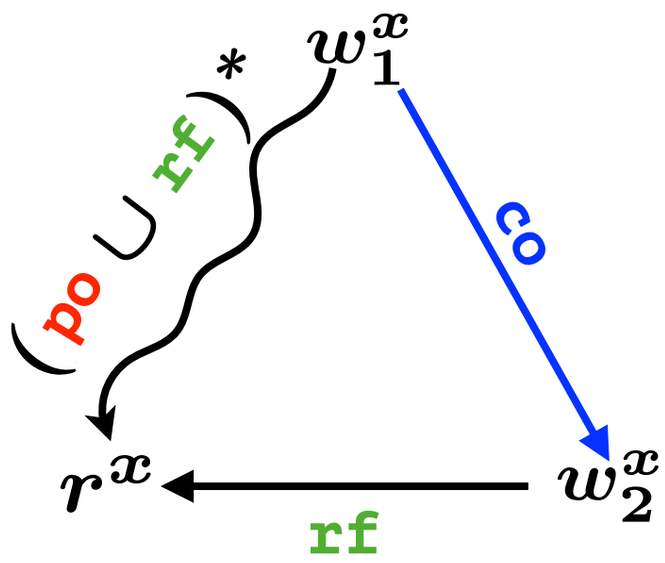
Adding Events to Traces

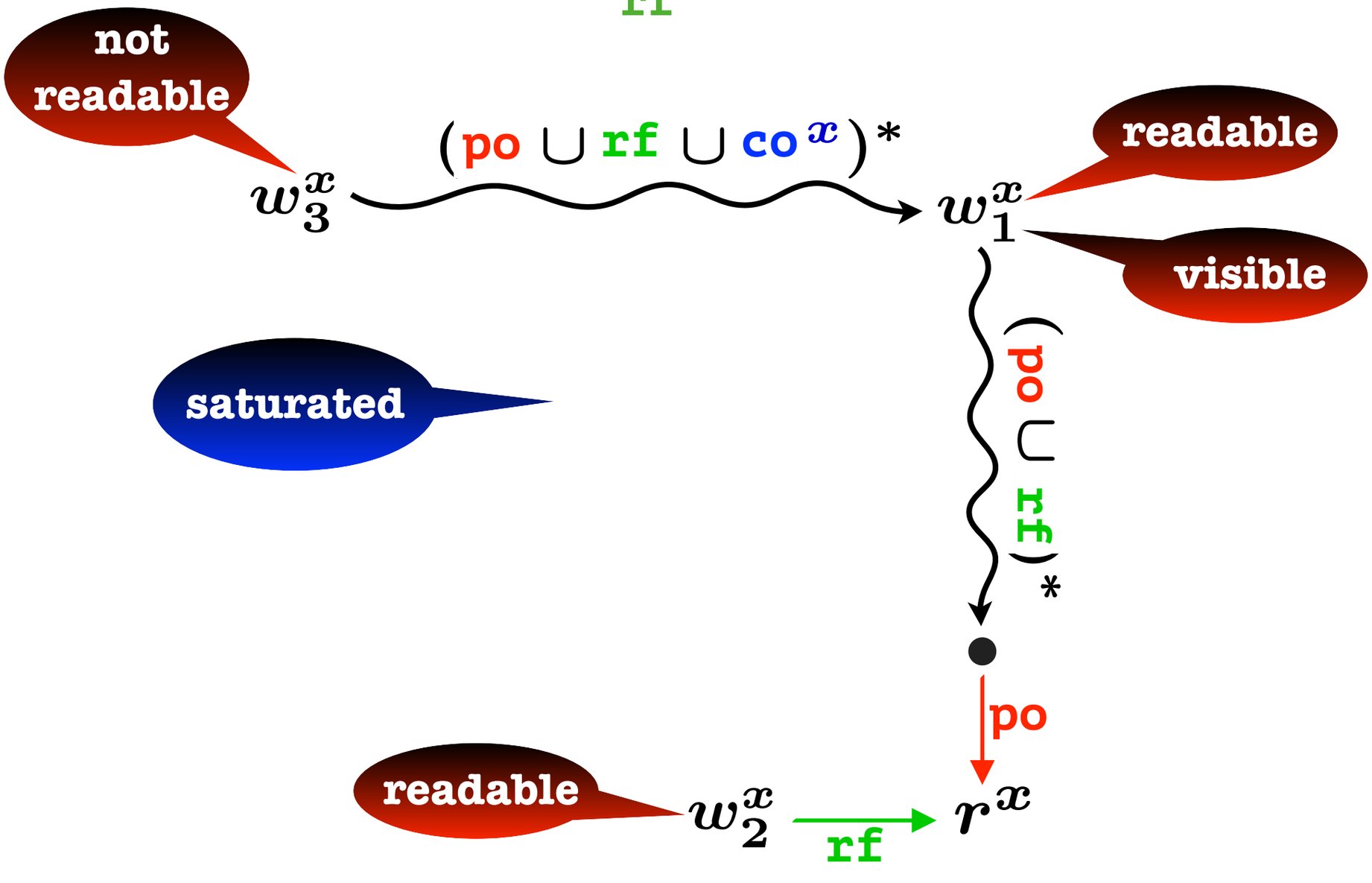




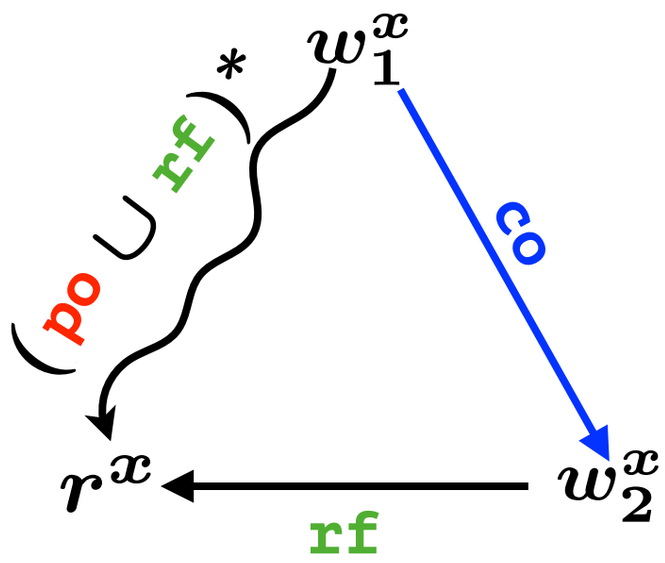


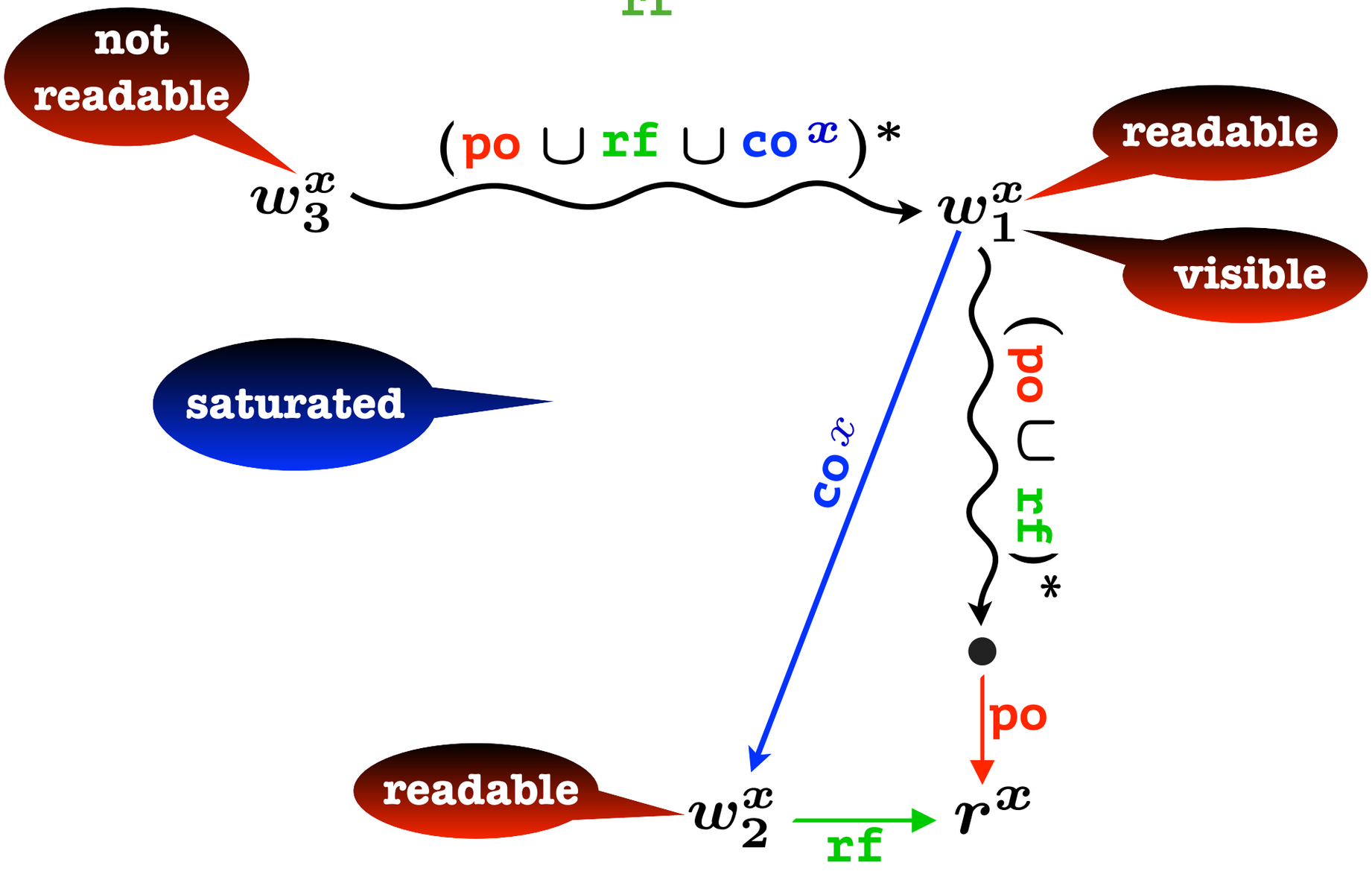
Adding Events to Traces



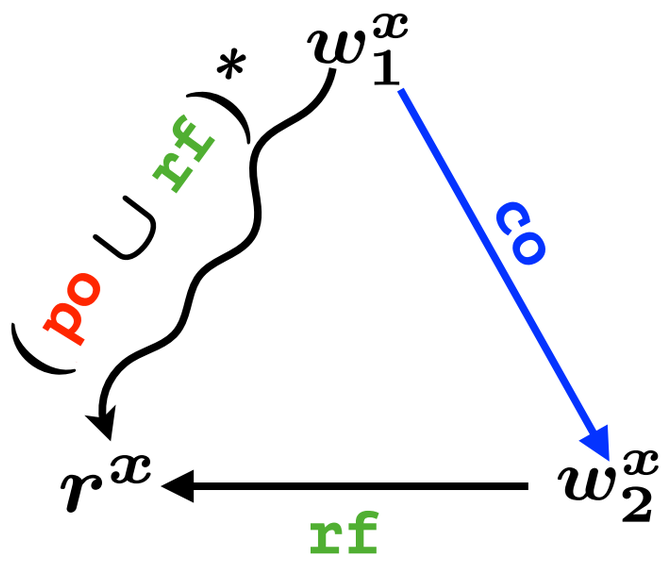


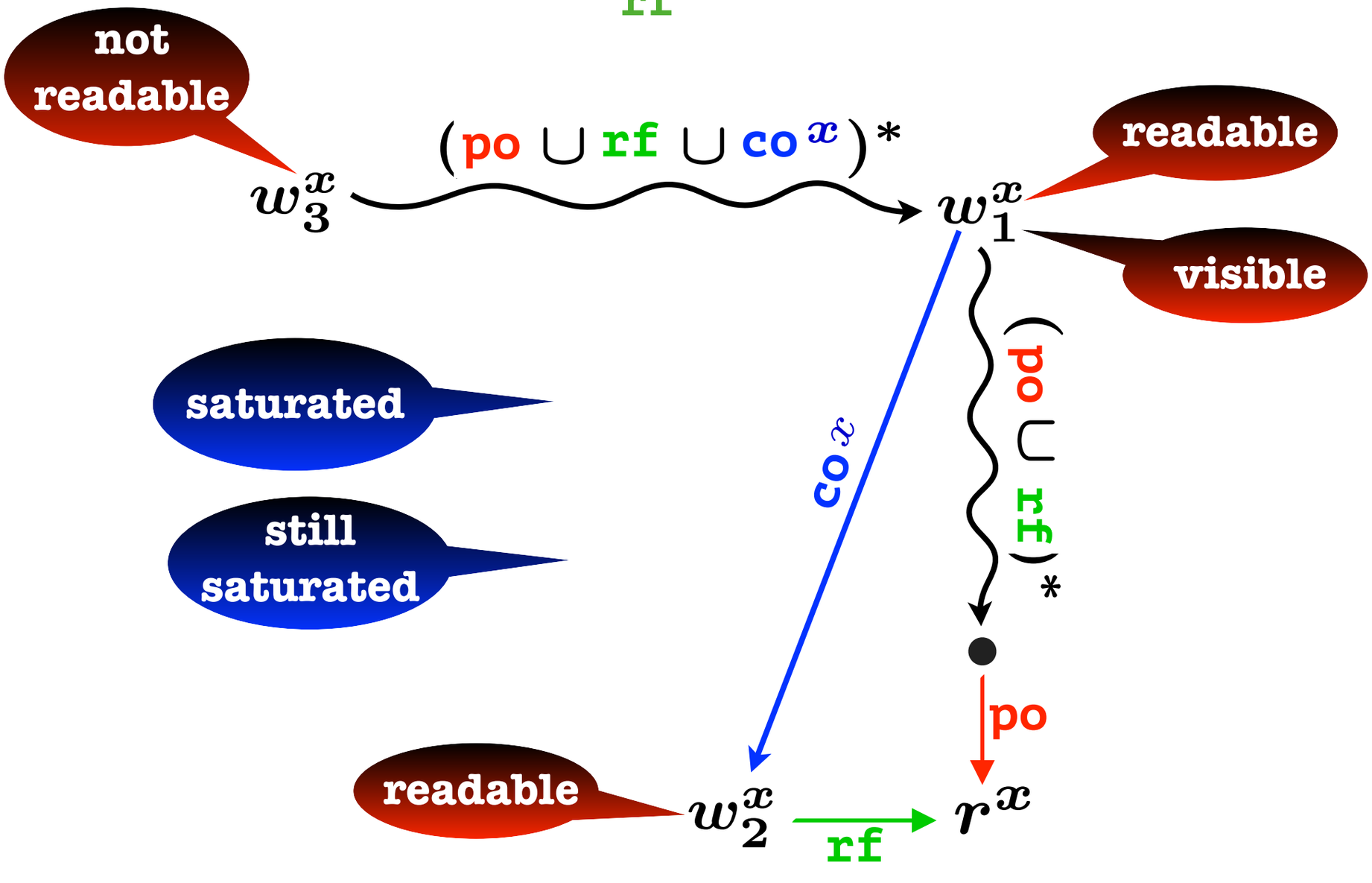
Adding Events to Traces



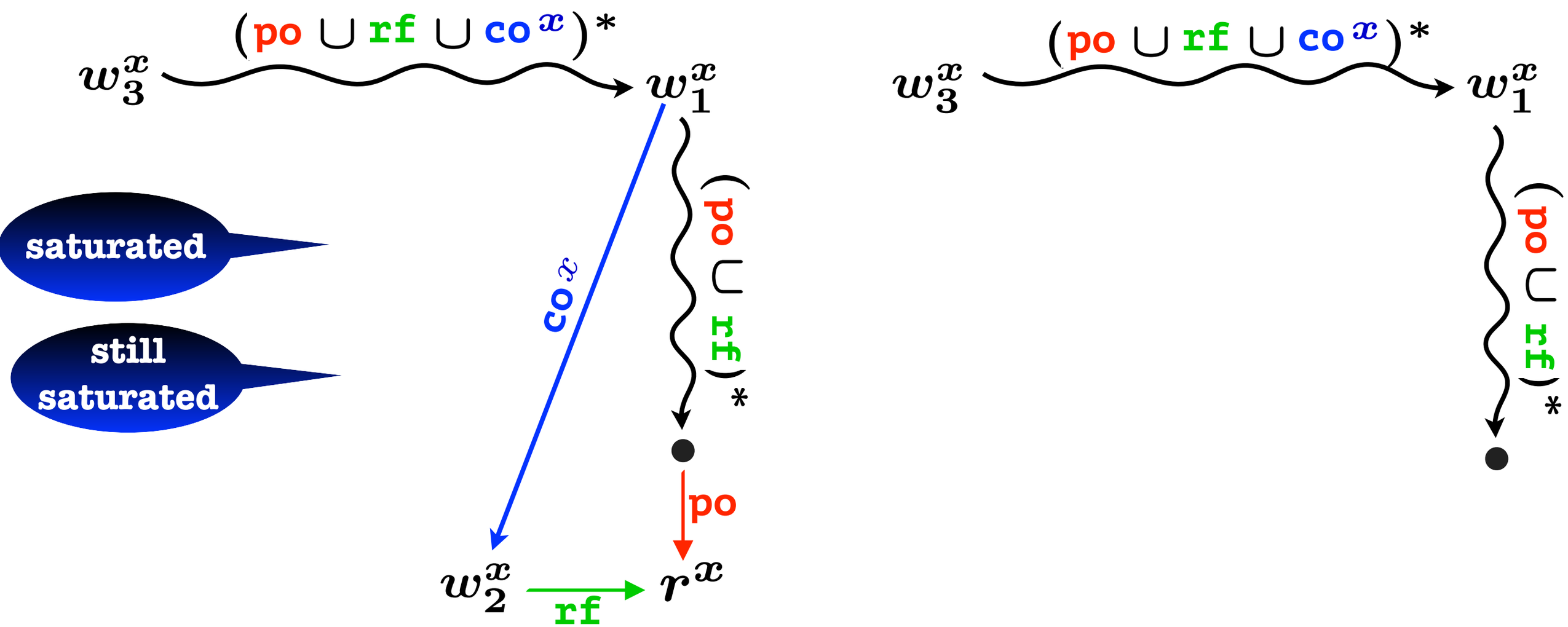


Adding Events to Traces

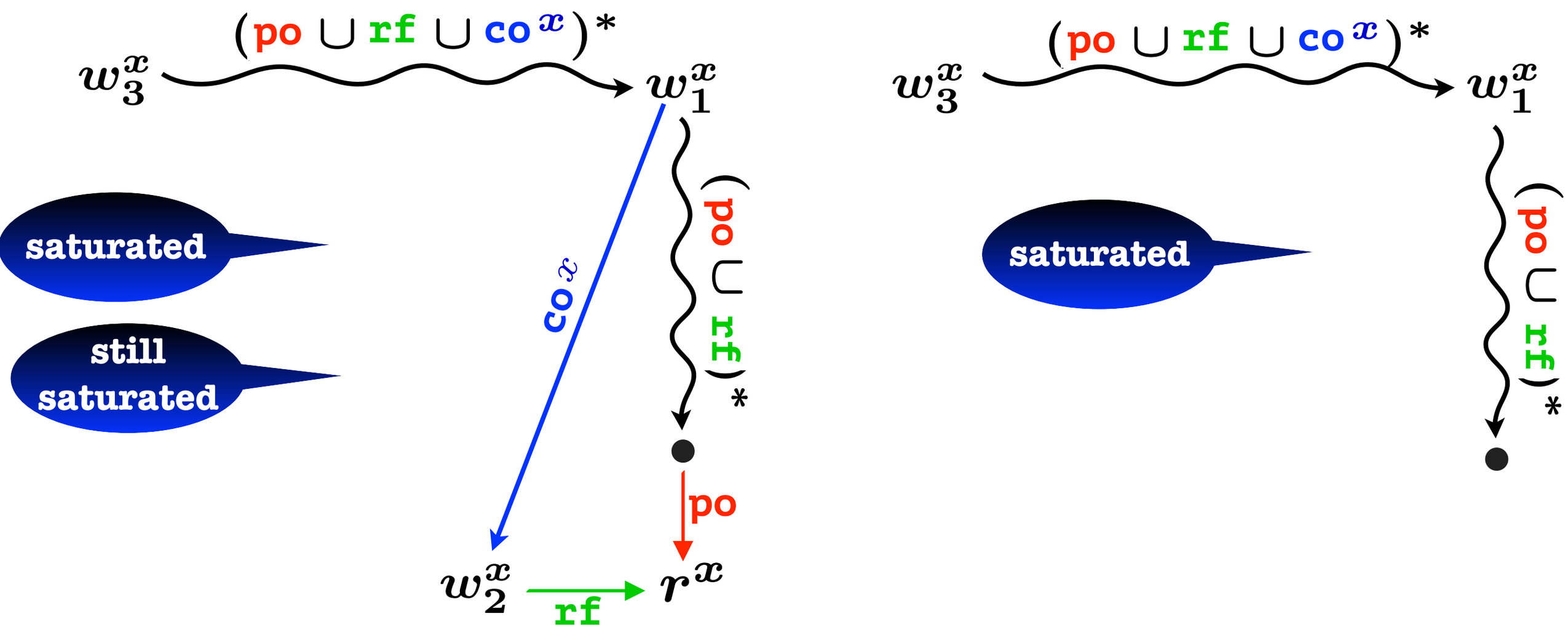




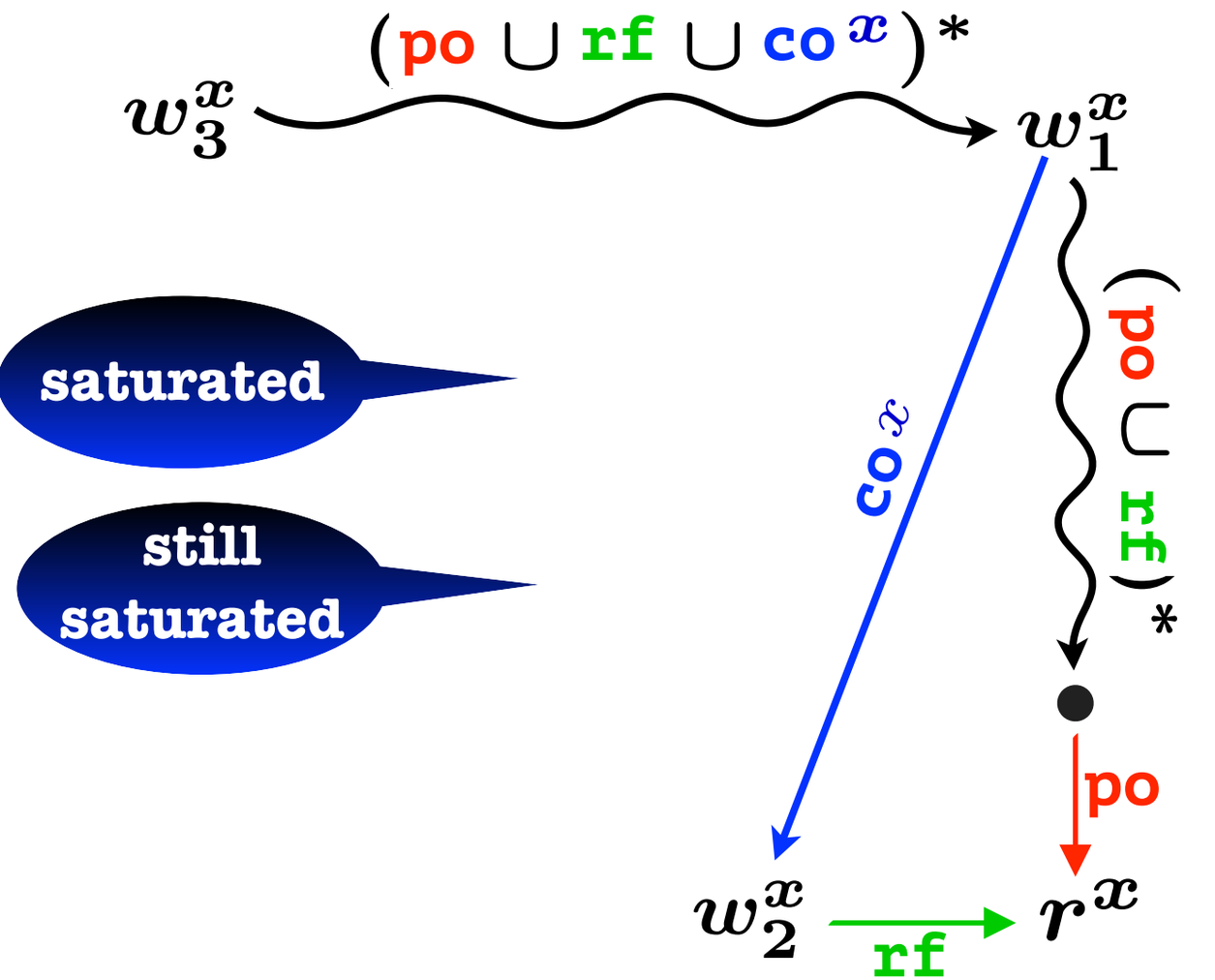
Adding Events to Traces

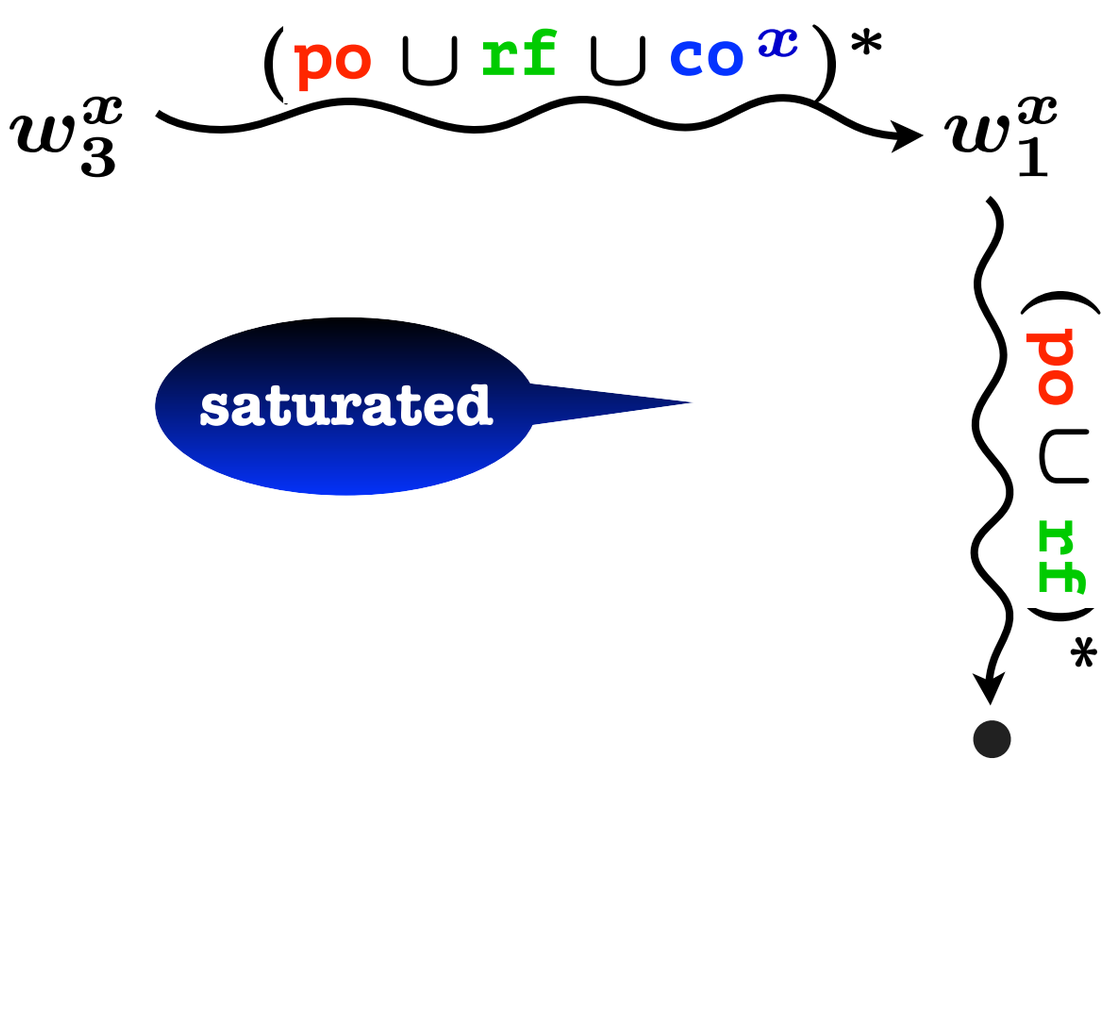


Adding Events to Traces

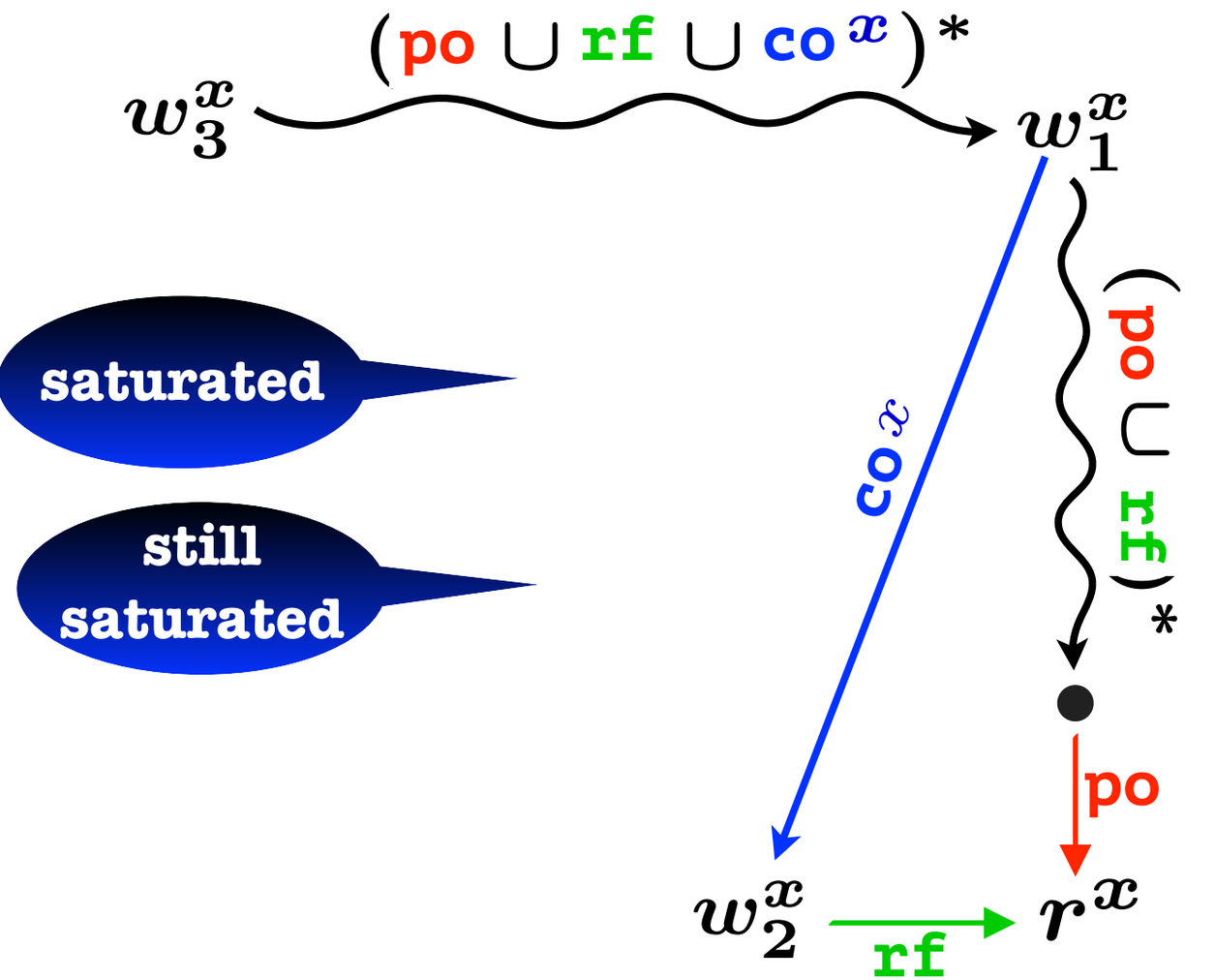


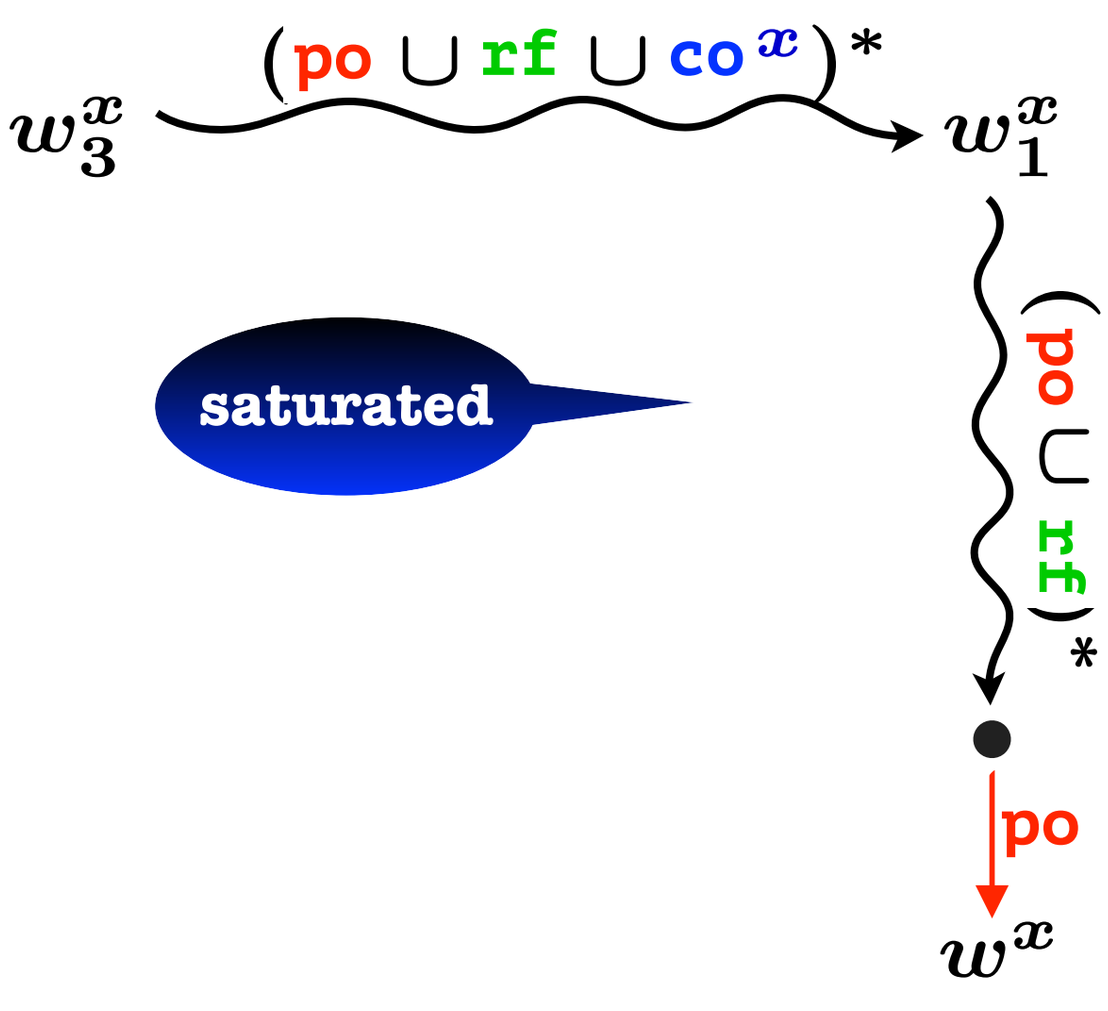
Adding Events to Traces



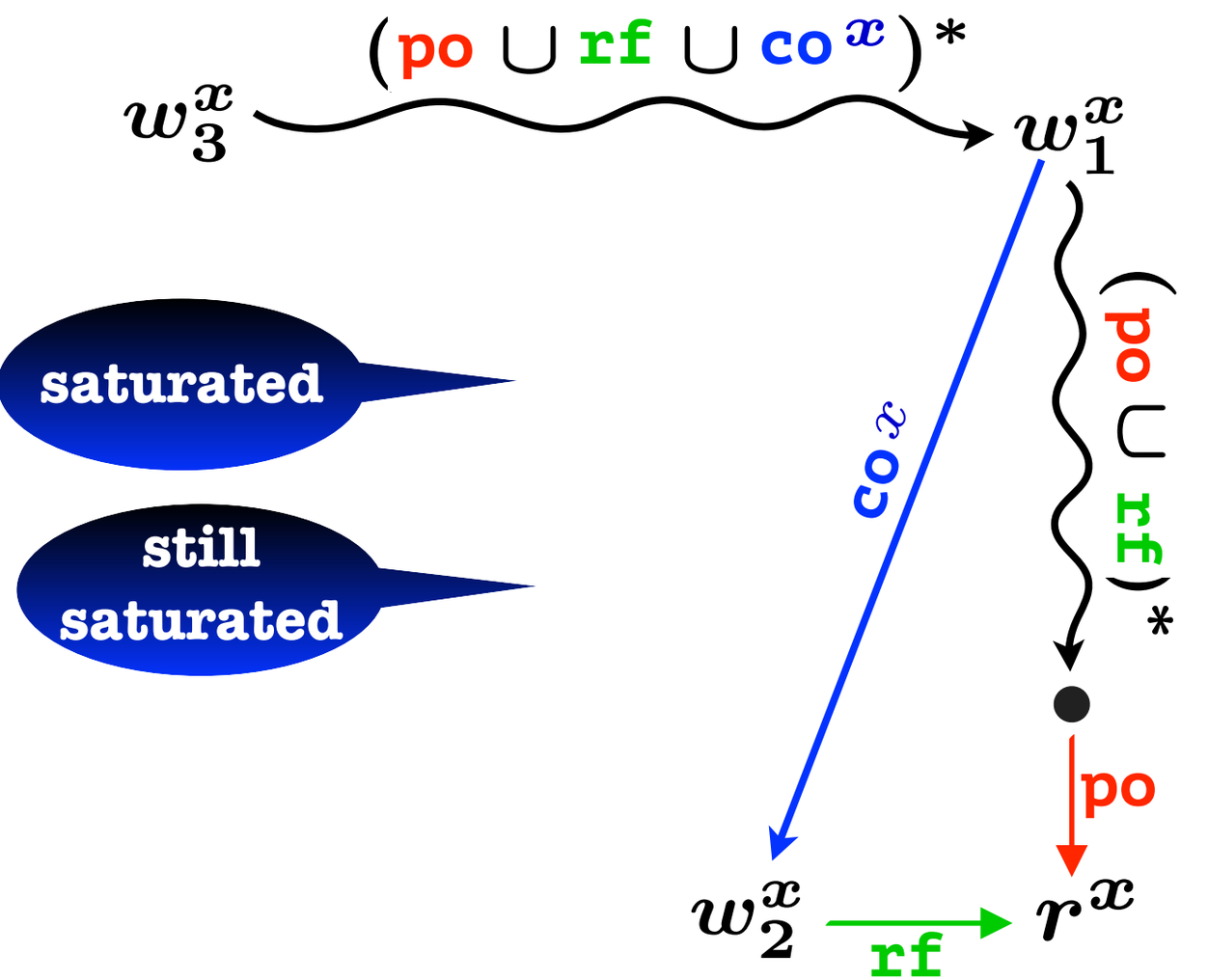


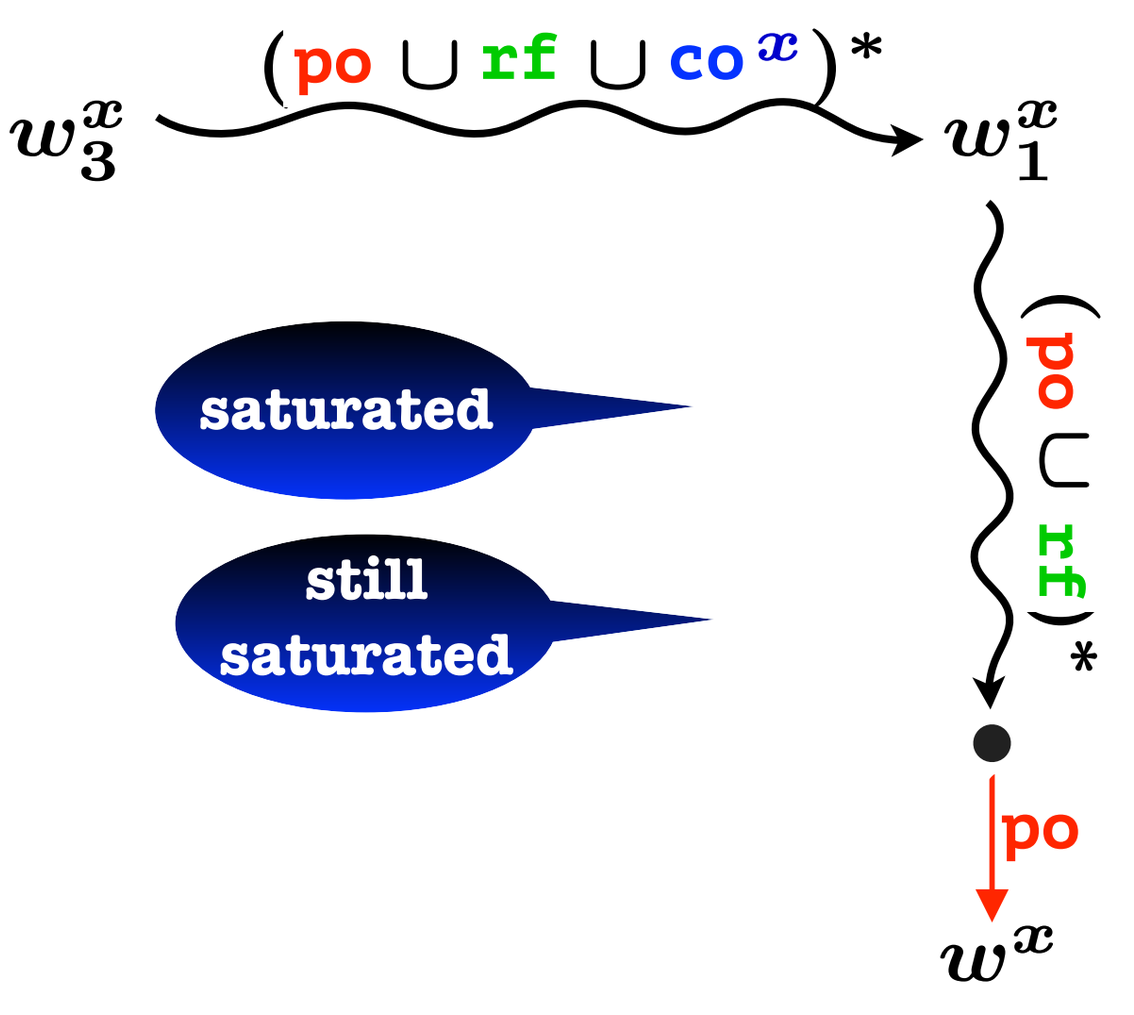
Adding Events to Traces





Adding Events to Traces





DPOR under the RA-semantics

- Build traces one after one
- For a given trace τ :
  - Build τ incrementally
  - extend τ by one event e at a time

Branch **&** **Postpone**

DPOR under the RA-semantics

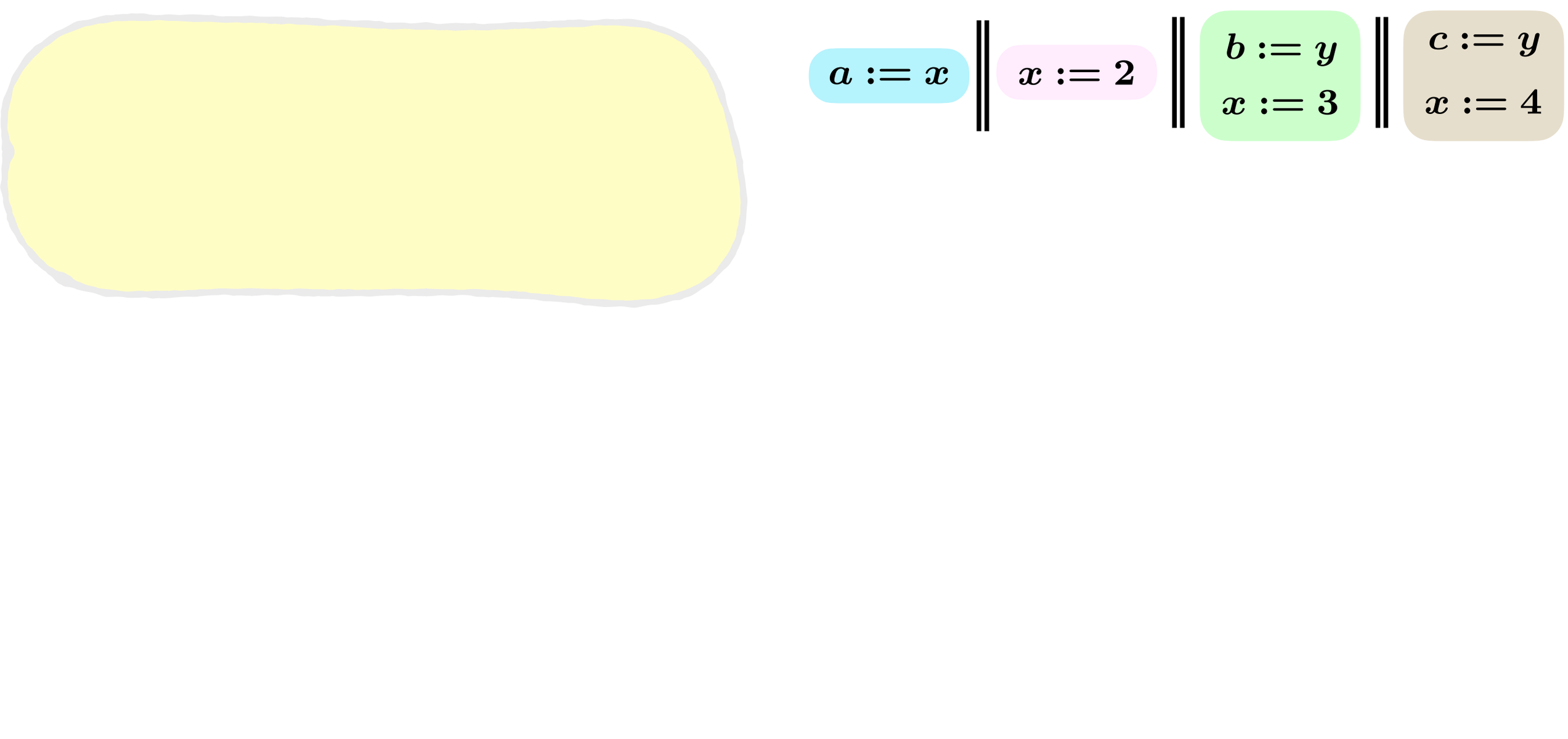
- Build traces one after one
- For a given trace τ :
- Build τ incrementally
- extend τ by one event e at a time

Branch **&** **Postpone**

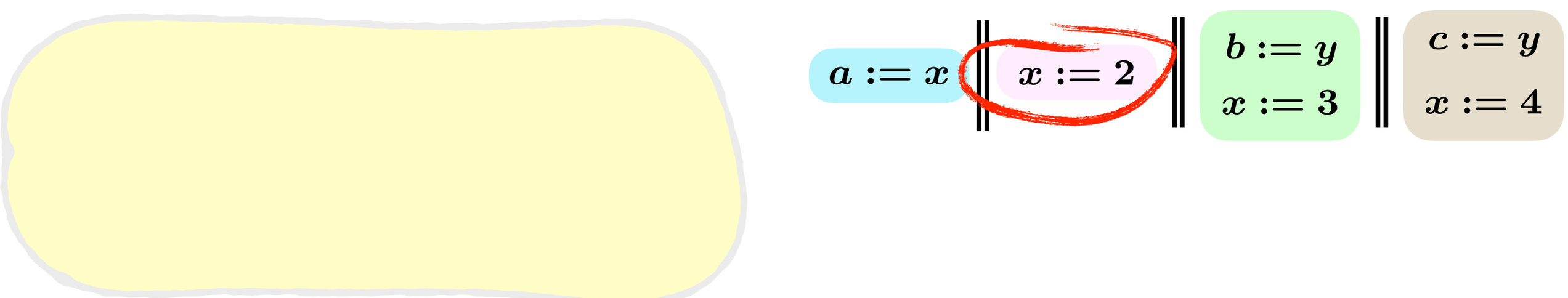
DPOR under the RA-semantics

- Build traces one after one
- For a given trace τ:
  - Build τ incrementally
    - extend τ by one event e at a time

Branch **&** **Postpone**



Branch **&** **Postpone**



Branch **&** **Postpone**

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

$x := 2$

Branch & **Postpone**

$x := 2$

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

$x := 2$

Branch **&** **Postpone**

$x := 2$

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

$x := 2$

Branch **&** **Postpone**

$x := 2$

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

$x := 2$

Branch **&** **Postpone**

$x := 2$

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

$x := 2$

$a := 0$ $a := 2$

Branch & Postpone

$x := 2$

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

$x := 2$

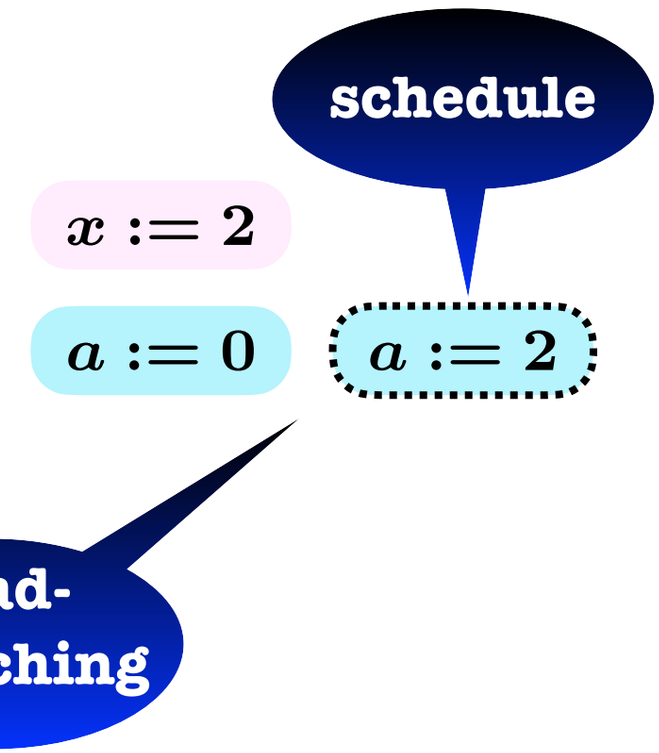
$a := 0$ $a := 2$

read-branching

Branch & Postpone

$x := 2$

$a := x \parallel x := 2 \parallel b := y \;\; x := 3 \parallel c := y \;\; x := 4$



Branch & Postpone

$x := 2$

$a := x$ ‖ $x := 2$ ‖ $b := y$ $x := 3$ ‖ $c := y$ $x := 4$

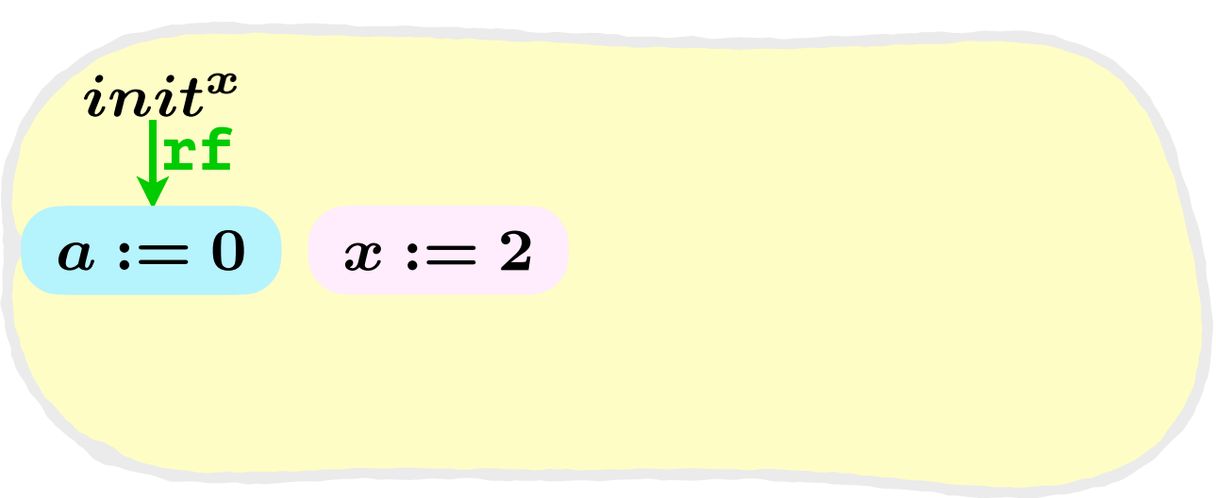
schedule

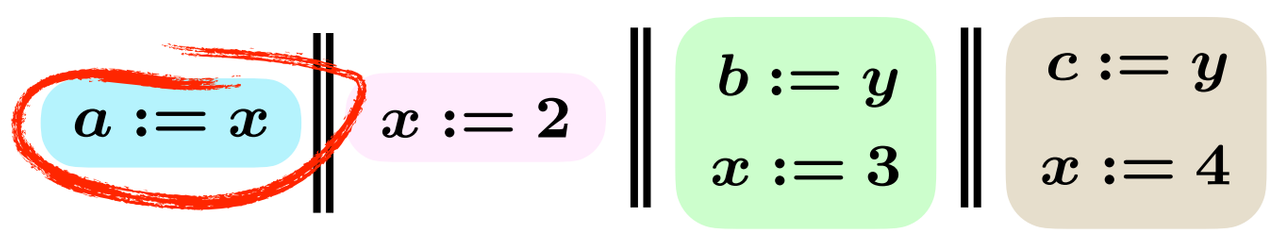
$x := 2$

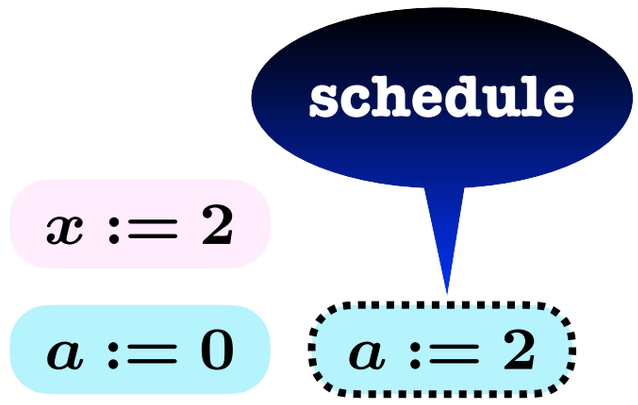
$a := 0$

$a := 2$

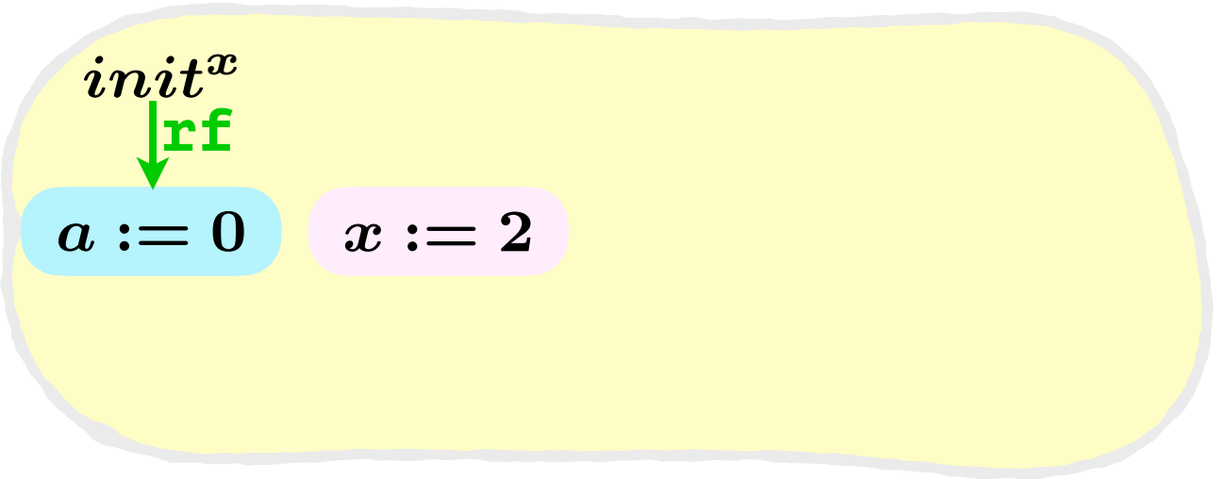
Branch **&** **Postpone**

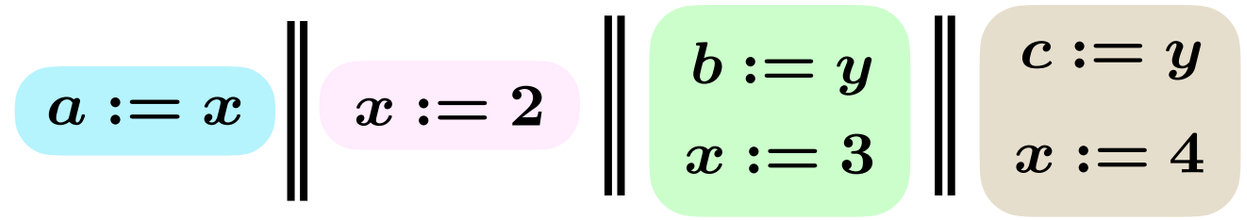


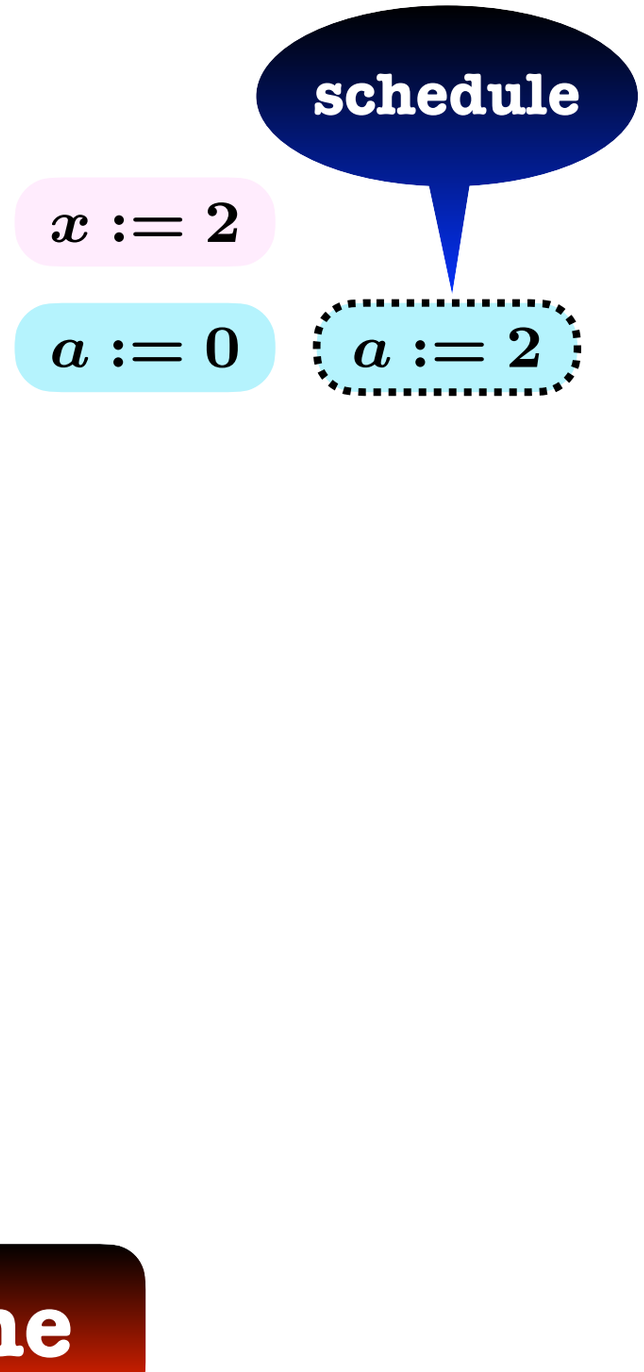




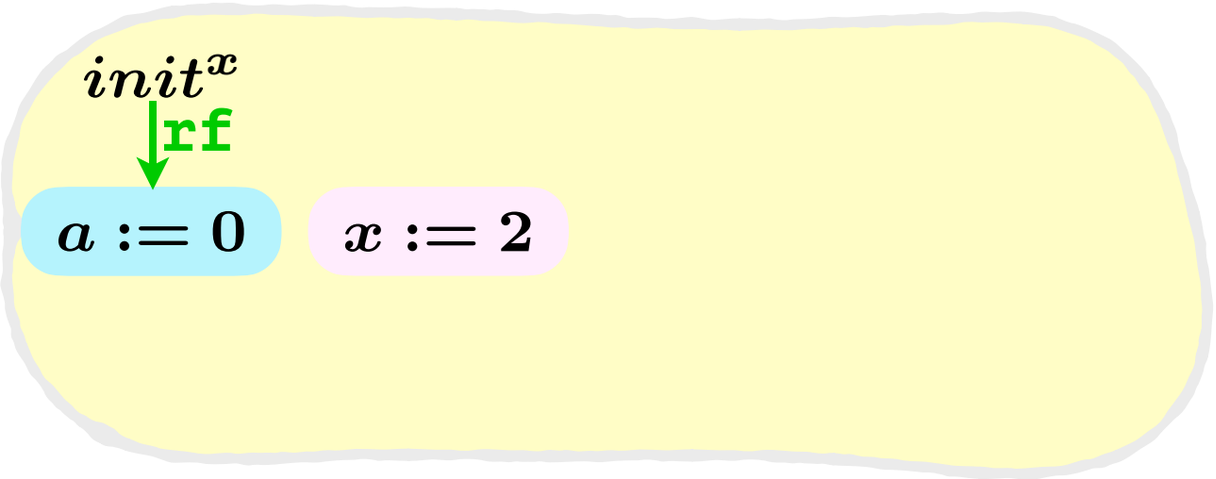
Branch & Postpone

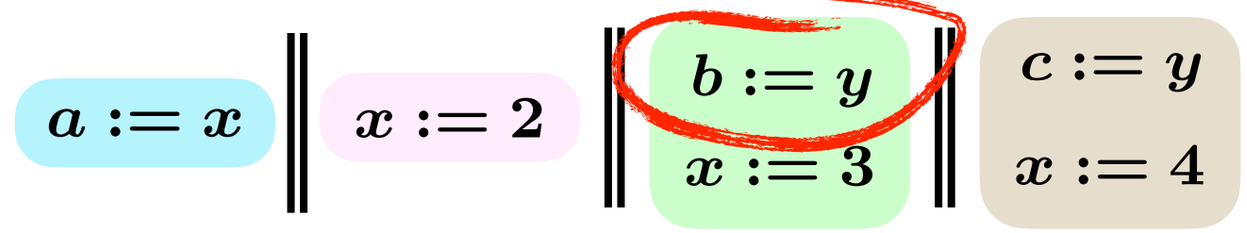


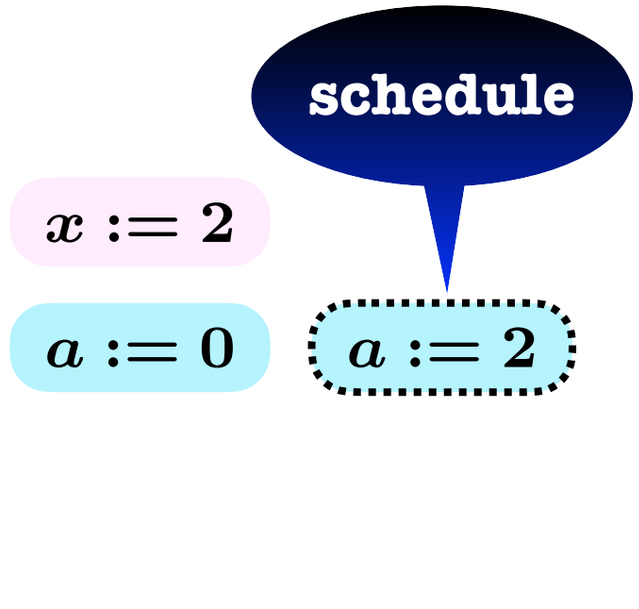




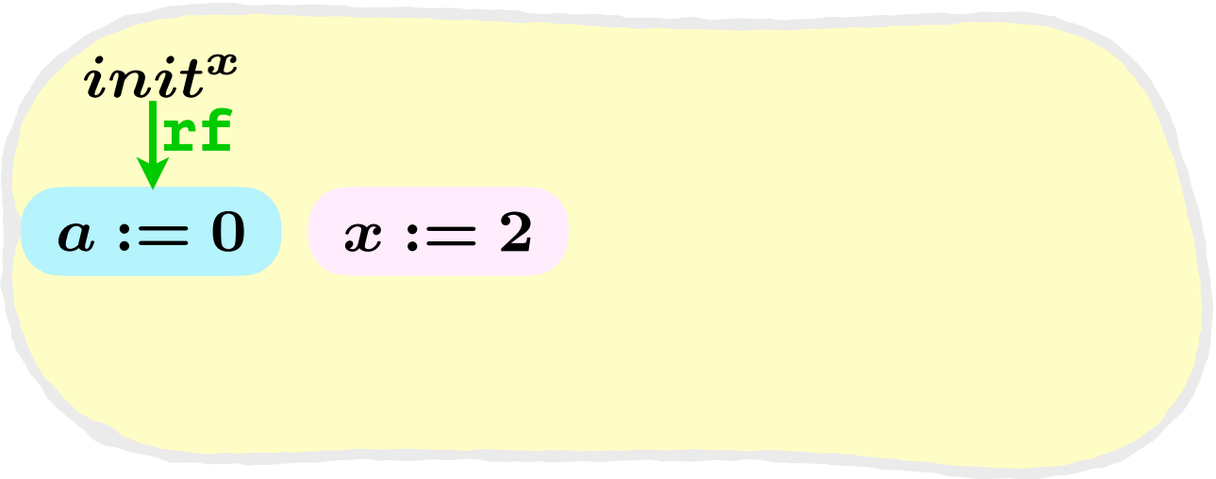
Branch & Postpone

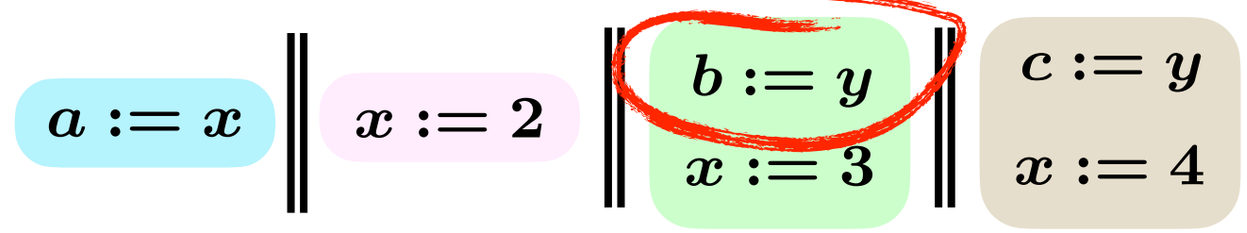


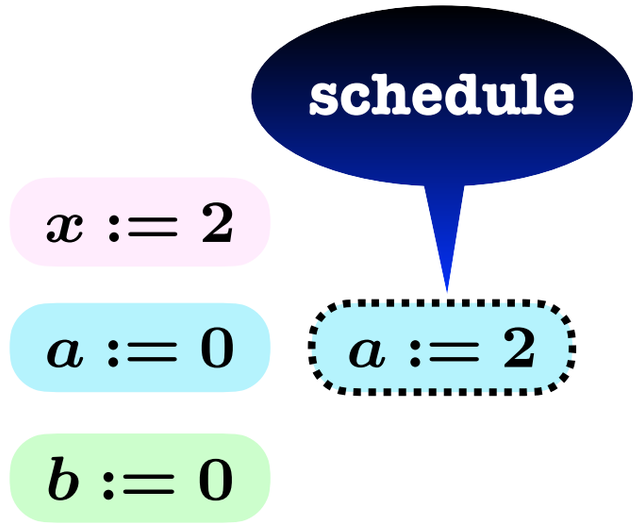




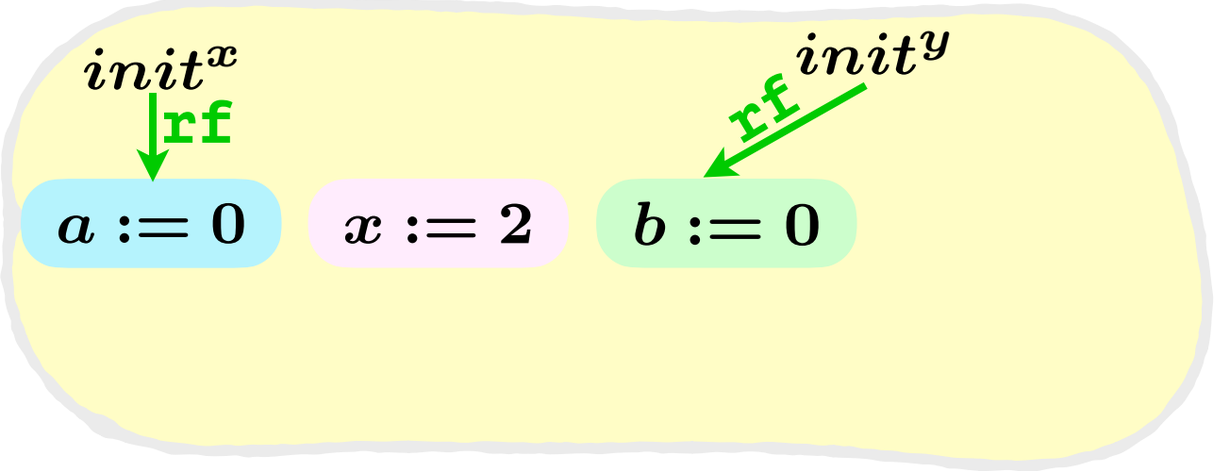
Branch & Postpone

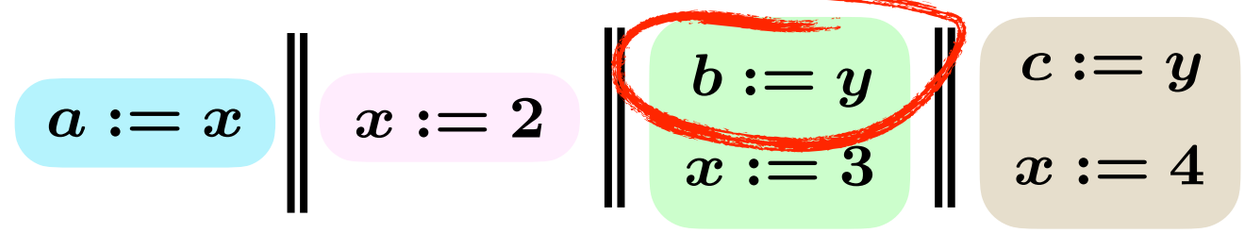


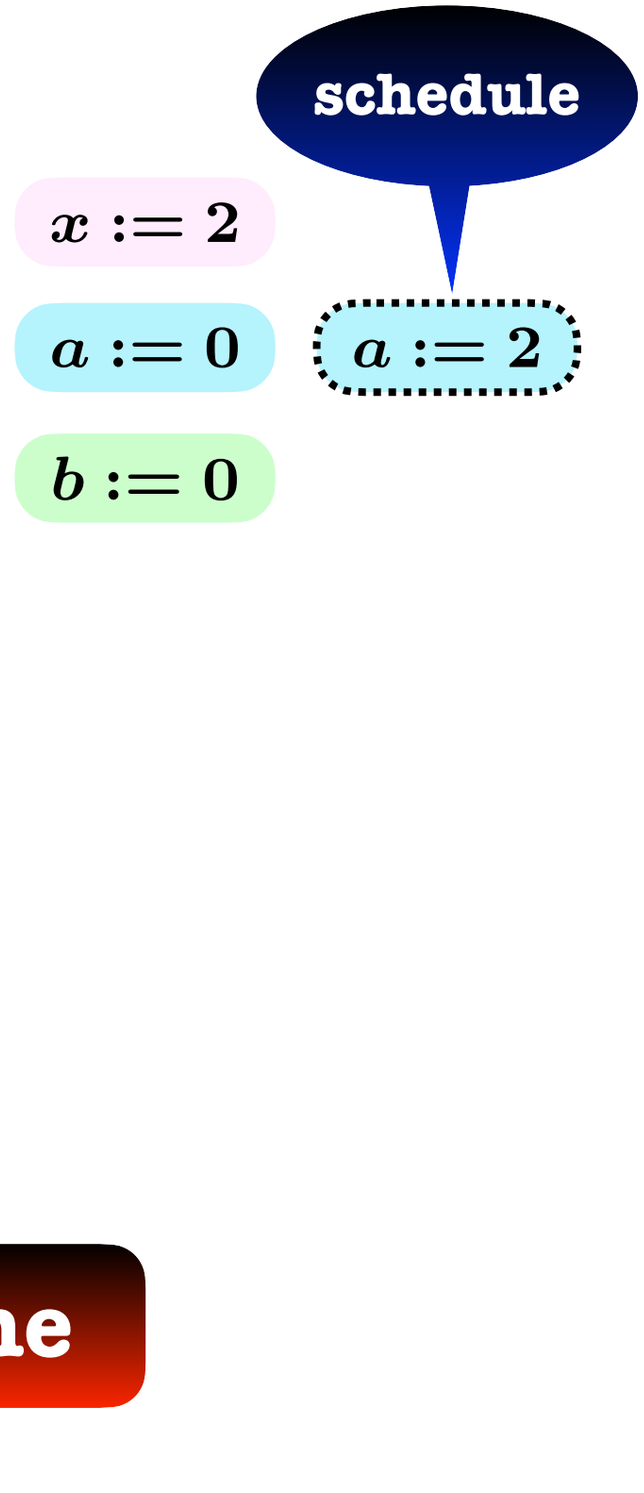




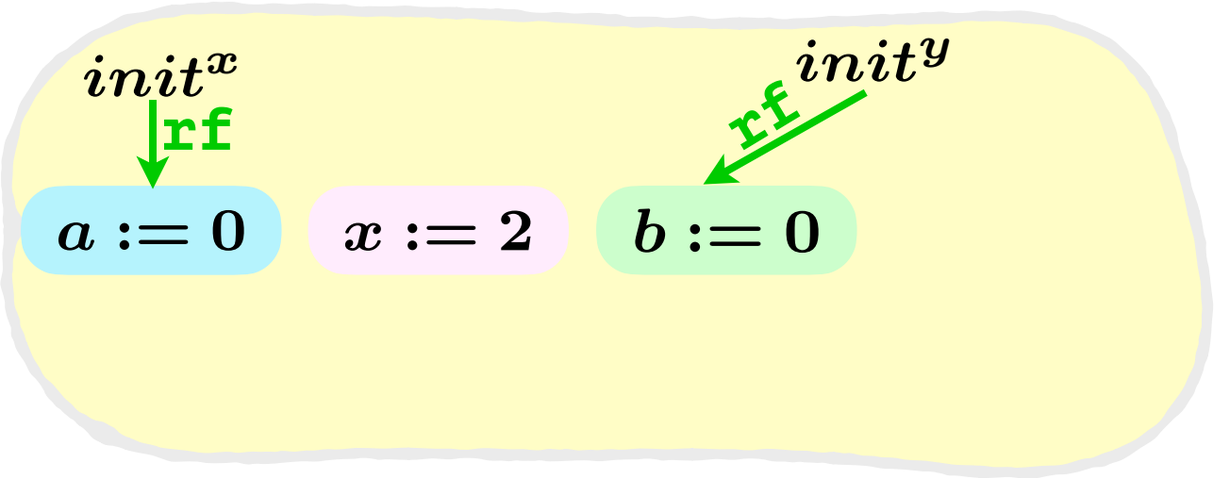
Branch & Postpone

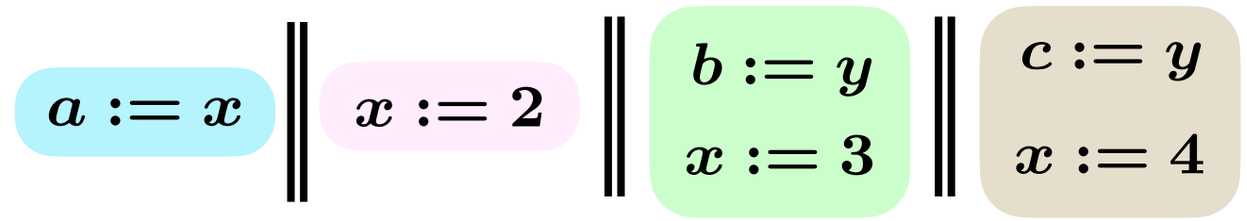


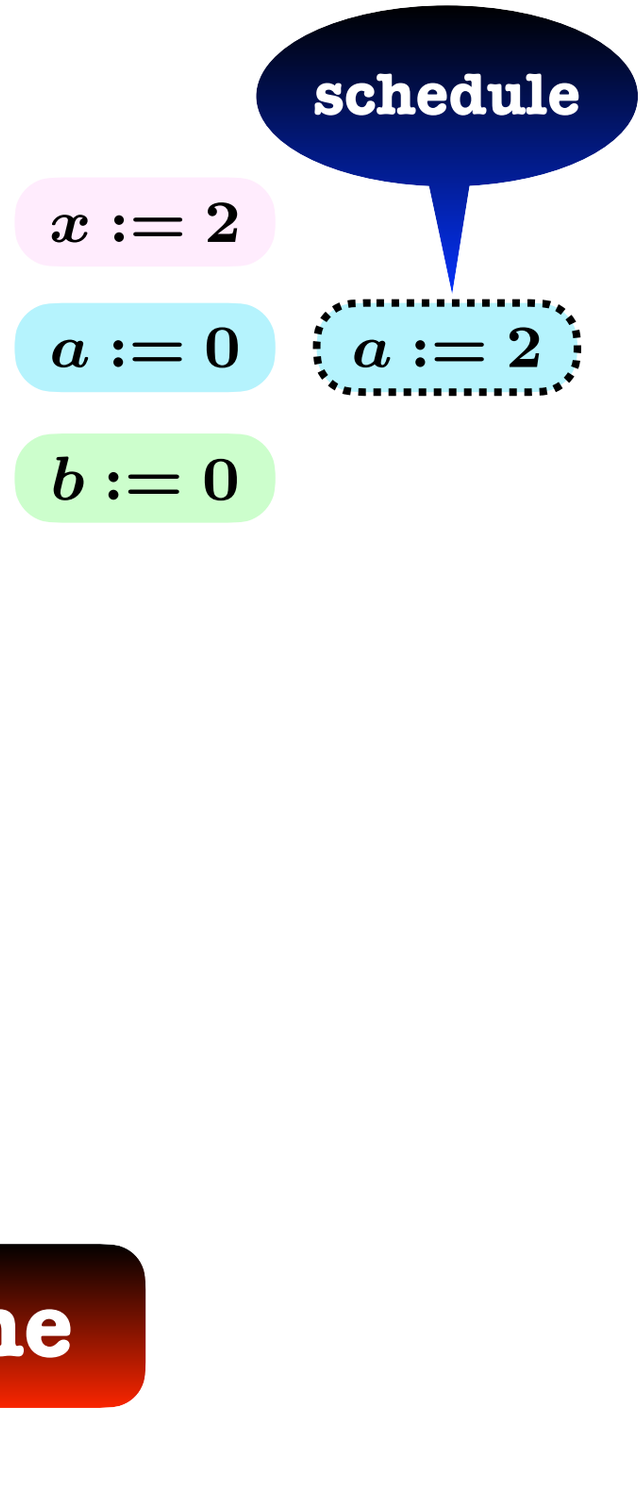




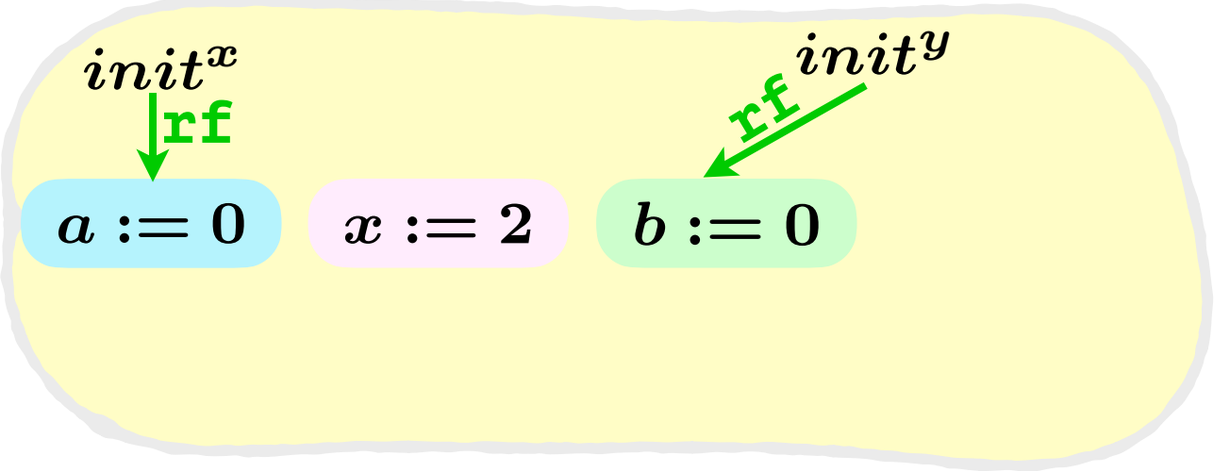
Branch & Postpone

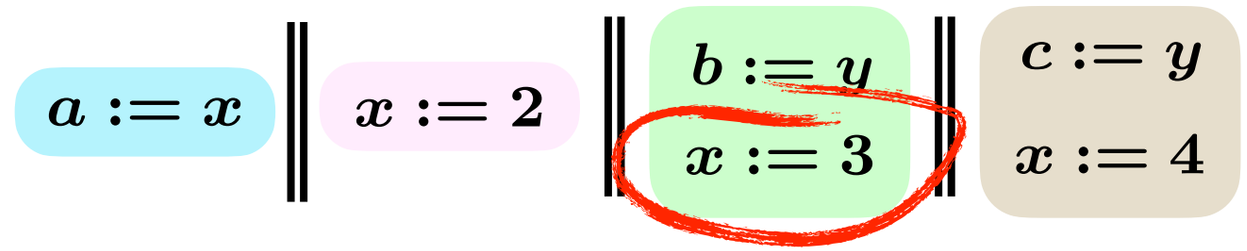


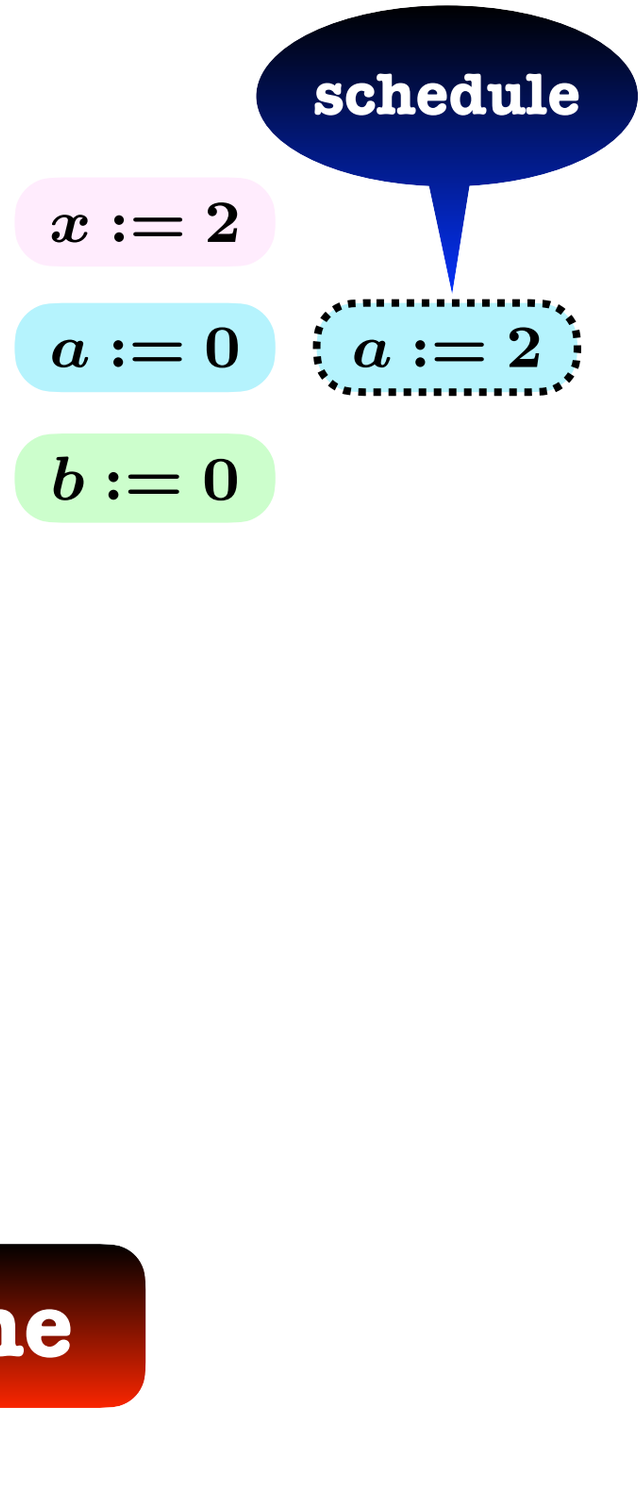




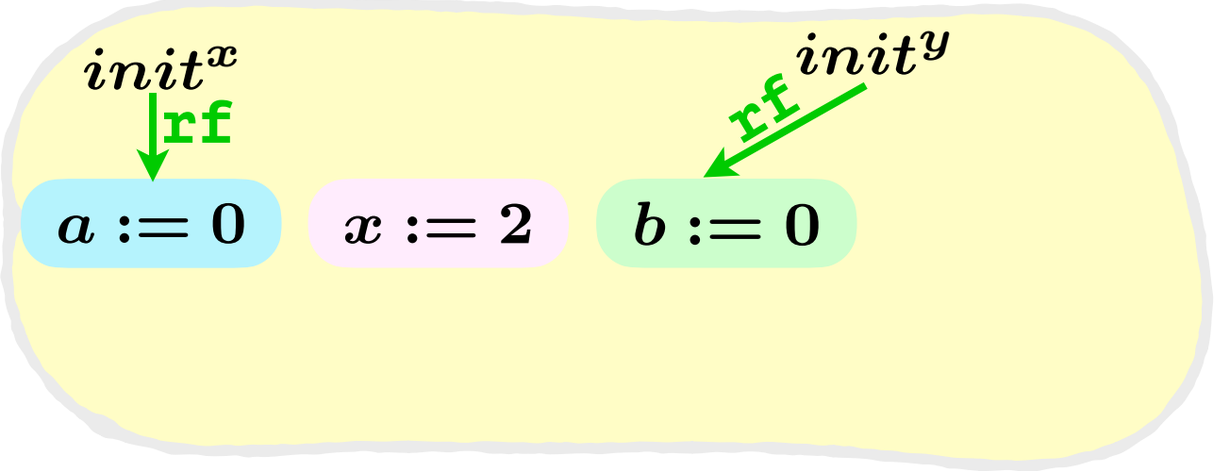
Branch & Postpone

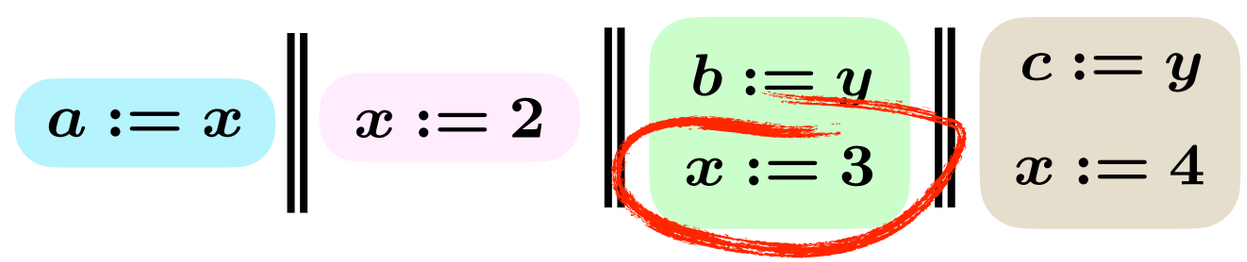


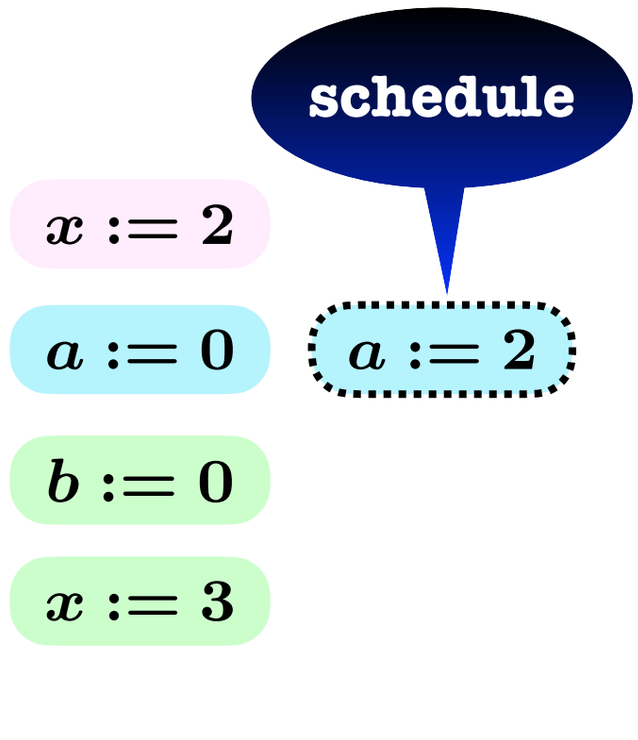




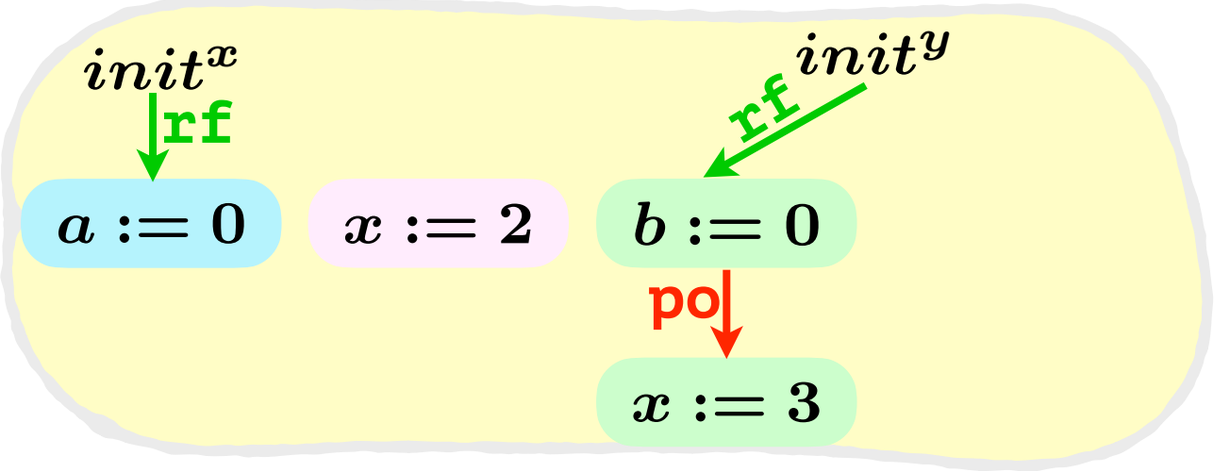
Branch & Postpone

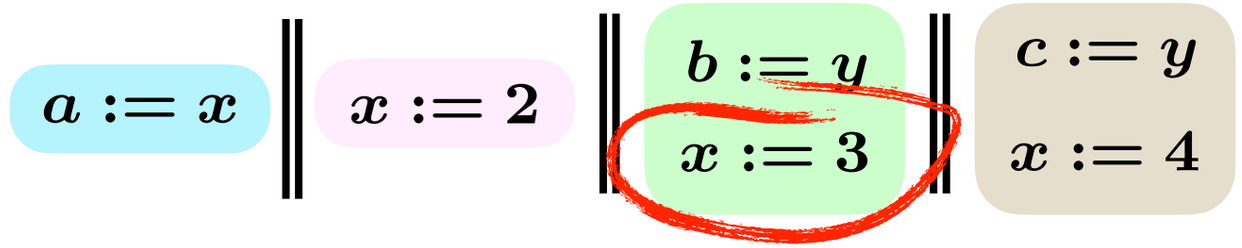


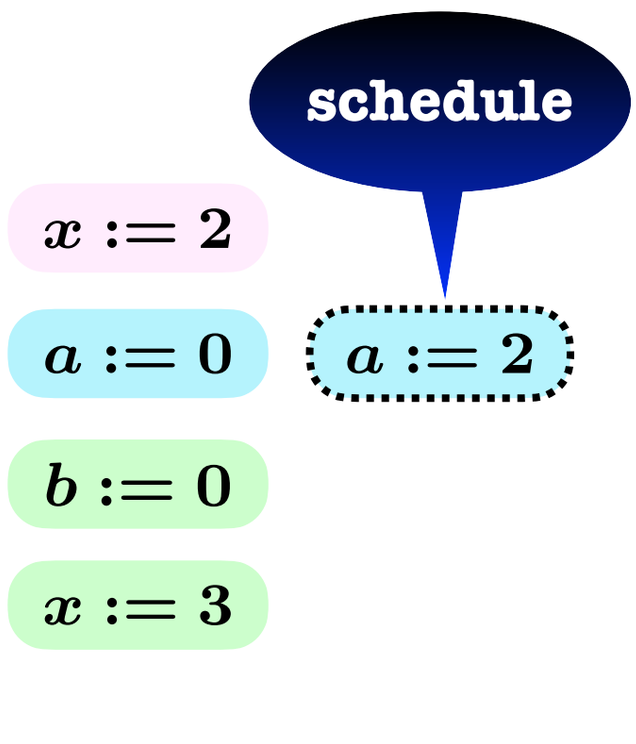




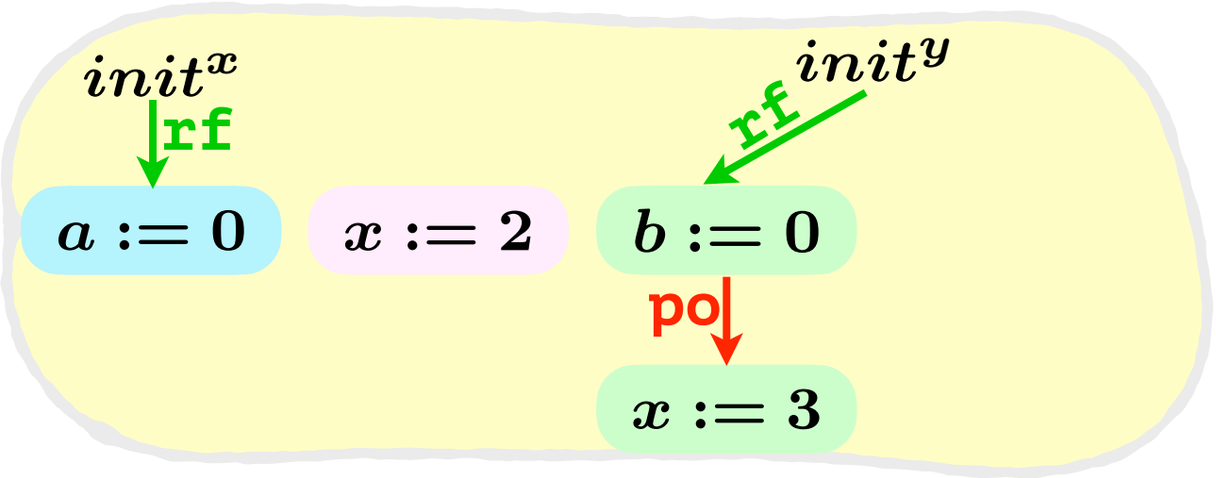
Branch **&** **Postpone**

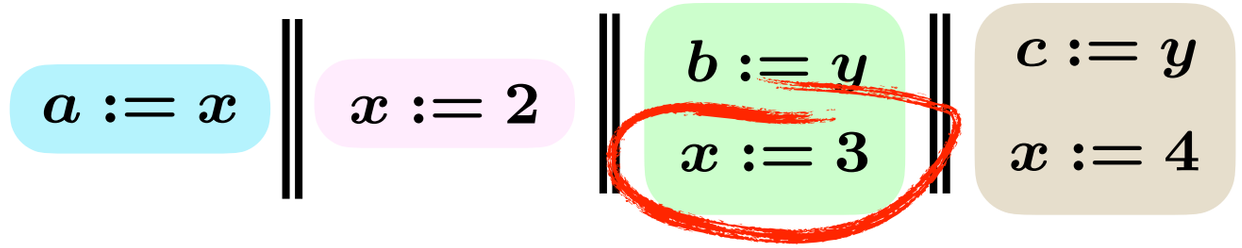


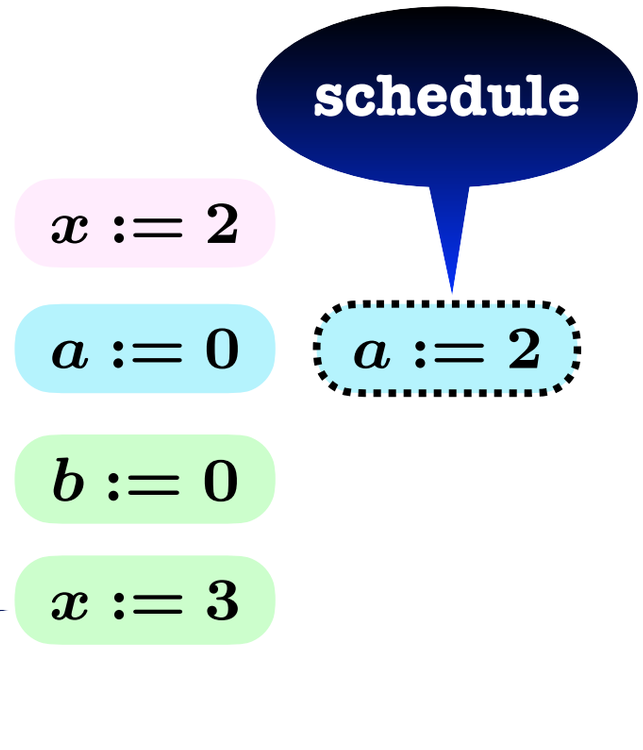




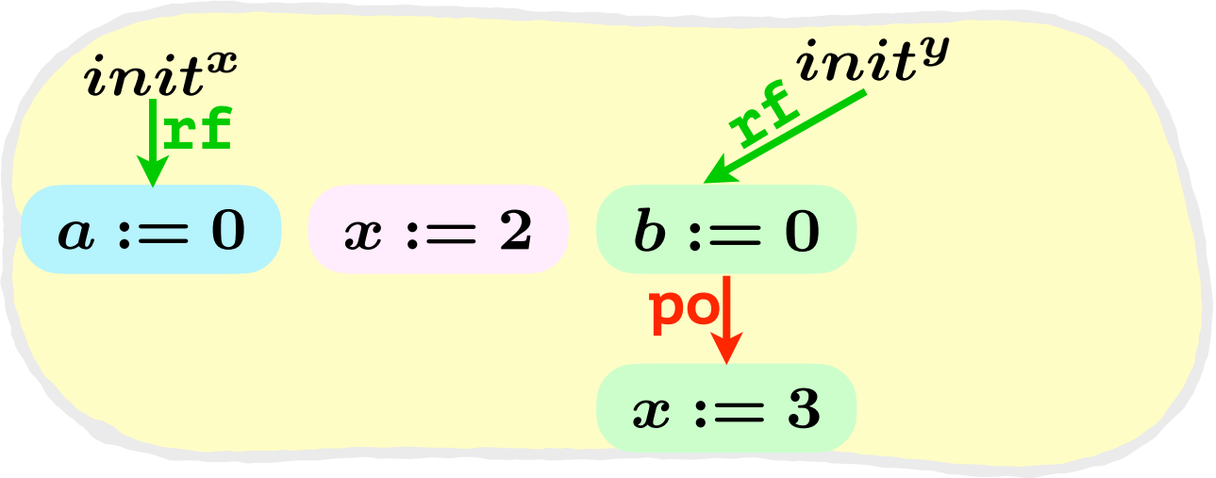
Branch **&** **Postpone**

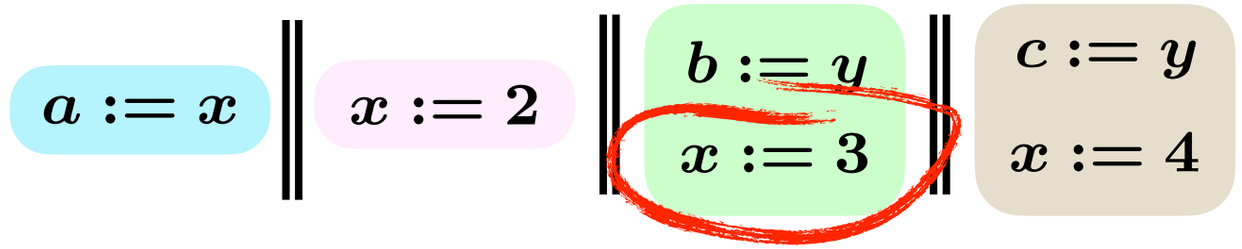


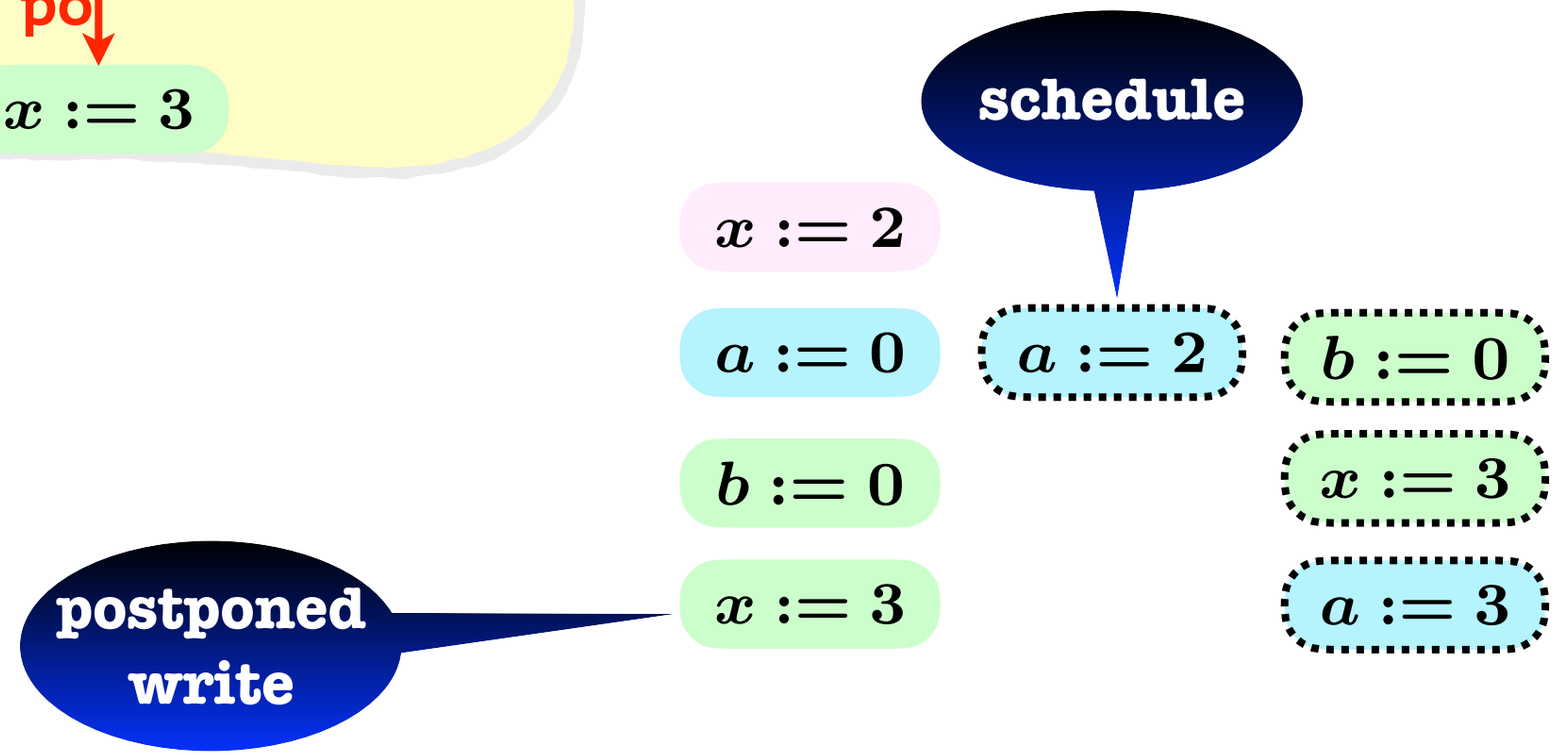




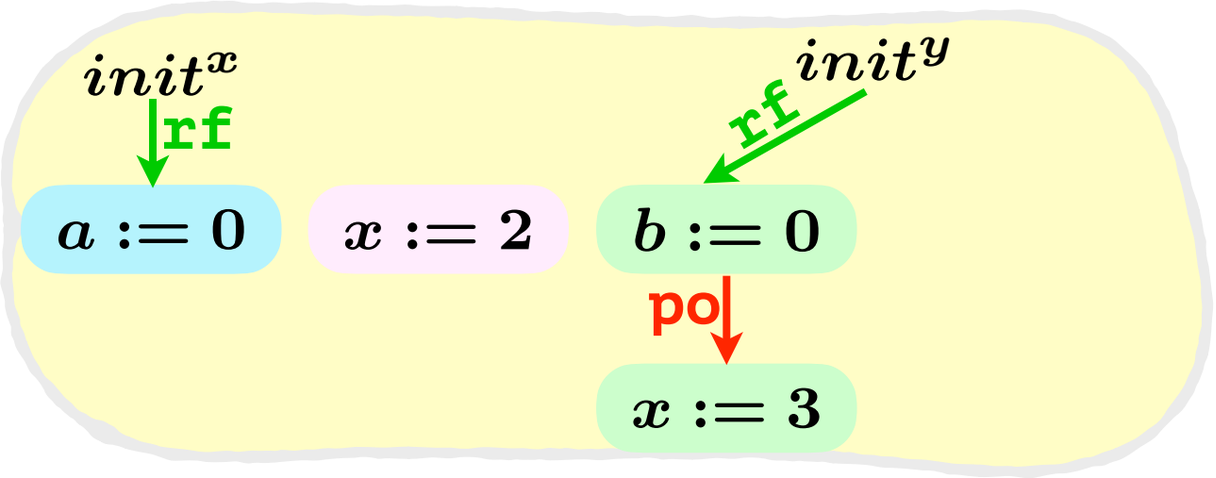
Branch & Postpone

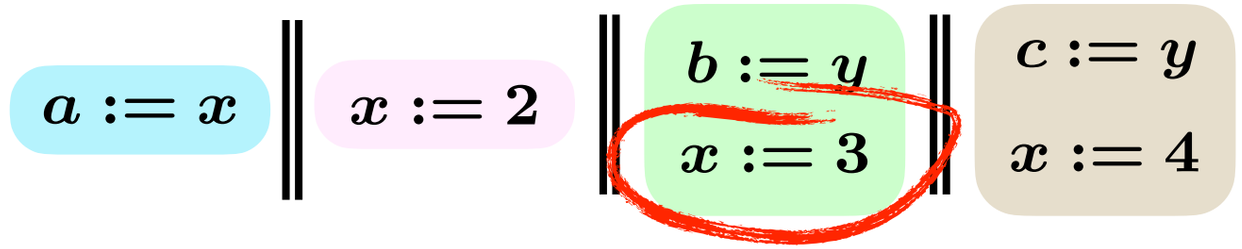


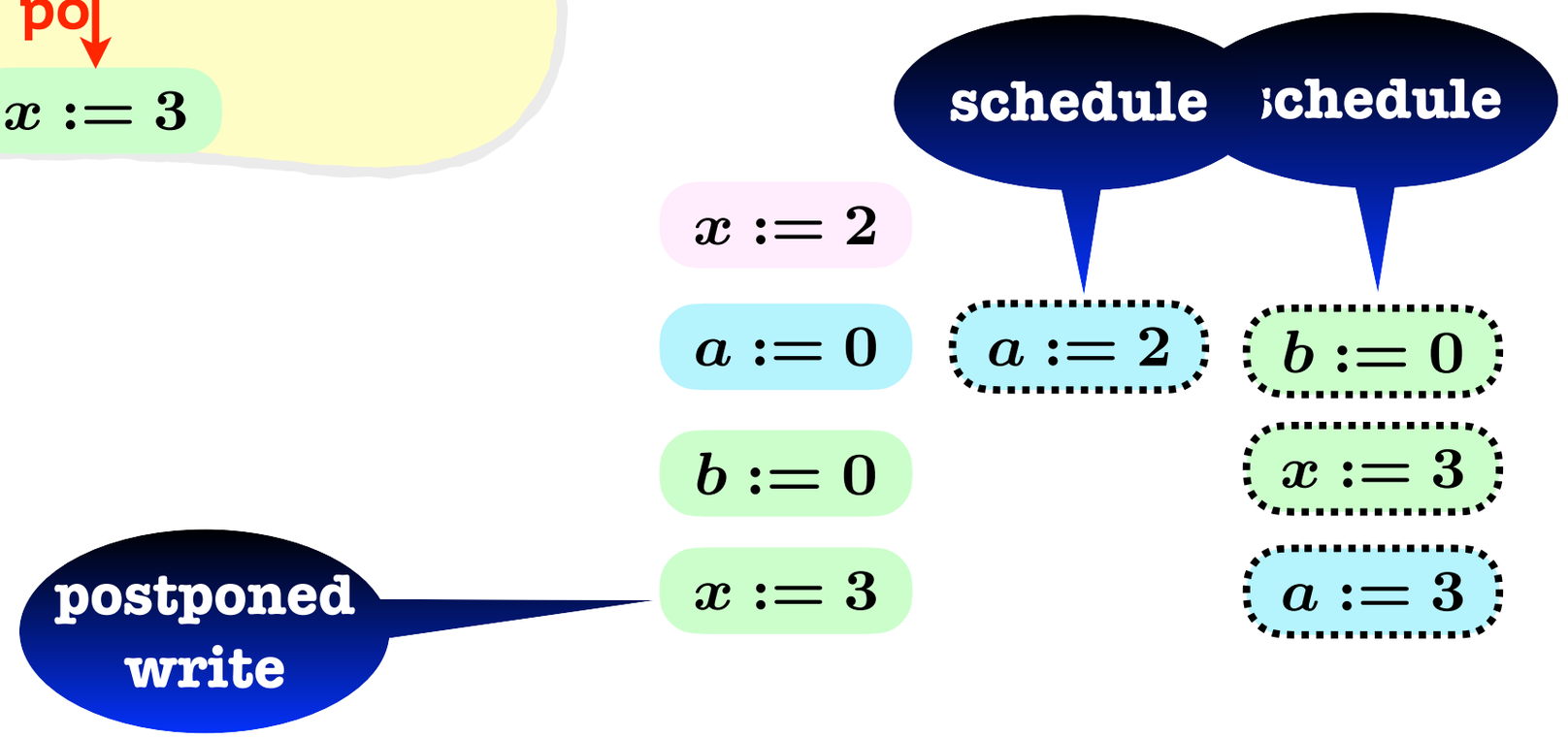




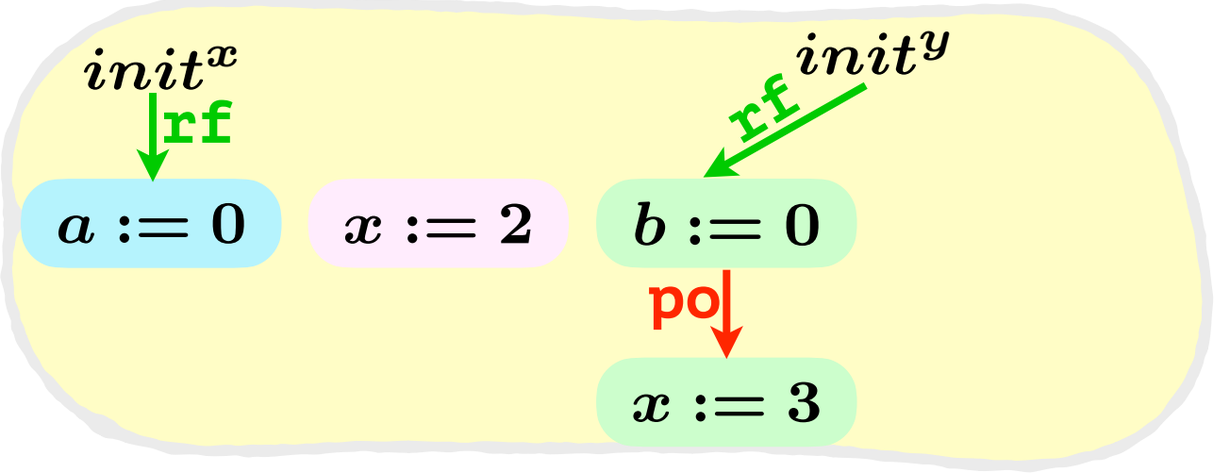
Branch & Postpone

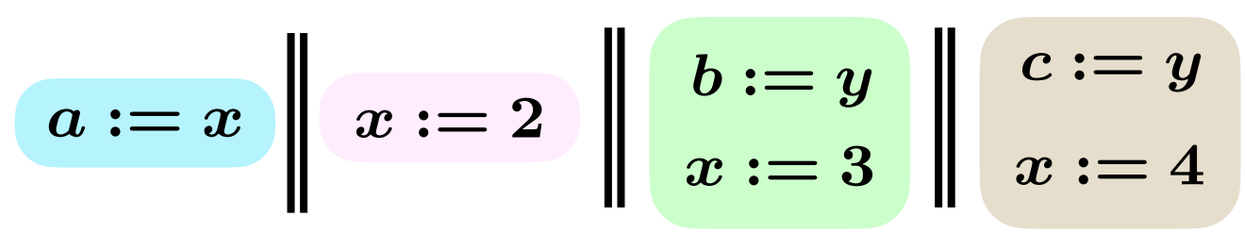


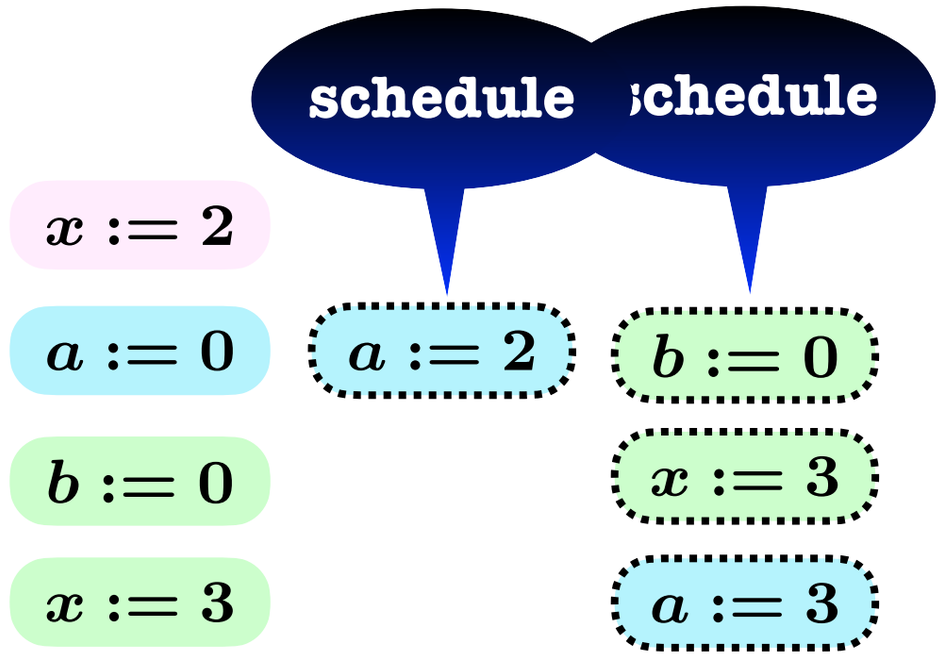




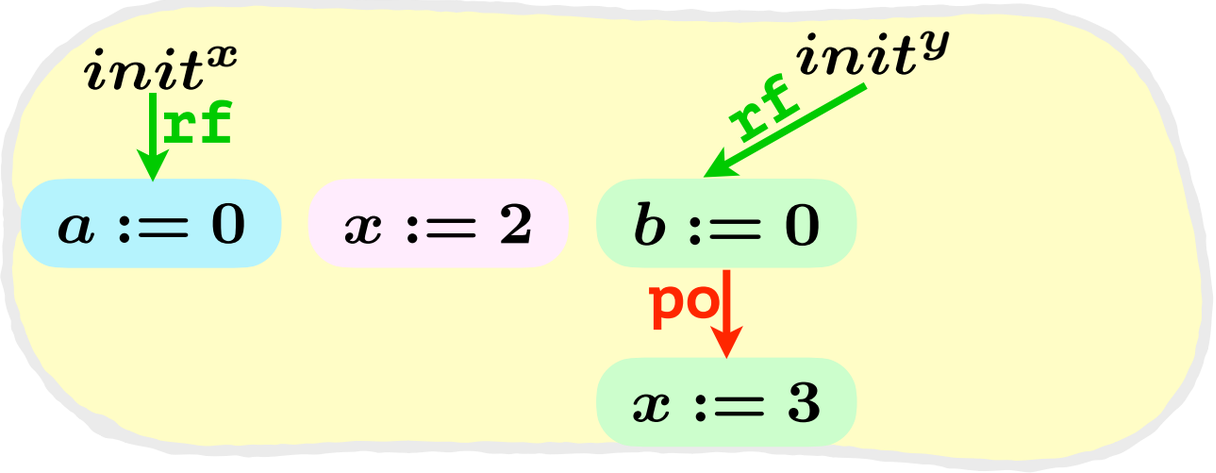
Branch **&** **Postpone**

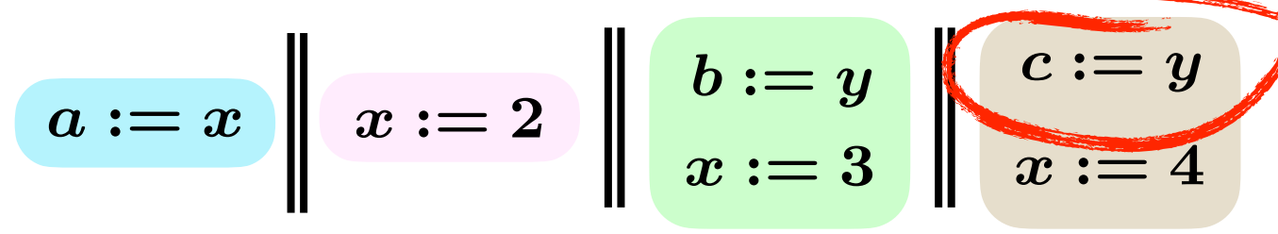


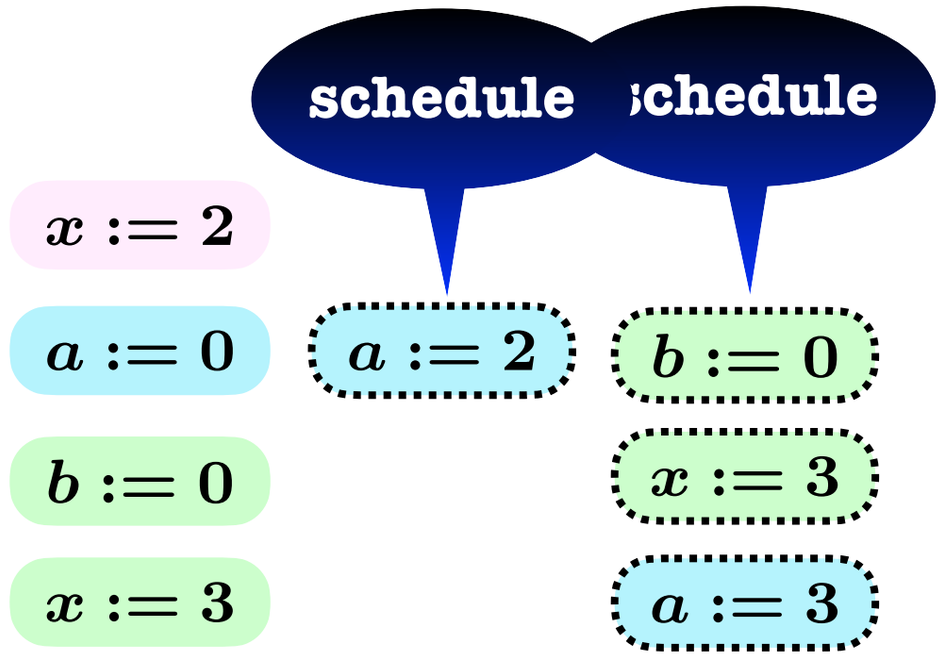




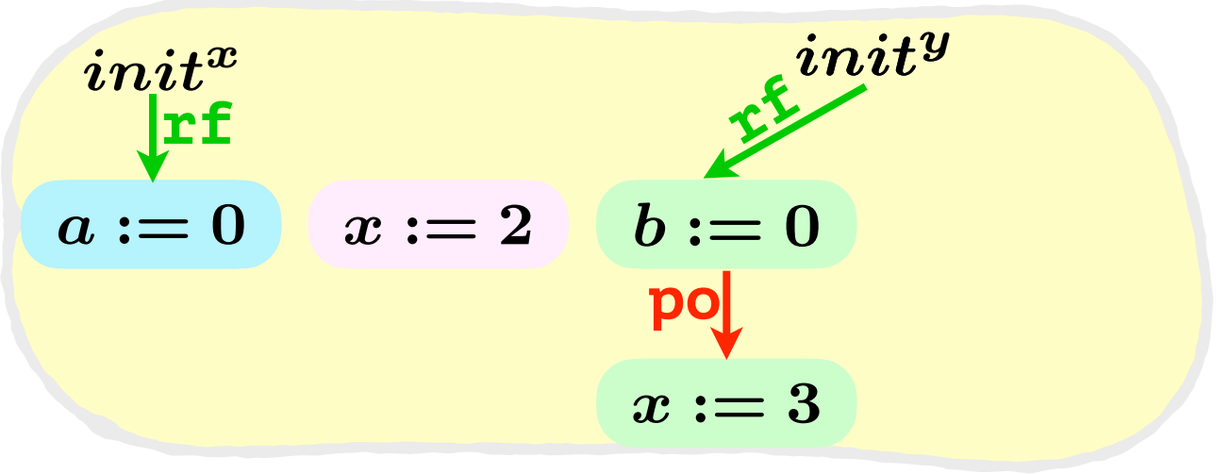
Branch **&** **Postpone**

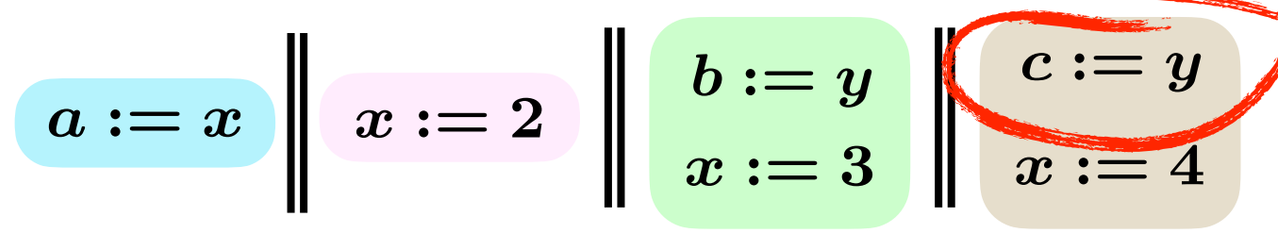


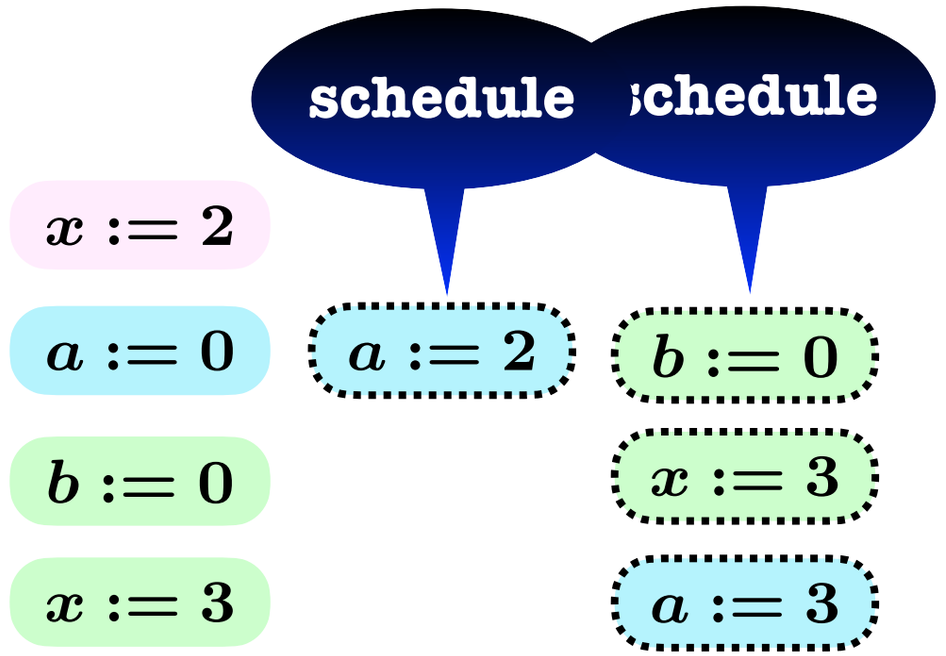




Branch **&** **Postpone**

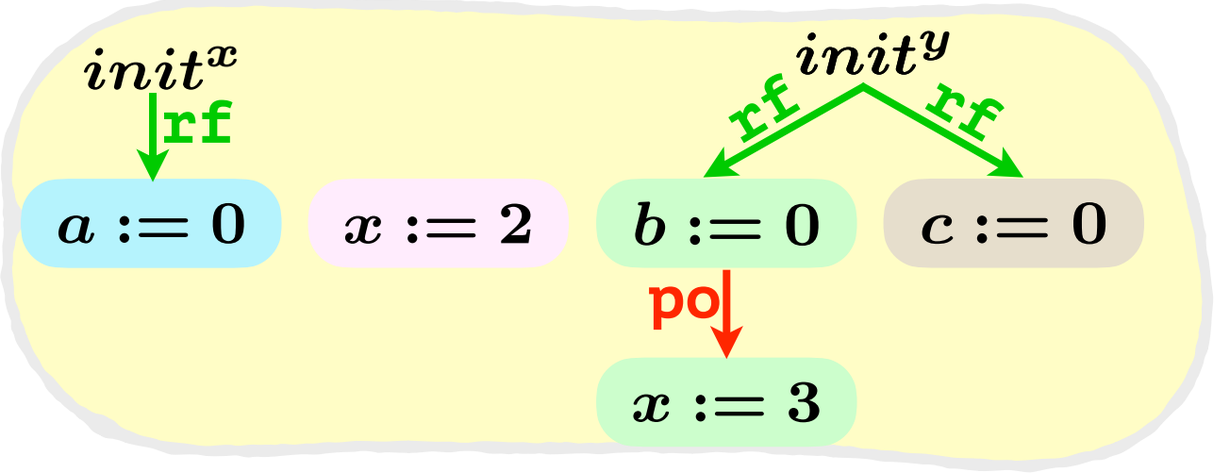


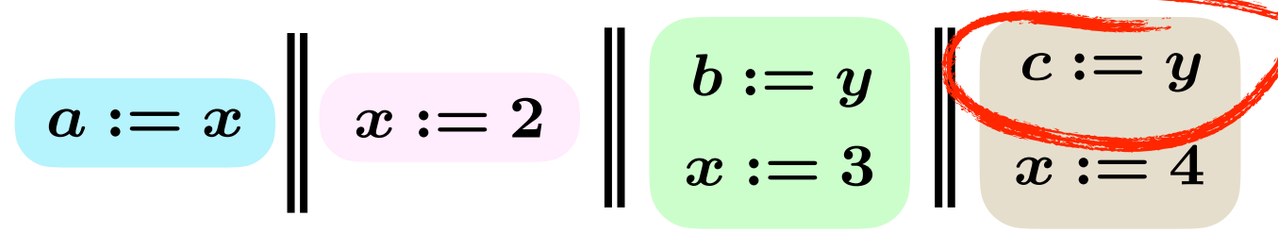


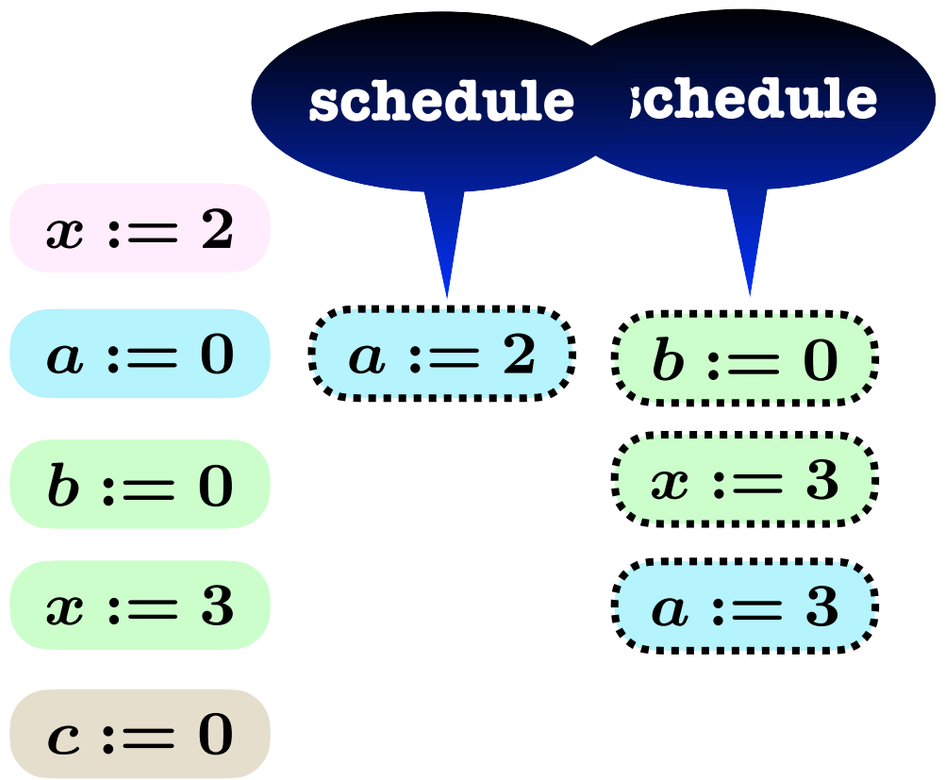


$c := 0$

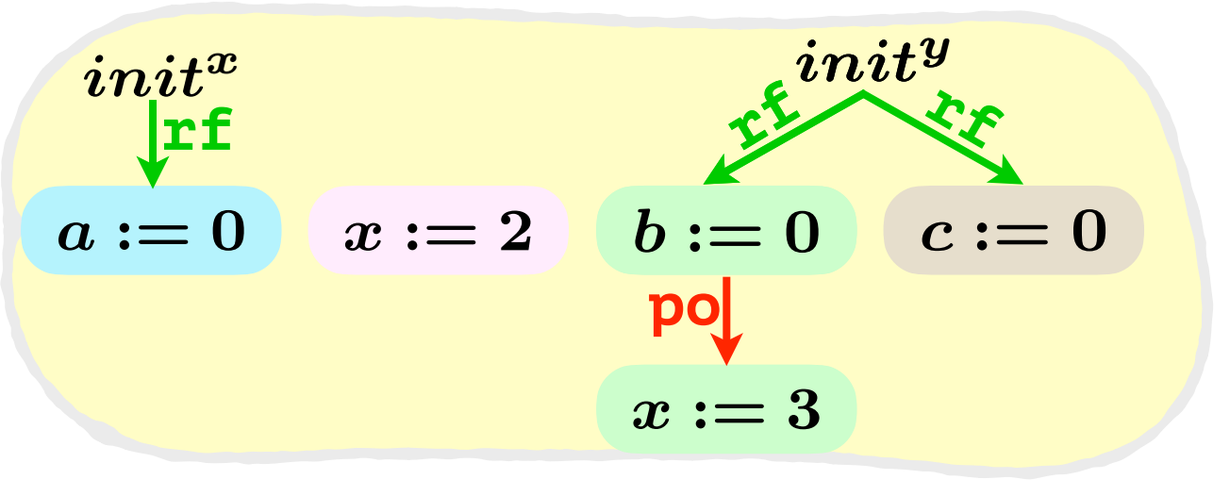
Branch & Postpone

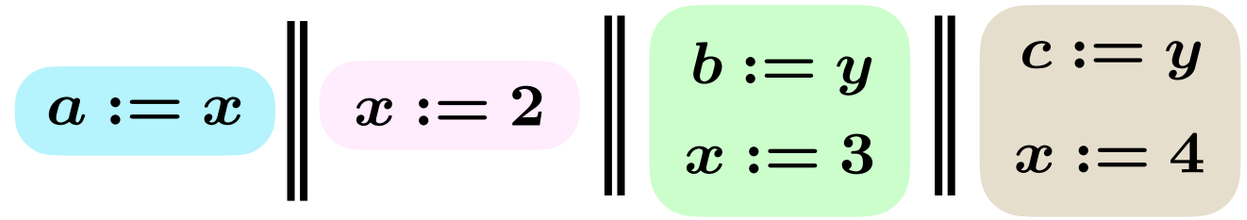


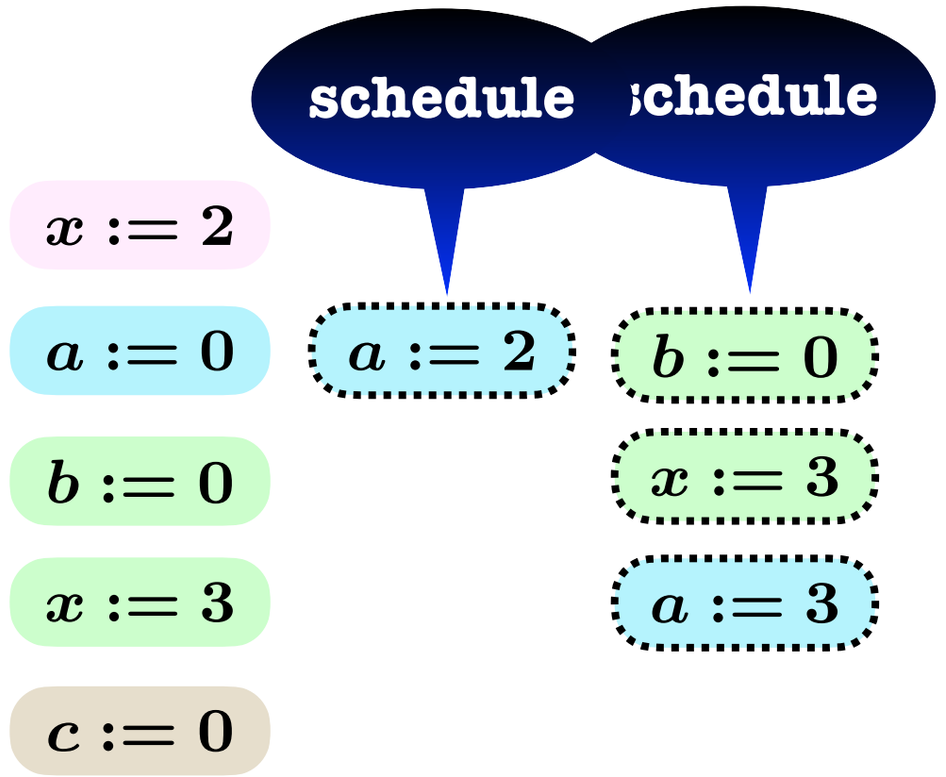




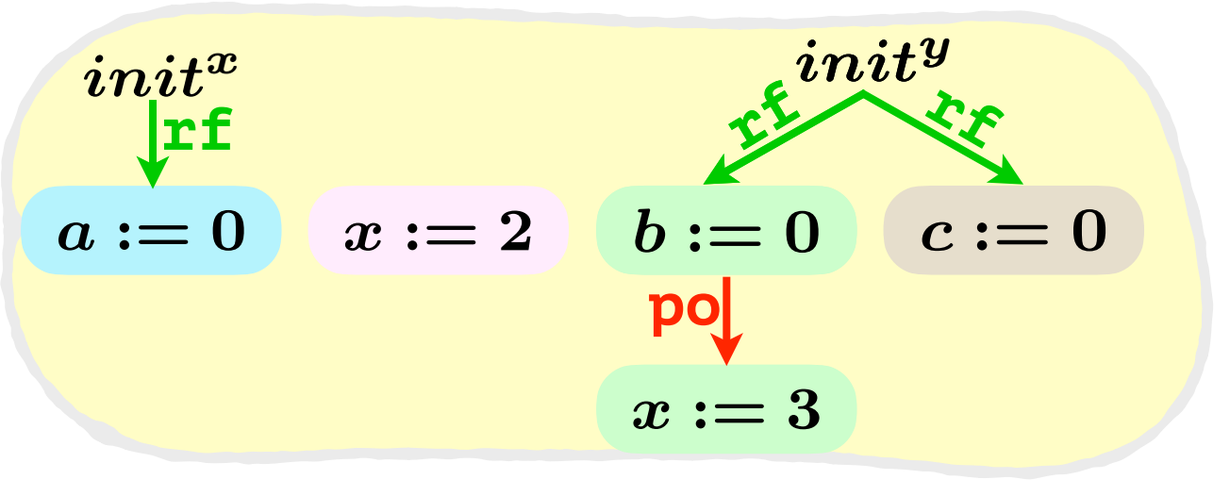
Branch & Postpone

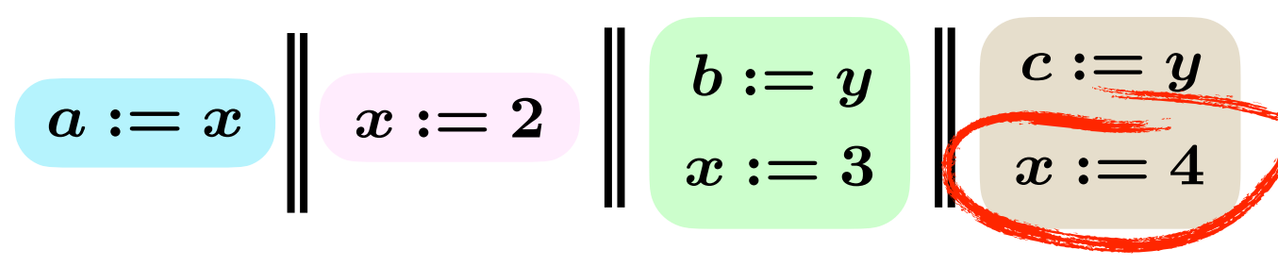


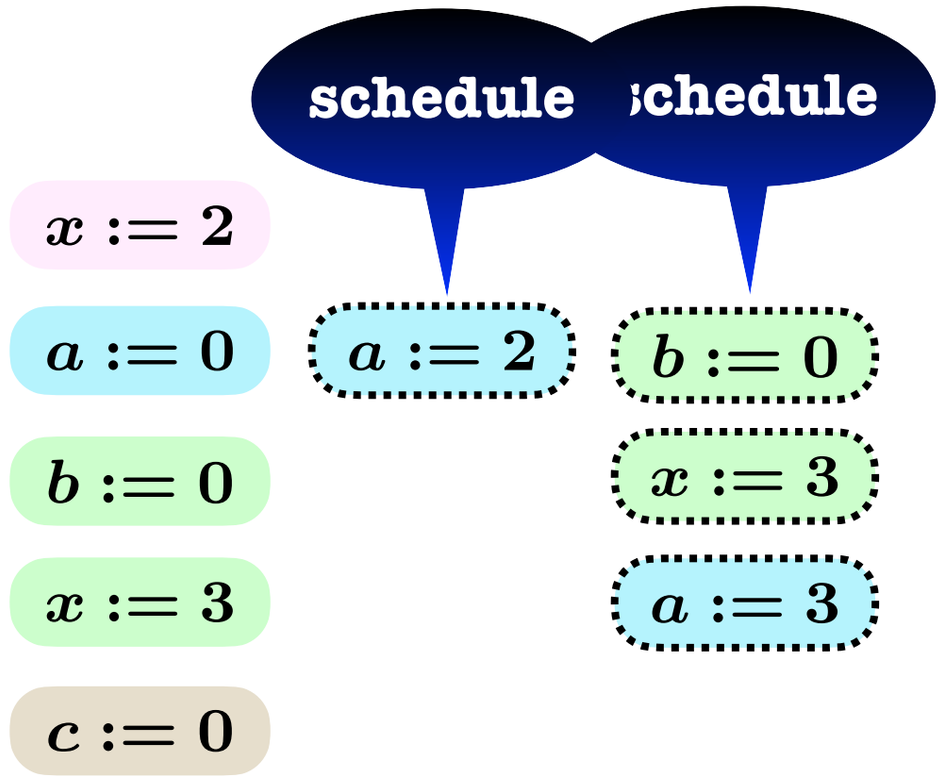




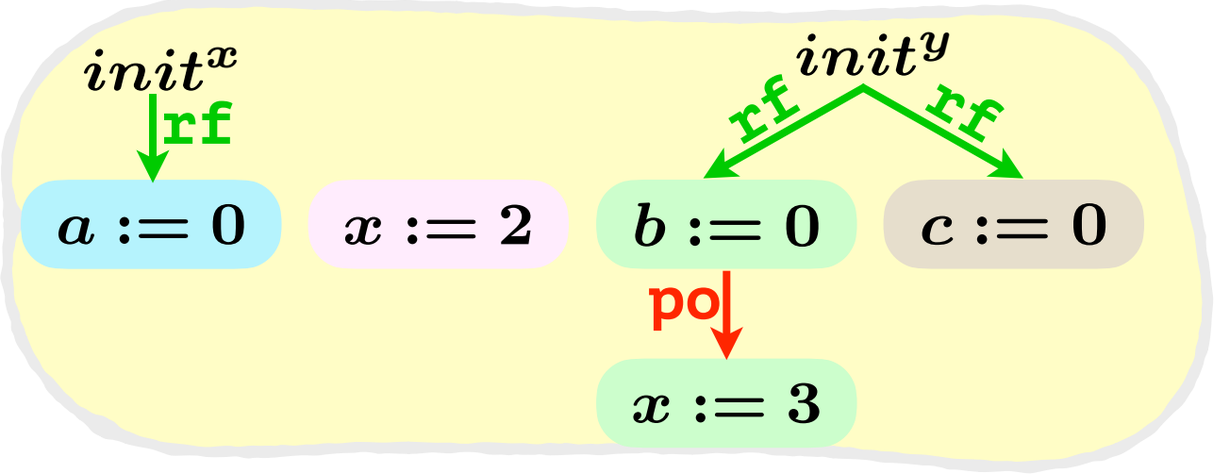
Branch & Postpone

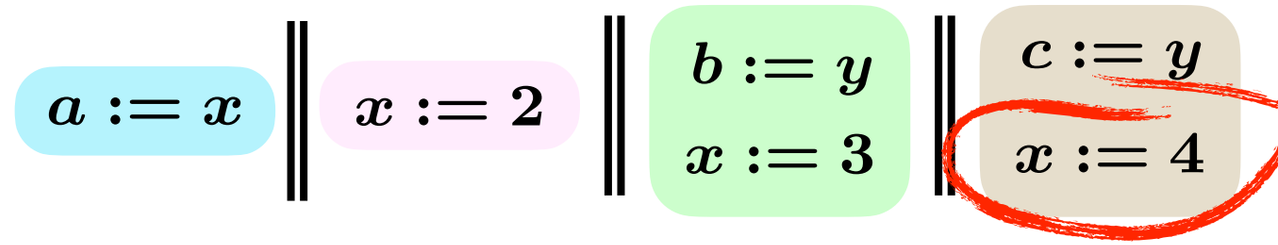


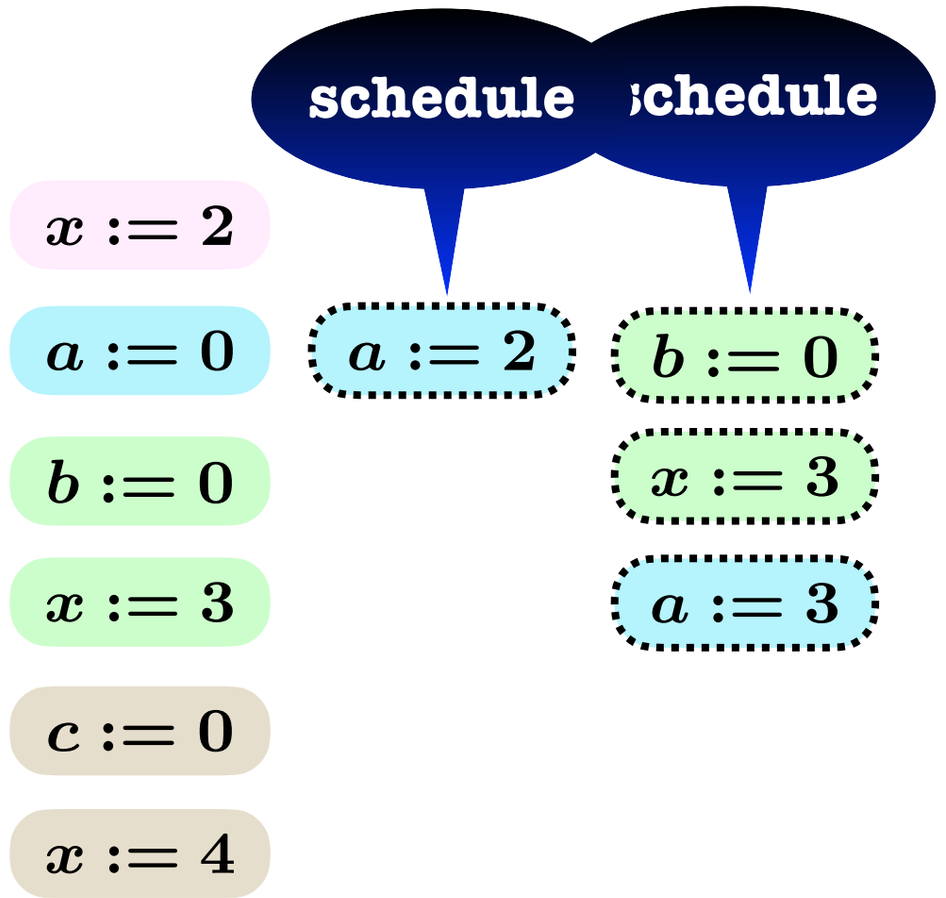




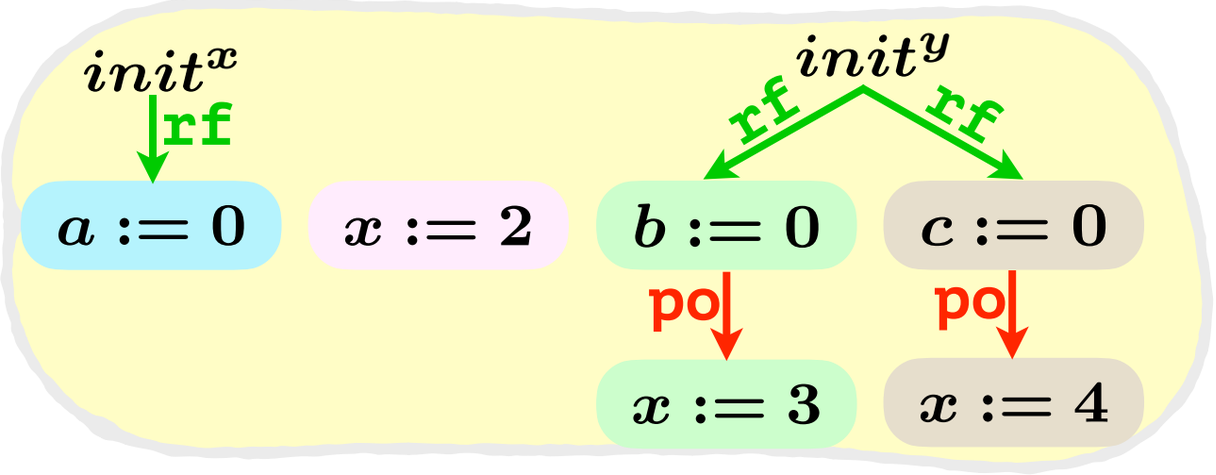
Branch & Postpone



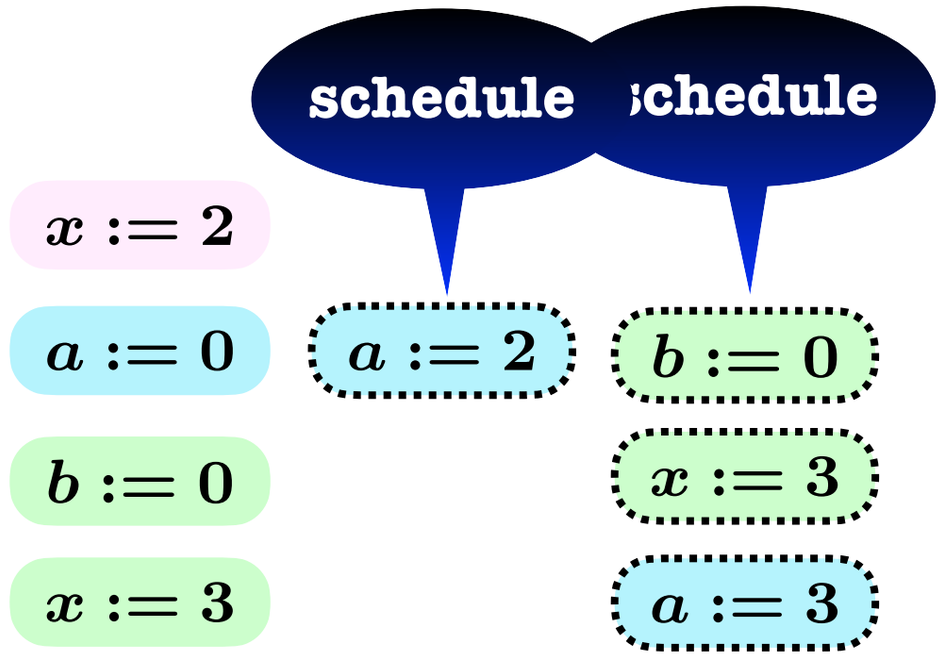




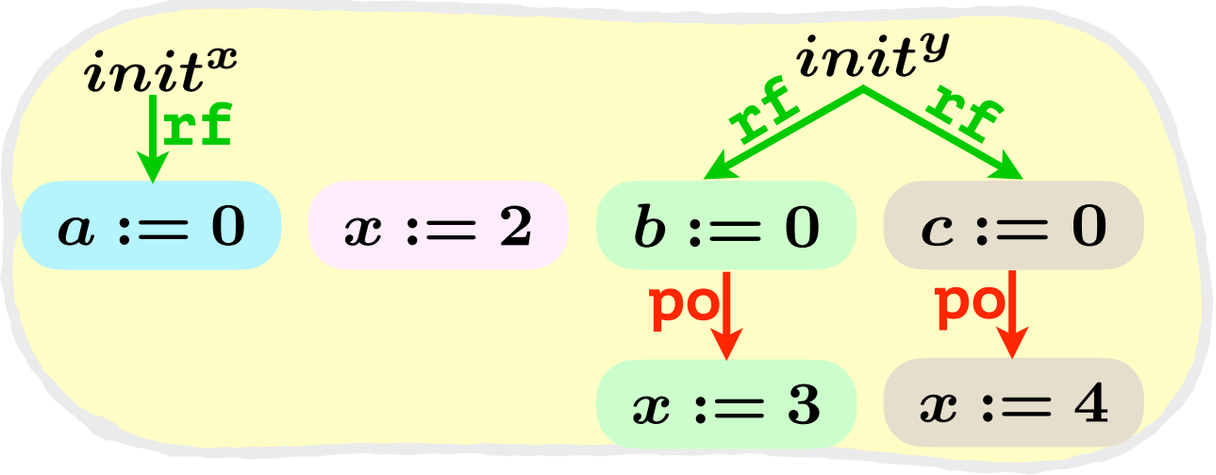
Branch & Postpone

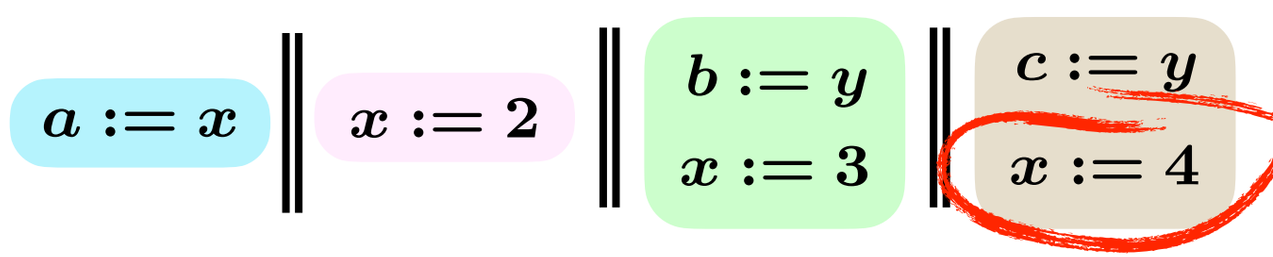


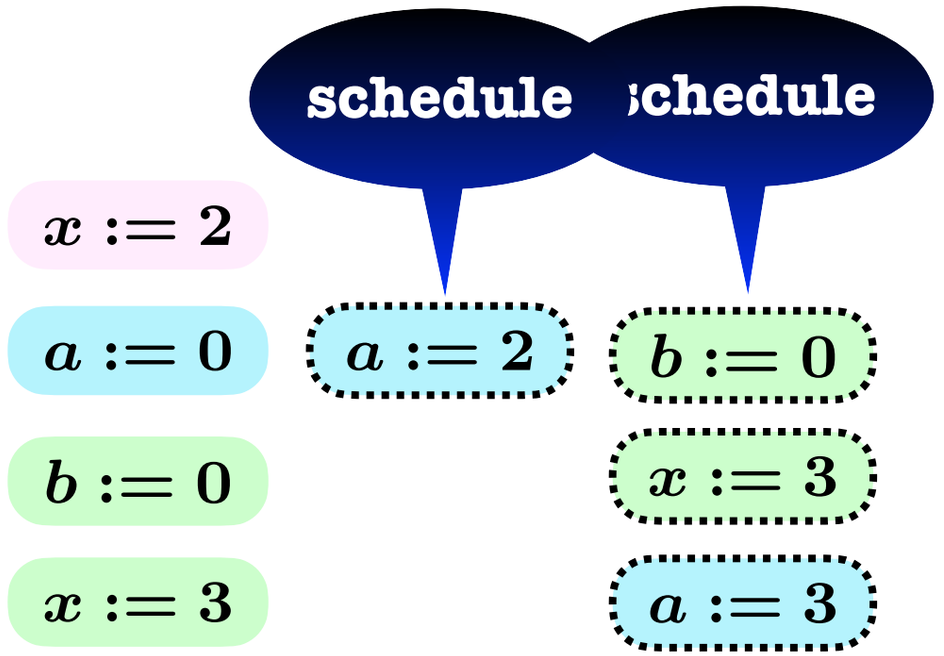
$a := x \parallel x := 2 \parallel b := y \; x := 3 \parallel c := y \; x := 4$



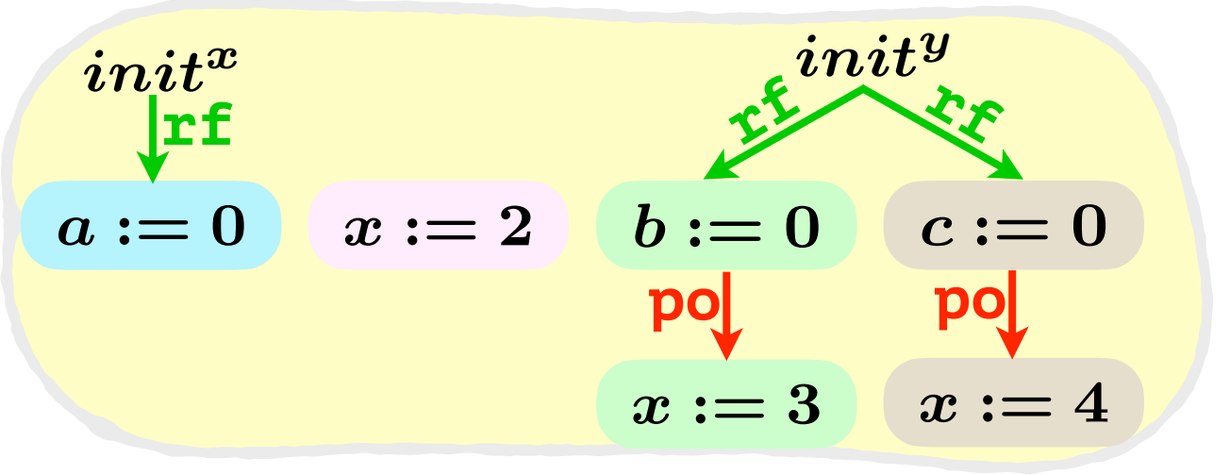
Branch & Postpone

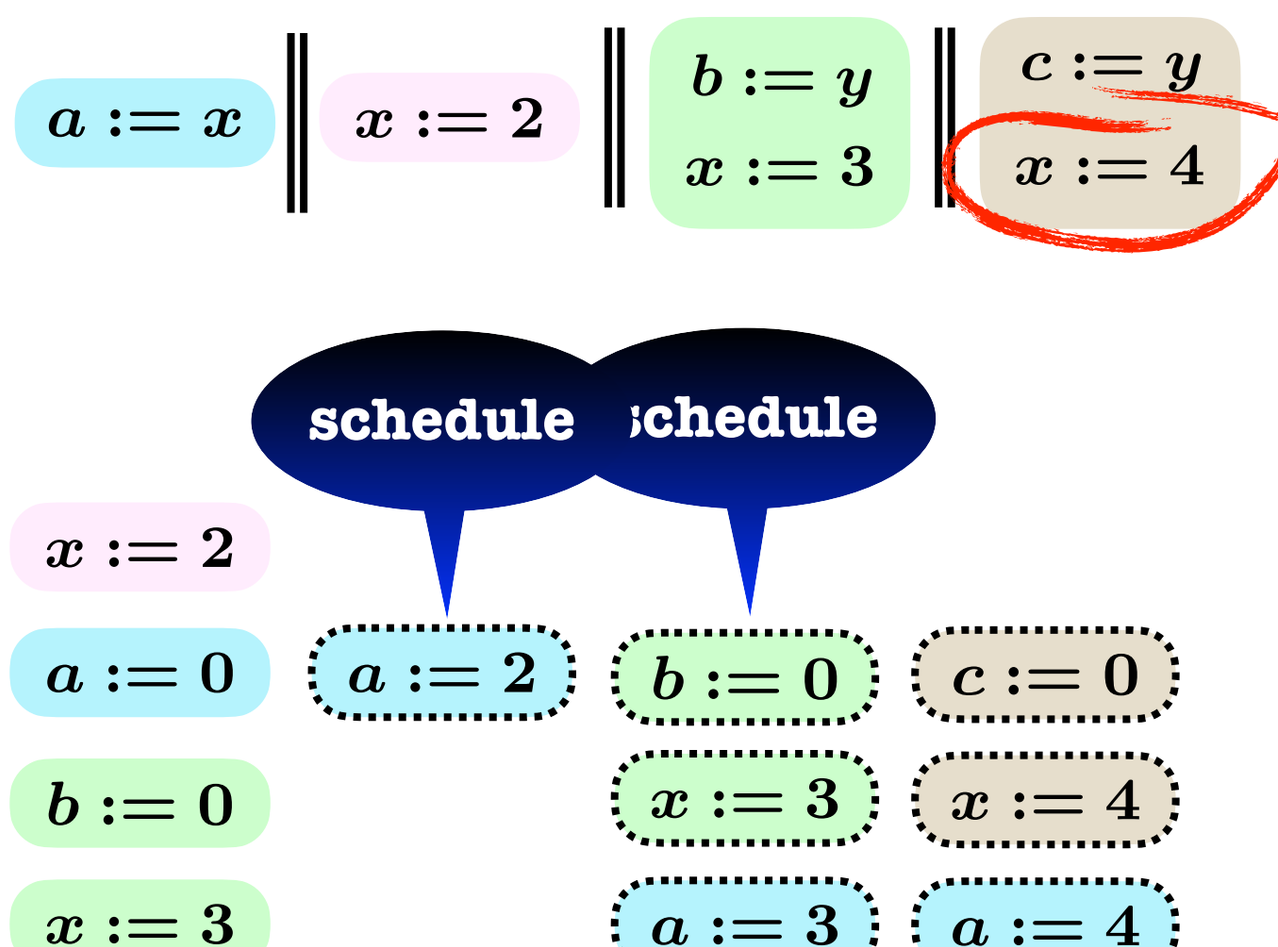




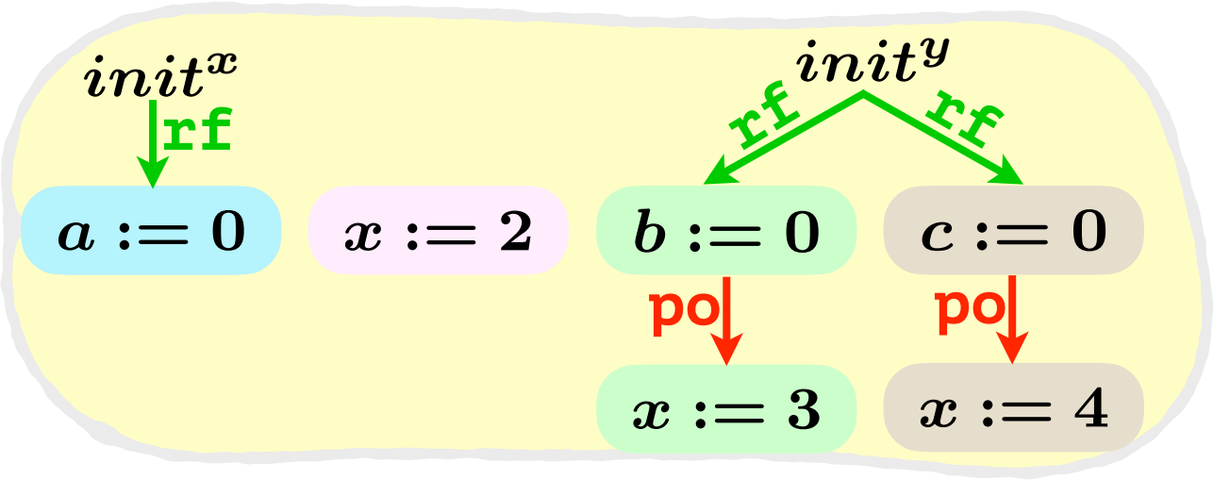


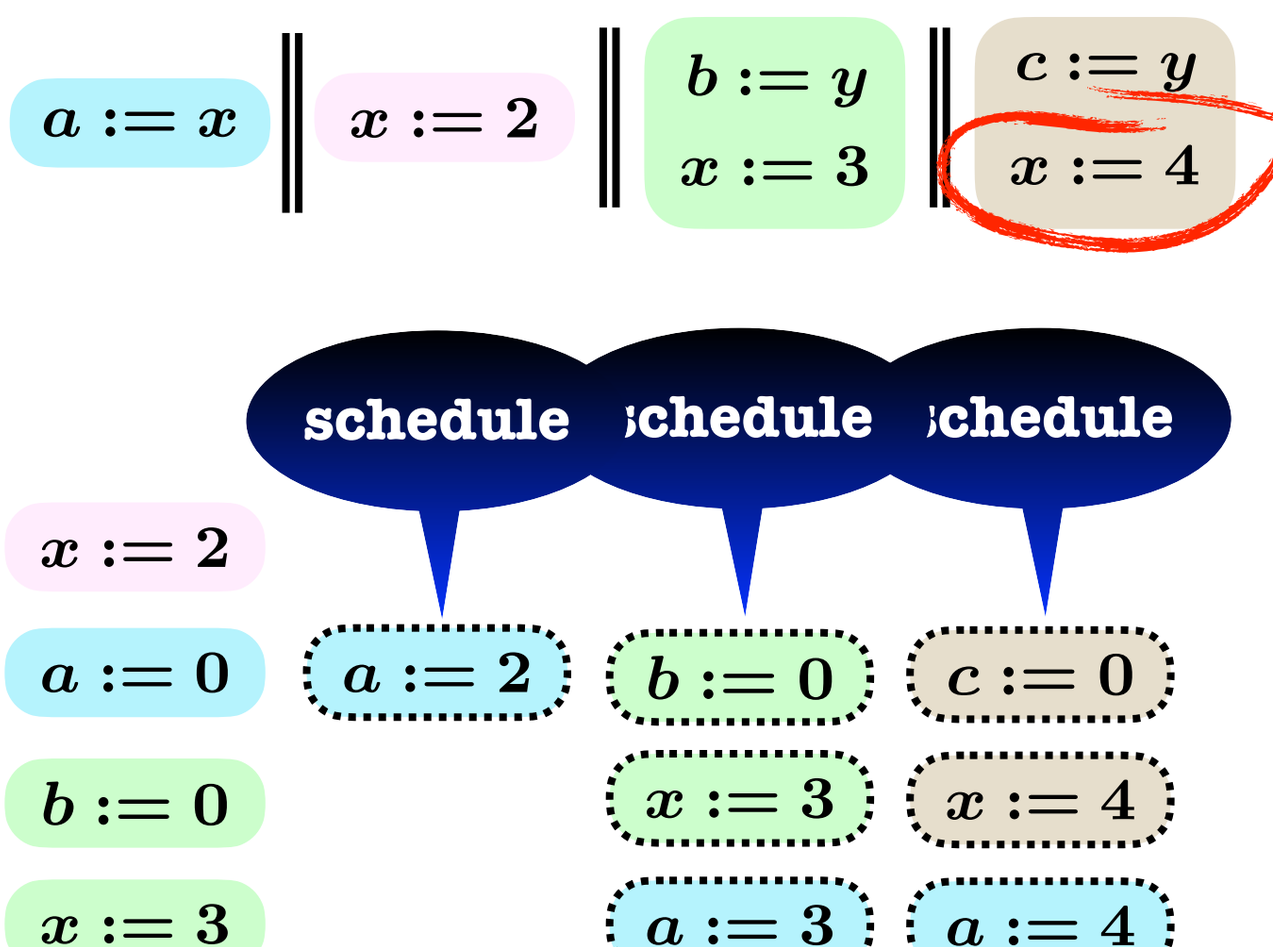
Branch & Postpone



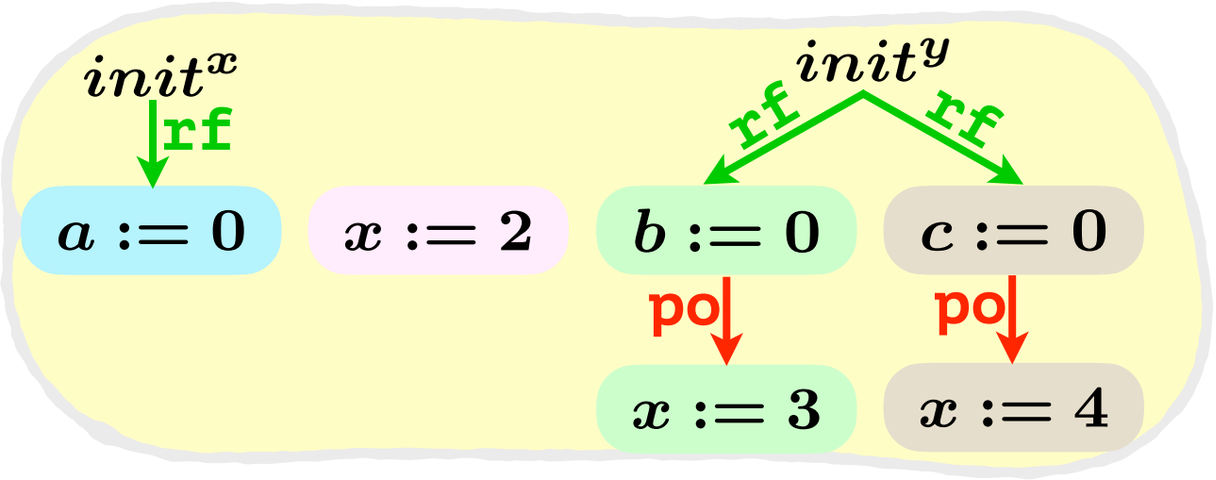


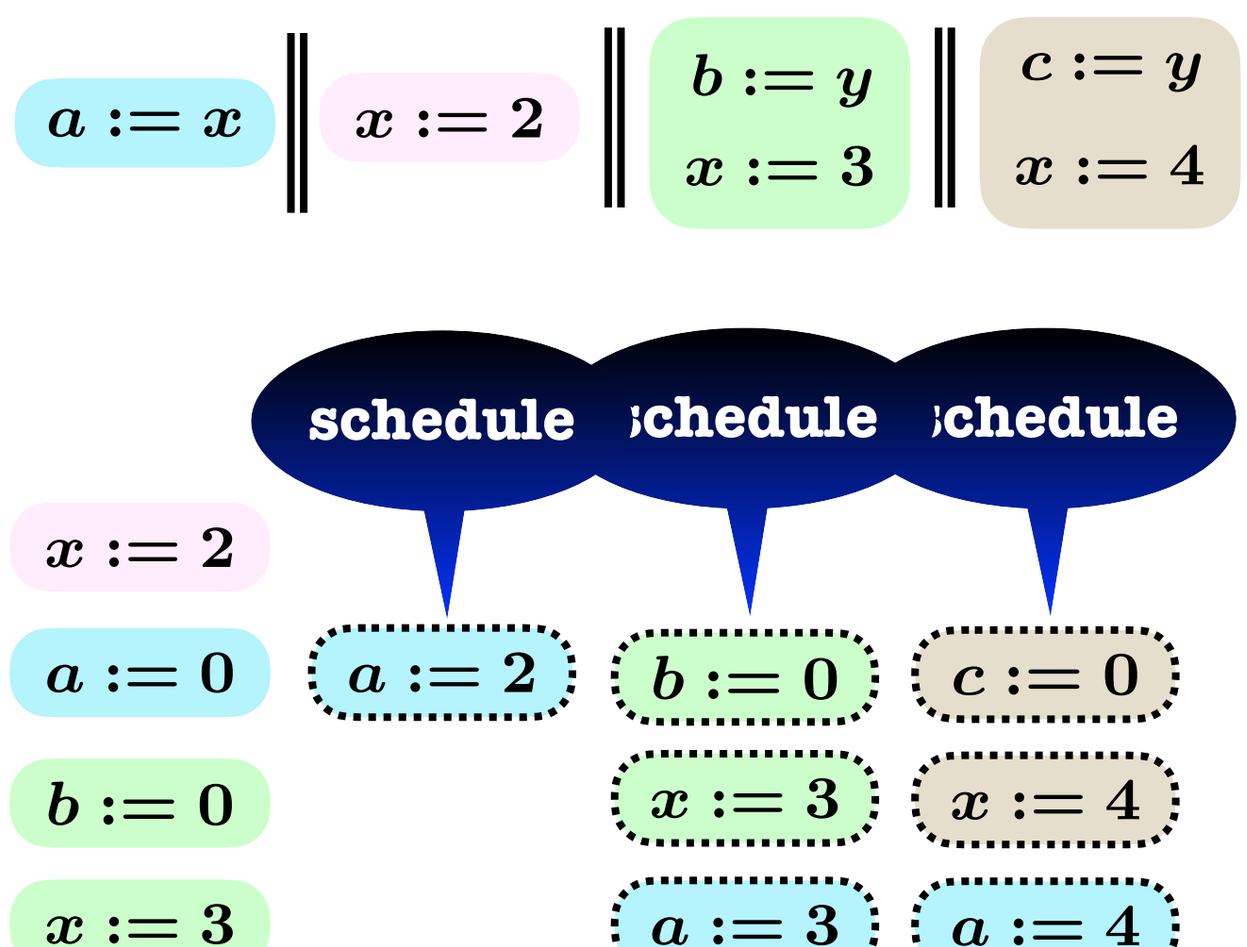
Branch **&** **Postpone**



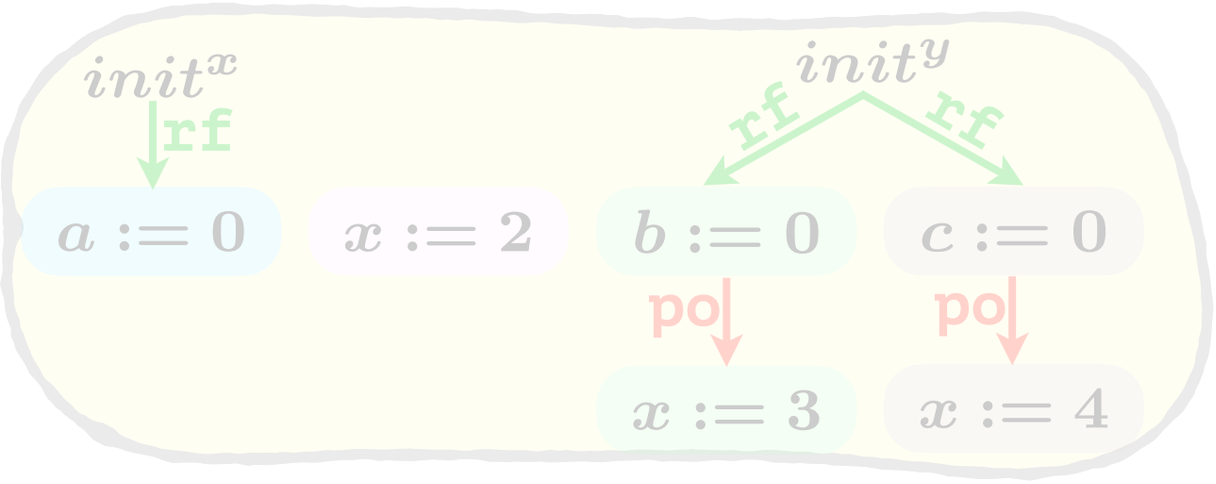


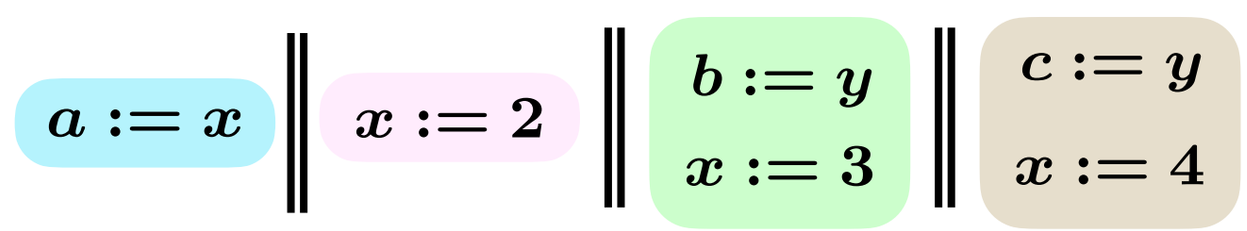
Branch **&** **Postpone**

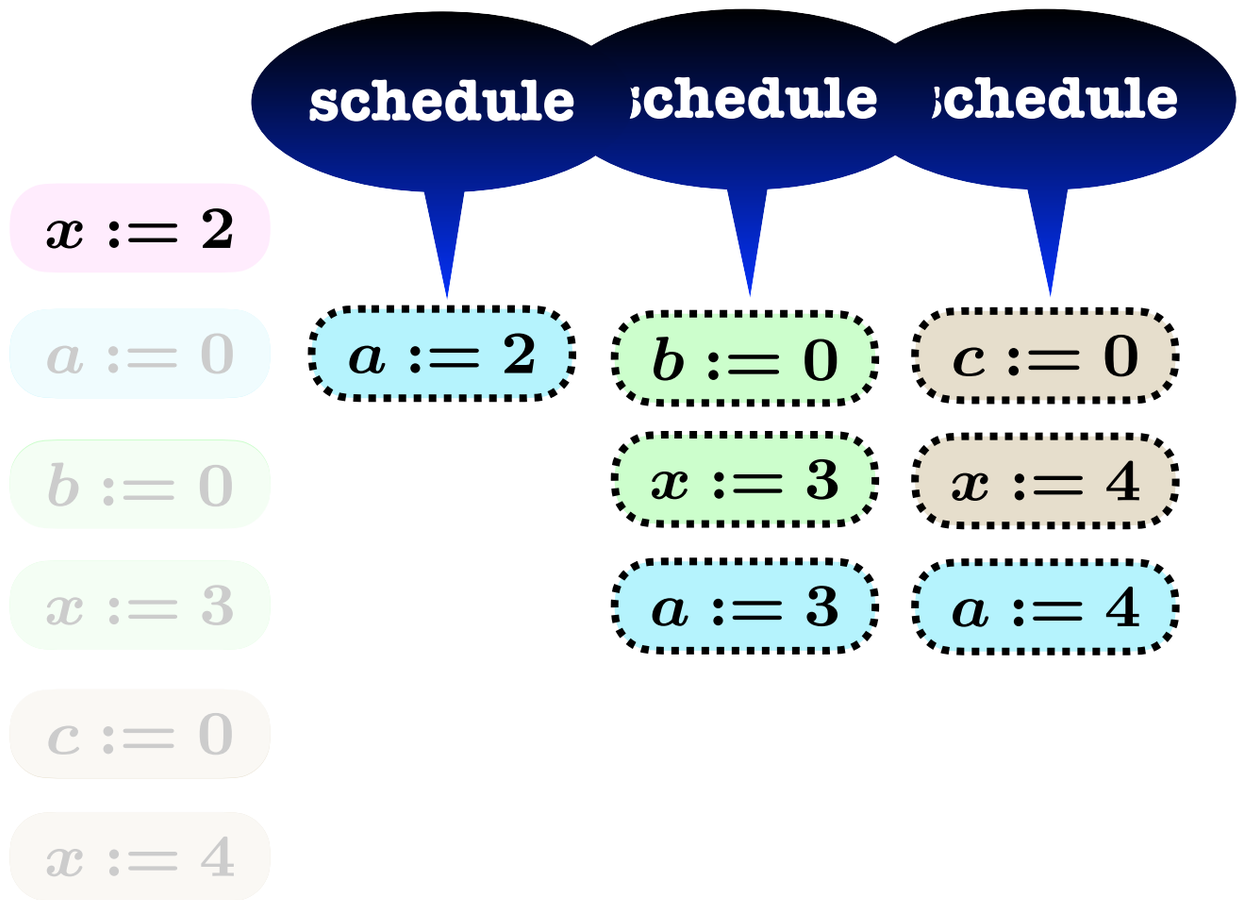




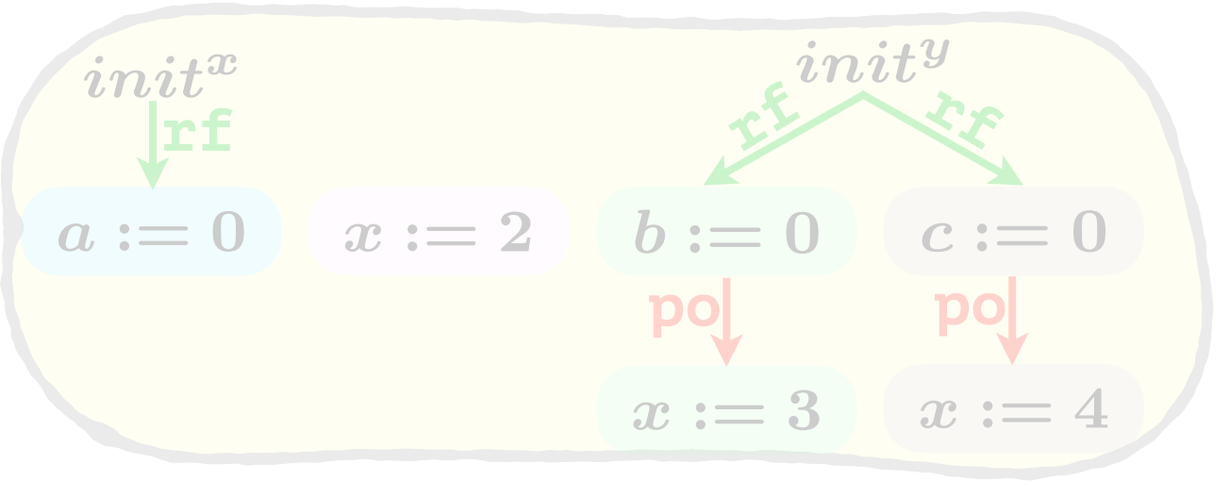
Branch **&** **Postpone**

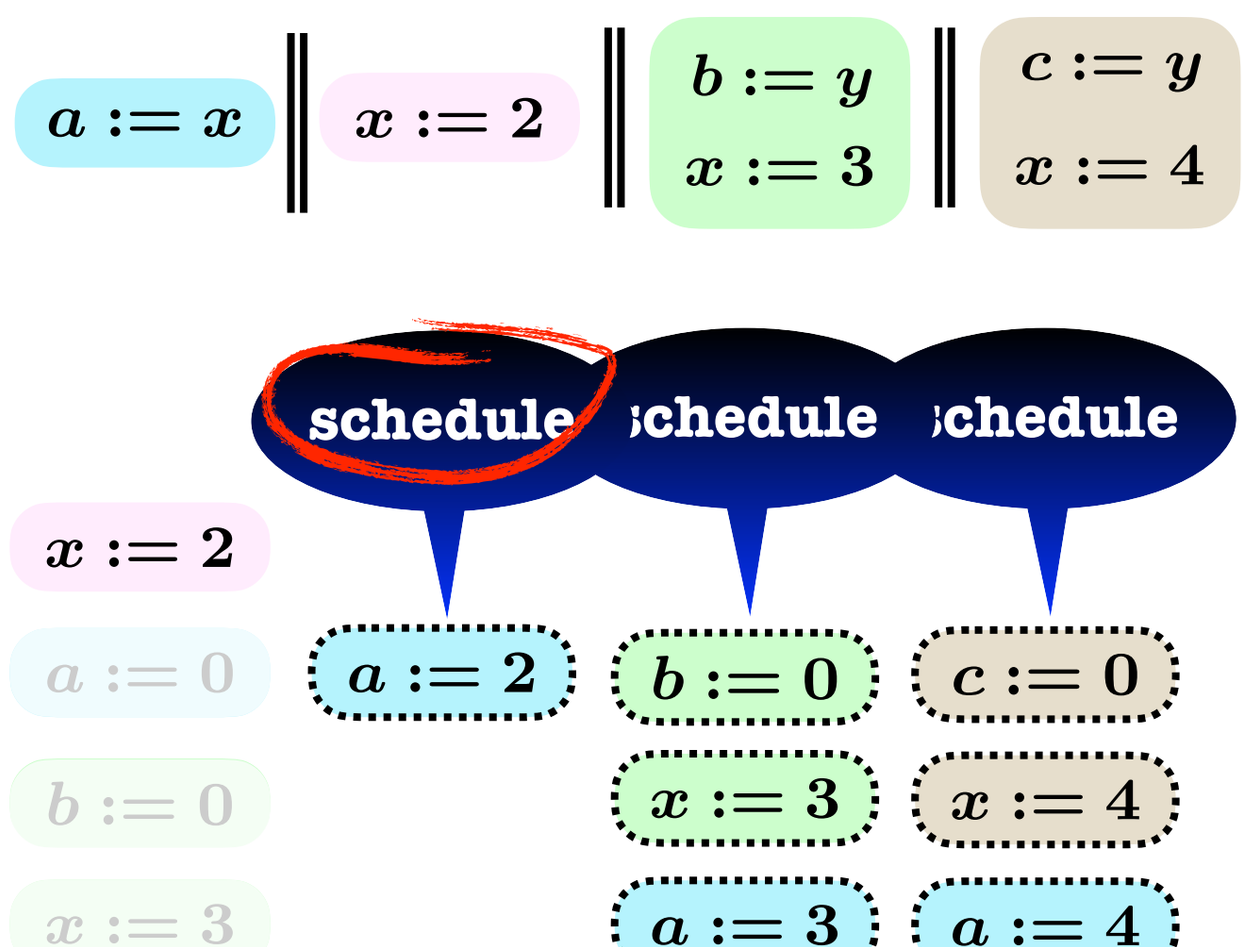




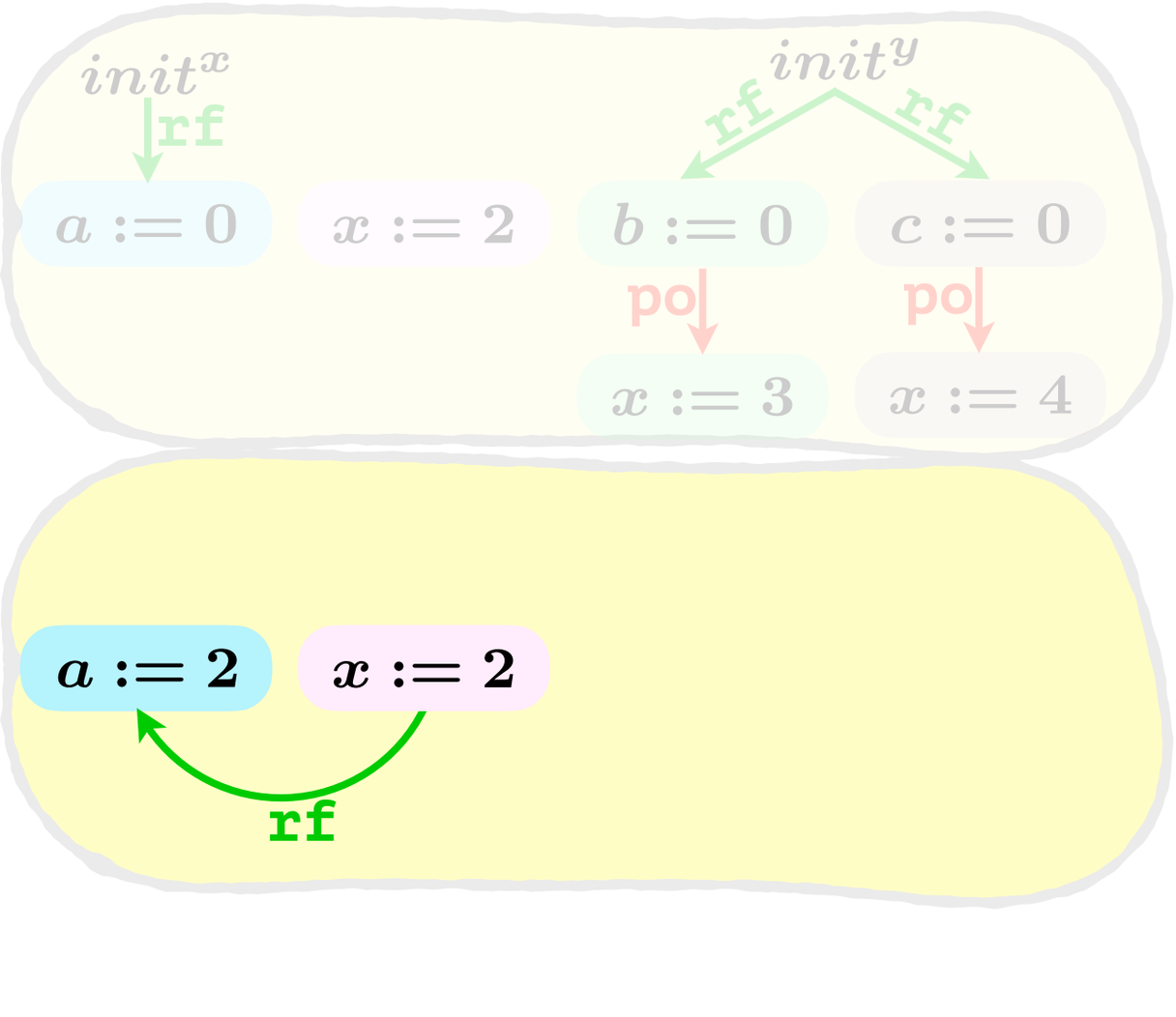


Branch & Postpone

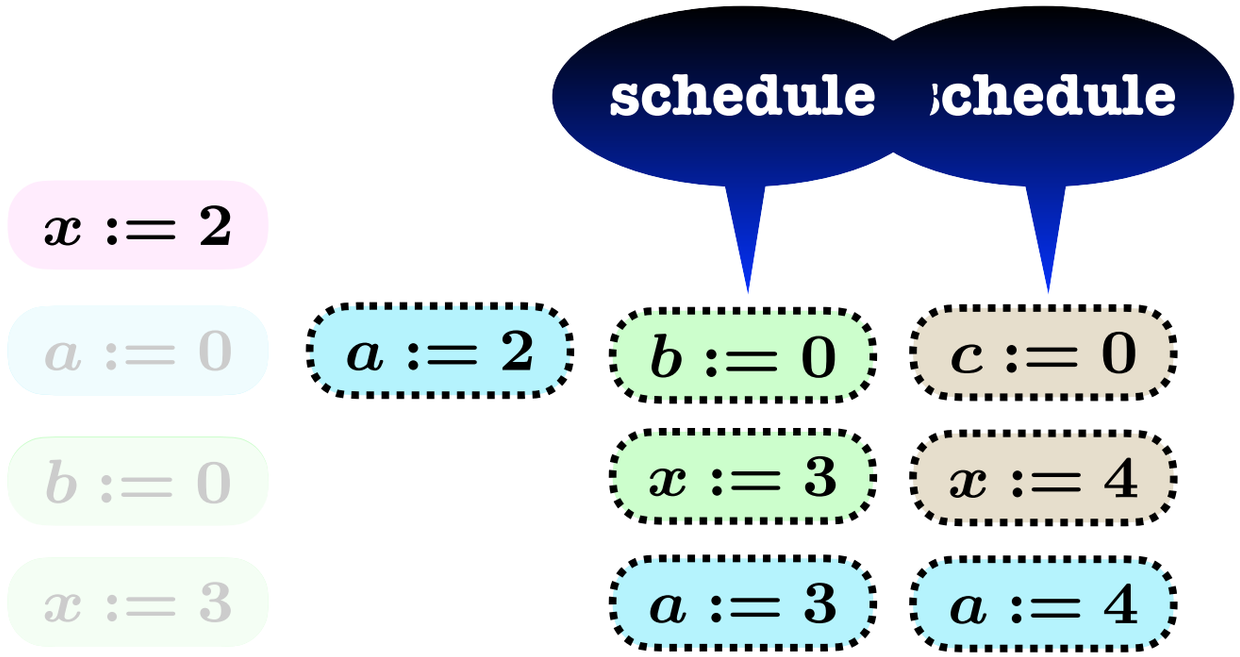




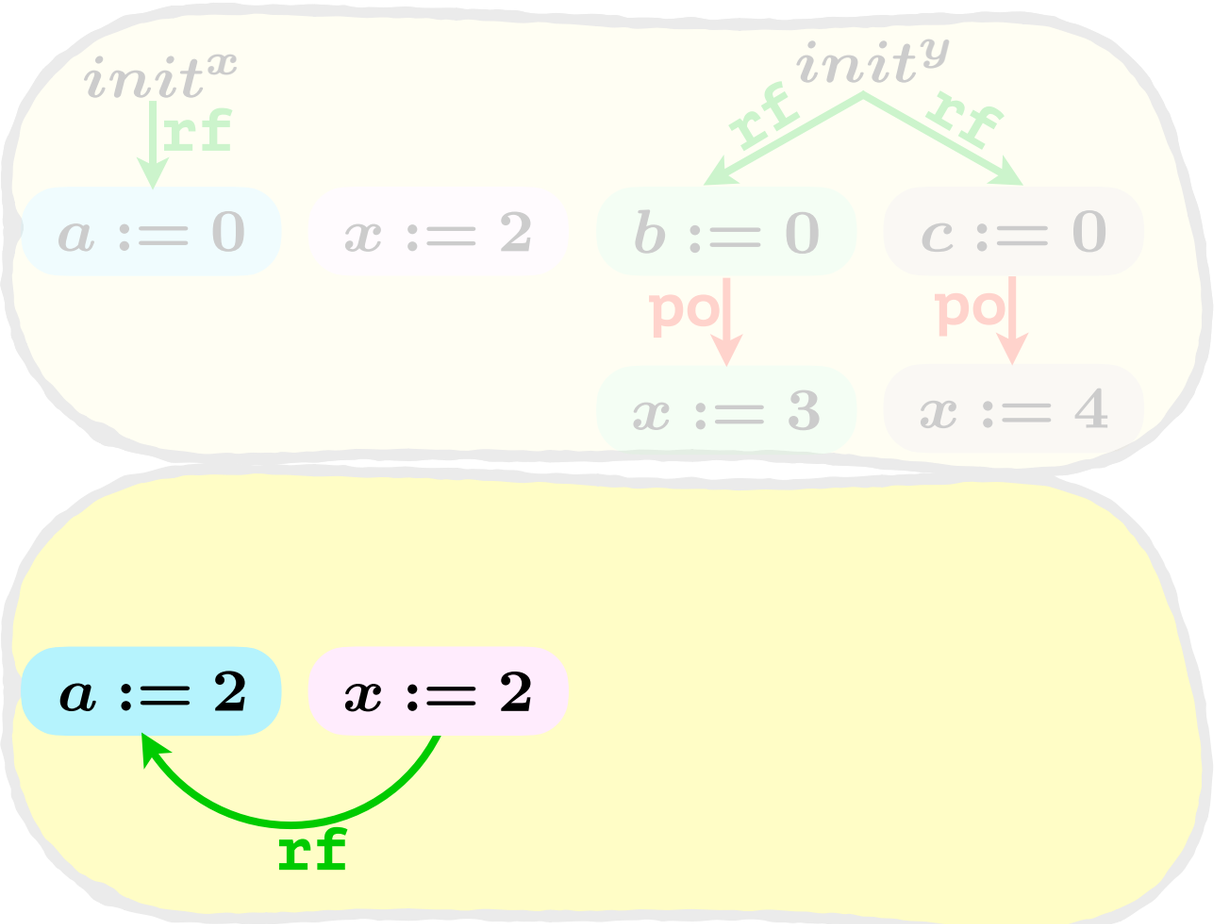
Branch **&** **Postpone**



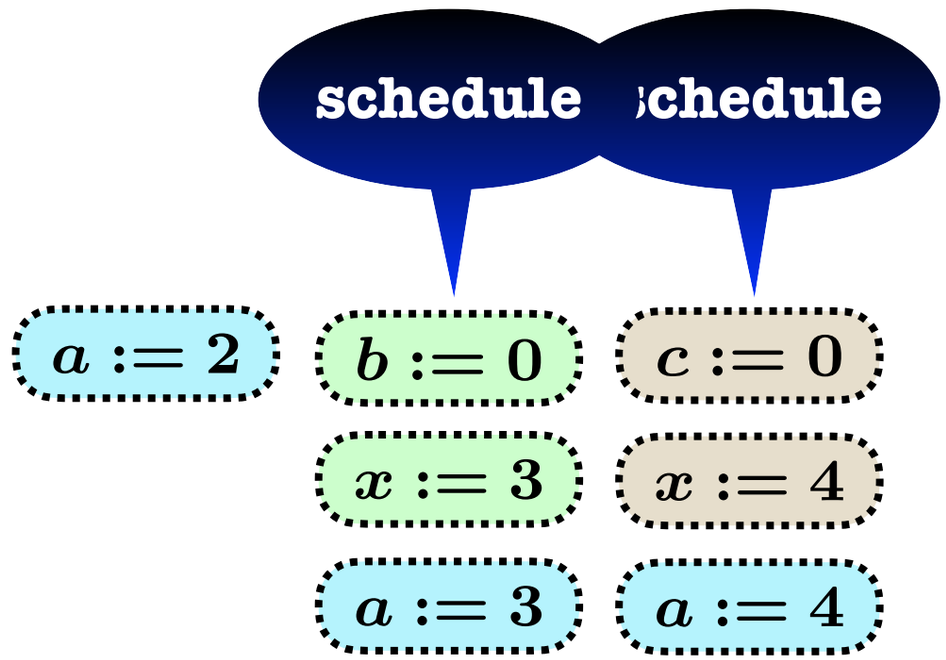
$a := x \parallel x := 2 \parallel b := y \; x := 3 \parallel c := y \; x := 4$



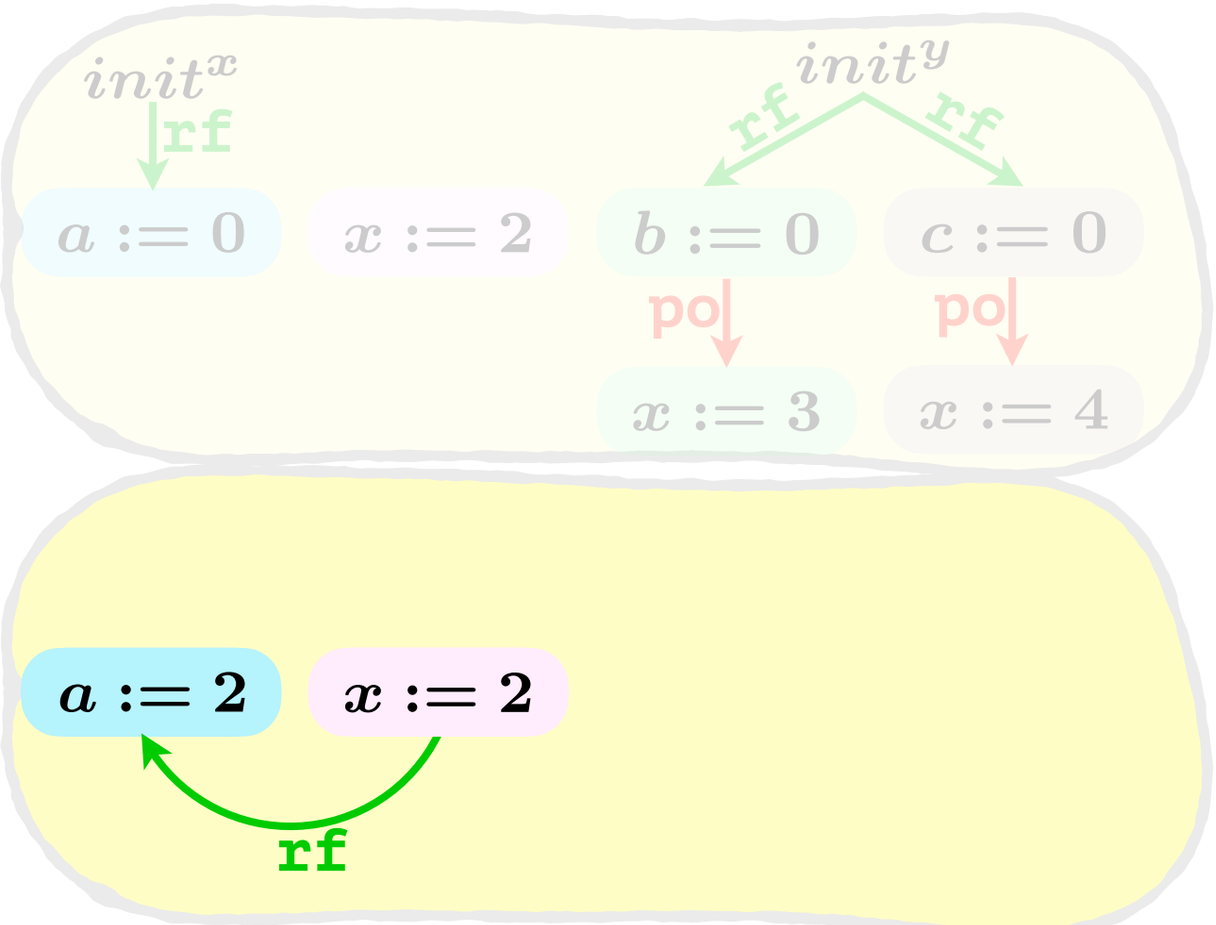
Branch & Postpone

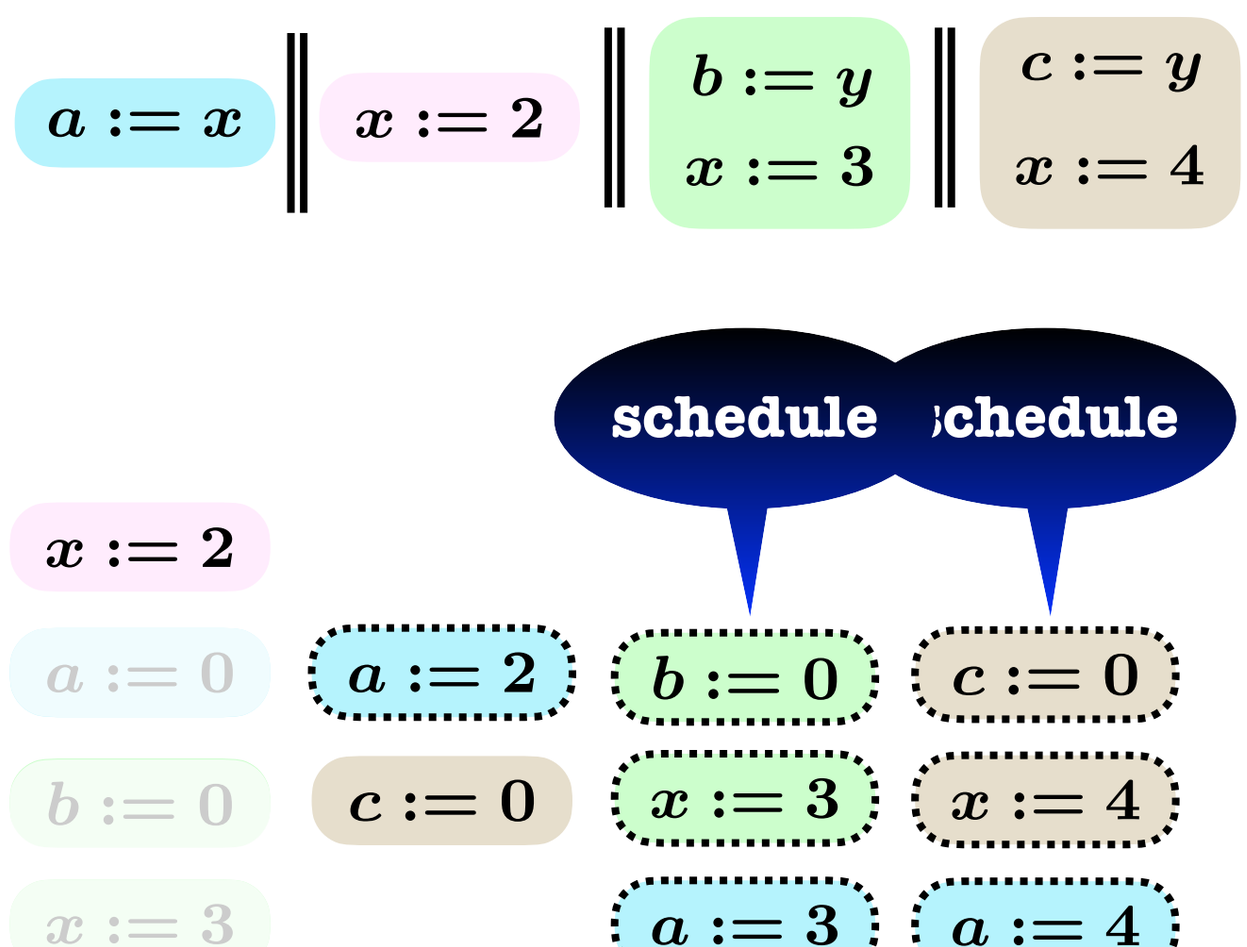


$a := x \parallel x := 2 \parallel b := y \; x := 3 \parallel c := y \; x := 4$

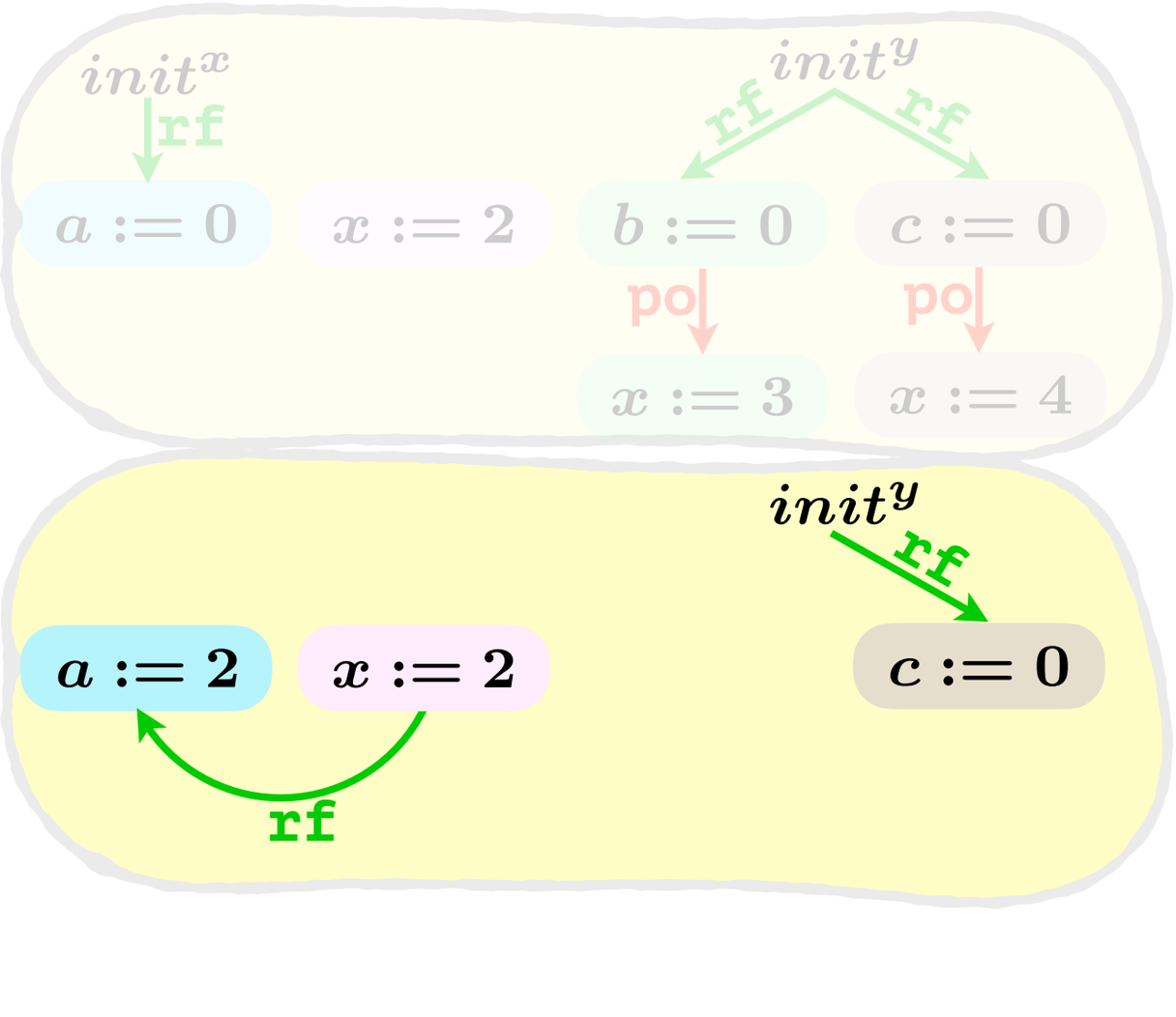


Branch & Postpone





Branch **&** **Postpone**



$a := x \parallel x := 2 \parallel b := y \; x := 3 \parallel c := y \; x := 4$

schedule schedule

$x := 2$

$a := 0$ $a := 2$ $b := 0$ $c := 0$

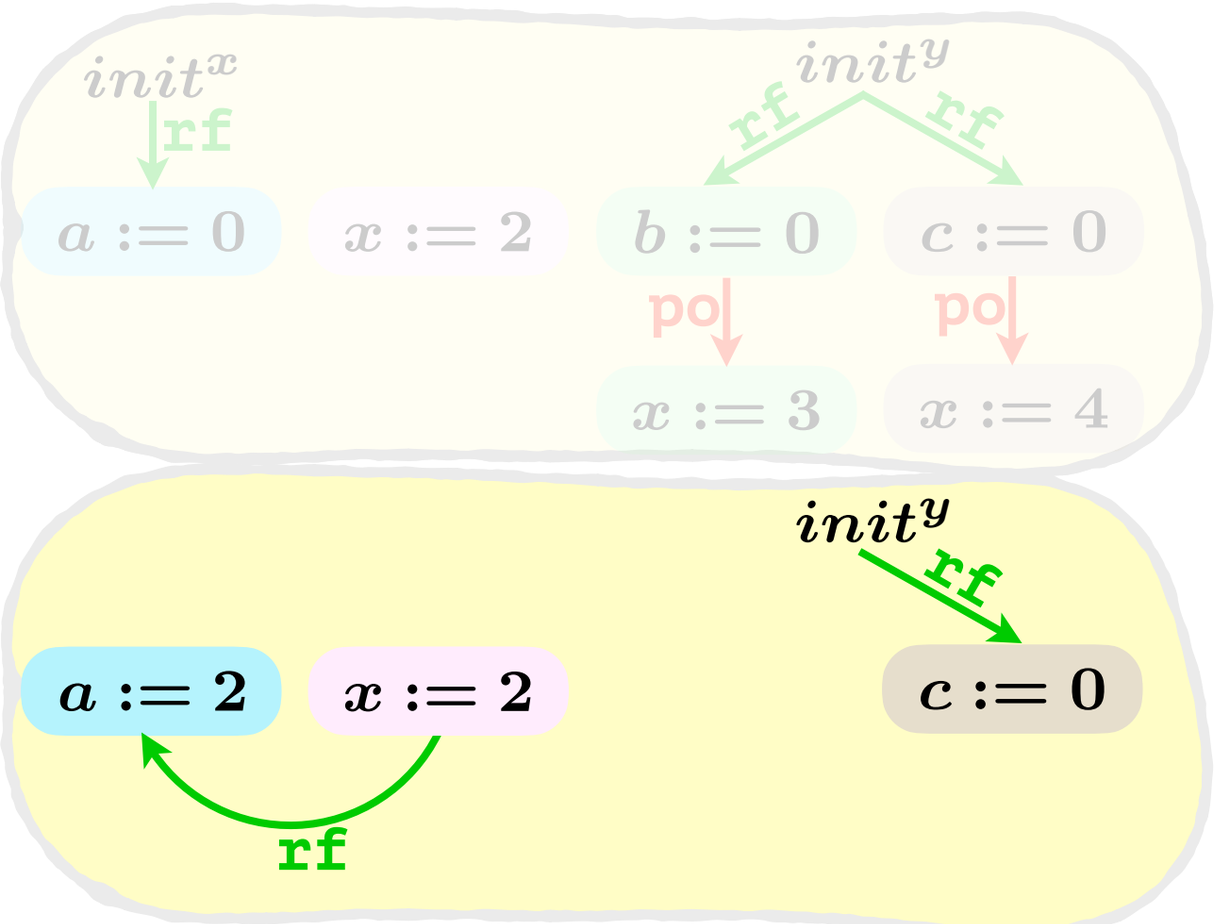
$b := 0$ $c := 0$ $x := 3$ $x := 4$

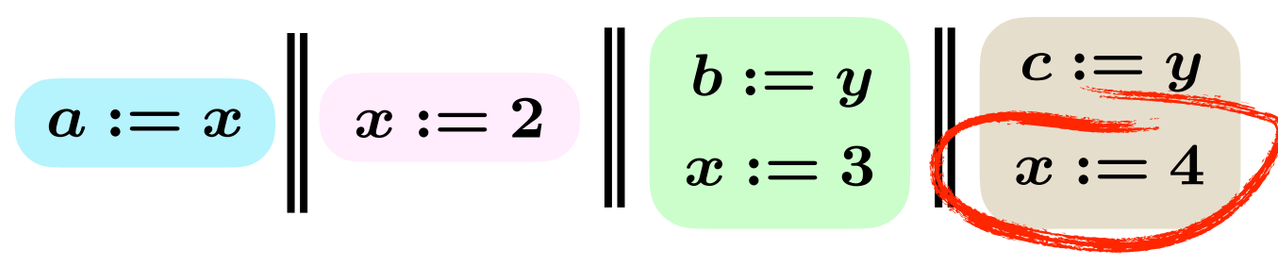
$x := 3$ $a := 3$ $a := 4$

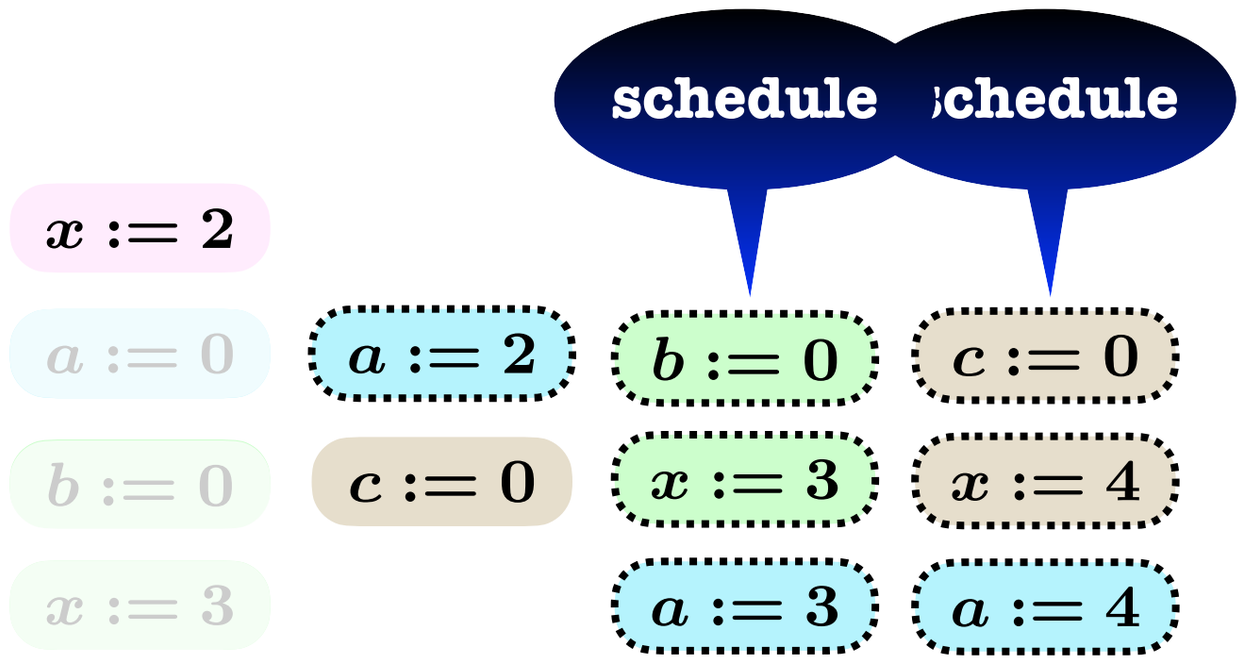
$c := 0$

$x := 4$

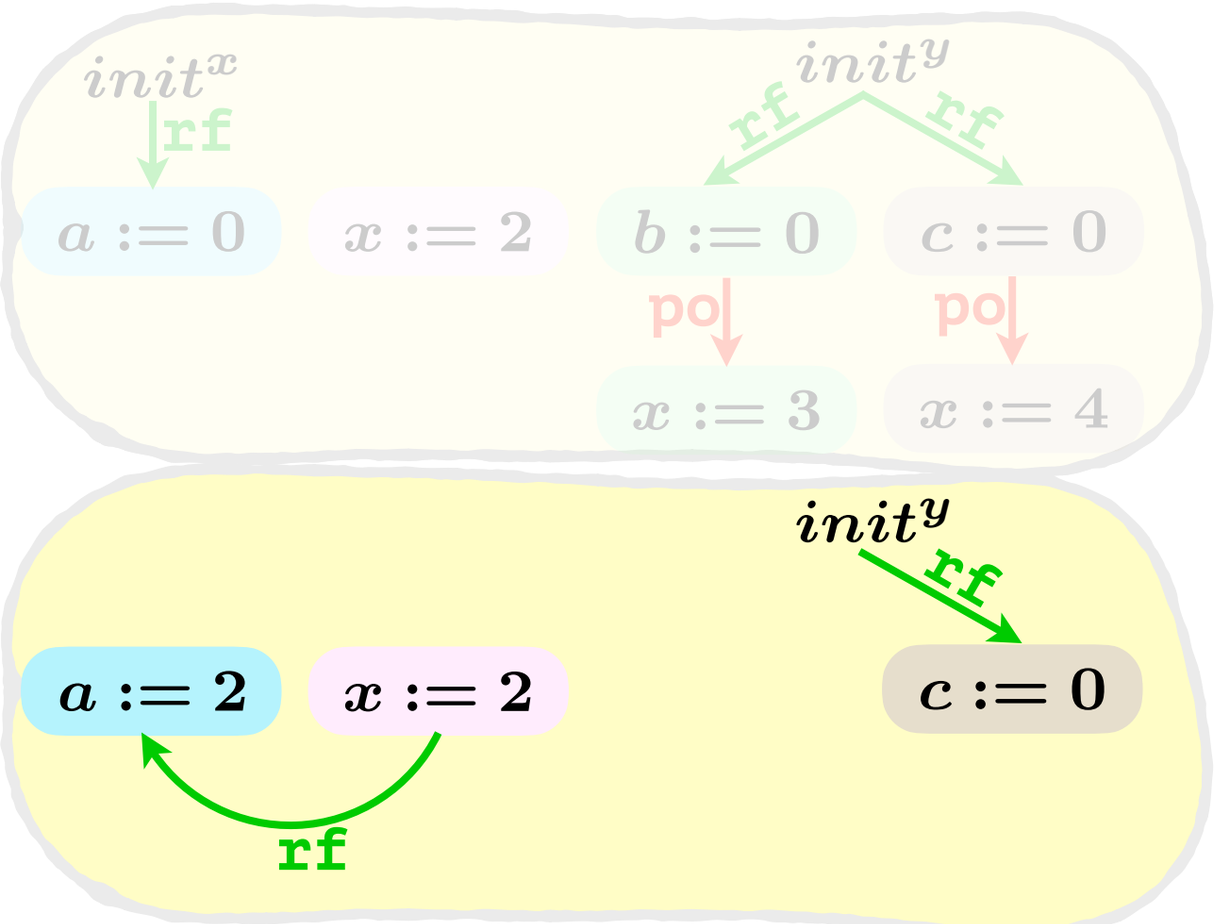
Branch & Postpone

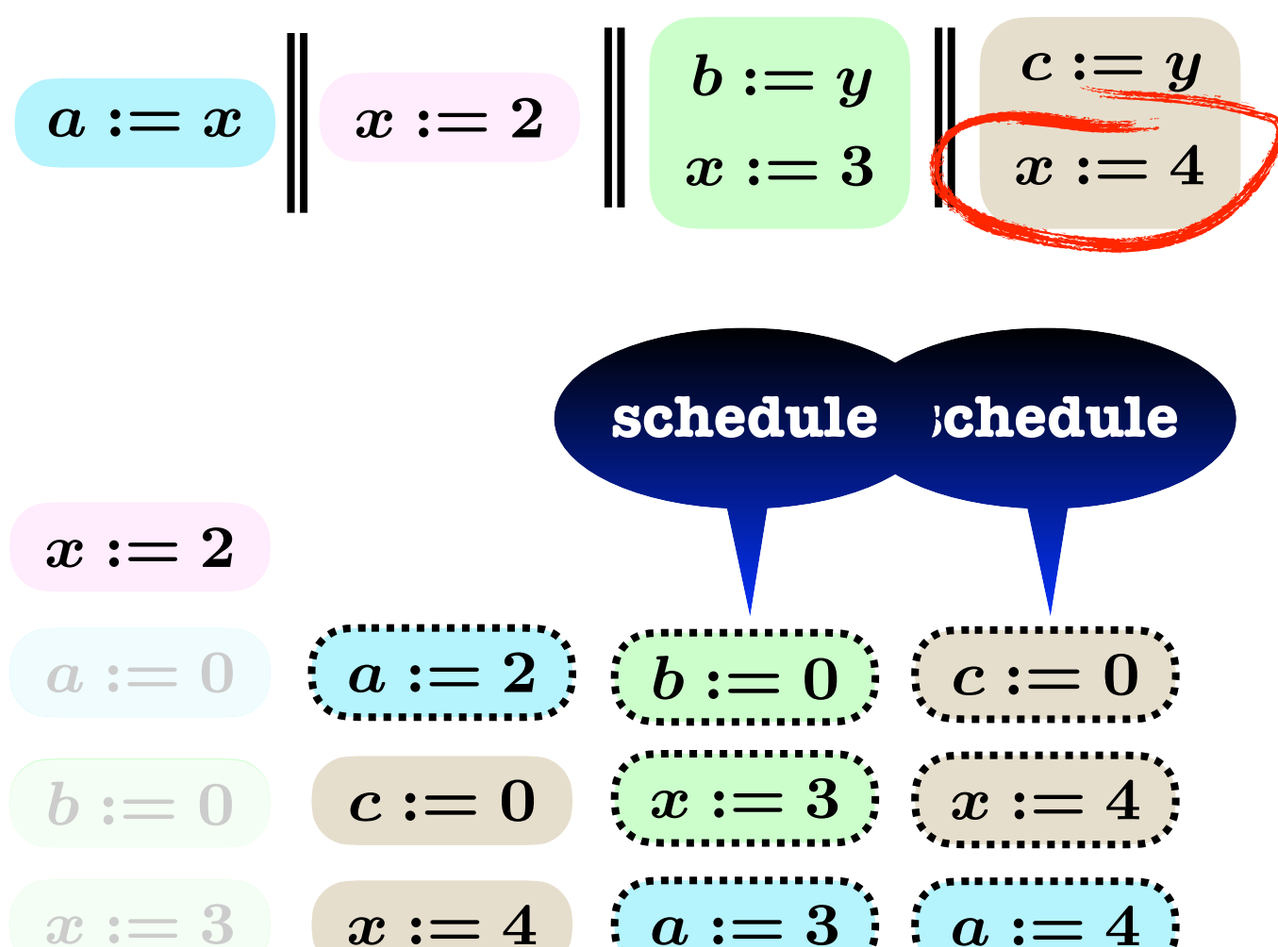




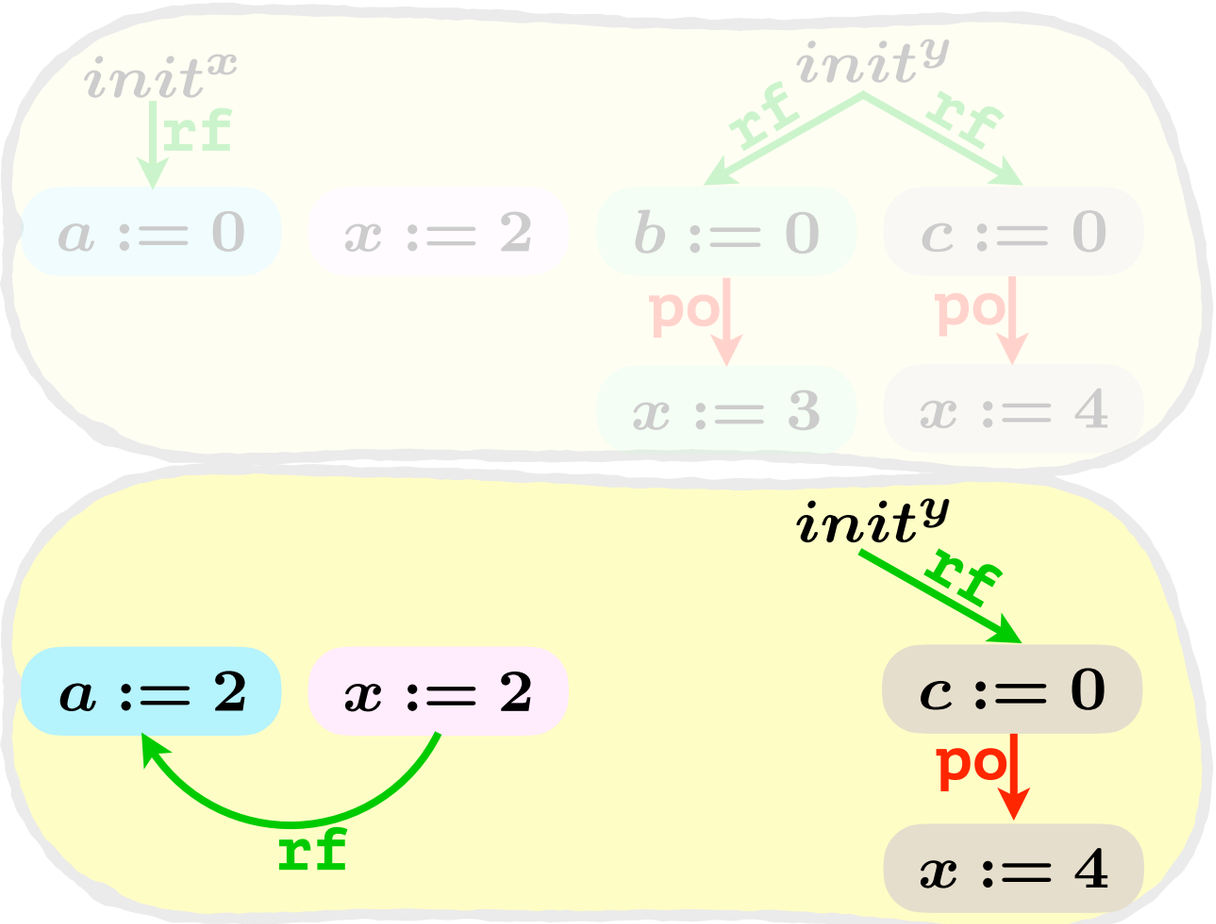


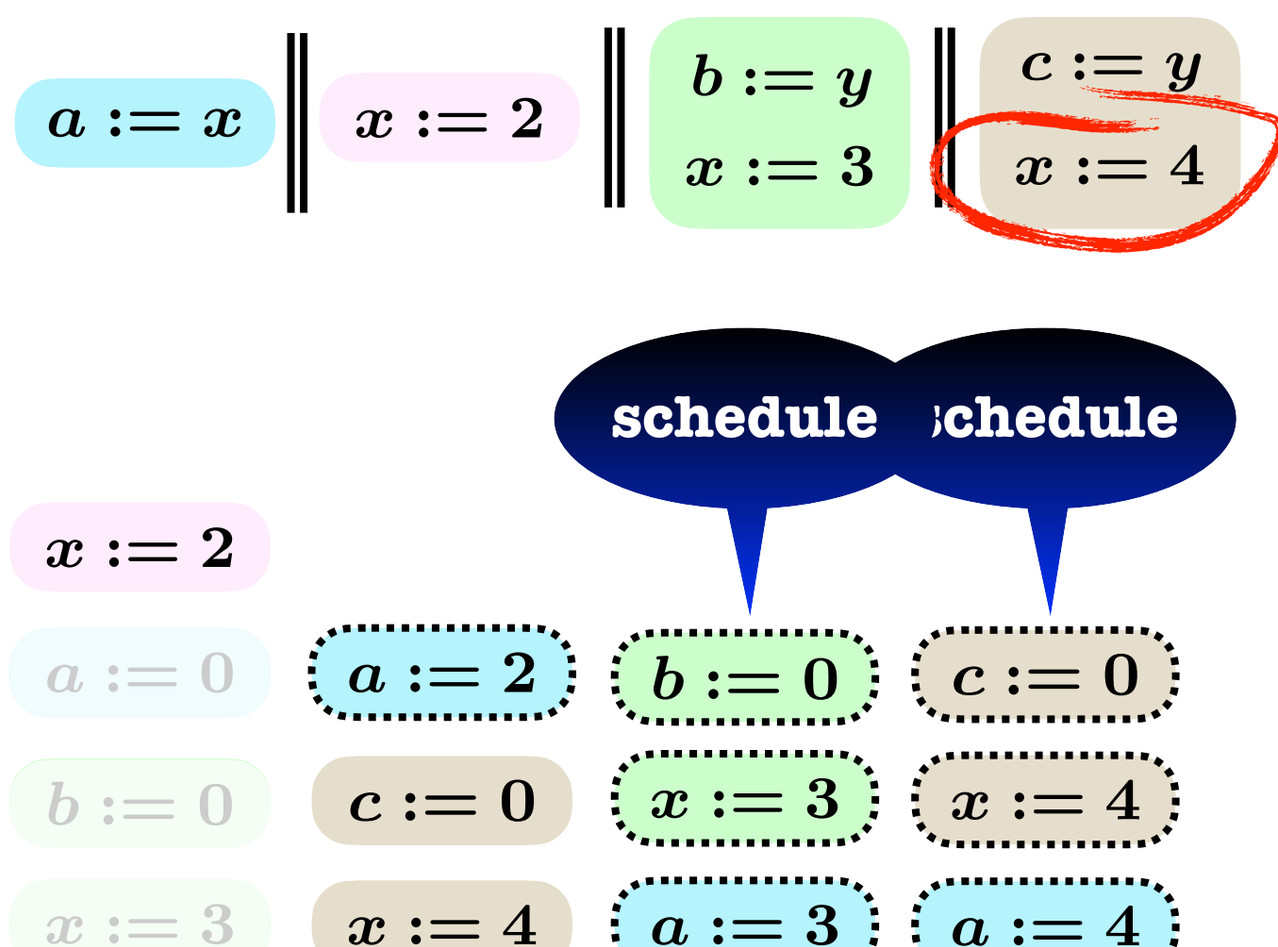
Branch & Postpone



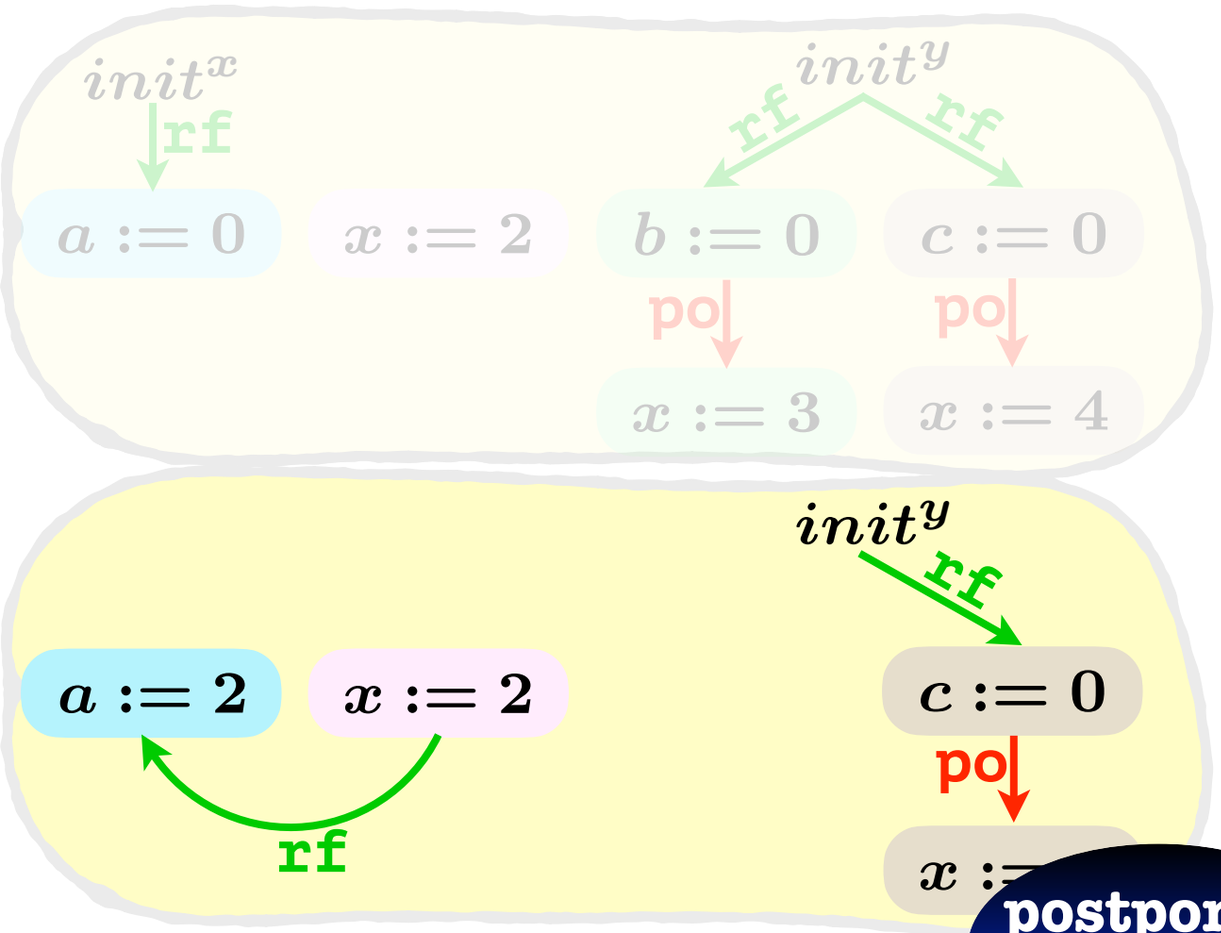


Branch & Postpone





Branch & Postpone



$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

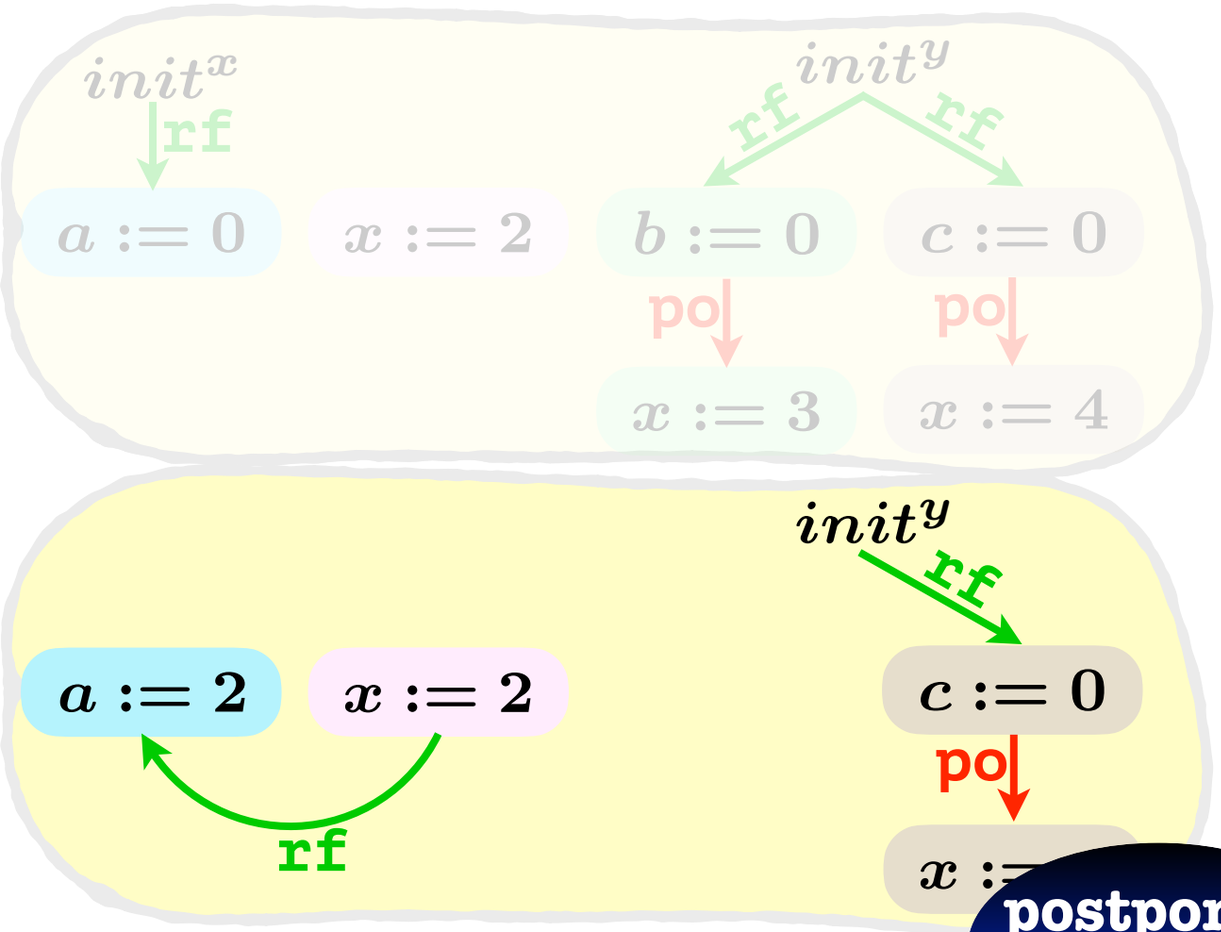
schedule schedule

$x := 2$

$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$			
$x := 4$			

postponed write

Branch & Postpone



$a := x \parallel x := 2 \parallel b := y \; x := 3 \parallel c := y \; x := 4$

schedule schedule

$x := 2$

$a := 0$ $a := 2$ $b := 0$ $c := 0$

$b := 0$ $c := 0$ $x := 3$ $x := 4$

$x := 3$ $x := 4$ $a := 3$ $a := 4$

postponed write

$c := 0$

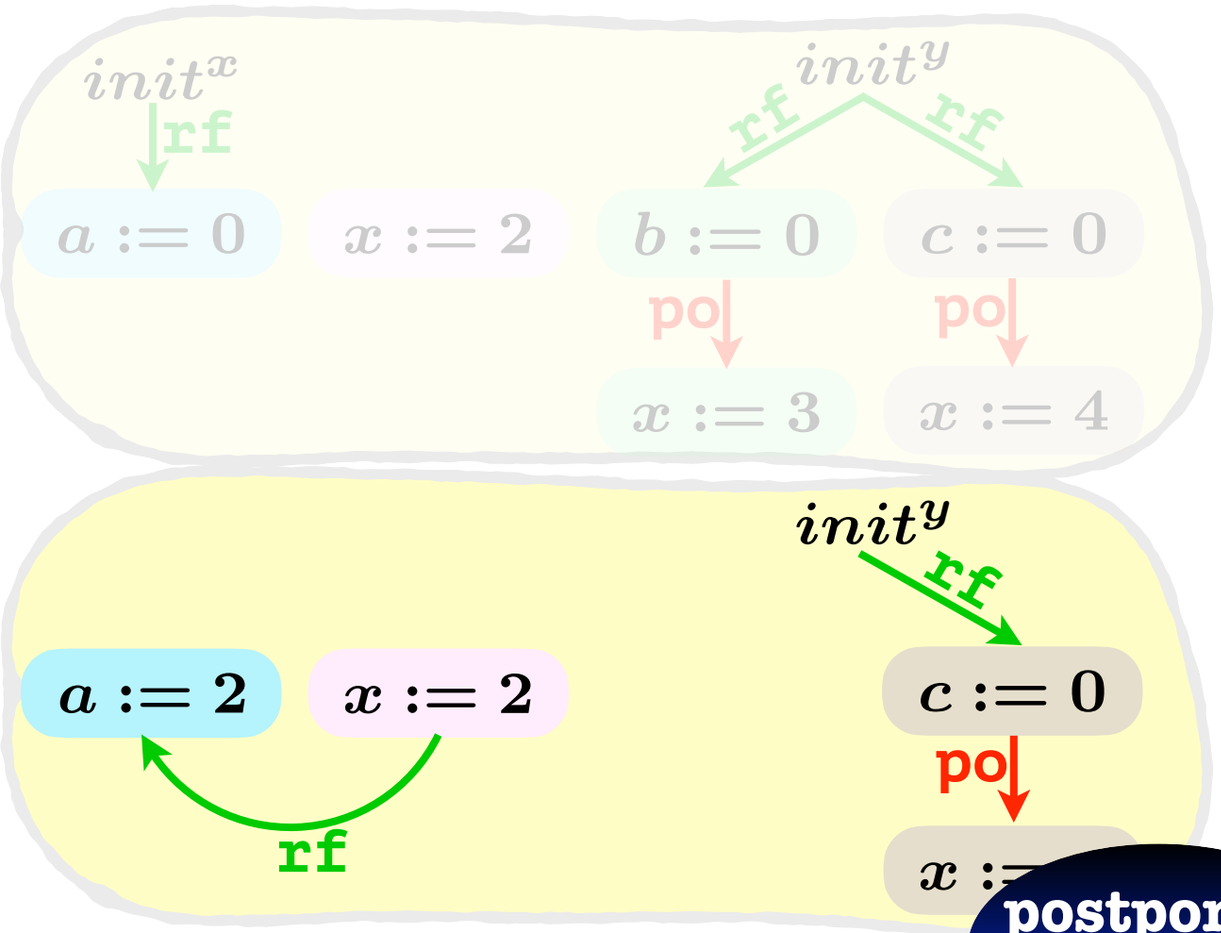
$x := 4$

$c := 0$

$x := 4$

$a := 4$

Branch & Postpone



$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

schedule

schedule

$x := 2$

$a := 0$ $\quad a := 2$ $\quad b := 0$ $\quad c := 0$

$b := 0$ $\quad c := 0$ $\quad x := 3$ $\quad x := 4$

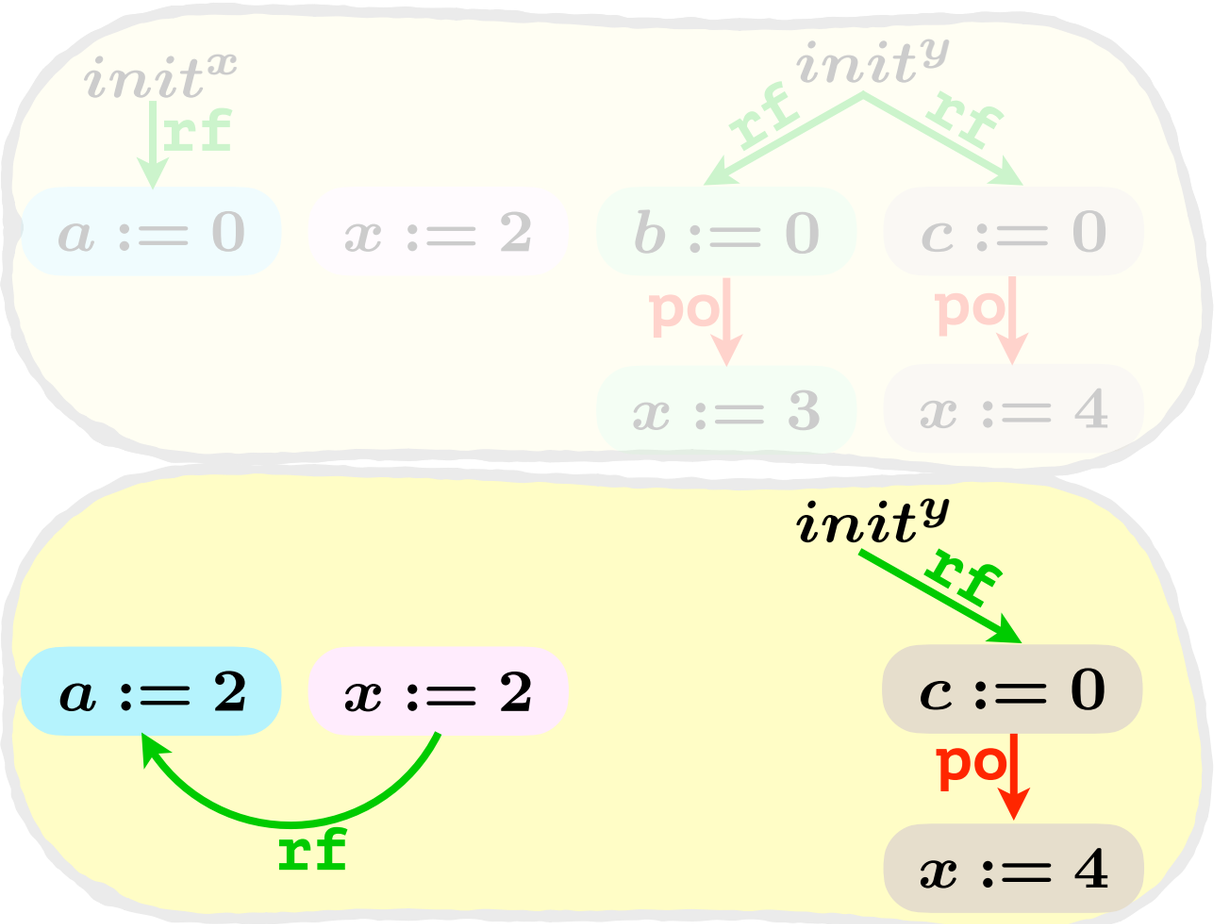
$x := 3$ $\quad x := 4$ $\quad a := 3$ $\quad a := 4$

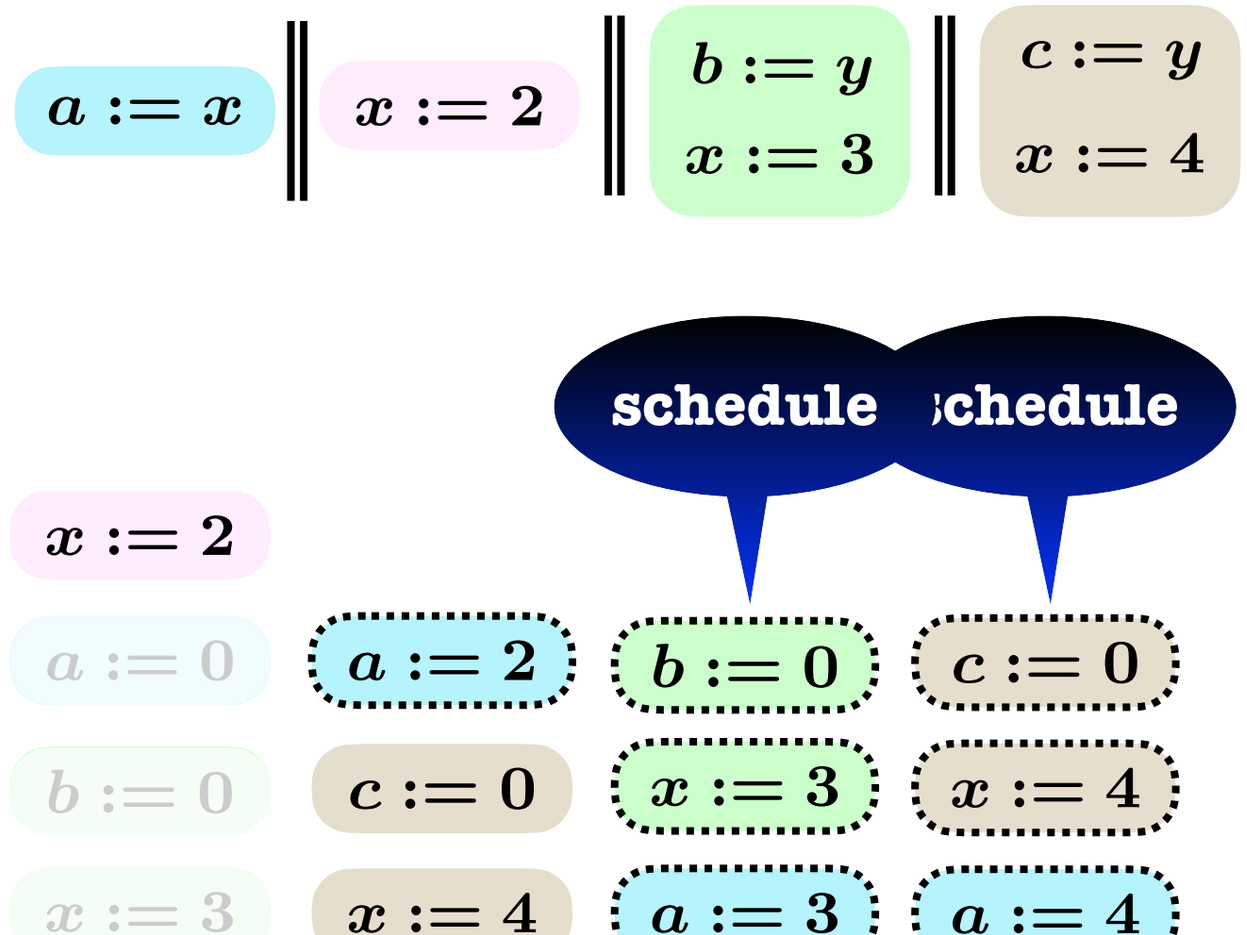
postponed write

$c := 0$

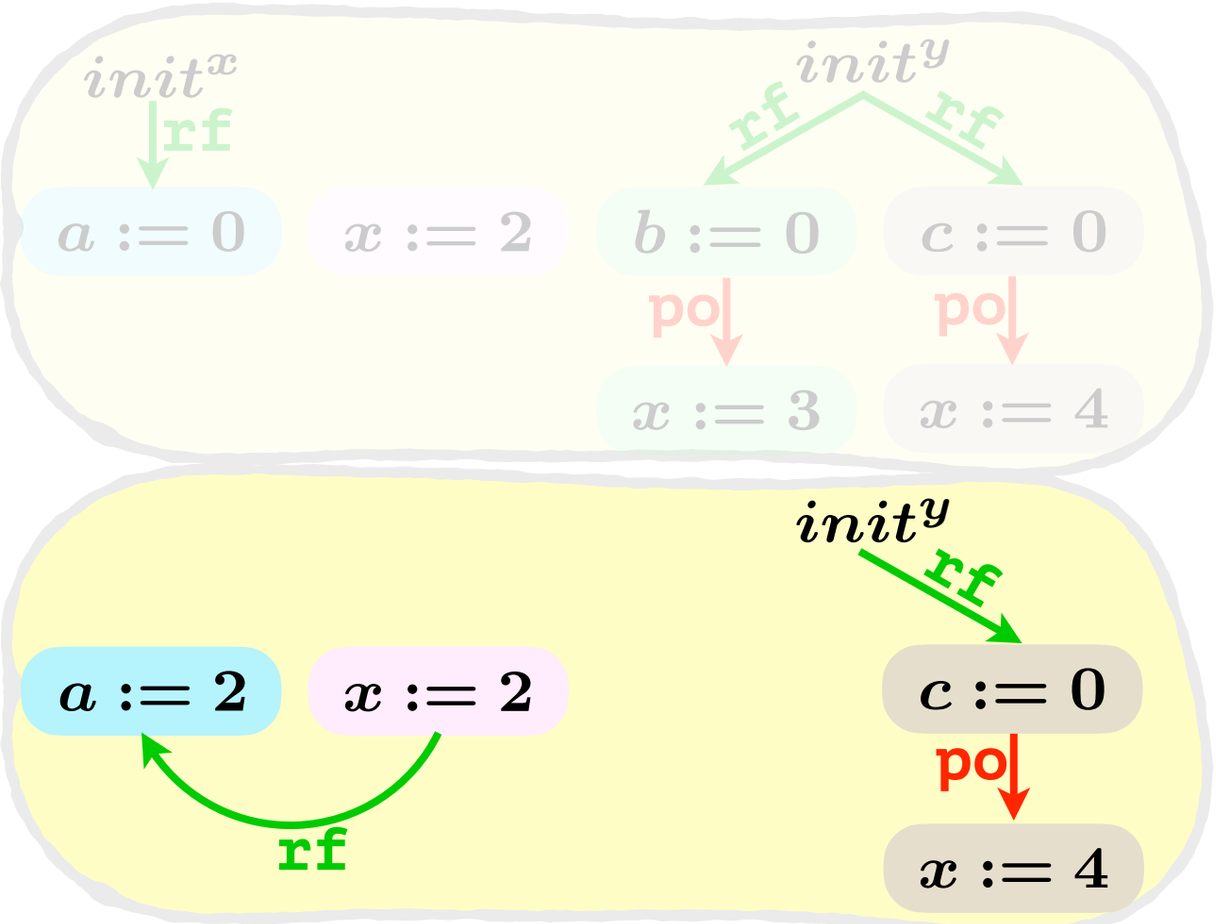
$x := 4$

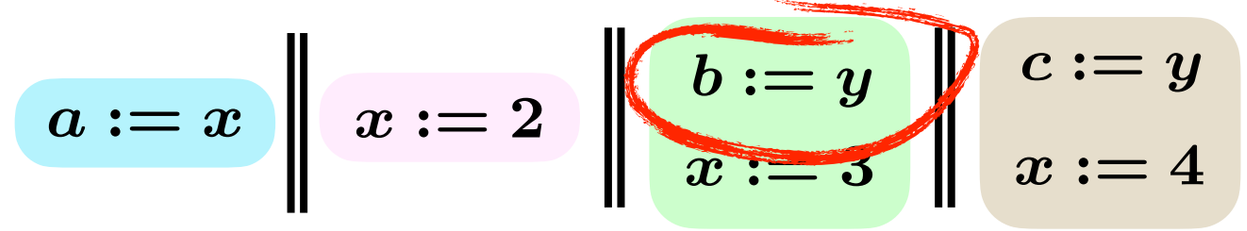
Branch **&** **Postpone**

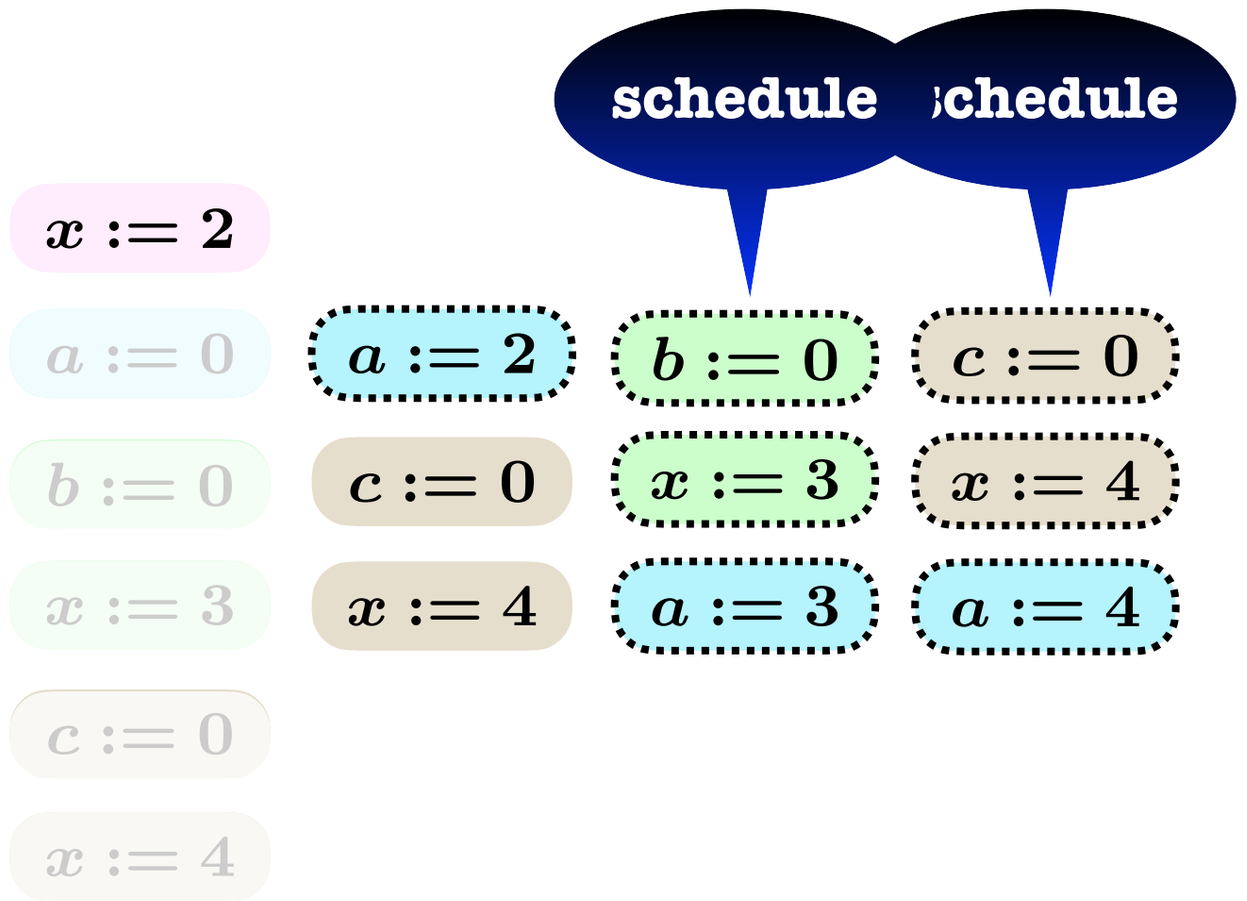




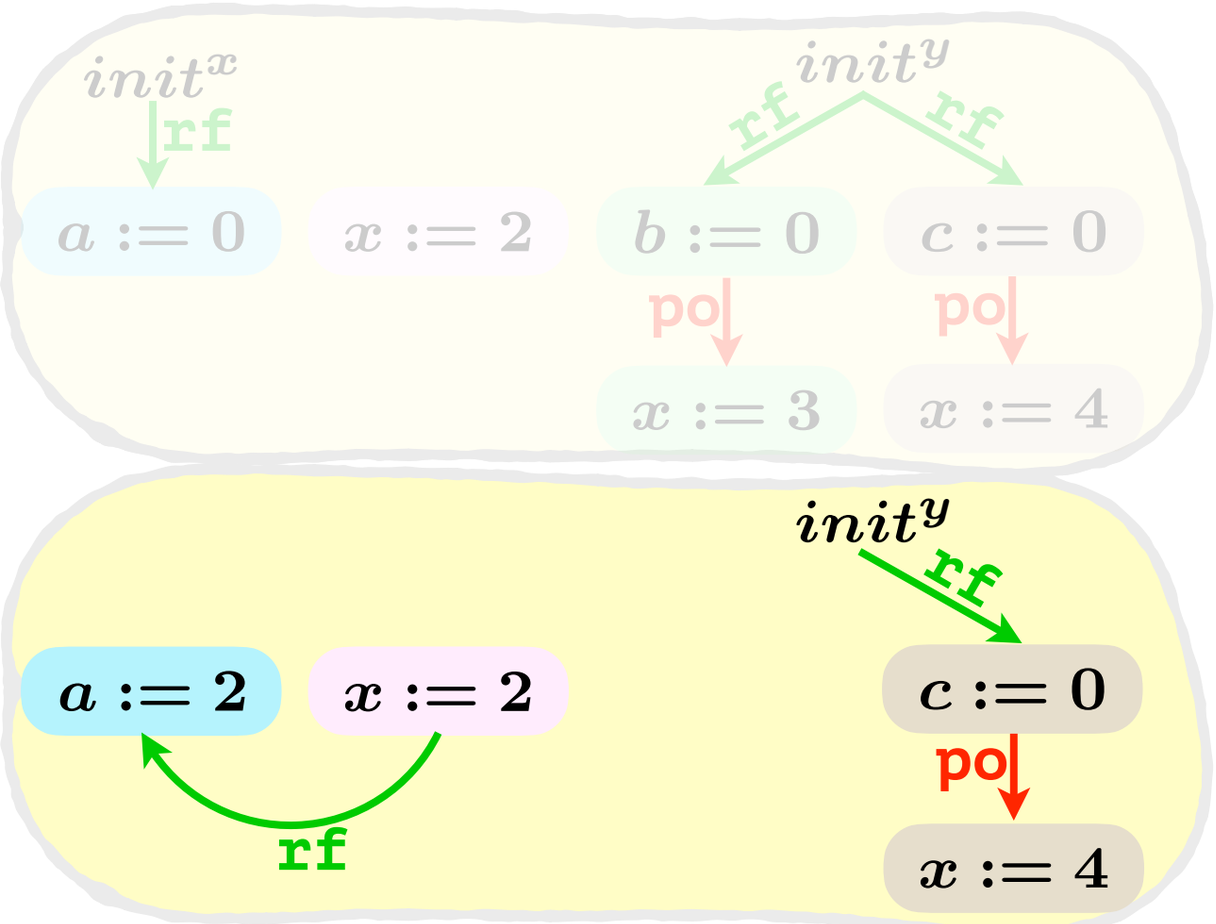
Branch & Postpone

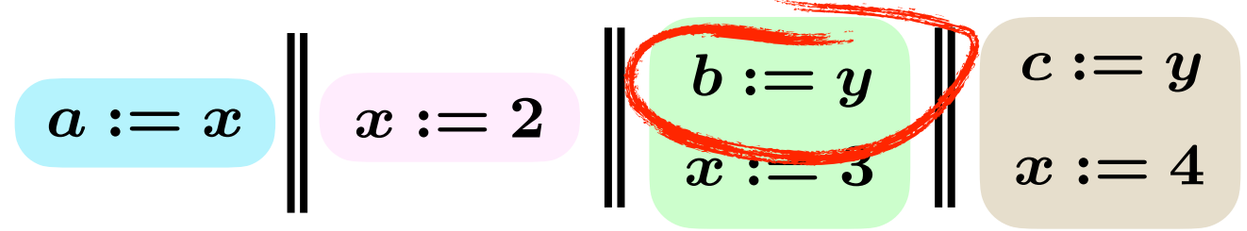


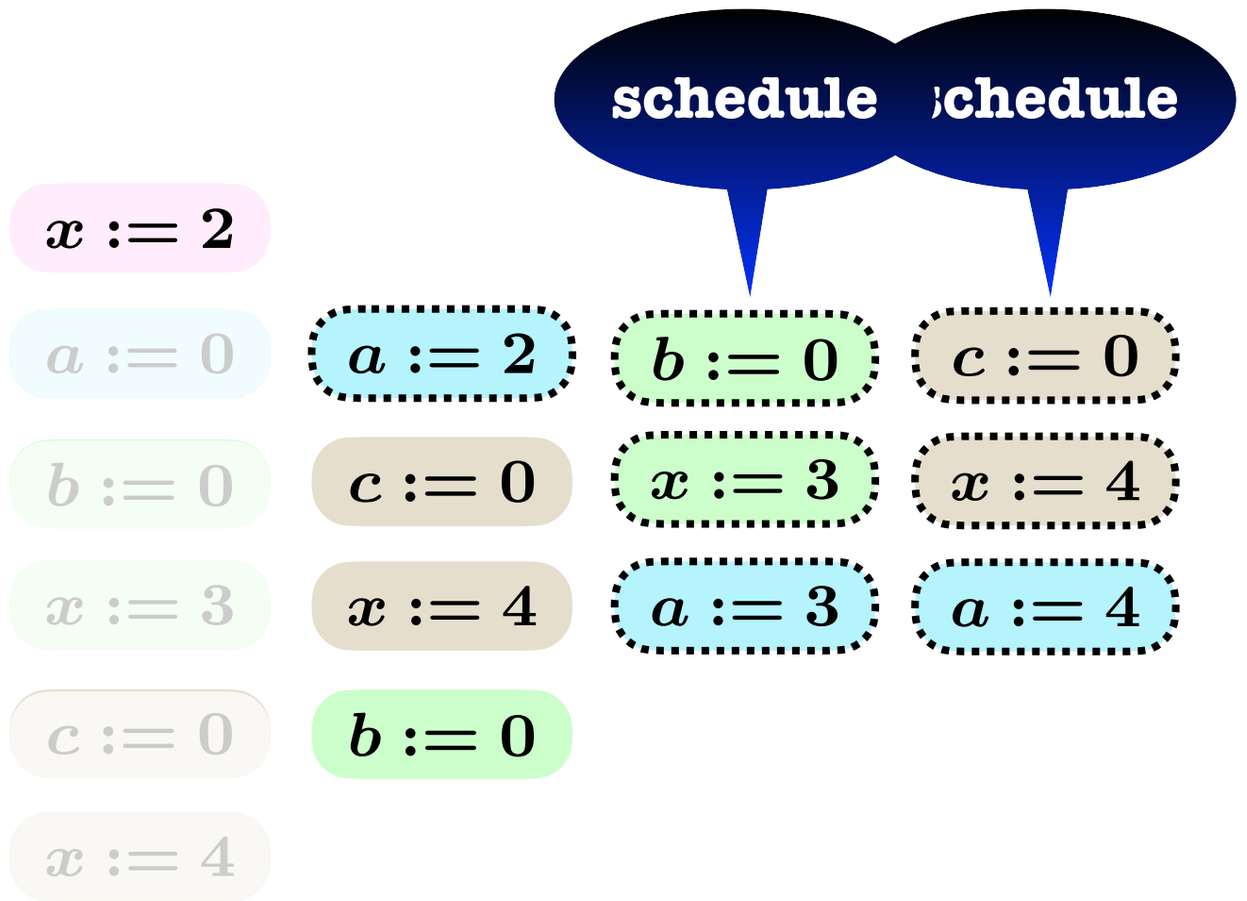




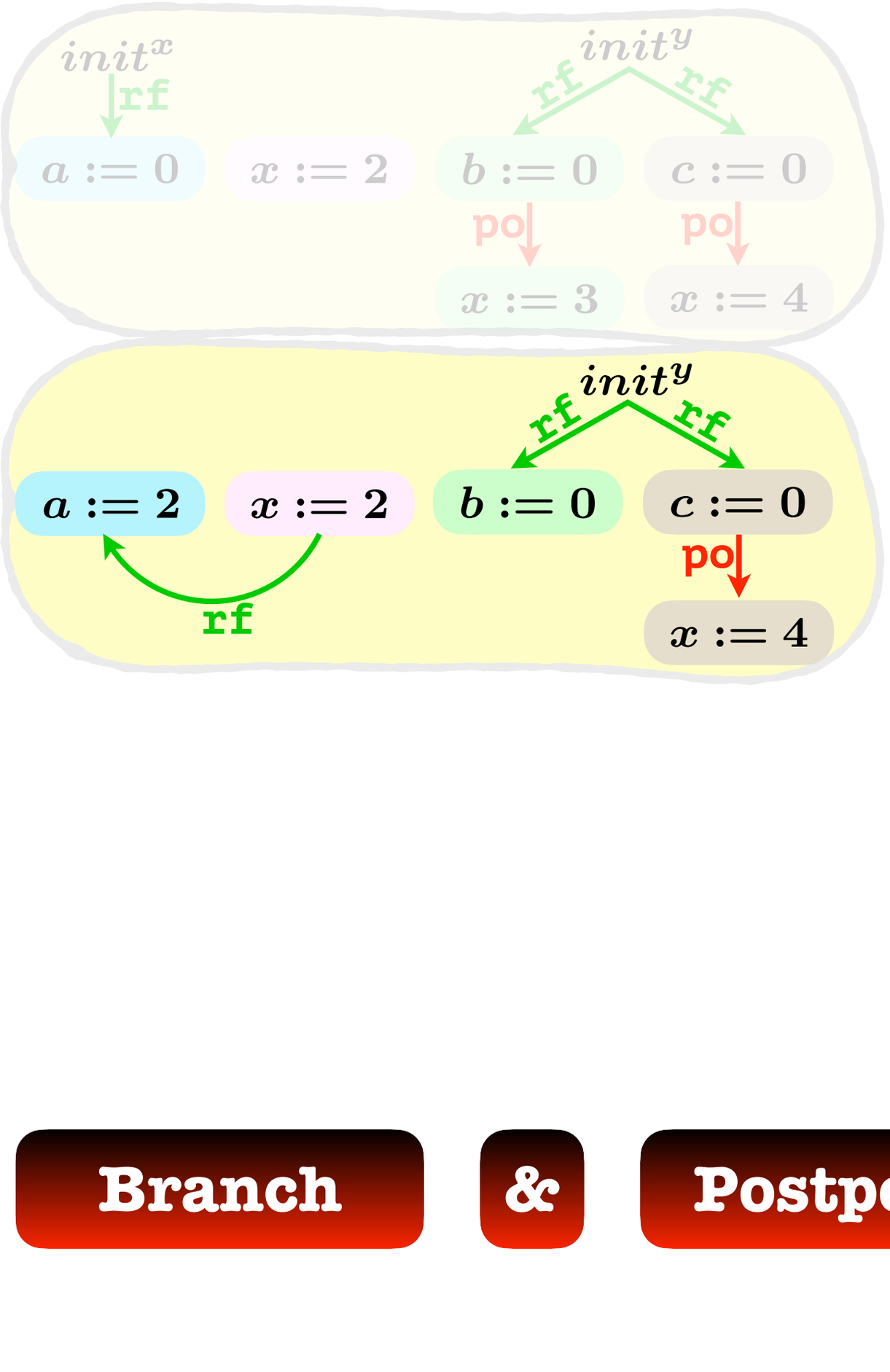
Branch & Postpone

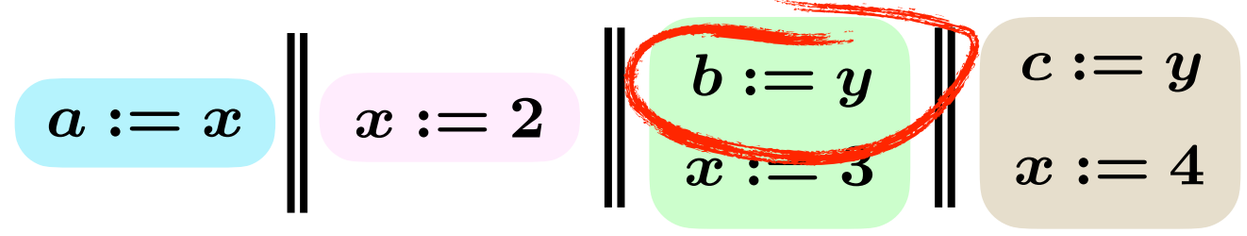


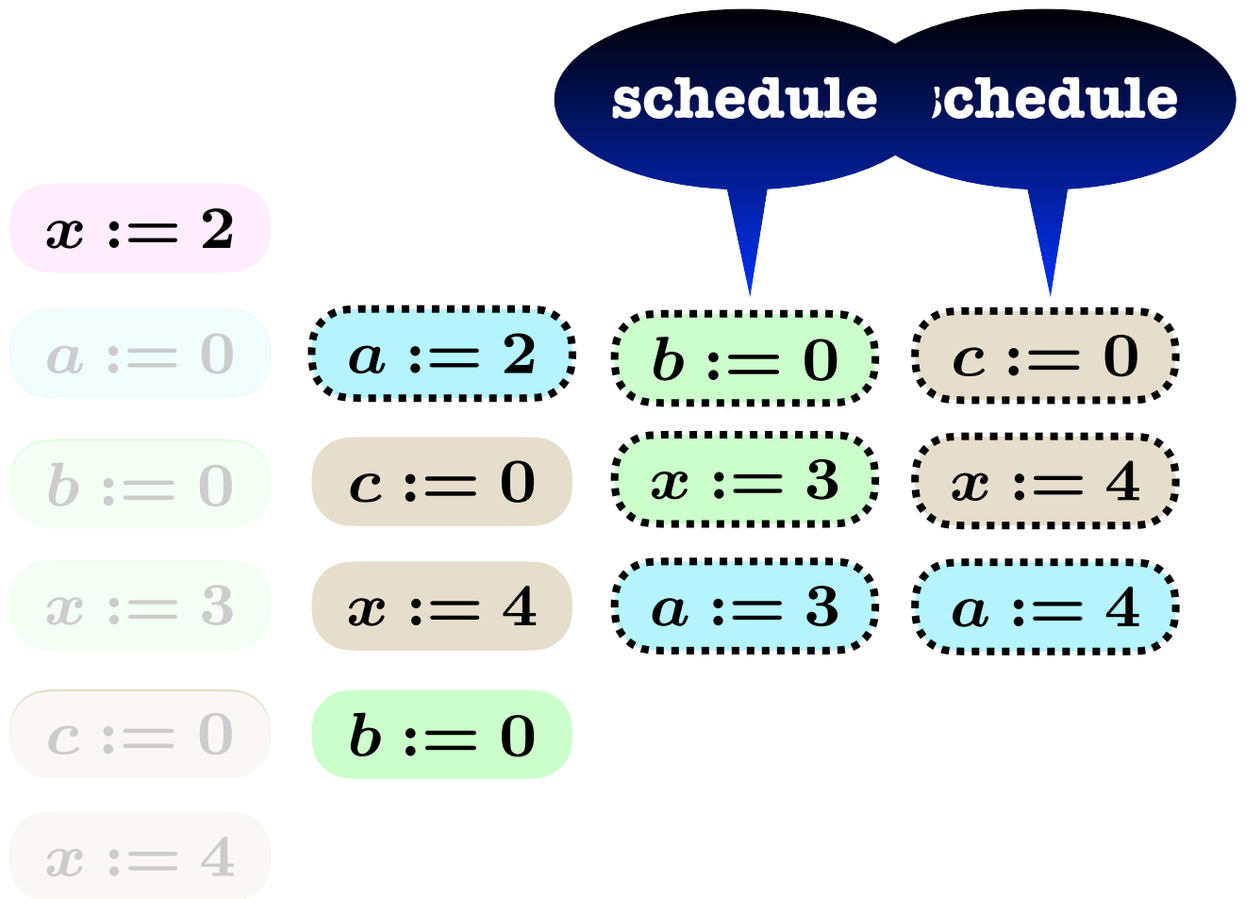




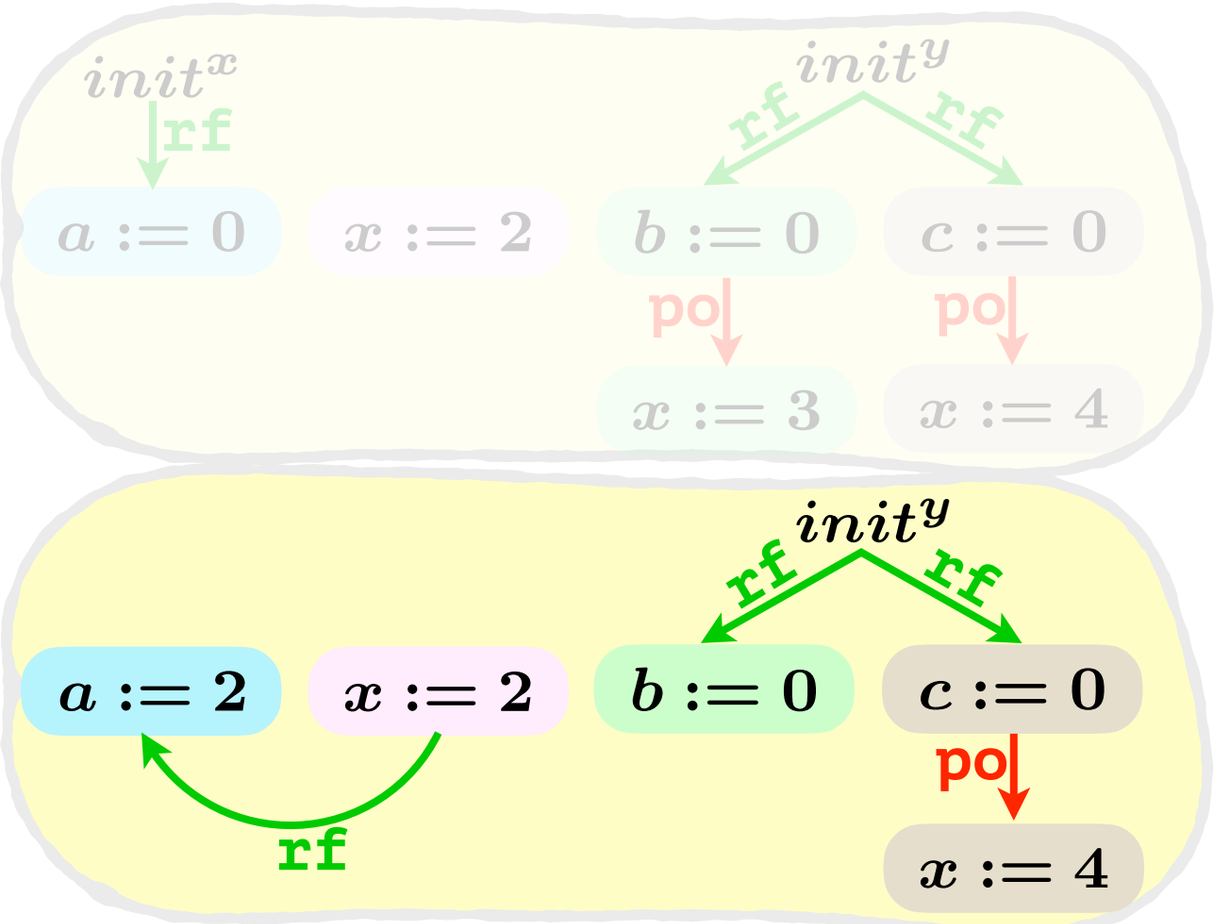
Branch & Postpone

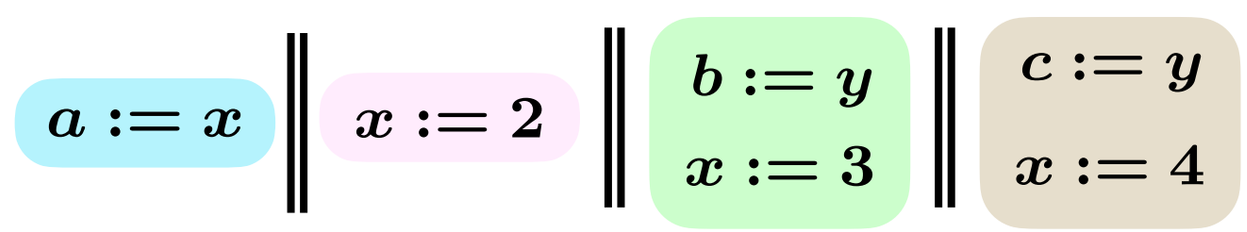


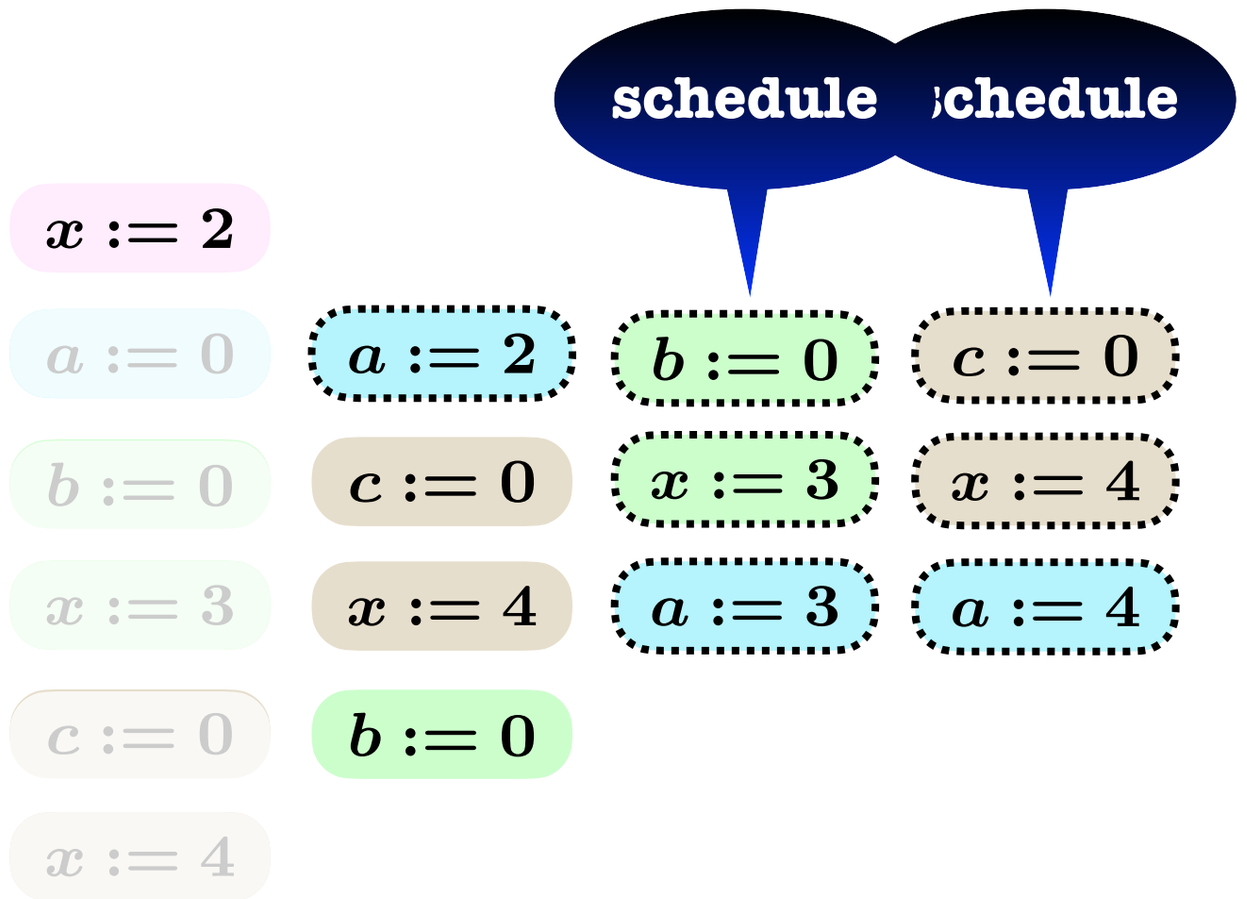




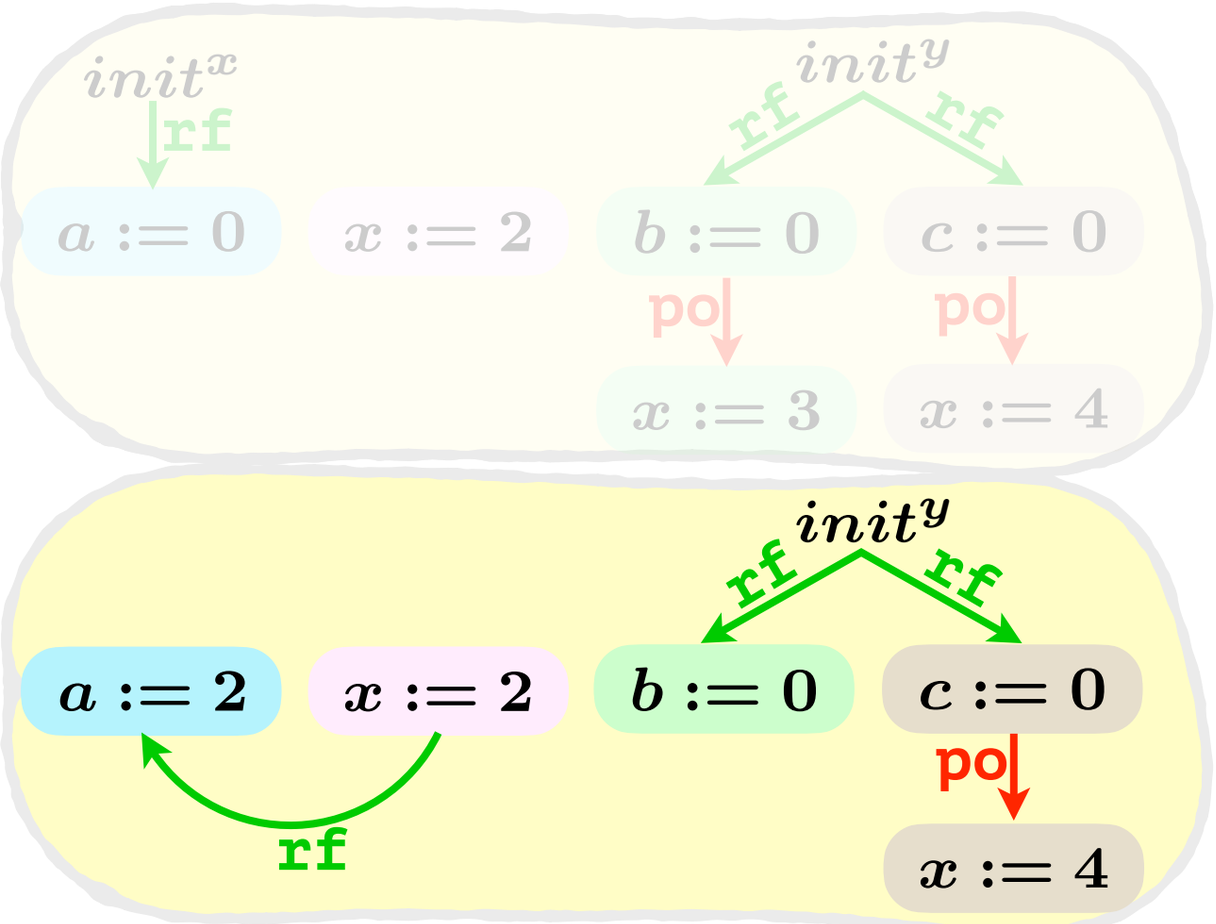
Branch & Postpone

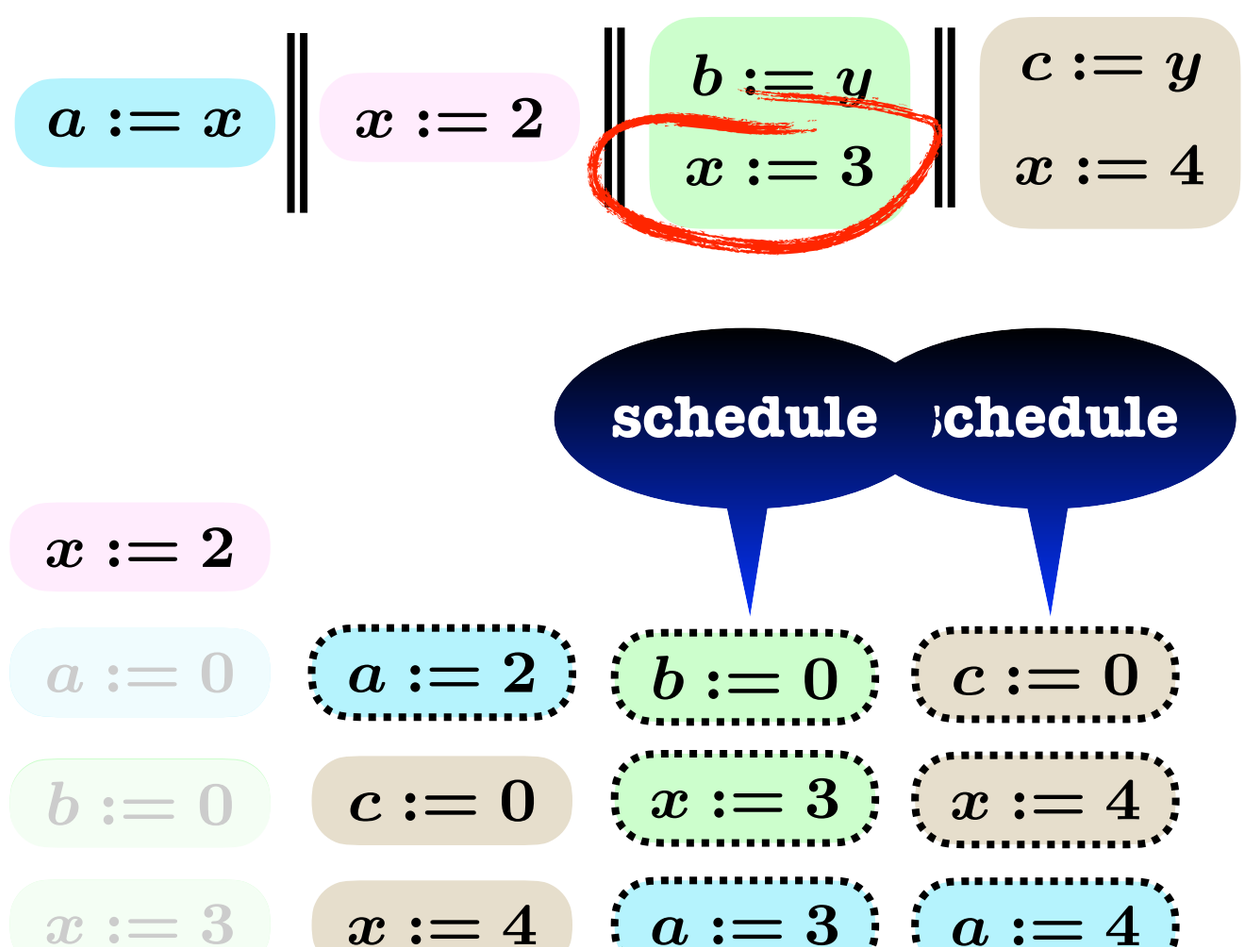




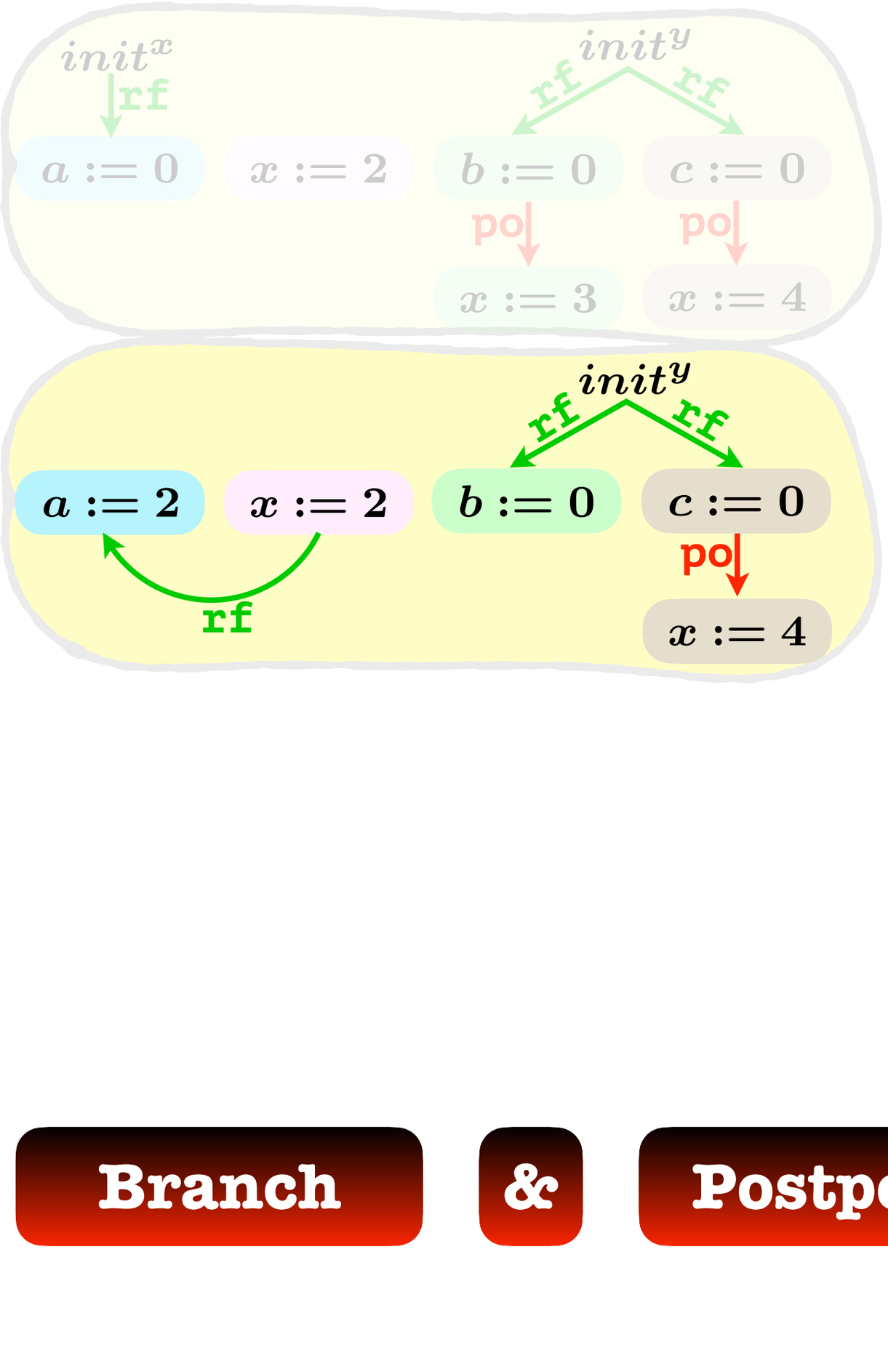


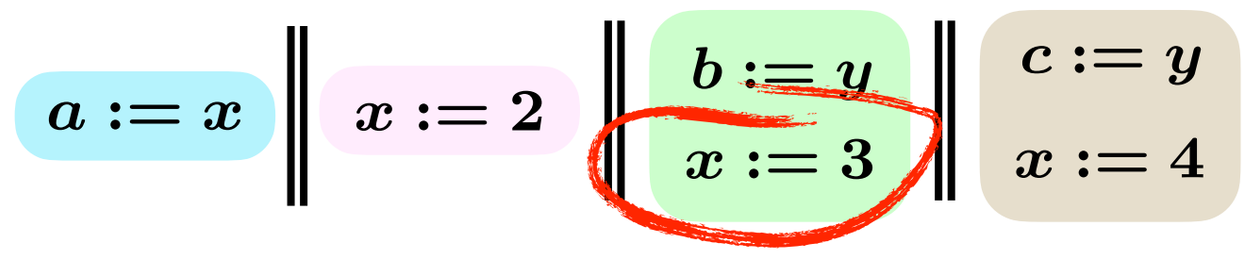
Branch & Postpone

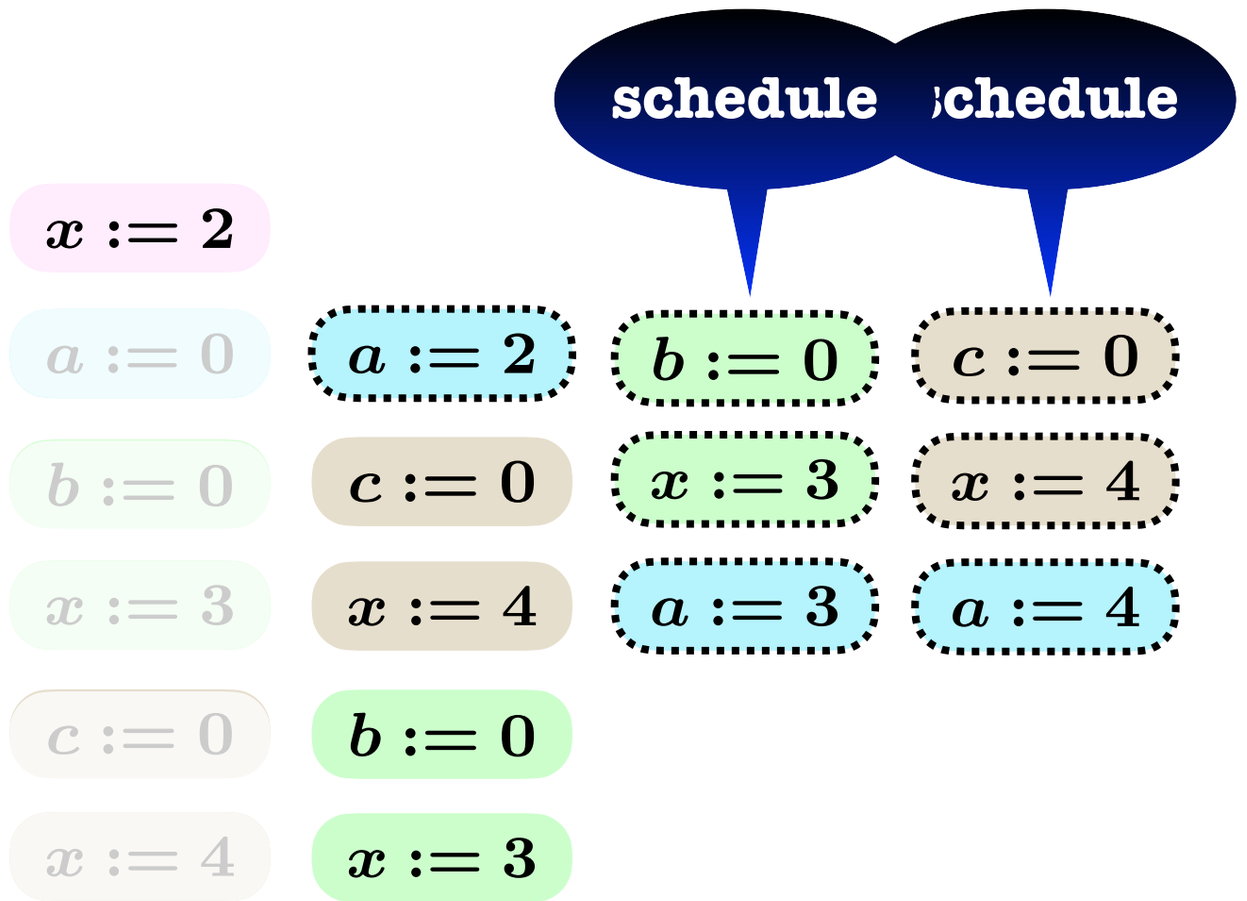




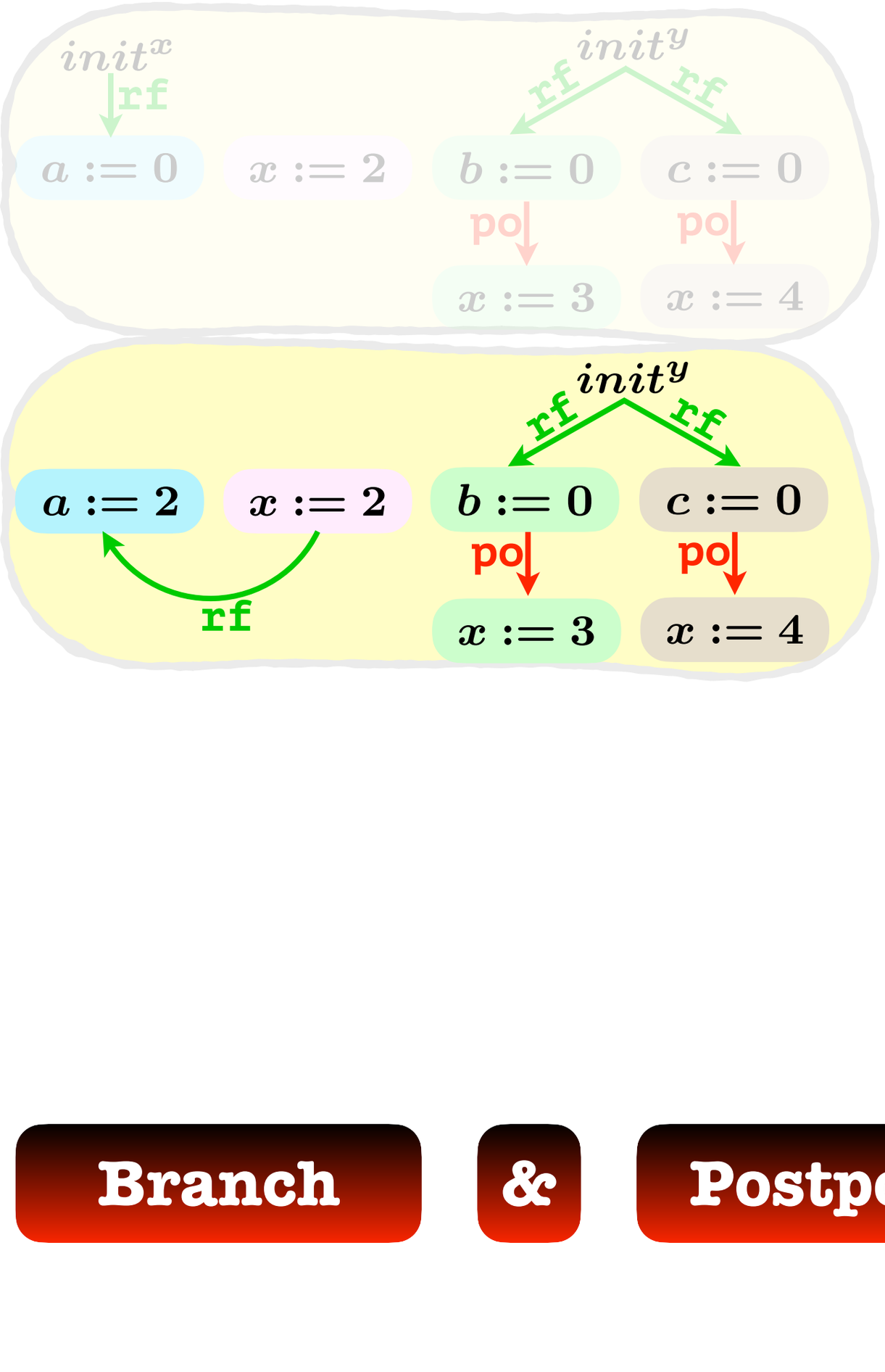
Branch **&** **Postpone**

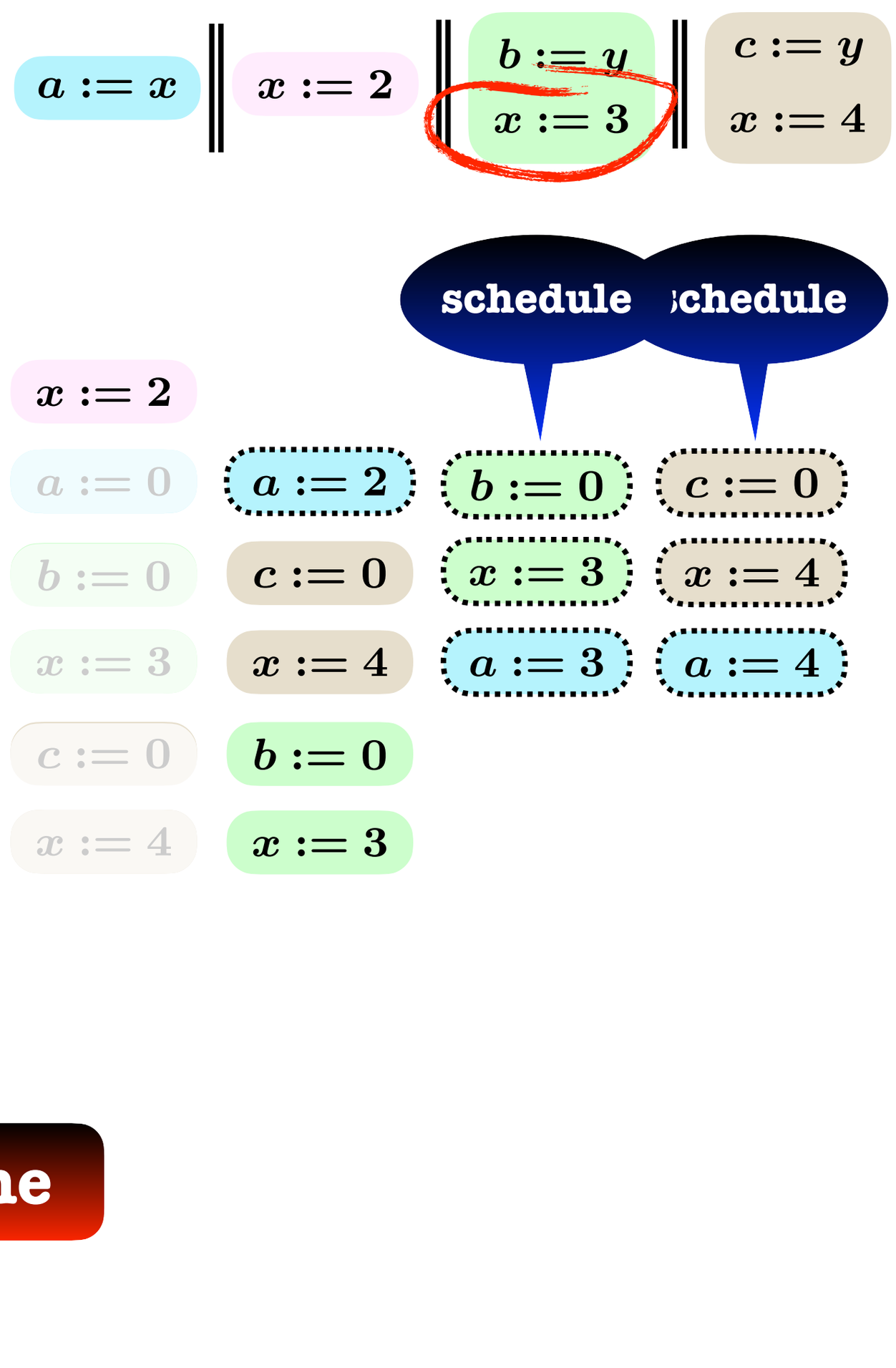




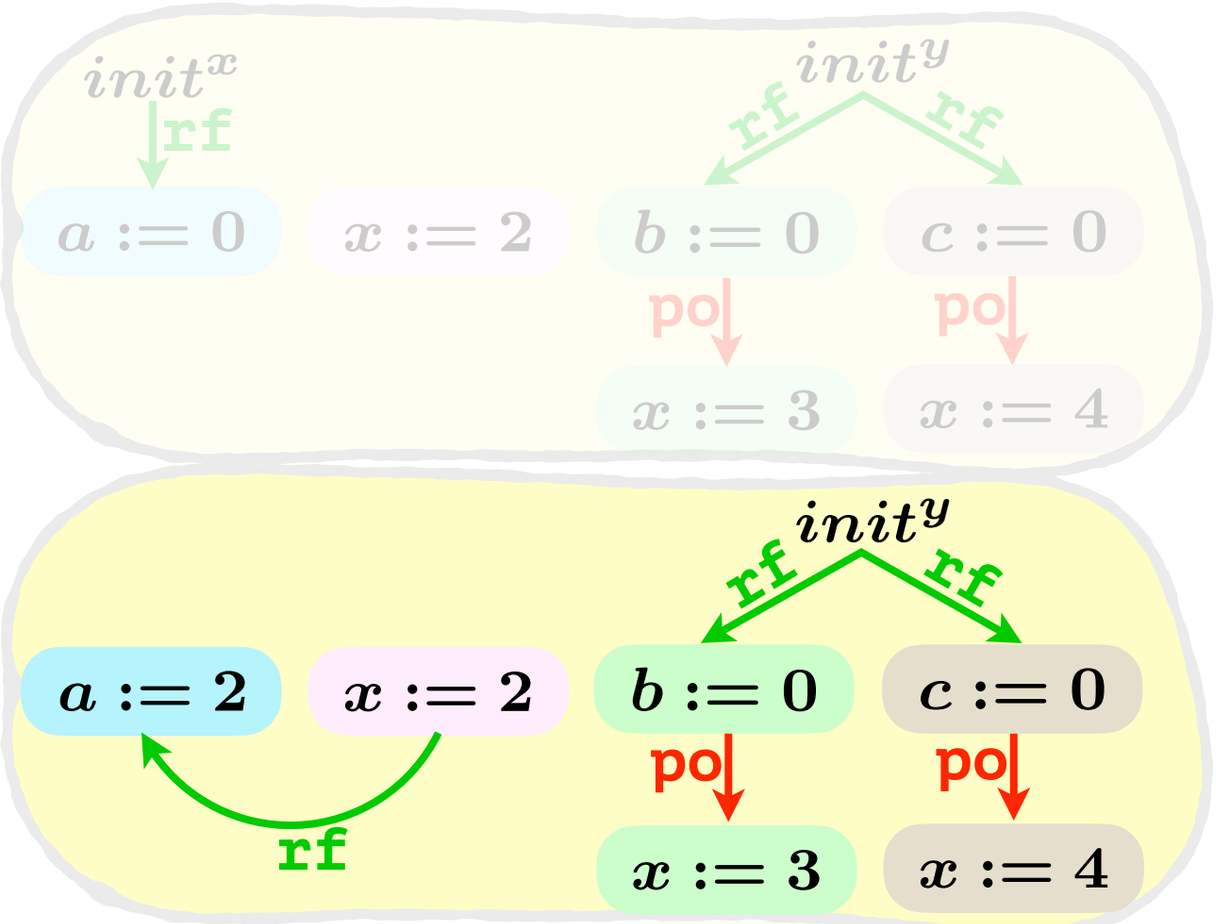


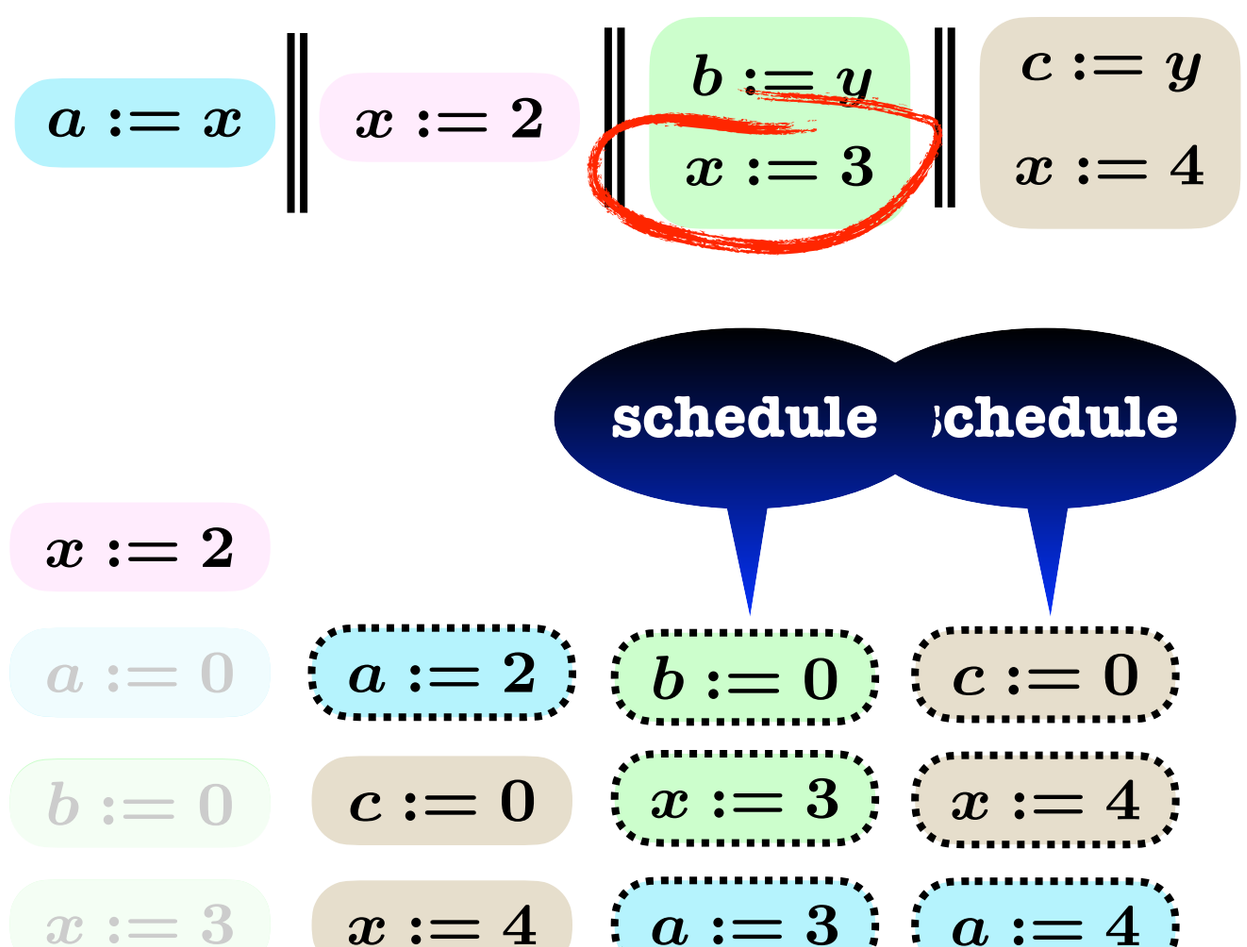
Branch & Postpone



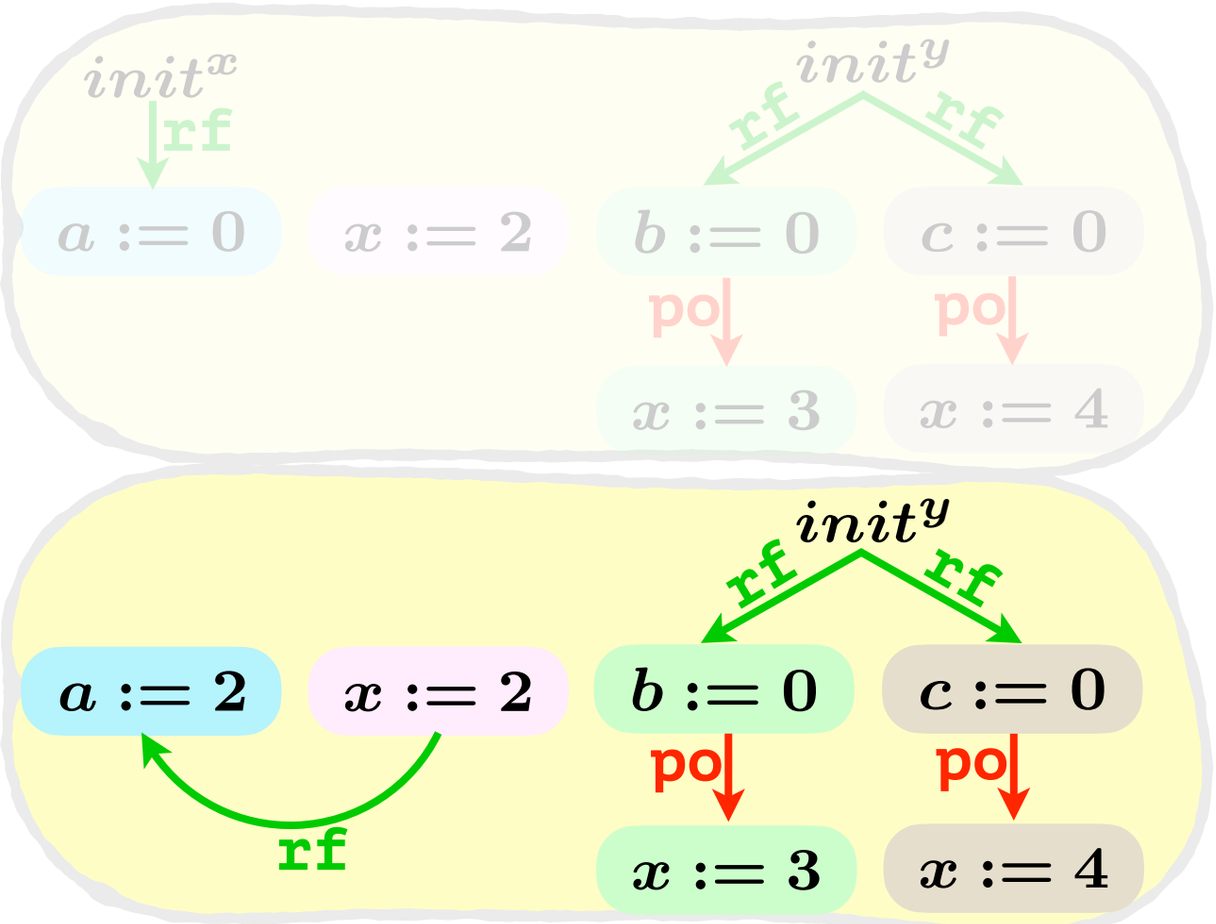


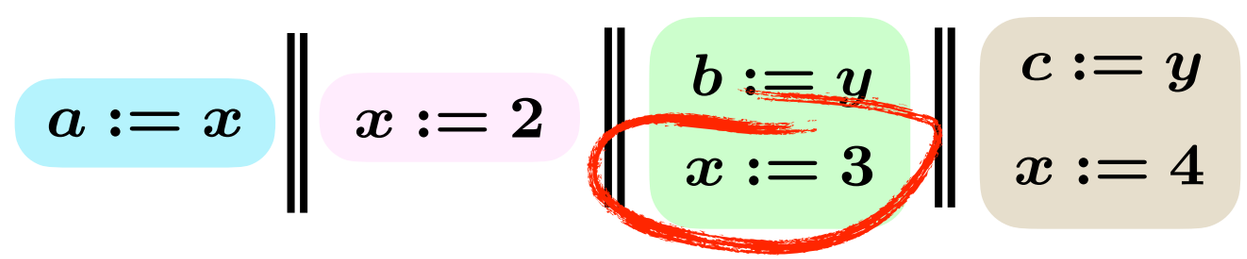
Branch **&** **Postpone**

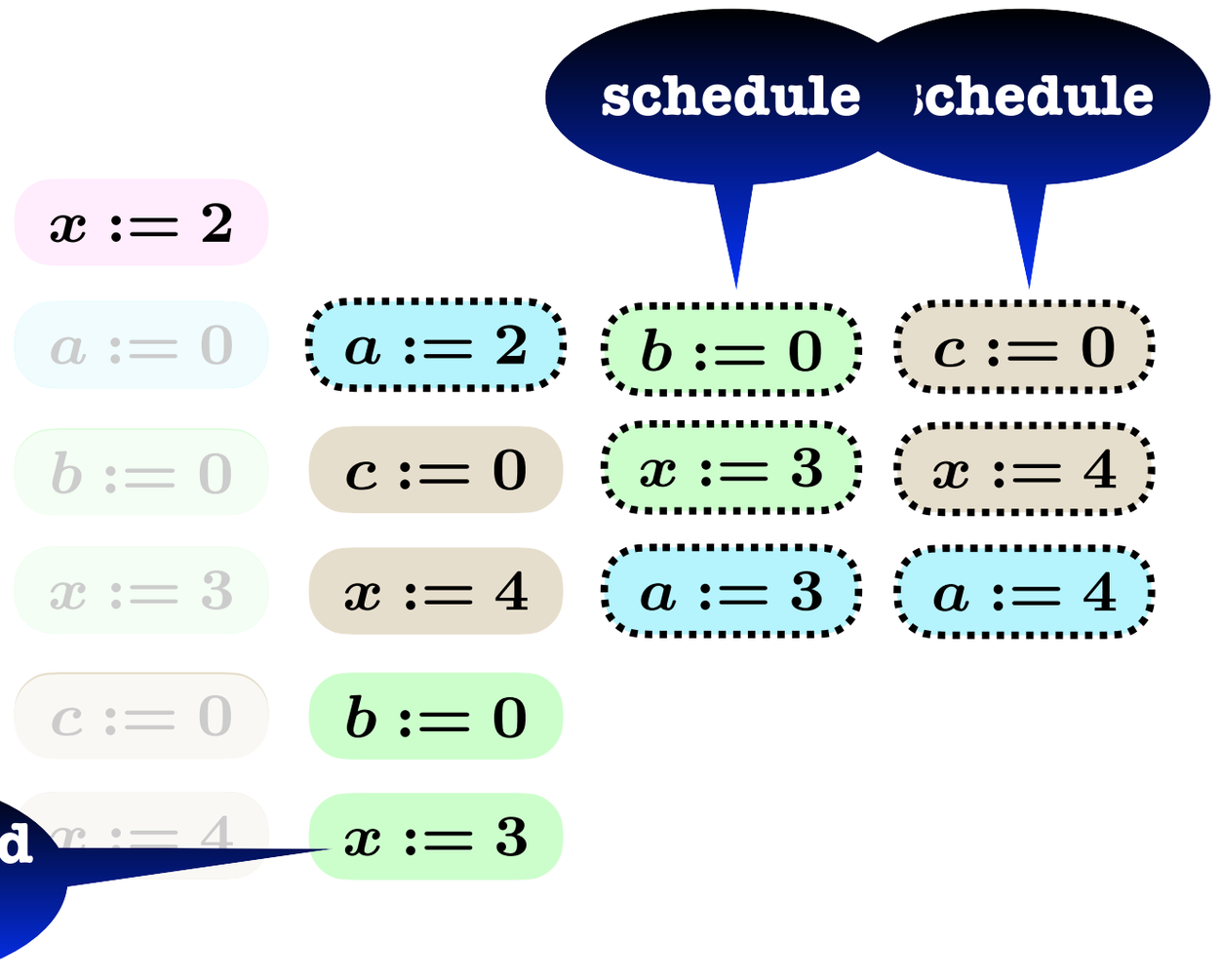


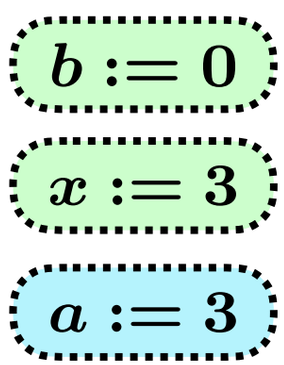


Branch & Postpone

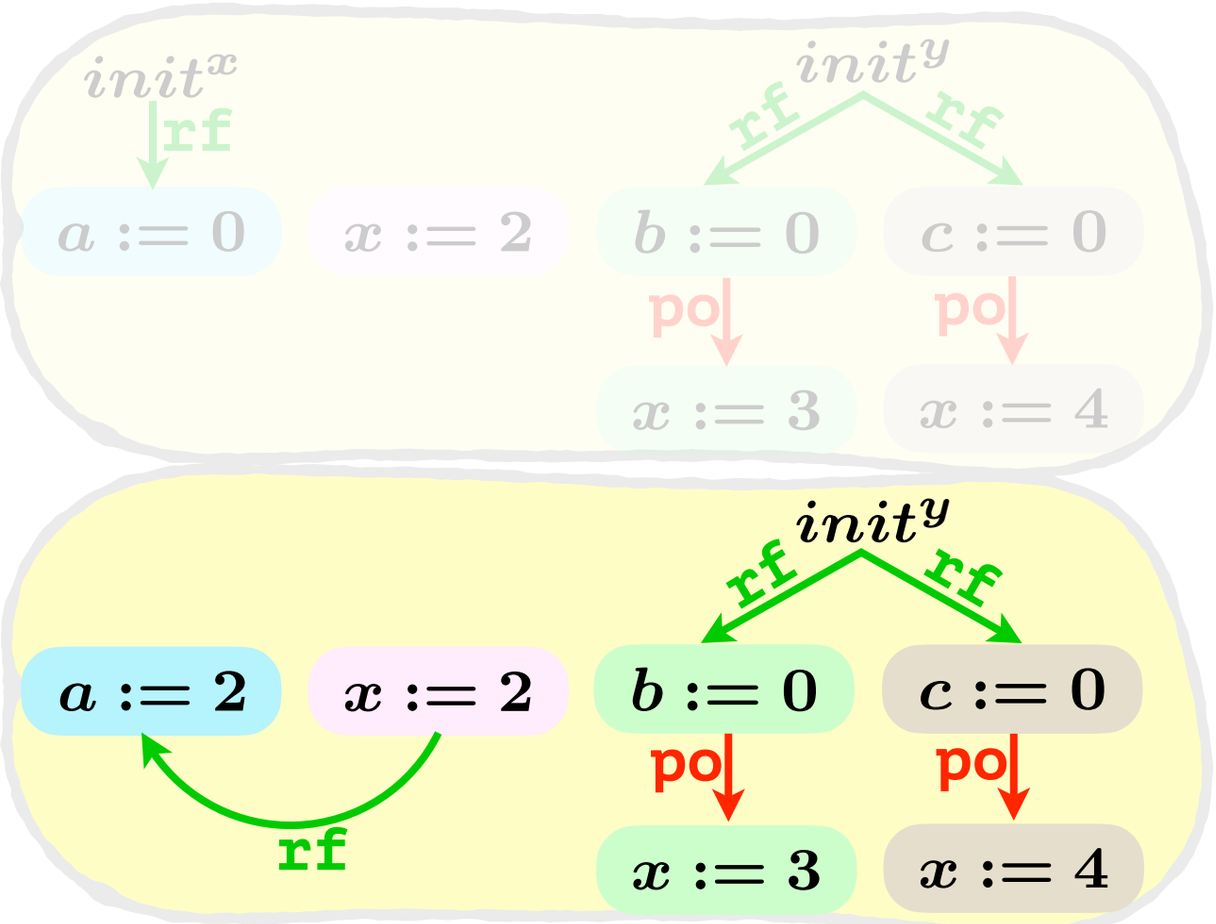


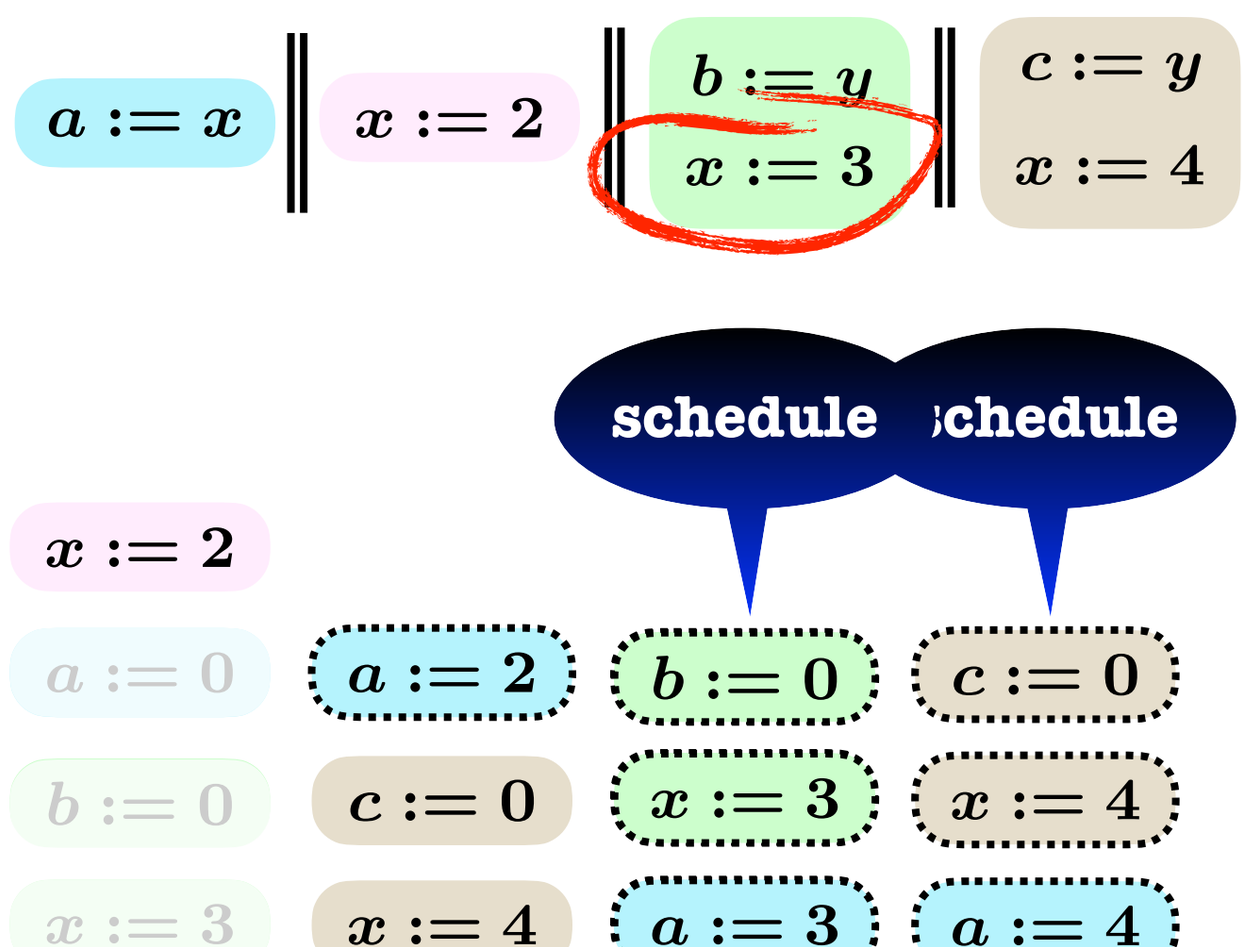




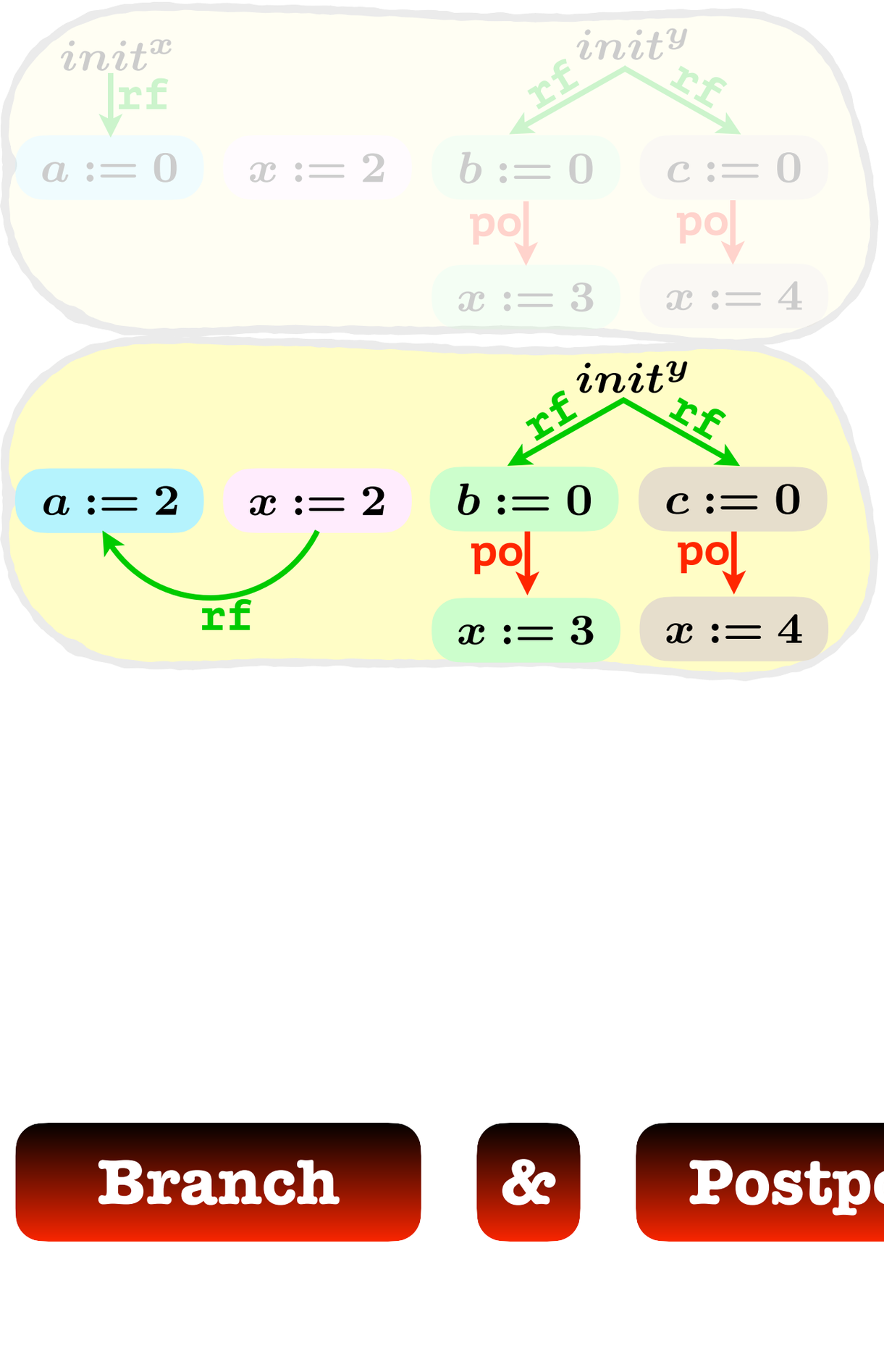


Branch **&** **Postpone**

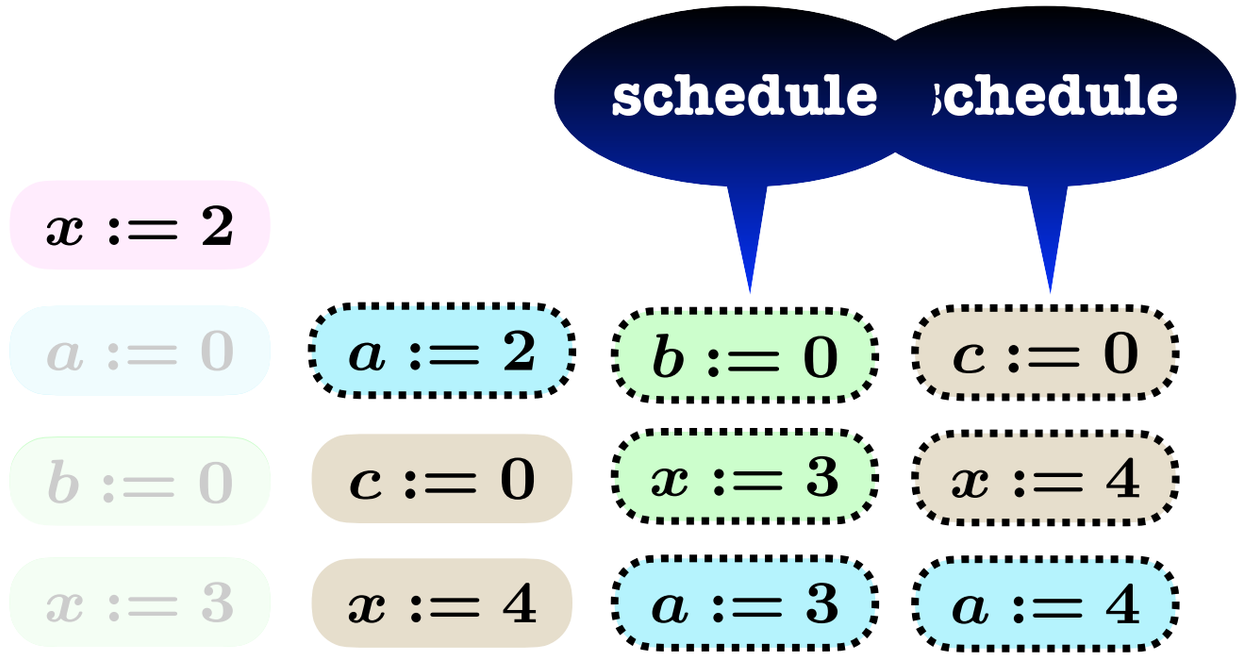




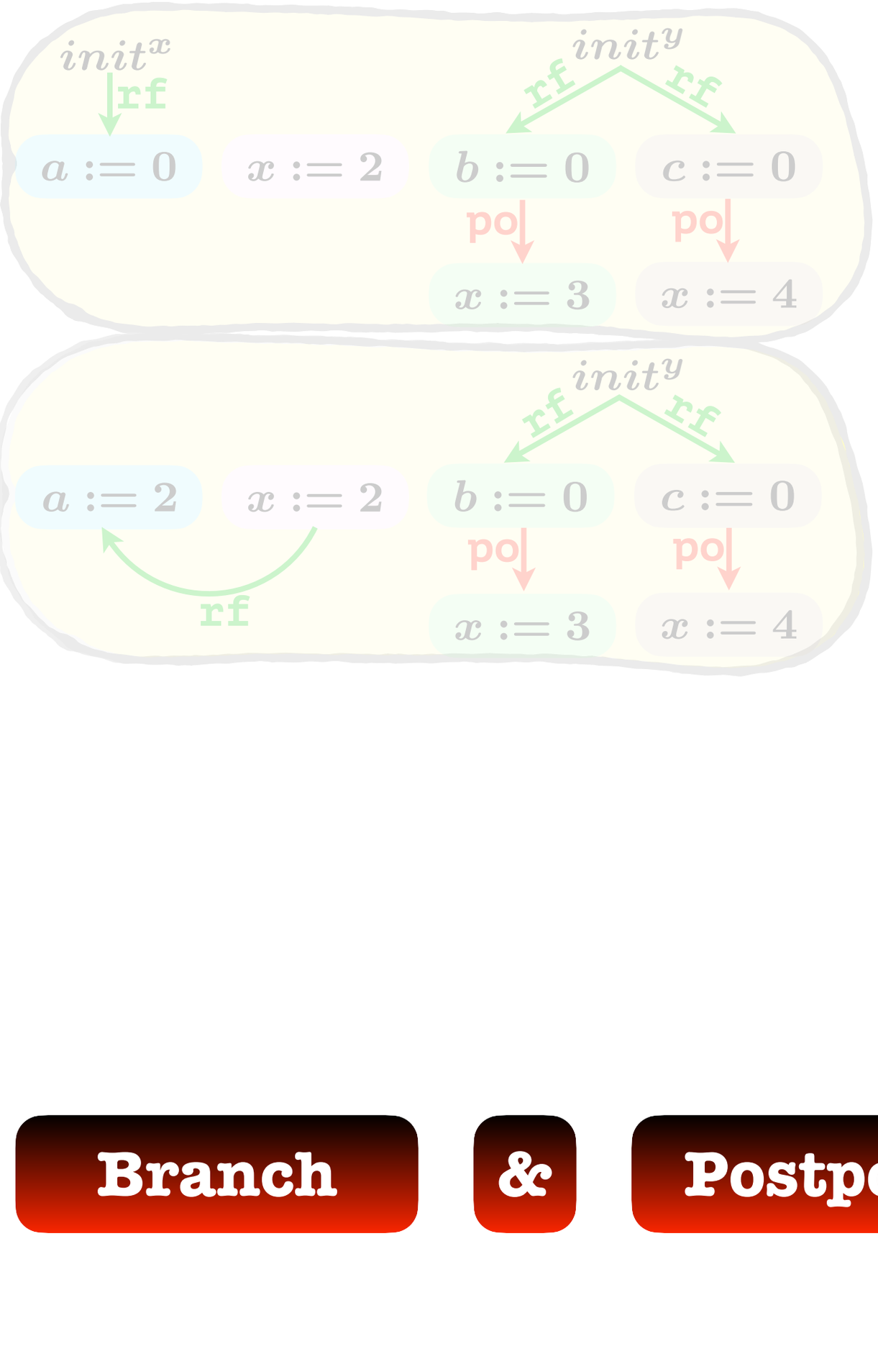
Branch & Postpone



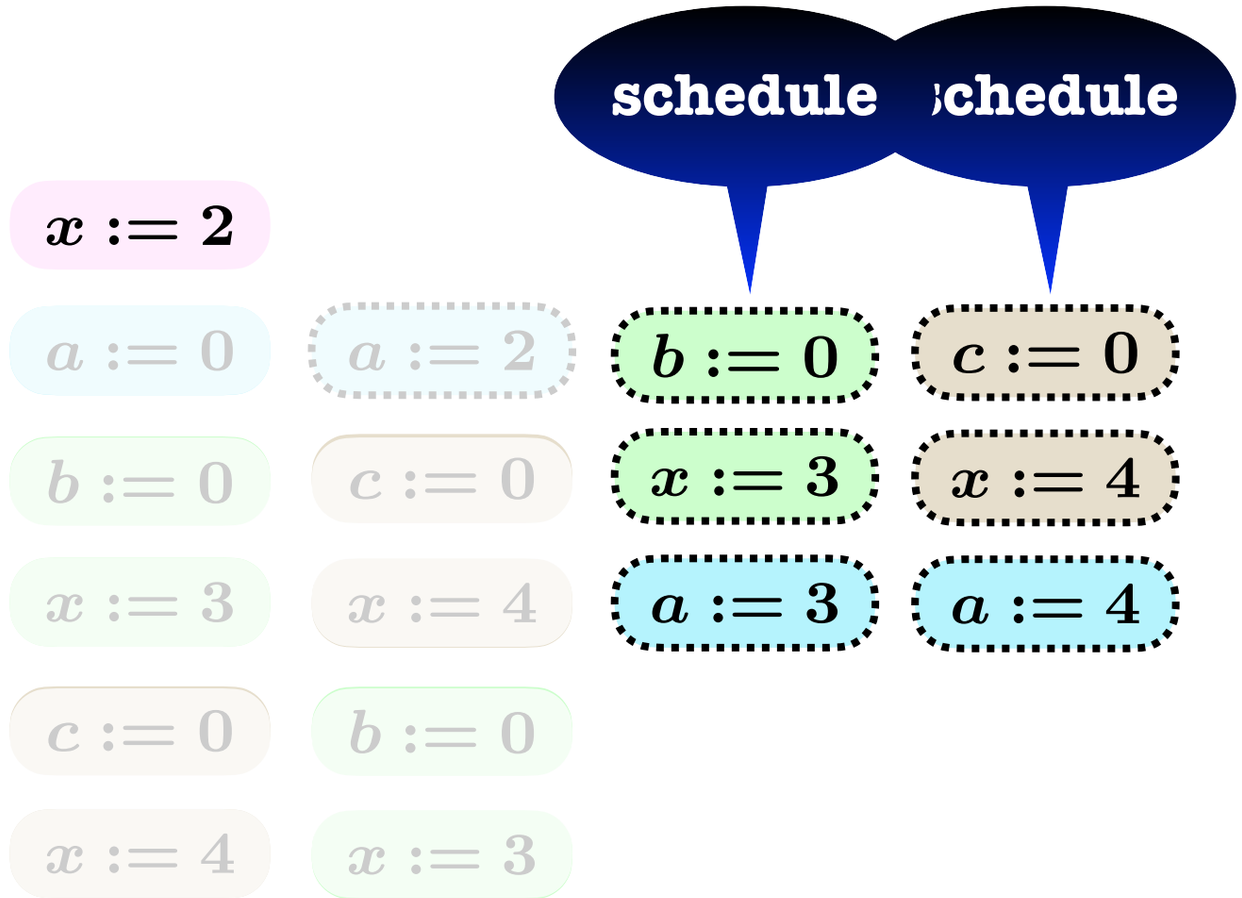
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$



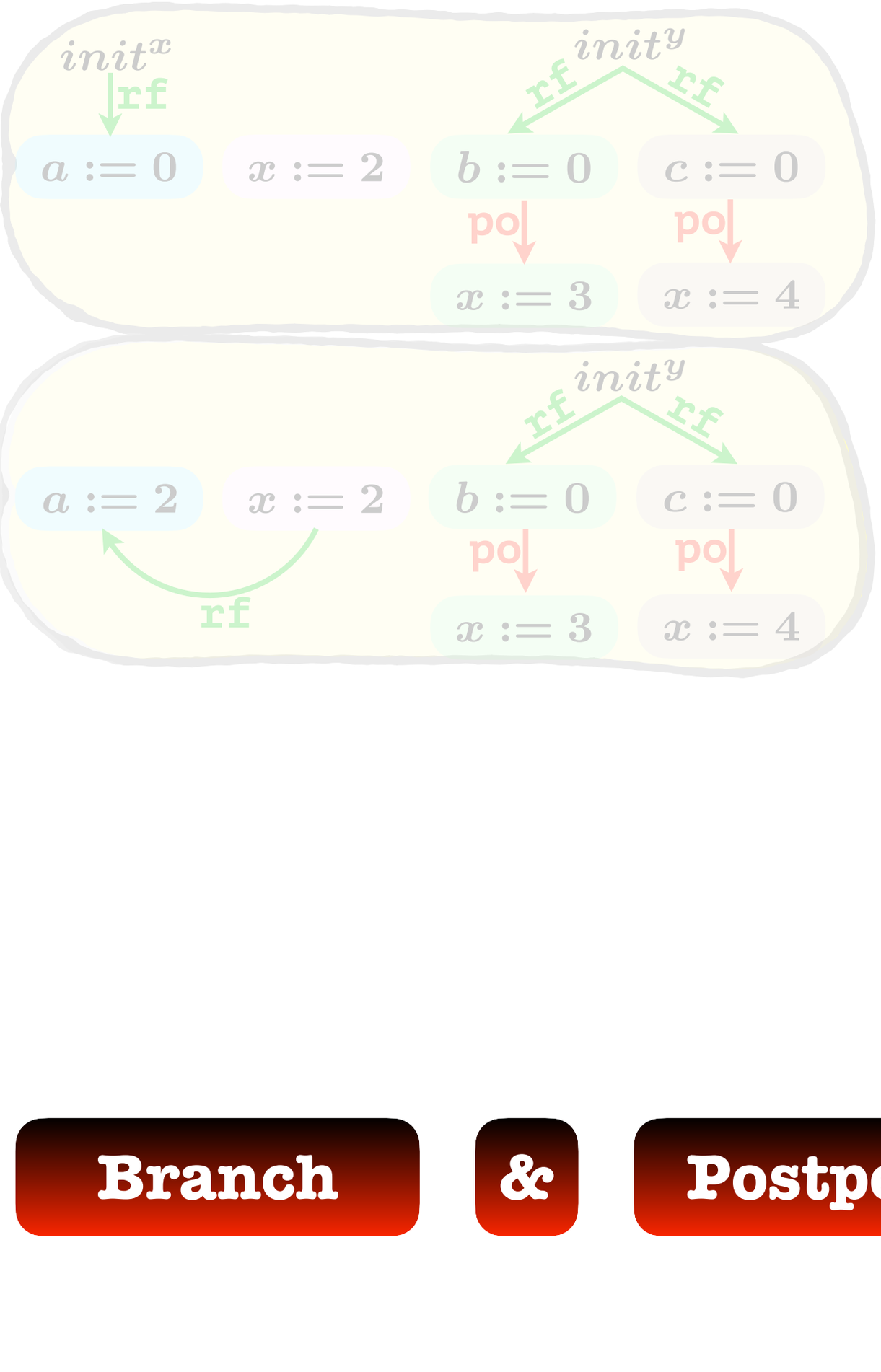
Branch **&** **Postpone**



$a := x \;\|\; x := 2 \;\|\; b := y,\ x := 3 \;\|\; c := y,\ x := 4$



Branch & Postpone



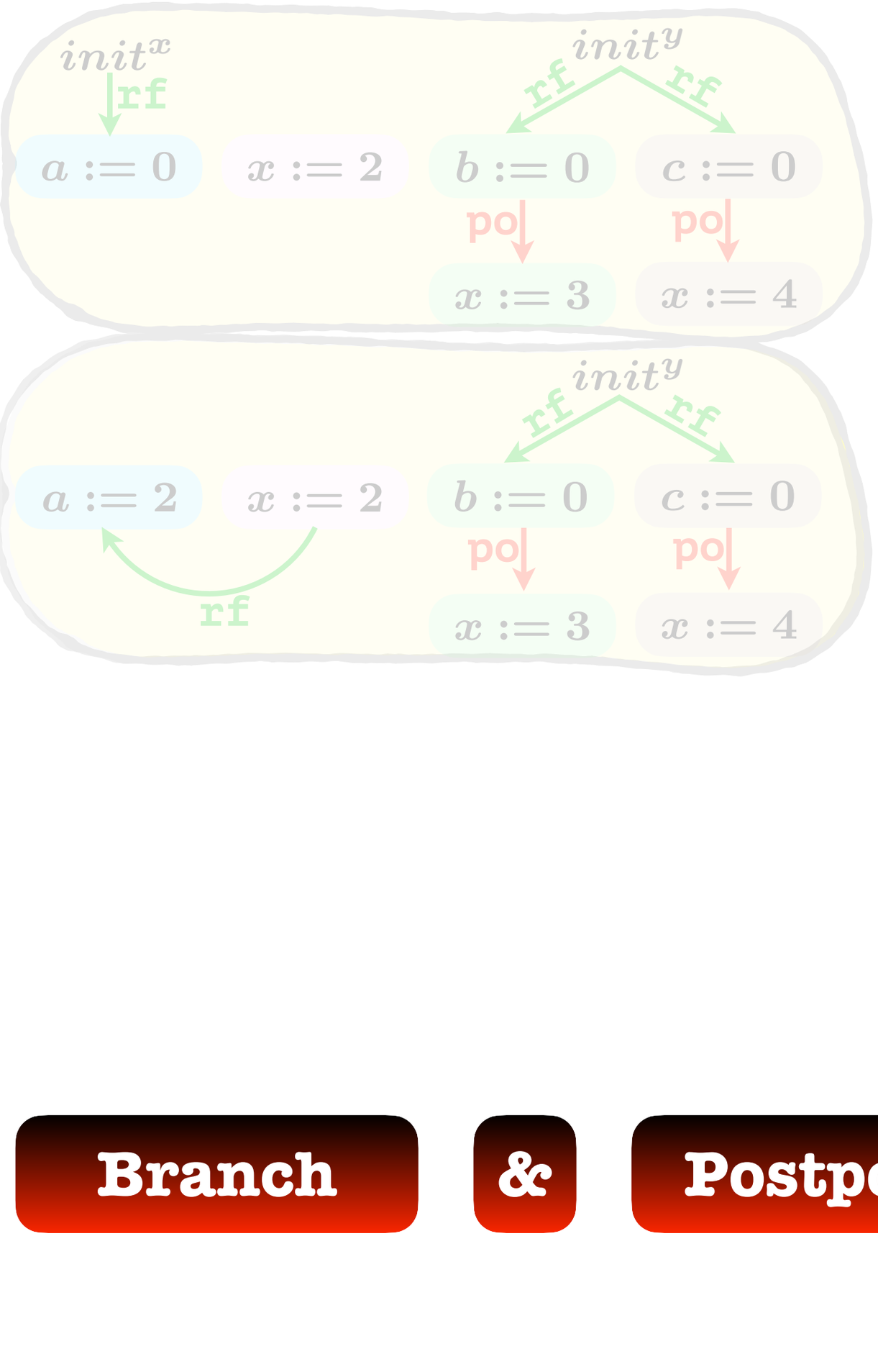
$a := x \;\|\; x := 2 \;\|\; b := y,\ x := 3 \;\|\; c := y,\ x := 4$

schedule schedule

$x := 2$

$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$		
$x := 4$	$x := 3$		

Branch & Postpone



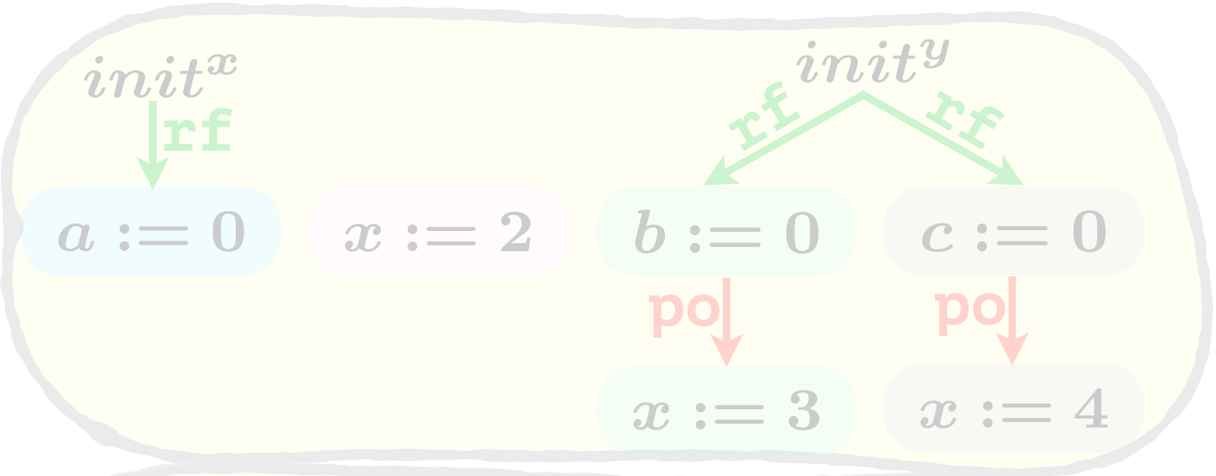
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

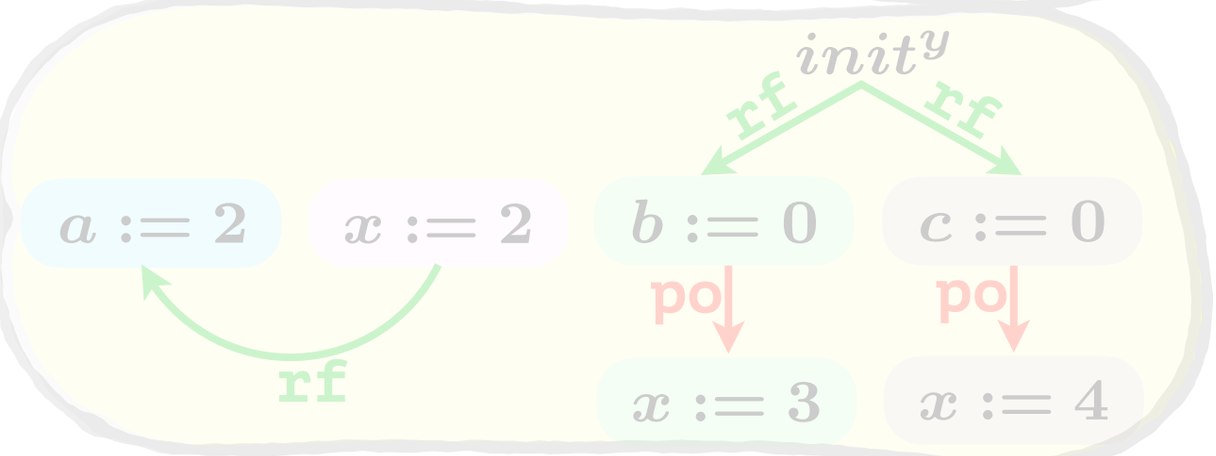
schedule

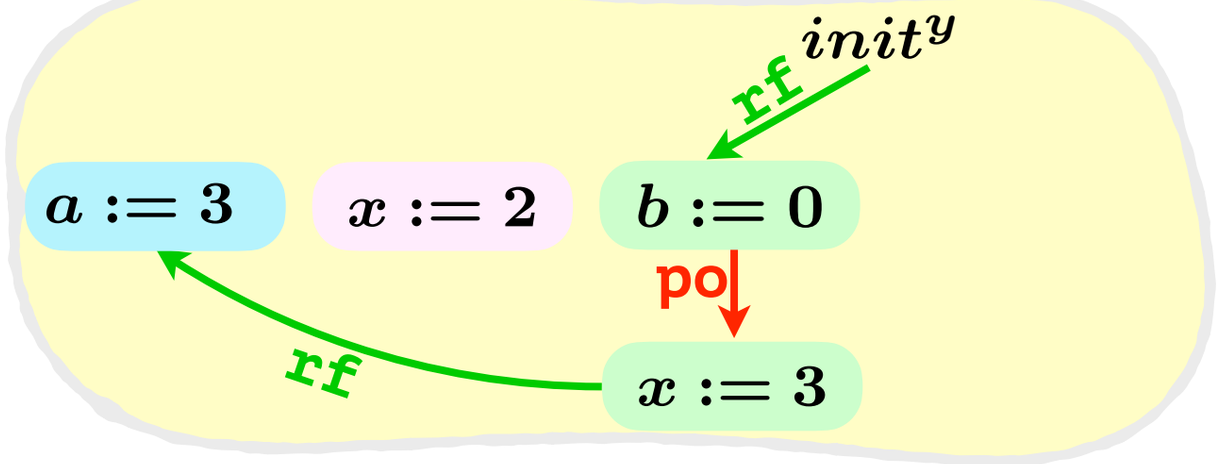
$x := 2$

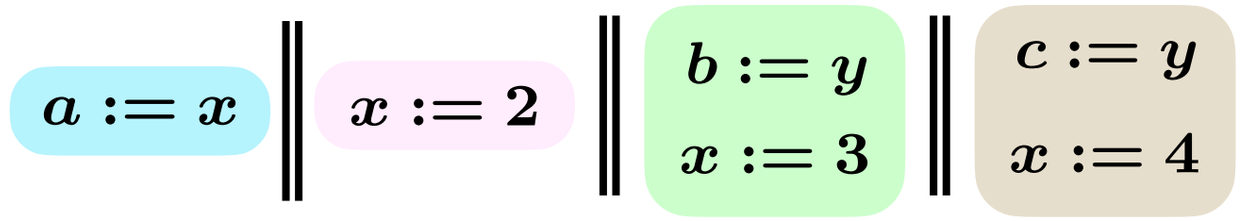
$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$		
$x := 4$	$x := 3$		

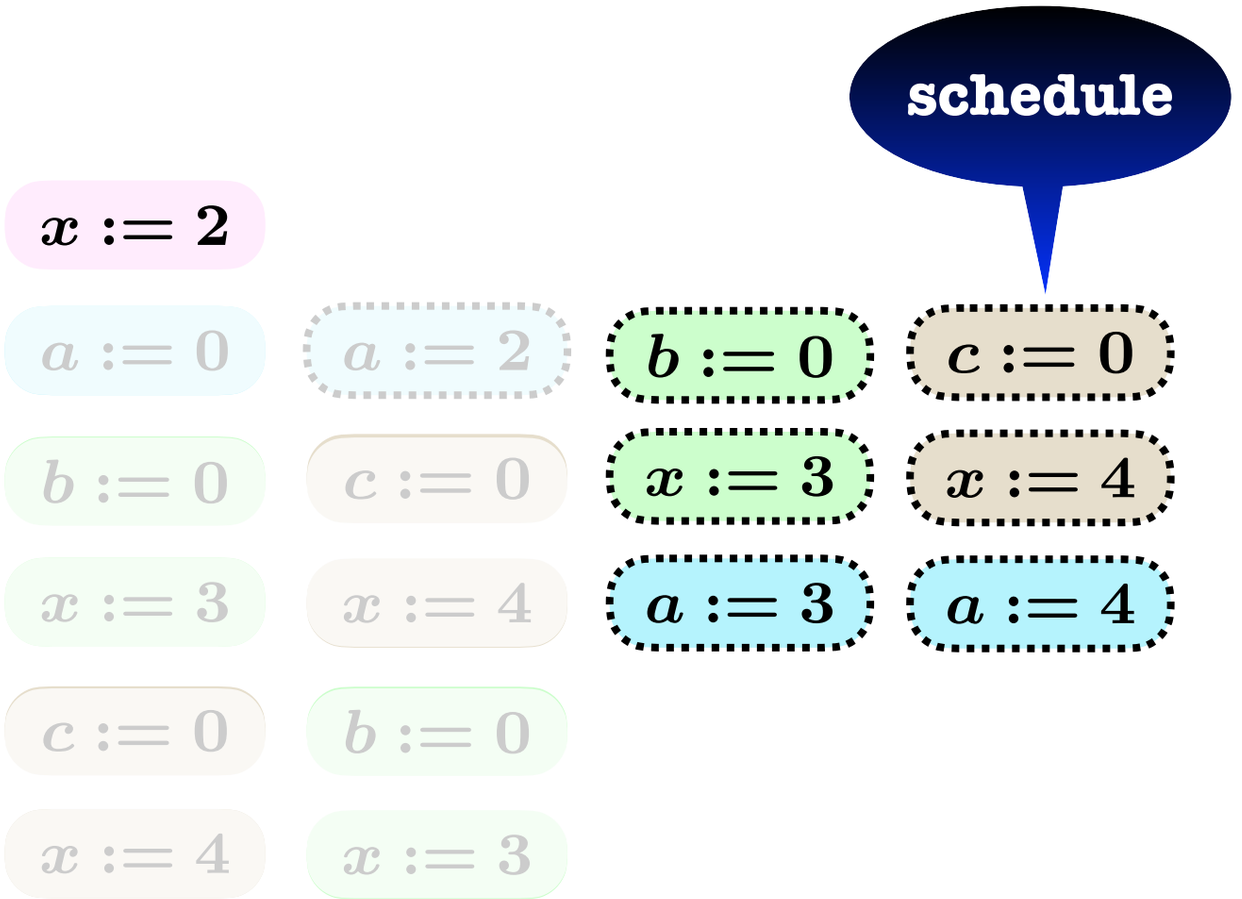
Branch **&** **Postpone**



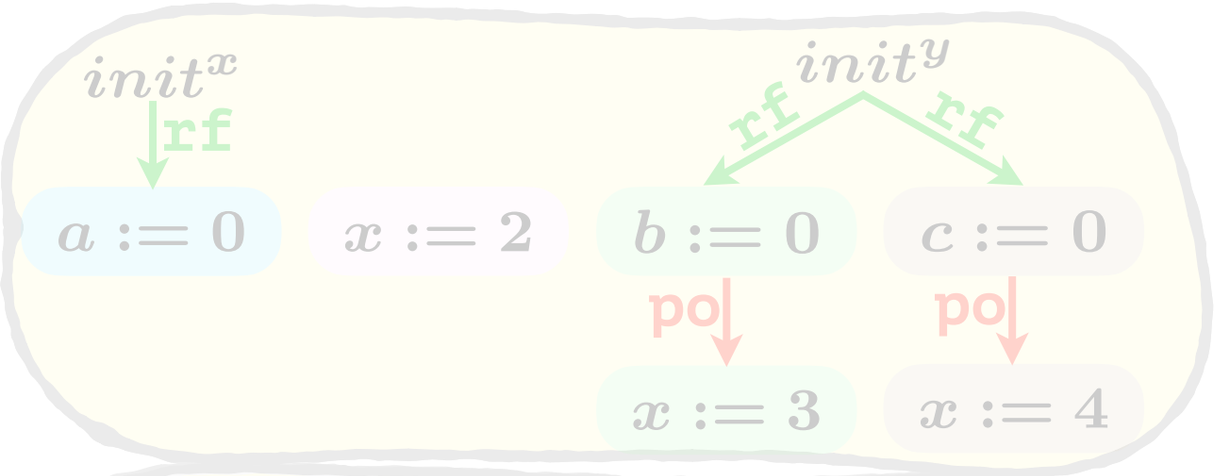


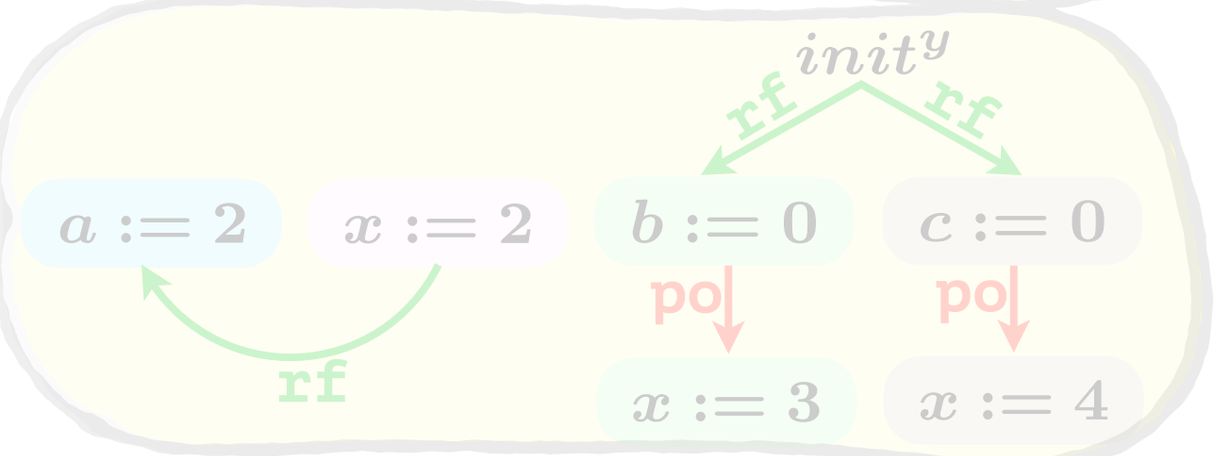


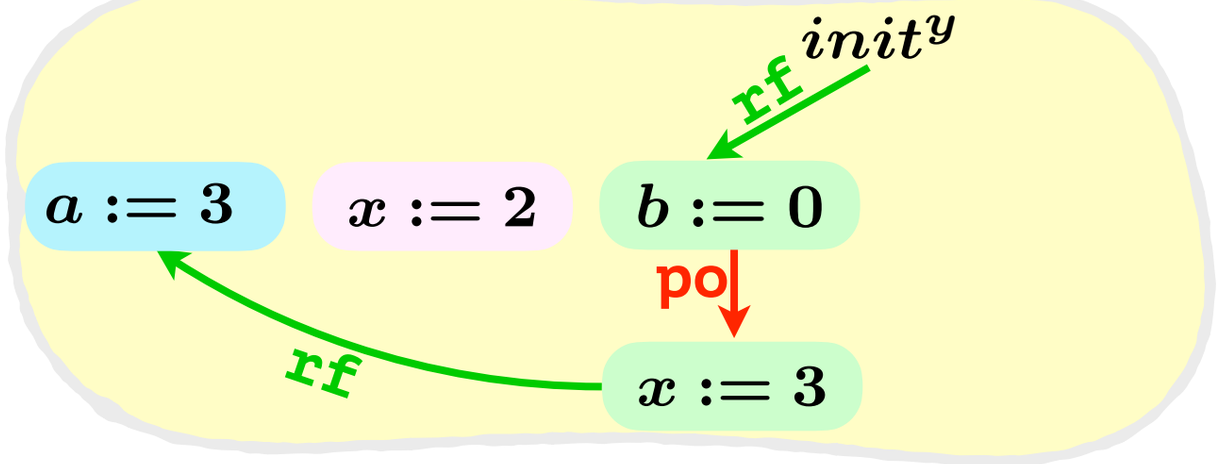


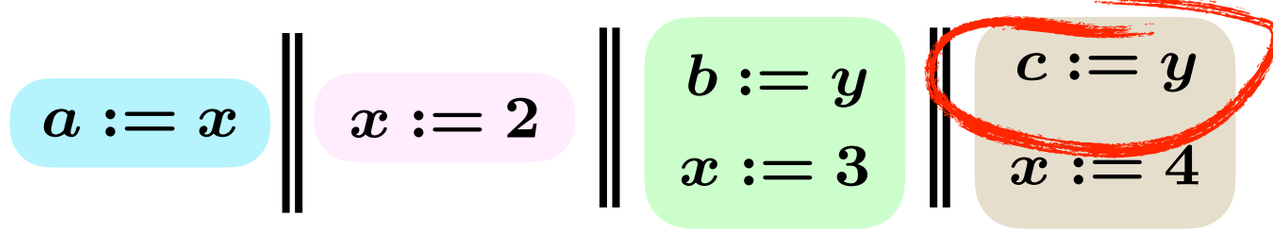


Branch **&** **Postpone**

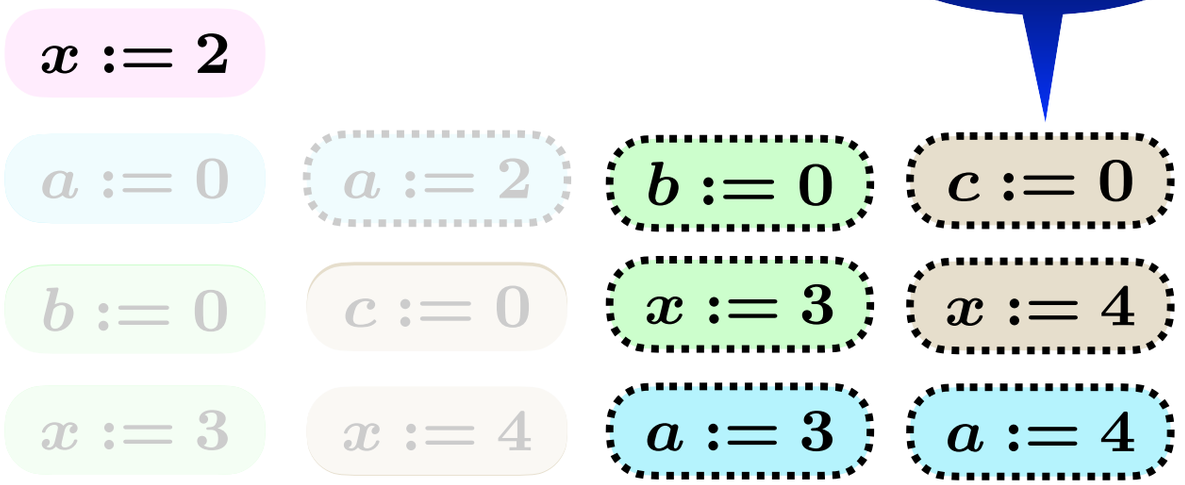




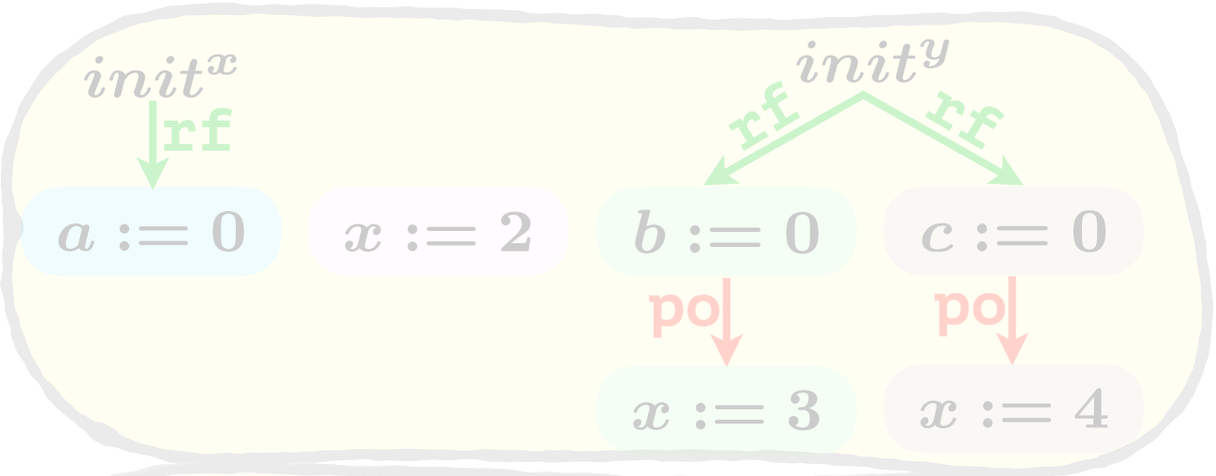


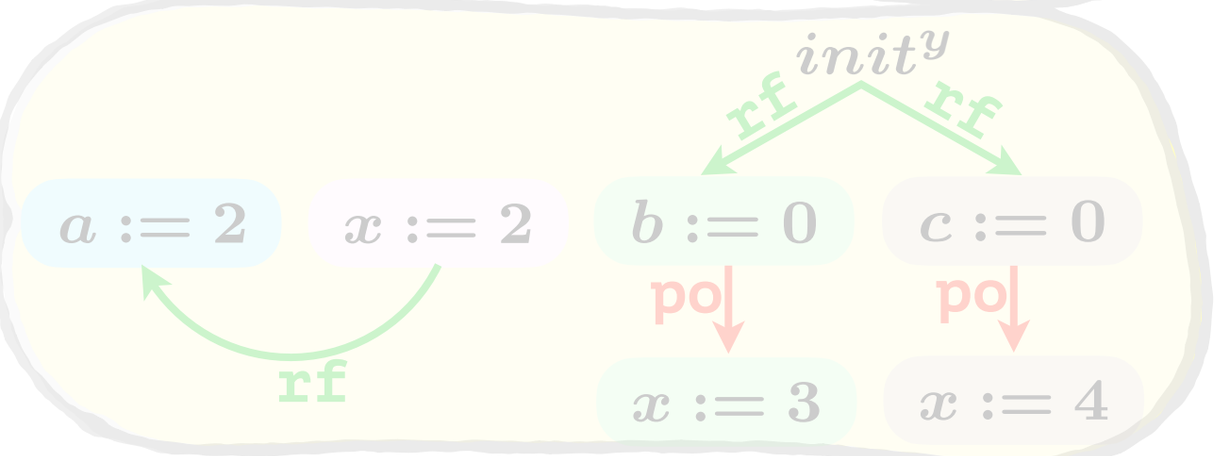


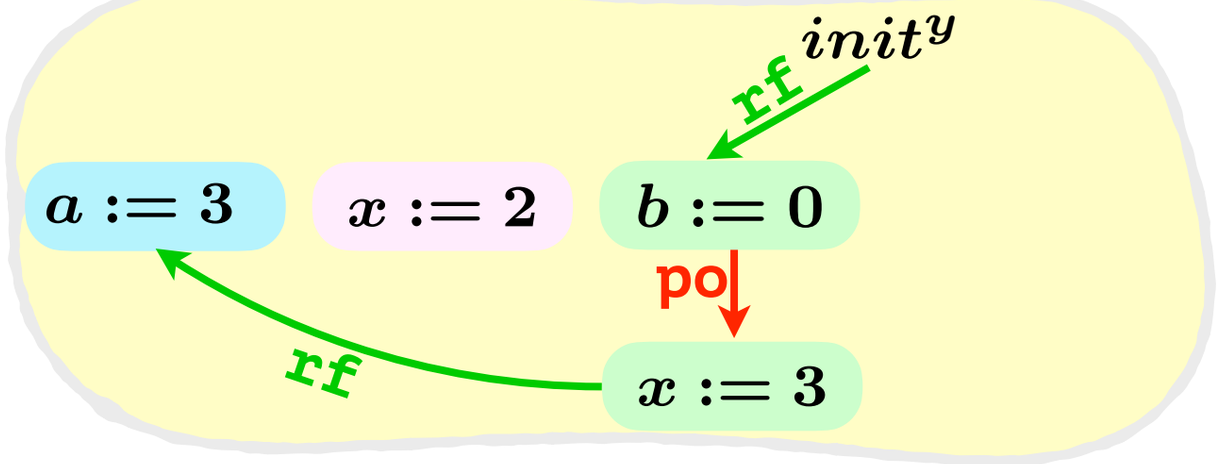
schedule

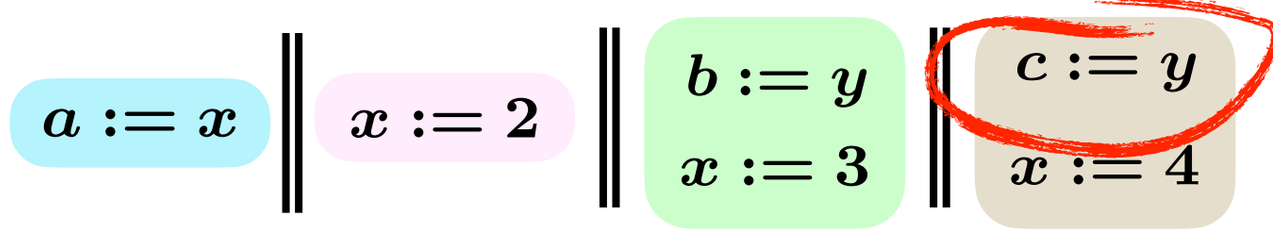


Branch **&** **Postpone**

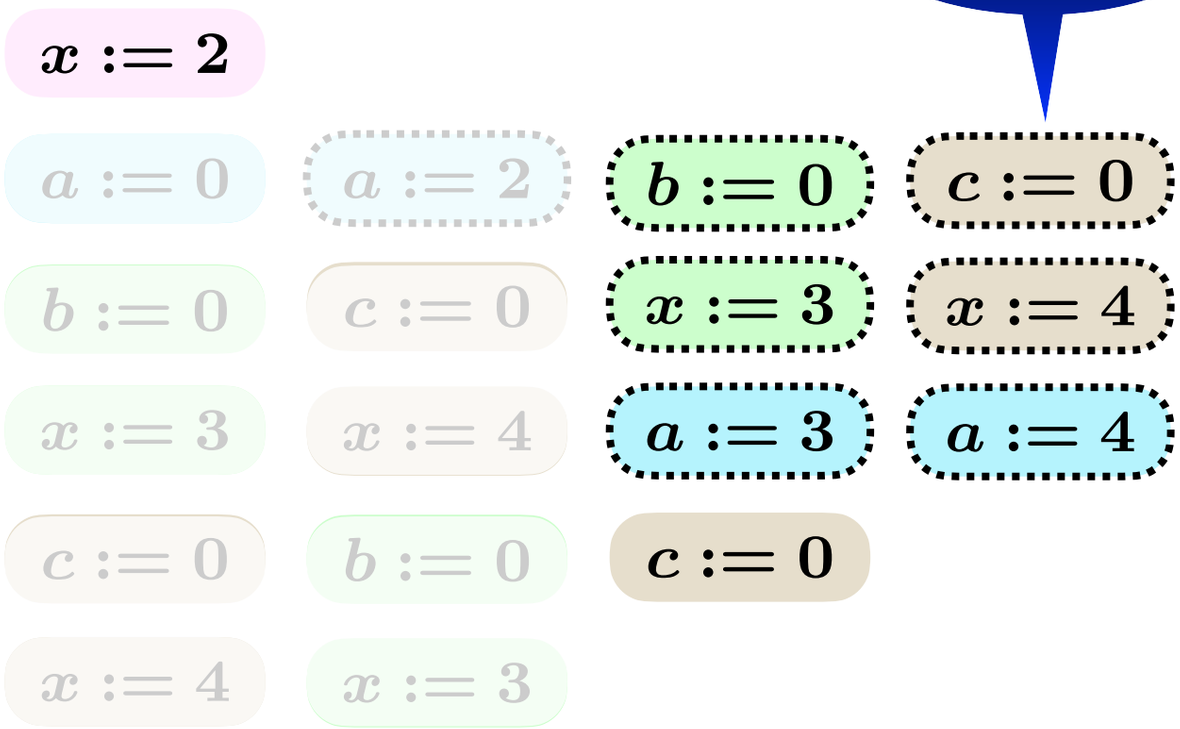




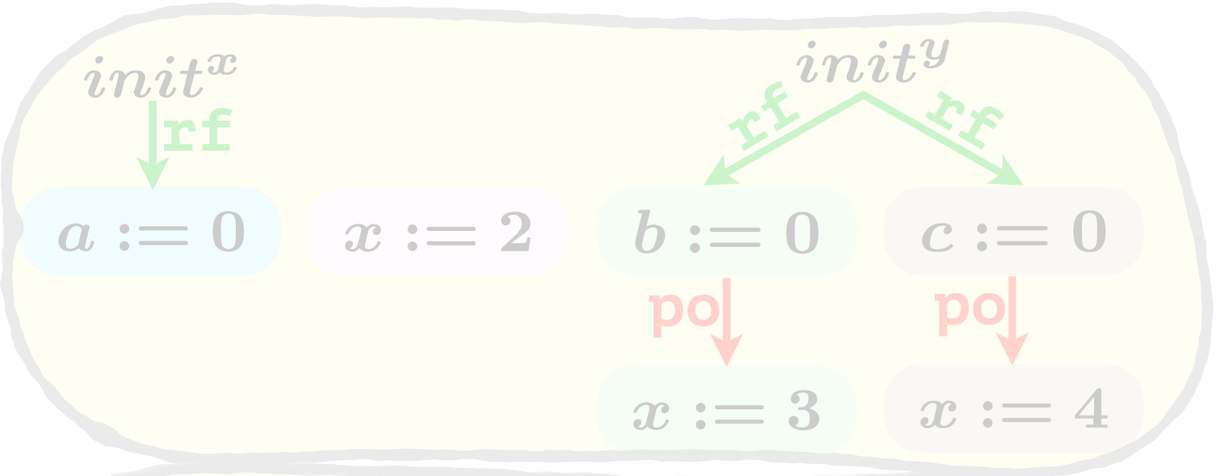


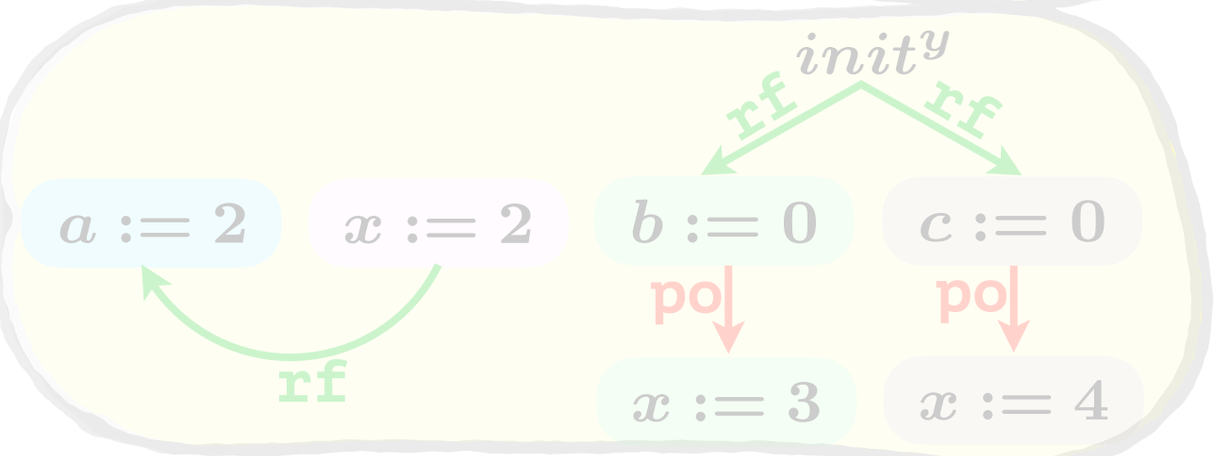


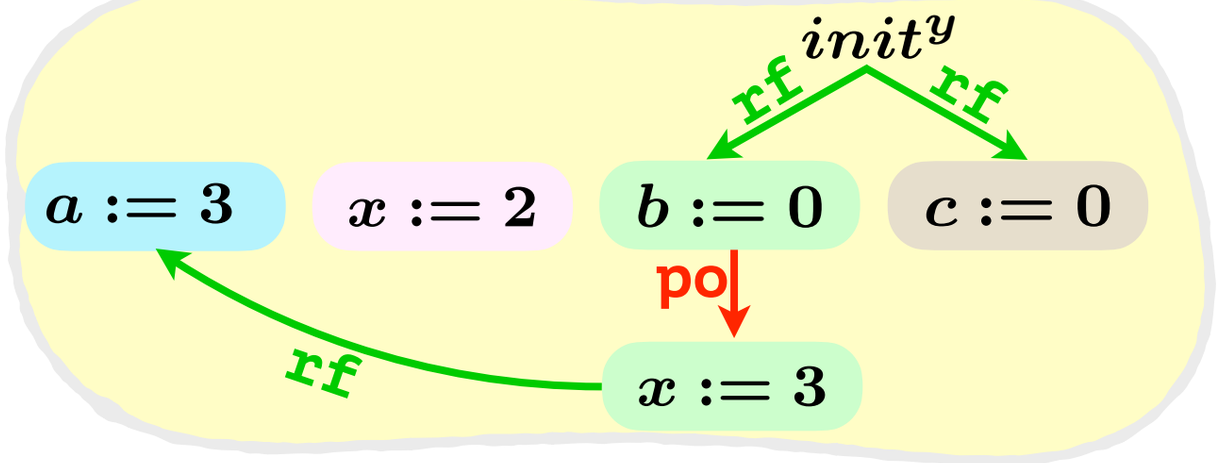
schedule

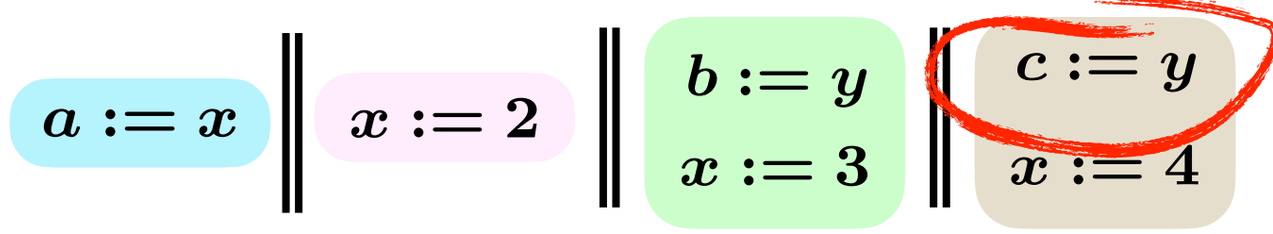


Branch **&** **Postpone**

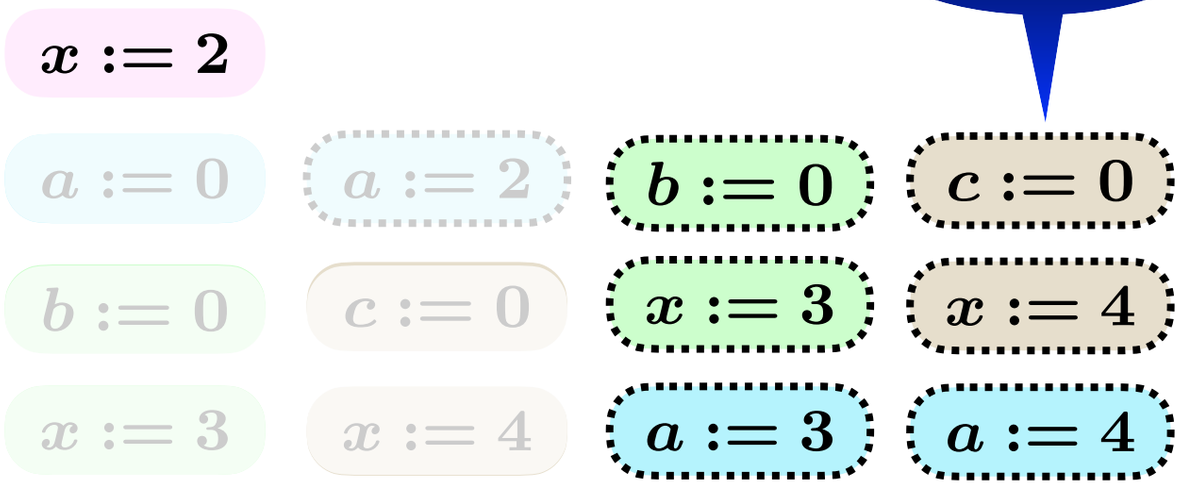


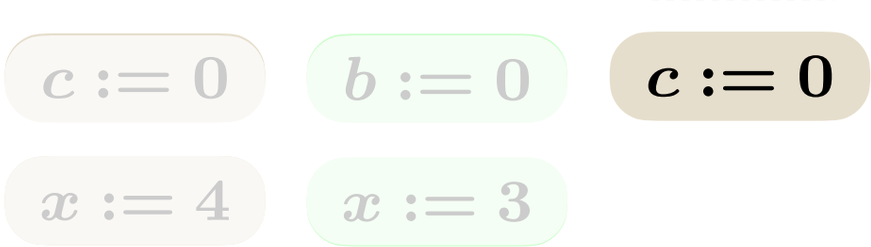




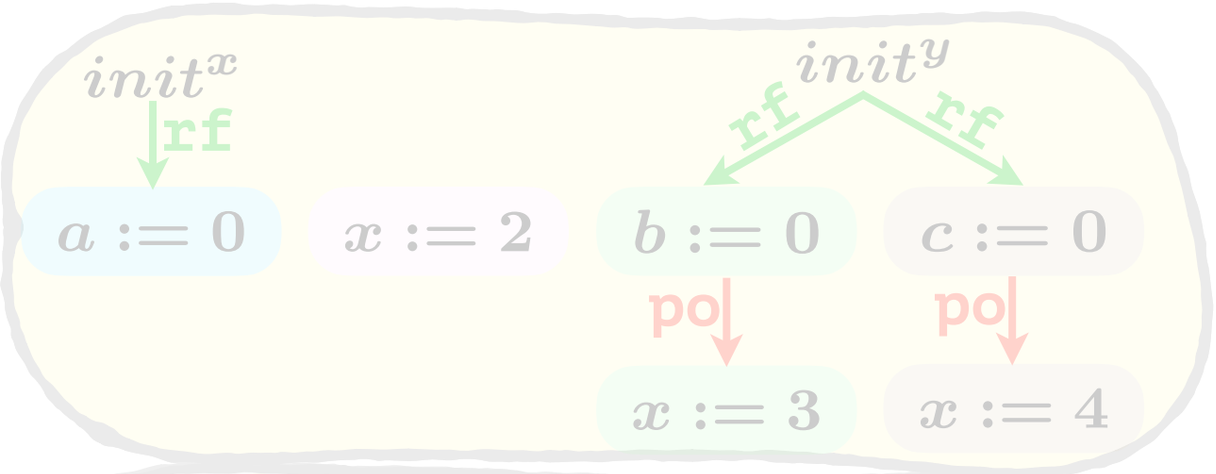


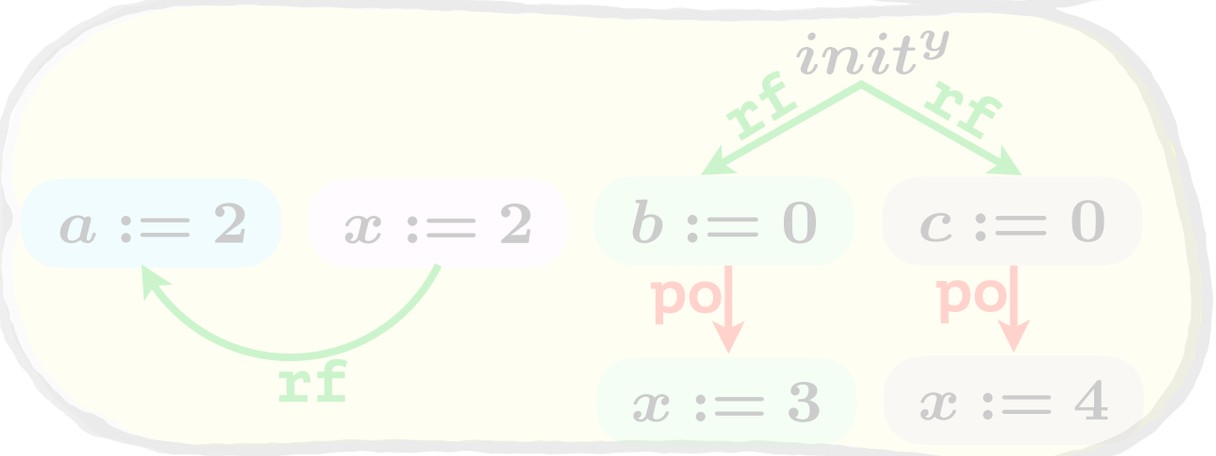
schedule

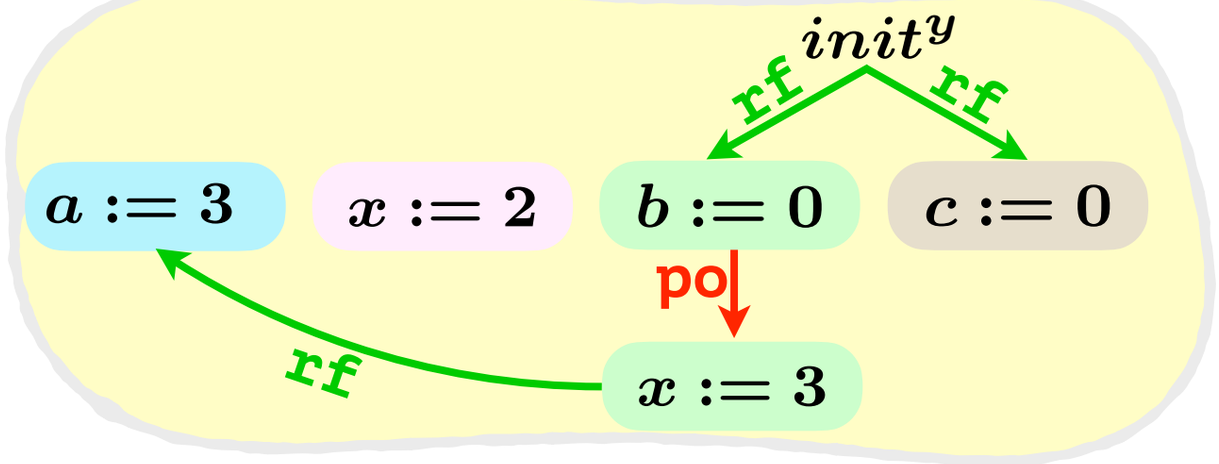


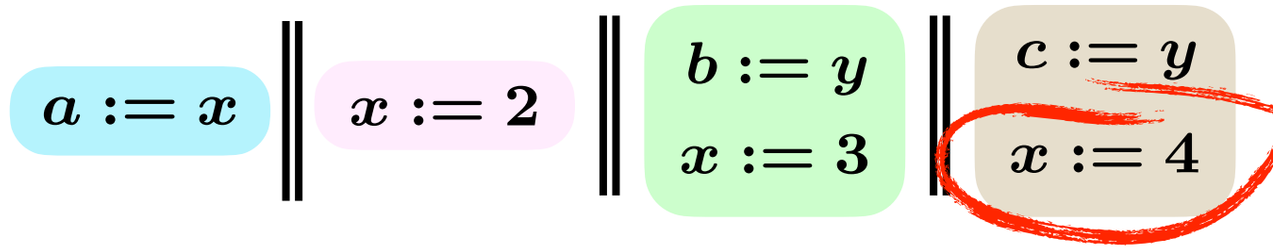


Branch **&** **Postpone**

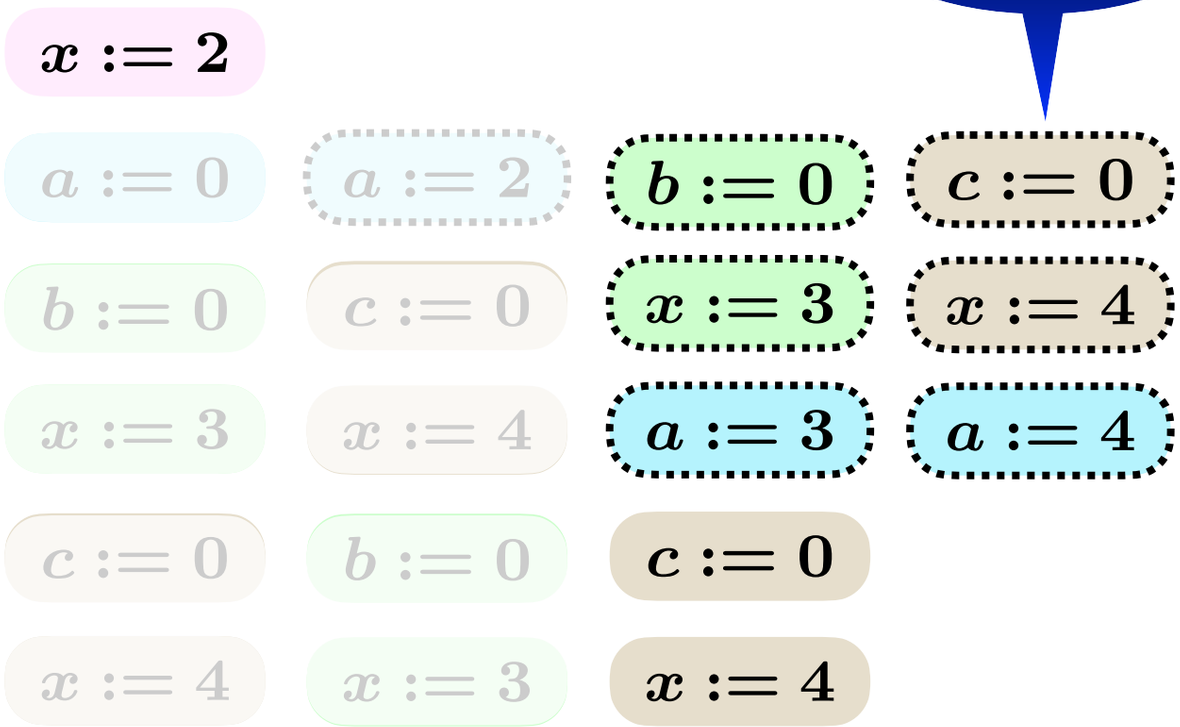




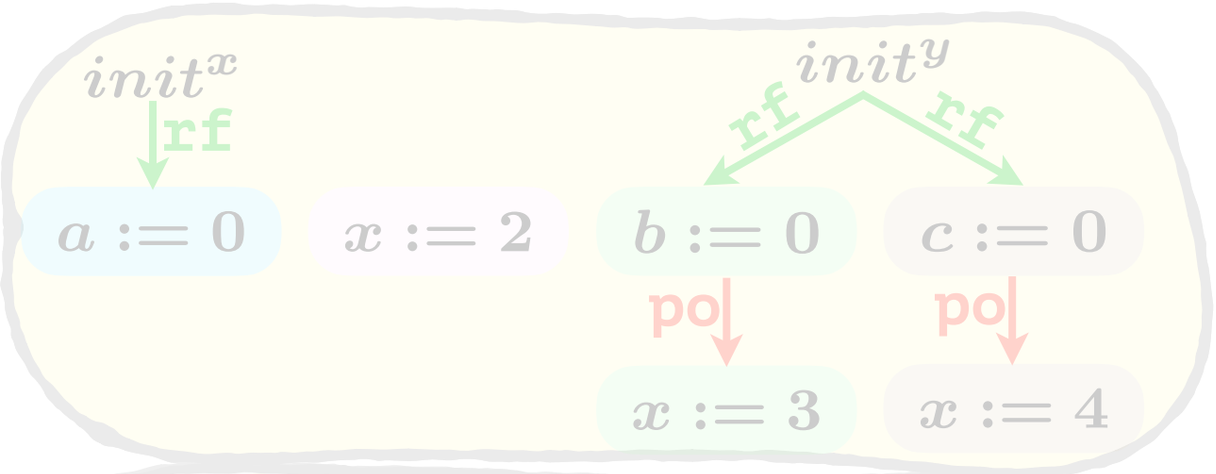


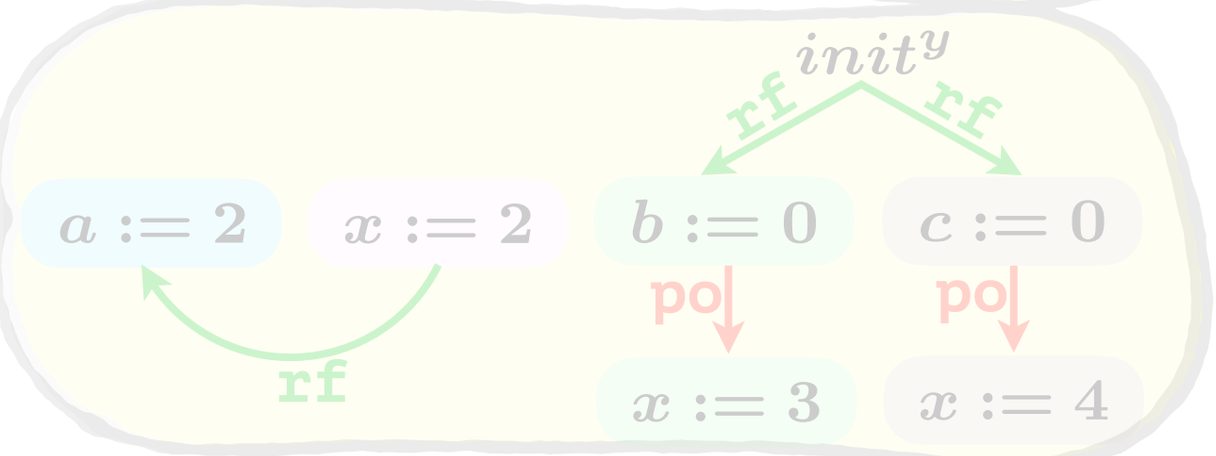


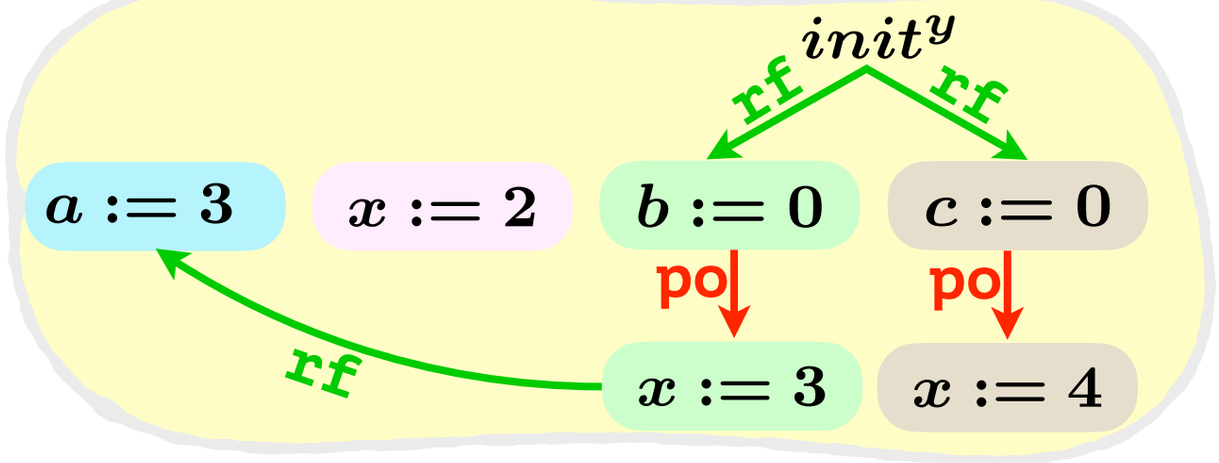
schedule

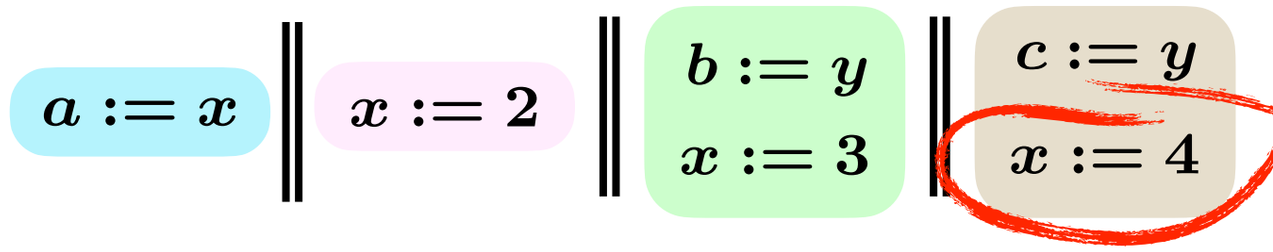


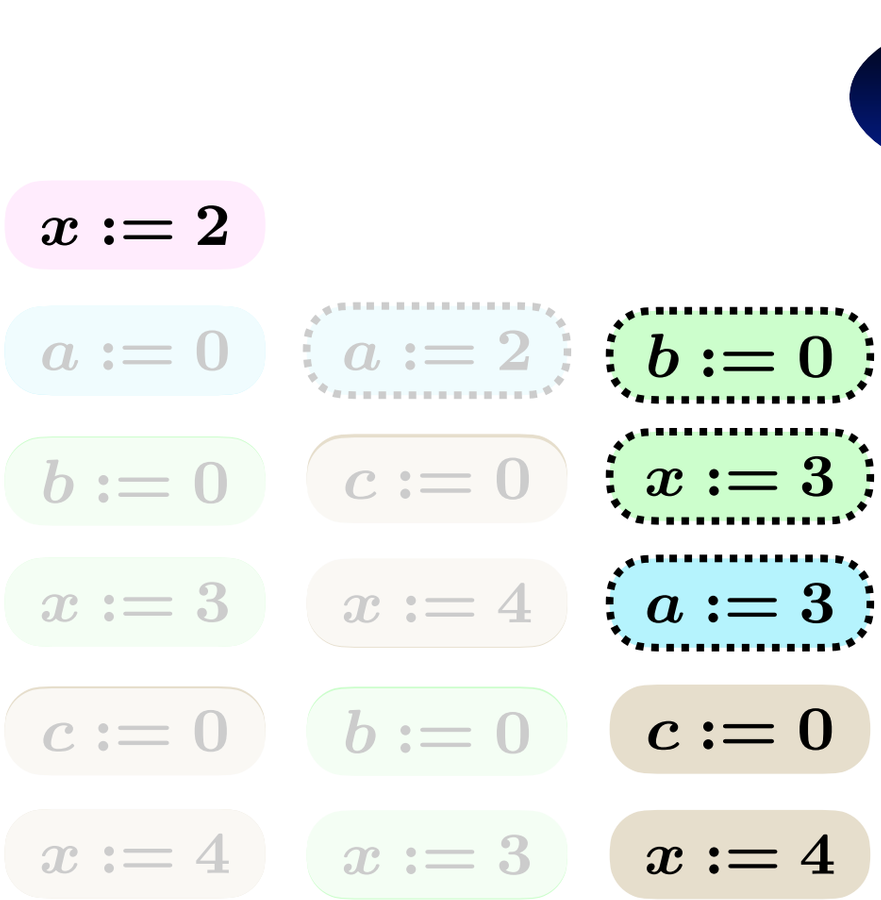
Branch **&** **Postpone**



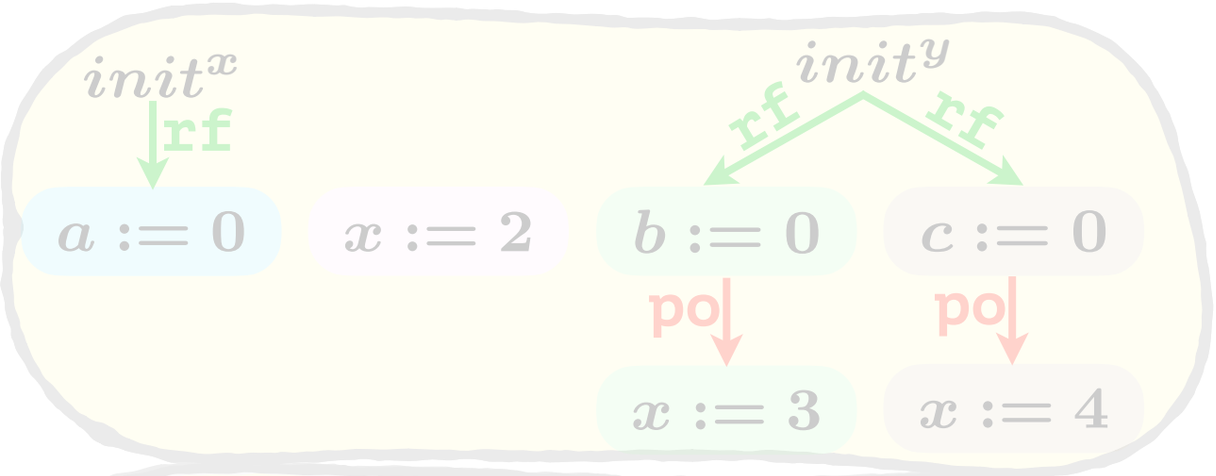


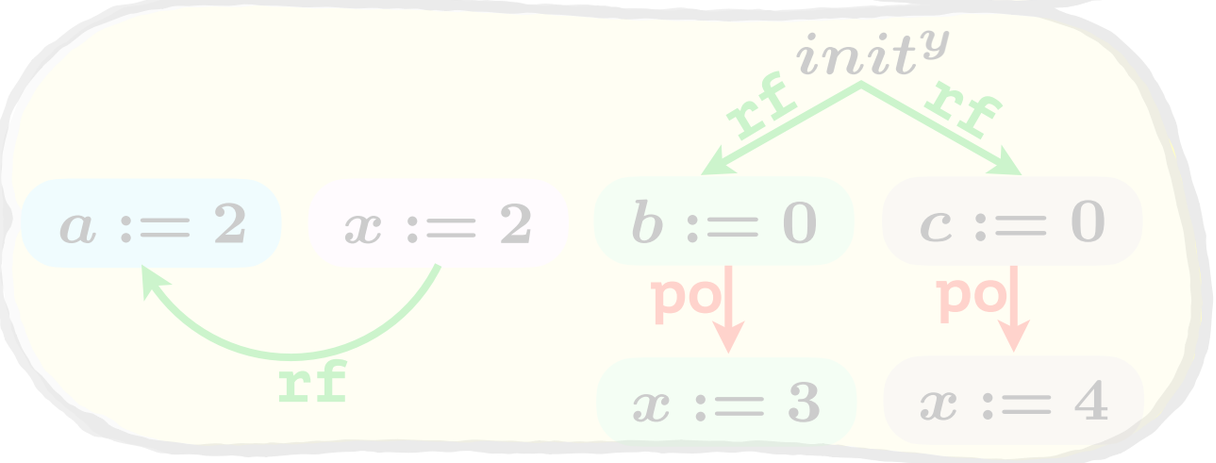


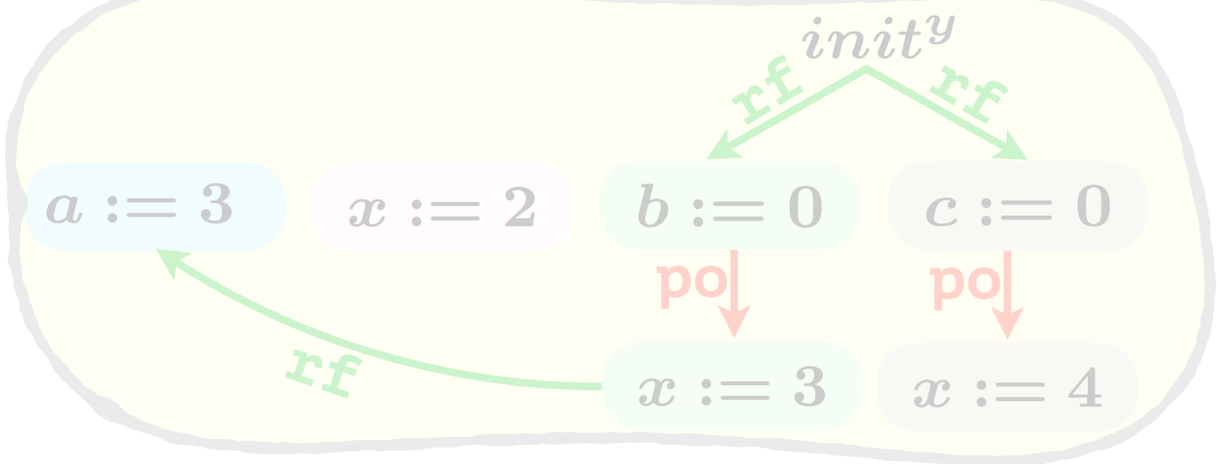




Branch **&** **Postpone**





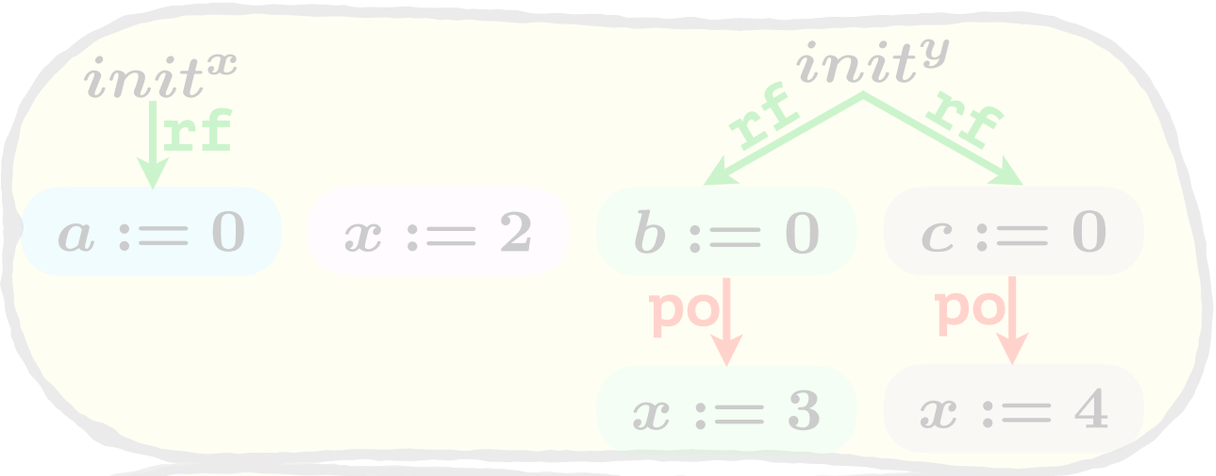


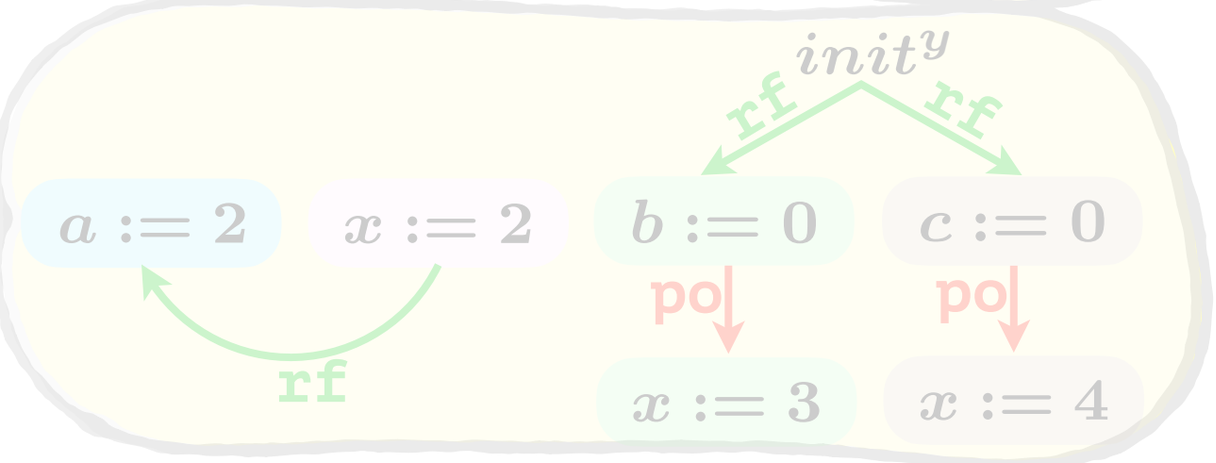
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

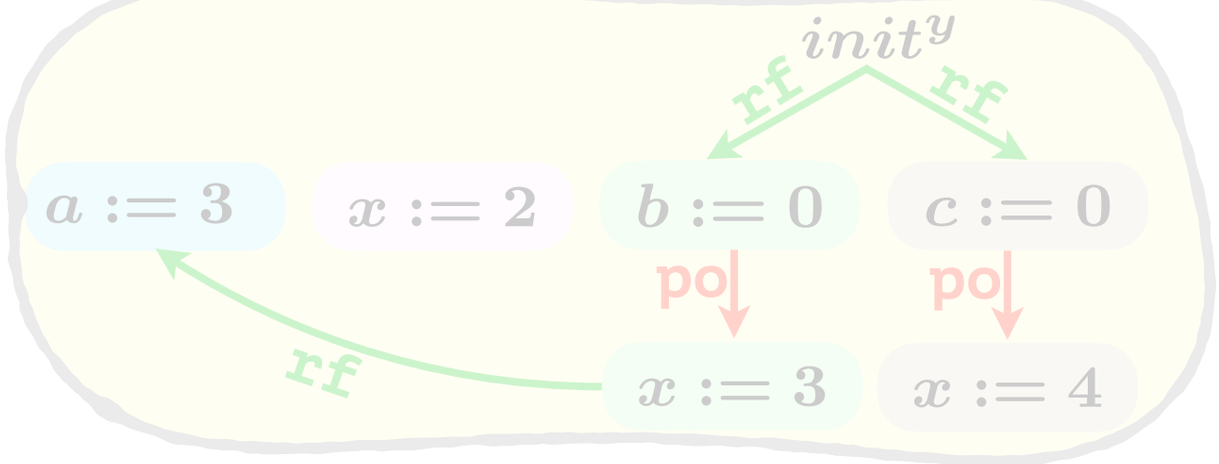
schedule

$x := 2$			
$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$	$c := 0$	
$x := 4$	$x := 3$	$x := 4$	

Branch **&** **Postpone**







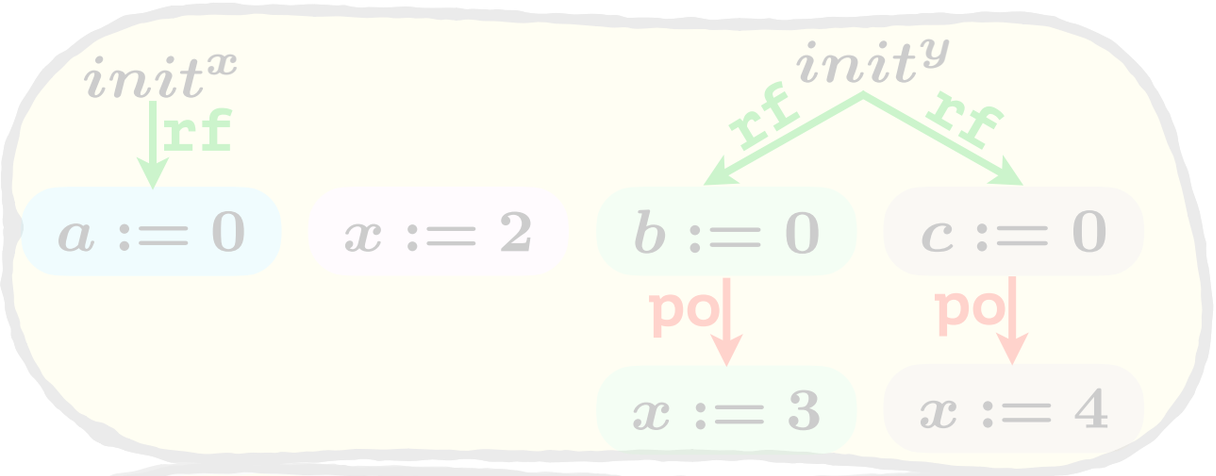
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

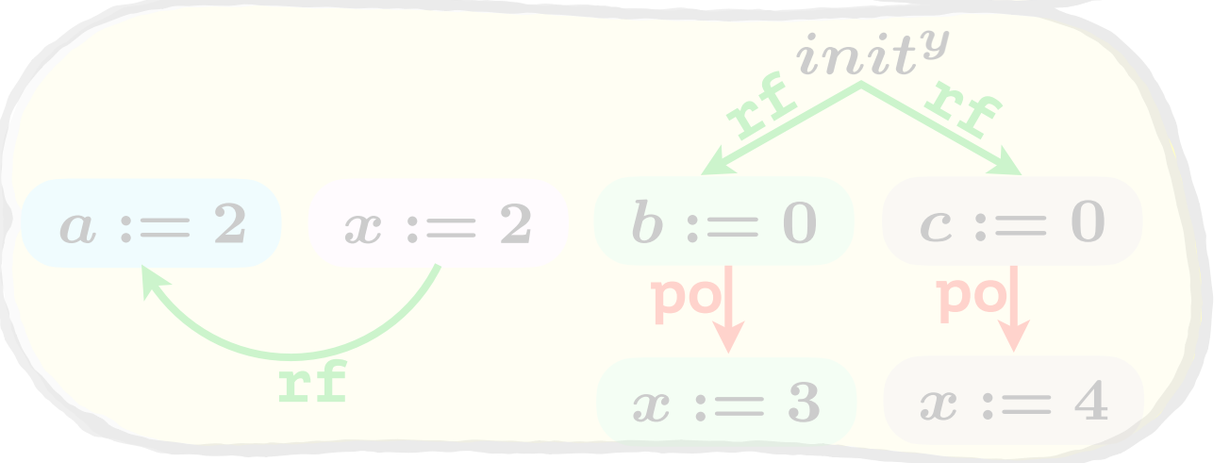
schedule

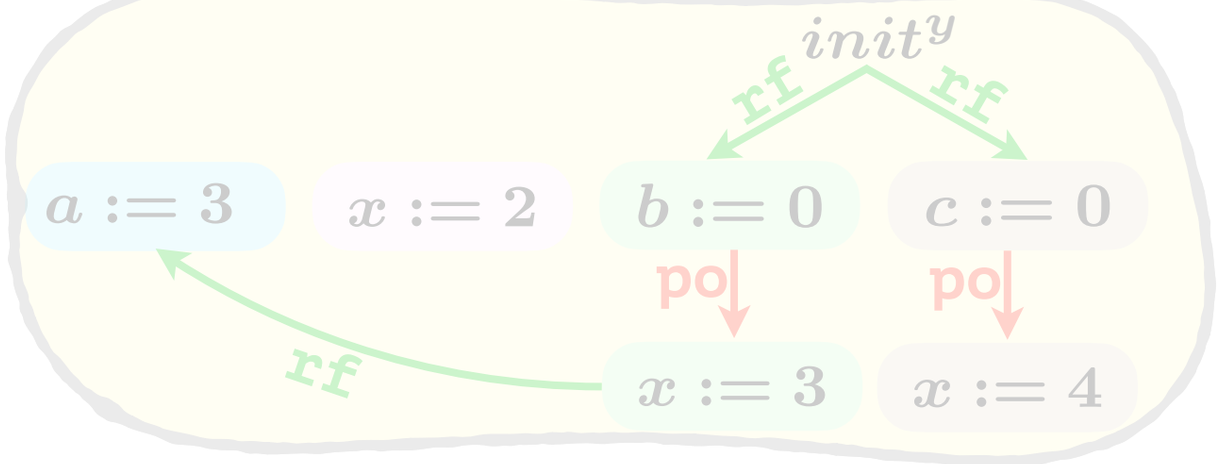
$x := 2$

$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$	$c := 0$	
$x := 4$	$x := 3$	$x := 4$	

Branch **&** **Postpone**





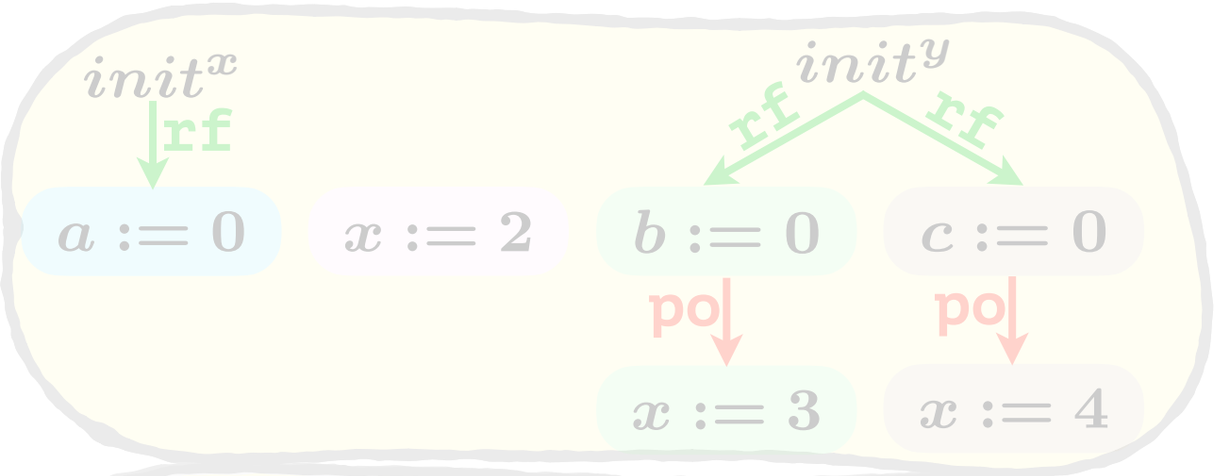


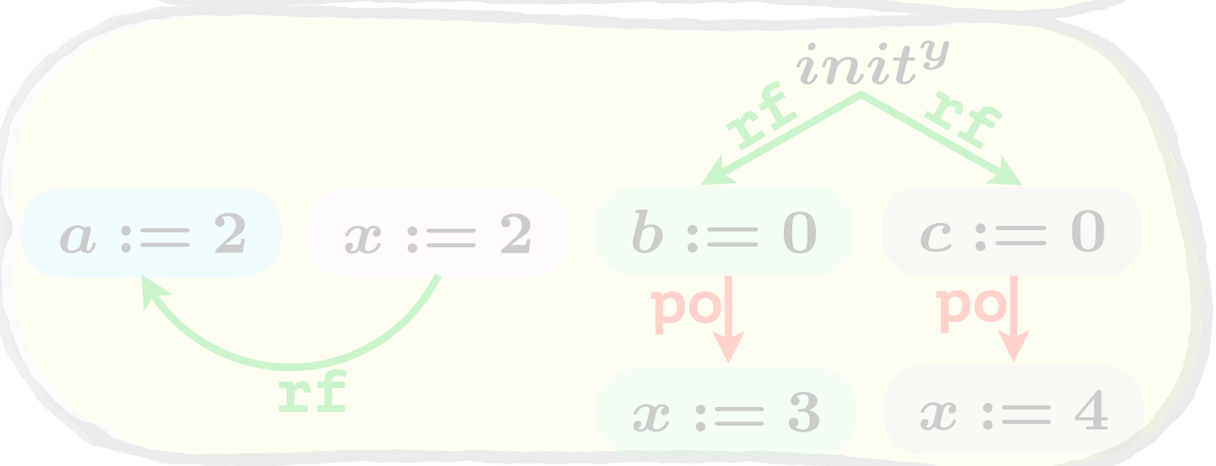
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

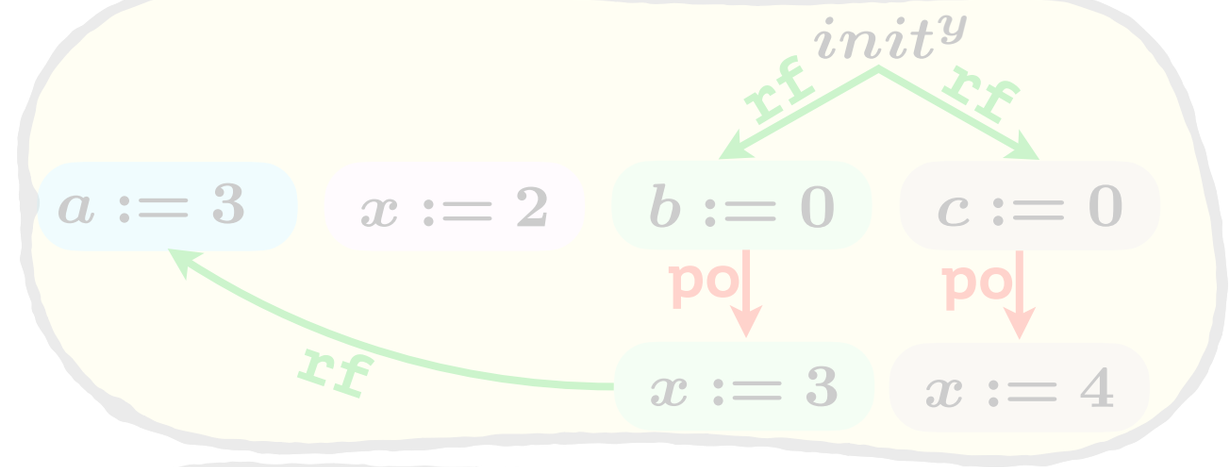
$x := 2$

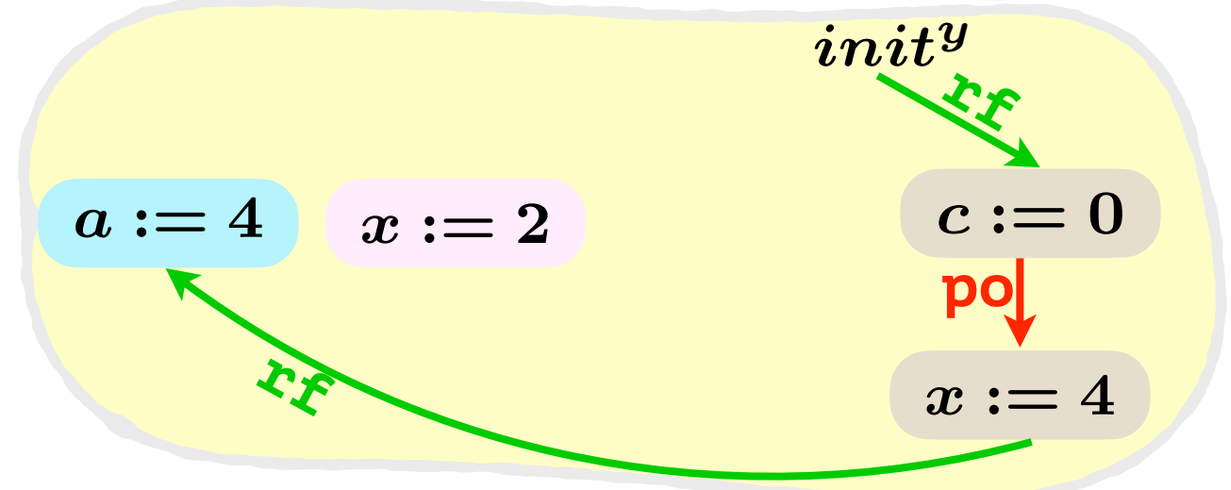
$a := 0$	$a := 2$	$b := 0$	**$c := 0$**
$b := 0$	$c := 0$	$x := 3$	**$x := 4$**
$x := 3$	$x := 4$	$a := 3$	**$a := 4$**
$c := 0$	$b := 0$	$c := 0$	
$x := 4$	$x := 3$	$x := 4$	

Branch & Postpone





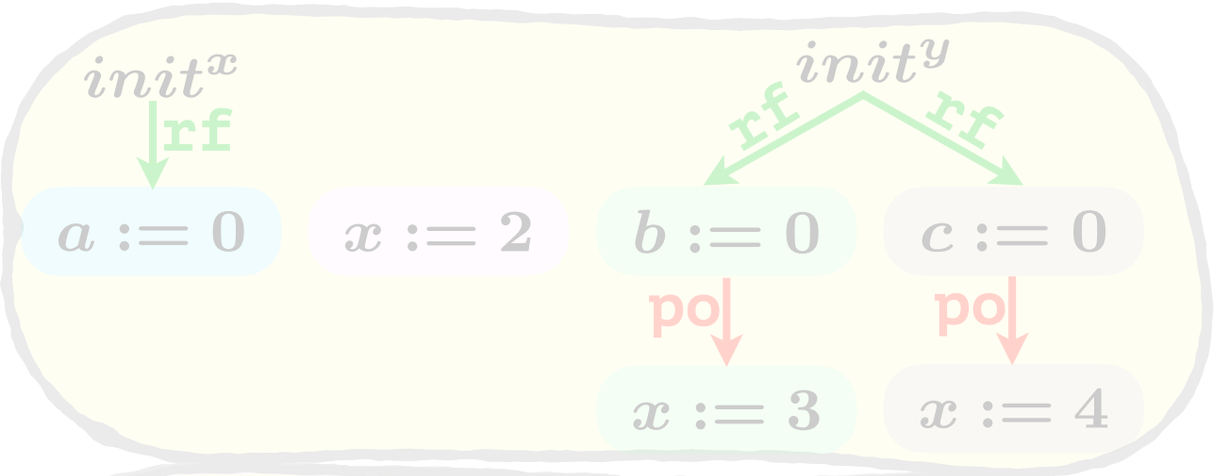


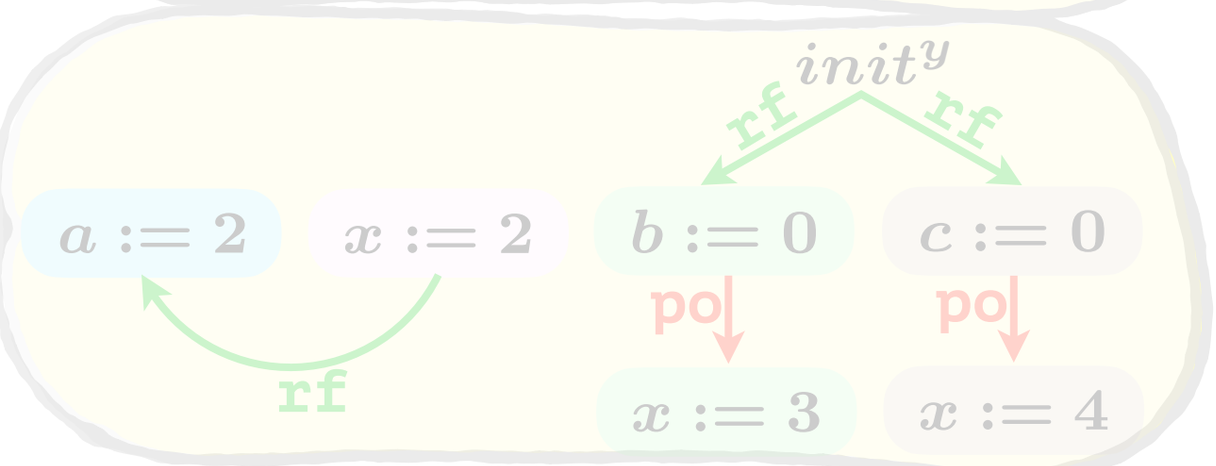


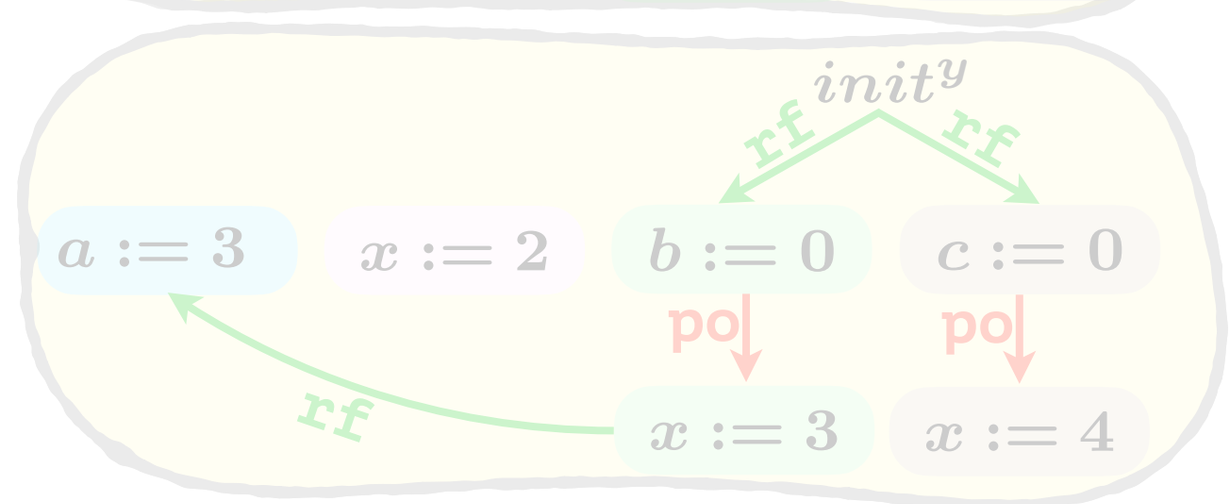
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

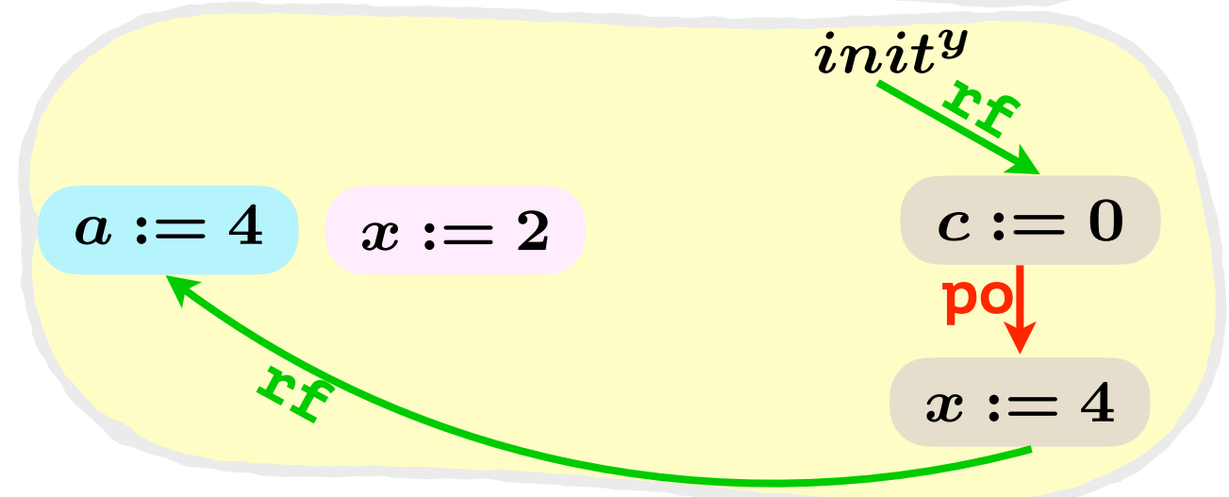
$x := 2$

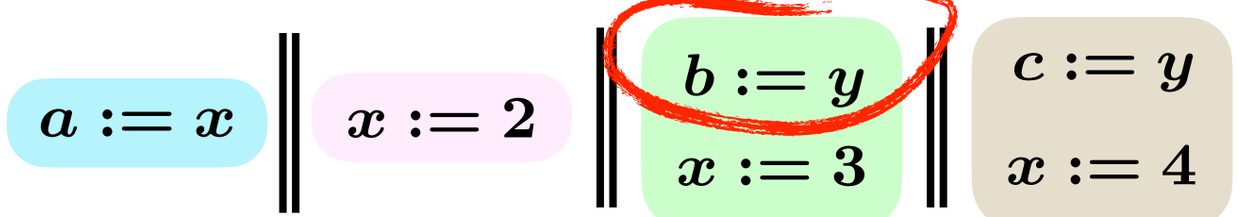
$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$	$c := 0$	
$x := 4$	$x := 3$	$x := 4$	

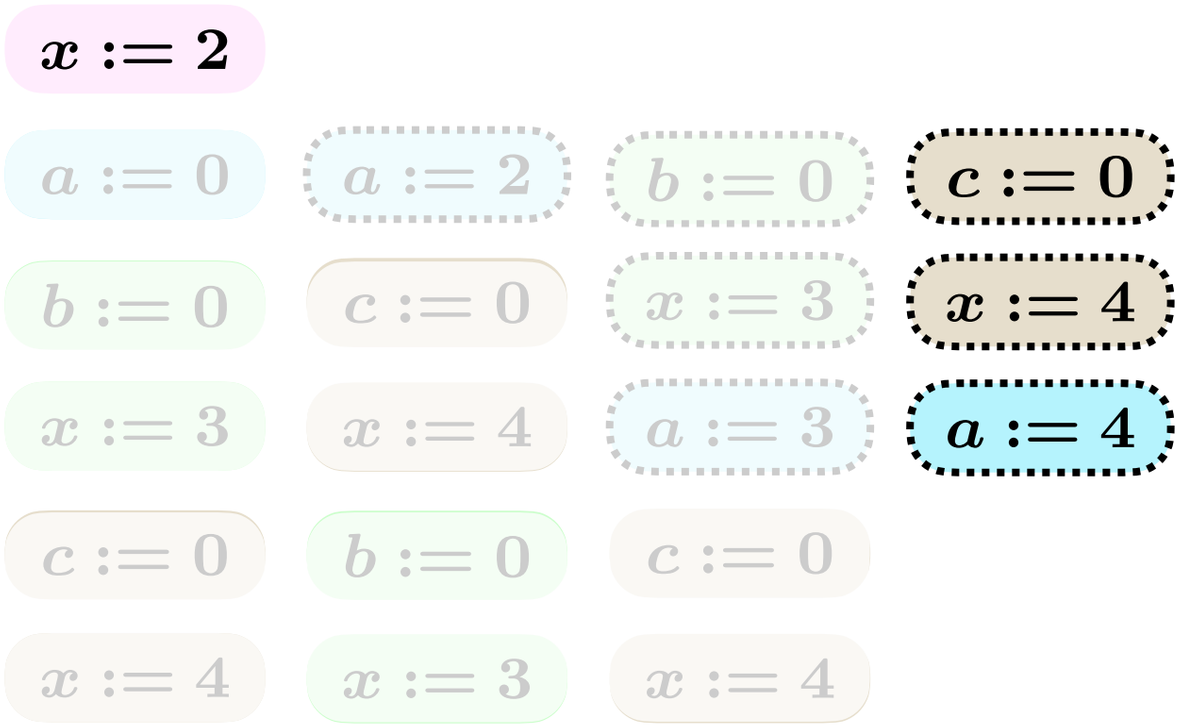

$init^x$
rf
$init^y$
rf
rf
$a := 0$
$x := 2$
$b := 0$
$c := 0$
po
po
$x := 3$
$x := 4$

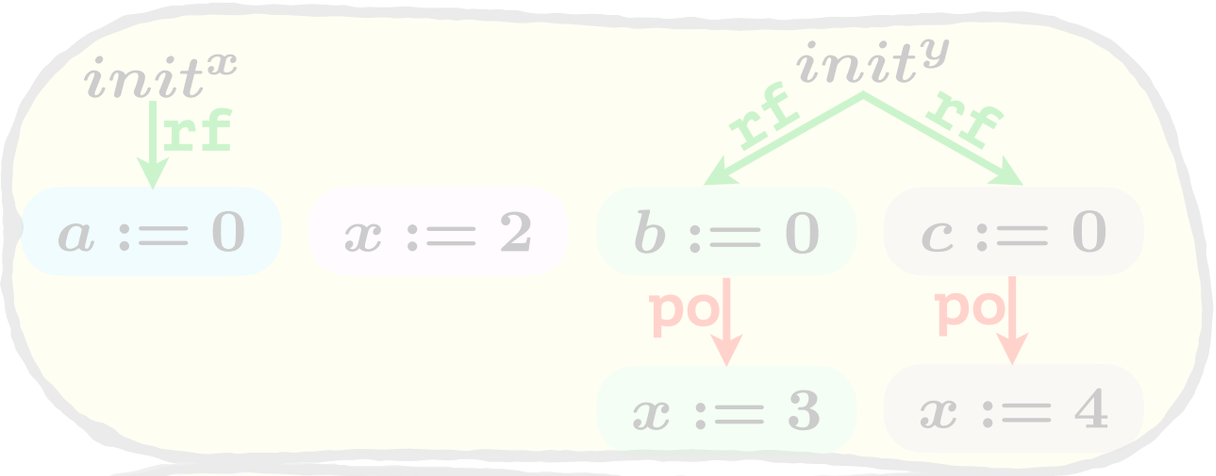

$init^y$
rf
rf
$a := 2$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

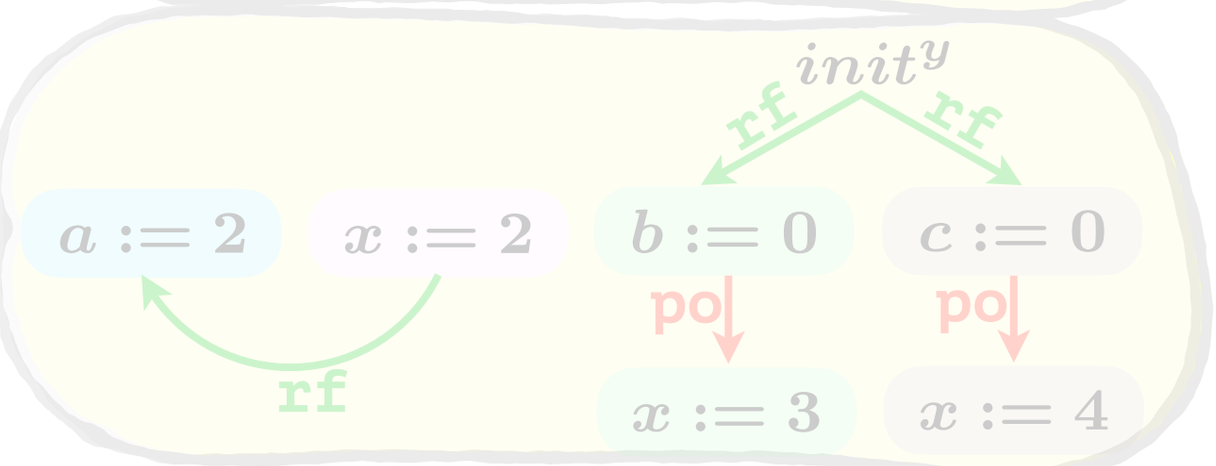

$init^y$
rf
rf
$a := 3$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

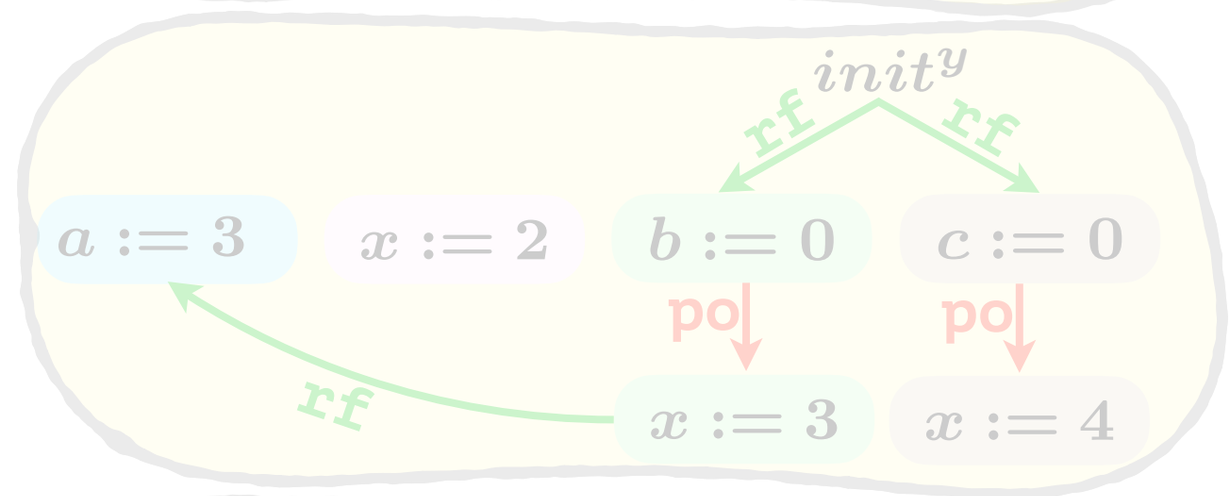

$init^y$
rf
$a := 4$
$x := 2$
$c := 0$
po
rf
$x := 4$

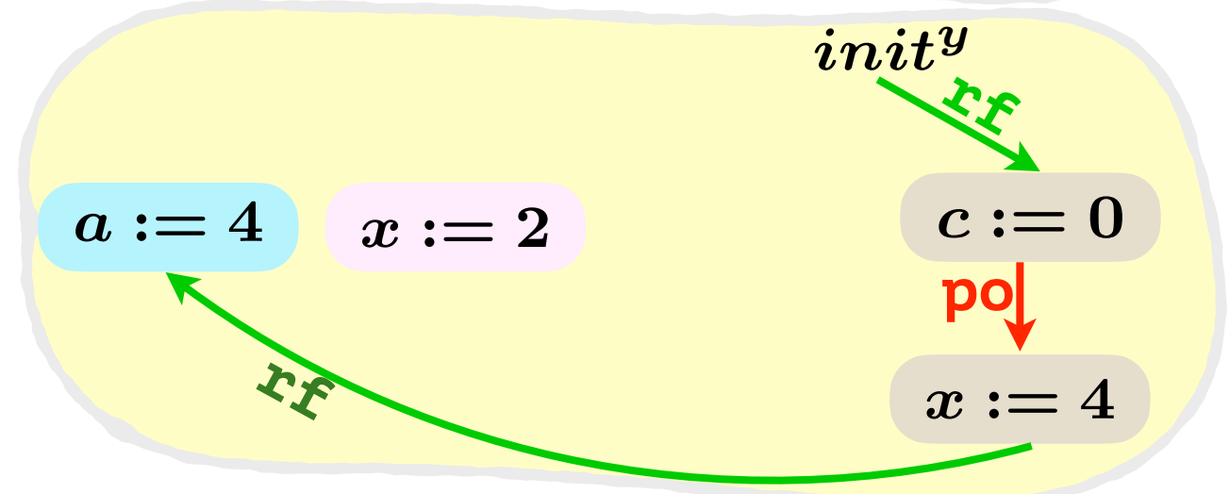

$a := x$
$x := 2$
$b := y$
$x := 3$
$c := y$
$x := 4$

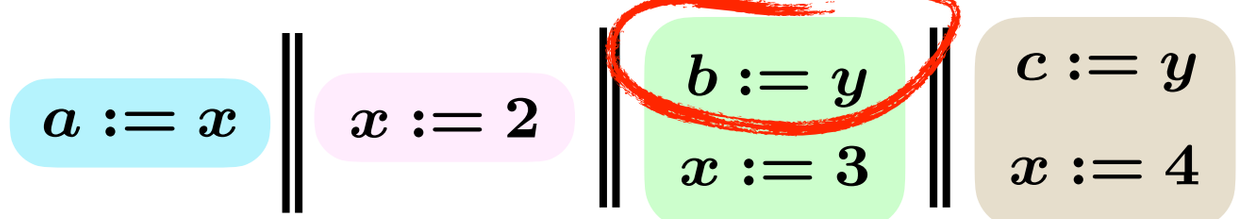

$x := 2$
$a := 0$
$a := 2$
$b := 0$
$c := 0$
$b := 0$
$c := 0$
$x := 3$
$x := 4$
$x := 4$
$a := 3$
$a := 4$
$c := 0$
$b := 0$
$c := 0$
$x := 4$
$x := 3$
$x := 4$

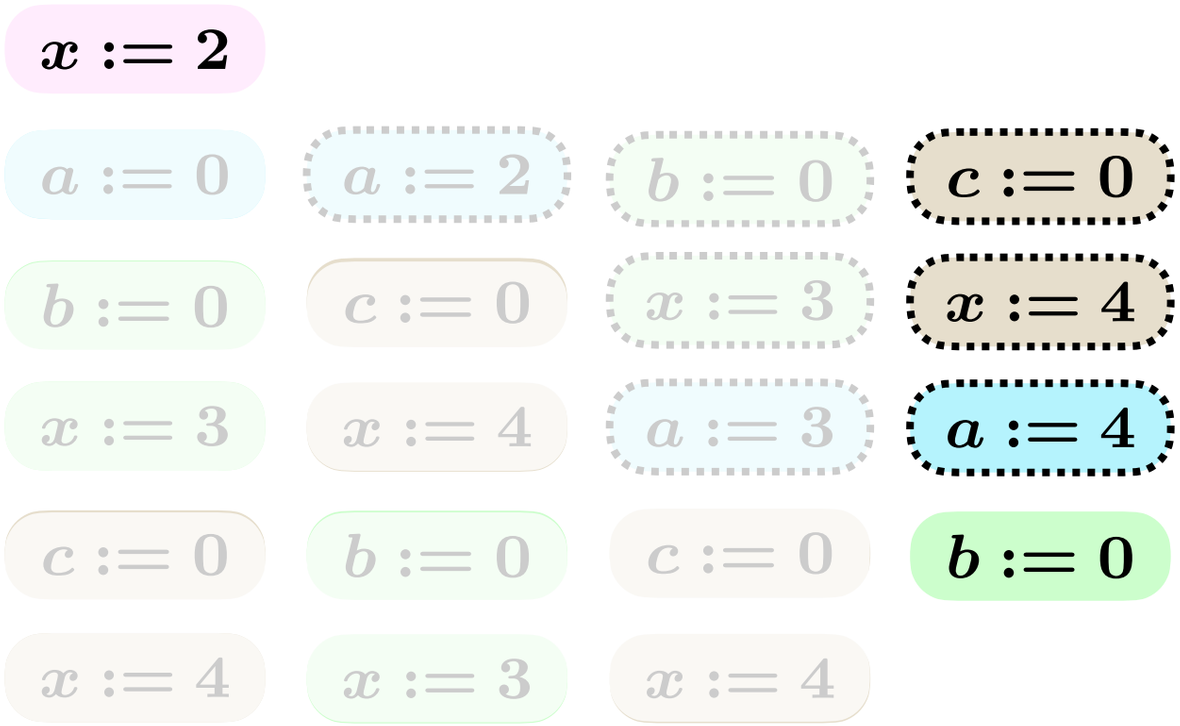

$init^x$
rf
$init^y$
rf
rf
$a := 0$
$x := 2$
$b := 0$
$c := 0$
po
po
$x := 3$
$x := 4$

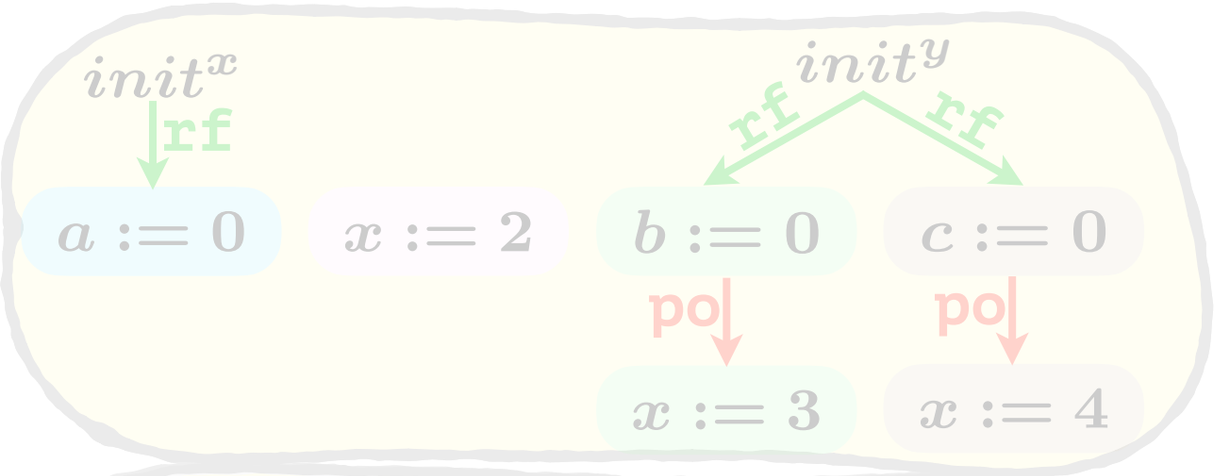

$init^y$
rf
rf
$a := 2$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

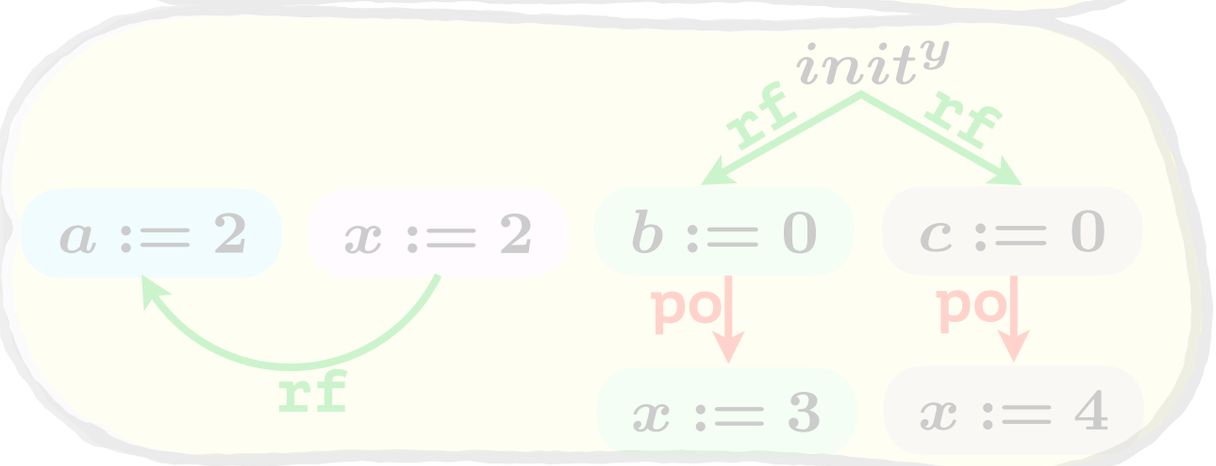

$init^y$
rf
rf
$a := 3$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

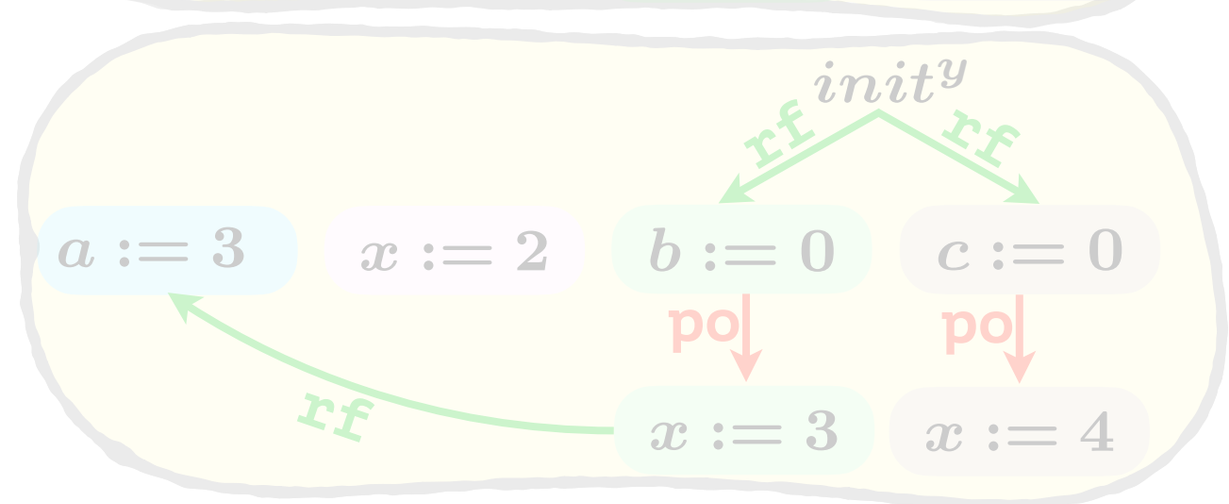

$init^y$
rf
$a := 4$
$x := 2$
$c := 0$
po
rf
$x := 4$

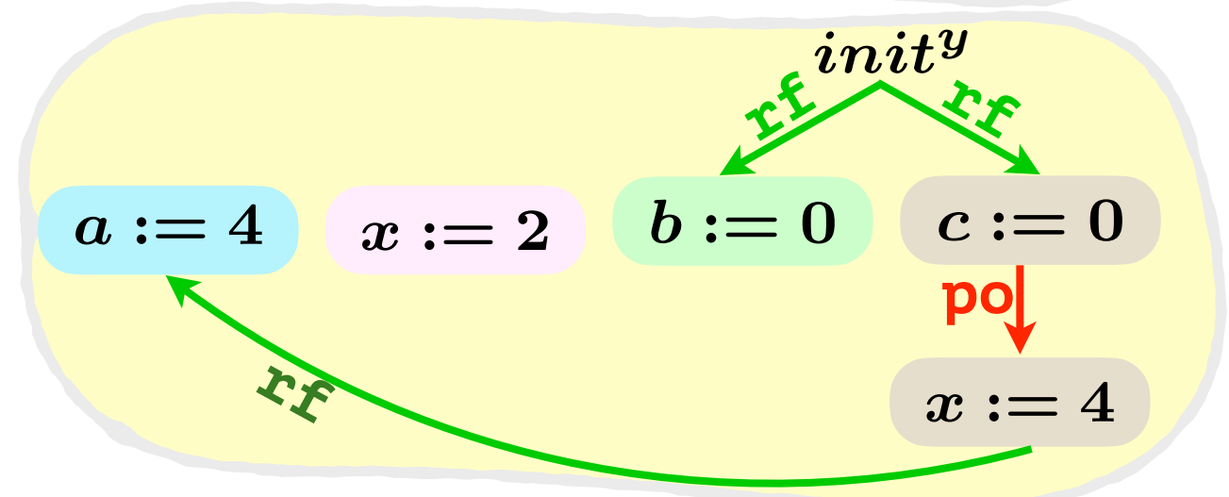

$a := x$
$x := 2$
$b := y$
$x := 3$
$c := y$
$x := 4$

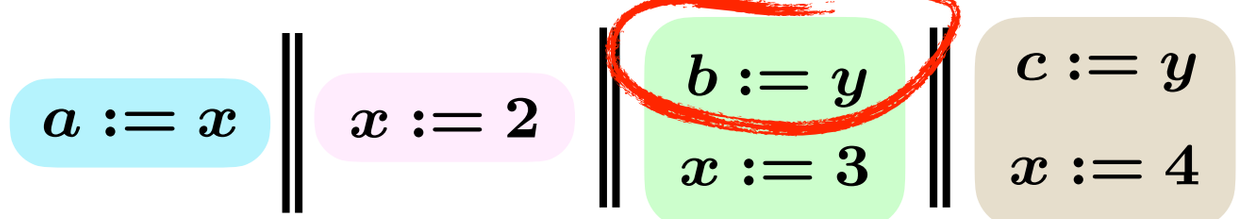

$x := 2$
$a := 0$
$a := 2$
$b := 0$
$c := 0$
$b := 0$
$c := 0$
$x := 3$
$x := 4$
$x := 4$
$a := 3$
$a := 4$
$c := 0$
$b := 0$
$c := 0$
$b := 0$
$x := 4$
$x := 3$
$x := 4$

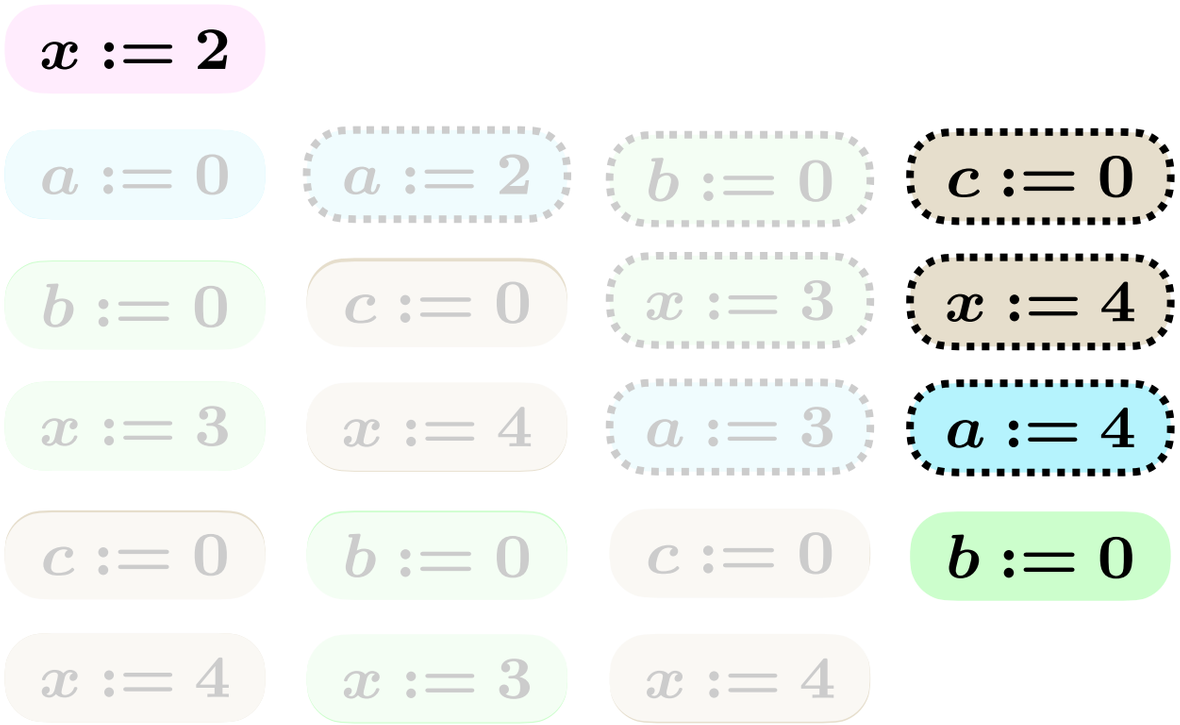

$init^x$
rf
$init^y$
rf
rf
$a := 0$
$x := 2$
$b := 0$
$c := 0$
po
po
$x := 3$
$x := 4$

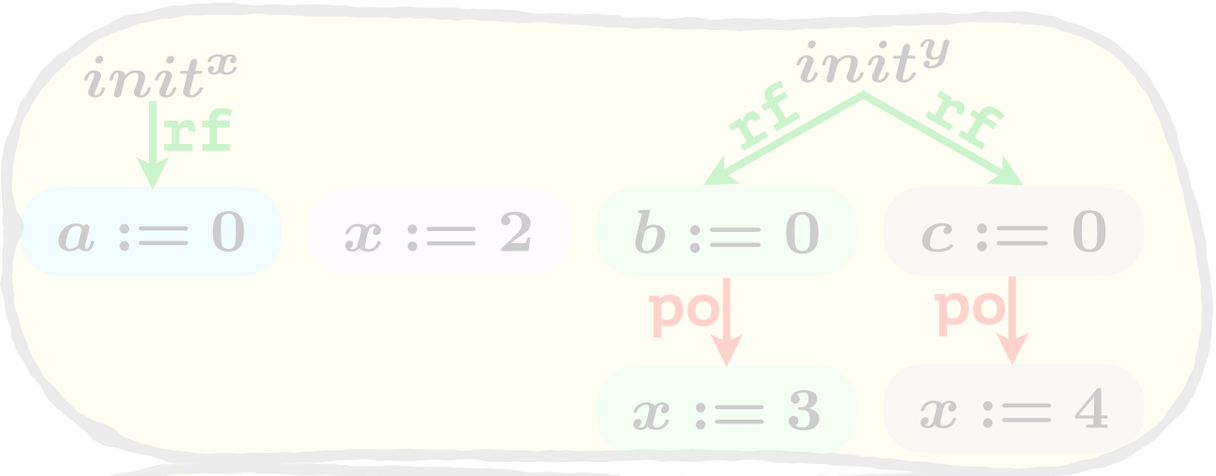

$init^y$
rf
rf
$a := 2$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

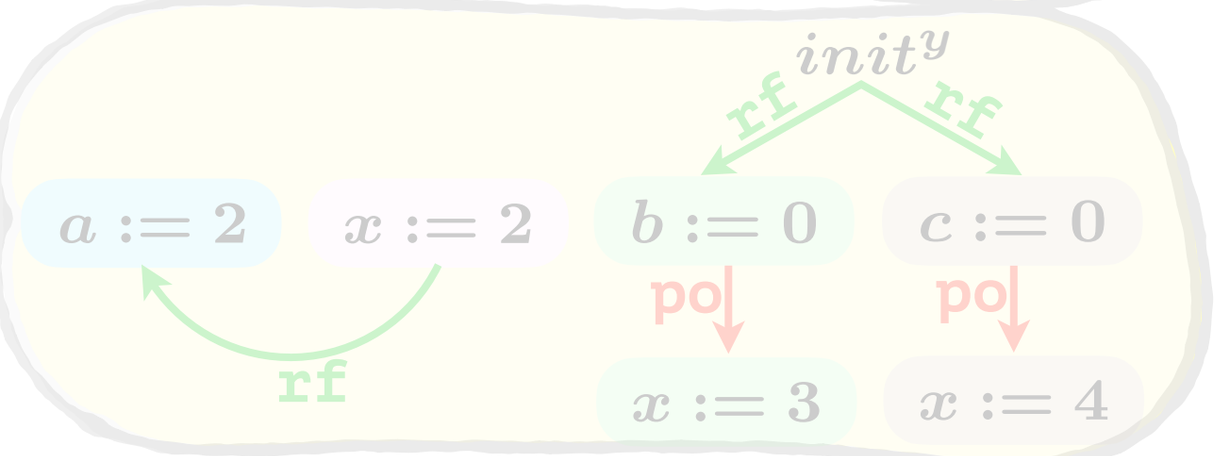

$init^y$
rf
rf
$a := 3$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

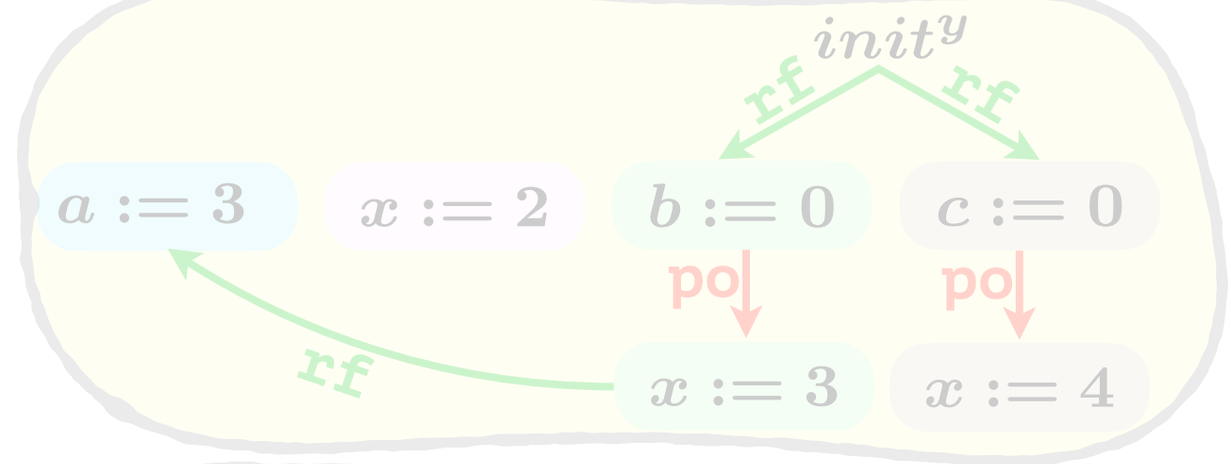

$init^y$
rf
rf
$a := 4$
$x := 2$
$b := 0$
$c := 0$
po
rf
$x := 4$

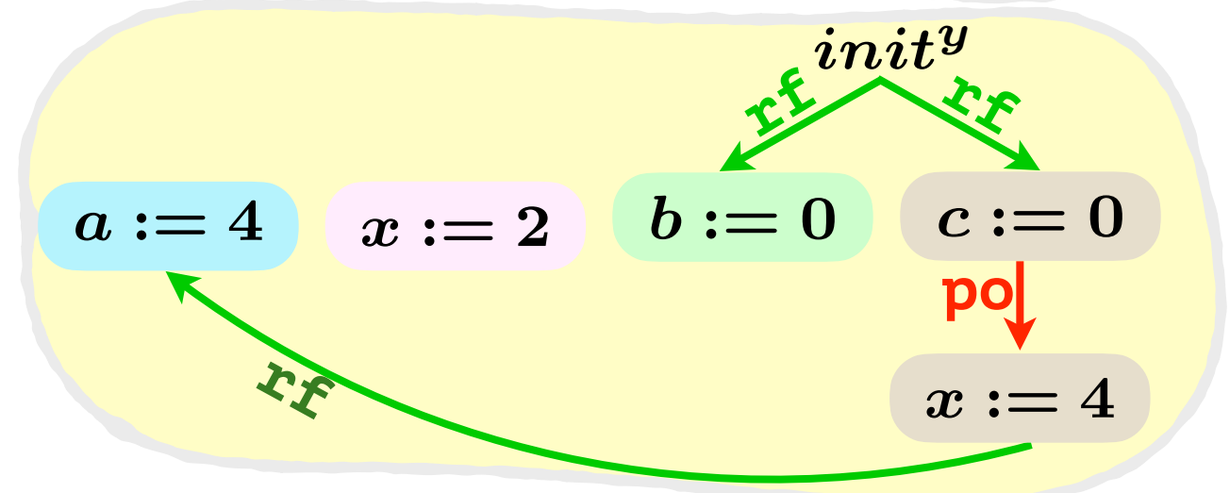

$a := x$
$x := 2$
$b := y$
$x := 3$
$c := y$
$x := 4$


$x := 2$
$a := 0$
$a := 2$
$b := 0$
$c := 0$
$b := 0$
$c := 0$
$x := 3$
$x := 4$
$c := 0$
$b := 0$
$c := 0$
$b := 0$
$x := 4$
$x := 3$
$x := 4$





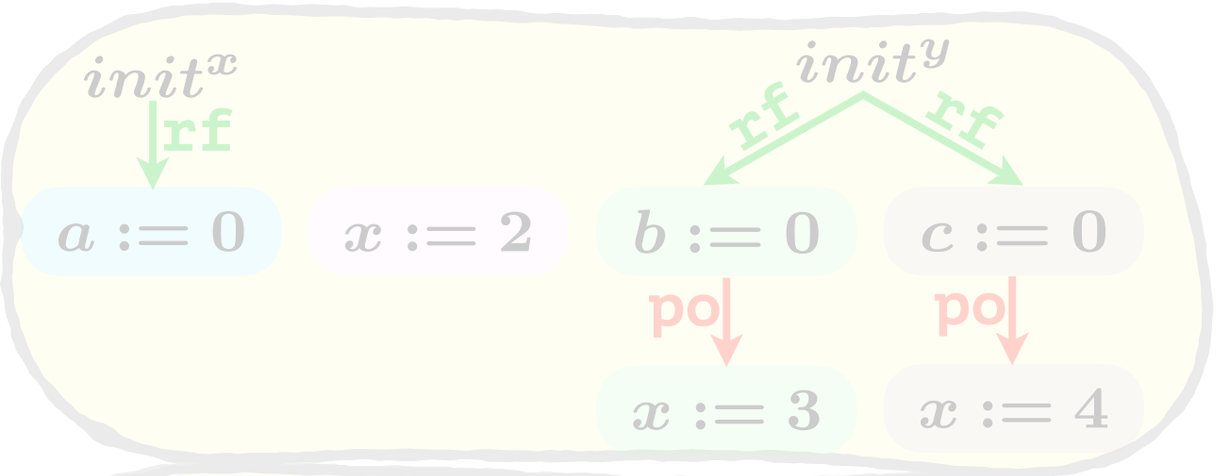


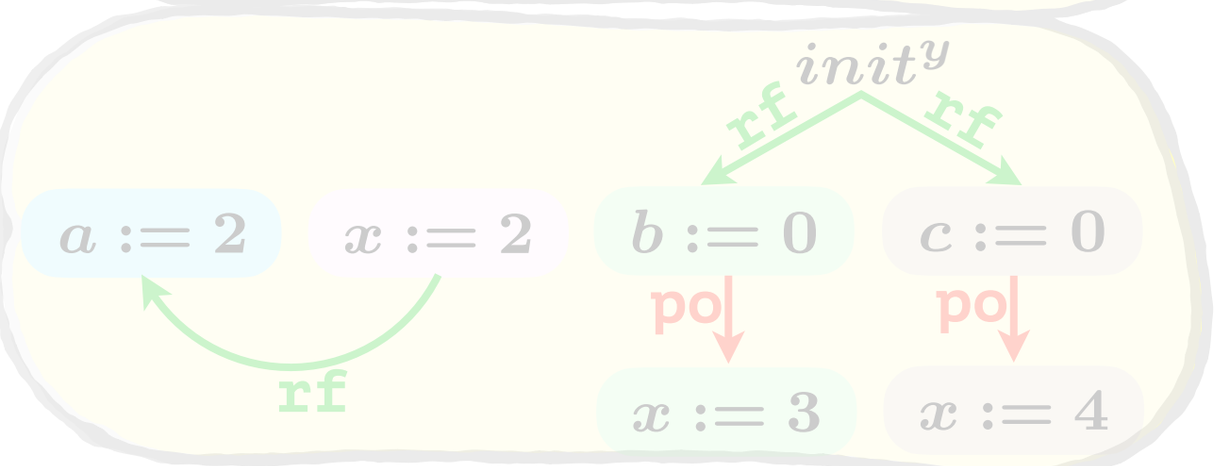


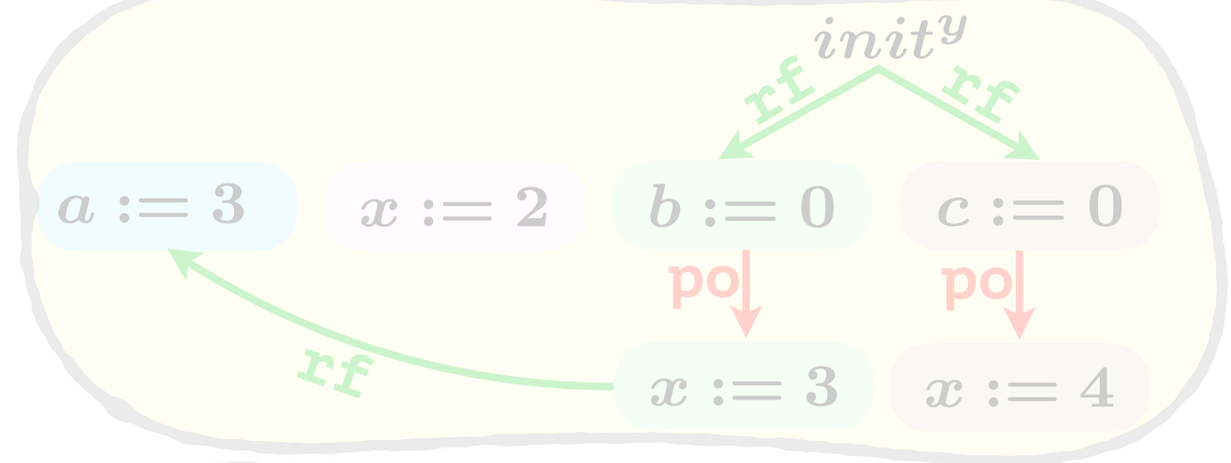
$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$

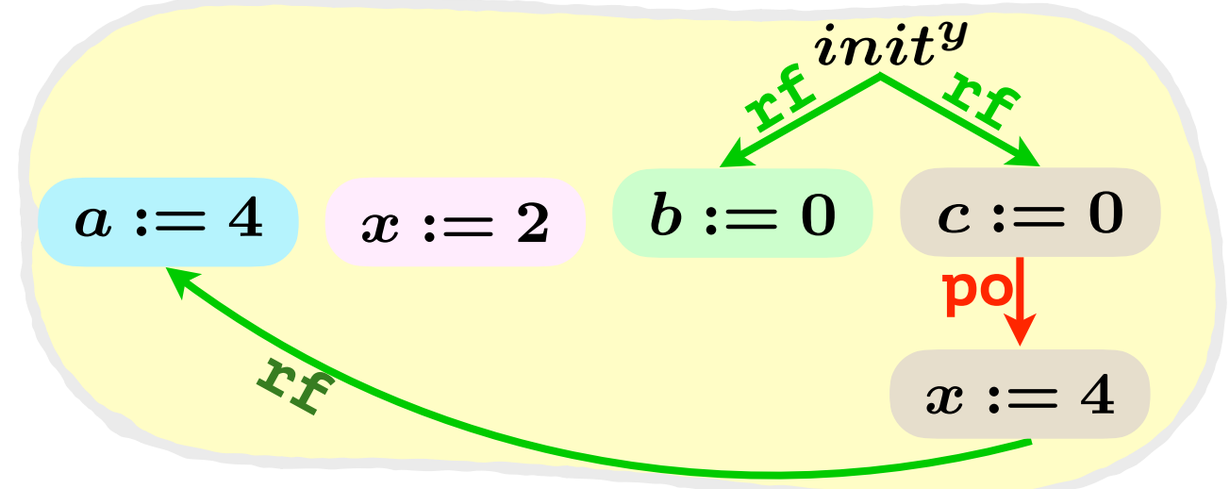
$x := 2$

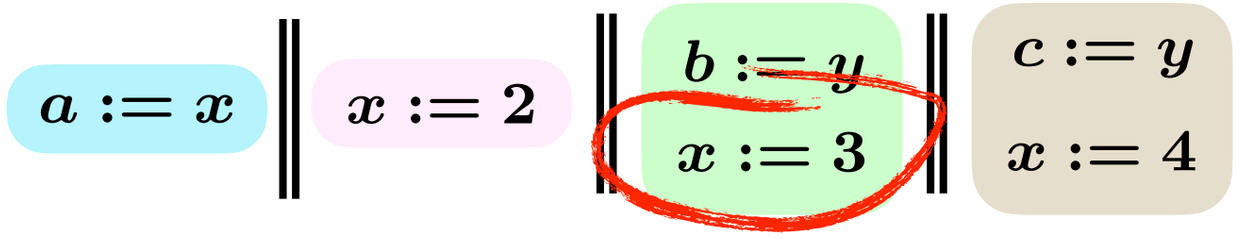
$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$	$c := 0$	$b := 0$
$x := 4$	$x := 3$	$x := 4$	





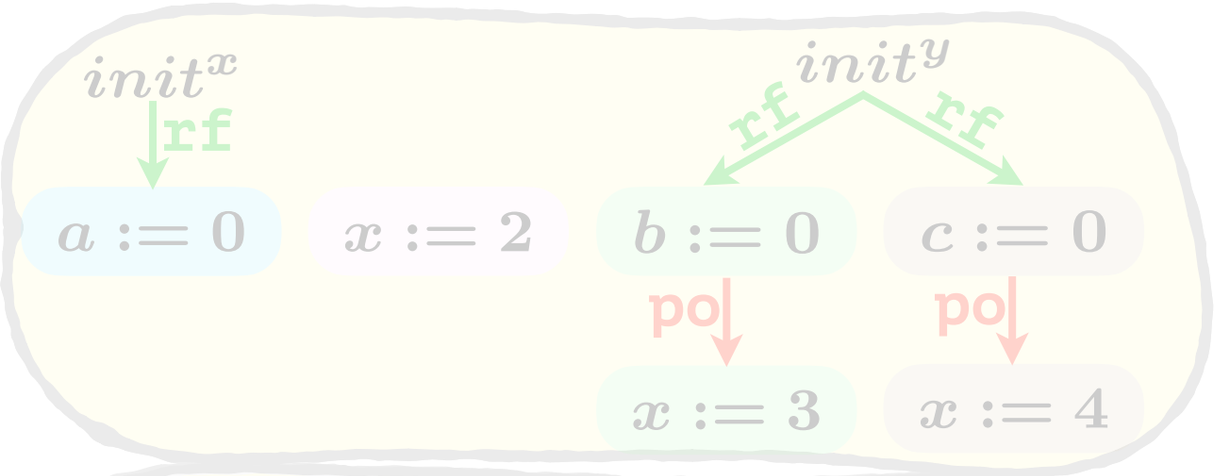


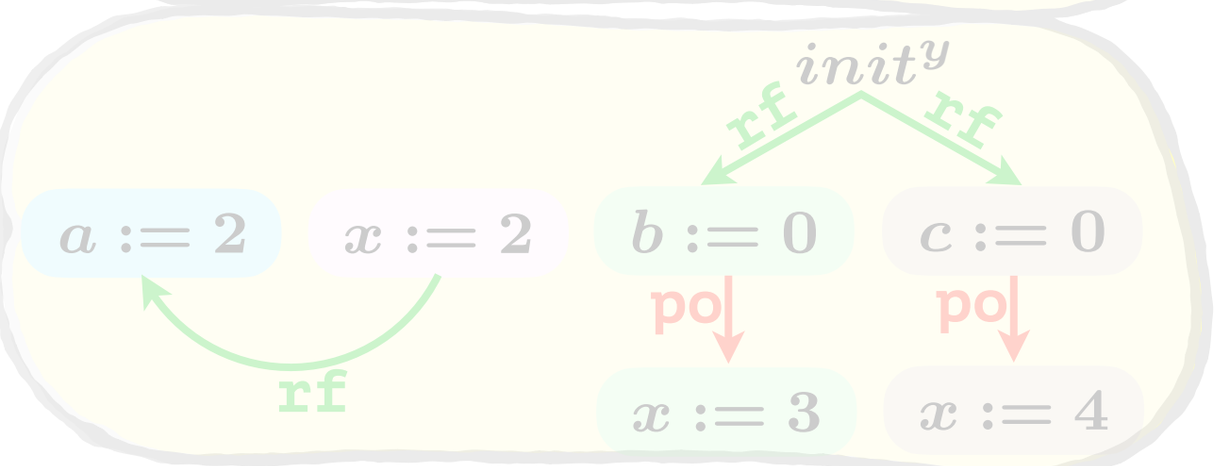


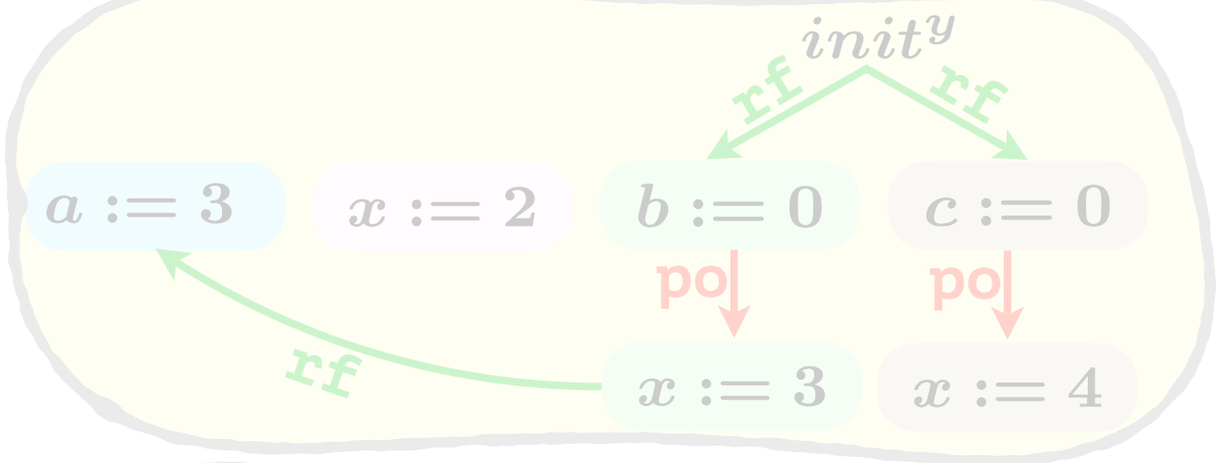


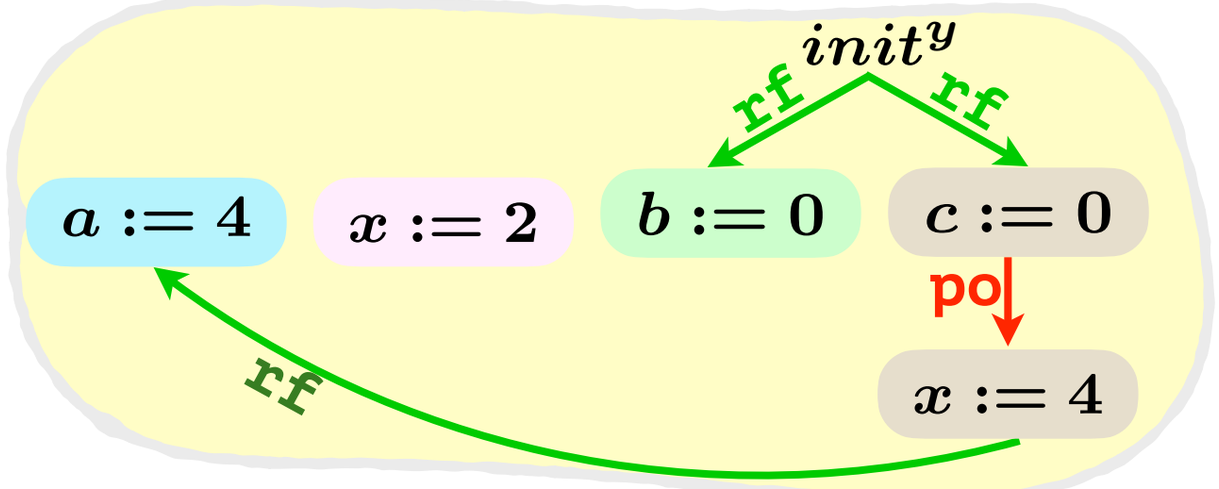
$x := 2$

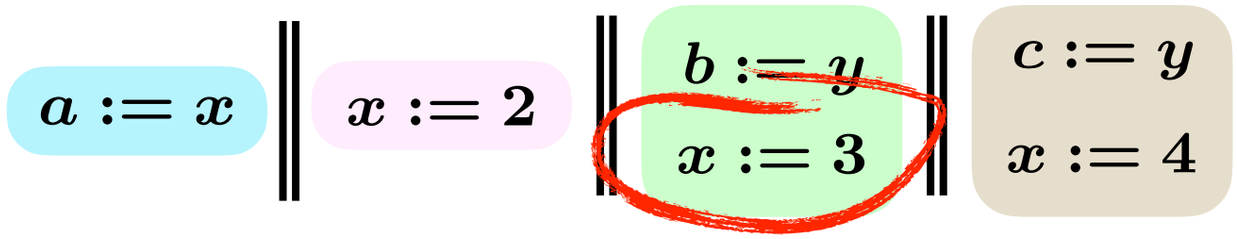
$a := 0$	$a := 2$	$b := 0$	$c := 0$
$b := 0$	$c := 0$	$x := 3$	$x := 4$
$x := 3$	$x := 4$	$a := 3$	$a := 4$
$c := 0$	$b := 0$	$c := 0$	$b := 0$
$x := 4$	$x := 3$	$x := 4$	





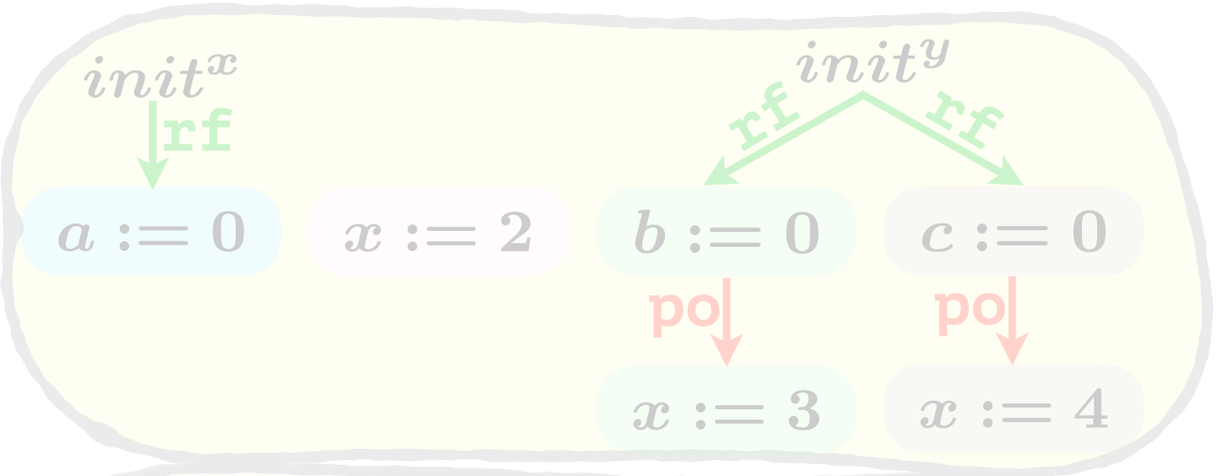


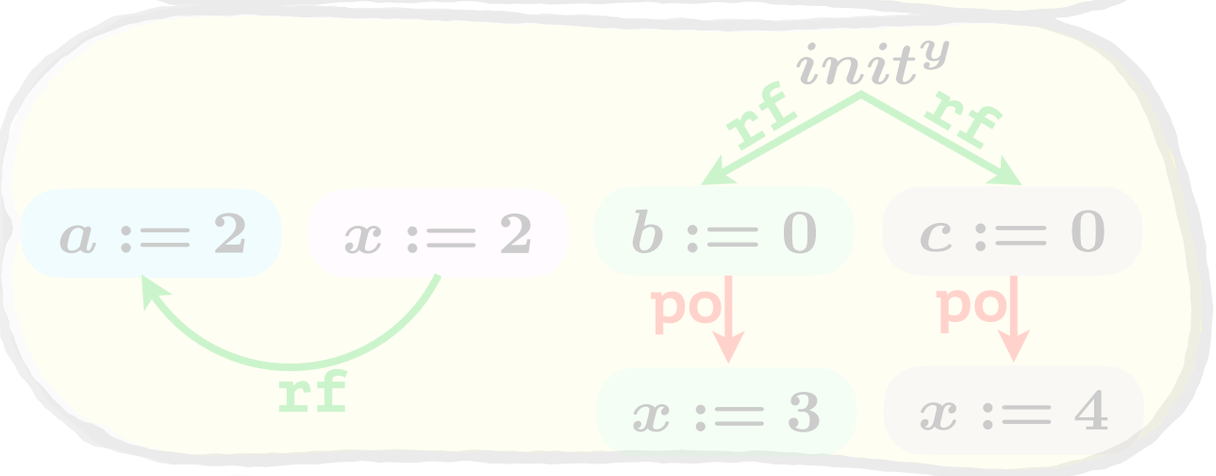


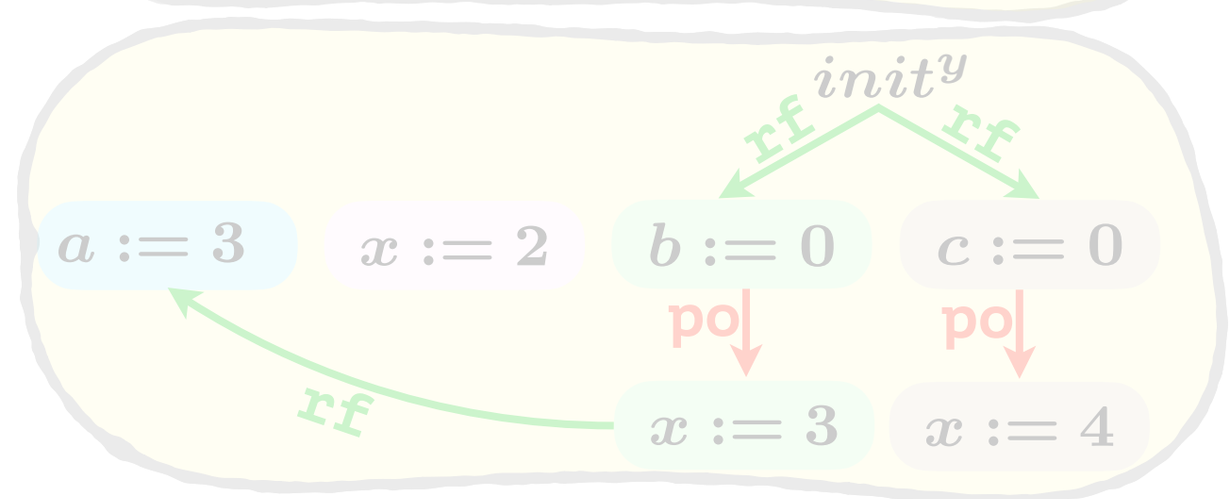


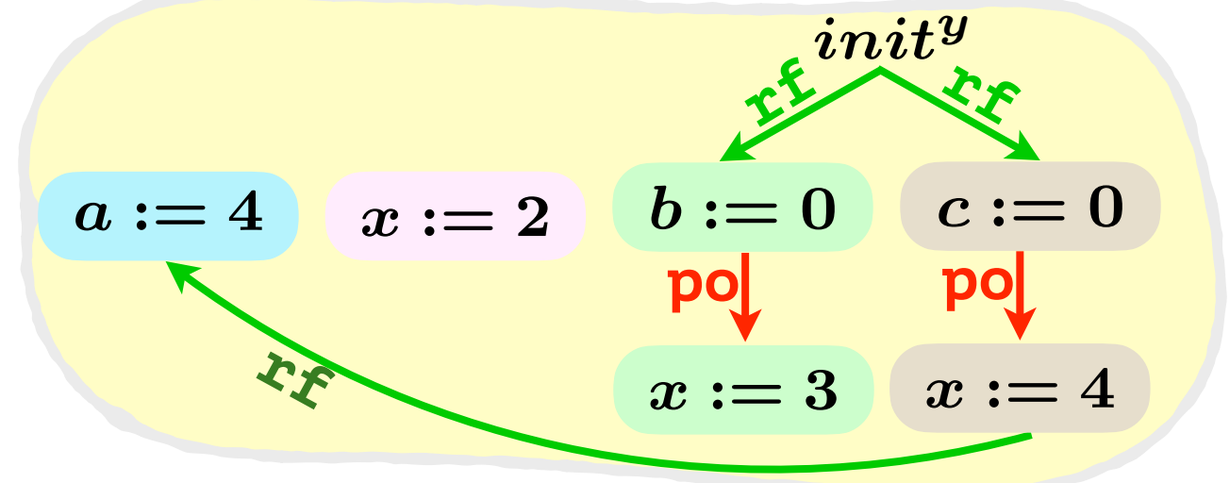
$x := 2$

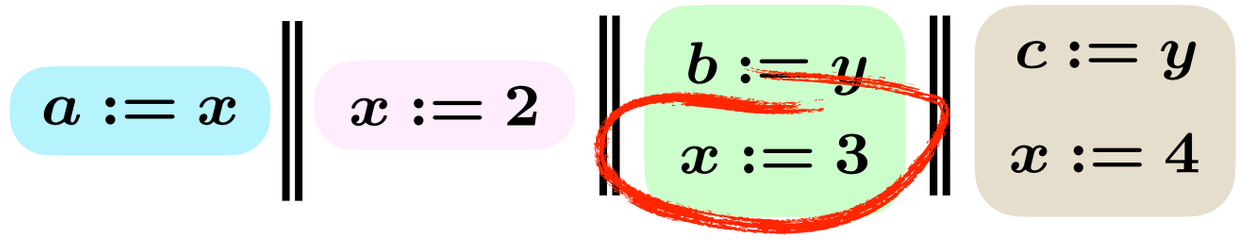
$a := 0$	$a := 2$	$b := 0$	**$c := 0$**
$b := 0$	$c := 0$	$x := 3$	**$x := 4$**
$x := 3$	$x := 4$	$a := 3$	**$a := 4$**
$c := 0$	$b := 0$	$c := 0$	**$b := 0$**
$x := 4$	$x := 3$	$x := 4$	**$x := 3$**

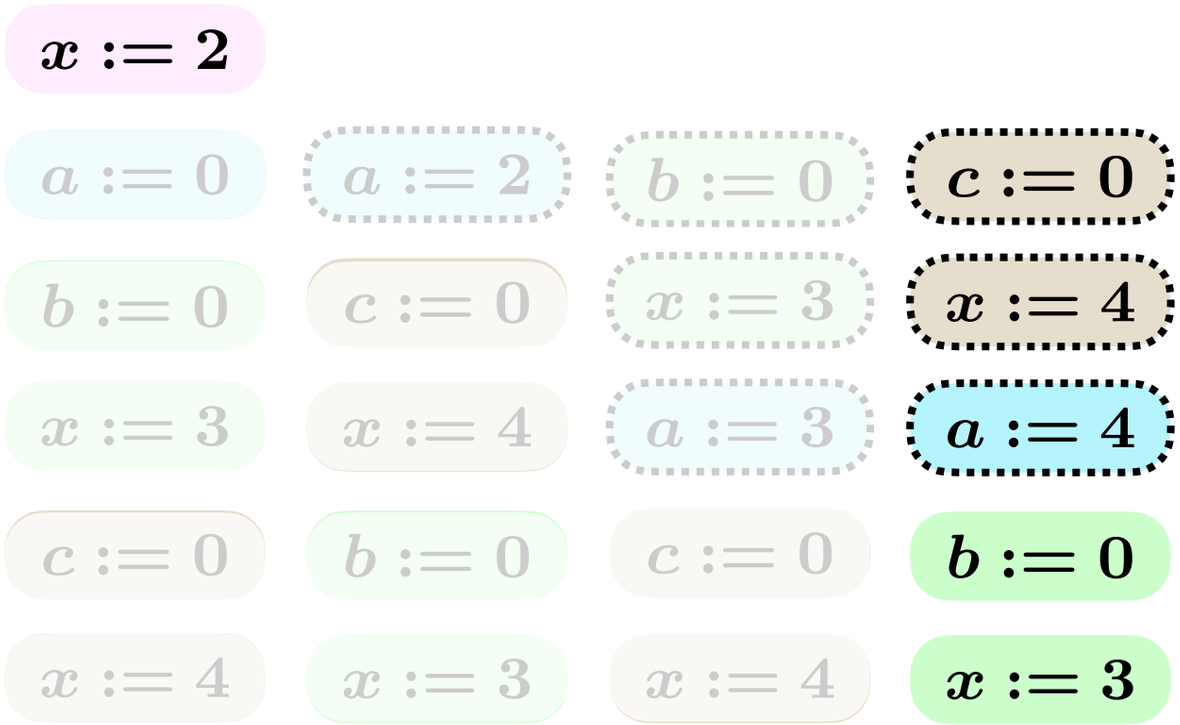

$init^x$
rf
$init^y$
rf
rf
$a := 0$
$x := 2$
$b := 0$
$c := 0$
po
po
$x := 3$
$x := 4$

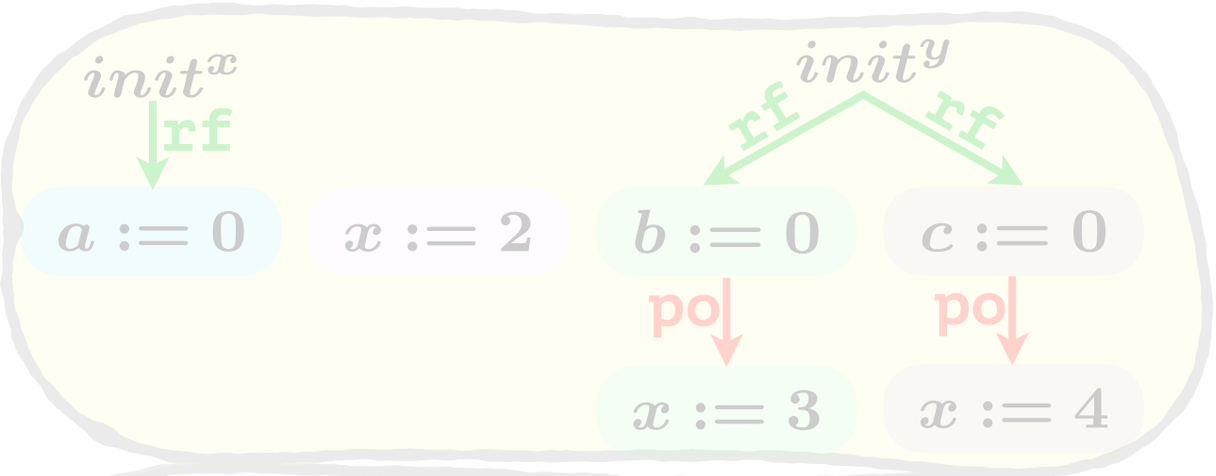

$init^y$
rf
rf
$a := 2$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

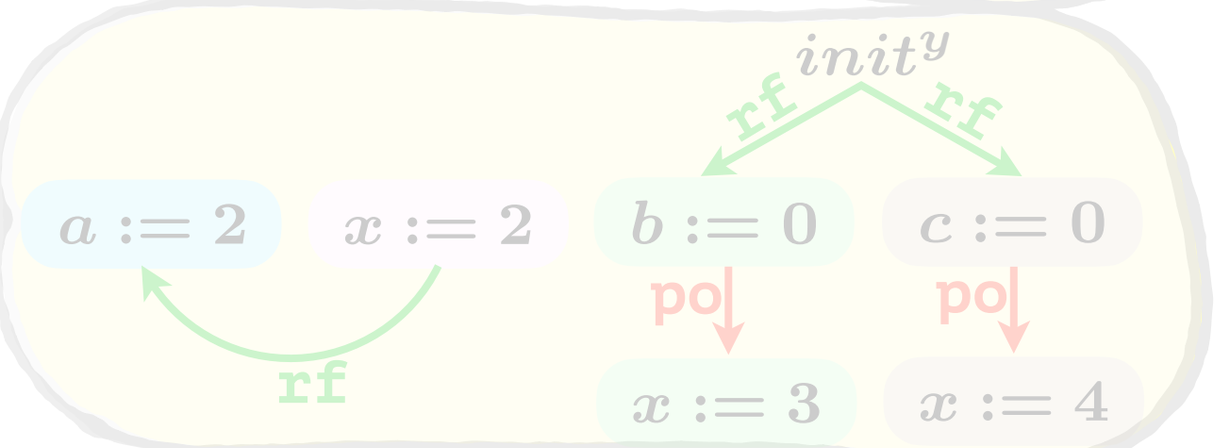

$init^y$
rf
rf
$a := 3$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

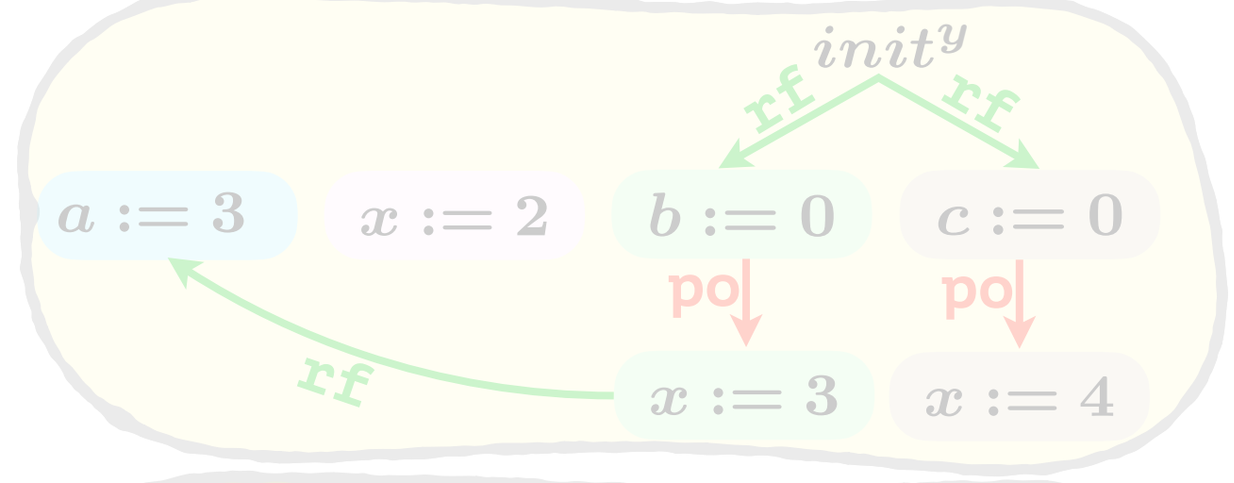

$init^y$
rf
rf
$a := 4$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

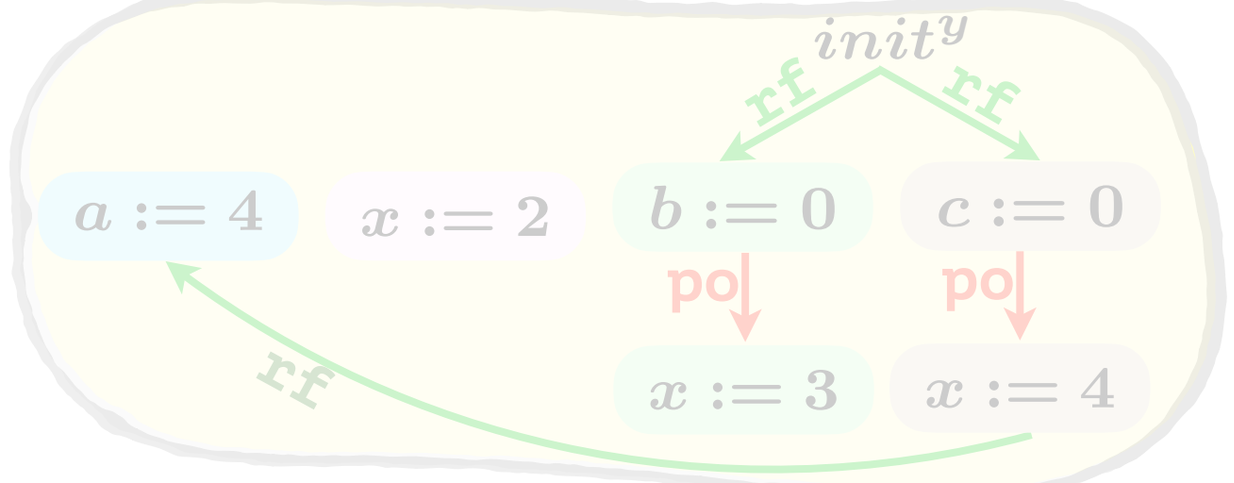

$a := x$
$x := 2$
$b := y$
$x := 3$
$c := y$
$x := 4$

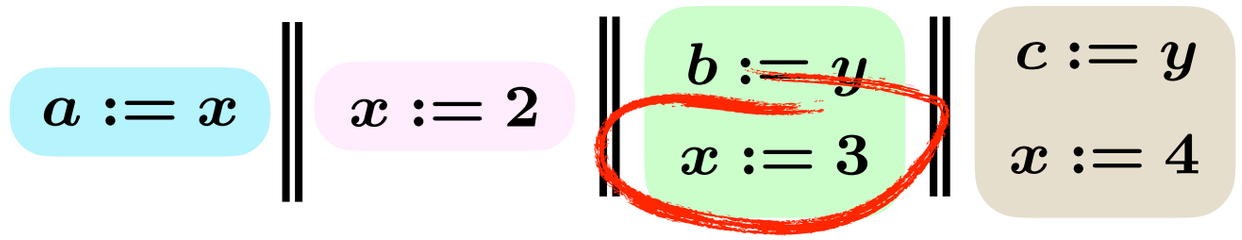

$x := 2$
$a := 0$
$a := 2$
$b := 0$
$c := 0$
$b := 0$
$c := 0$
$x := 3$
$x := 4$
$x := 4$
$a := 3$
$a := 4$
$c := 0$
$b := 0$
$c := 0$
$b := 0$
$x := 4$
$x := 3$
$x := 4$
$x := 3$

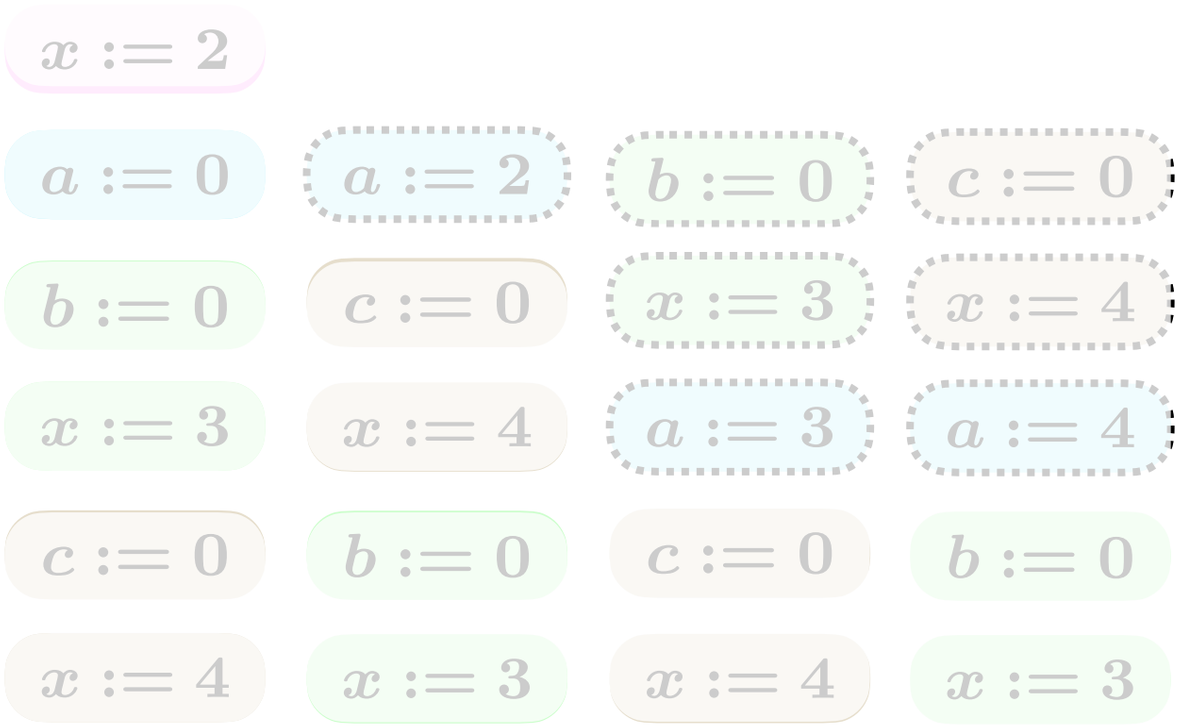

$init^x$
rf
$init^y$
rf
rf
$a := 0$
$x := 2$
$b := 0$
$c := 0$
po
po
$x := 3$
$x := 4$


$init^y$
rf
rf
$a := 2$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

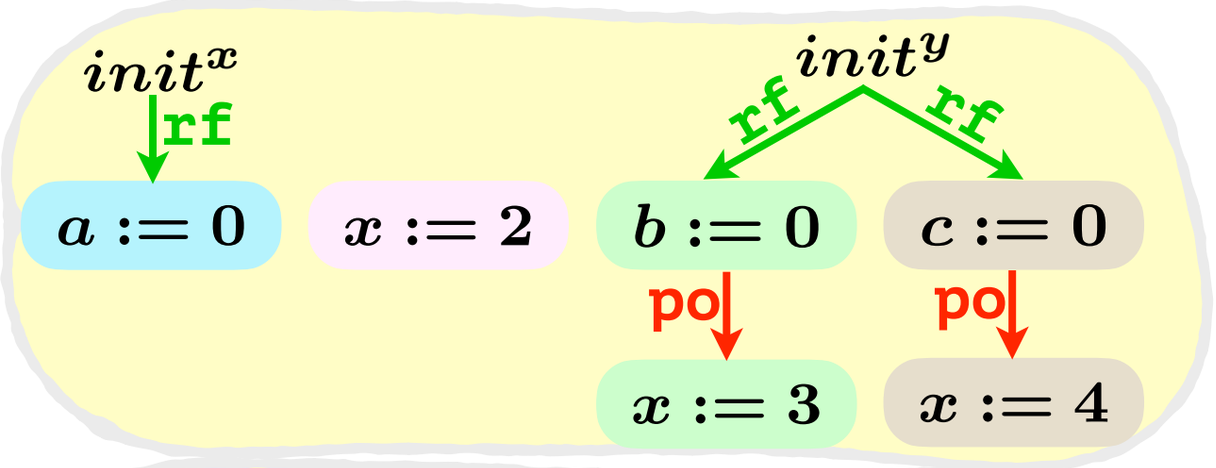

$init^y$
rf
rf
$a := 3$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

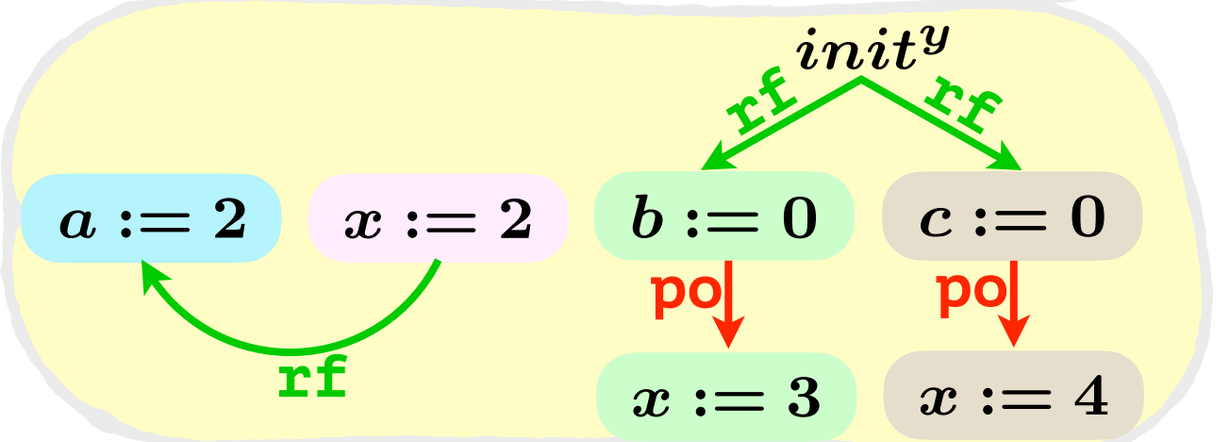

$init^y$
rf
rf
$a := 4$
$x := 2$
$b := 0$
$c := 0$
po
po
rf
$x := 3$
$x := 4$

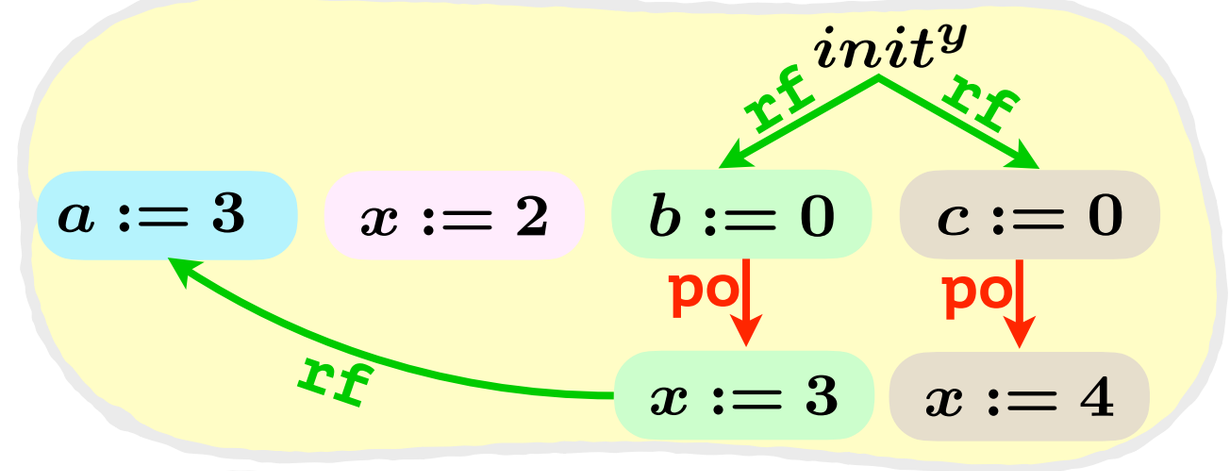

$a := x$
$x := 2$
$b := y$
$x := 3$
$c := y$
$x := 4$

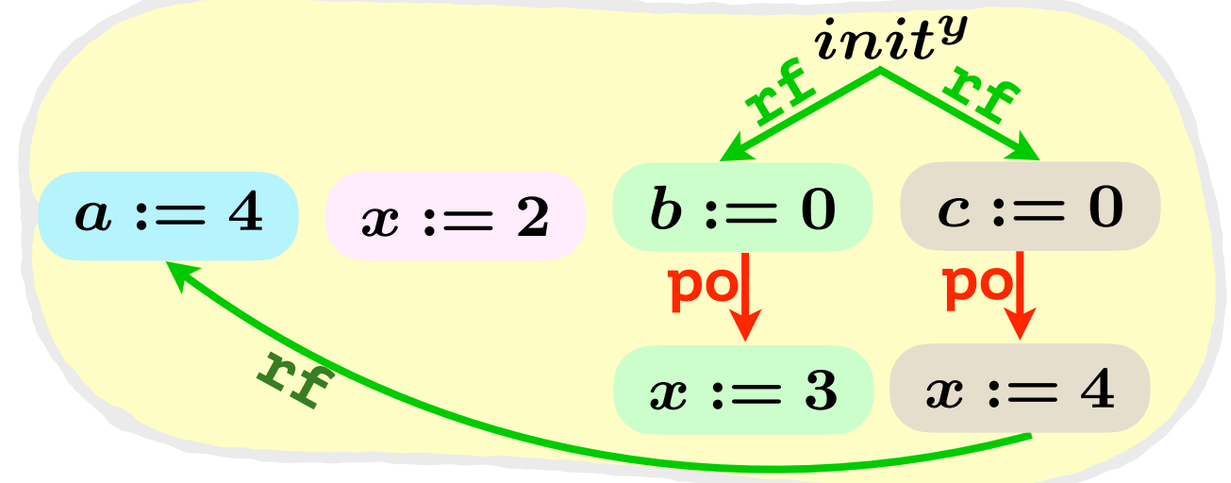

$x := 2$
$a := 0$
$a := 2$
$b := 0$
$c := 0$
$b := 0$
$c := 0$
$x := 3$
$x := 4$
$c := 0$
$b := 0$
$c := 0$
$b := 0$
$x := 4$
$x := 3$
$x := 4$
$x := 3$

$$a := x \;\|\; x := 2 \;\|\; \begin{matrix} b := y \\ x := 3 \end{matrix} \;\|\; \begin{matrix} c := y \\ x := 4 \end{matrix}$$









$x := 2$

$a := 0$ $a := 2$ $b := 0$ $c := 0$

$b := 0$ $c := 0$ $x := 3$ $x := 4$

$x := 3$ $x := 4$ $a := 3$ $a := 4$

$c := 0$ $b := 0$ $c := 0$ $b := 0$

$x := 4$ $x := 3$ $x := 4$ $x := 3$

$a := x$ ‖ $x := 2$ ‖ $b := y$ / $x := 3$ ‖ $c := y$ / $x := 4$

- Sound
  - only valid traces
- Complete
  - all valid traces
- Optimal
  - different traces
- Deadlock-free

Tool	Wrong answer	Execs	Time	Average Execs (*Execs/Nums of tests*)	Average Time (*Time/Execs*)
CDSChecker	0	2 383 704	6m28s	295.59	**0.16ms**
Rcmc/Rc11	0	727 942	14m00s	90.27	1.15ms
Rcmc/Wrc11	81	668 574	12m15s	82.91	1.10ms
Tracer	0	**521 288**	**3m03s**	**64.64**	0.35ms

Litmus Tests

Program	LB	Executions Explored			Total Running Time		
		CDSChecker	Rcmc	Tracer	CDSChecker	Rcmc	Tracer
Indexer(13)	20	190	**64**	**64**	0.15s	0.62s	**0.04s**
Indexer(14)	20	3 075	**512**	**512**	2.72s	11.68s	**0.38s**
Indexer(15)	20	48 261	**4 096**	**4 096**	45.32s	4m22s	**2.98s**
Indexer(16)	20	740 889	*t/o*	**32 768**	12m21s	*t/o*	**25.78s**
Filesystem(16)	20	27	**8**	**8**	0.04s	0.05s	**0.01s**
Filesystem(19)	20	728	**64**	**64**	0.52s	2.02s	**0.03s**
Filesystem(22)	20	19 678	**512**	**512**	13.53s	2m29s	**0.33s**
Filesystem(25)	20	531 415	*t/o*	**4096**	7m04s	*t/o*	**3.12s**

Standard Benchmarks

Program	LB	Executions Explored			Total Running Time		
		CDSChecker	Rcmc	Tracer	CDSChecker	Rcmc	Tracer
Pthread_demo	10	184 758	**184 756**	**184 756**	24.96s	1m03s	**18.00s**
Gcd	8	8 814 044	**1 162 333**	**1 162 333**	38m20s	7m07s	**2m48s**
Fibonacci	6	*t/o*	**525 630**	**525 630**	*t/o*	**31.06s**	57.95s
Szymanski	6	*t/o*	26 037 490	**12 209 410**	*t/o*	44m52s	**14m54s**
Dekker	10	7 306 447	**3 121 870**	**3 121 870**	15m25s	5m12s	**4m52s**
Lamport	8	*t/o*	6 580 870	**3 372 868**	*t/o*	14m40s	**6m58s**
Sigma(5)	5	1 279	945	**120**	0.09s	0.16s	**0.01s**
Peterson	6	*t/o*	**1 897 228**	**1 897 228**	*t/o*	3m16s	**3m15s**
Stack_true	12	2 704 157	**2 704 156**	**2 704 156**	19m03s	54m25s	**10m12s**
Queue_ok	12	581 790	*t/o*	**362 880**	33m27s	*t/o*	**12m52s**

SV-COMP

Program	Executions Explored			Total Running Time		
	CDSChecker	Rcmc	Tracer	CDSChecker	Rcmc	Tracer
N_writers_a_reader(7)	**8**	40 320	**8**	0.01s	0.46s	**0.00s**
N_writers_a_reader(8)	**9**	362 880	**9**	0.01s	4.19s	**0.00s**
N_writers_a_reader(9)	**10**	3 628 800	**10**	0.01s	46.13s	**0.00s**
N_writers_a_reader(10)	**11**	39 916 800	**11**	0.01s	9m35s	**0.00s**
Redundant_co(5)	581	16 632	**91**	0.03s	0.39s	**0.01s**
Redundant_co(10)	10 631	42 678 636	**331**	0.64s	23m56s	**0.02s**
Redundant_co(15)	59 056	*t/o*	**721**	4.57s	*t/o*	**0.06s**
Redundant_co(20)	197 231	*t/o*	**1 261**	20.27s	*t/o*	**0.14s**

Synthetic Programs

Program	LB	Executions Explored			Total Running Time		
		CDSChecker	Rcmc	Tracer	CDSChecker	Rcmc	Tracer
Sigma(6)	6	25 357	10 395	**720**	2.20s	1.96s	**0.04s**
Sigma(7)	7	605 714	135 135	**5 040**	1m02s	29.06s	**0.40s**
Sigma(8)	8	16 667 637	2 027 025	**40 320**	33m18s	8m02s	**3.28s**
Sigma(9)	9	*t/o*	*t/o*	**362 880**	*t/o*	*t/o*	**33.59s**
Control_flow(6)	0	896	55 440	**77**	0.09s	1.97s	**0.01s**
Control_flow(8)	0	4 608	11 007 360	**273**	0.53s	7m58s	**0.03s**
Control_flow(10)	0	22 528	*t/o*	**1 045**	3.27s	*t/o*	**0.16s**
Control_flow(12)	0	106 496	*t/o*	**4 121**	19.10s	*t/o*	**0.79s**
Exponential_bug(6)	6	983 386	1 203 446	**15 601**	1m18s	56.75s	**0.96s**
Exponential_bug(7)	7	2 250 290	2 833 112	**22 841**	3m13s	2m26s	**1.46s**
Exponential_bug(8)	8	4 844 378	6 158 718	**32 313**	7m15s	5m28s	**2.23s**
Exponential_bug(9)	9	9 896 954	12 526 576	**44 428**	15m29s	11m48s	**3.22s**

Parameterized Programs

Program	LB	Executions Explored			Total Running Time		
		CDSChecker	Rcmc	Tracer	CDSChecker	Rcmc	Tracer
Linux_locks(2)	6	47	**21**	**21**	0.03s	0.02s	**0.01s**
Linux_locks(3)	6	14 187 799	**412 814**	**412 814**	16m01s	36.36s	**33.21s**
Ticket_mutex_c11(3)	10	8 054	**4 026**	**4 026**	0.69s	50.76s	**0.25s**
Correia_Ramalhete(3)	5	**5 355**	**5 355**	**5 355**	1.10s	**0.77s**	0.92s

Concurrent Data Structures

Program	LB	Executions Explored		Total Running Time	
		CDSChecker	Tracer	CDSChecker	Tracer
Barrier(3)	10	62 649	**31 944**	5.45s	**1.88s**
Seqlock(3)	5	17 792	**14 864**	1.17s	**0.77s**
MPMC_queue(3)	5	621 882	**239 254**	2m11s	**44.50s**
MCS_lock(2)	5	92 210	**70 072**	8.91s	**5.19s**
Cliffc_hashtable(4)	0	9 520	**4 576**	2.68s	**0.94s**
Concurrent_hashmap(4)	0	110	**42**	0.01s	**0.00s**
Chase_Lev_deque(2) (**deadlock**)	0	162 306	**20 852**	20.65s	**2.10s**
SPSC_queue(2) (**deadlock**)	3	754	**57**	0.18s	**0.01s**

Concurrent Data Structures

Program	LB	Executions Explored		Total Running Time	
		CDSChecker	Tracer	CDSChecker	Tracer
Tidex(3)	5	4 676	**748**	0.37s	**0.04s**
Tidex_nps(2)	5	10 257	**10 254**	6m15s	**22.58s**
CLH_c11(3)	10	2 562	**732**	0.22s	**0.06s**
CLH_rwlock_c11(3)	10	20	**6**	0.01s	**0.00s**
MPSC_c11(3)	10	12 937	**4 824**	1.20s	**0.36s**
Correia_Ramalhete_turn(3) (**mutex broken**)	2	441 494	**96 184**	1m17s	**12.78s**

Concurrent Data Structures

Conclusions And Future Work

- Semantics
- Decidability and Complexity
- Approximation Techniques
- Program Repair

DPOR

Conclusions

- (Optimal) DPOR
- Release-Acquire Semantics
- Experiments

DPOR

Conclusions

- (Optimal) DPOR
- Release-Acquire Semantics
- Experiments

Future Work

- Other Memory Models
  - Full C11, SC, TSO, POWER, ARM, Causal Consistency, …
- Weaker Notions of Traces
- Other Properties:
  - Race Detection
  - Liveness: Progress, Termination, …