

# Second Lecture:

## Basics of model-checking for finite and timed systems

Jean-François Raskin

Université Libre de Bruxelles  
Belgium

Artist2 Asian Summer School - Shanghai - July 2008

# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

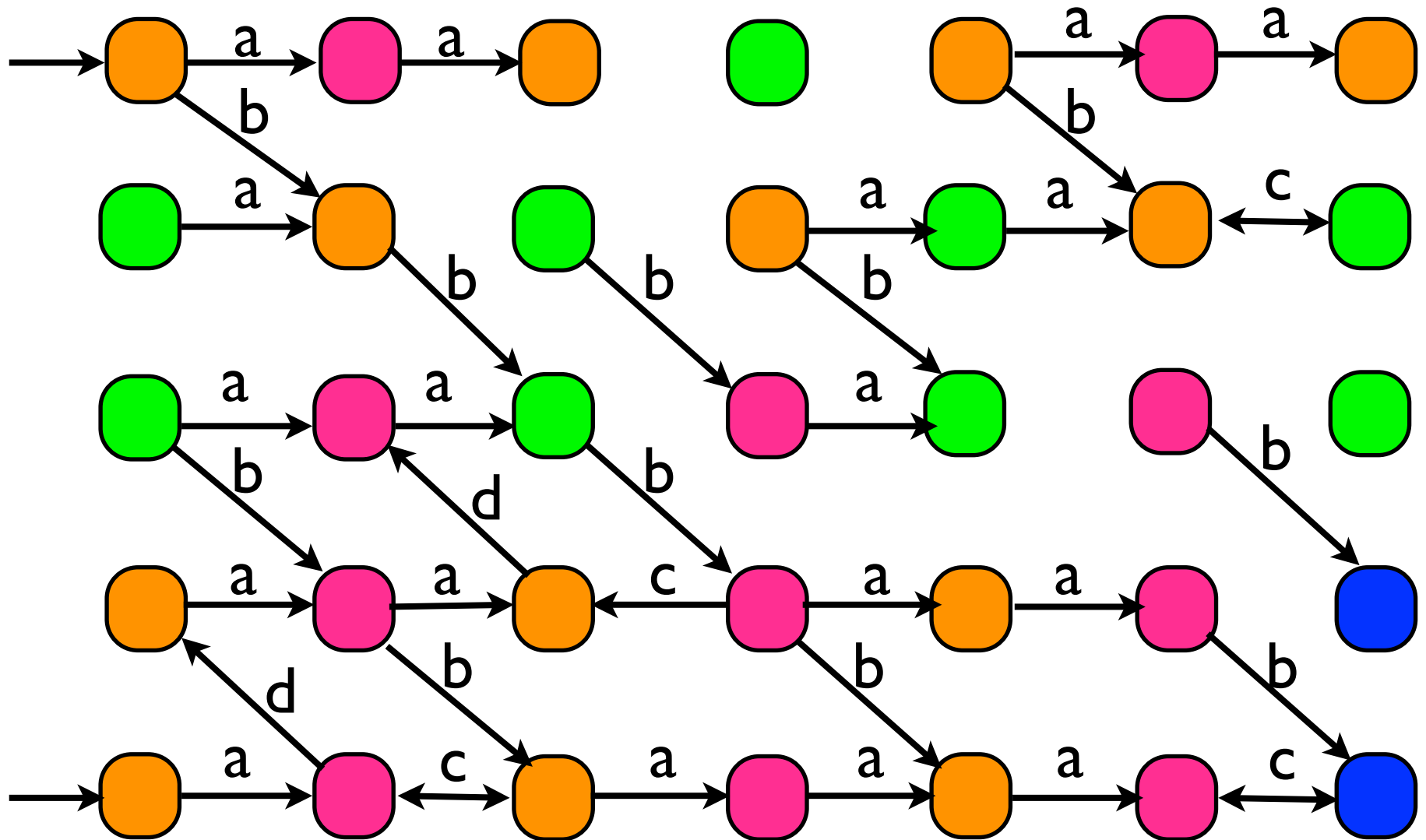
# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

# Labeled transition systems

- A **labeled transition system**, LTS for short, is a tuple  $(S, S_0, \Sigma, T, C, \lambda)$  where:
  - $S$  is a (finite or infinite) set of states
  - $S_0 \subseteq S$  is the subset of initial states
  - $\Sigma$  is an event or action set (finite or infinite)
  - $C$  is a (finite or infinite) set of colors
  - $\lambda : S \rightarrow C$  is a labeling function that labels each state with a color.

# A labelled transition system:



# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

# Reachability

- **Reachability verification problem**

**Instance:** a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , a set  $\text{Goal} \subseteq S$ .

**Question:** is there an execution of the LTS that starts in  $S_0$  and reaches  $\text{Goal}$  ?

More formally, is there a sequence  $s_0 \sigma_0 s_1 \sigma_1 s_2 \sigma_2 \dots \sigma_{n-1} s_n$  such that

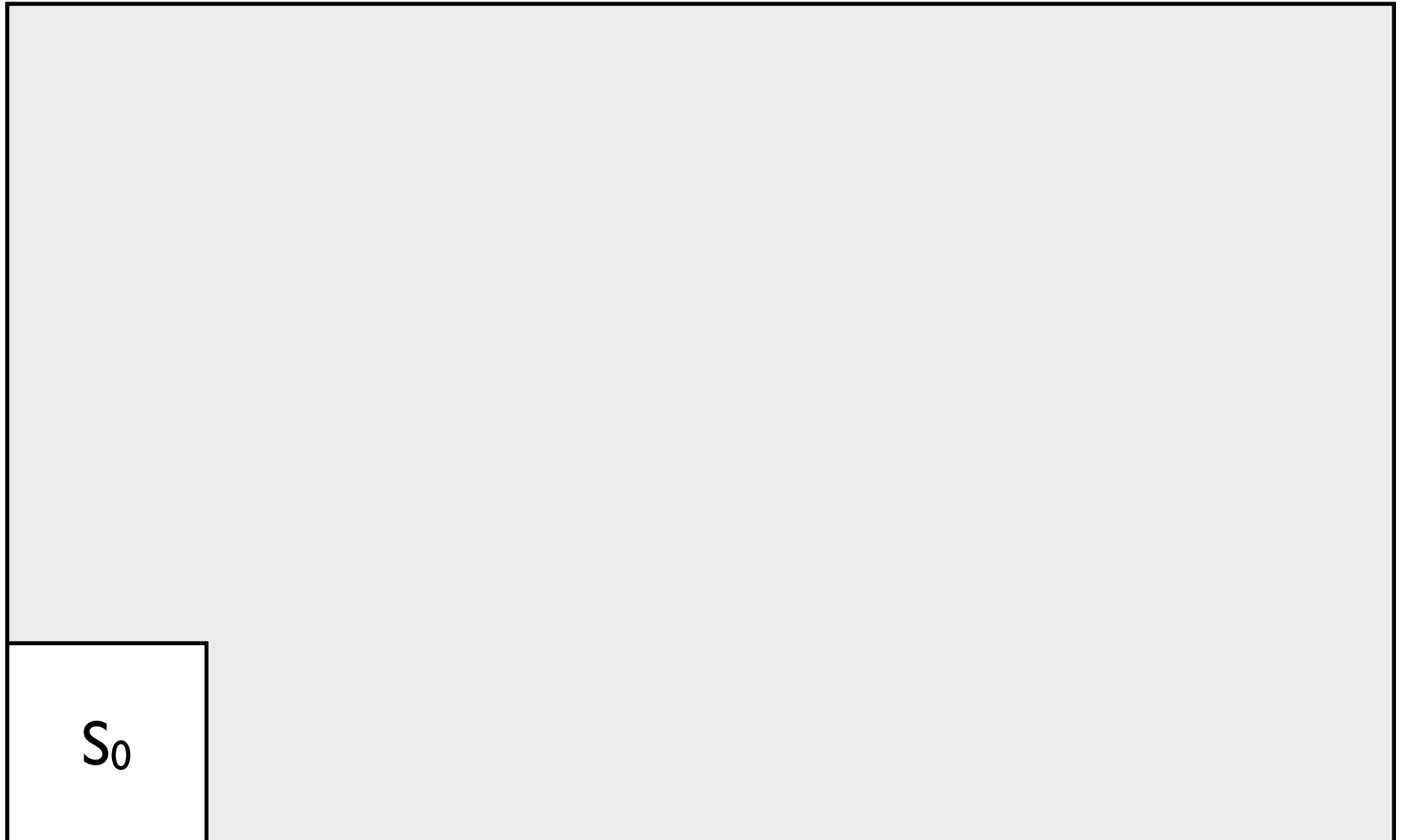
(1)  $s_0 \in S_0$ , (2)  $\forall i \cdot 0 \leq i < n \cdot T(s_i, \sigma_i, s_{i+1})$ , and (3)  $s_n \in \text{Goal}$  ?

- The **set of reachable states** of a LTS  $(S, S_0, \Sigma, T, C, \lambda)$  is the set of states  $s \in S$  such that  
there is a sequence  $s_0 \sigma_0 s_1 \sigma_1 s_2 \sigma_2 \dots \sigma_{n-1} s_n$   
and (1)  $s_0 \in S_0$ , (2)  $\forall i \cdot 0 \leq i < n \cdot T(s_i, \sigma_i, s_{i+1})$ , (3)  $s_n = s$ .

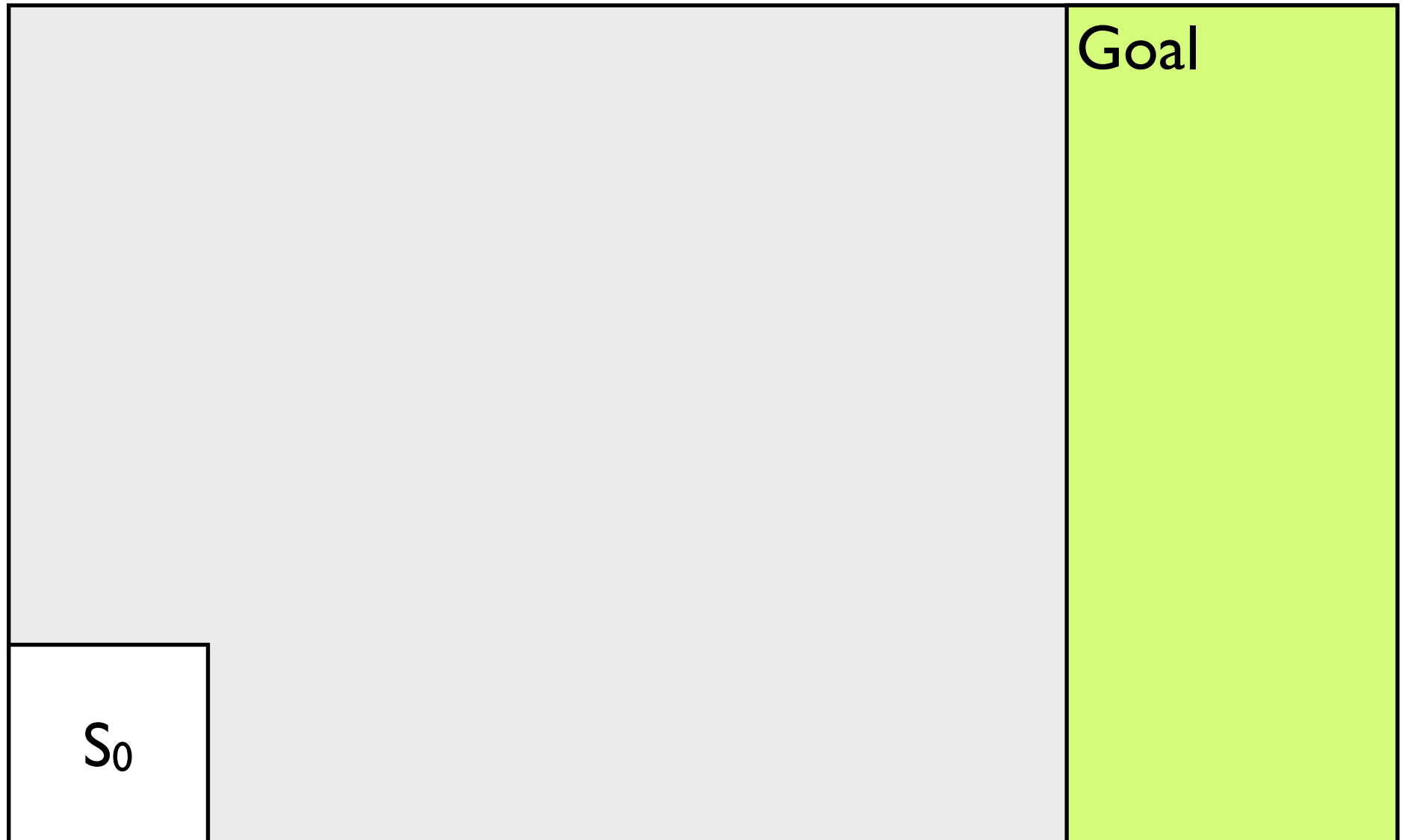
Let **Reach**( $S_0$ ) denote the set of reachable states.

- Clearly, there is a path that starts in  $S_0$  and reaches  $G$  iff **Reach**( $S_0$ )  $\cap$  **Goal**  $\neq \emptyset$ .

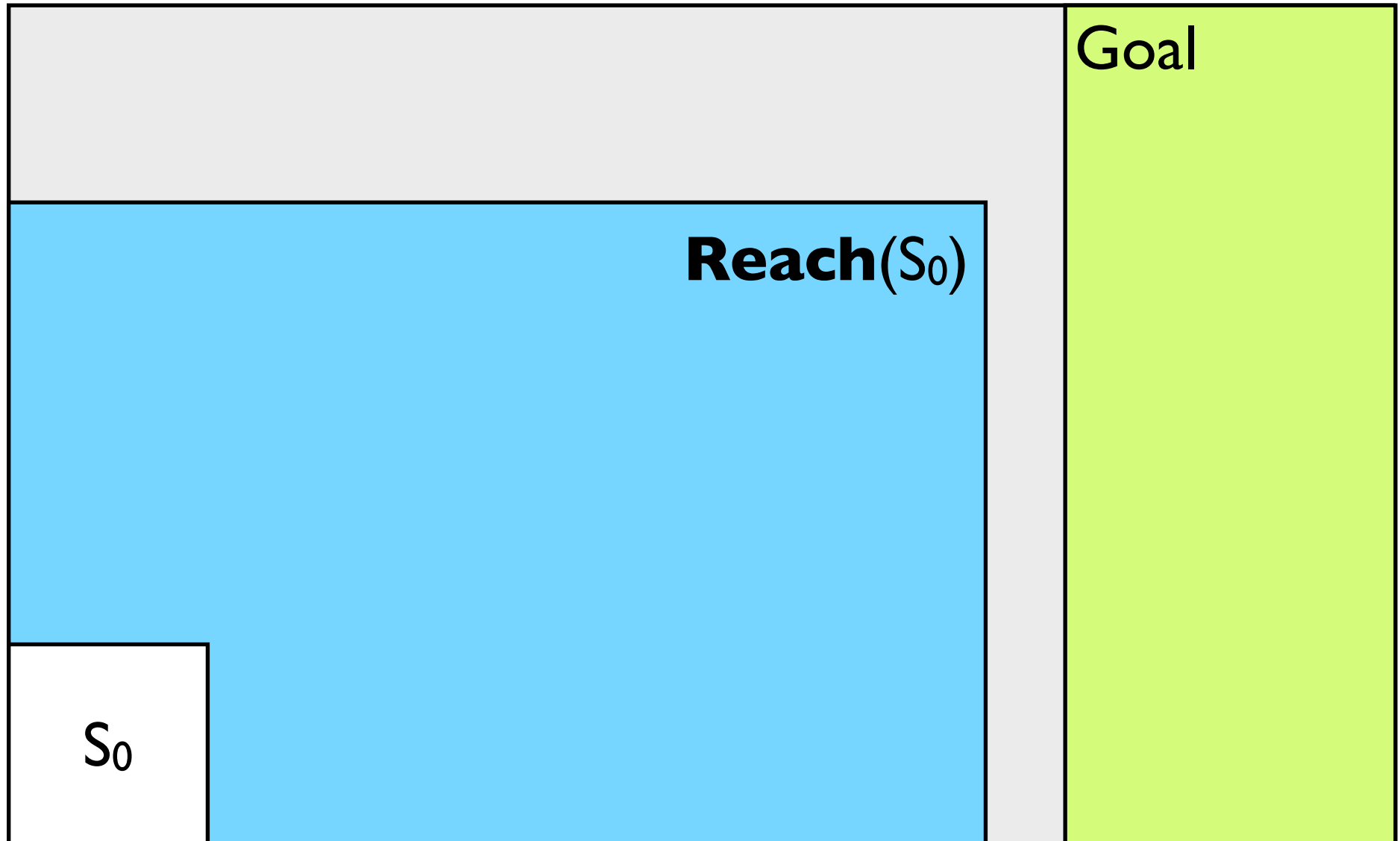
# Reachability



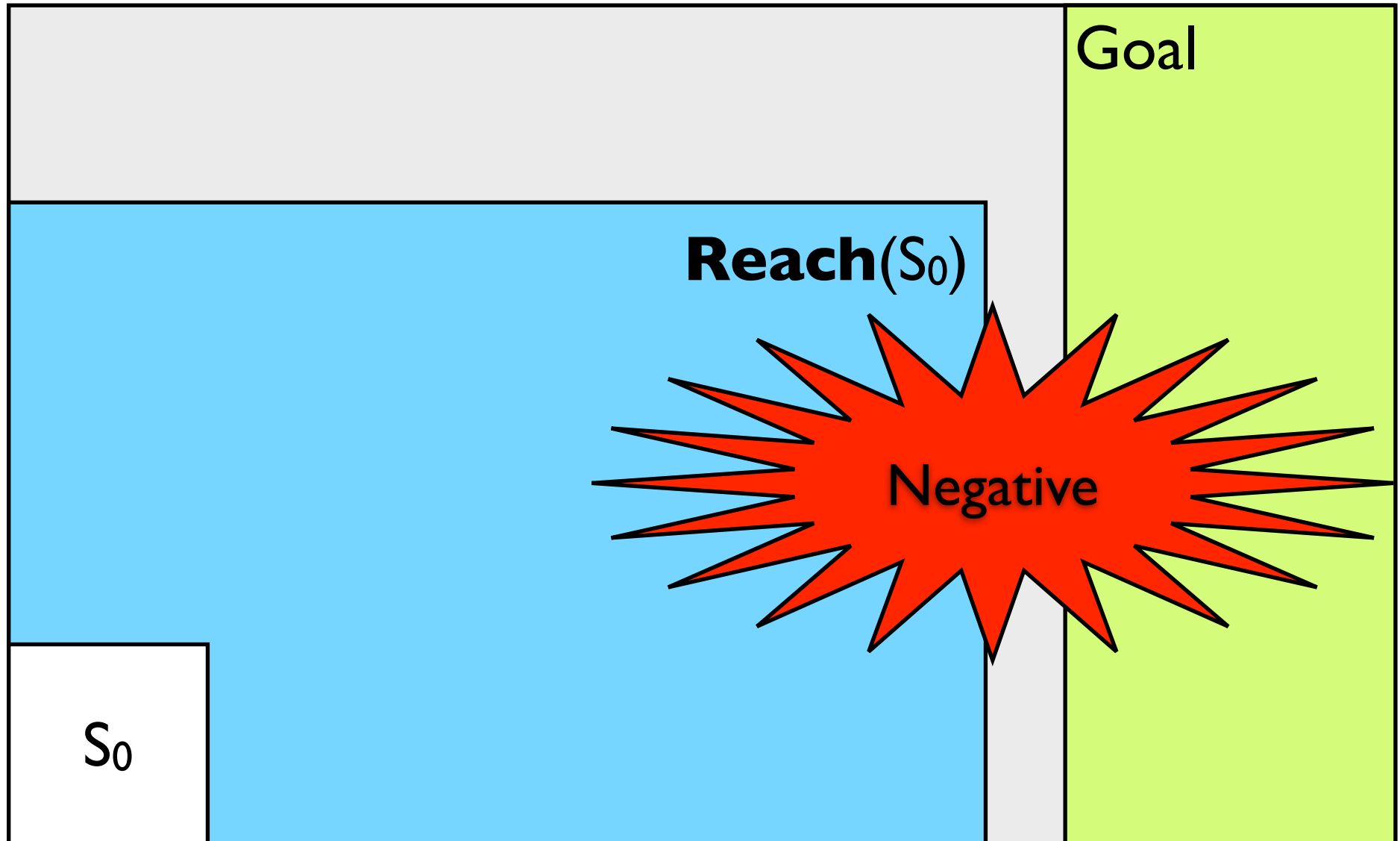
# Reachability



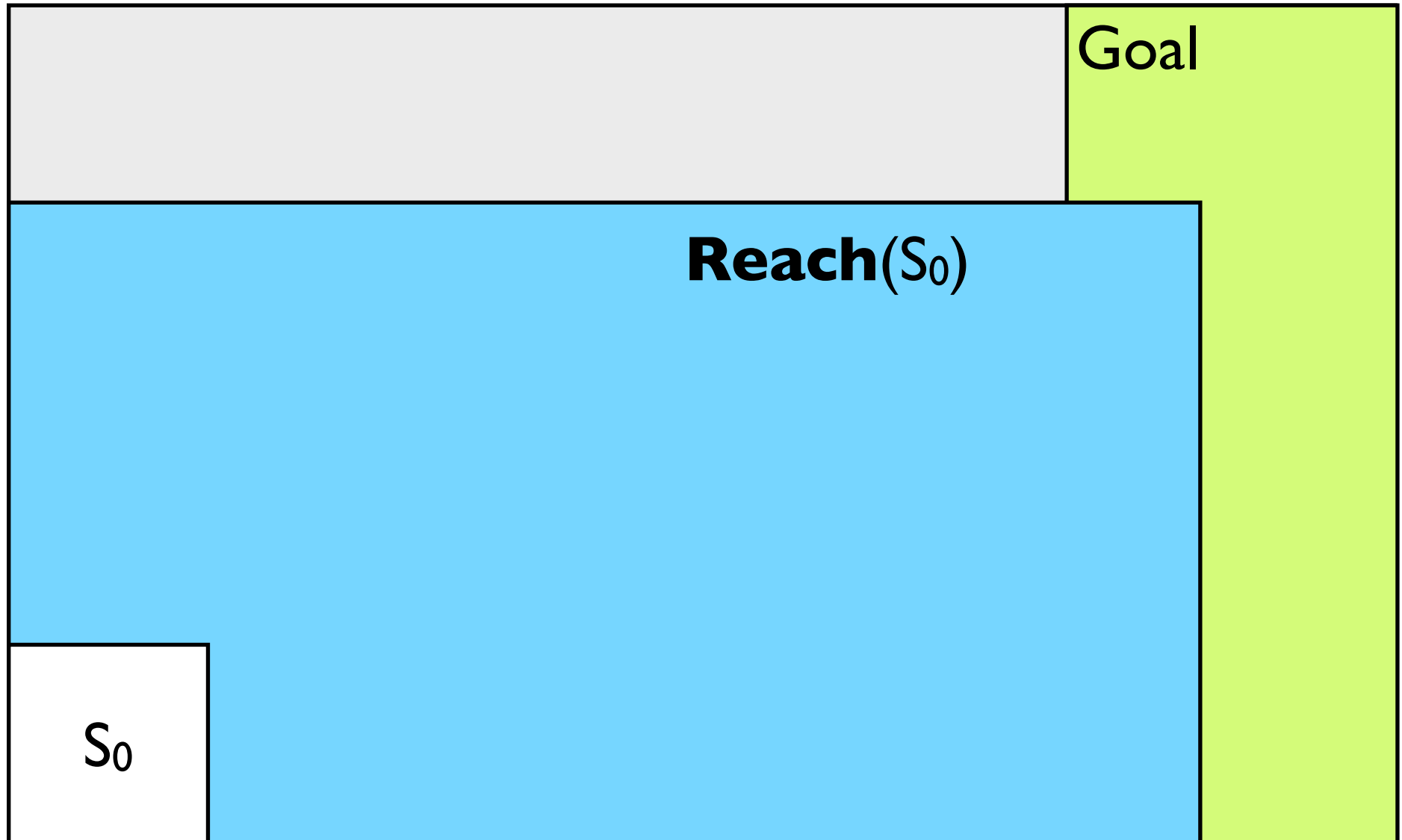
# Reachability



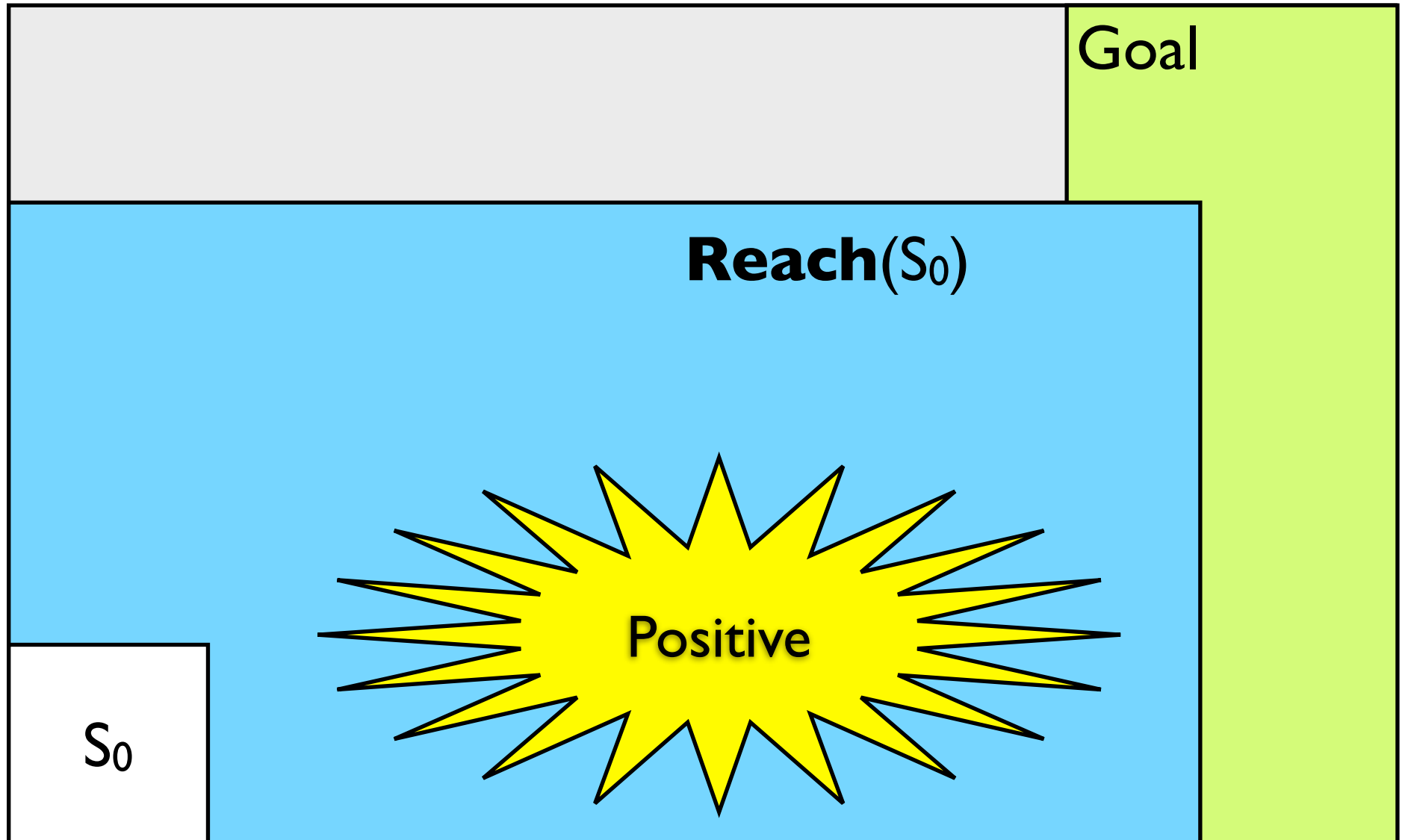
# Reachability



# Reachability



# Reachability



# Safety

- **Safety verification problem**

**Instance:** a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , a set of states **Safe**  $\subseteq S$ .

**Question:** are all paths that starts in  $S_0$  staying within **Safe**.

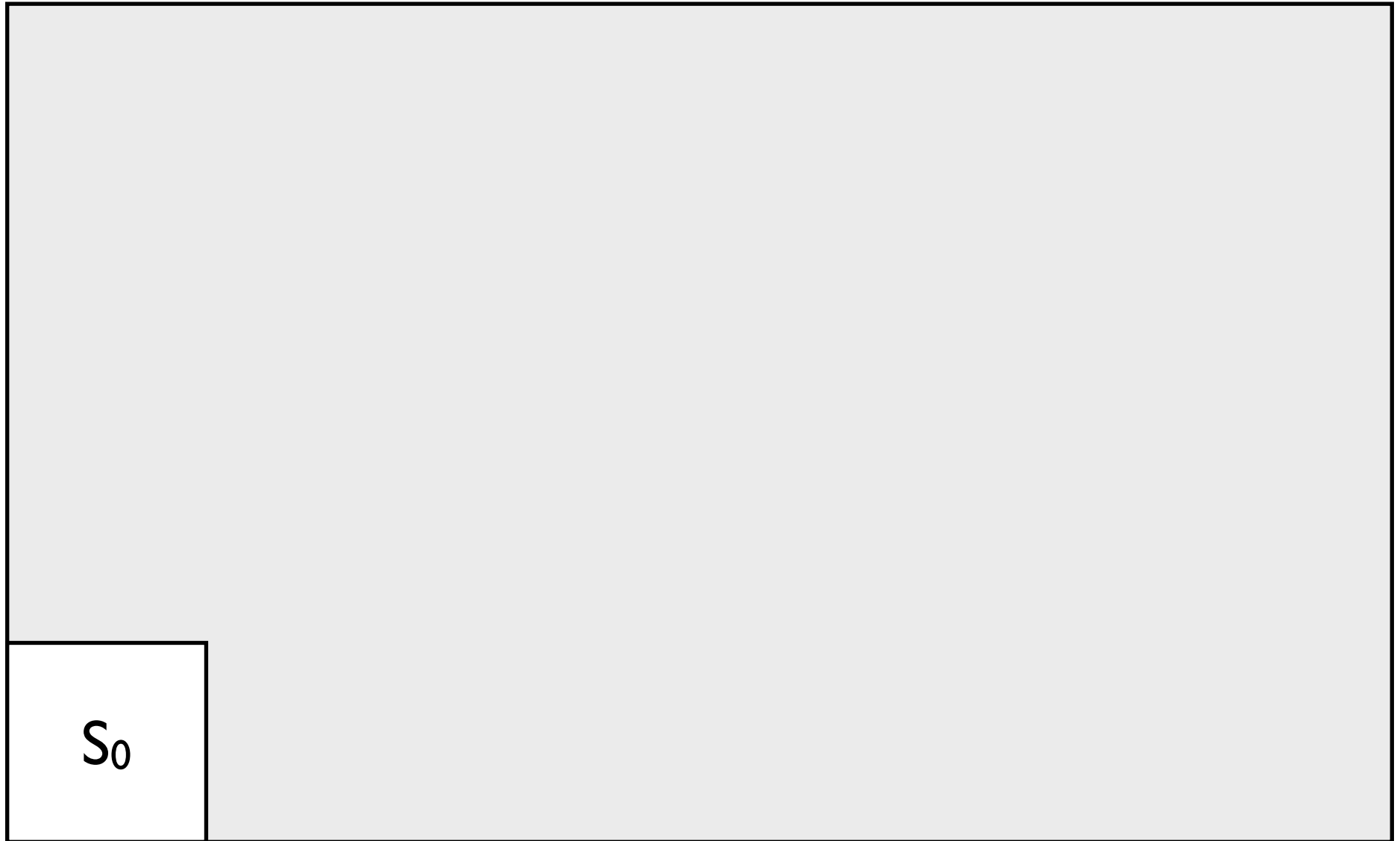
More formally, for all sequences  $s_0\sigma_0s_1\sigma_1s_2\sigma_2\dots\sigma_{n-1}s_n$  such that

(1)  $s_0 \in S_0$ , (2)  $\forall i \cdot 0 \leq i < n \cdot T(s_i, \sigma_i, s_{i+1})$ ,

is it the case that (3)  $\forall i \cdot 0 \leq i \leq n \cdot s_i \in \text{Safe}$  ?

- Clearly all paths that start in  $S_0$  are staying within **Safe**  
iff  **$\text{Reach}(S_0) \cap (S \setminus \text{Safe}) = \emptyset$** .
- So, the safety and reachability problems are **dual problems**.

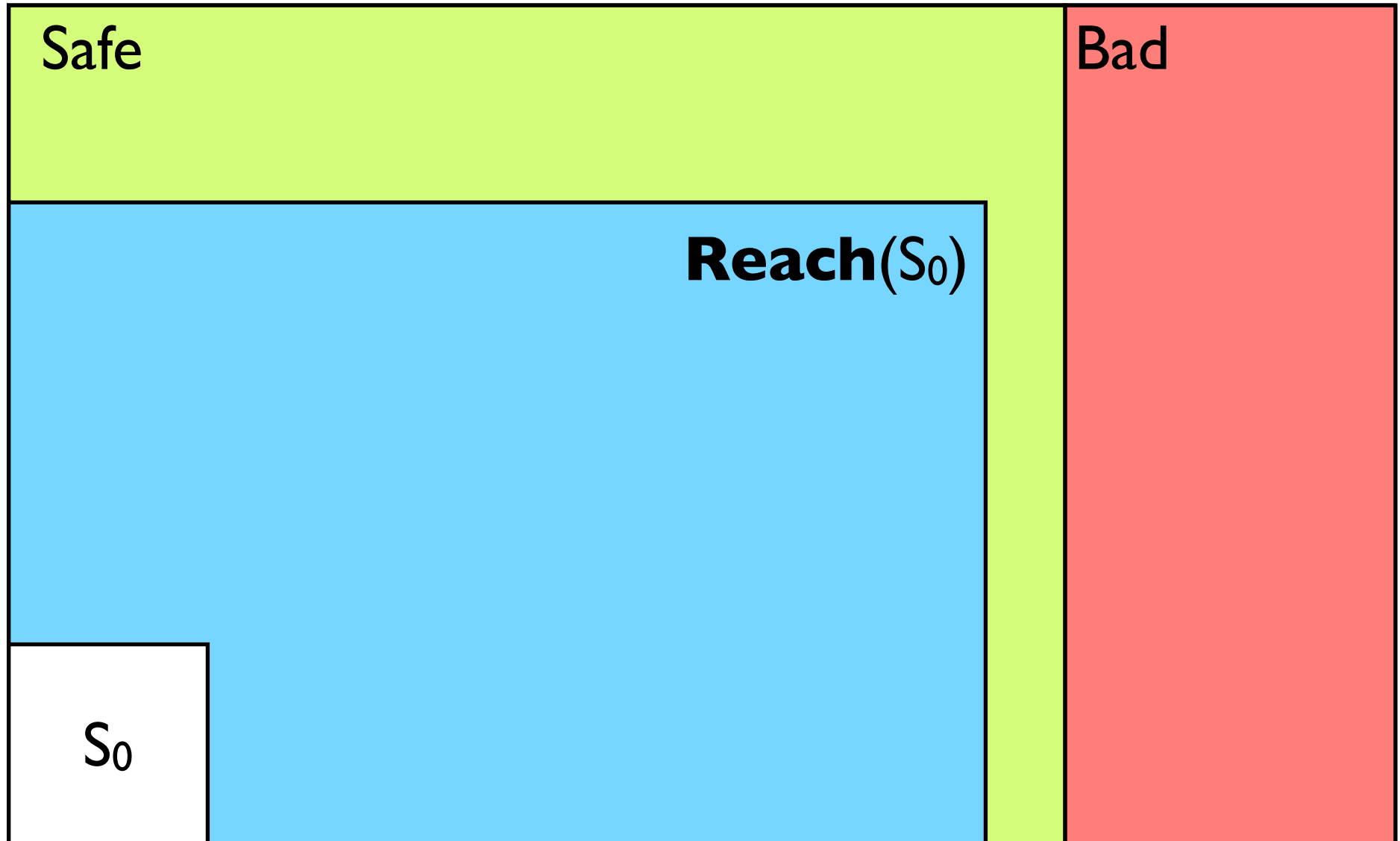
# Safety



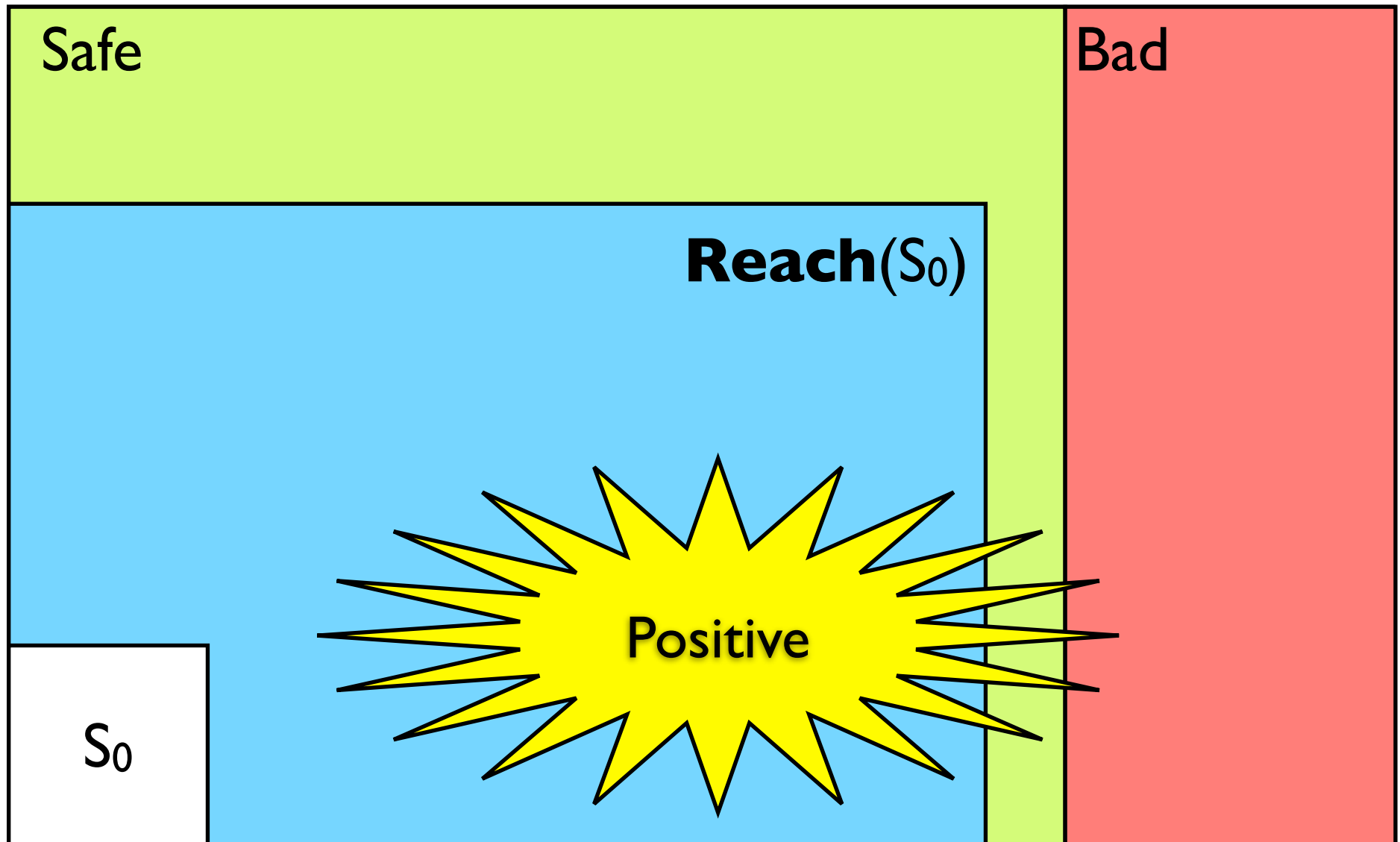
# Safety



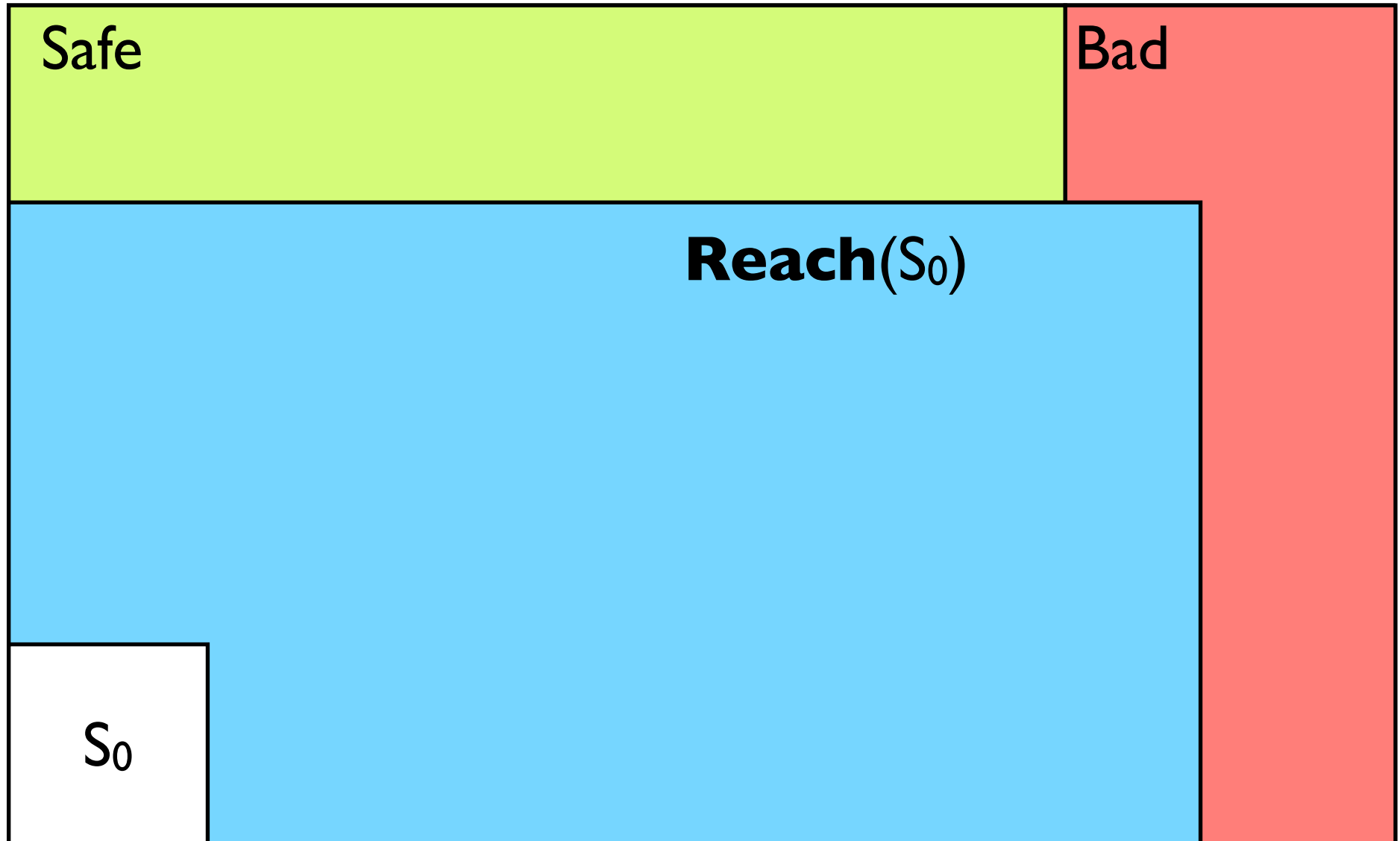
# Safety



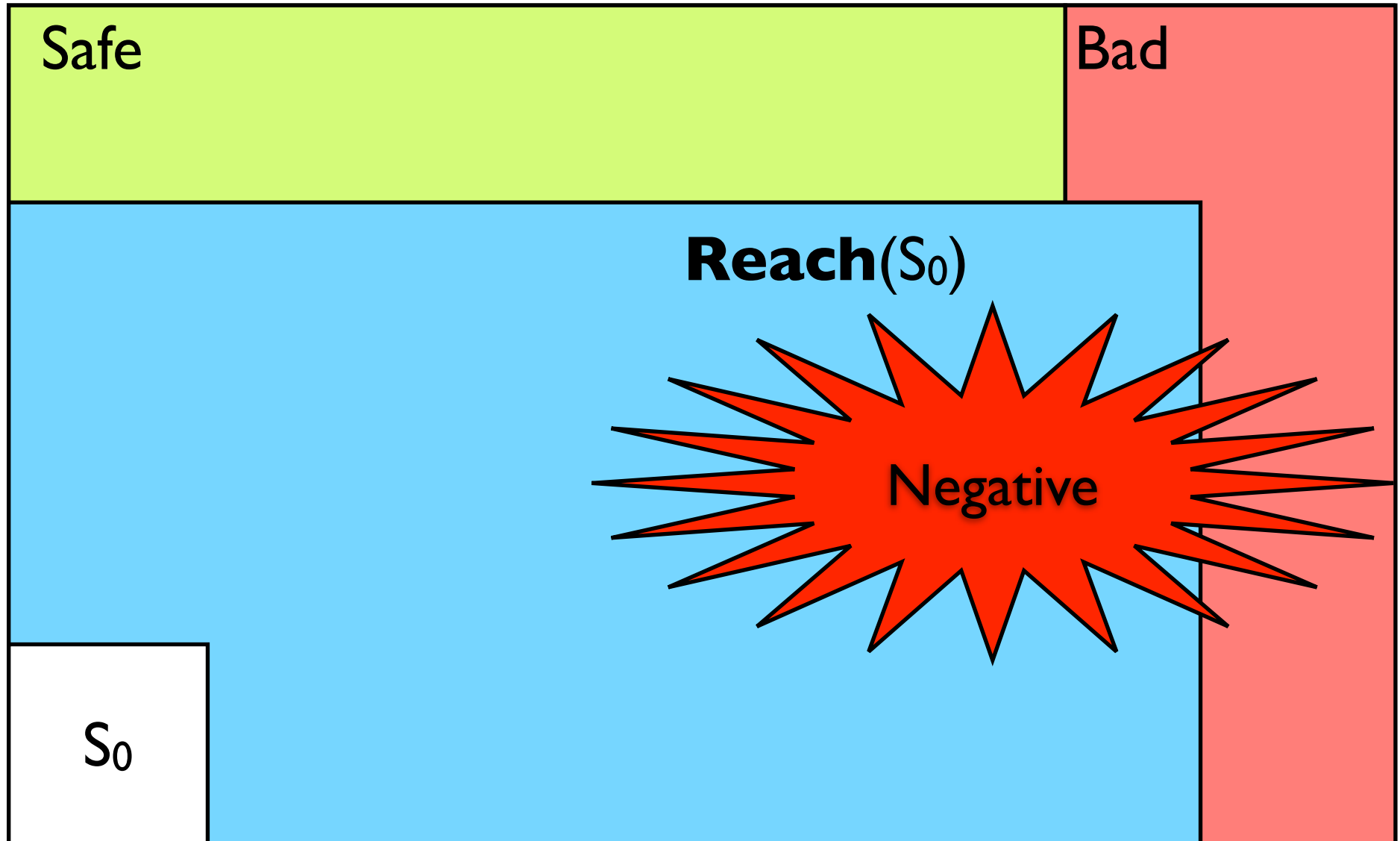
# Safety



# Safety



# Safety



# Büchi condition

- **Büchi verification problem**

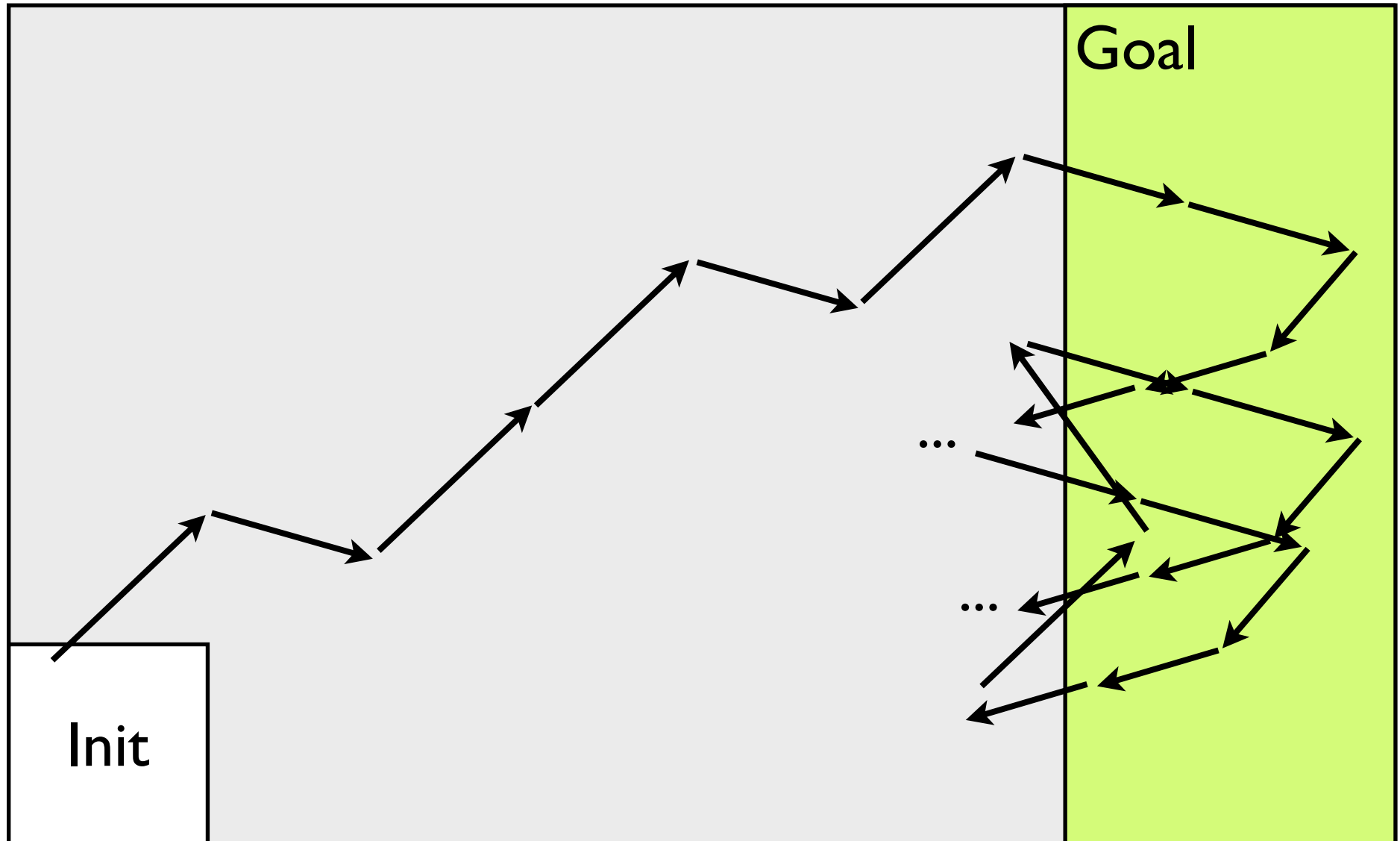
**Instance:** a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , a set  $\text{Goal} \subseteq S$ .

**Question:** is there one execution of the LTS that starts in  $S_0$  and passes infinitely often by the set  $\text{Goal} \subseteq S$  ?

More formally, is there an execution  $s_0\sigma_0s_1\sigma_1s_2\sigma_2...\sigma_{n-1}s_n...$  such that

- (1)  $s_0 \in S_0$ ,
- (2)  $\forall i \cdot 0 \leq i \cdot T(s_i, \sigma_i, s_{i+1})$ ,
- (3)  $\forall i \geq 0 \cdot \exists j \geq i$  such that  $s_j \in \text{Goal}$  ?

# Büchi condition



# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

# Post, Pre and Apre operators

- We will design verification algorithms for the reachability, safety and Büchi properties.
- Our algorithms will manipulate **sets of states**.
- Besides set operations, we will need to compute the set of states that are successors (Post), or predecessors (Pre and Apre) of a set of states.

The **Post** :  $2^S \times \Sigma \rightarrow 2^S$

takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of successors of  $X$  by  $\sigma$

$$\mathbf{Post}(X, \sigma) = \{ y \in X \mid \exists x \in X \cdot T(x, \sigma, y) \}$$

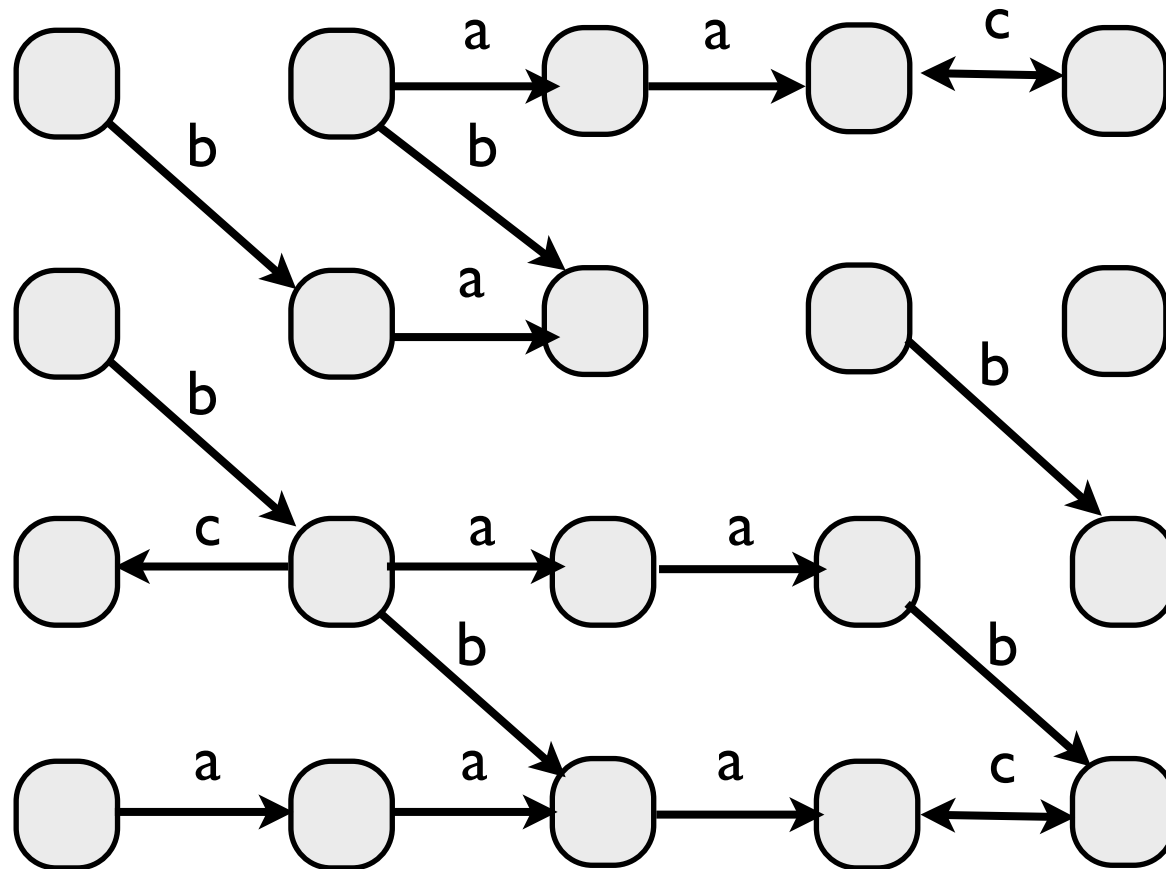
The **Post** :  $2^S \times \Sigma \rightarrow 2^S$

takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of successors of  $X$  by  $\sigma$

$$\mathbf{Post}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(x, \sigma, y) \}$$



The **Post** :  $2^S \times \Sigma \rightarrow 2^S$

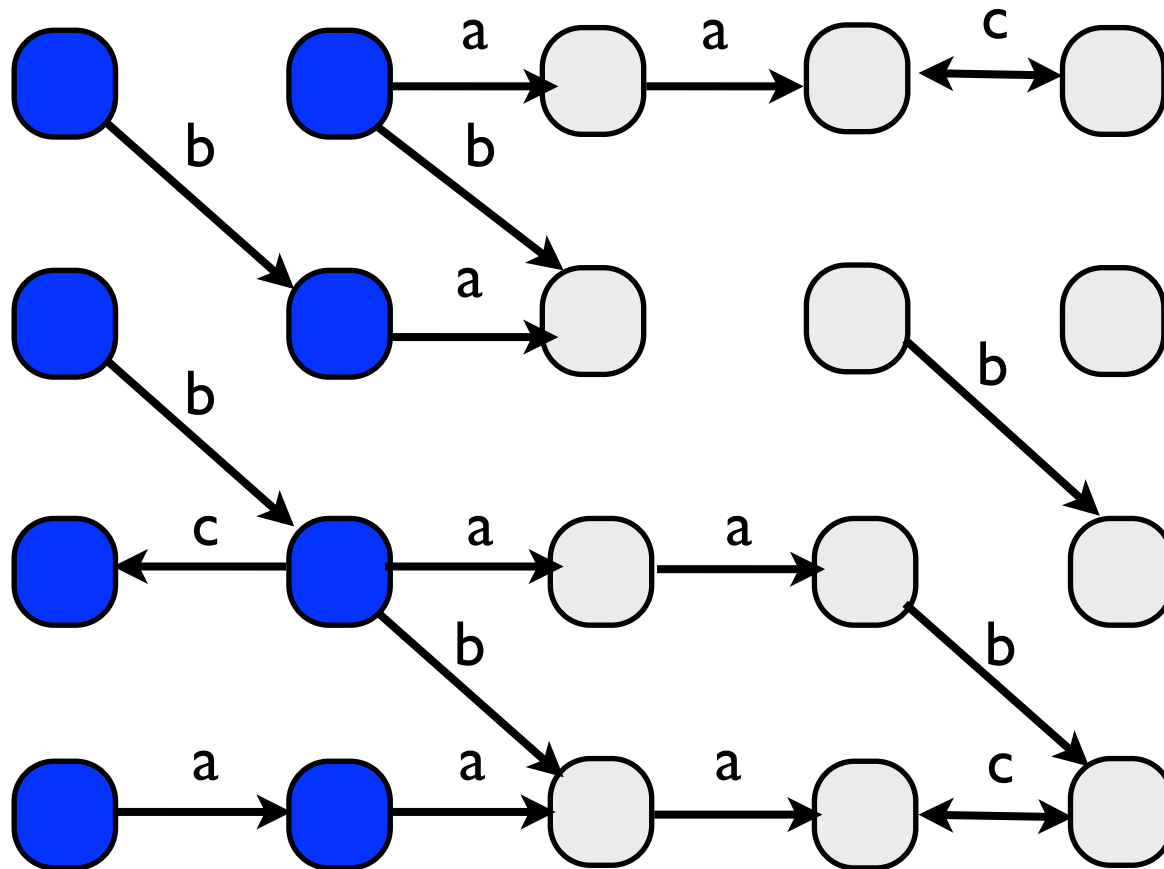
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of successors of  $X$  by  $\sigma$

$$\mathbf{Post}(X,\sigma)=\{ y \in X \mid \exists x \in X \cdot T(x,\sigma,y) \}$$

**Post**(X,b)=



The **Post** :  $2^S \times \Sigma \rightarrow 2^S$

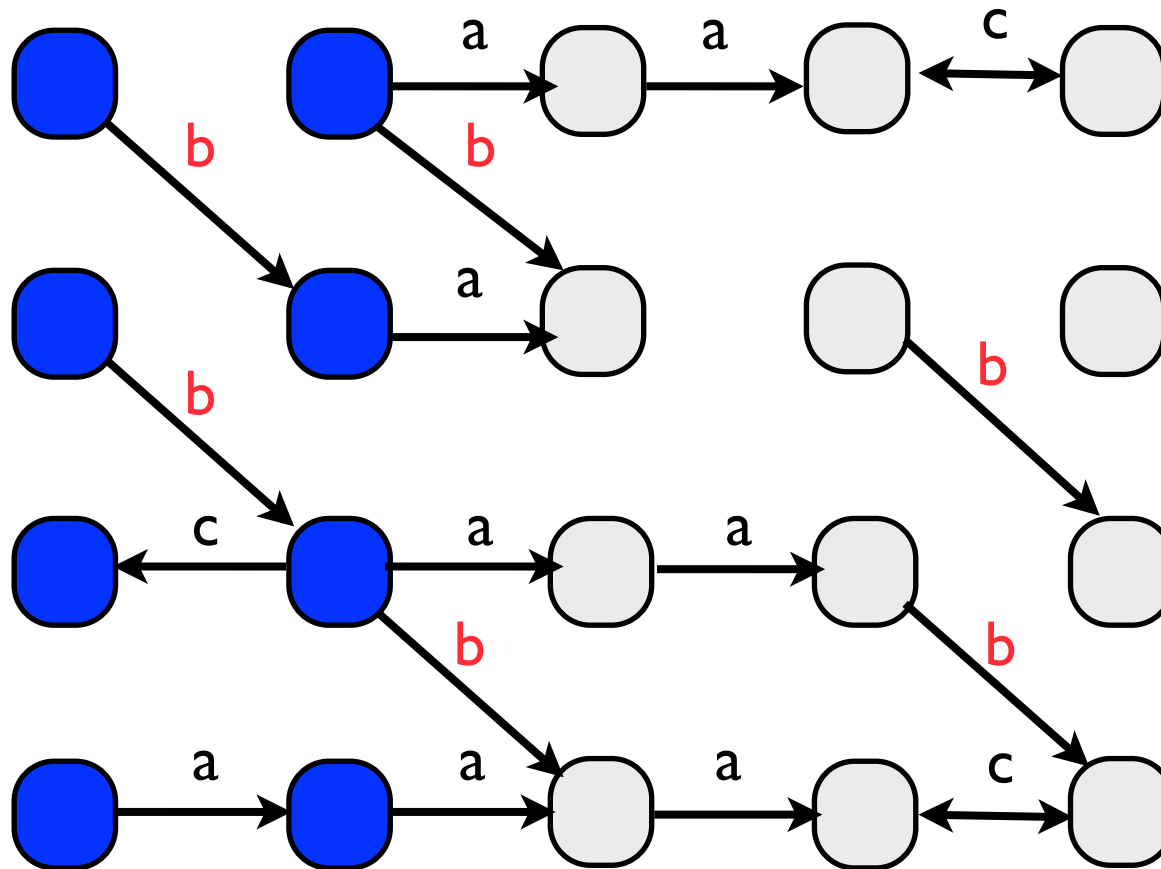
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of successors of  $X$  by  $\sigma$

$$\mathbf{Post}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(x, \sigma, y) \}$$

**Post**( $X, b$ ) =



The **Post** :  $2^S \times \Sigma \rightarrow 2^S$

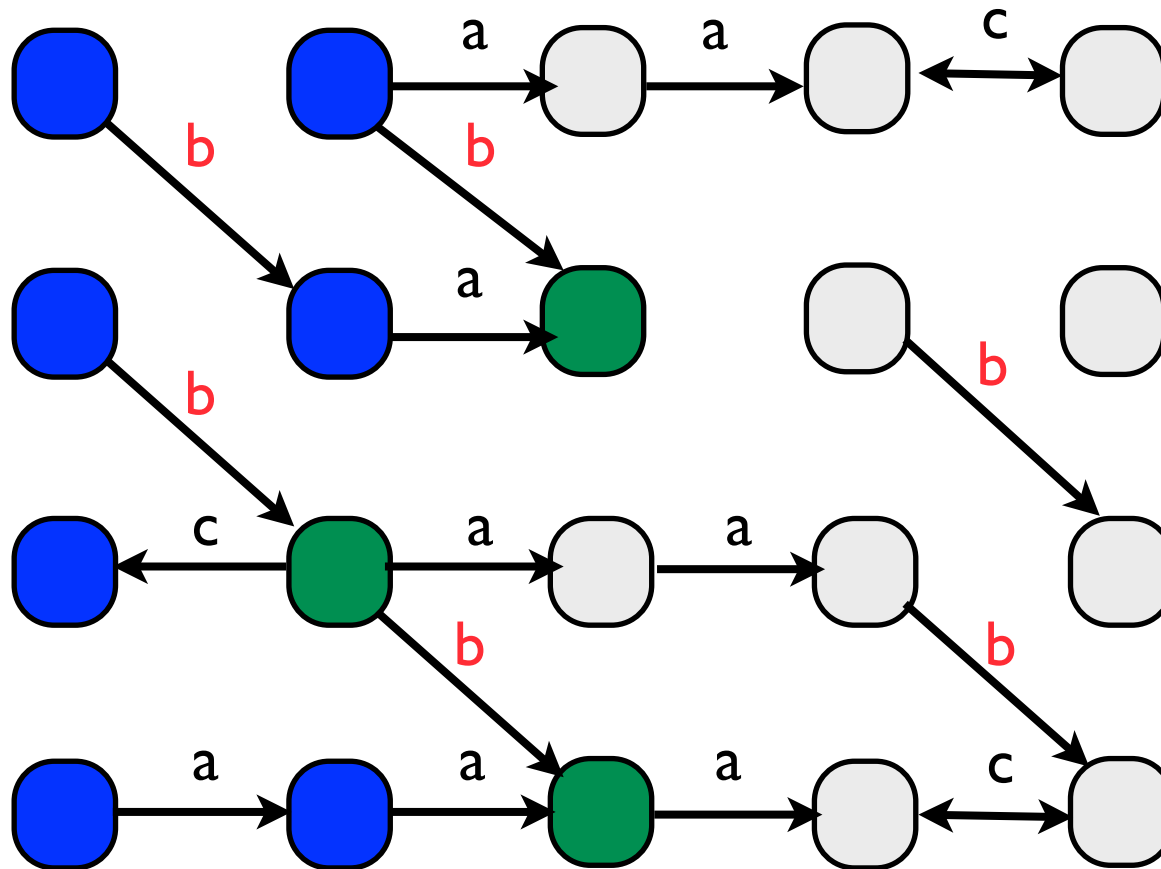
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of successors of  $X$  by  $\sigma$

$$\mathbf{Post}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(x, \sigma, y) \}$$

$$\mathbf{Post}(X, b) = Y$$



The **Pre** :  $2^S \times \Sigma \rightarrow 2^S$

takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of predecessors of  $X$  by  $\sigma$

$$\mathbf{Pre}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(y, \sigma, x) \}$$

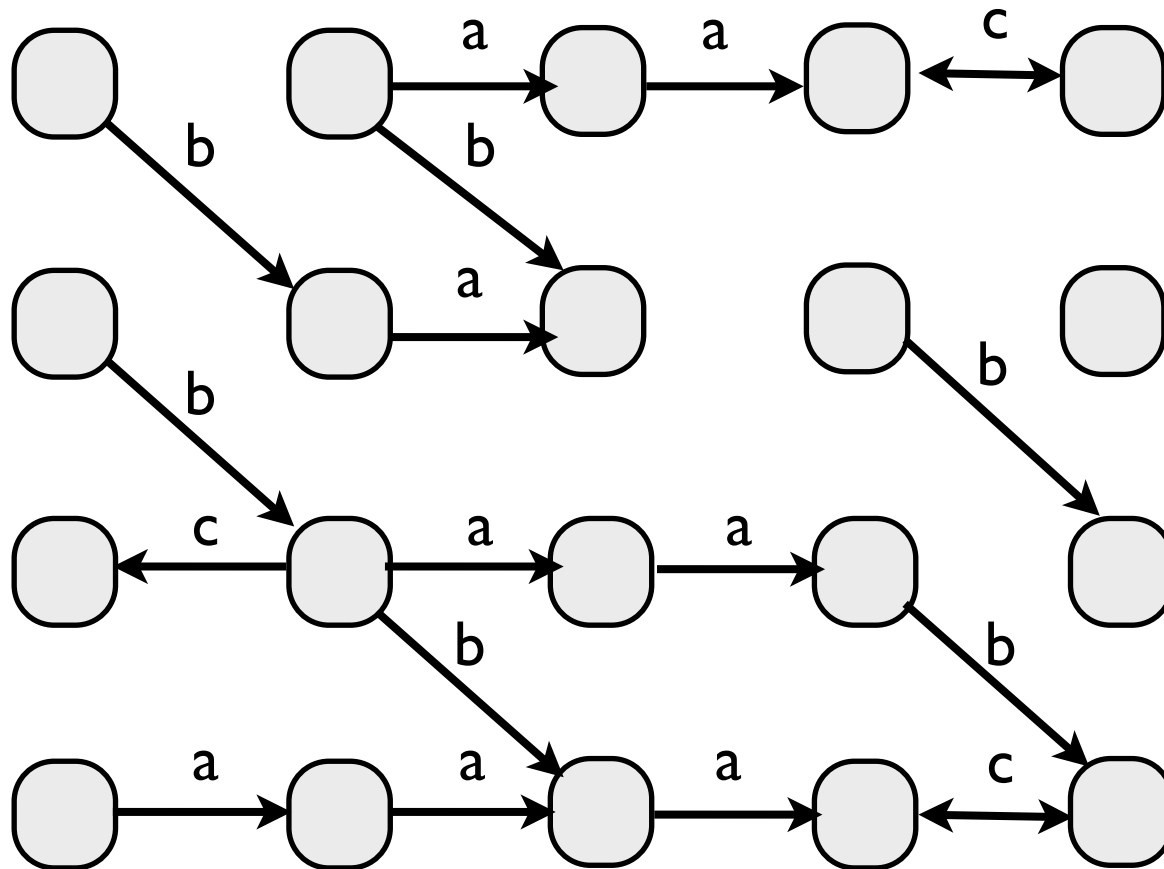
The **Pre** :  $2^S \times \Sigma \rightarrow 2^S$

takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of predecessors of  $X$  by  $\sigma$

$$\mathbf{Pre}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(y, \sigma, x) \}$$



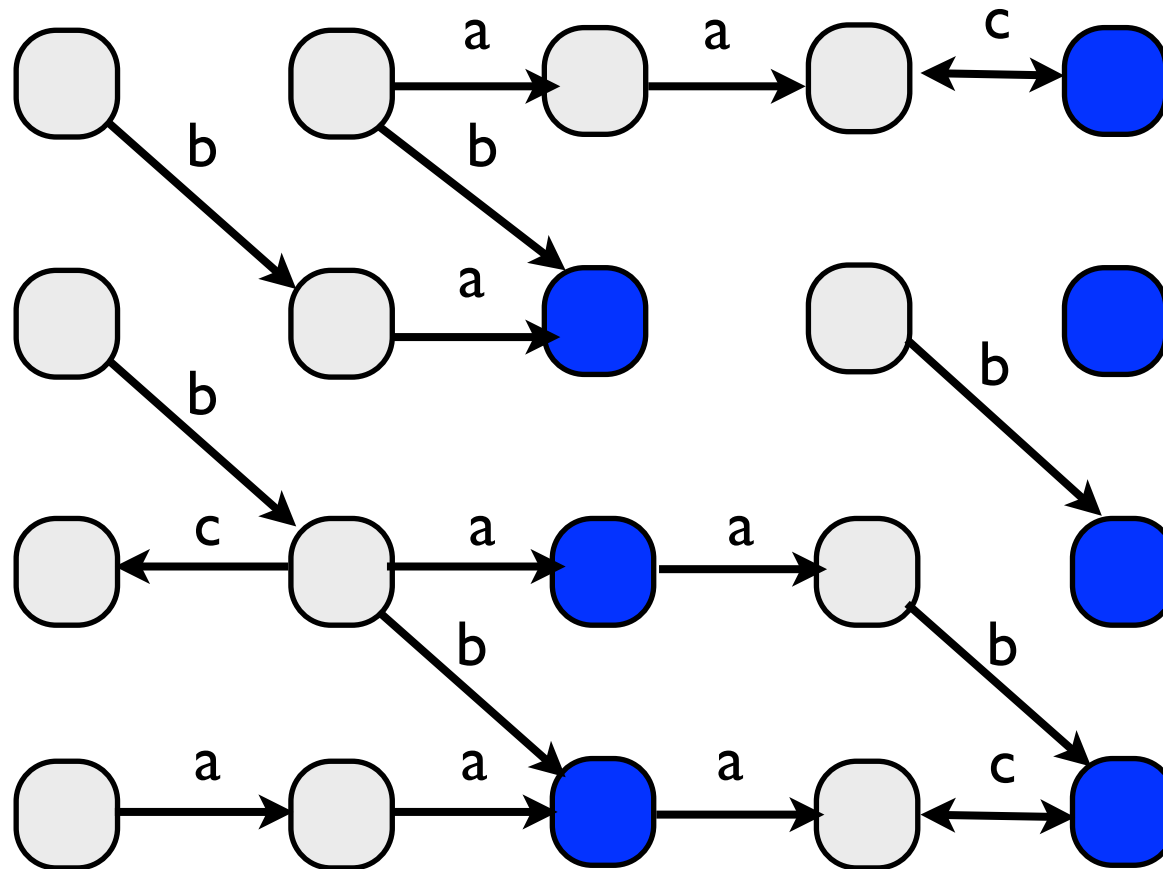
The **Pre** :  $2^S \times \Sigma \rightarrow 2^S$   
 takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of predecessors of  $X$  by  $\sigma$

$$\mathbf{Pre}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(y, \sigma, x) \}$$

**Pre**( $X, b$ ) =



The **Pre** :  $2^S \times \Sigma \rightarrow 2^S$

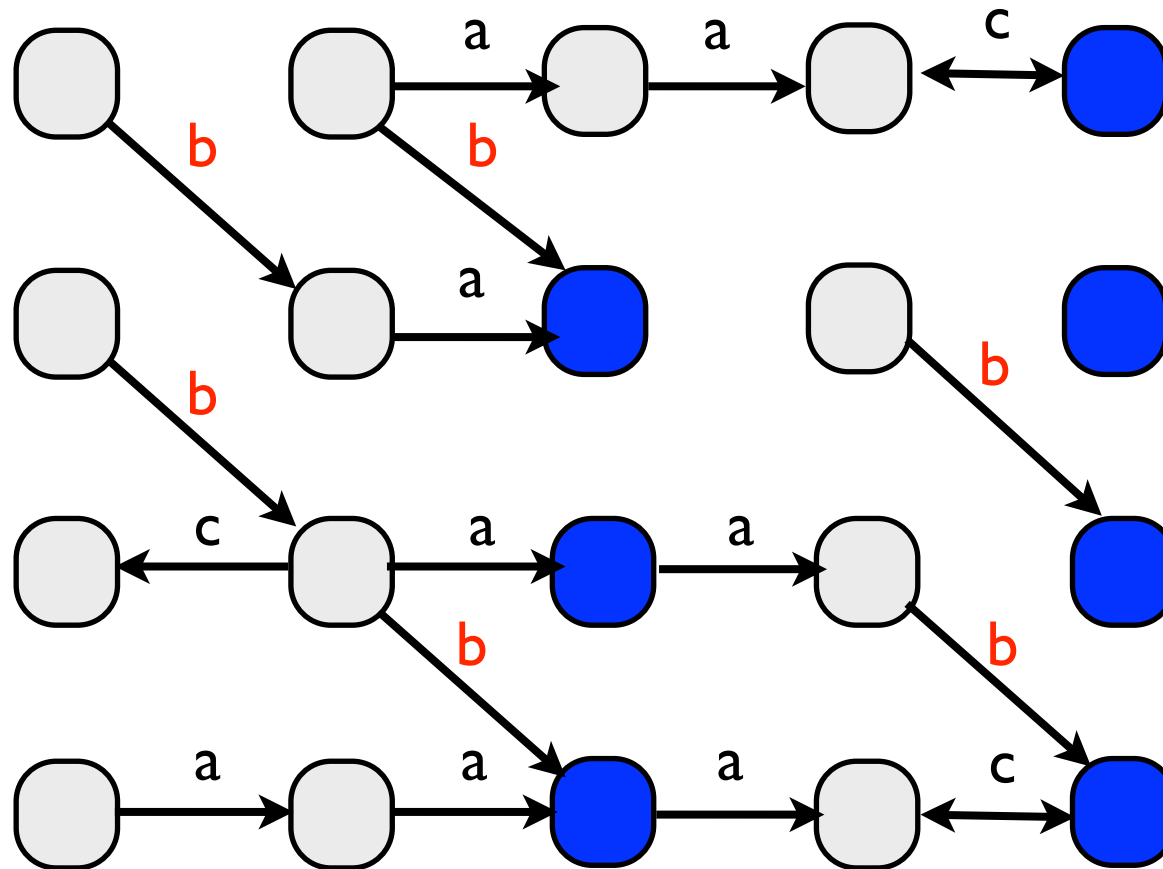
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of predecessors of  $X$  by  $\sigma$

$$\mathbf{Pre}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(y, \sigma, x) \}$$

**Pre**( $X, b$ ) =



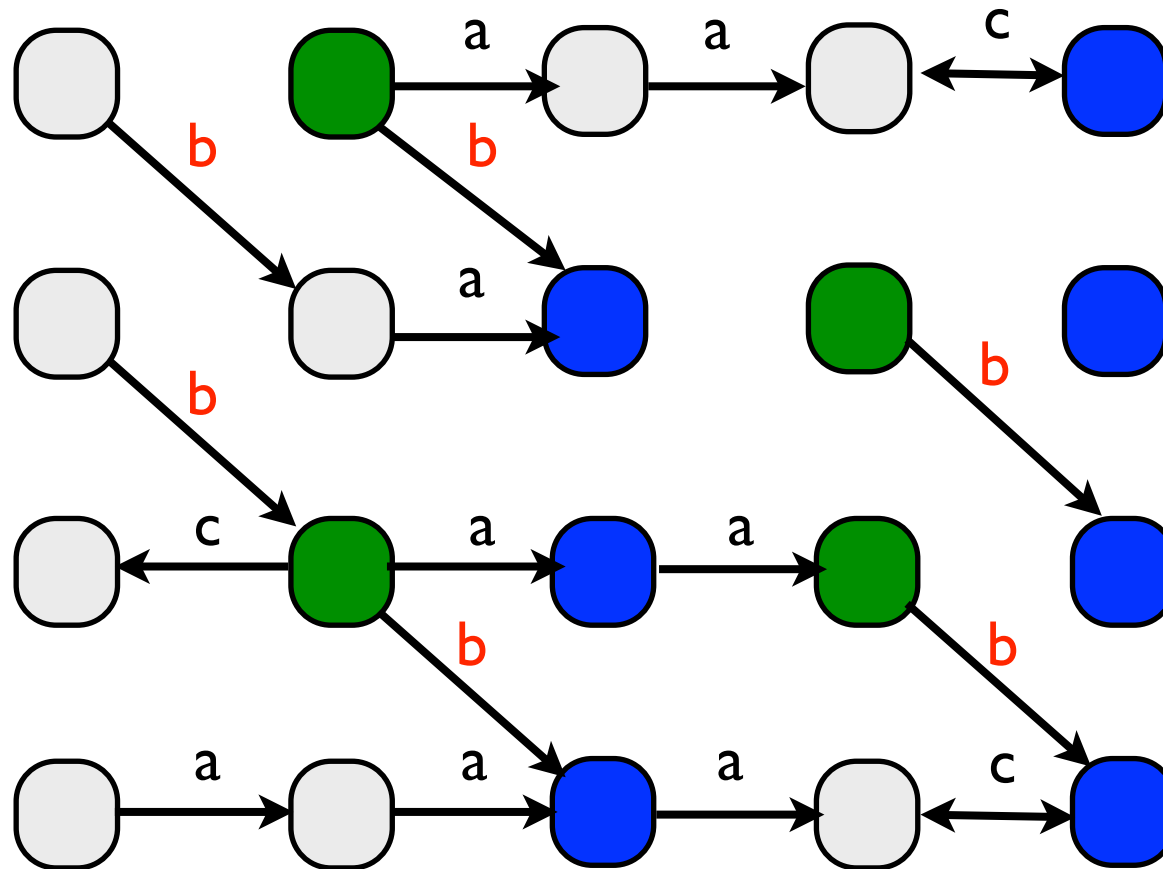
The **Pre** :  $2^S \times \Sigma \rightarrow 2^S$   
 takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of predecessors of  $X$  by  $\sigma$

$$\mathbf{Pre}(X, \sigma) = \{ y \in S \mid \exists x \in X \cdot T(y, \sigma, x) \}$$

$$\mathbf{Pre}(X, b) = Y$$



The **Apre** :  $2^S \times \Sigma \rightarrow 2^S$

takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of states that have all their successors by  $\sigma$  in  $X$

$$\mathbf{Apre}(X, \sigma) = \{ x \in S \mid \forall y \in S \cdot T(x, \sigma, y) \Rightarrow y \in X \}$$

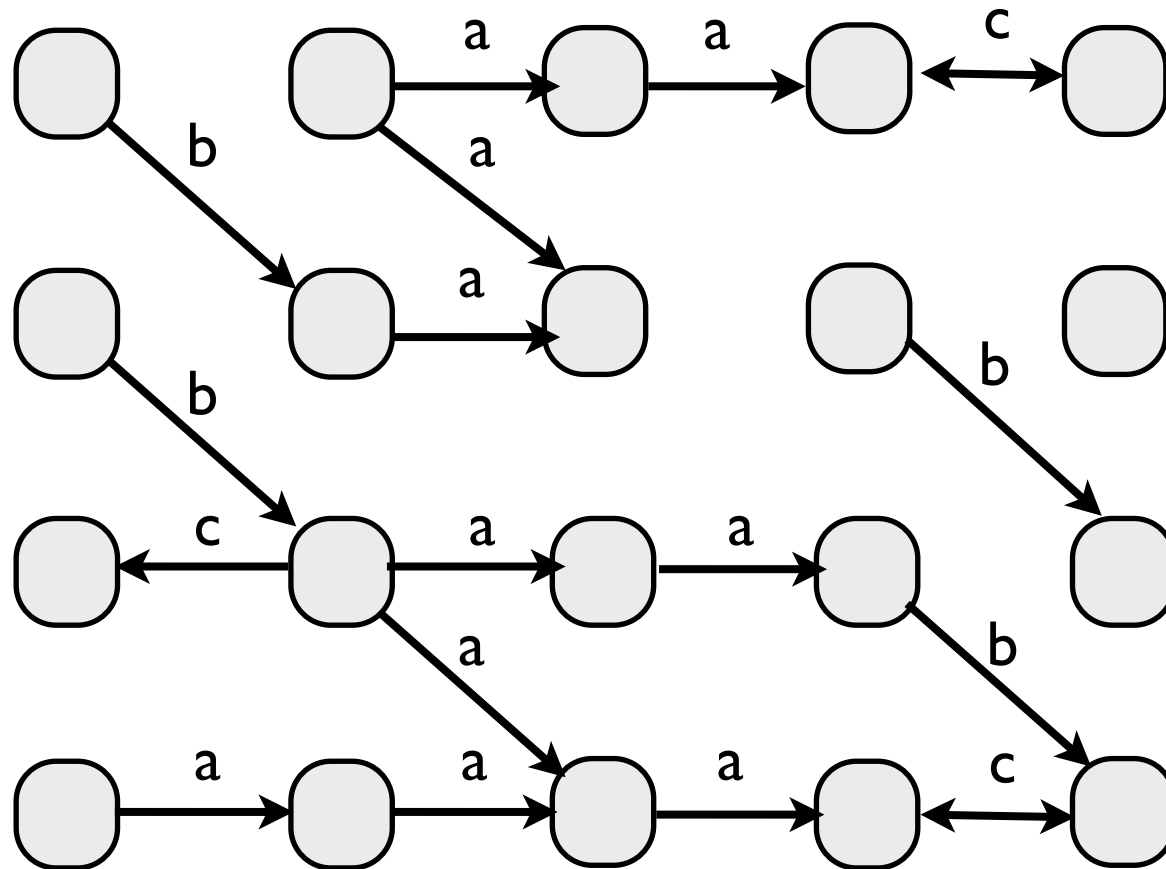
The **Apre** :  $2^S \times \Sigma \rightarrow 2^S$

takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of states that have all their successors by  $\sigma$  in  $X$

$$\mathbf{Apre}(X, \sigma) = \{ x \in S \mid \forall y \in S \cdot T(x, \sigma, y) \Rightarrow y \in X \}$$



The **Apre** :  $2^S \times \Sigma \rightarrow 2^S$

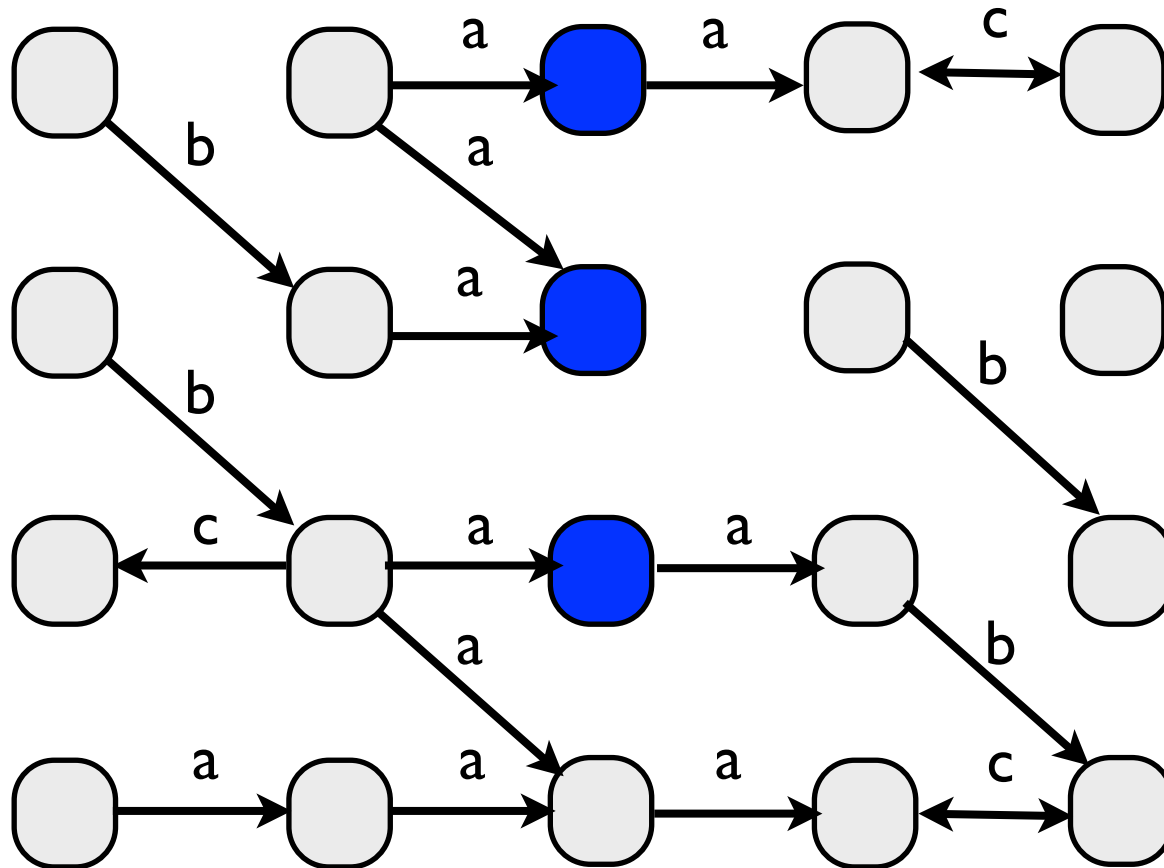
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of states that have all their successors by  $\sigma$  in  $X$

$$\mathbf{Apre}(X, \sigma) = \{ x \in S \mid \forall y \in S \cdot T(x, \sigma, y) \Rightarrow y \in X \}$$

**Apre**( $X, a$ )



The **Apre** :  $2^S \times \Sigma \rightarrow 2^S$

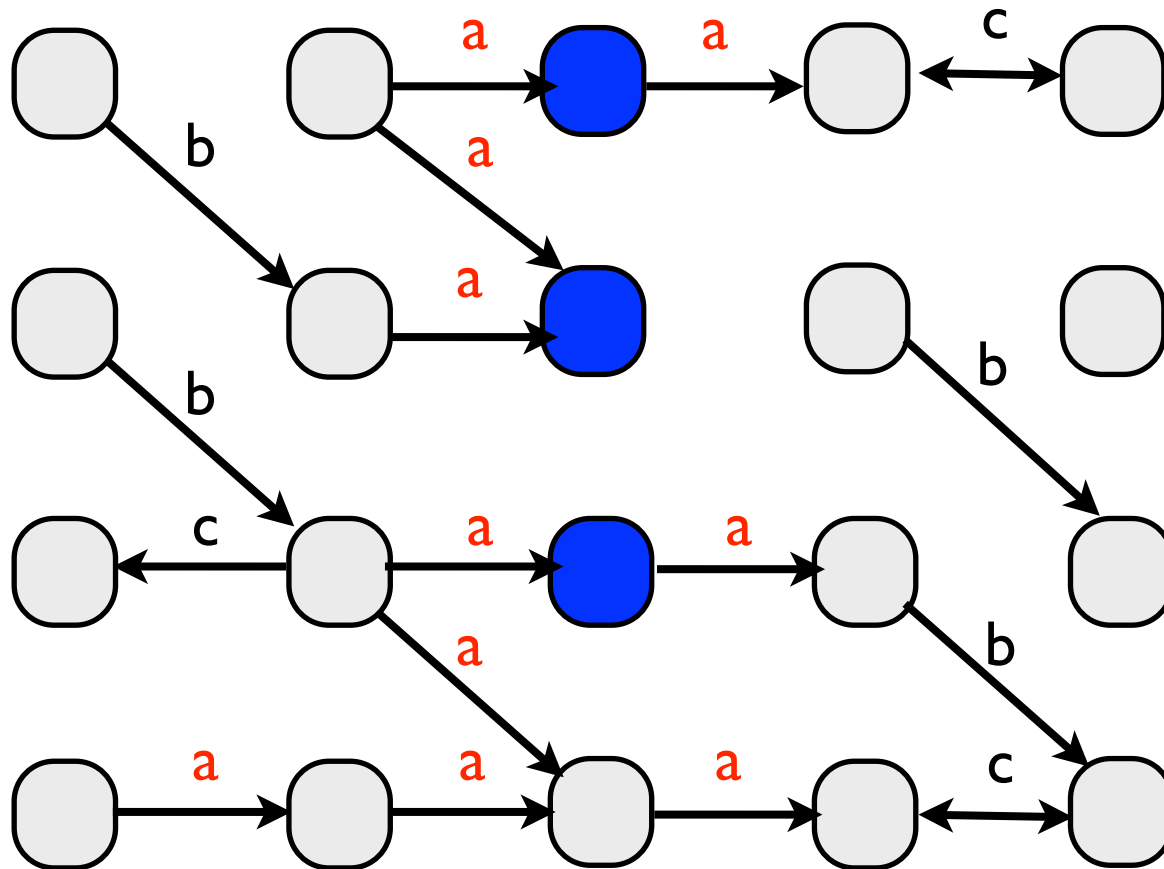
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of states that have all their successors by  $\sigma$  in  $X$

$$\mathbf{Apre}(X, \sigma) = \{ x \in S \mid \forall y \in S \cdot T(x, \sigma, y) \Rightarrow y \in X \}$$

**Apre**( $X, a$ )



The **Apre** :  $2^S \times \Sigma \rightarrow 2^S$

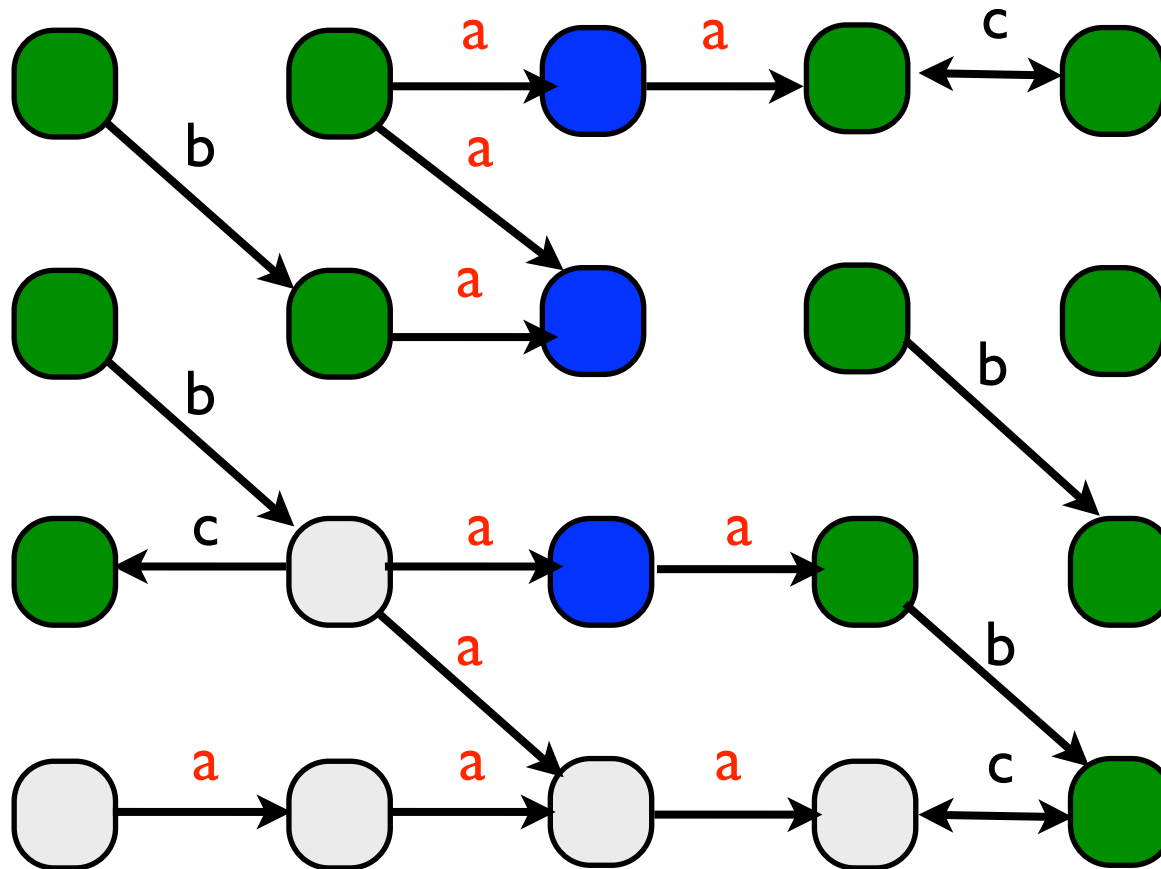
takes (1) a set of states  $X$

(2) an action  $\sigma$

and returns the set of states that have all their successors by  $\sigma$  in  $X$

$$\mathbf{Apre}(X, \sigma) = \{ x \in S \mid \forall y \in S \cdot T(x, \sigma, y) \Rightarrow y \in X \}$$

$$\mathbf{Apre}(X, a) = Y$$



# PRE-POST-APRE

From the **Pre** :  $2^S \times \Sigma \rightarrow 2^S$ , the **Post** :  $2^S \times \Sigma \rightarrow 2^S$ , and the **Apre** :  $2^S \times \Sigma \rightarrow 2^S$ , we can define their generalizations over the entire alphabet of actions:

The **POST** :  $2^S \rightarrow 2^S$  takes a set of states  $X$  and returns the set of states  $Y$  that are reachable in one step from  $X$ , i.e. :

$$\mathbf{POST}(X) = \{ y \in S \mid \exists x \in X \cdot \exists \sigma \in \Sigma \cdot T(x, \sigma, y) \}$$

The **PRE** :  $2^S \rightarrow 2^S$  takes a set of states  $X$  and returns the set of states  $Y$  that can reach  $X$  in one step, i.e. :

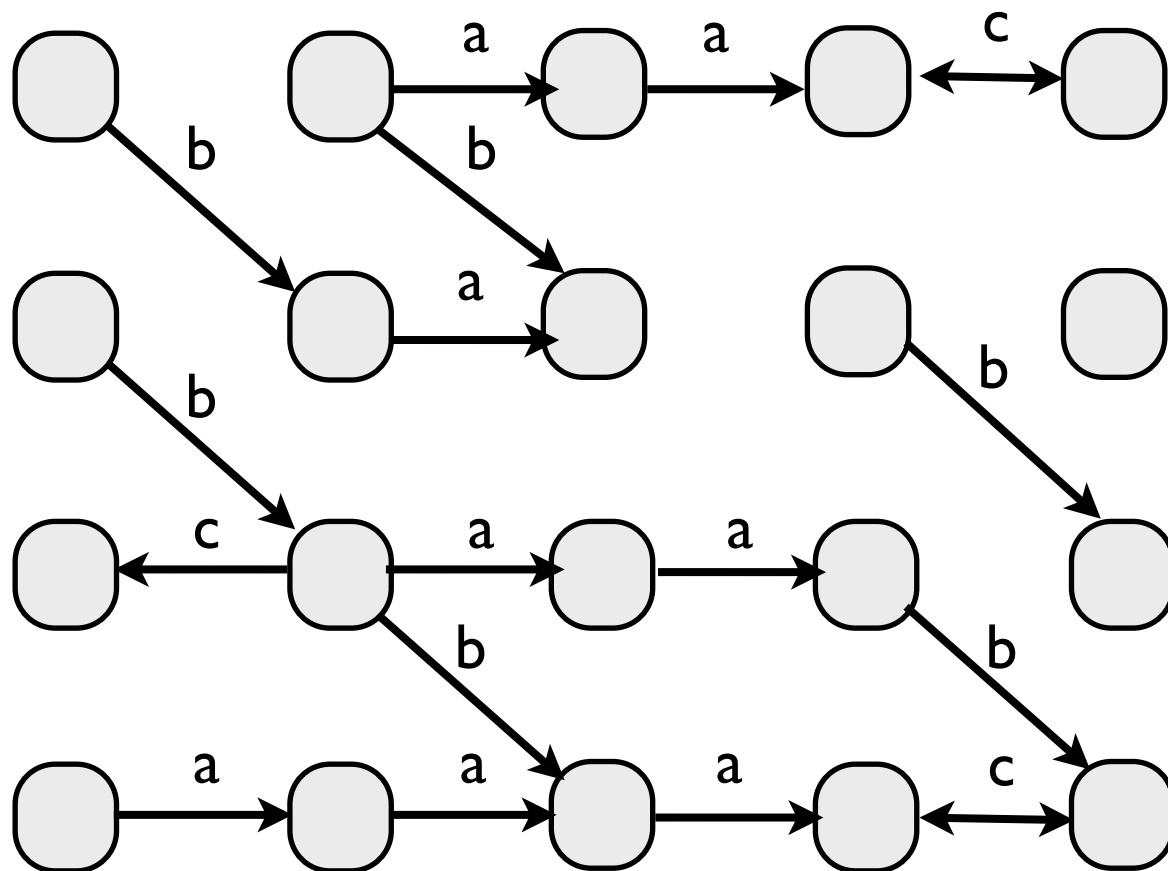
$$\mathbf{PRE}(X) = \{ y \in S \mid \exists x \in X \cdot \exists \sigma \in \Sigma \cdot T(y, \sigma, x) \}$$

The **APRE** :  $2^S \rightarrow 2^S$  takes a set of states  $X$  and returns the set of states  $Y$  that have all their one step successors in  $X$ , i.e. :

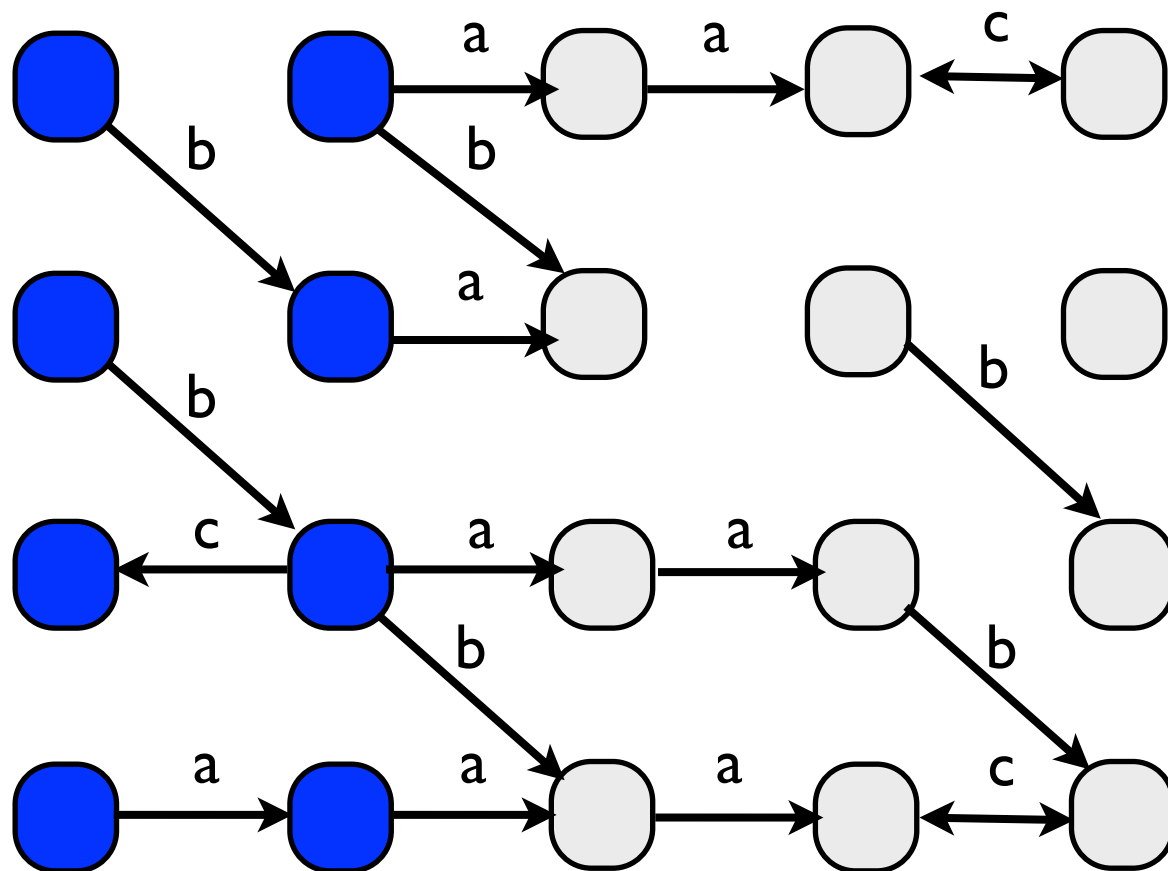
$$\mathbf{APRE}(X) = \{ y \in S \mid \forall x \in S \cdot \forall \sigma \in \Sigma \cdot T(y, \sigma, x) \Rightarrow x \in X \}$$

**Exercise** : proof that  $\mathbf{APRE}(X) = S \setminus \mathbf{PRE}(S \setminus X)$ .

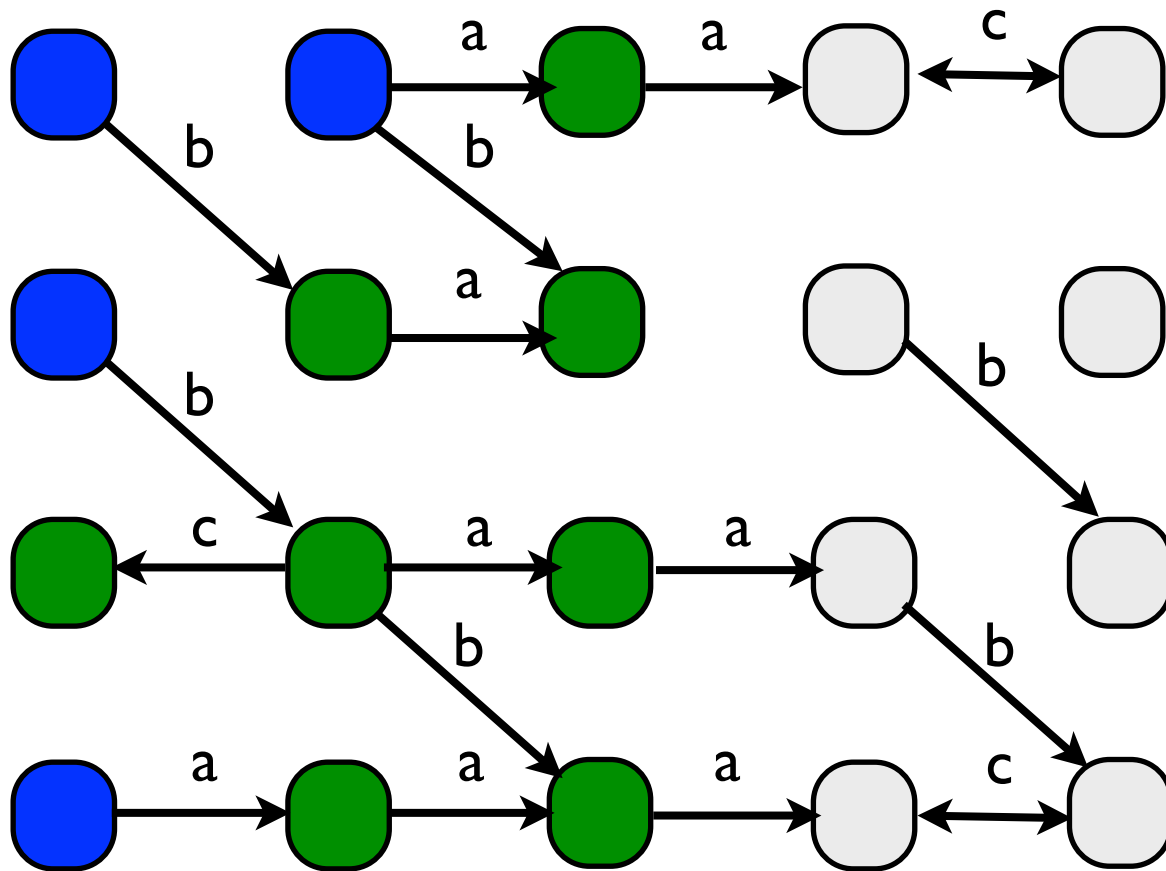
**POST**(X)=



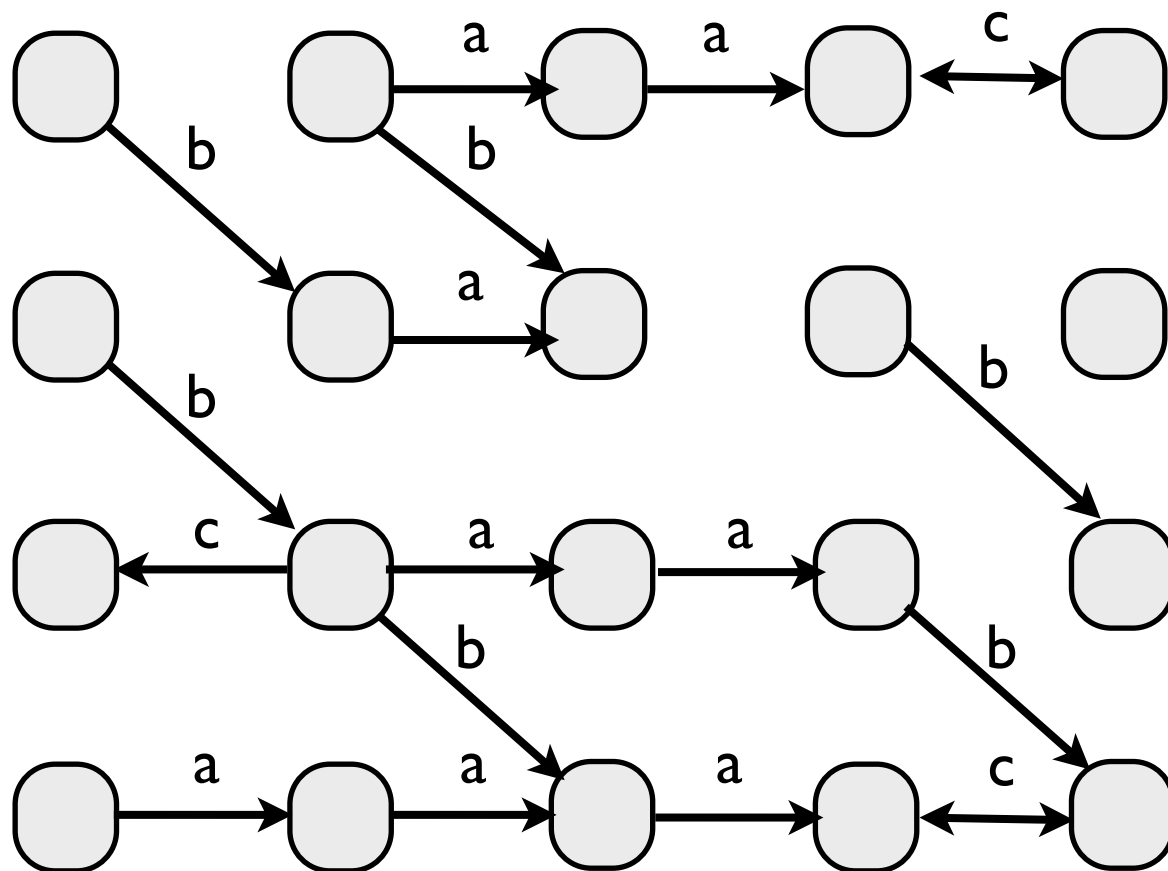
**POST**(X)=



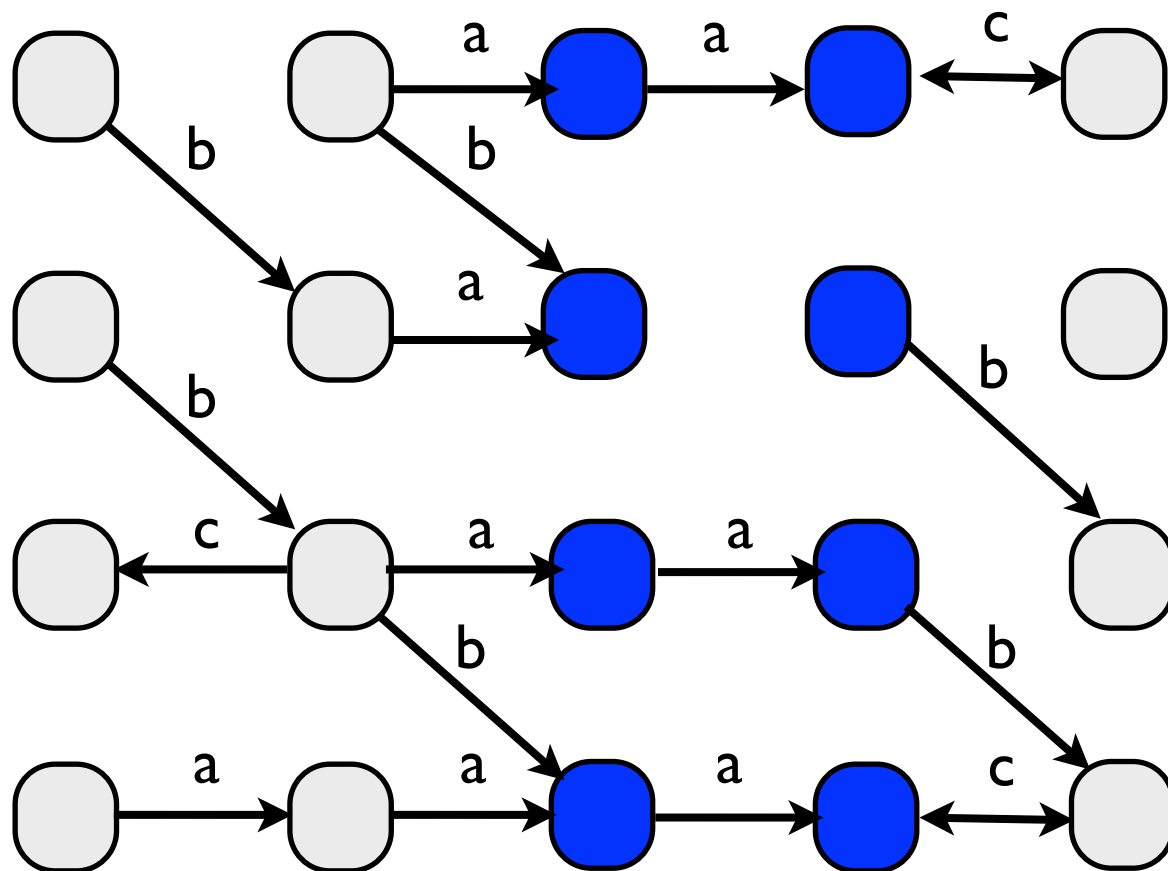
$$\mathbf{POST}(\mathbf{X})=\mathbf{Y}$$



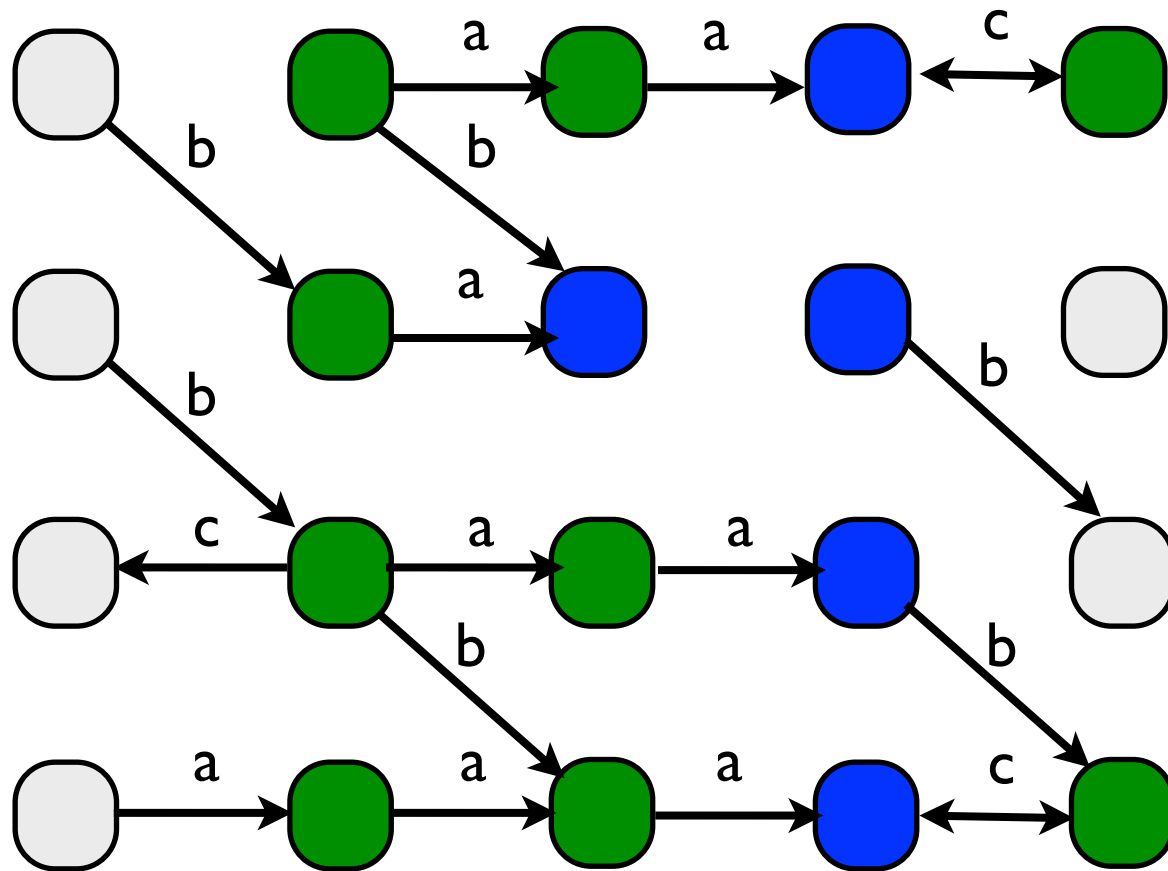
**PRE(X)=**



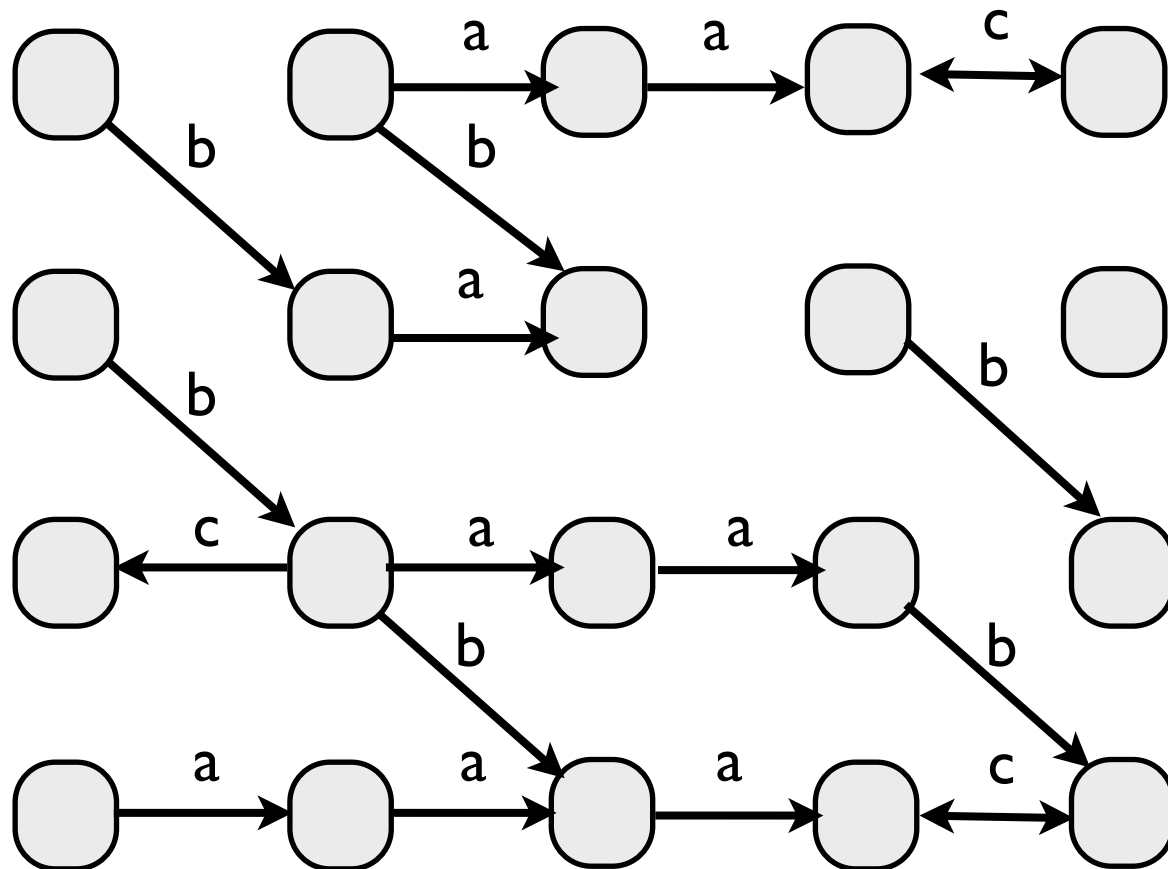
**PRE(X)=**



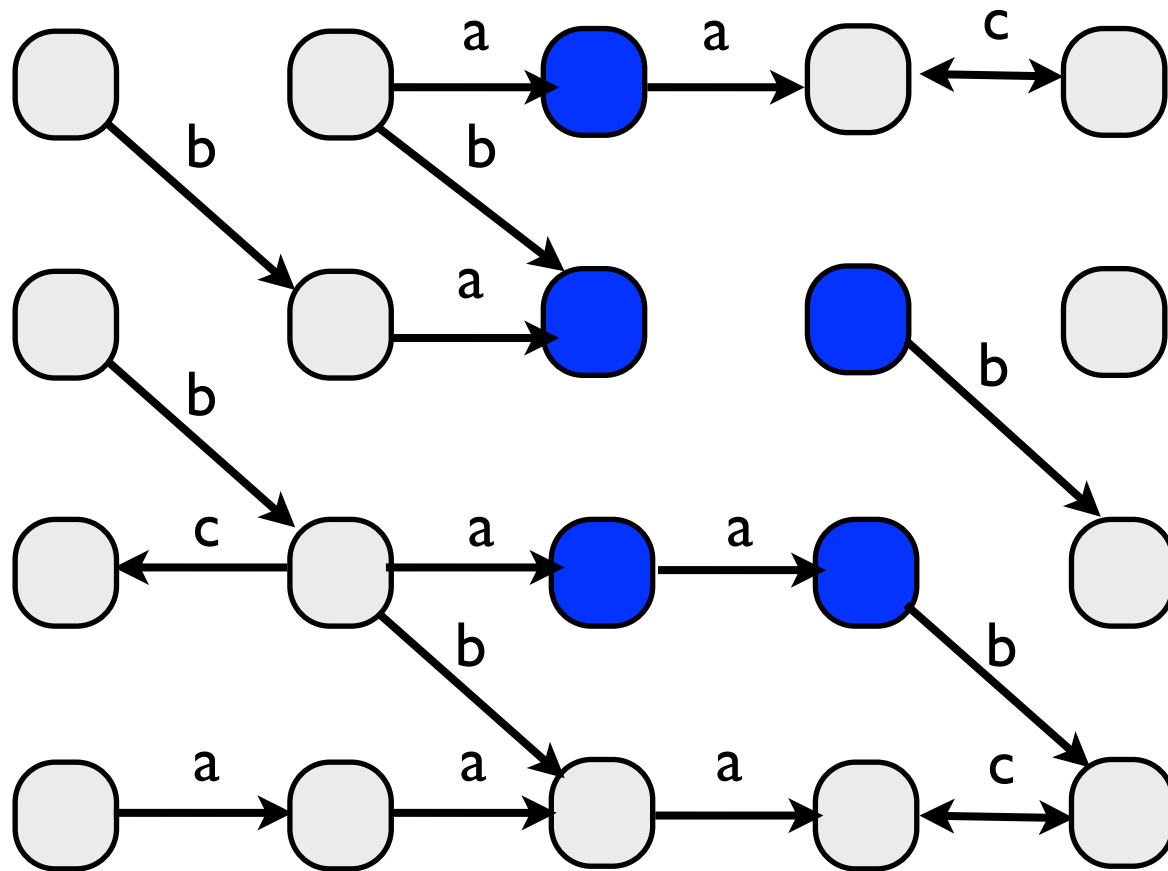
$$\mathbf{PRE}(X)=Y$$



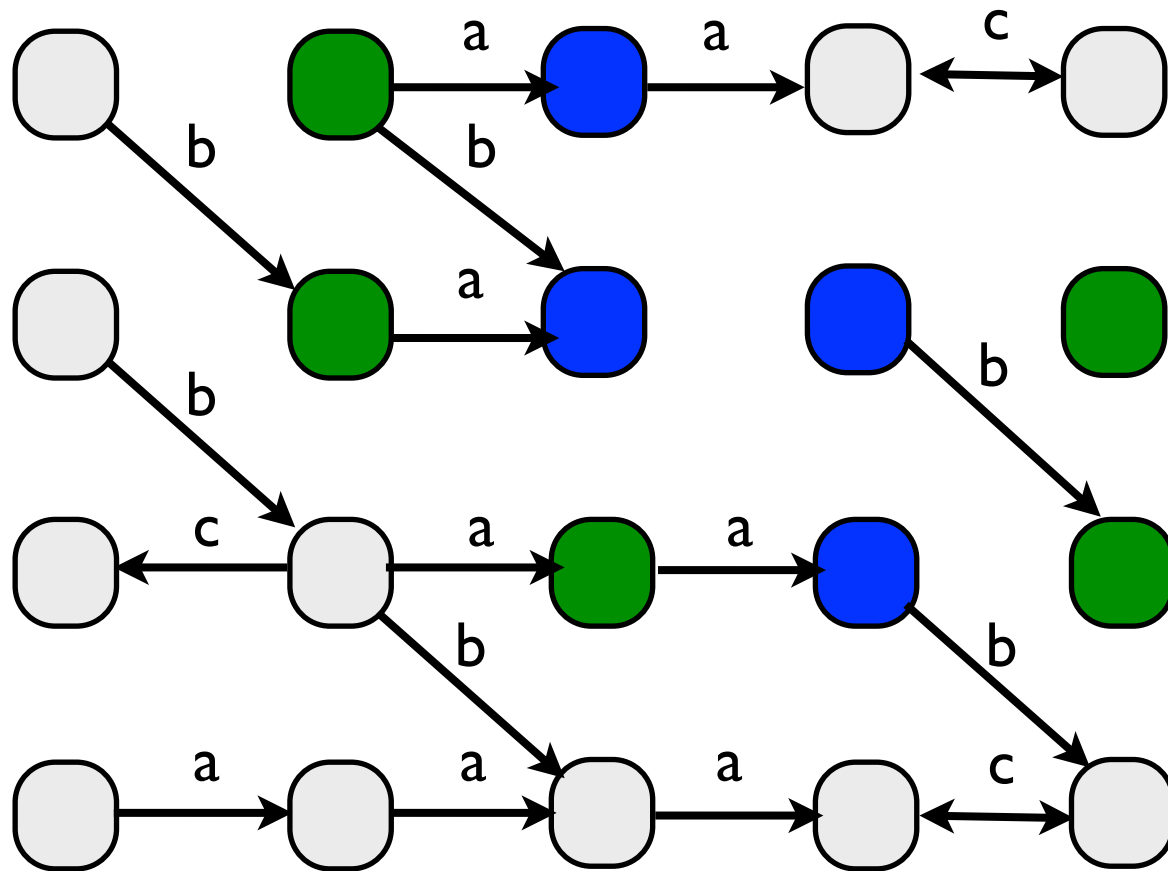
**APRE(X)=**



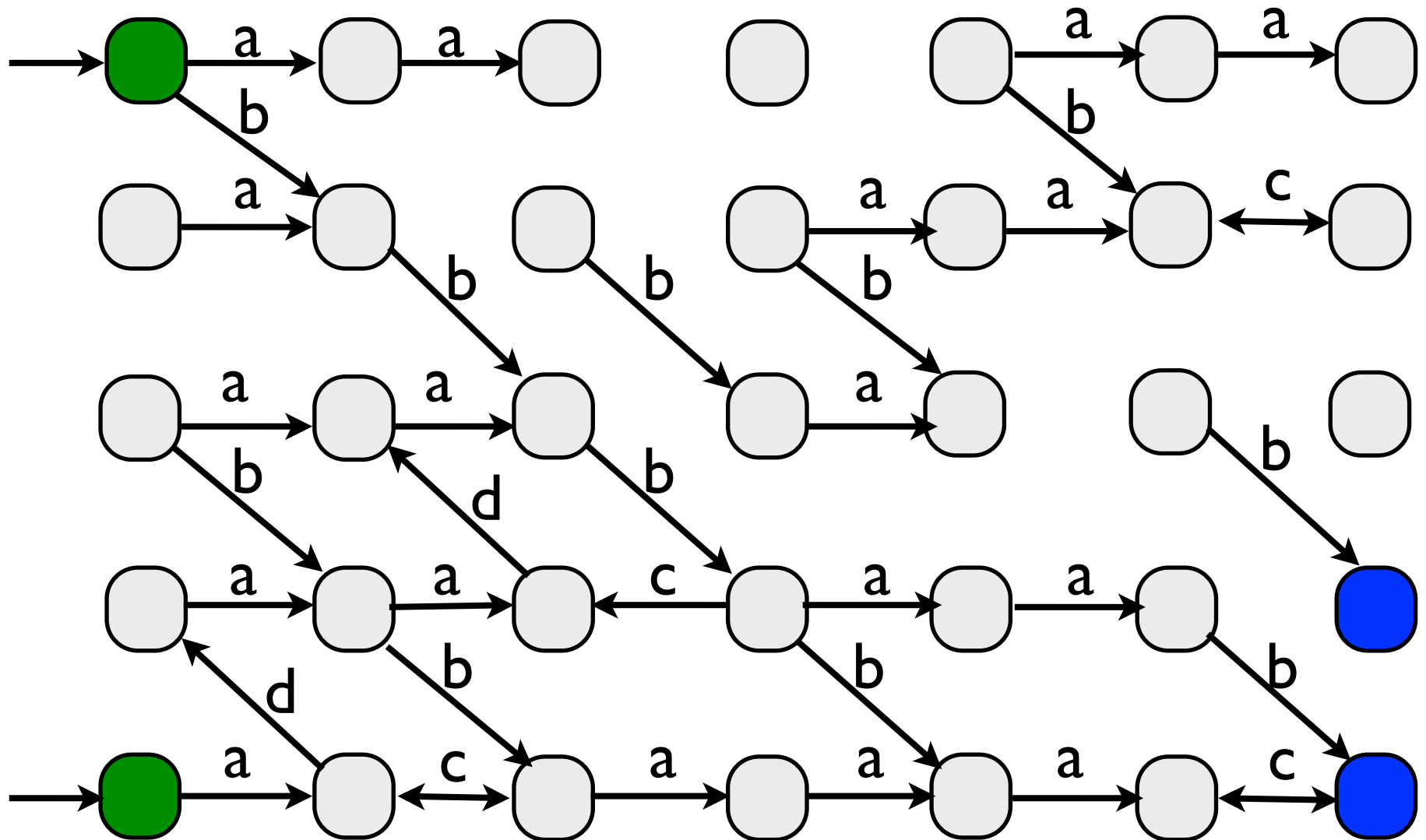
**APRE(X)=**



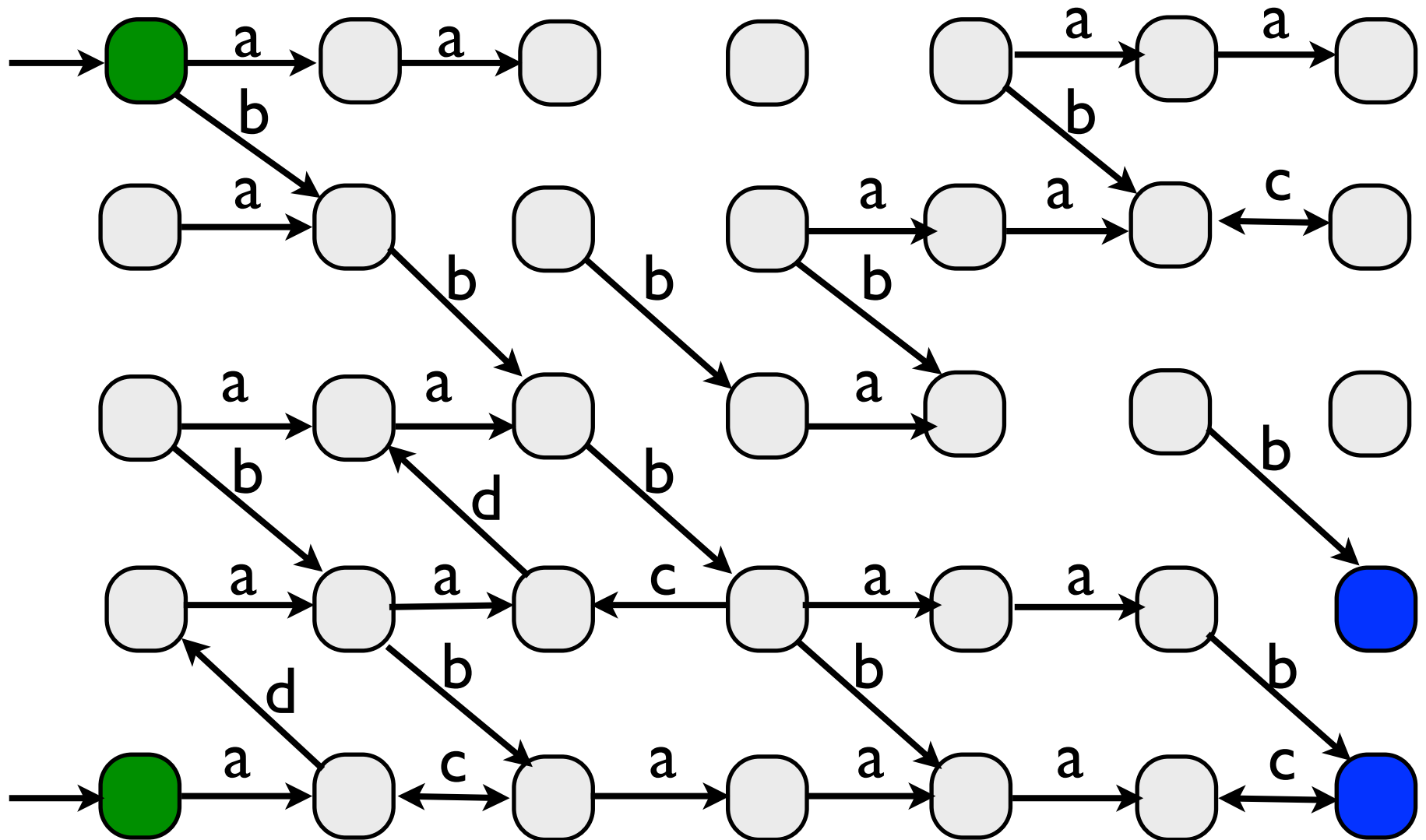
$$\mathbf{APRE}(\mathbf{X})=\mathbf{Y}$$



How can we use the **POST** operator to solve the following **reachability** question:  
Can we reach the **blue states** from **initial states** ?

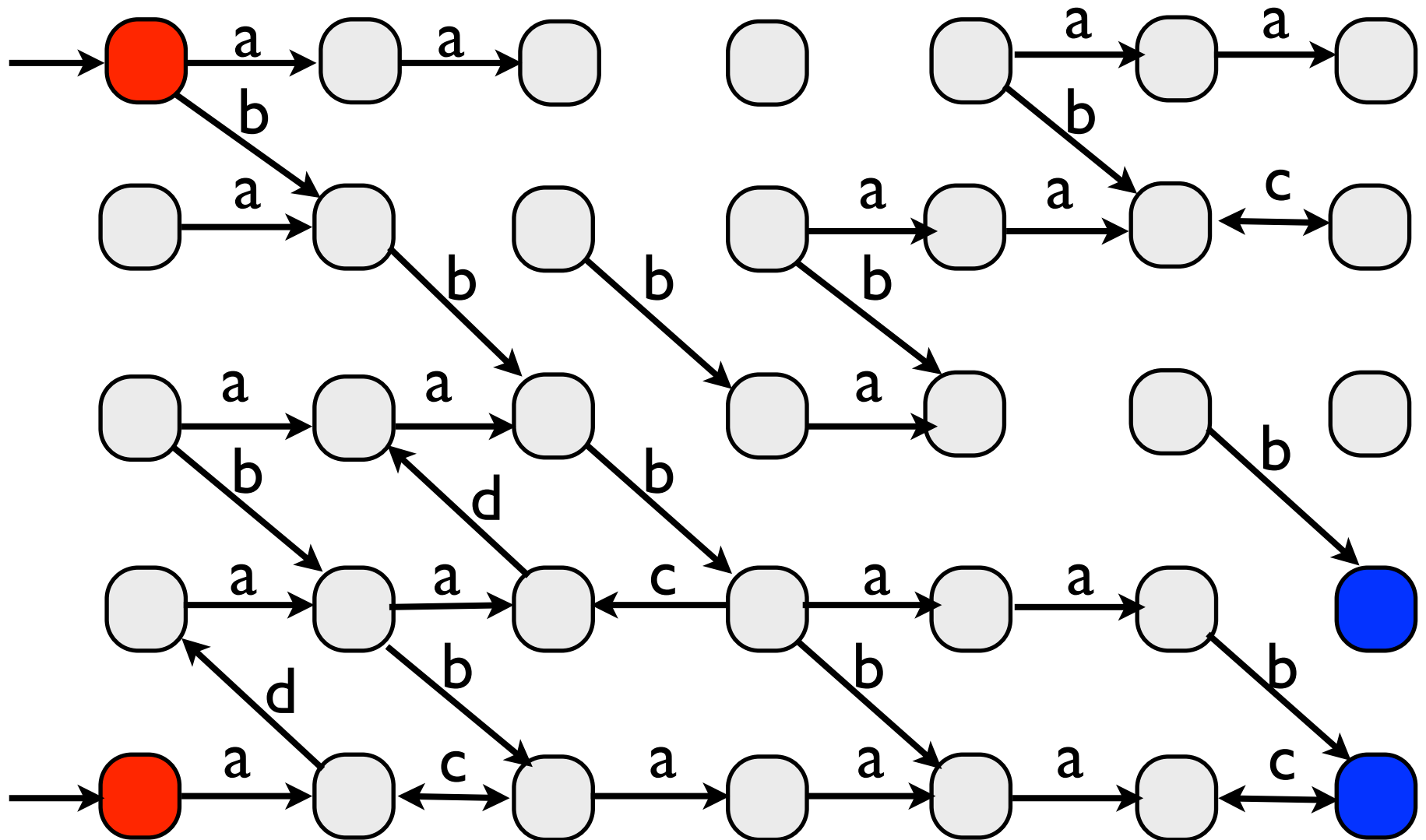


How can we use the **POST** operator to solve the following **reachability** question:  
Can we reach the **blue states** from **initial states** ?  
... By iterating the POST operator !



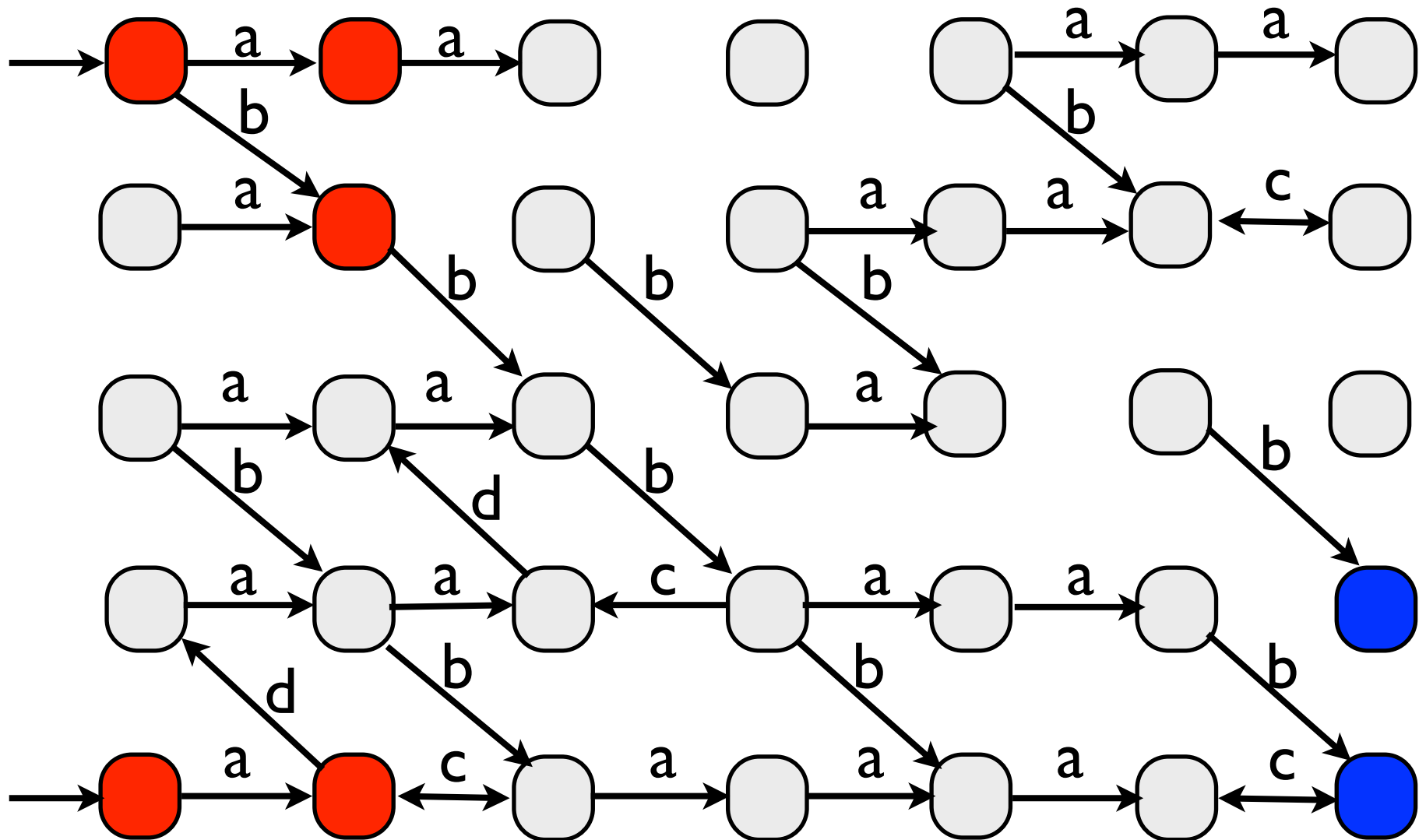
How can we use the **POST** operator to solve the following **reachability** question:  
Can we reach the **blue states** from **initial states** ?  
... By iterating the POST operator !

# 0 step



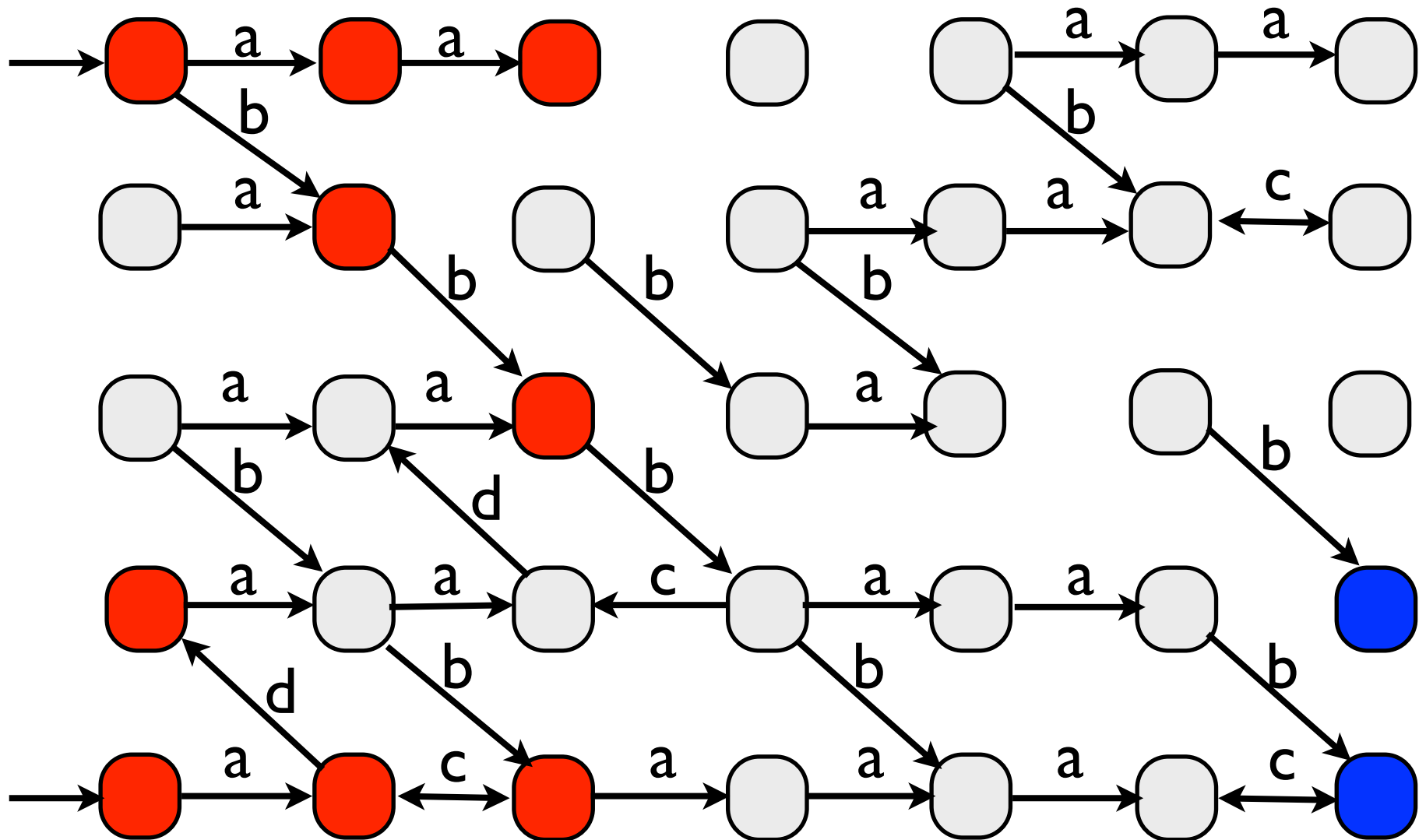
How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0 or 1 step



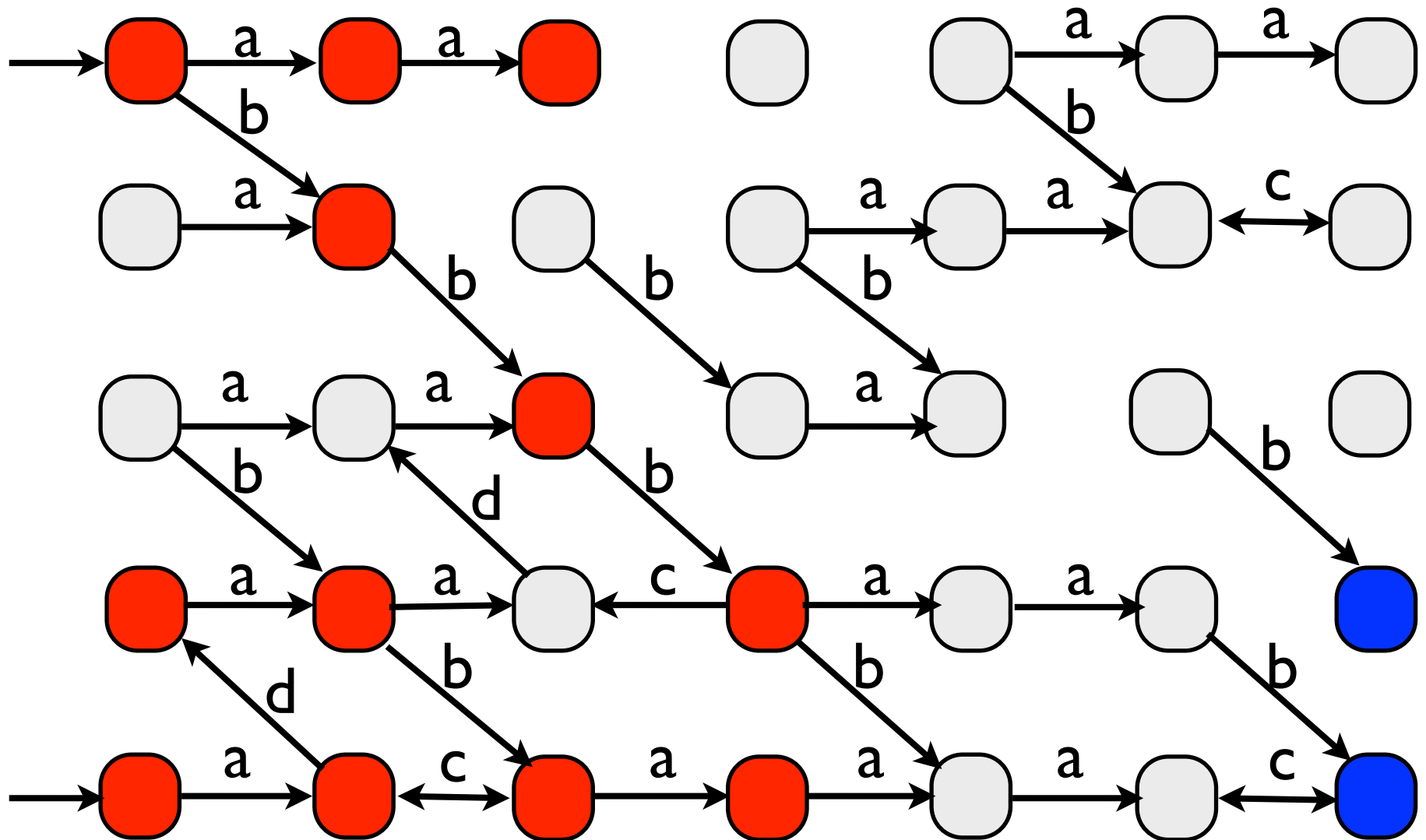
How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0, 1, 2 steps



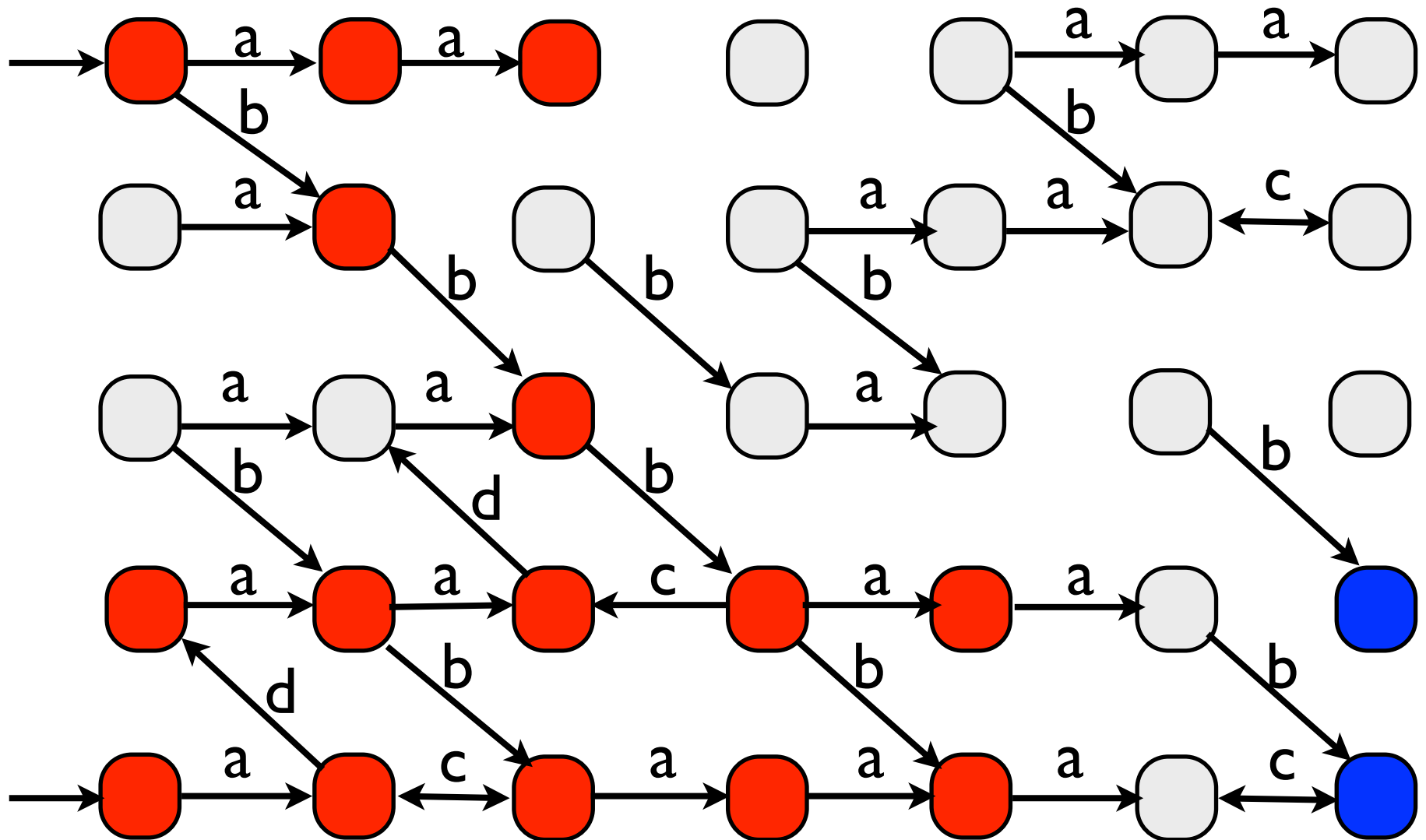
How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0, 1, 2, 3 steps



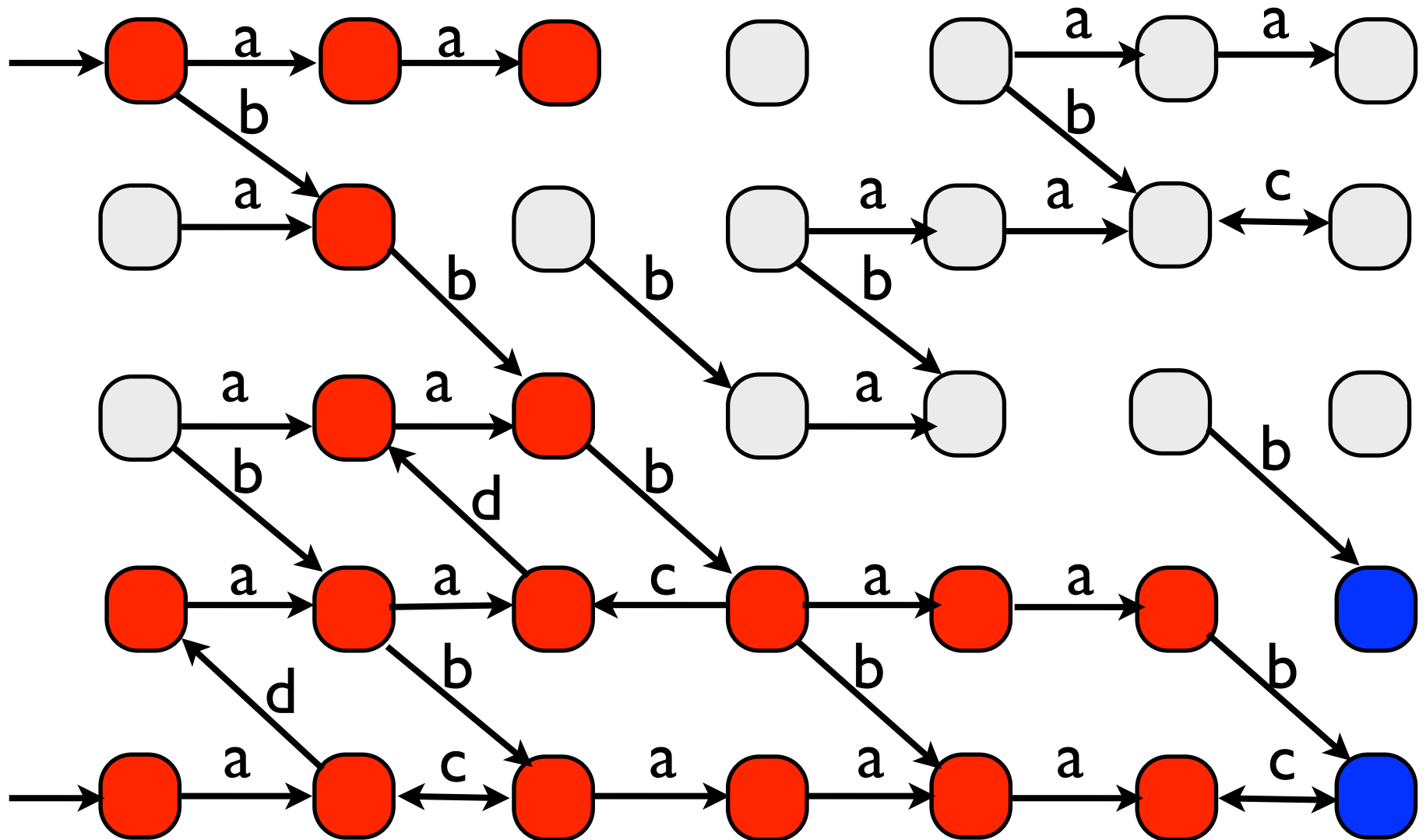
How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0, 1, 2, 3, 4 steps



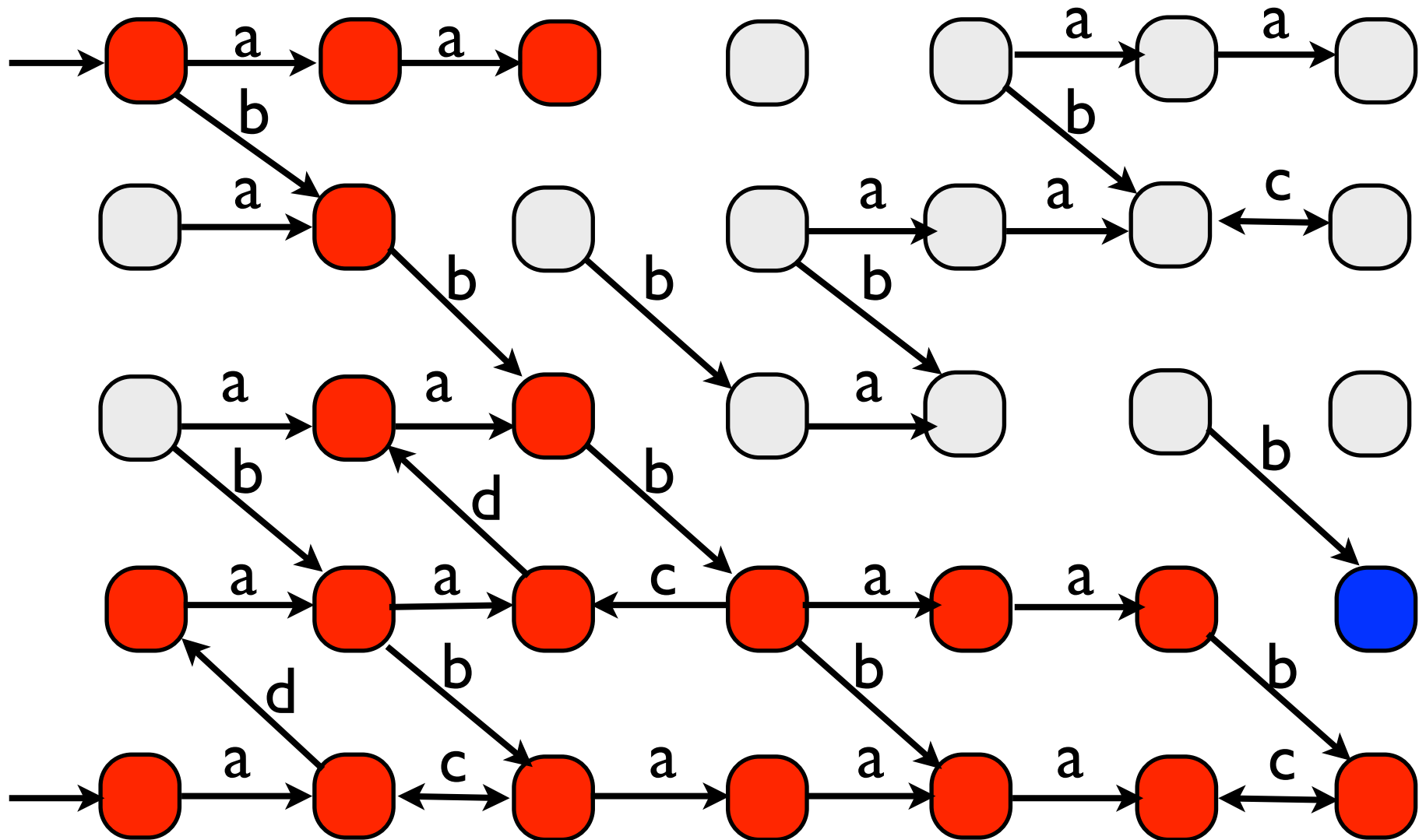
How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0, 1, 2, 3, 4, 5 steps



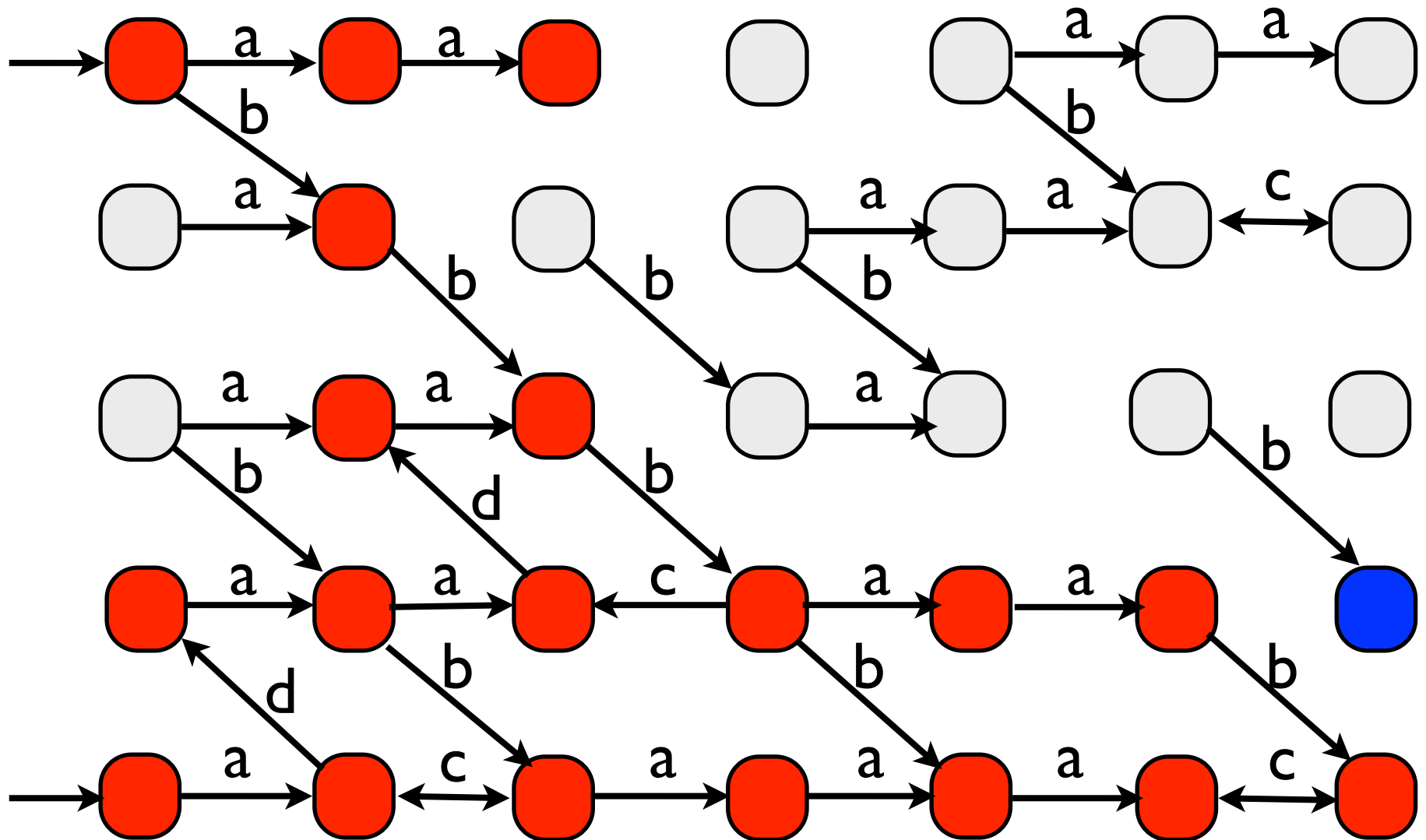
How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0, 1, 2, 3, 4, 5, 6 steps



How can we use the **POST** operator to solve the following **reachability** question:  
 Can we reach the **blue states** from **initial states** ?  
 ... By iterating the POST operator !

0, 1, 2, 3, 4, 5, 6,  $\infty$  steps - we have reached a **fixed point**



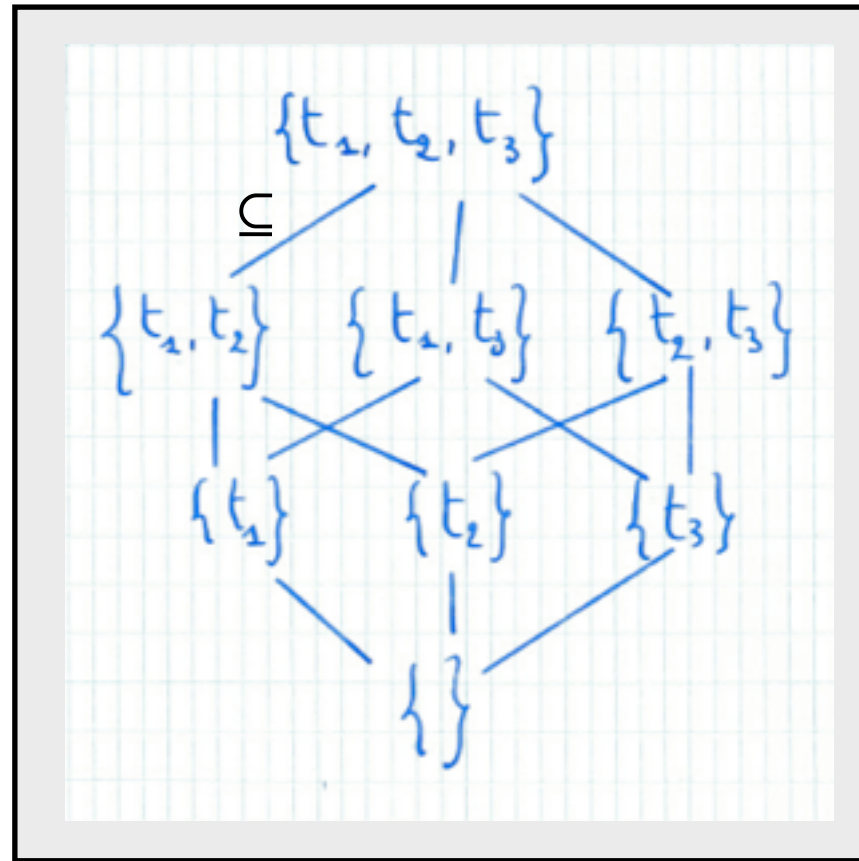
# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

# Partial orders

- Let  $\mathbf{S}$  be a set. A **partial order** over  $\mathbf{S}$  is a relation  $\leq \subseteq \mathbf{S} \times \mathbf{S}$  such that the following properties hold:
  - (i) **reflexivity**:  $\forall s \in \mathbf{S}: s \leq s$ ,
  - (ii) **transitivity**:  $\forall s_1, s_2, s_3 \in \mathbf{S}: s_1 \leq s_2 \wedge s_2 \leq s_3 \rightarrow s_1 \leq s_3$ ,
  - (iii) **antisymmetry**:  $\forall s_1, s_2 \in \mathbf{S}: s_1 \leq s_2 \wedge s_2 \leq s_1 \rightarrow s_1 = s_2$ .
- A pair  $(\mathbf{S}, \leq)$  such that  $\leq$  is a partial order over  $\mathbf{S}$  is called a **partially ordered set**.
- Let  $\mathbf{T}$  be a set, we note  $P(\mathbf{T})$  for the set of subsets of  $\mathbf{T}$ . Example: if  $\mathbf{T} = \{t_1, t_2, t_3\}$  then  $P(\mathbf{T}) = \{\{\}, \{t_1\}, \{t_2\}, \{t_3\}, \{t_1, t_2\}, \{t_2, t_3\}, \{t_1, t_3\}, \{t_1, t_2, t_3\}\}$ . Clearly, for any set  $\mathbf{T}$ ,  $(P(\mathbf{T}), \subseteq)$  is a **partially ordered set**.

# Graphical representation of $P(\mathbf{T})$



$(P(\mathbf{T}), \subseteq)$  is a partially ordered set:

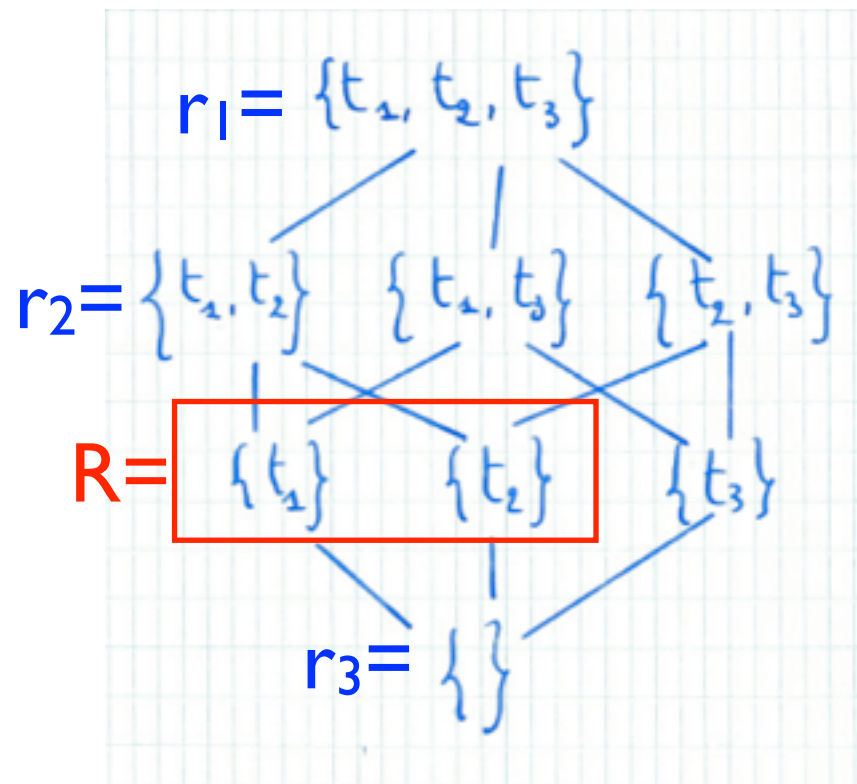
$\subseteq$  is reflexive:  $\{\} \subseteq \{\}, \{t_1\} \subseteq \{t_1\}, \dots, \{t_1, t_2\} \subseteq \{t_1, t_2\}, \dots$

$\subseteq$  is transitive:  $\{t_1\} \subseteq \{t_1, t_2\} \wedge \{t_1, t_2\} \subseteq \{t_1, t_2, t_3\} \rightarrow \{t_1\} \subseteq \{t_1, t_2, t_3\}$

and clearly,  $\subseteq$  is antisymmetric.

# Lower and upper bounds

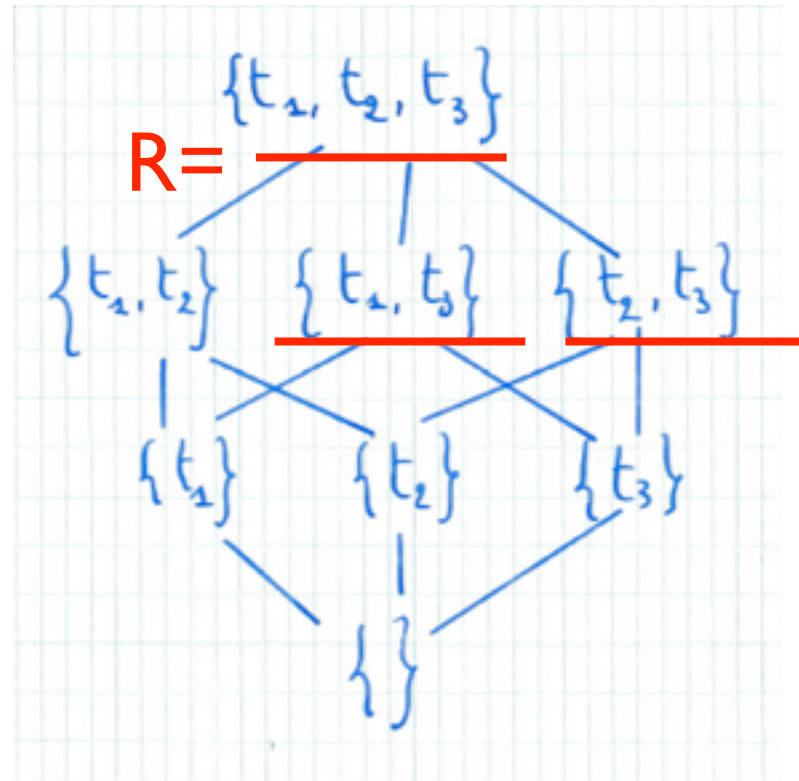
- Let  $(\mathbf{S}, \leq)$  be a partially ordered set. Let  $s \in \mathbf{S}$  and  $\mathbf{S}' \subseteq \mathbf{S}$ ,  
s is a **lower-bound** of  $\mathbf{S}'$  iff  $\forall s' \in \mathbf{S}' \cdot s \leq s'$ .  
s is a **upper-bound** of  $\mathbf{S}'$  iff  $\forall s' \in \mathbf{S}' \cdot s' \leq s$ .
- Let s be lower-bound for  $\mathbf{S}'$ , we say that s is the **greatest lower-bound (glb)** for  $\mathbf{S}'$  iff for all lower-bound  $s'$  for  $\mathbf{S}'$ , we have  $s' \leq s$ . We note **glb**( $\mathbf{S}'$ ) the **glb** of  $\mathbf{S}'$  if it exists.
- Let s be upper-bound for  $\mathbf{S}'$ , we say that s is the **least upper-bound (lub)** for  $\mathbf{S}'$  iff for all upper-bound  $s'$  for  $\mathbf{S}'$ , we have  $s \leq s'$ . We note **lub**( $\mathbf{S}'$ ) the **lub** of  $\mathbf{S}'$  if it exists.



⇒  $r_1$  and  $r_2$  are **upper bounds** of  $R$ .

⇒  $r_2$  is the **least upper bound** of  $R$ .

⇒  $r_3$  is the only **lower-bound** of  $R$  and so it is the **greatest lower bound** of  $R$ .



The **lub** of a set of sets  $R_i$  is equal to  $\cup_i R_i$

ex: **lub**  $R = \{t_1, t_2, t_3\}$

The **glb** of a set of sets  $R_i$  is equal to  $\cap_i R_i$

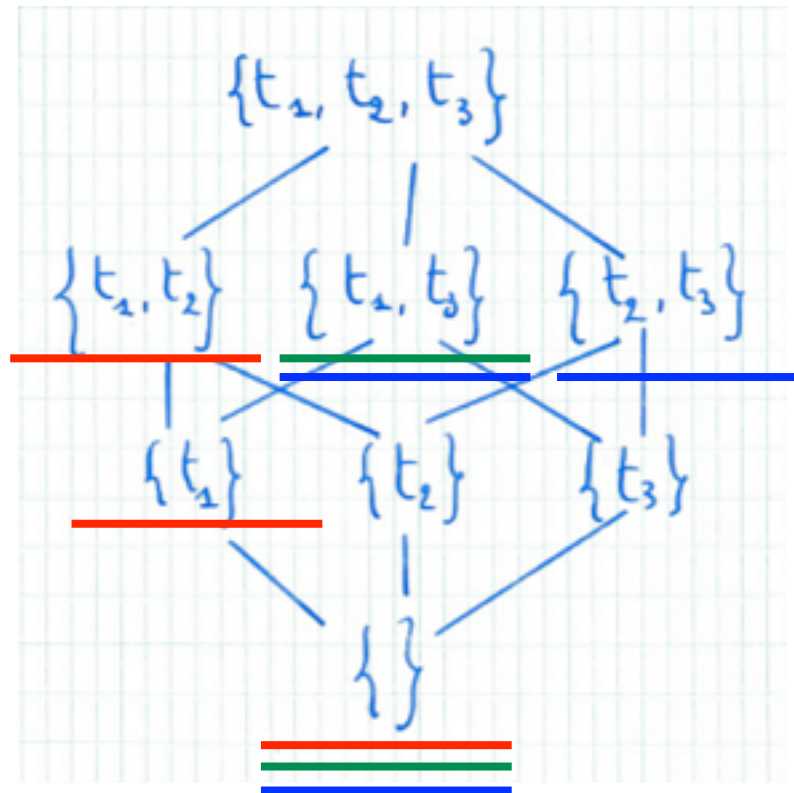
ex: **glb**  $R = \{t_3\}$

A set of elements  $\mathbf{S}' \subseteq \mathbf{S}$  is a **chain**

iff

$$\forall s, s' \in \mathbf{S}' \cdot s \leq s' \vee s' \leq s$$

i.e. all pairs of elements are ordered by  $\leq$   
(increasing sequence of elements).

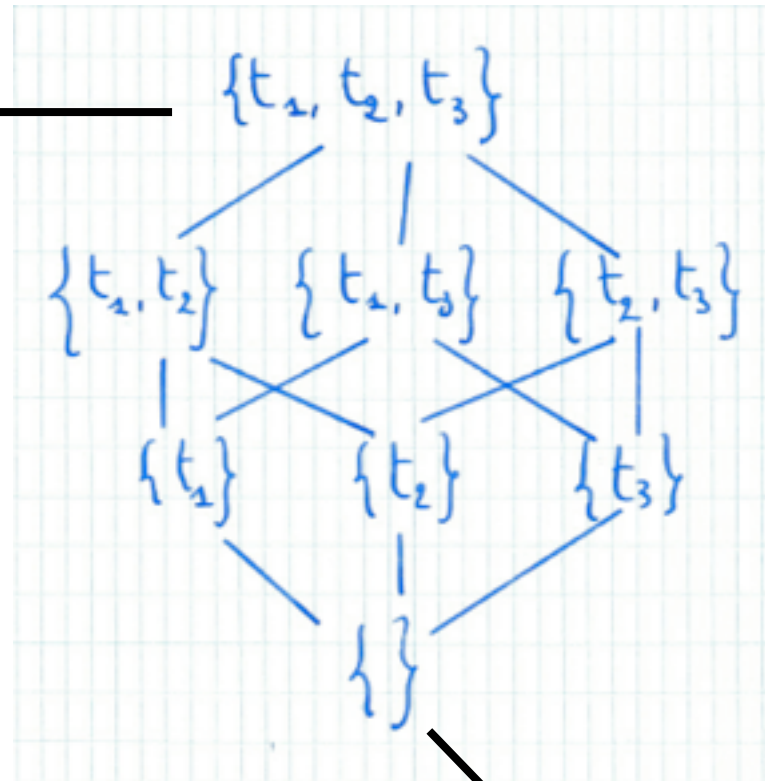


$R = \{\{\}, \{t_1\}, \{t_1, t_2\}\}$  is a chain in  $P(\mathbf{T})$ .

$R' = \{\{\}, \{t_1, t_3\}\}$  is a chain in  $P(\mathbf{T})$ .

$R'' = \{\{\}, \{t_1, t_3\}, \{t_2, t_3\}\}$  is not a chain in  $P(\mathbf{T})$ .

- A partially ordered set  $(\mathbf{S}, \leq)$  is a **complete partial order** if every chain in  $\mathbf{S}$  has a **lub** in  $\mathbf{S}$ .
- A complete partial order  $(\mathbf{S}, \leq)$  is a **complete lattice** if every subset  $\mathbf{S}'$  of  $\mathbf{S}$  has a **lub** in  $(\mathbf{S}, \leq)$ .
- Note that **glb**  $\mathbf{S}' = \text{lub}\{s \in \mathbf{S} \mid \forall s' \in \mathbf{S}': s \leq s'\}$ , so every subset  $\mathbf{S}'$  in a complete lattice has also a **glb**.
- Example:  $(\mathbf{P}(\mathbf{T}), \subseteq)$  is a complete lattice. Indeed, remember that the **lub** of a set of sets  $R_i$  is equal to  $\cup_i R_i$  and the **glb** of a set of sets  $R_i$  is equal to  $\cap_i R_i$ .



The maximal element of  $P(\mathbf{T})$ ,  
i.e. the least upper-bound of  $P(\mathbf{T})$ ,  
noted **Max**( $P(\mathbf{T})$ ).

The minimal element of  $P(\mathbf{T})$ ,  
i.e. the greatest lower-bound of  $P(\mathbf{T})$ ,  
noted **Min**( $P(\mathbf{T})$ ).

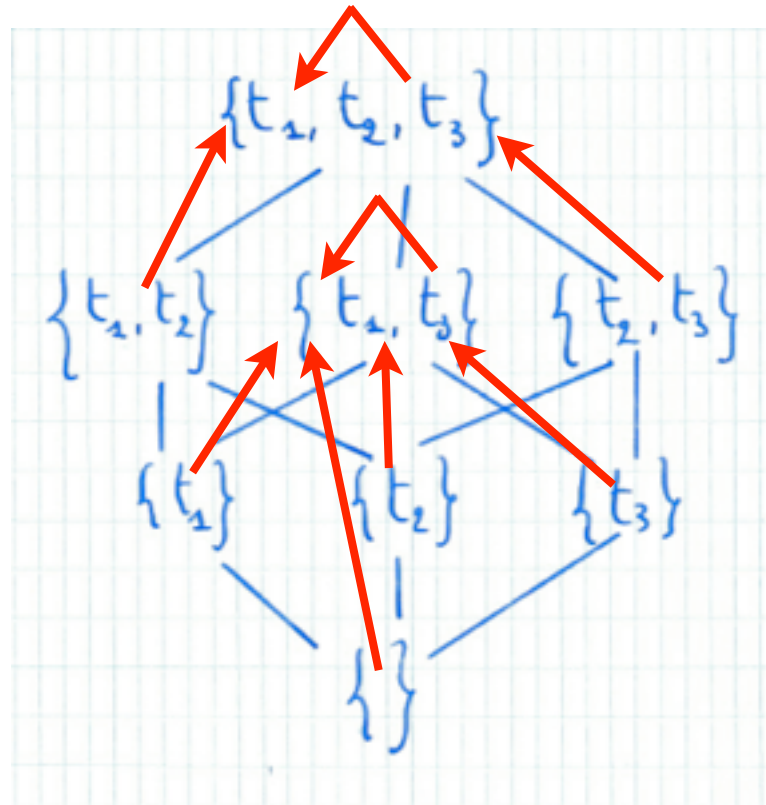
**Min** and **Max** elements in a complete lattice

- Let  $(\mathbf{S}, \leq)$  be a partially ordered set. A function  $\mathbf{f}: \mathbf{S} \rightarrow \mathbf{S}$  is **monotone** (order preserving) iff  $\forall s, s' \in \mathbf{S} \cdot s \leq s' \rightarrow \mathbf{f}(s) \leq \mathbf{f}(s')$ .
- Let  $(\mathbf{S}, \leq)$  be a complete partially ordered set. A function  $\mathbf{f}: \mathbf{S} \rightarrow \mathbf{S}$  is **continuous** iff  $\mathbf{f}$  is monotone and for all non-empty chain  $\mathbf{S}'$  in  $\mathbf{S}$ :  **$\mathbf{f}(\text{lub}(\mathbf{S}')) = \text{lub}(\mathbf{f}(\mathbf{S}'))$** .

*Remark.* In any finite complete partially ordered set  $\mathbf{S}$ , if  $\mathbf{f}$  is monotone then  $\mathbf{f}$  is continuous.

- $s \in \mathbf{S}$  is a **fixed point** of  $\mathbf{f}: \mathbf{S} \rightarrow \mathbf{S}$  if  $\mathbf{f}(s) = s$ . The set of fixed points of  $\mathbf{f}$  is noted  $\mathbf{Fx}(\mathbf{f})$ .
- **Theorem** (Tarski) Any monotone function on a **complete lattice** has a:
  - (i) **least fixed point, lfp(f)**, equal to **glb(Fx(f))**
  - (ii) **greatest fixed point, gfp(f)**, equal to **lub(Fx(f))**

Let us consider **f** as depicted by red arrows

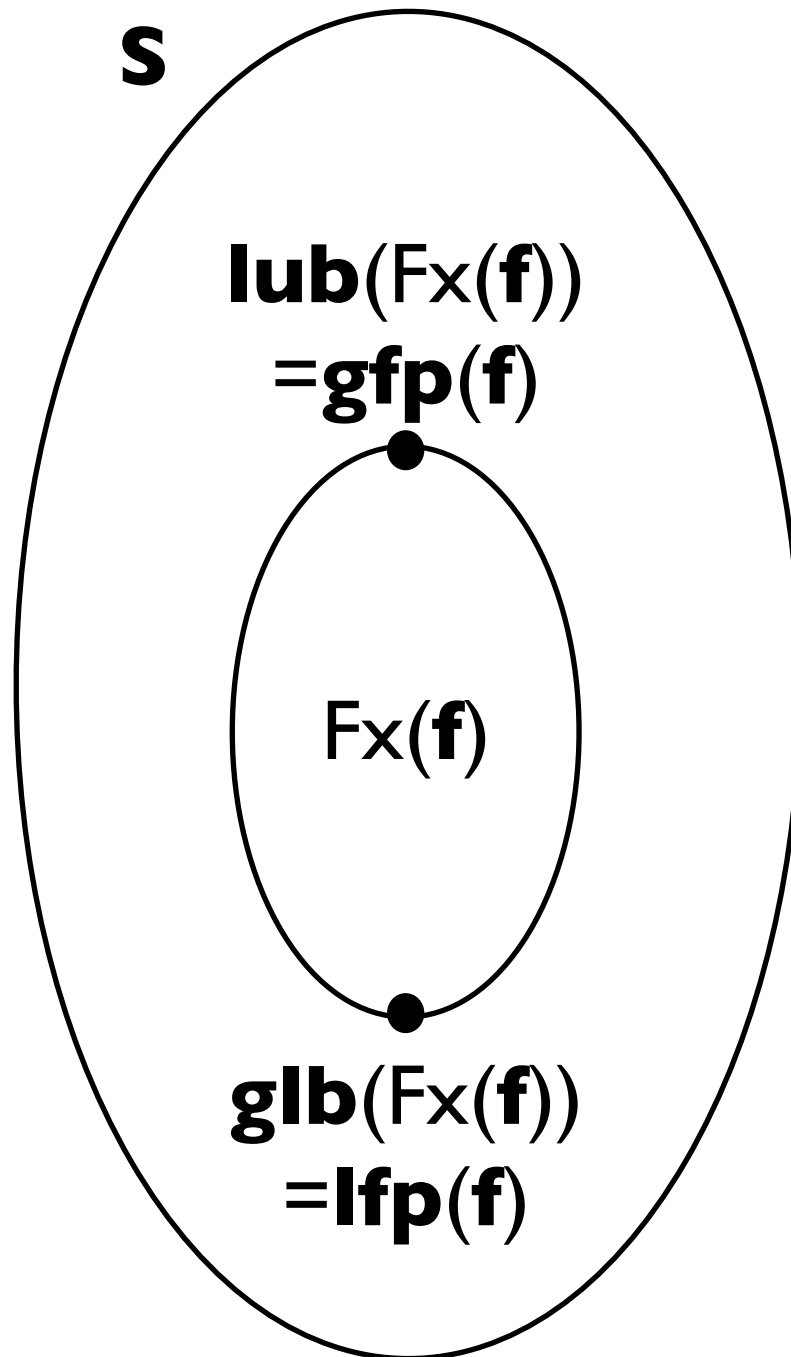


Clearly, **f** is monotone (and so continuous).

$$\mathbf{Fx}(\mathbf{f}) = \{\{t_1, t_3\}, \{t_1, t_2, t_3\}\}.$$

$$\mathbf{lfp}(\mathbf{f}) = \{t_1, t_3\} = \mathbf{glb}(\mathbf{Fx}(\mathbf{f})).$$

$$\mathbf{gfp}(\mathbf{f}) = \{t_1, t_2, t_3\} = \mathbf{lub}(\mathbf{Fx}(\mathbf{f})).$$



- Let  $\mathbf{f}^i$  be defined **inductively** as
  - for  $i=0$ :  $\mathbf{f}^0 = \mathbf{f}$
  - for all  $i > 0$ :  $\mathbf{f}^i = \mathbf{f}(\mathbf{f}^{i-1})$ .

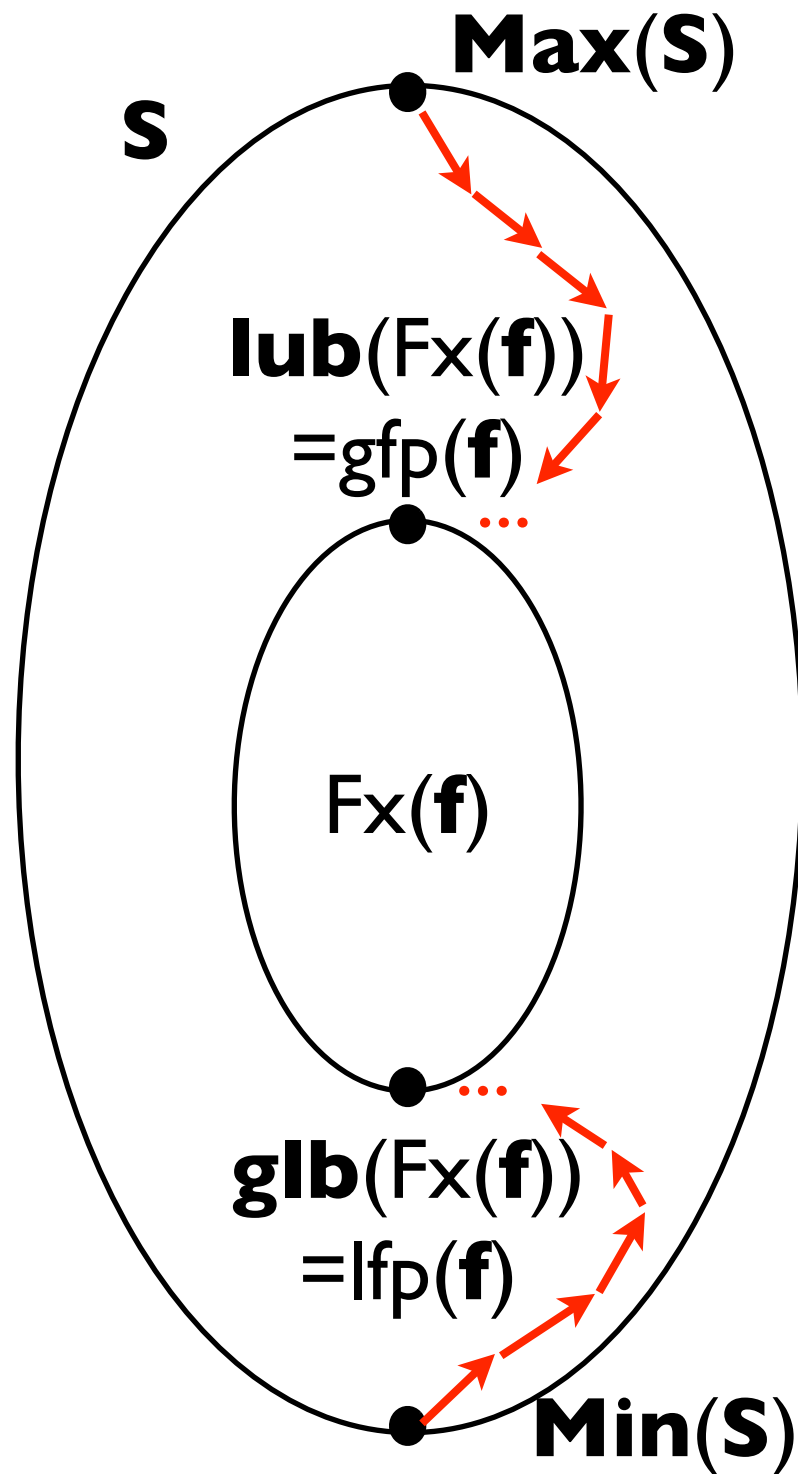
- **Theorem** (Kleene-Tarski) Let  $(\mathbf{S}, \leq)$  be a complete lattice, let  $\mathbf{f}: \mathbf{S} \rightarrow \mathbf{S}$  be a continuous:

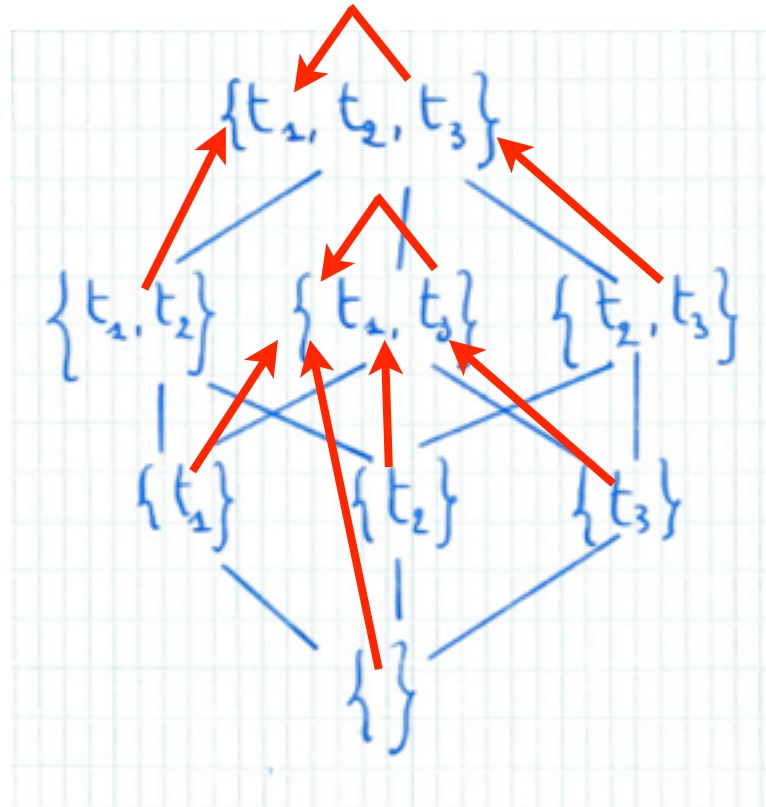
$$\mathbf{lfp}(\mathbf{f}) = \mathbf{glb} \{ \mathbf{f}^i(\mathbf{Min}(\mathbf{S})) \mid i \geq 0 \} \text{ and}$$

$$\mathbf{gfp}(\mathbf{f}) = \mathbf{lub} \{ \mathbf{f}^i(\mathbf{Max}(\mathbf{S})) \mid i \geq 0 \}.$$

- This gives us an **iterative schema** to compute the **lfp(f)** (**gfp(f)**) of a continuous function  $\mathbf{f}$ :

$\Rightarrow$  iterate the function from the **Min** (**Max**) of the set until stabilization.

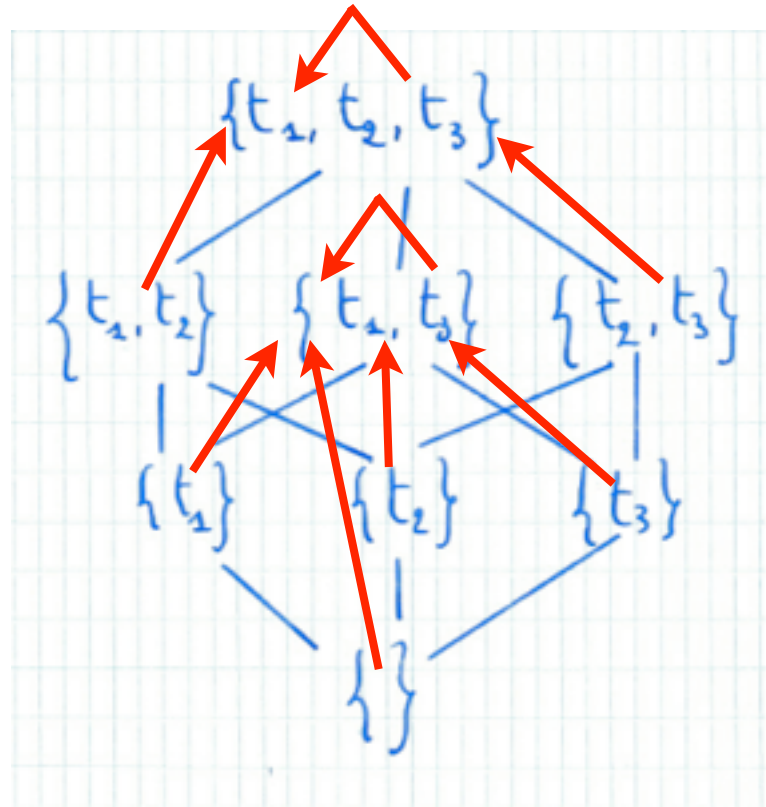




Computation of **lfp(f)**

$$\mathbf{f}^0(\{\}) = \{t_1, t_2\}$$

$$\mathbf{f}^1(\{\}) = \mathbf{f}(\{t_1, t_2\}) = \{t_1, t_2\} = \mathbf{lfp}(\mathbf{f})$$



Computation of **gfp(f)**

$$\mathbf{R}_0 = \mathbf{f}^0(\{t_1, t_2, t_3\}) = \{t_1, t_2, t_3\} = \mathbf{gfp}(\mathbf{f})$$

# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

# Symbolic model-checking

- The reachability, safety, and Büchi objectives can be solved using **fixed point equations**.
- Solving those equations will be done by iteration of functions built from the **Pre**, **Apri** or **Post** operators on sets of states.
- Those algorithms are called **symbolic** because they **manipulate sets** of states directly instead of manipulating **individual states** as it is done in so-called **explicit** model-checking algorithms.

# Fixed points for reachability

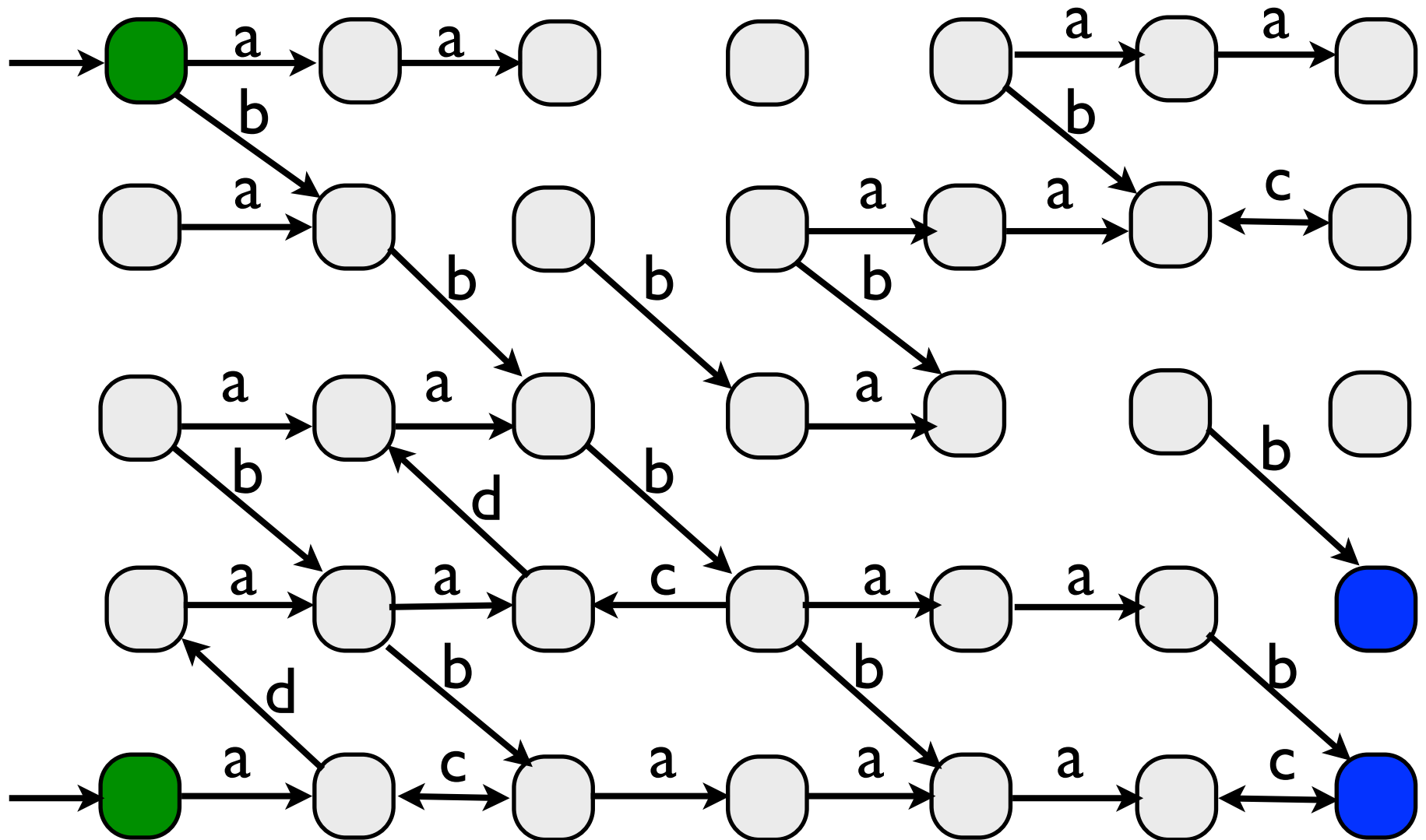
- Let us consider an instance of the **reachability problem** given by the LTS  $L=(S,S_0,\Sigma,T,C,\lambda)$ , and a set of states **Goal**  $\subseteq S$ ;
- **Goal** is reachable in the LTS

iff **lfp**  $(\lambda X. S_0 \cup \mathbf{POST}(X)) \cap \mathbf{Goal} \neq \emptyset$   
this is a **forward algorithm**

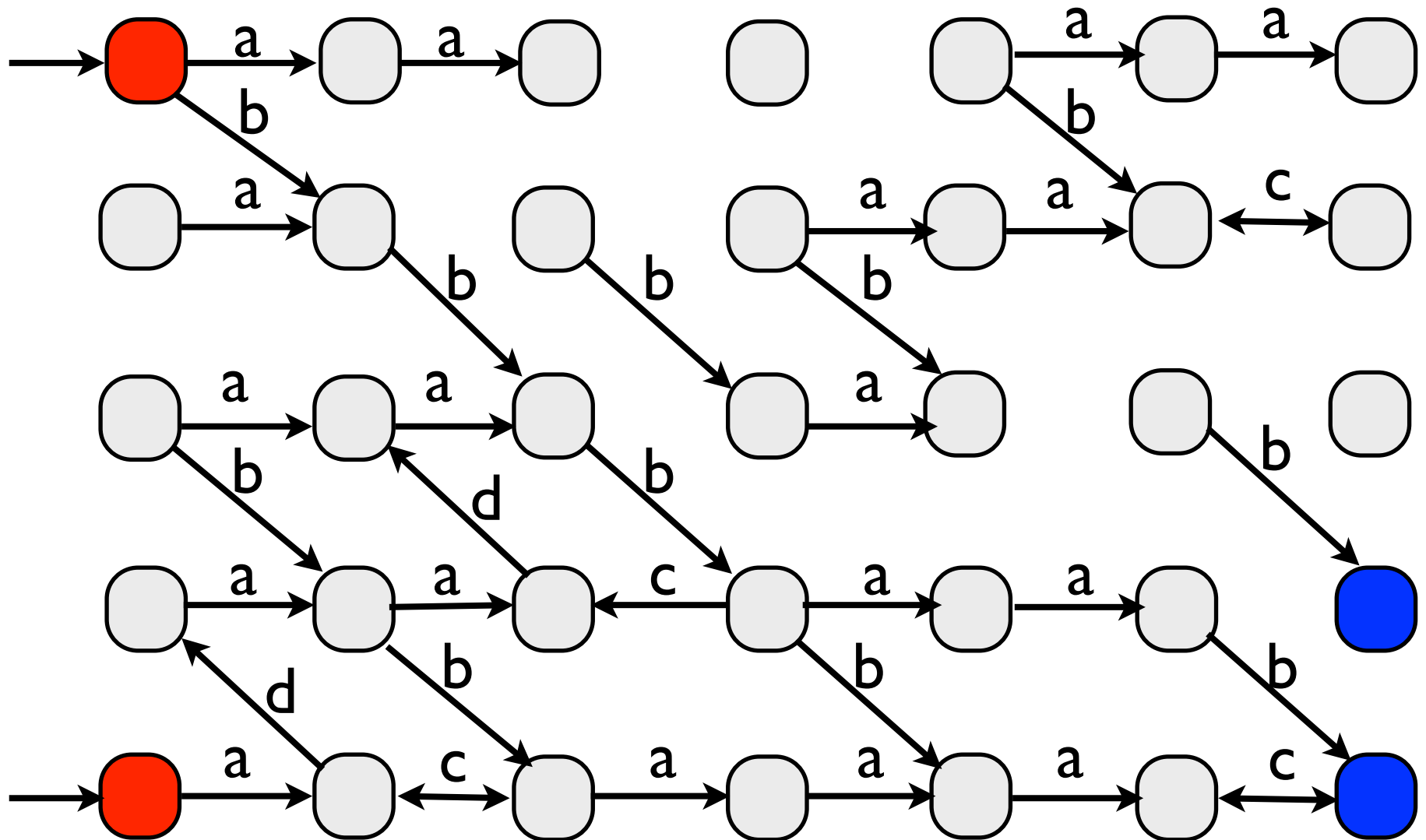
iff **lfp**  $(\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap S_0 \neq \emptyset$   
this is a **backward algorithm**

# Reachability - Forward algorithm

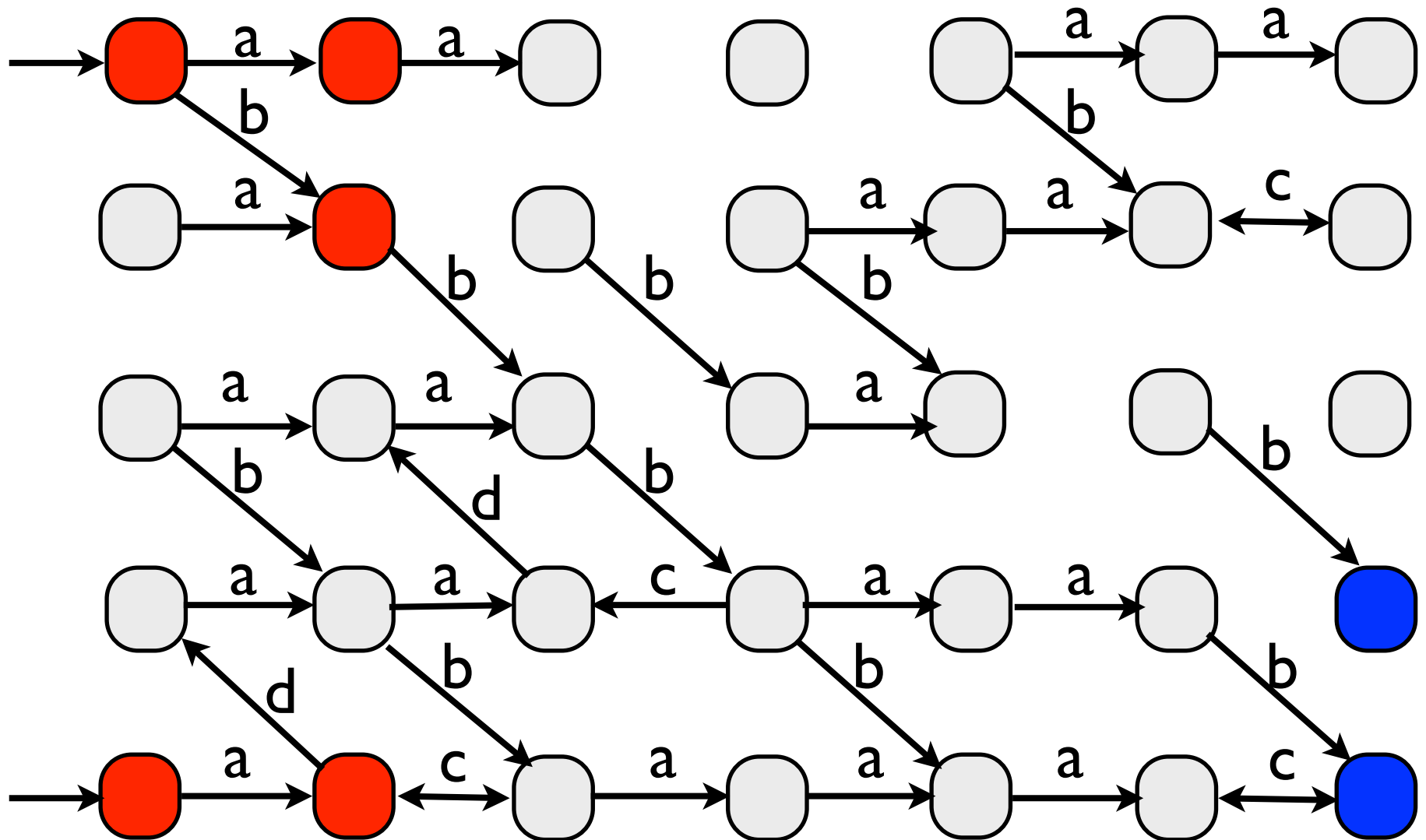
$$\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X)) \cap \mathbf{Goal} \neq \emptyset$$



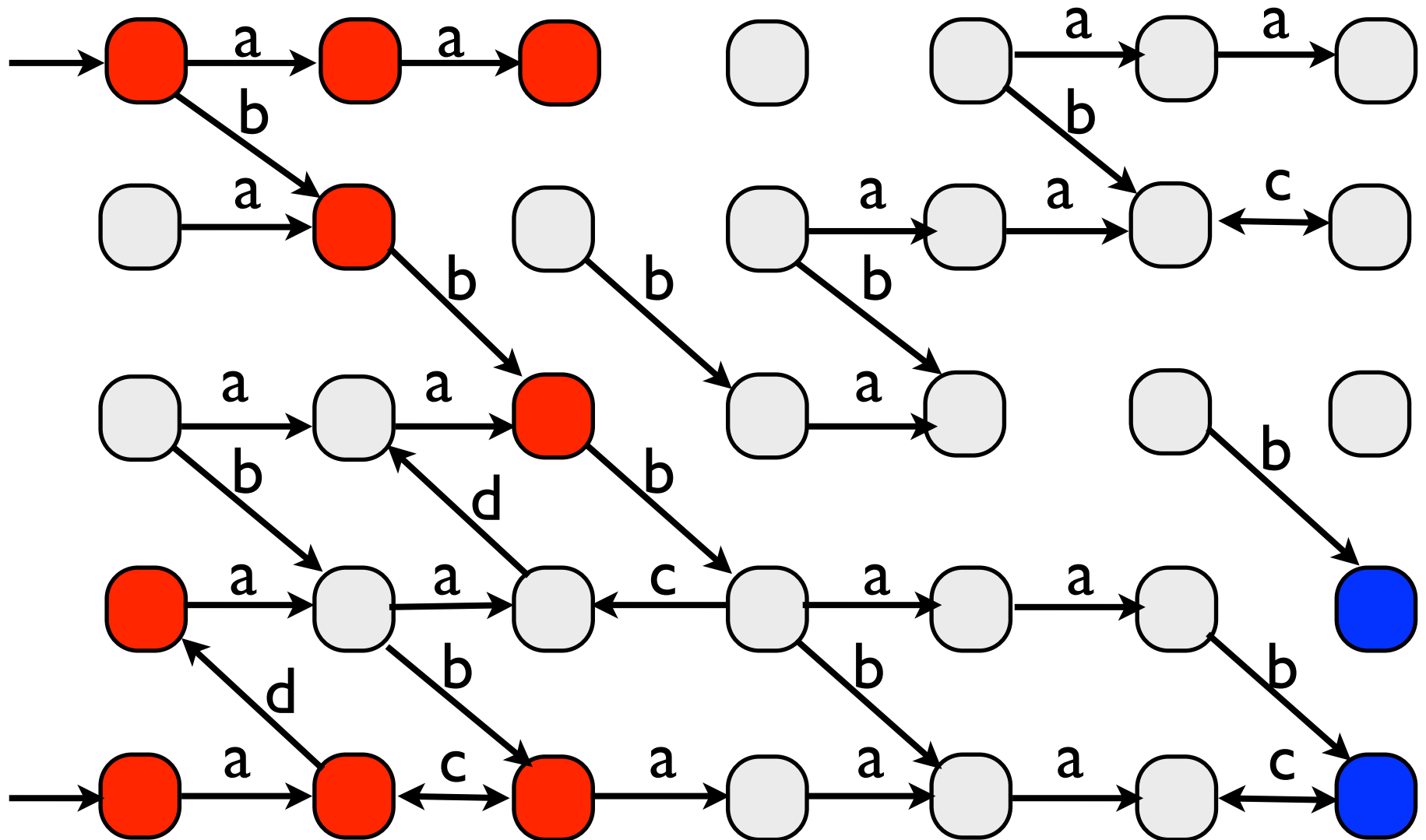
# Iterative evaluation of $\mathbf{lfp} (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$



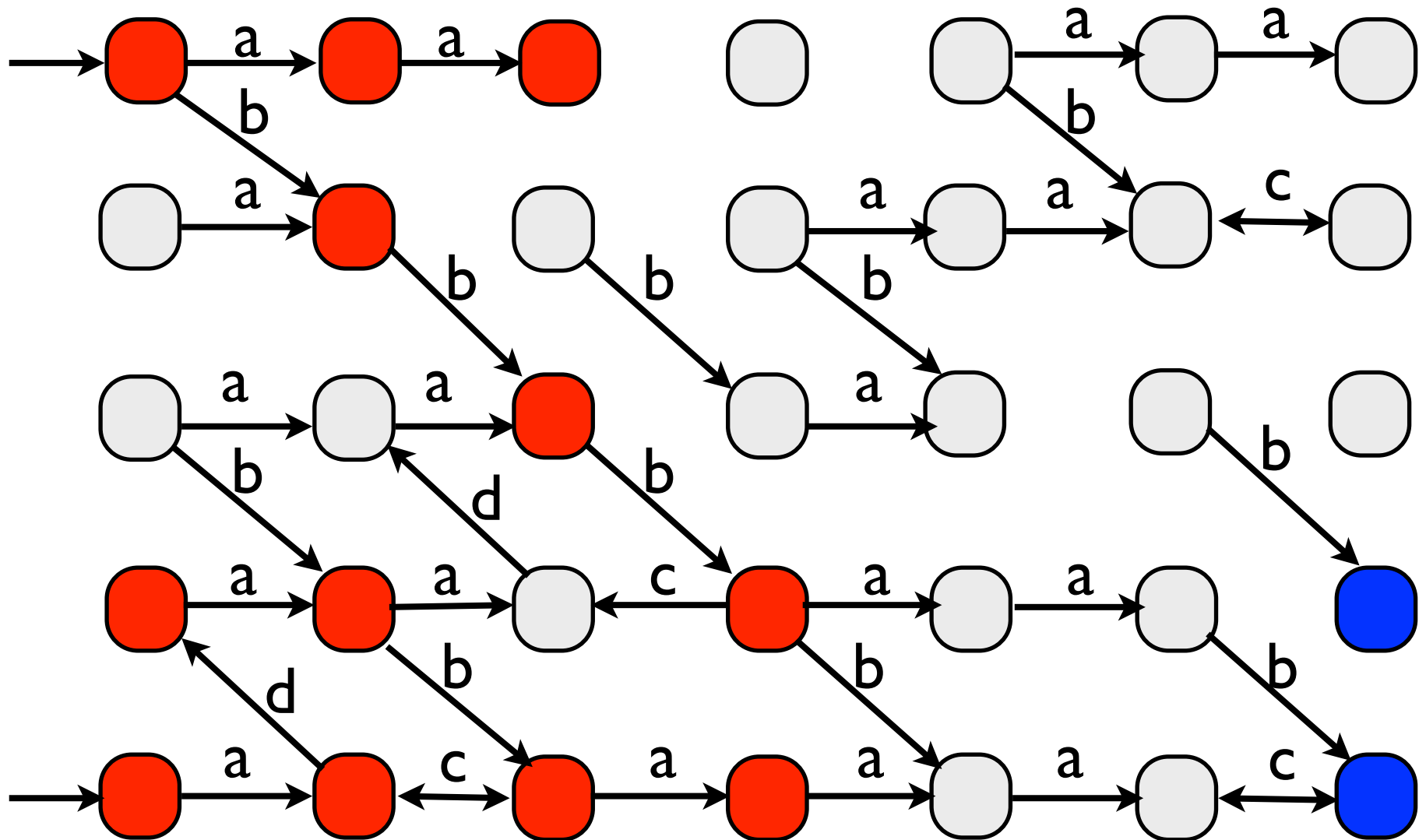
# Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$



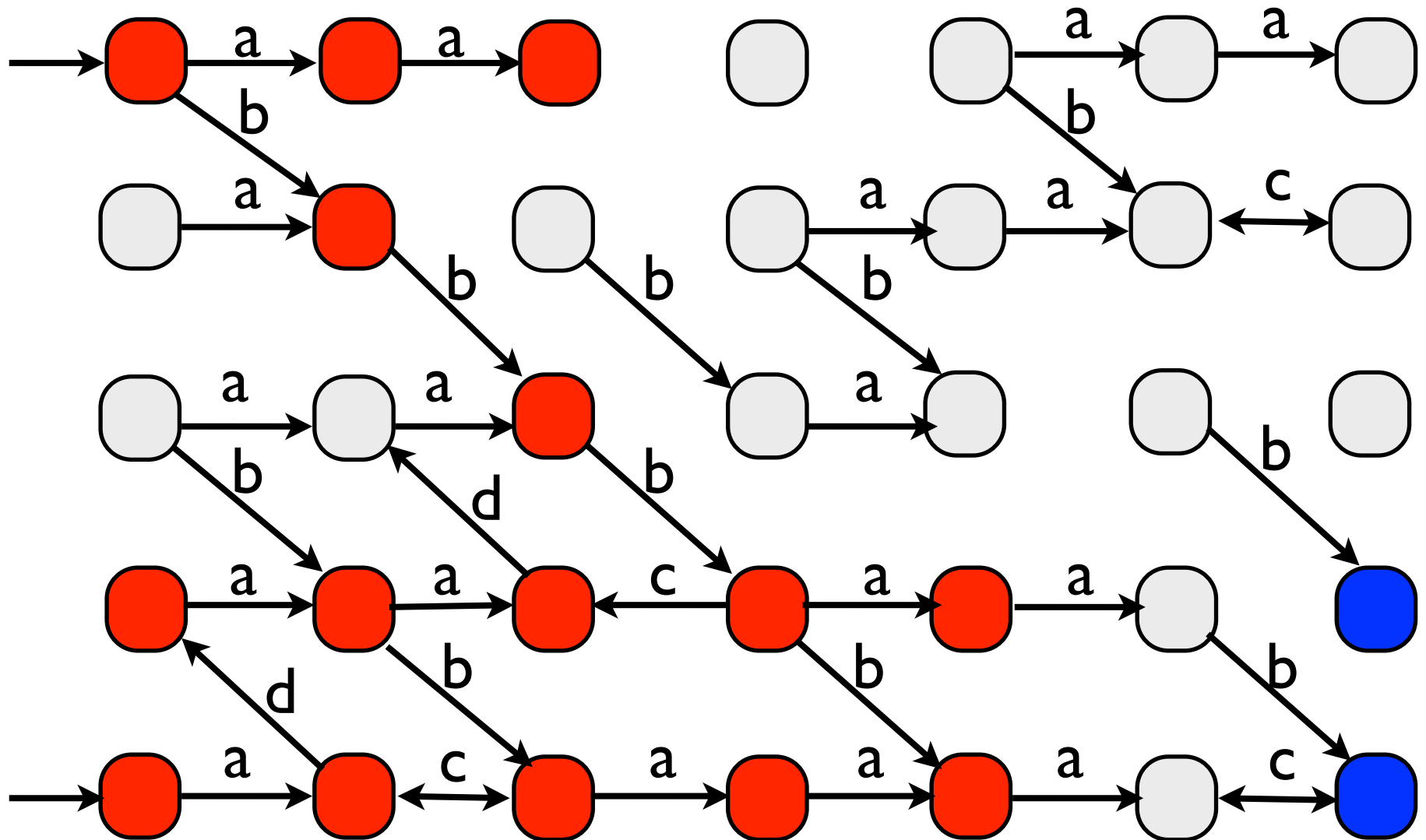
# Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$



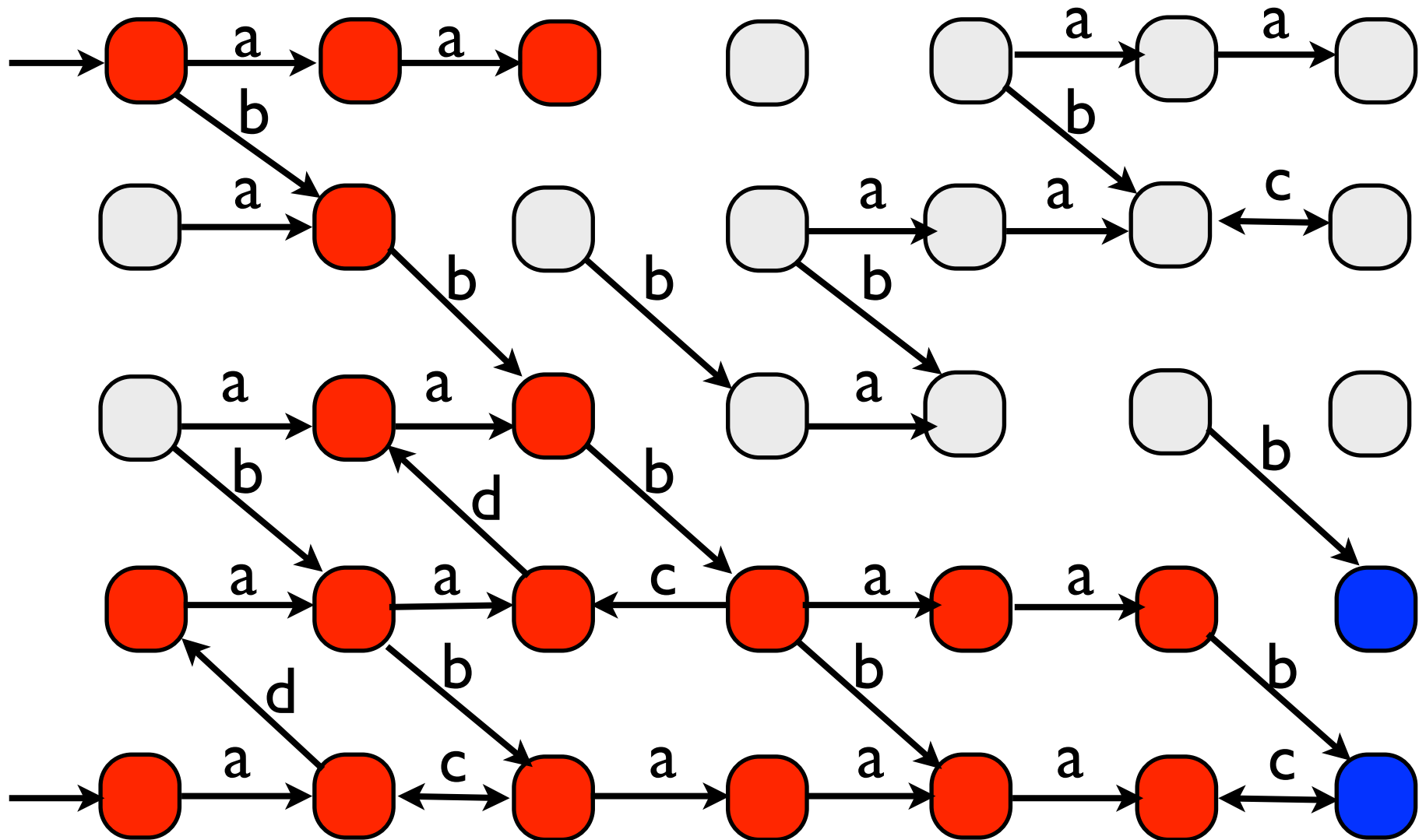
# Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$



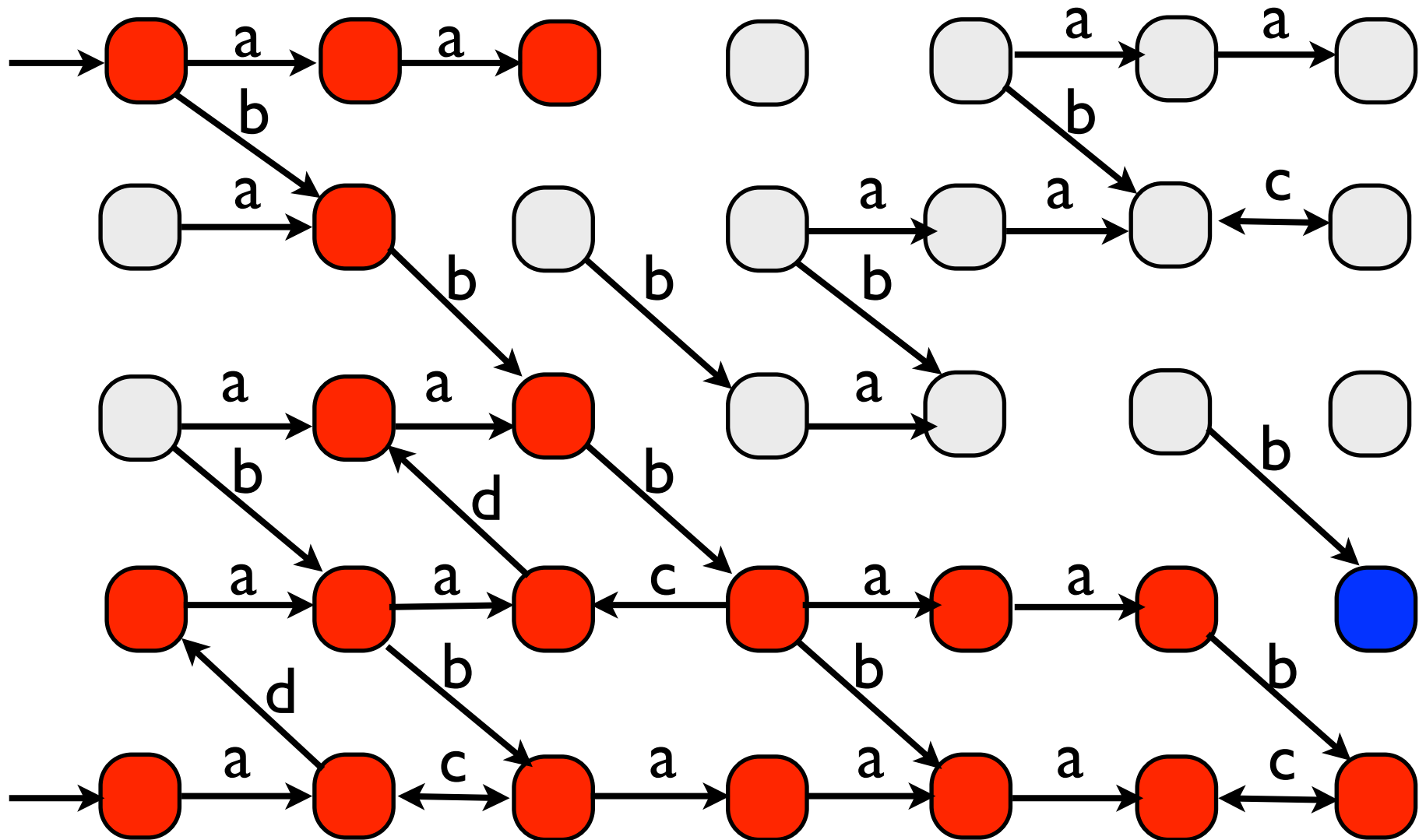
## Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$



# Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$

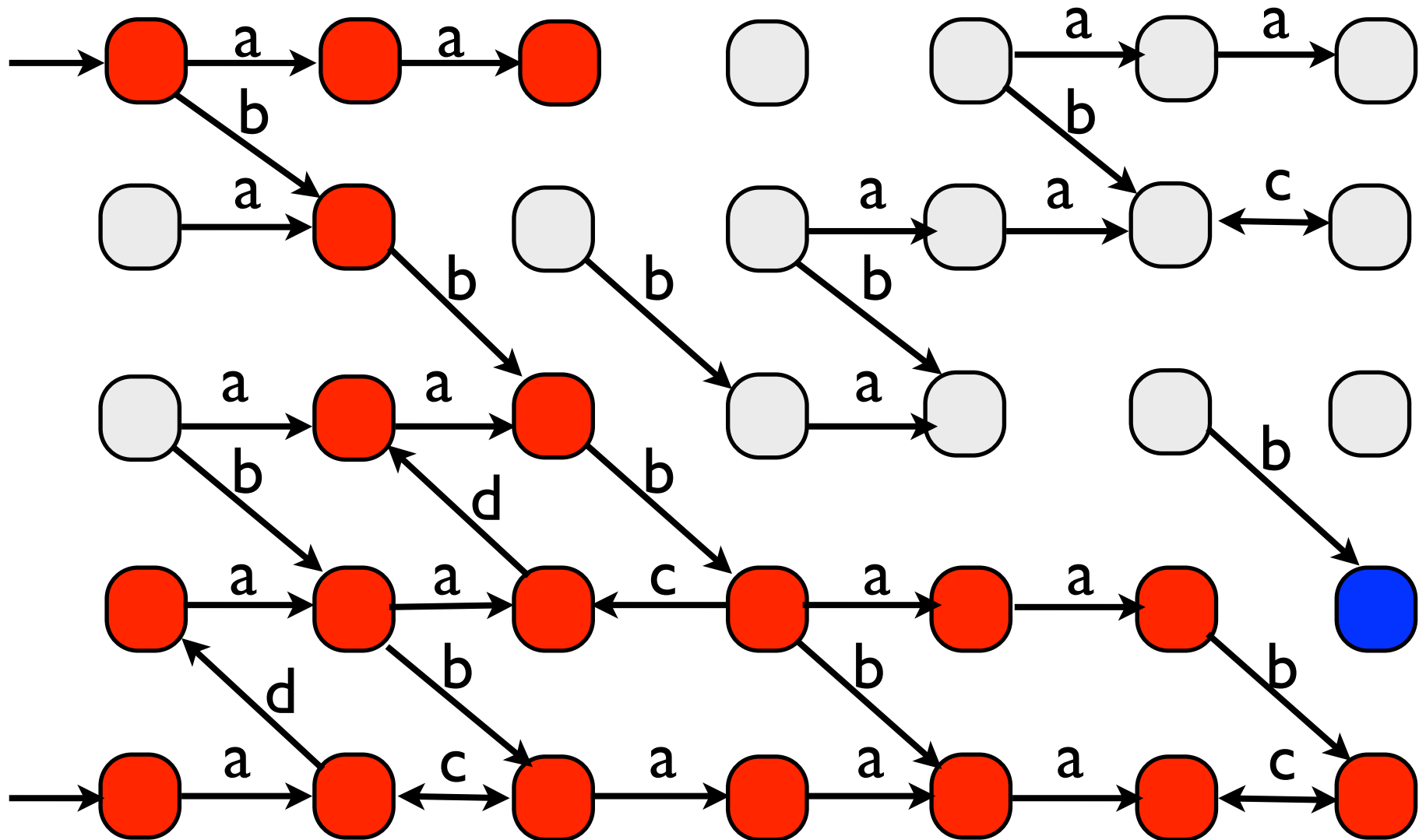


# Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$



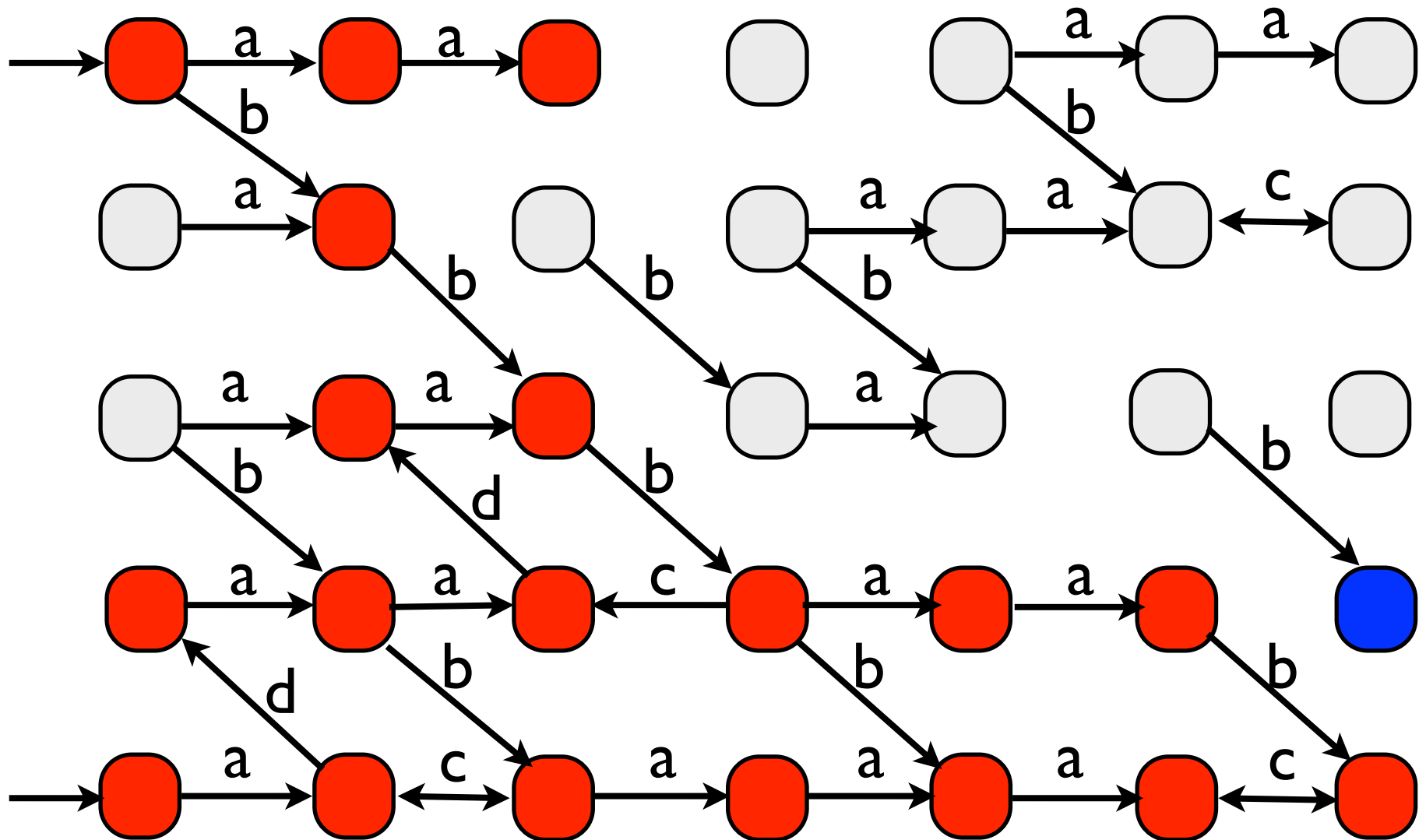
# Iterative evaluation of $\text{Ifp } (\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X))$

Fixed point !



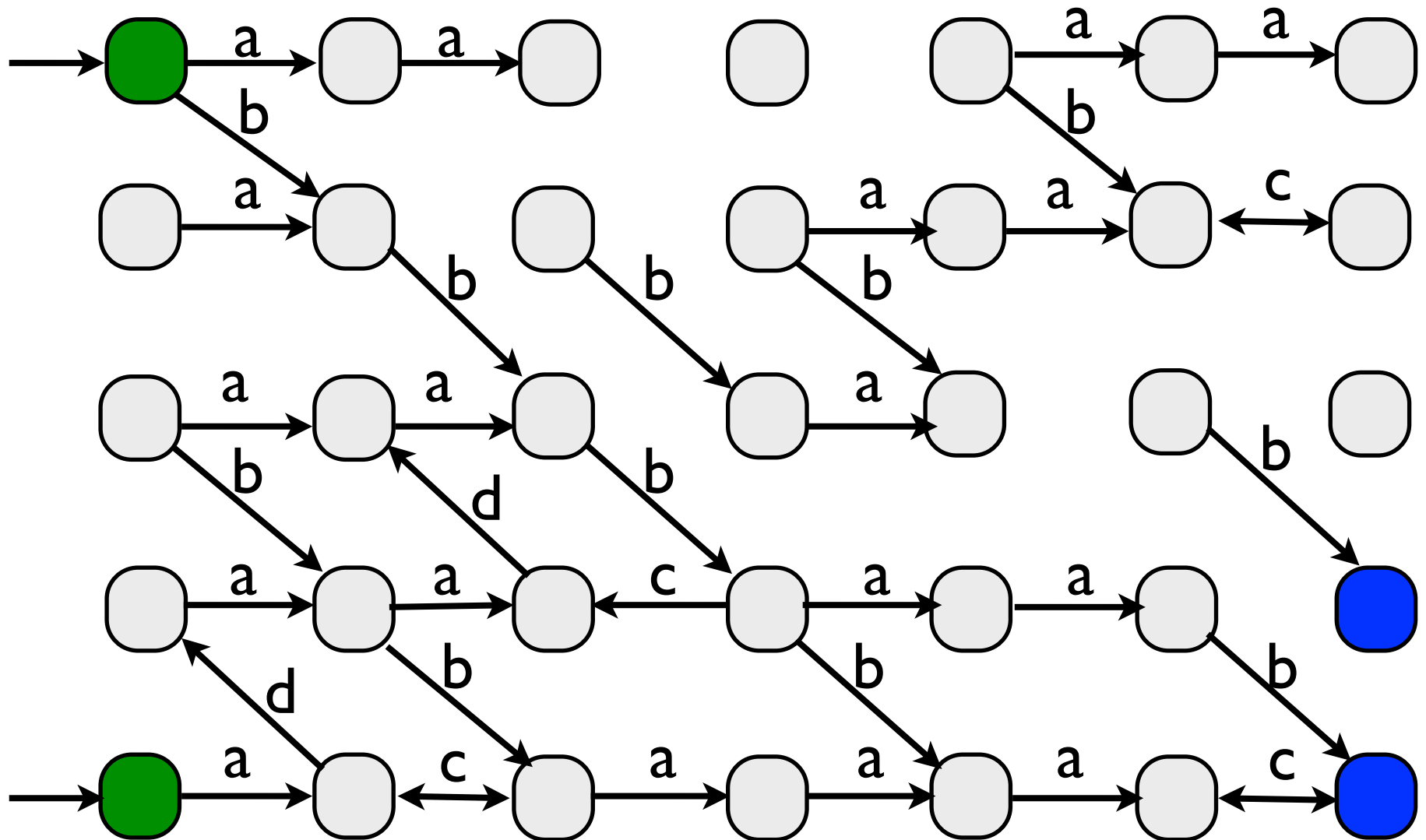
Iterative evaluation of **Ifp** ( $\lambda X. \mathbf{S_0} \cup \mathbf{POST}(X)$ )

**Fixed point !** It intersects **Goal !** Positive instance.

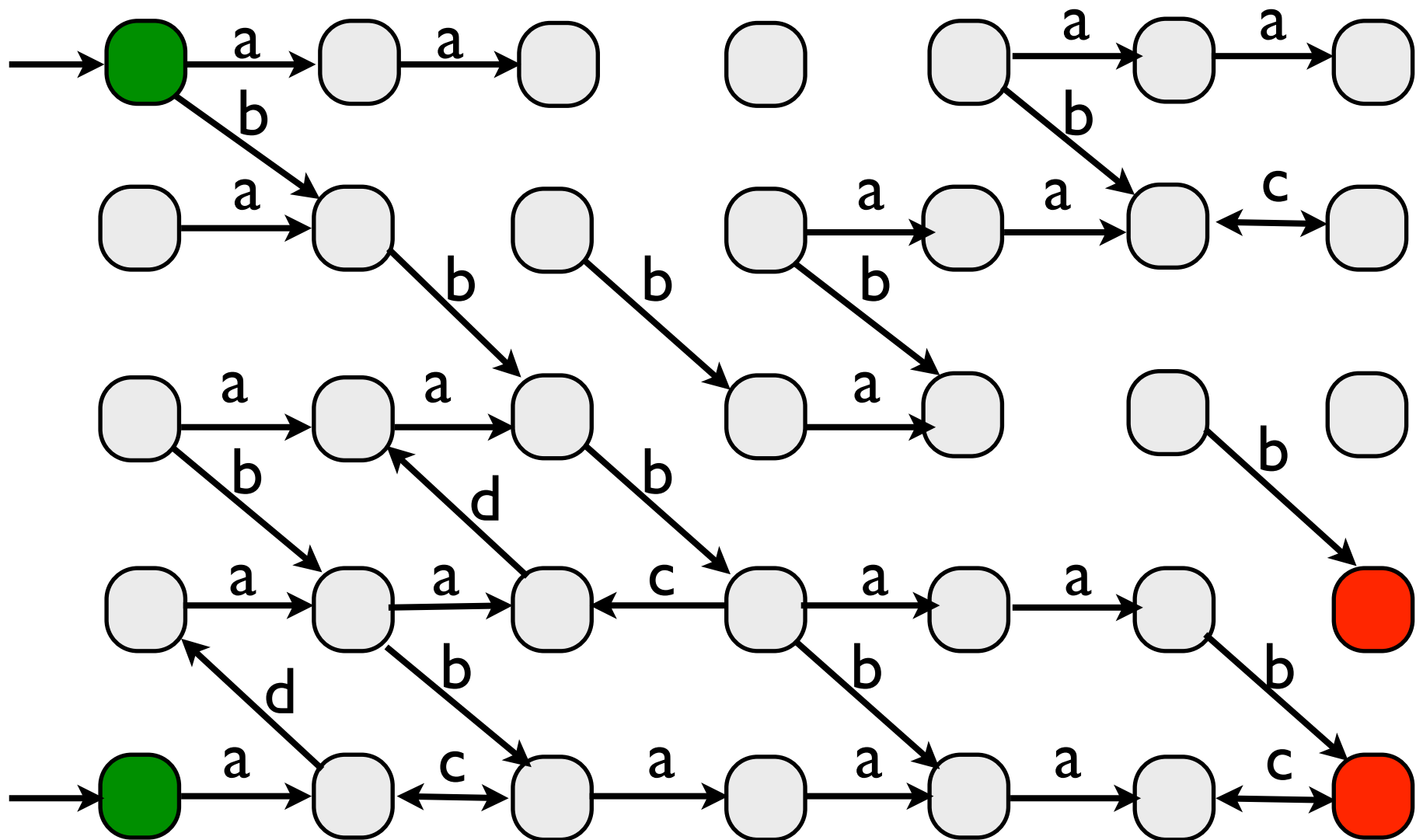


# Reachability - Backward algorithm

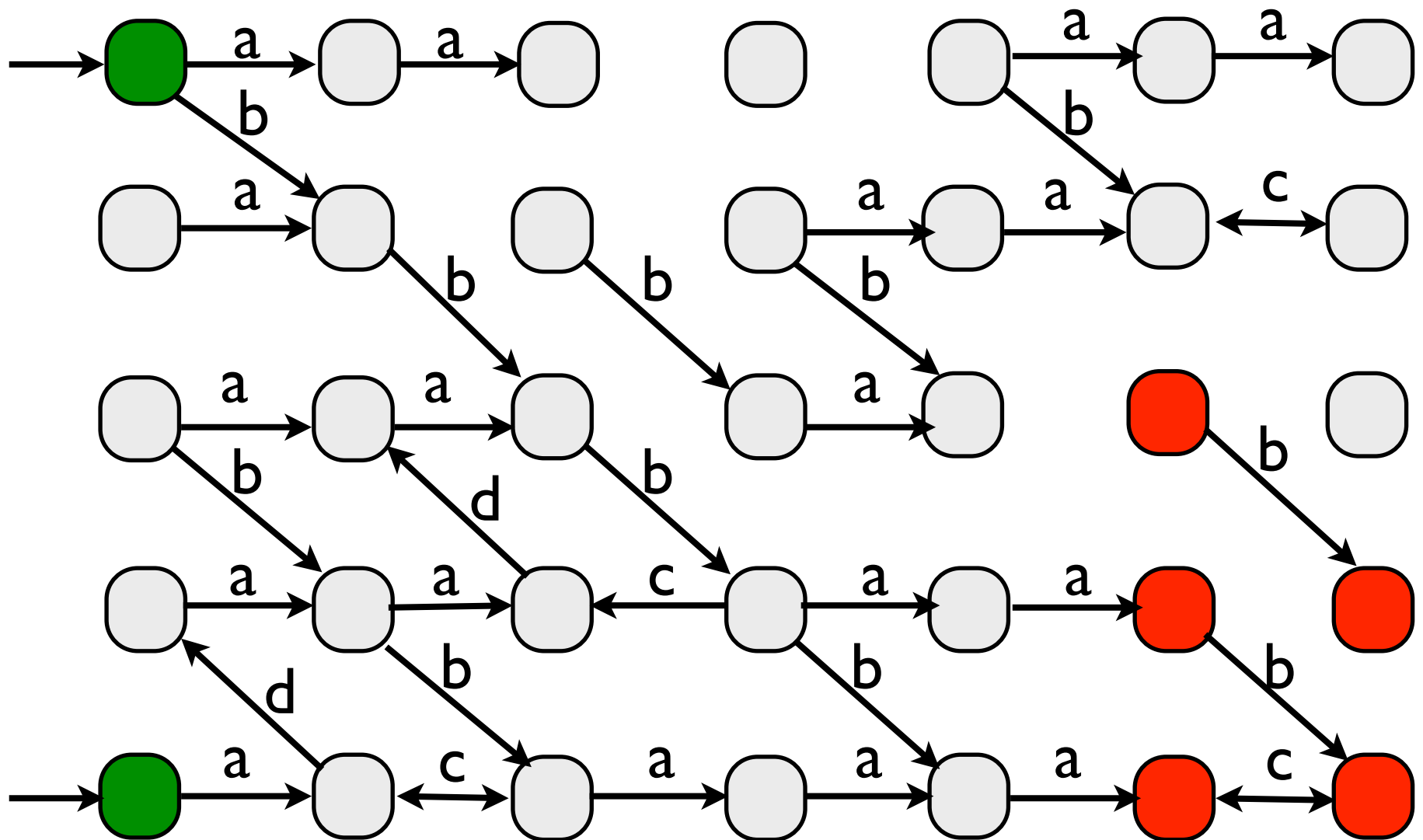
$$\text{Ifp } (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S}_0 \neq \emptyset$$



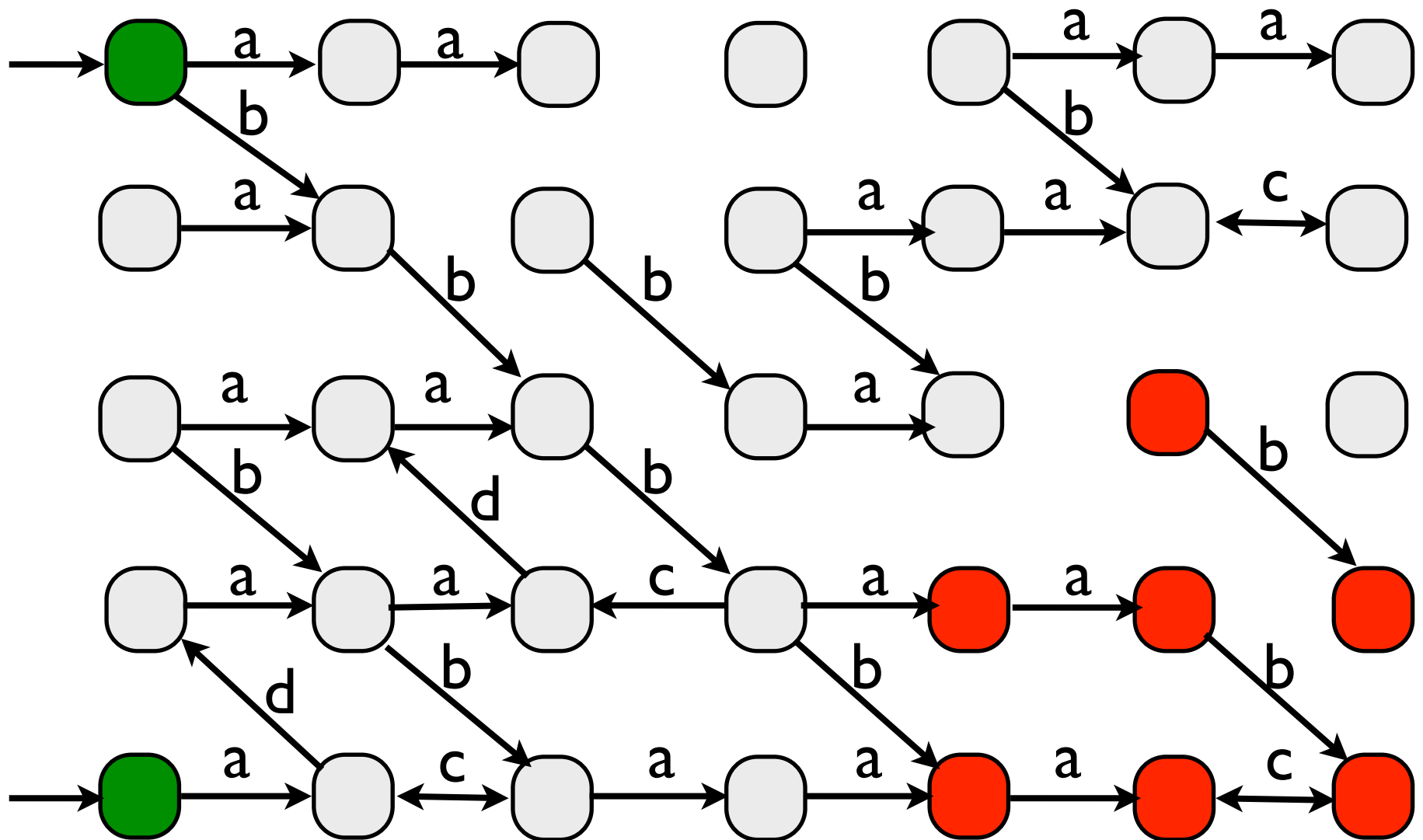
Iterative evaluation of  $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$



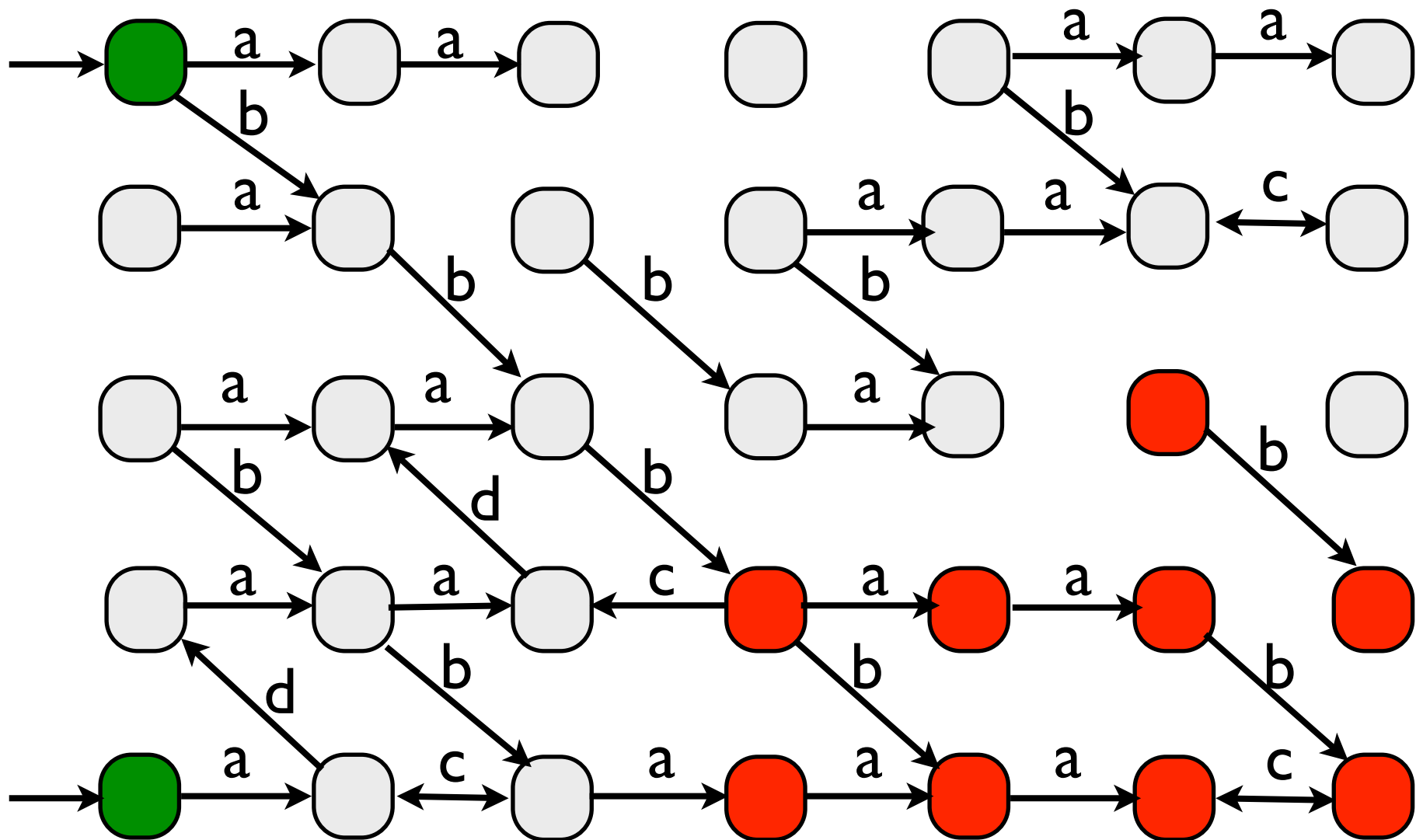
Iterative evaluation of  $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$



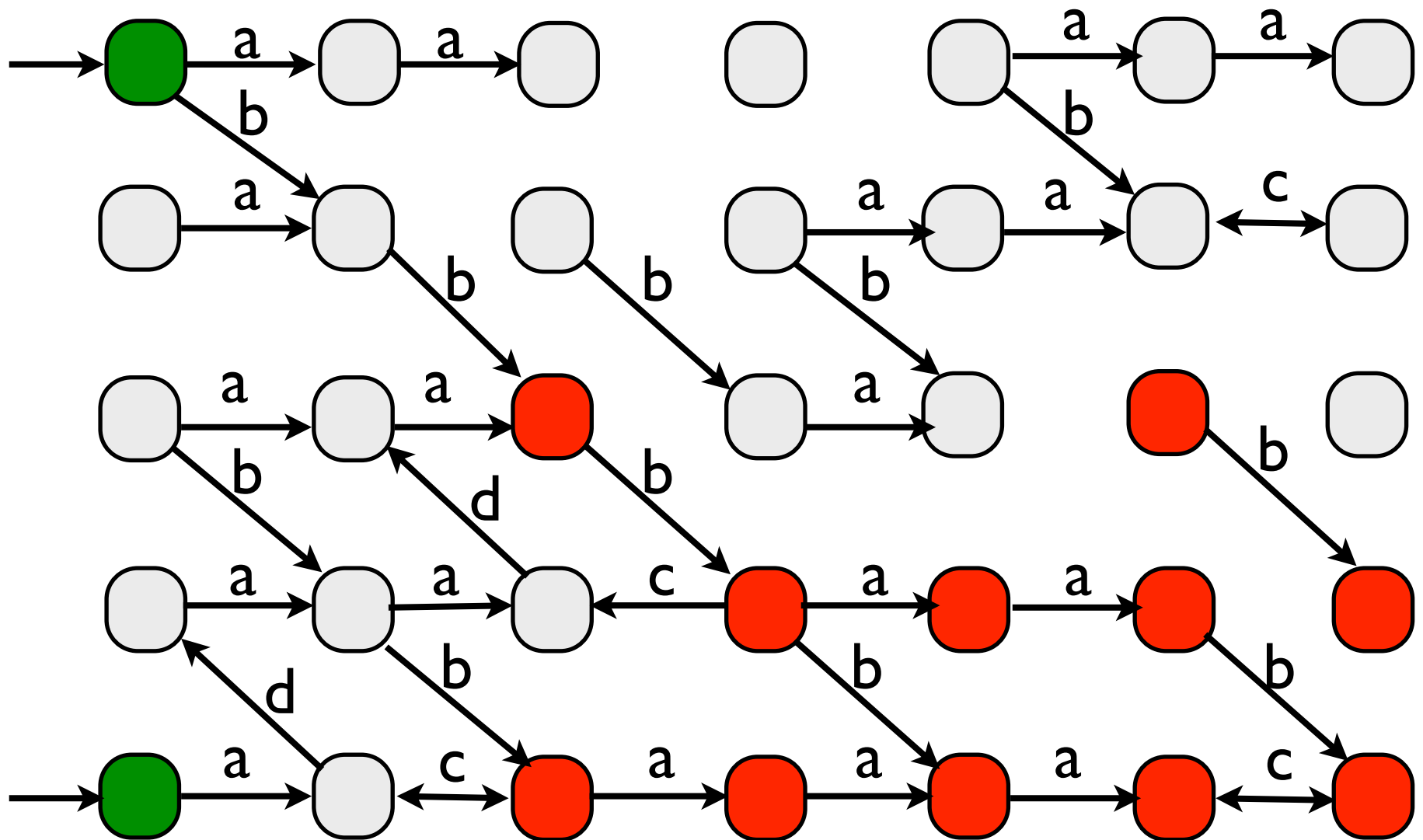
Iterative evaluation of  $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$

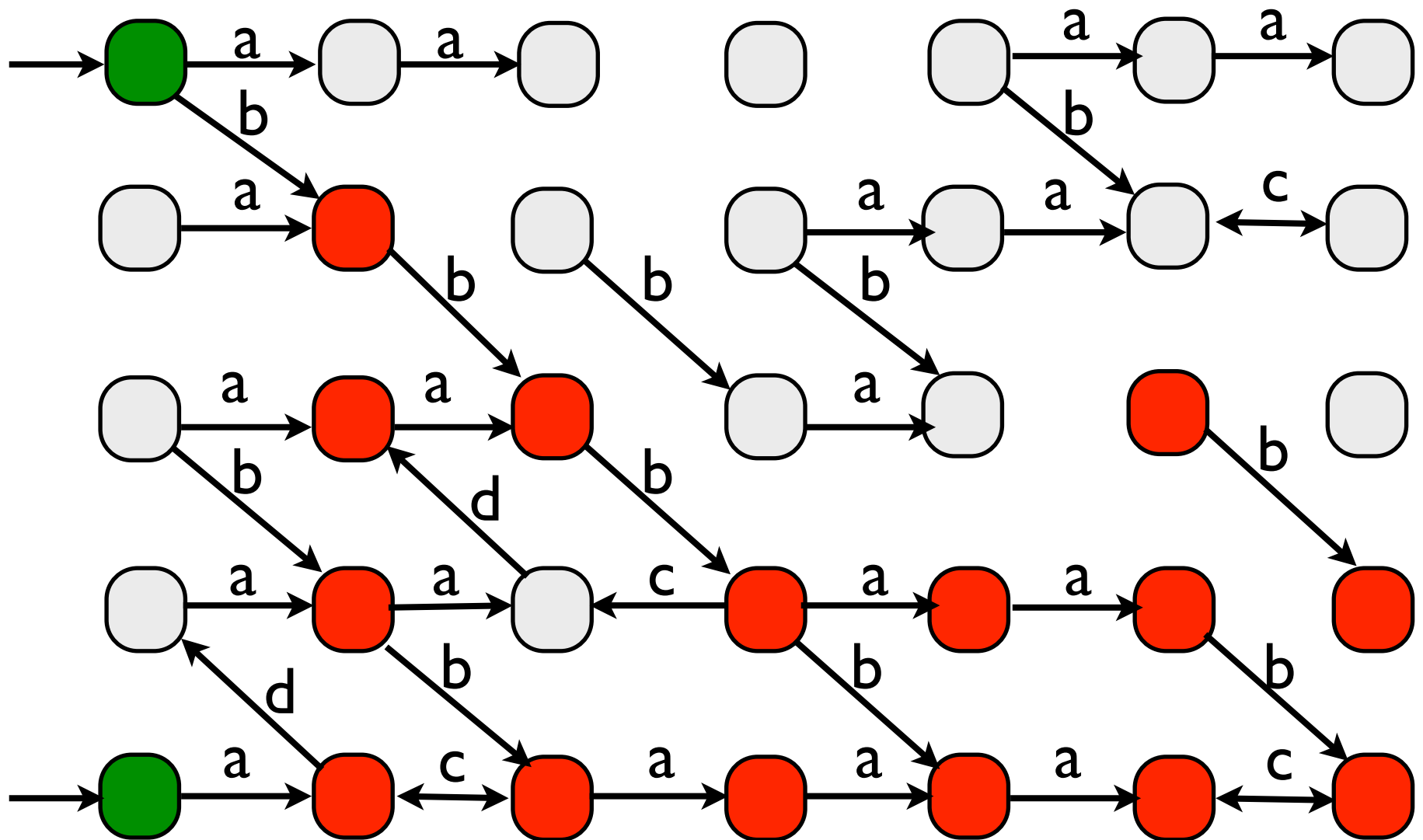


Iterative evaluation of  $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$

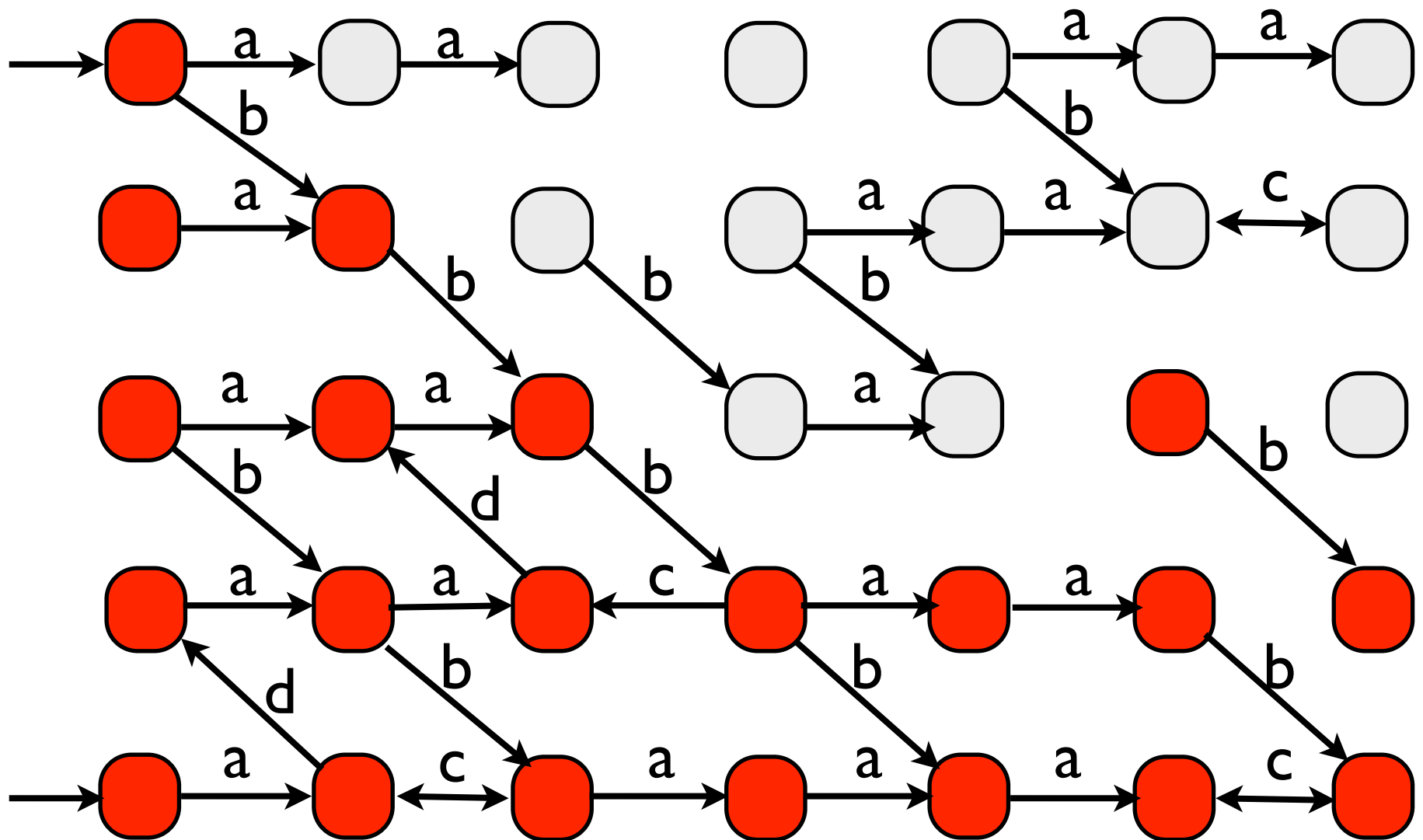


Iterative evaluation of  $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$





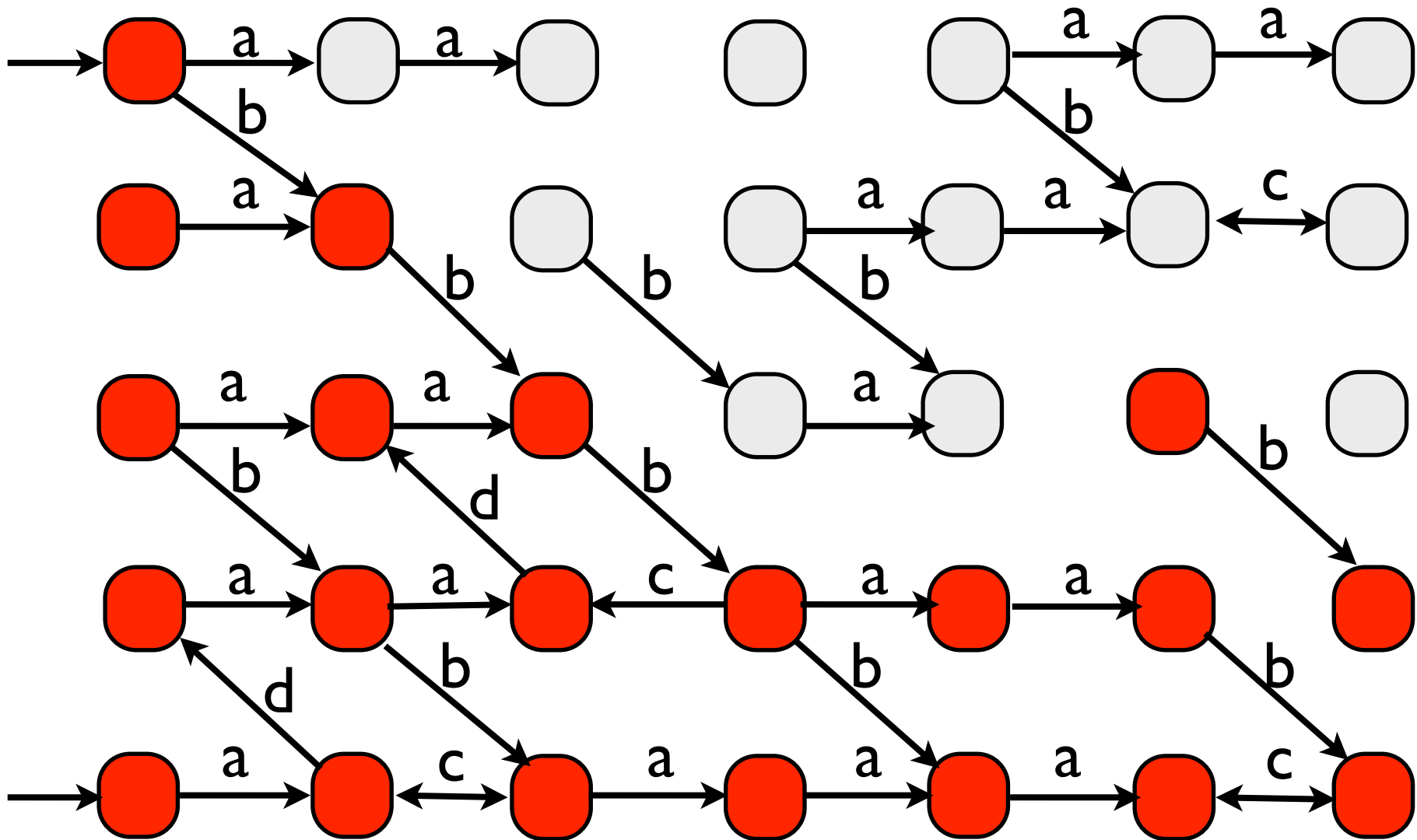
Iterative evaluation of  $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$



## Iterative evaluation of $\mathbf{lfp} (\lambda X. \mathbf{Goal} \cup \mathbf{PRE}(X)) \cap \mathbf{S_0} \neq \emptyset$

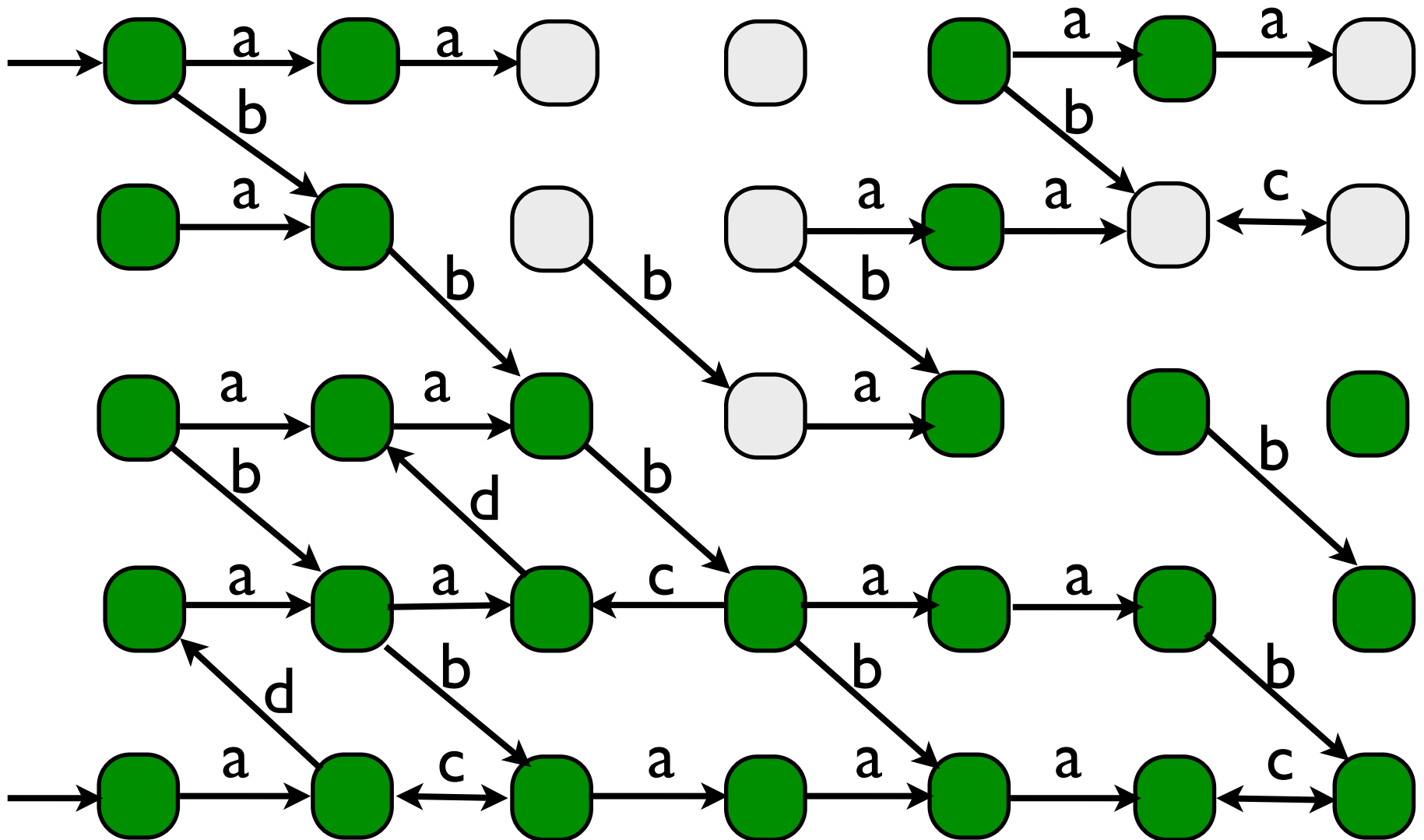
# Fixed point !

It intersects  $S_0$  ! Positive instance.

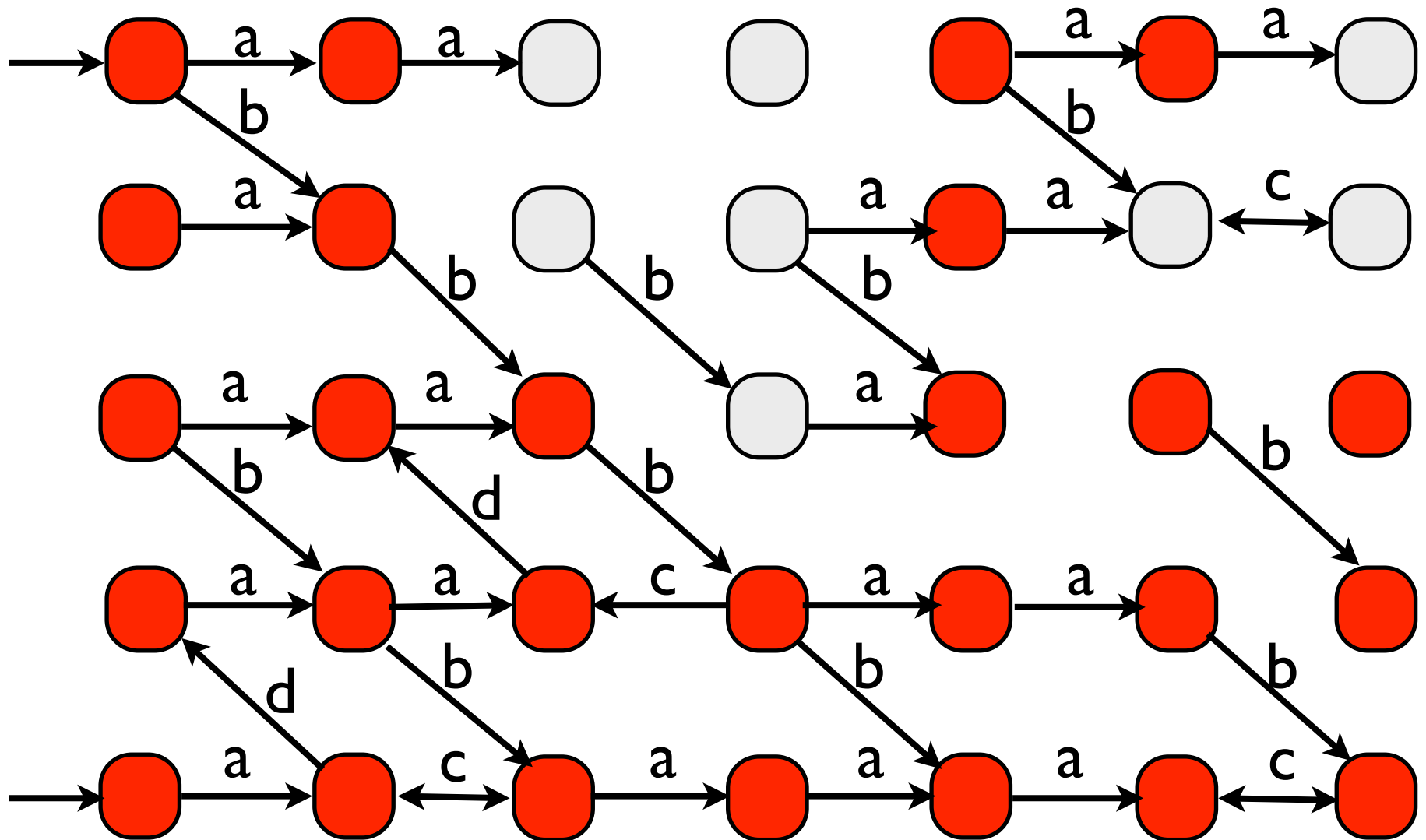


# Safety - Backward algorithm

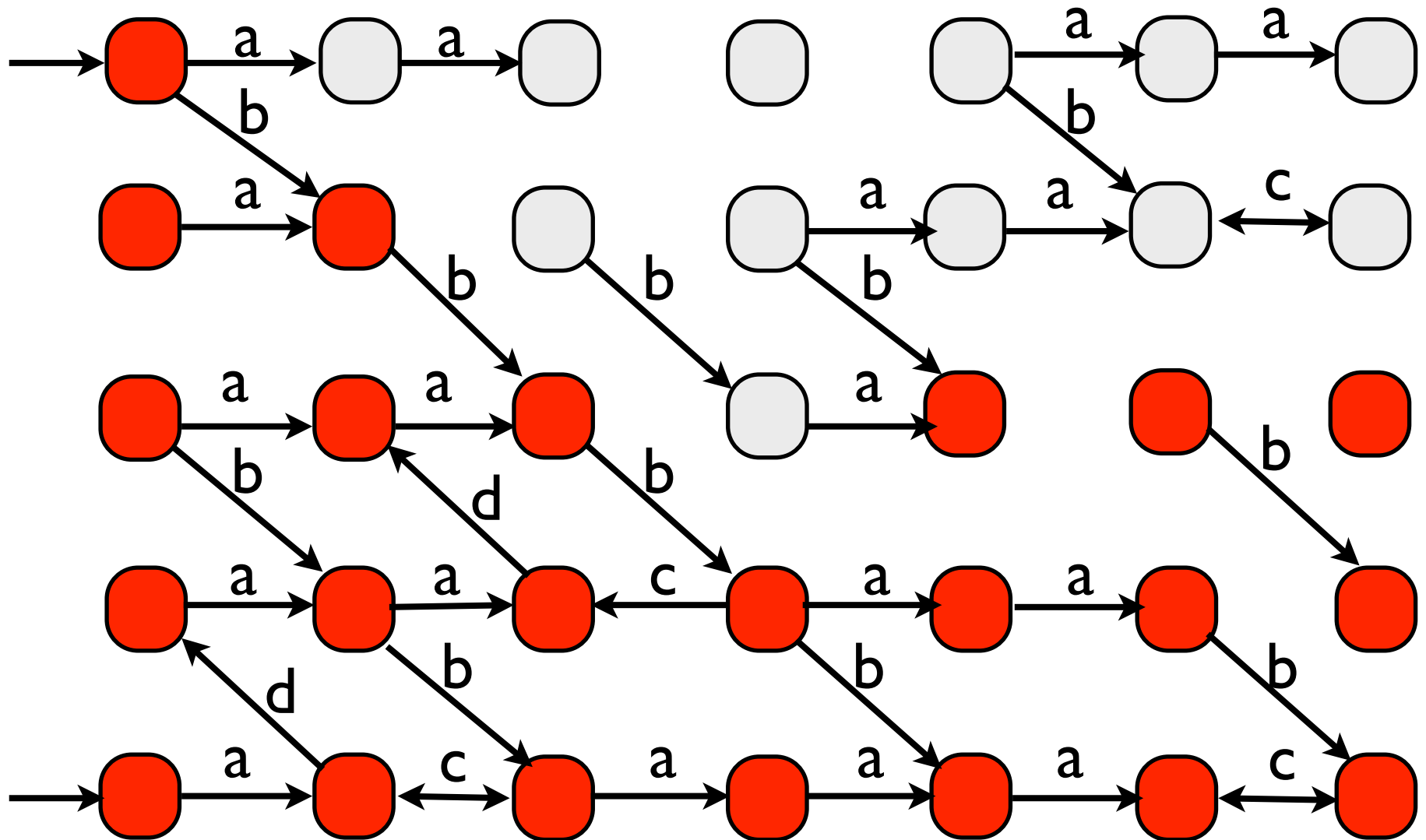
$$S_0 \subseteq \text{gfp } (\lambda X. \text{ Safe} \cap \mathbf{APRE}(X))$$



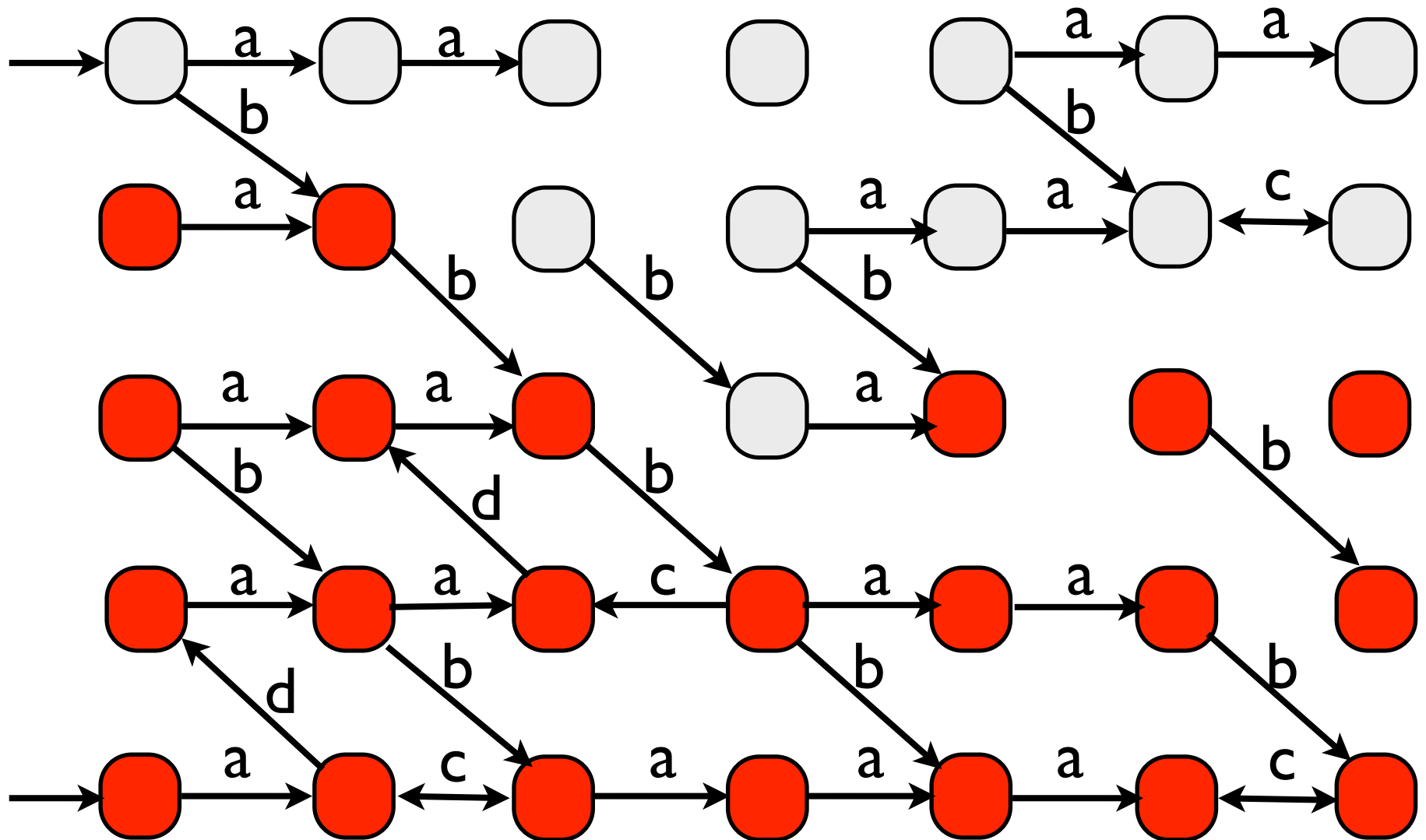
# Iterative evaluation of **gfp** ( $\lambda X. \text{Safe} \sqcap \mathbf{APRE}(X)$ )



# Iterative evaluation of **gfp** ( $\lambda X. \text{Safe} \sqcap \mathbf{APRE}(X)$ )

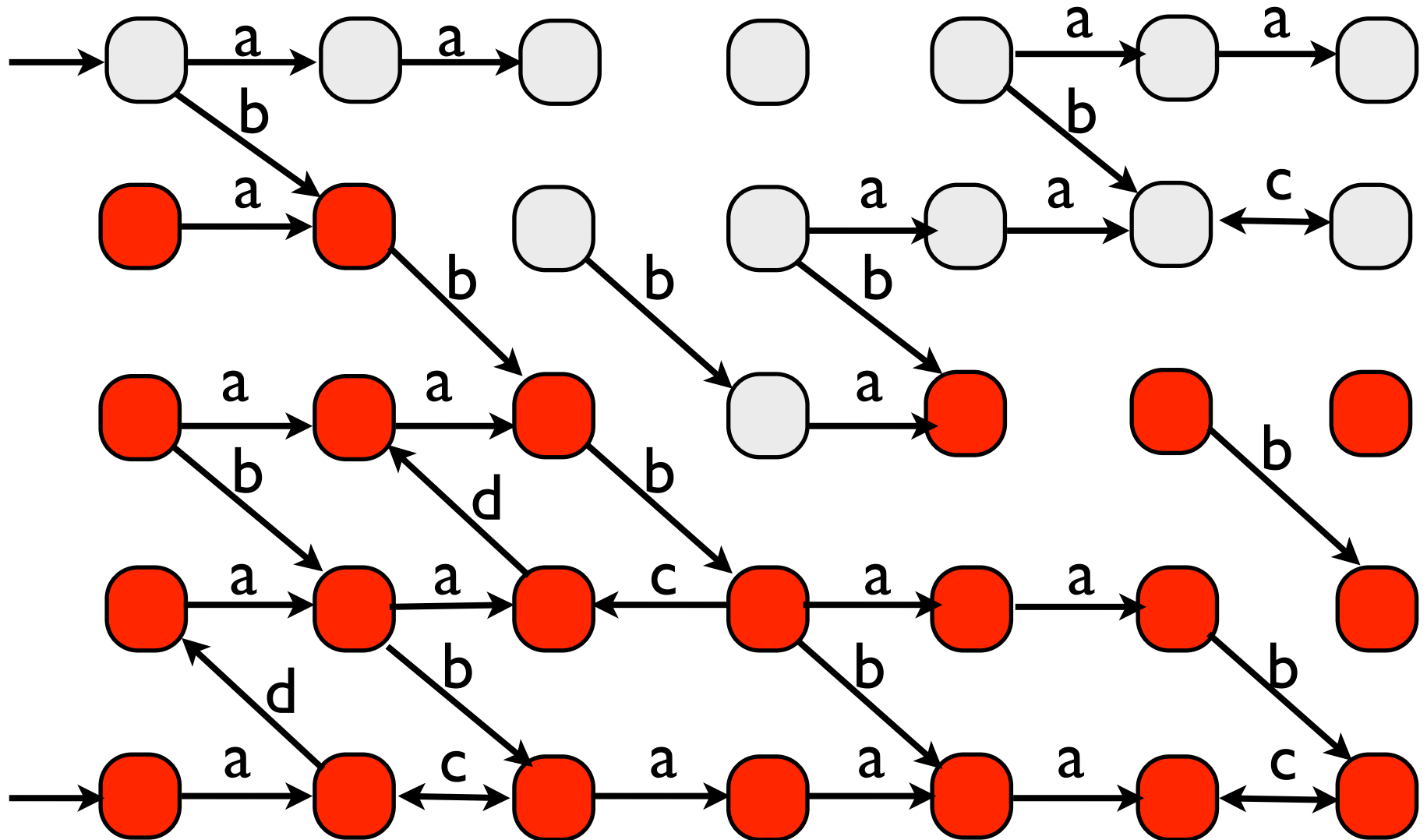


# Iterative evaluation of **gfp** ( $\lambda X. \text{Safe} \sqcap \mathbf{APRE}(X)$ )



Iterative evaluation of **gfp** ( $\lambda X. \text{Safe} \sqcap \mathbf{APRE}(X)$ )

**Fixed point !** Negative instance as  $S_0 \not\sqsubseteq \mathbf{gfp} (\lambda X. \text{Safe} \sqcap \mathbf{APRE}(X))$ .



# Fixed points for Büchi objectives

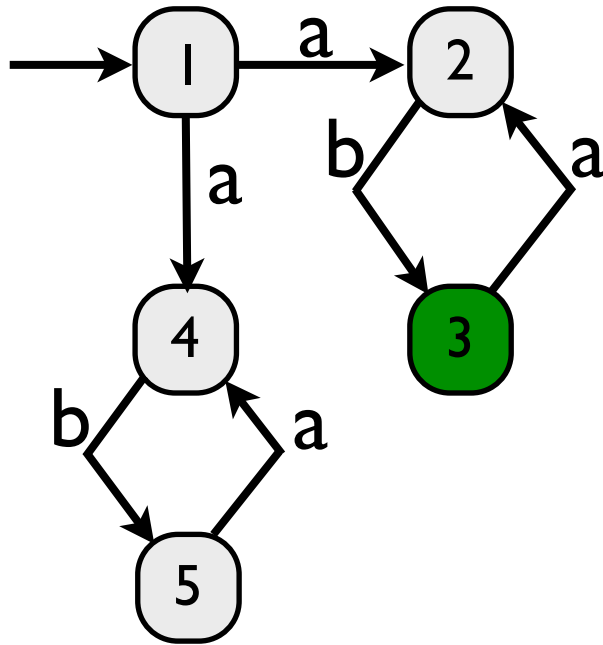
- Let consider an instance of the **Büchi verification problem** given by the LTS  $L=(S,S_0,\Sigma,T,C,\lambda)$ , a set of states **Goal**  $\subseteq S$ ;
- **Goal** is reachable infinitely often from an initial states in L

iff

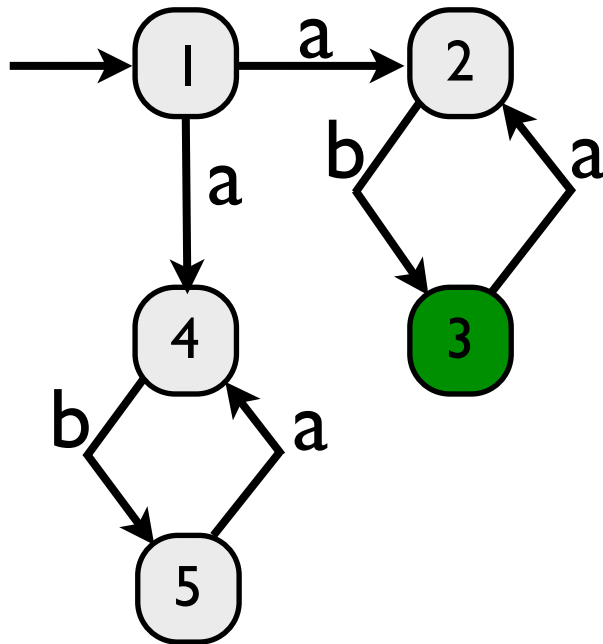
$$\mathbf{gfp}(\lambda Y. \mathbf{lfp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y)))) \cap \mathbf{S_0} \neq \emptyset$$

this is a **backward algorithm**

# Fixed points for Büchi objectives

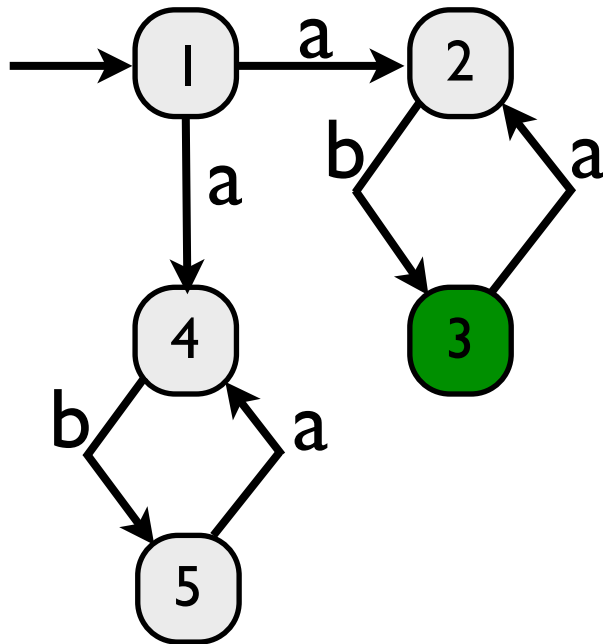


# Fixed points for Büchi objectives



We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

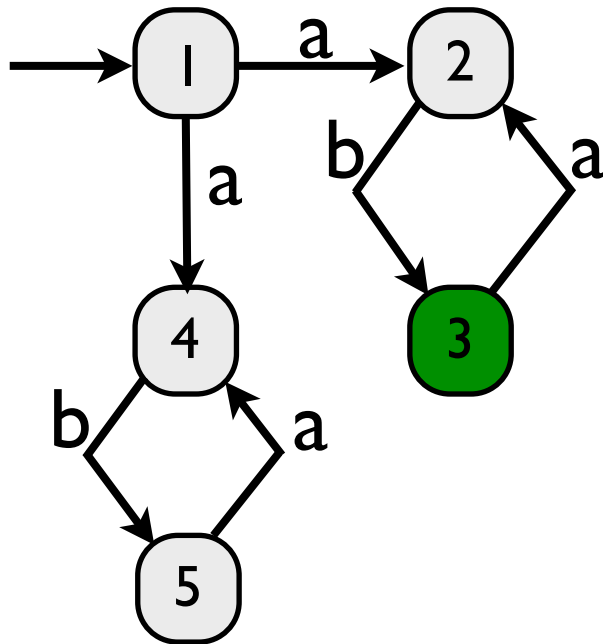
# Fixed points for Büchi objectives



We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{lfp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

# Fixed points for Büchi objectives

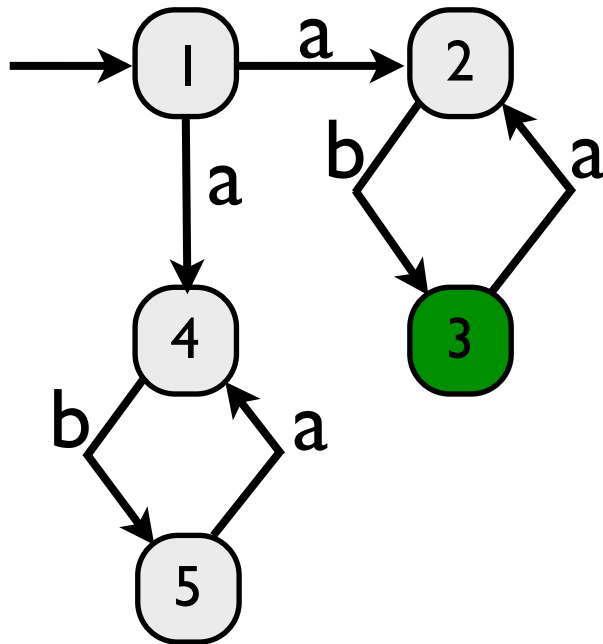


We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{lfp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

$Y_0 = \{1, 2, 3, 4, 5\}$

# Fixed points for Büchi objectives



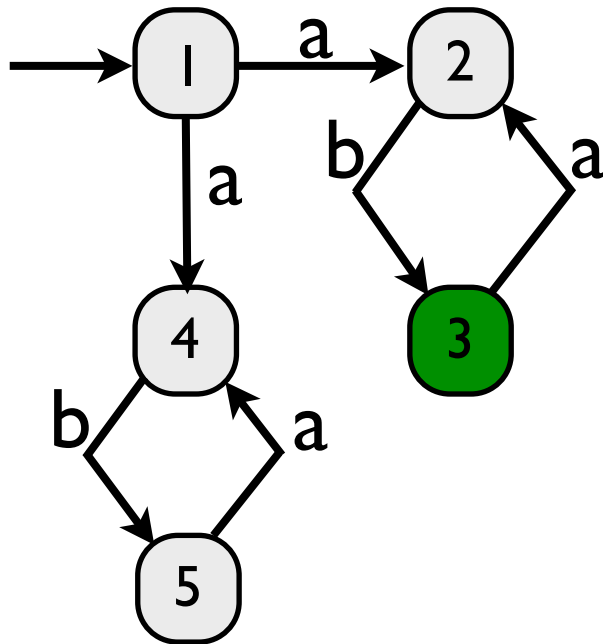
We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

$$Y_0 = \{1, 2, 3, 4, 5\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_0))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal})) \\ &= \{1, 2, 3\} = Y_1 \end{aligned}$$

# Fixed points for Büchi objectives



We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

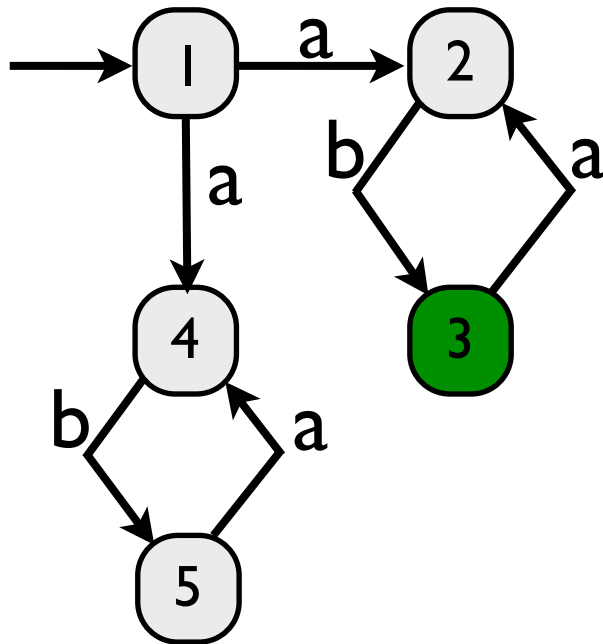
For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

$$Y_0 = \{1, 2, 3, 4, 5\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_0))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal})) \\ &= \{1, 2, 3\} = Y_1 \end{aligned}$$

$$Y_1 = \{1, 2, 3\}$$

# Fixed points for Büchi objectives



We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

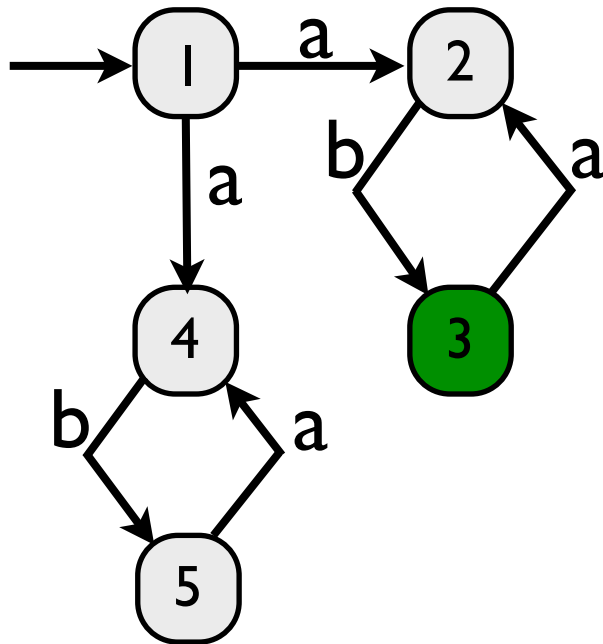
$$Y_0 = \{1, 2, 3, 4, 5\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_0))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal})) \\ &= \{1, 2, 3\} = Y_1 \end{aligned}$$

$$Y_1 = \{1, 2, 3\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_1))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \{1, 2, 3\})) \\ &= \{1, 2, 3\} = Y_2 = Y_1 \end{aligned}$$

# Fixed points for Büchi objectives



We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

$$Y_0 = \{1, 2, 3, 4, 5\}$$

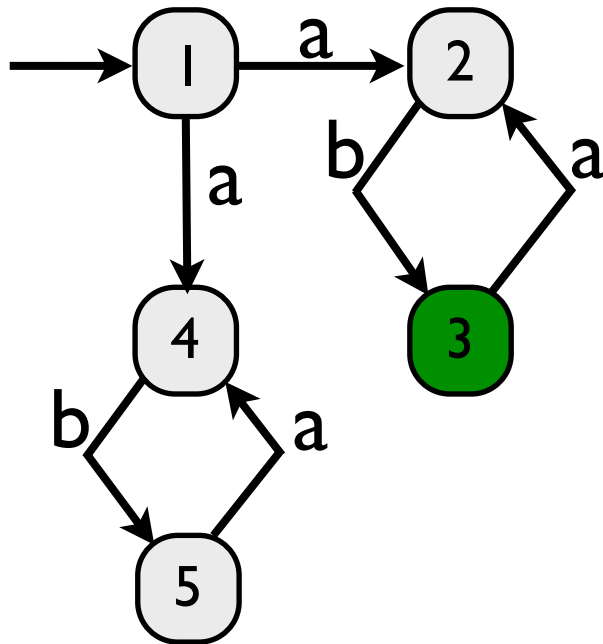
$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_0))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal})) \\ &= \{1, 2, 3\} = Y_1 \end{aligned}$$

$$Y_1 = \{1, 2, 3\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_1))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \{1, 2, 3\})) \\ &= \{1, 2, 3\} = Y_2 = Y_1 \end{aligned}$$

**Fixed point !**

# Fixed points for Büchi objectives



We want to check if  $\{3\}$  can be reached infinitely often from the initial state.

For that we evaluate the fixed point expression  $\mathbf{gfp}(\lambda Y. \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y))))$

$$Y_0 = \{1, 2, 3, 4, 5\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_0))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal})) \\ &= \{1, 2, 3\} = Y_1 \end{aligned}$$

$$Y_1 = \{1, 2, 3\}$$

$$\begin{aligned} & \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \mathbf{PRE}(Y_1))) \\ &= \mathbf{Ifp}(\lambda X. \mathbf{PRE}(X) \cup (\mathbf{Goal} \cap \{1, 2, 3\})) \\ &= \{1, 2, 3\} = Y_2 = Y_1 \end{aligned}$$

**Fixed point !**

As  $S_0 \cap Y_2 \neq \emptyset$ , the Büchi property is verified by the LTS L

# Trace pre-orders, Trace equivalence, Simulations, Bisimulations and quotients

# Traces of a LTS

- Let  $(S, S_0, \Sigma, T, C, \lambda)$  be a LTS. Let  $s_0 \sigma_0 s_1 \sigma_1 s_2 \sigma_2 \dots \sigma_{n-1} s_n \dots$  be such that (1)  $s_0 \in S_0$ , (2)  $\forall i \cdot 0 \leq i \cdot T(s_i, \sigma_i, s_{i+1})$ , the sequence

$$\lambda(s_0) \sigma_0 \lambda(s_1) \sigma_1 \lambda(s_2) \sigma_2 \dots \sigma_{n-1} \lambda(s_n) \dots$$

is called a **trace** of the LTS.

- Note that two **different** paths in the LTS may generate the **same** trace.
- The color of a state is meant to model the **important properties** of that state. So the notion of trace allows us to **concentrate** on the important properties of the system.

# Traces of a LTS

- We note **Traces**(L) the set of traces generated by the LTS L.
- Two LTS  $L_1$  and  $L_2$  are **trace equivalent** if

$$\mathbf{Traces}(L_1) = \mathbf{Traces}(L_2)$$

- **Trace equivalence and verification.**

If we have two LTS  $L_1$  and  $L_2$  such that **Traces**( $L_1$ )=**Traces**( $L_2$ ), and  $L_2$  is (much) **smaller** than  $L_1$ , it may be very advantageous to do verification on  $L_2$  instead on  $L_1$ . As we will see  $L_1$  may be infinite while  $L_2$  is finite. We will illustrate that with TA.

- Unfortunately, minimizing a system using the notion of trace equivalence is **costly** computationally. We will introduce now **stronger notions of equivalence** than are easier to compute.

# Simulation relations

# Simulation relations

- Given a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , a **simulation relation** is a relation  $\mathbf{R} \subseteq S \times S$  such that

for all  $(s_1, s_2) \in \mathbf{R}$  :

(1)  $s_1 \in S_0$  iff  $s_2 \in S_0$

(2)  $\lambda(s_1) = \lambda(s_2)$

(3)  $\forall \sigma \in \Sigma \bullet \forall s_3 \in S: T(s_1, \sigma, s_3) \implies \exists s_4 \in S \bullet T(s_2, \sigma, s_4) \wedge (s_3, s_4) \in \mathbf{R}$

# Simulation relations

- Given a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , a **simulation relation** is a relation  $\mathbf{R} \subseteq S \times S$  such that

for all  $(s_1, s_2) \in \mathbf{R}$  :

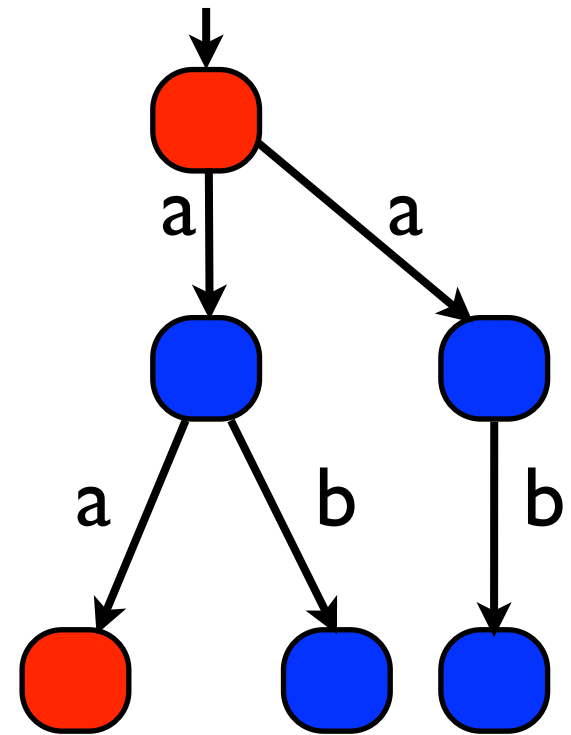
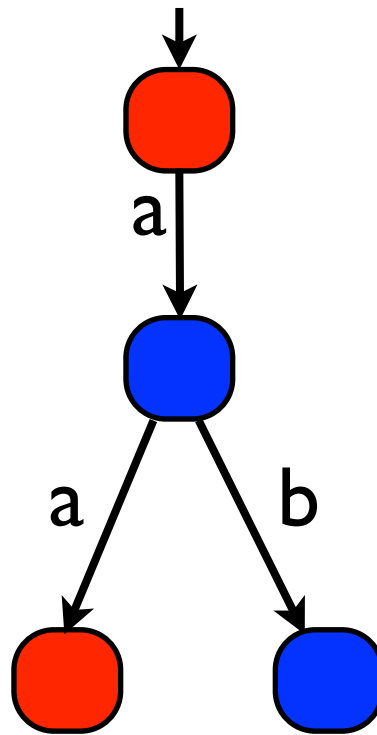
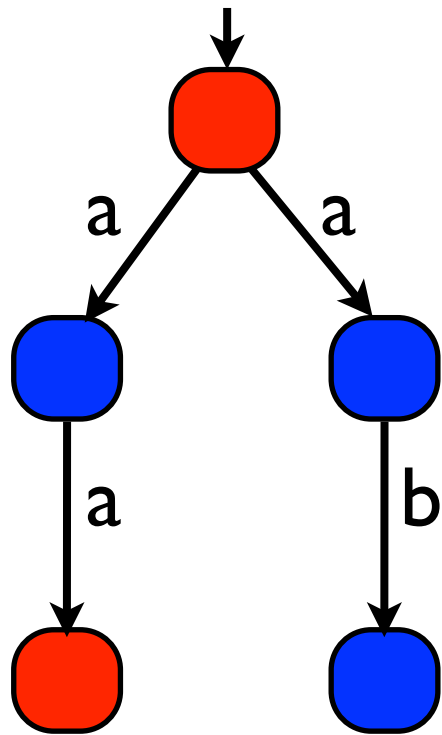
(1)  $s_1 \in S_0$  iff  $s_2 \in S_0$

(2)  $\lambda(s_1) = \lambda(s_2)$

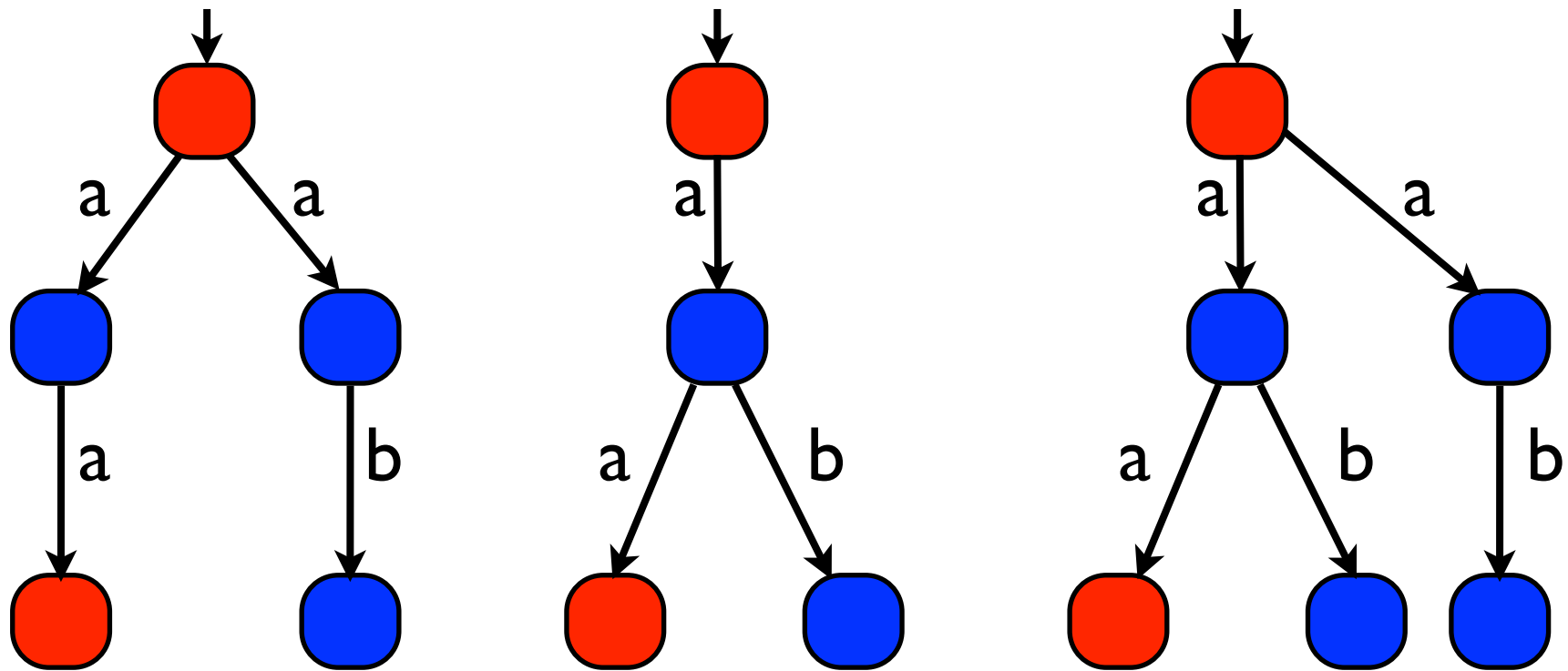
(3)  $\forall \sigma \in \Sigma \cdot \forall s_3 \in S: T(s_1, \sigma, s_3) \implies \exists s_4 \in S \cdot T(s_2, \sigma, s_4) \wedge (s_3, s_4) \in \mathbf{R}$

- When  $(s_1, s_2) \in \mathbf{R}$ , we say that  **$s_1$  is simulated by  $s_2$** .

# Simulation relations

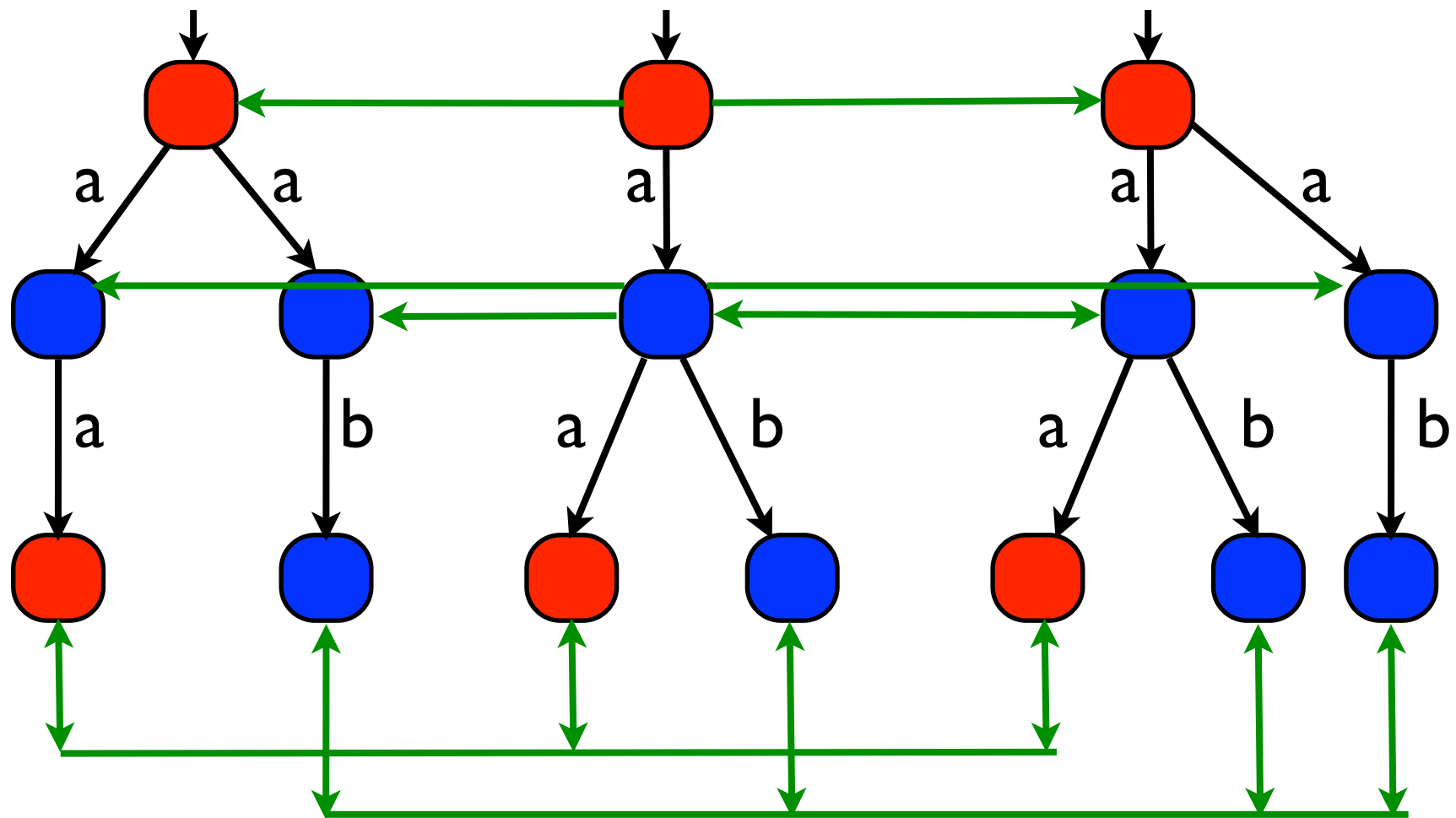


# Simulation relations



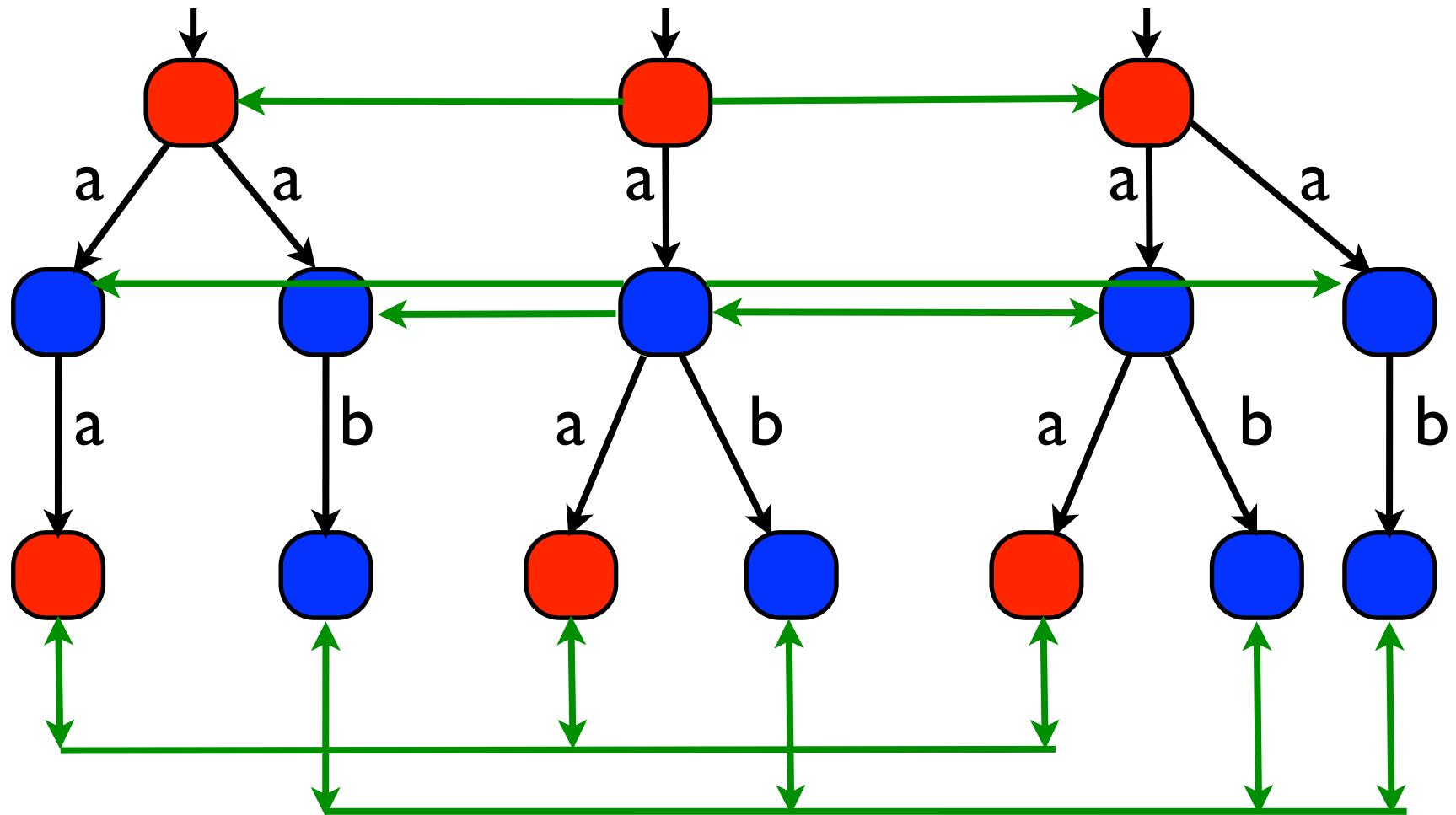
Who can simulate who ?

# Simulation relations



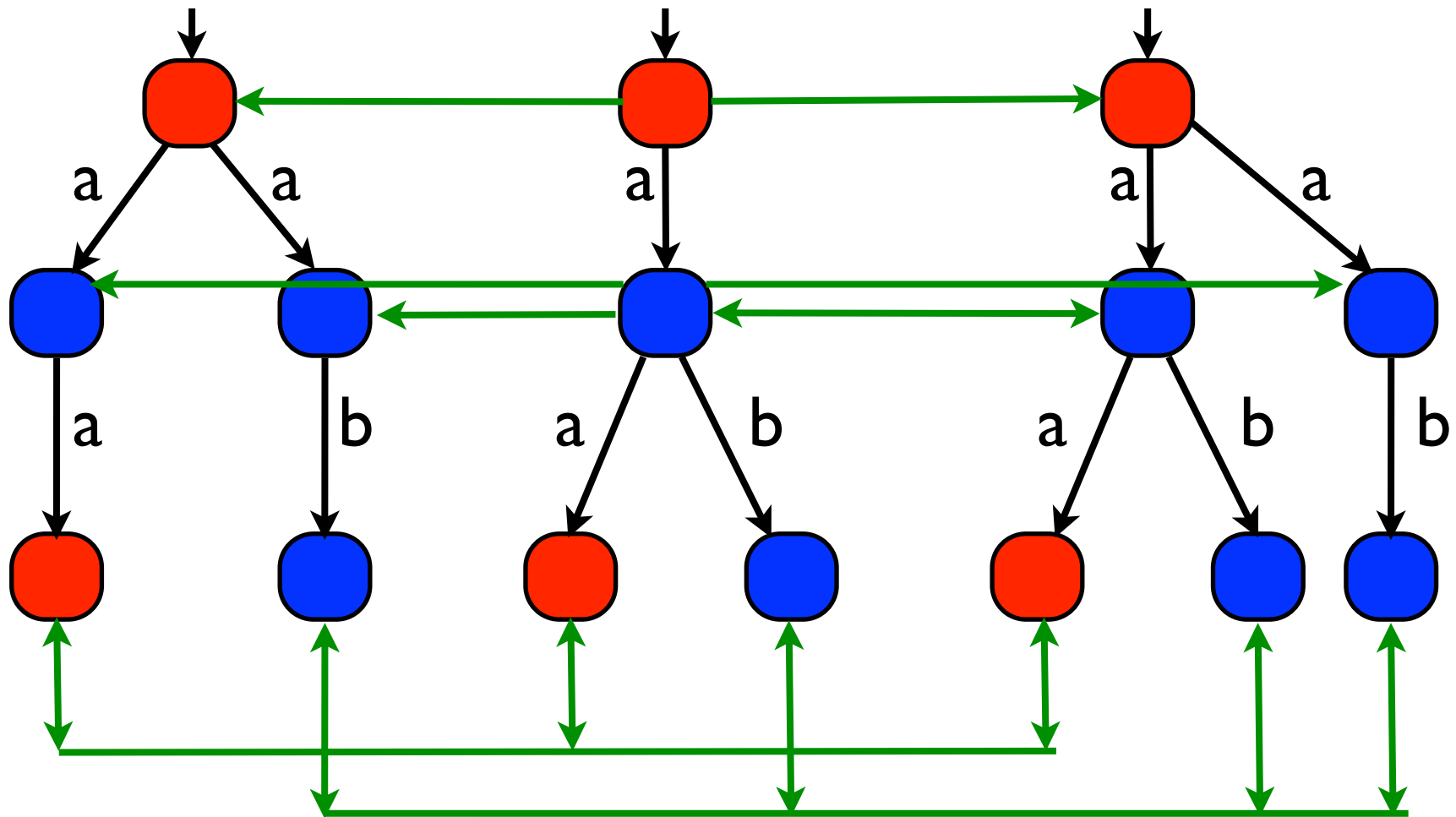
This is a simulation relation

# Simulation relations



Is this the largest one ?

# Simulation relations



Is this the largest one ? **NO.**

# Simulation relations and bisimulations

# Simulation relations and bisimulations

- Given a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , there exists a unique **largest** simulation relation  $\mathbf{R} \subseteq S \times S$ ;

# Simulation relations and bisimulations

- Given a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , there exists a unique **largest** simulation relation  $\mathbf{R} \subseteq S \times S$ ;
- A relation  $\mathbf{R} \subseteq S \times S$  is **symmetric** iff for all  $s_1, s_2$  such that  $\mathbf{R}(s_1, s_2)$  we have also  $\mathbf{R}(s_2, s_1)$ ;

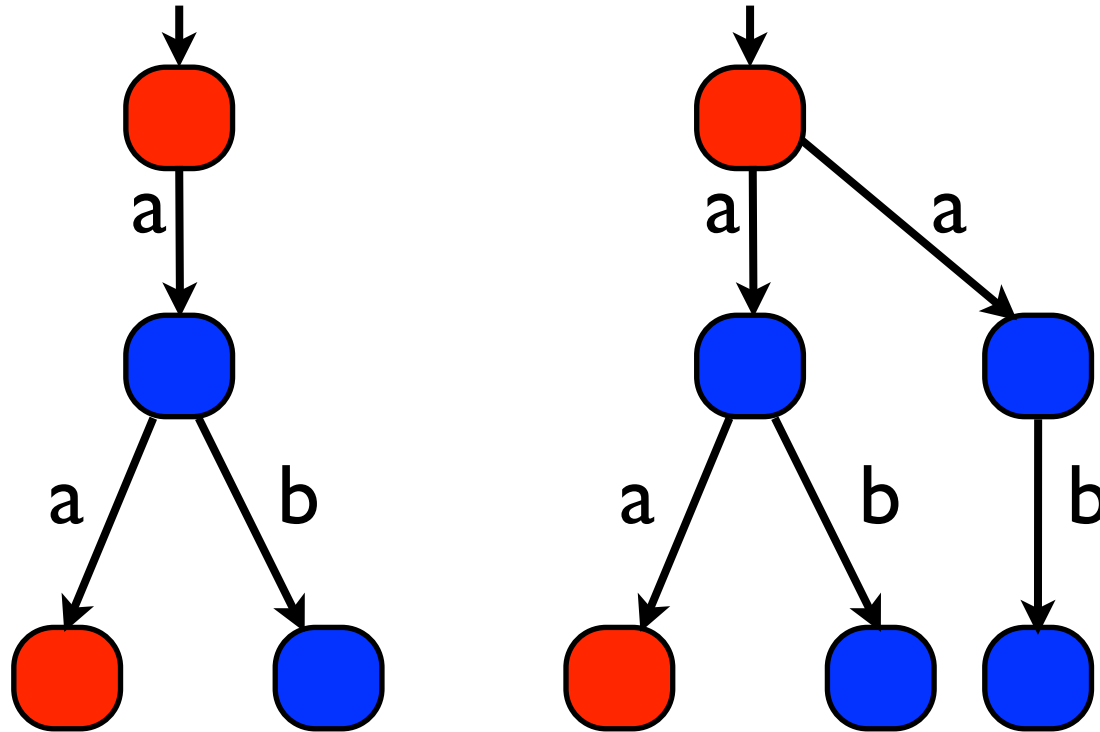
# Simulation relations and bisimulations

- Given a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , there exists a unique **largest** simulation relation  $\mathbf{R} \subseteq S \times S$ ;
- A relation  $\mathbf{R} \subseteq S \times S$  is **symmetric** iff for all  $s_1, s_2$  such that  $\mathbf{R}(s_1, s_2)$  we have also  $\mathbf{R}(s_2, s_1)$ ;
- A simulation relation  $\mathbf{R}$  which is **symmetric** is called a **bisimulation**.

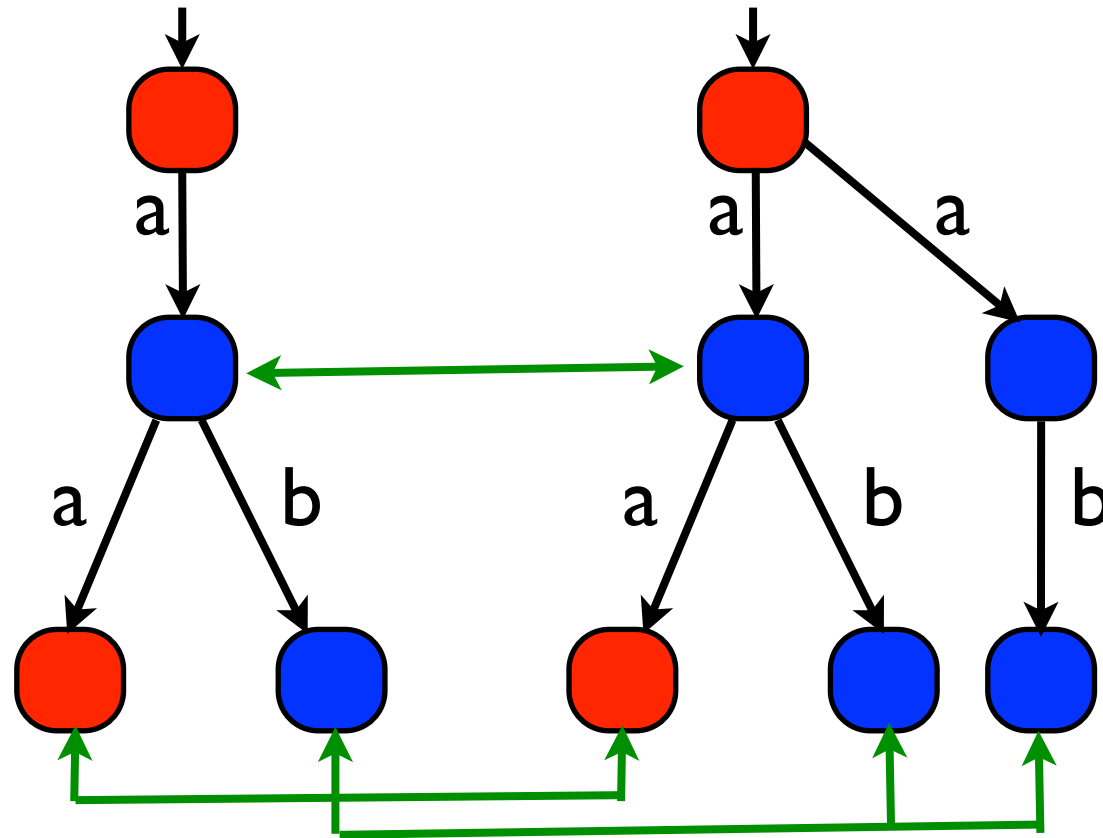
# Simulation relations and bisimulations

- Given a LTS  $(S, S_0, \Sigma, T, C, \lambda)$ , there exists a unique **largest** simulation relation  $\mathbf{R} \subseteq S \times S$ ;
- A relation  $\mathbf{R} \subseteq S \times S$  is **symmetric** iff for all  $s_1, s_2$  such that  $\mathbf{R}(s_1, s_2)$  we have also  $\mathbf{R}(s_2, s_1)$ ;
- A simulation relation  $\mathbf{R}$  which is **symmetric** is called a **bisimulation**.
- Given a bisimulation relation  $\mathbf{R}$  and two states  $s_1, s_2$  such that  $\mathbf{R}(s_1, s_2)$  (note that we have also  $\mathbf{R}(s_2, s_1)$  by definition), we say that  $s_1$  and  $s_2$  are **bisimilar**, this is noted  $s_1 \approx_{\mathbf{R}} s_2$  (or  $s_1 \approx s_2$  if  $\mathbf{R}$  is clear from the context).  
The relation  $\approx_{\mathbf{R}}$  is an **equivalence relation**.

# Bisimulation



# Bisimulation

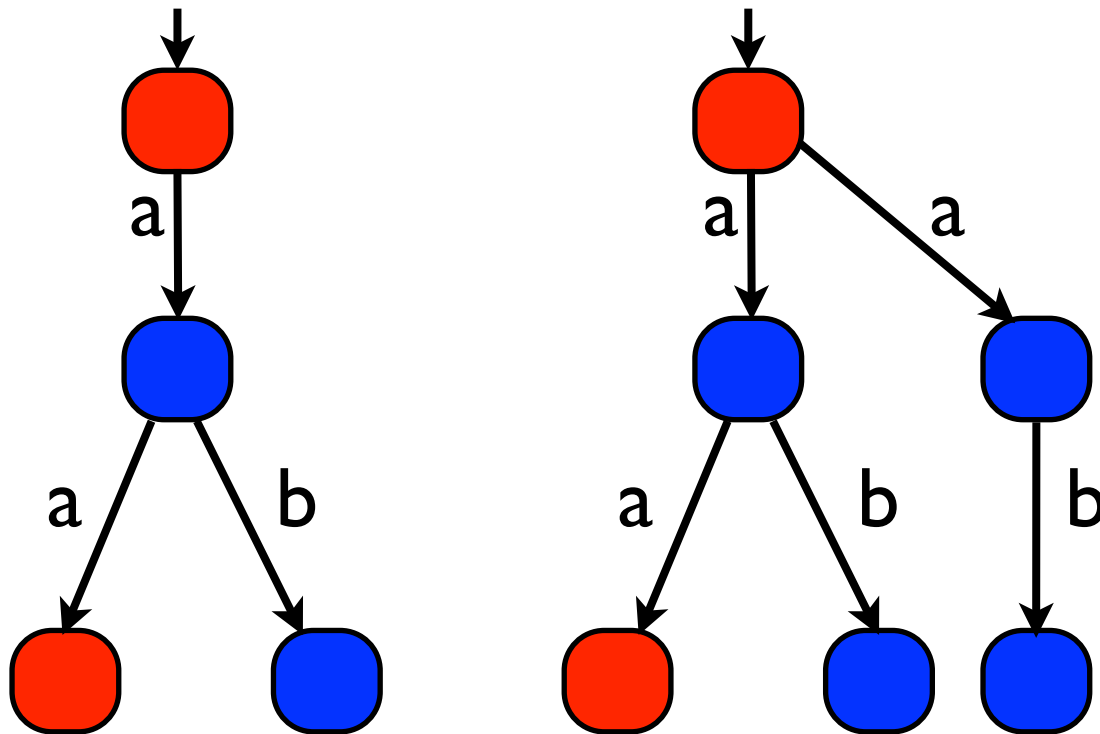


This is a bisimulation

# Quotient of a LTS using bisimulation

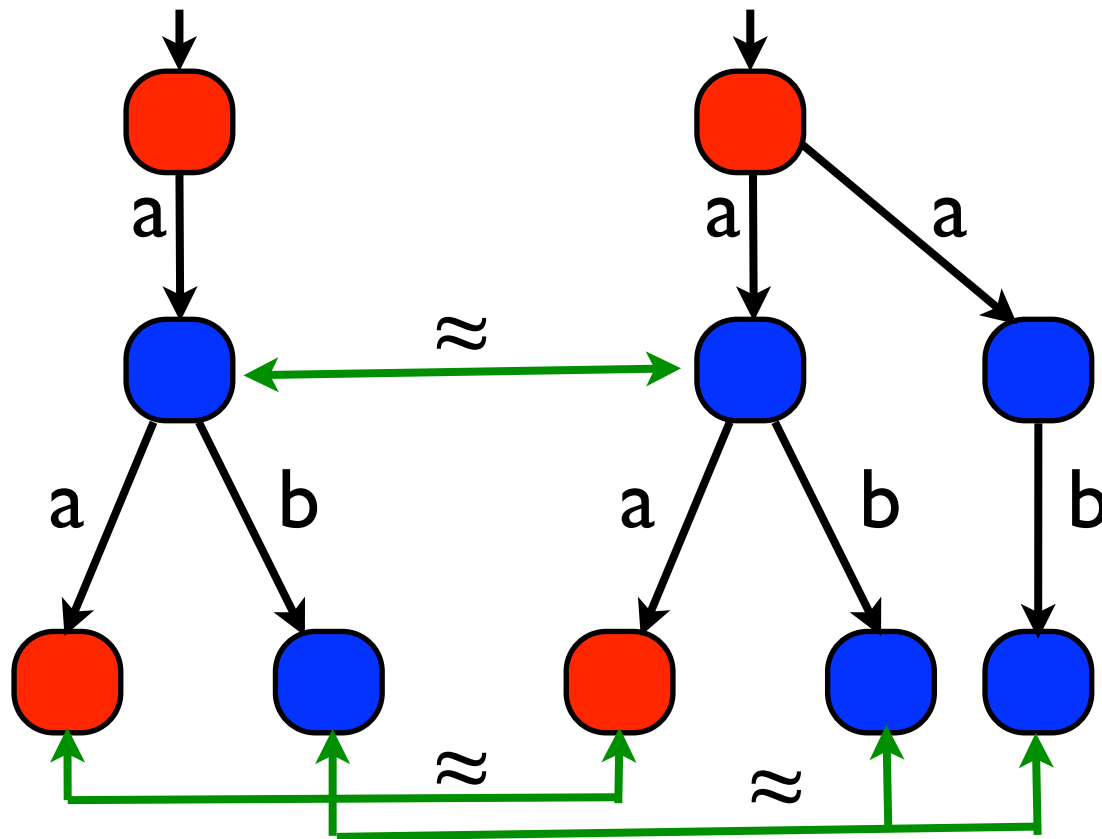
- Let  $L=(S,S_0,\Sigma,T,C,\lambda)$  be a LTS,  $\mathbf{R}\subseteq S\times S$  be a **bisimulation relation** over the state space of  $L$ , and let  $\approx_{\mathbf{R}}$  be the associated equivalence relation.
- The **quotient by  $\approx_{\mathbf{R}}$**  of  $L$  is the LTS  $L_{\approx}=(S_{\approx},S_{0\approx},\Sigma,T_{\approx},C,\lambda_{\approx})$ :
  - $S_{\approx}$  are the equivalence classes for  $\approx_{\mathbf{R}}$  ;
  - $S_{0\approx}$  are the equivalence classes  $\mathbf{s}$  for  $\approx_{\mathbf{R}}$  such that for all  $s\in\mathbf{s}$ ,  $s\in S_0$  ;
  - $T_{\approx}$  is such that  $T_{\approx}(\mathbf{s}_1,\sigma,\mathbf{s}_2)$  iff  $\exists s_1\in\mathbf{s}_1, s_2\in\mathbf{s}_2:T(s_1,\sigma,s_2)$  ;
  - $\lambda_{\approx}$  is such that  $\lambda_{\approx}(\mathbf{s})=\lambda(s)$  for any  $s\in\mathbf{s}$ .
- **Theorem:**  
Let  $L$  be a LTS and  $\mathbf{R}$  a bisimulation over the state space of  $L$ , let  $L_{\approx}$  be the quotient of  $L$  by  $\approx_{\mathbf{R}}$ , then **Traces**( $L$ )=**Traces**( $L_{\approx}$ ).

# Quotient of a LTS using bisimulation



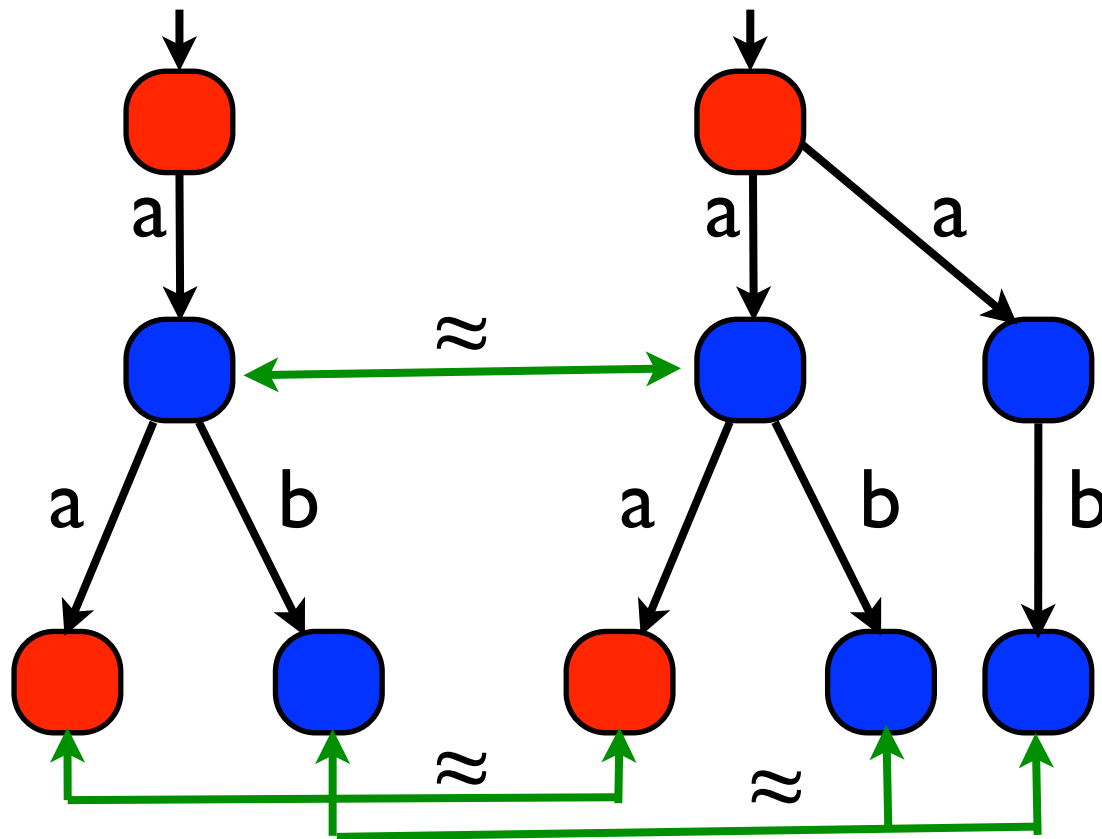
The LTS

# Quotient of a LTS using bisimulation

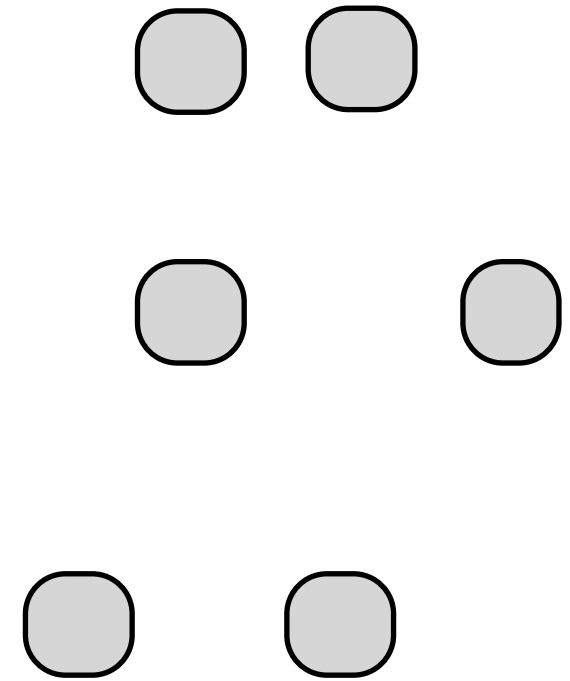


The LTS

# Quotient of a LTS using bisimulation

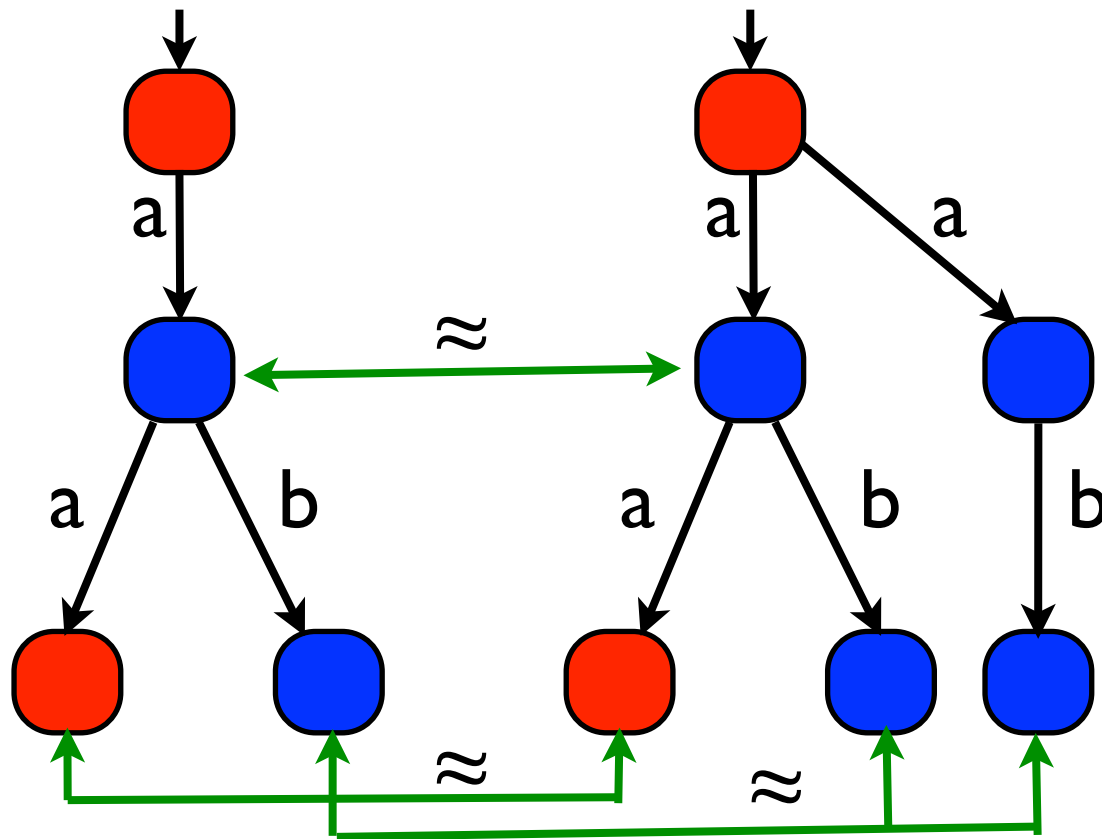


The LTS

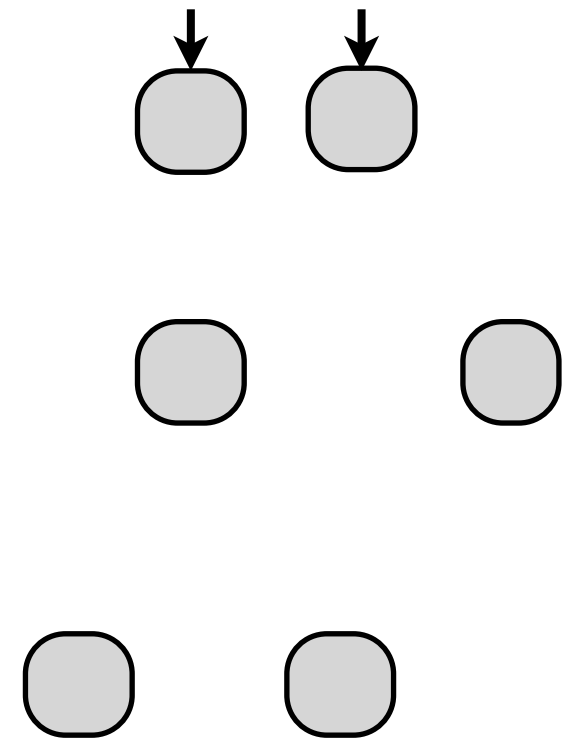


The quotient by  $\approx$

# Quotient of a LTS using bisimulation

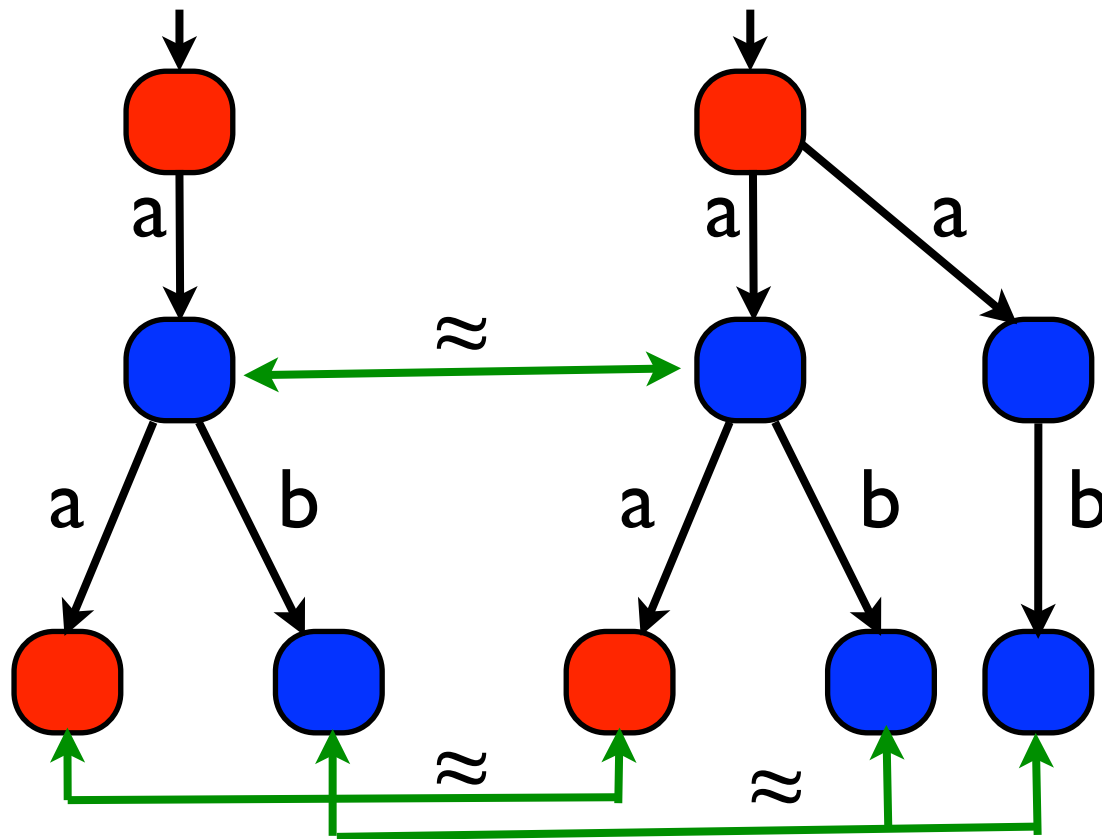


The LTS

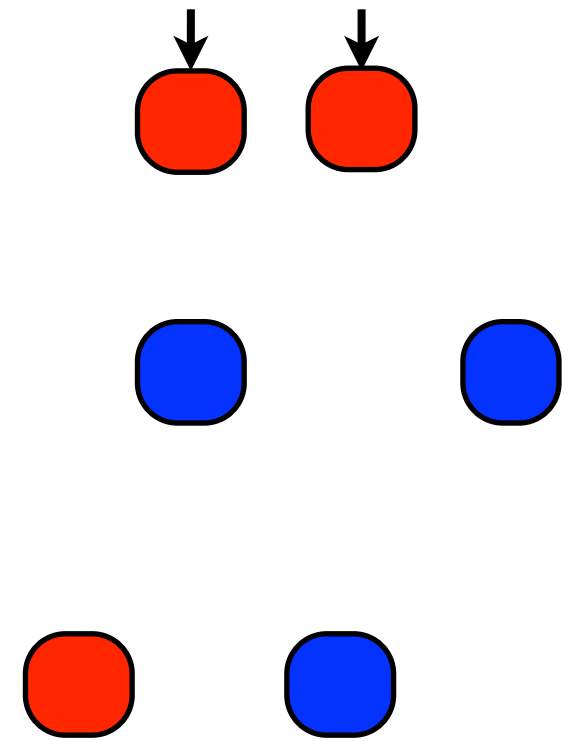


The quotient by  $\approx$

# Quotient of a LTS using bisimulation

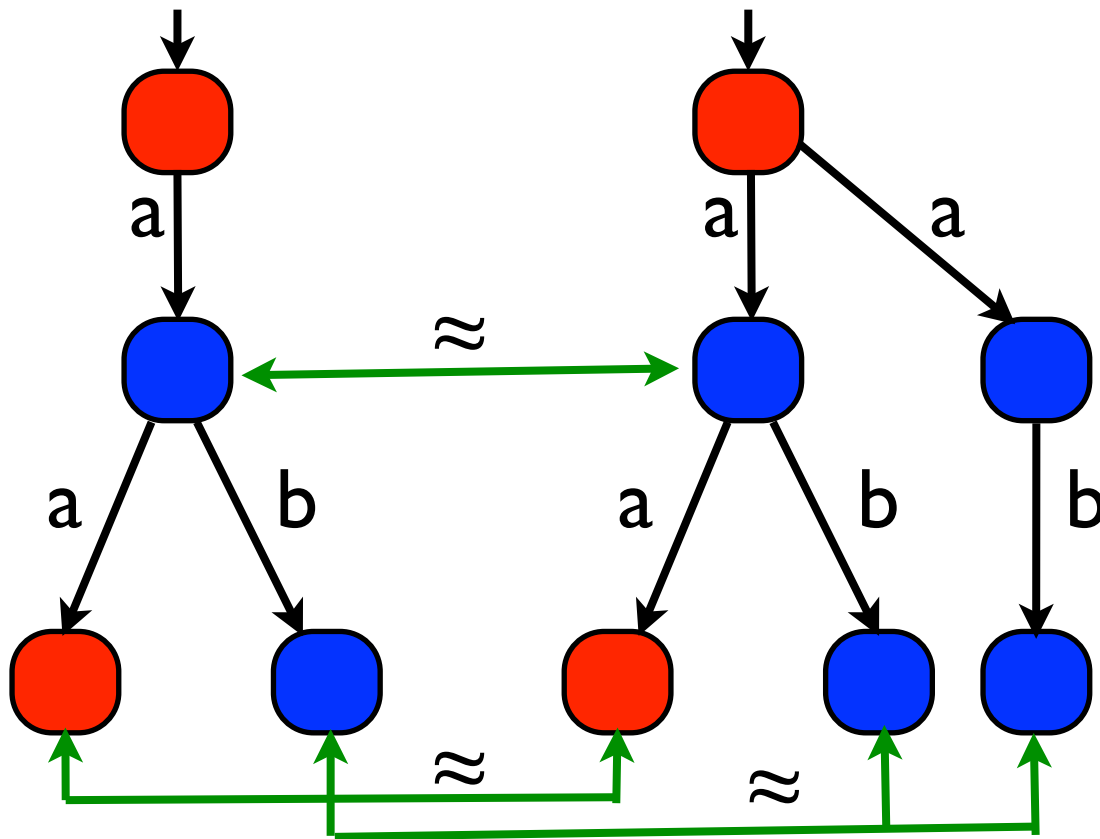


The LTS

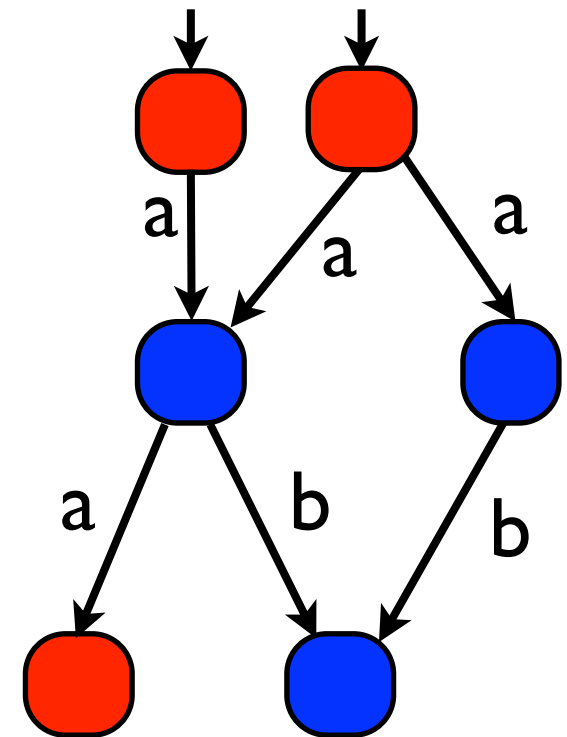


The quotient by  $\approx$

# Quotient of a LTS using bisimulation

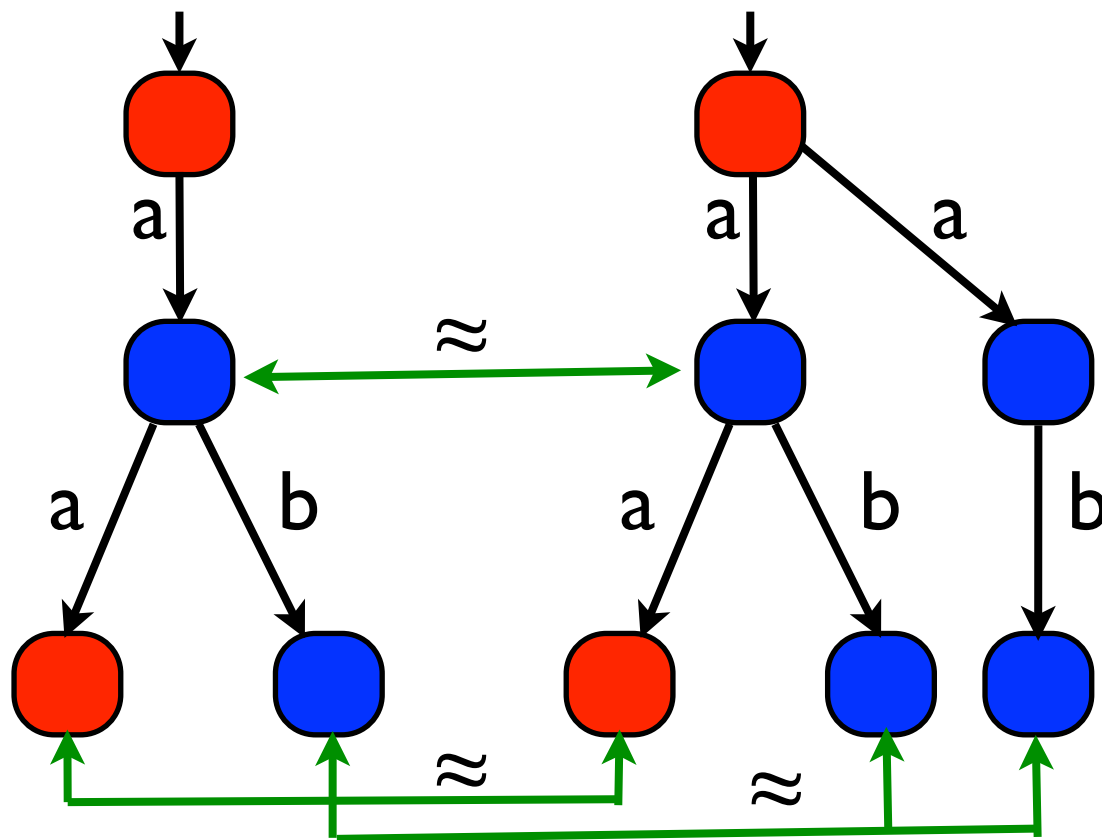


The LTS



The quotient by  $\approx$

# Bisimulation is not complete for trace equivalence



Clearly, the two initial states are trace equivalent but they are not bisimilar.

The LTS

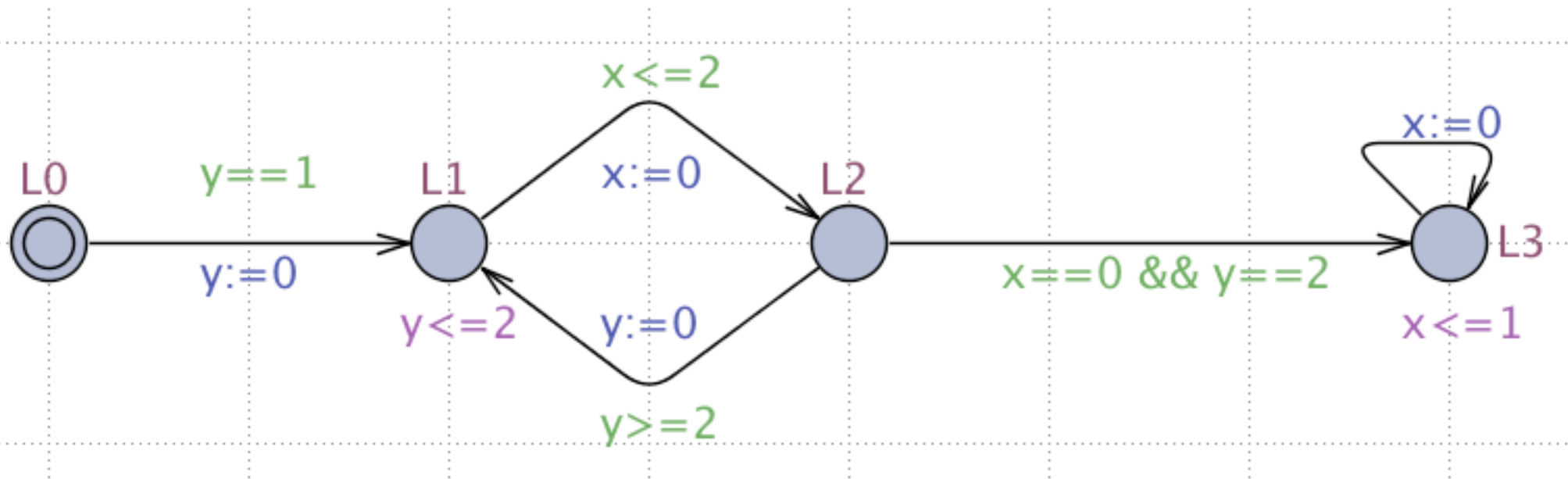
# Plan of the talk

- Labelled transition systems
- Properties of labeled transition systems:  
Reachability - Safety - Büchi properties
- Pre-Post operators
- Partial orders - Fixed points
- Symbolic model-checking
- Application to TA: region equivalence, region automata, zones

# Algorithmic verification of timed automata

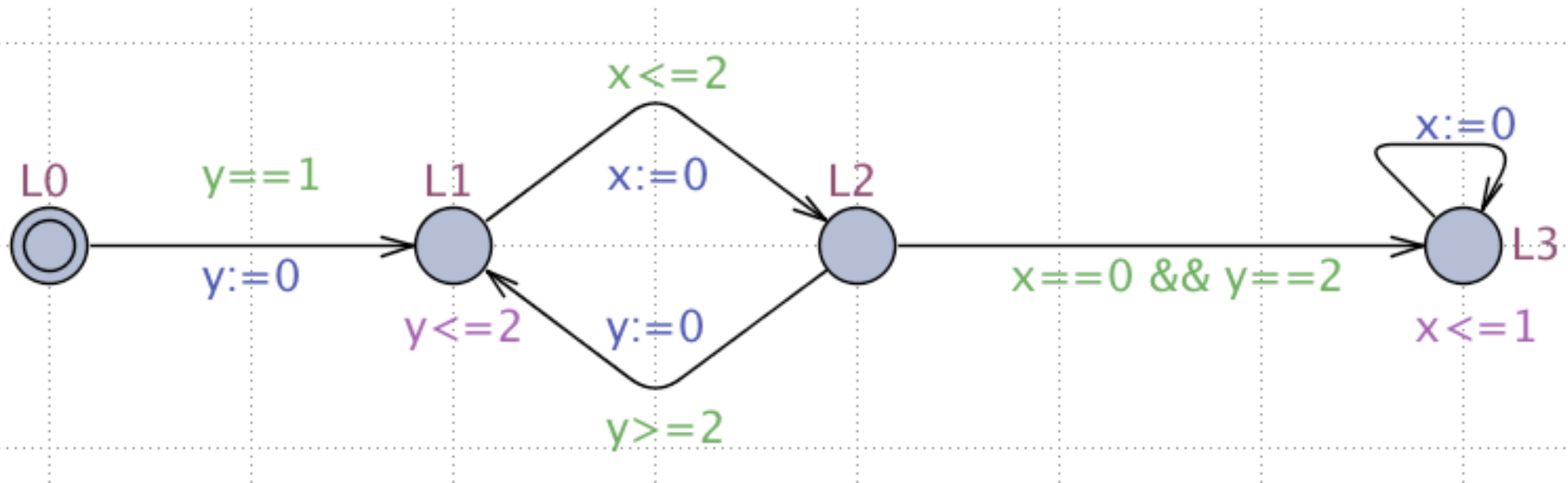
- We will show now how to apply the concepts that we have introduced so far to obtain algorithms to verify properties of timed automata;
- We will show how to use the **pre-post operators** to build **fixed points algorithms**;
- We will show that those algorithms are terminating by showing that they operate over **finite state time-abstract bisimulation quotients**.

# A timed automaton



Question: Can L3 be reached ?

# A timed automaton



Question: Can L3 be reached ?

This question can be reduced to a reachability verification problem over the labeled transition system of the TA.

# Labeled transition system of a TA

- The  $LTS=(S,S_0,\Sigma,T,C,\lambda)$  of a TA  $A=(Q,Q_0,\Sigma,P,CI,E,L,F,Inv)$ , is as follows:
  - $S$  is the set of pairs  $(q,v)$  where  $q \in Q$  is a location of  $A$  and  $v : CI \rightarrow \mathbb{R}_{\geq 0}$  such that  $v \models Inv(q)$ ;
  - $S_0 = \{(q_0, \langle 0, 0, \dots, 0, \rangle) \mid q_0 \in Q_0\}$ ;
  - $T \subseteq S \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times S$  defined by two types of transitions:
    - Discrete transitions:  
 $(q_1, v_1) \rightarrow_a (q_2, v_2) \in T$  iff there exists  $(q_1, a, \Phi, \Delta, q_2) \in E$ ,  $v_1 \models \Phi$ , and  $v_2 = v_1[\Delta := 0]$ .
    - Continuous transitions:  
 $(q_1, v_1) \rightarrow_\delta (q_2, v_2) \in T$  iff  $q_1 = q_2$ ,  $\delta \in \mathbb{R}_{\geq 0}$ ,  $v_2 = v_1 + \delta$ , and  $\forall \delta', 0 \leq \delta' \leq \delta, v_1 + \delta' \models Inv(q_1)$ .
  - $C = 2^P$ ,  $\lambda((q,v)) = L(q)$ , for any  $(q,v) \in Q$ .
- Clearly, this transition system has a (continuous) infinite number of states. How do we handle it ?

# Time abstract-labeled transition system

- The  $LTS=(S,S_0,\Sigma,T,C,\lambda)$  of a TA  $A=(Q,Q_0,\Sigma,P,CI,E,L,F,Inv)$ , is as follows:

- $S$  is the set of pairs  $(q,v)$  where  $q \in Q$  is a location of  $A$  and  $v : CI \rightarrow \mathbb{R}_{\geq 0}$  such that  $v \models Inv(q)$ ;
- $S_0 = \{(q_0, \langle 0, 0, \dots, 0, \rangle) \mid q_0 \in Q_0\}$ ;
- $T \subseteq S \times (\Sigma \cup \{\text{Delay}\}) \times S$  defined by two types of transitions:

Discrete transitions:

$(q_1, v_1) \rightarrow_a (q_2, v_2) \in T$  iff there exists  $(q_1, a, \Phi, \Delta, q_2) \in E$ ,  $v_1 \models \Phi$ , and  $v_2 := v_1[\Delta := 0]$ .

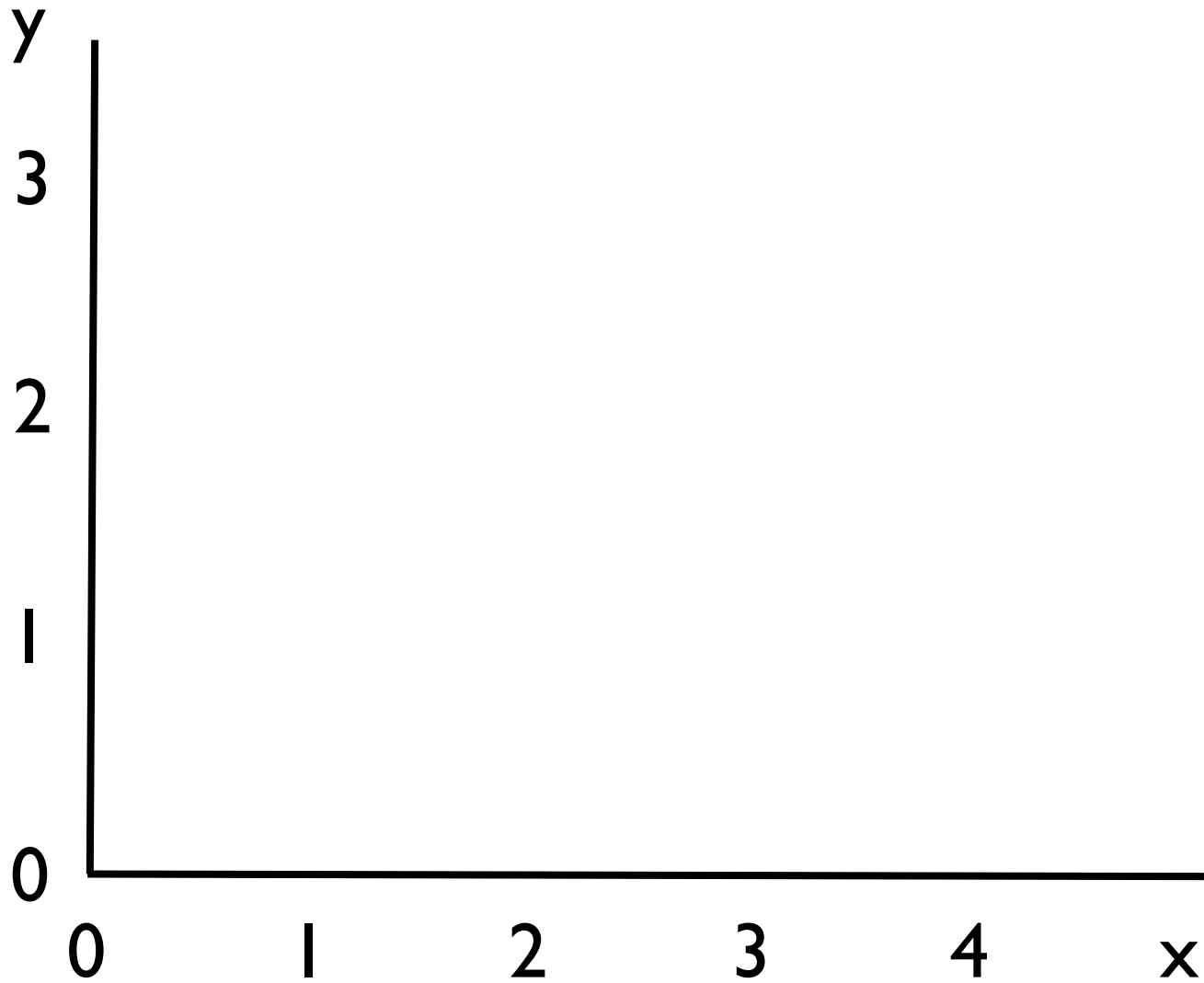
Continuous transitions:

$(q_1, v_1) \rightarrow_{\text{Delay}} (q_2, v_2) \in T$  iff  $q_1 = q_2$ ,  $\exists \delta \in \mathbb{R}_{\geq 0}$ ,  $v_2 = v_1 + \delta$ , and  $\forall \delta', 0 \leq \delta' \leq \delta, v_1 + \delta' \models Inv(q_1)$ .

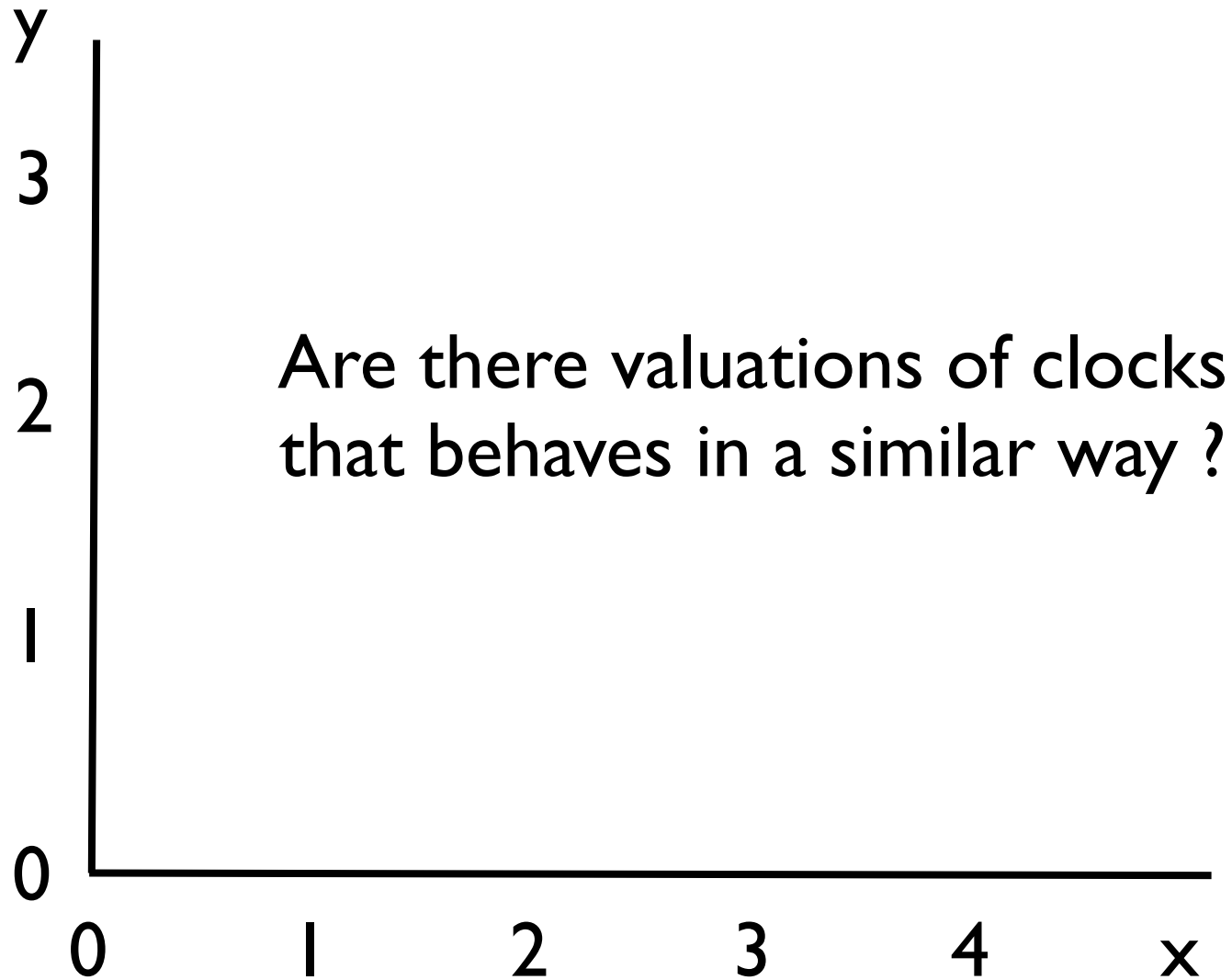
- $C = 2^P$ ,  $\lambda((q,v)) = L(q)$ , for any  $(q,v) \in Q$ .

- Clearly, this transition system has a (continuous) infinite number of states. How do we handle it ?

# Continuous state space

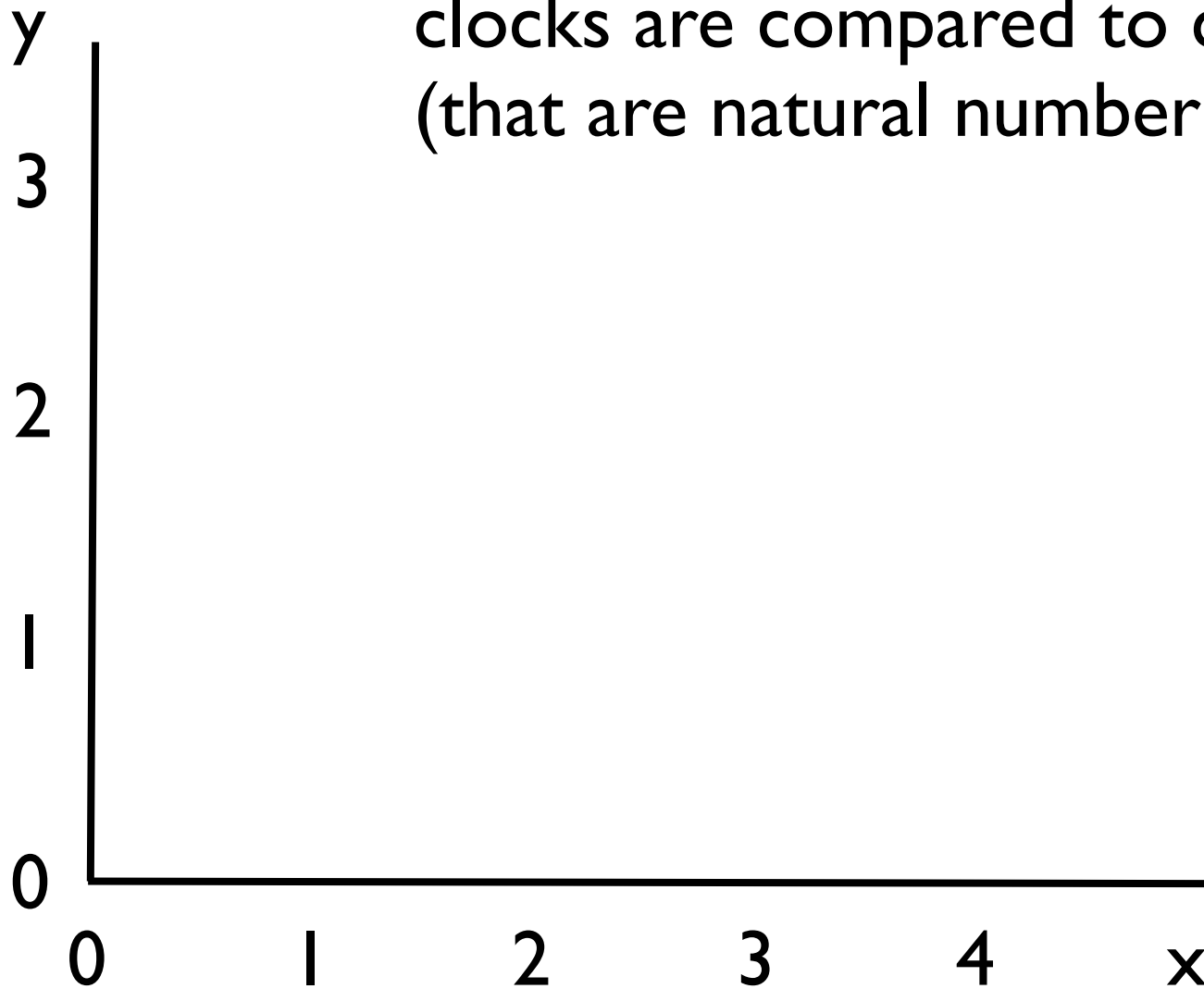


# Continuous state space



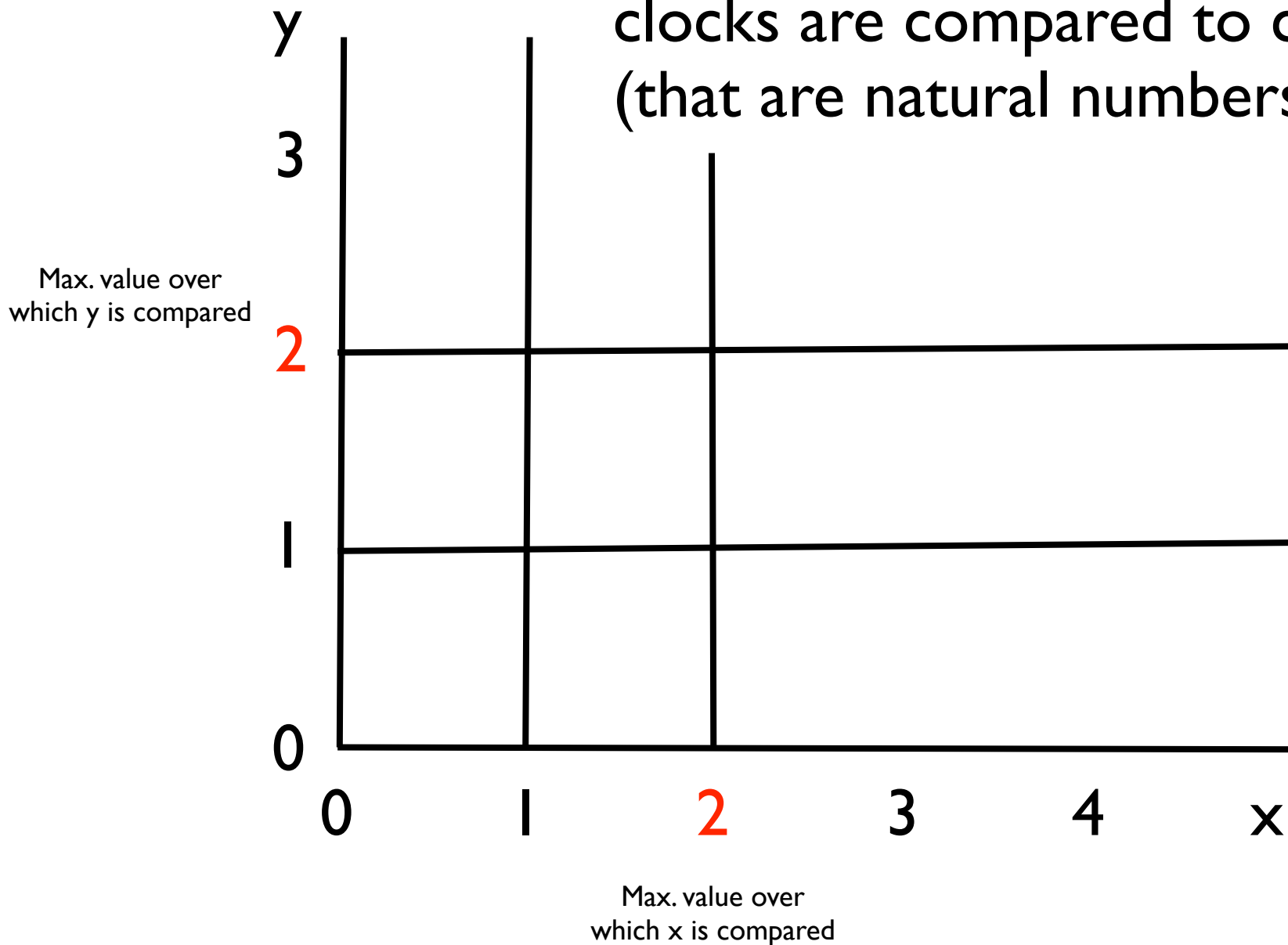
# Continuous state space

First, let us note that clocks are compared to constants (that are natural numbers).



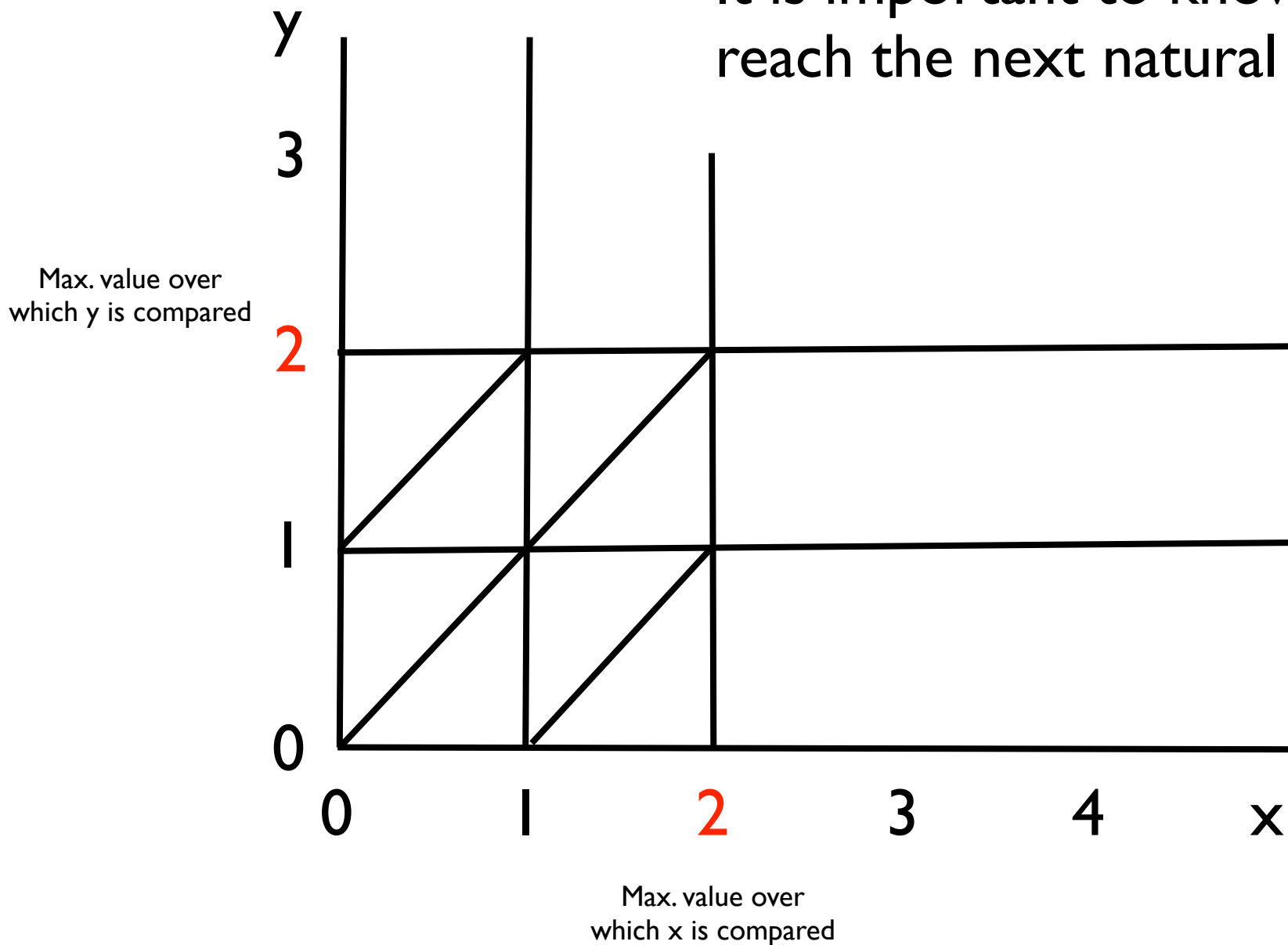
# Continuous state space

First, let us note that clocks are compared to constants (that are natural numbers).

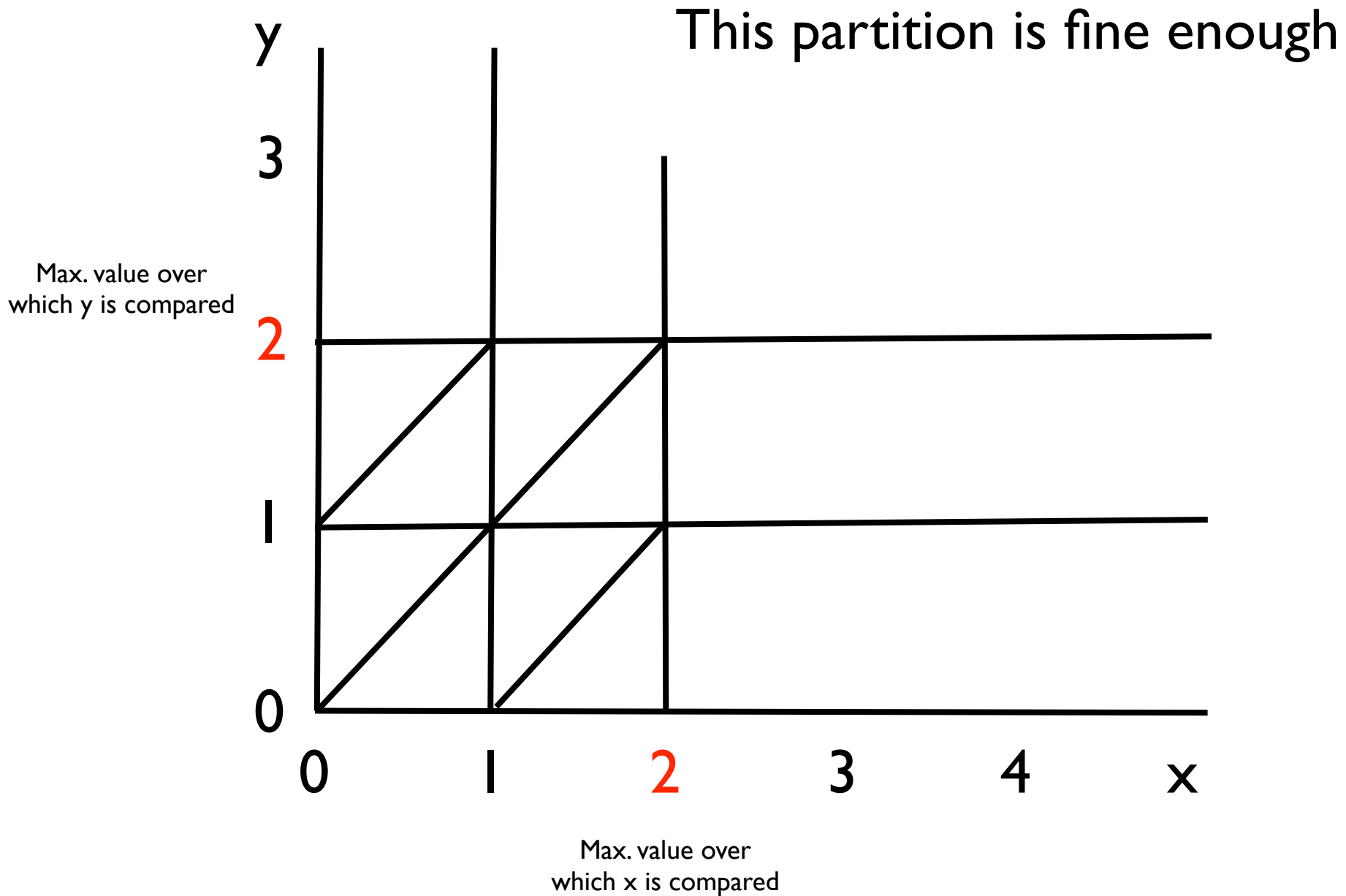


# Continuous state space

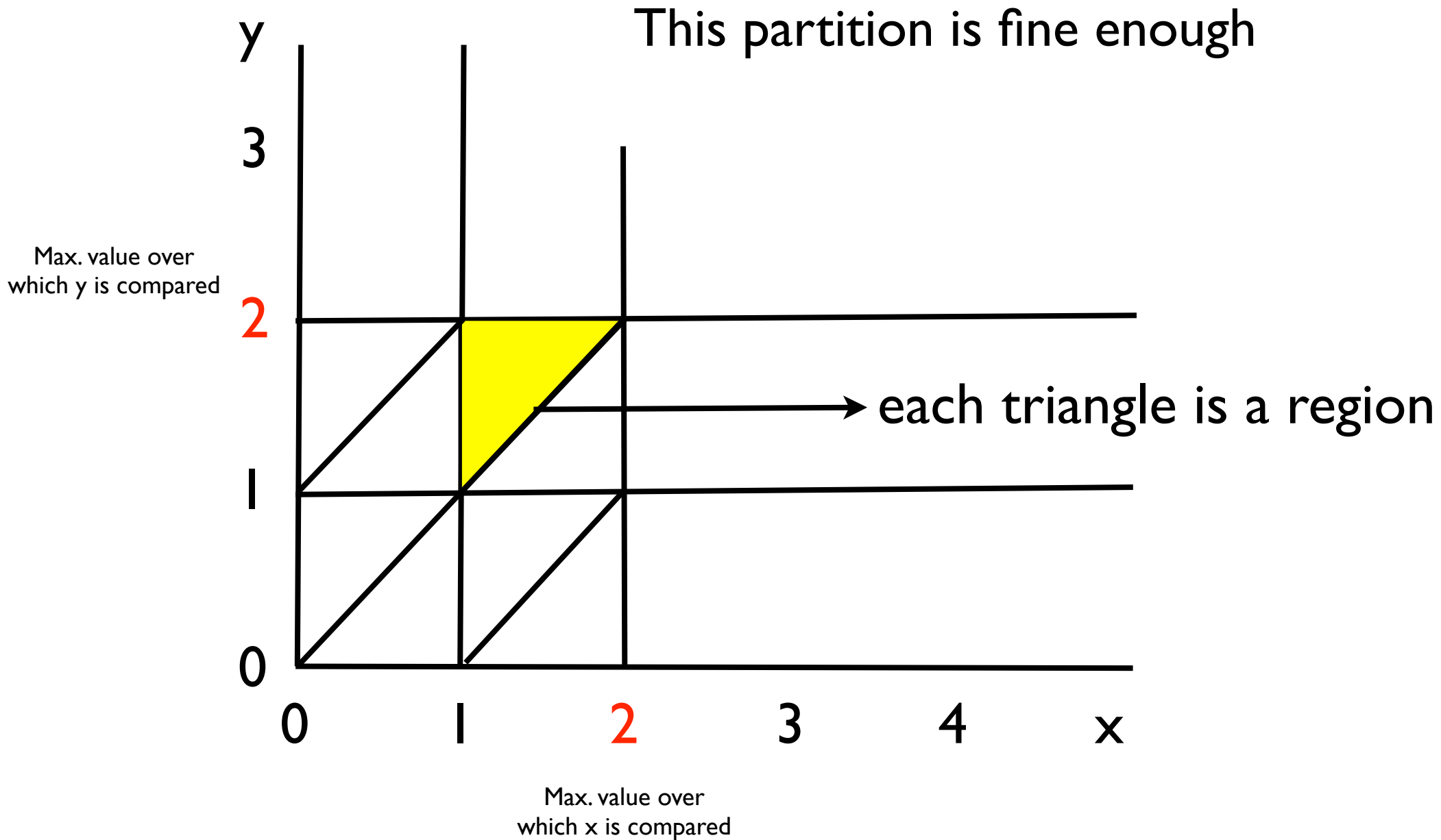
It is important to know if  $x$  or  $y$  reach the next natural number first.



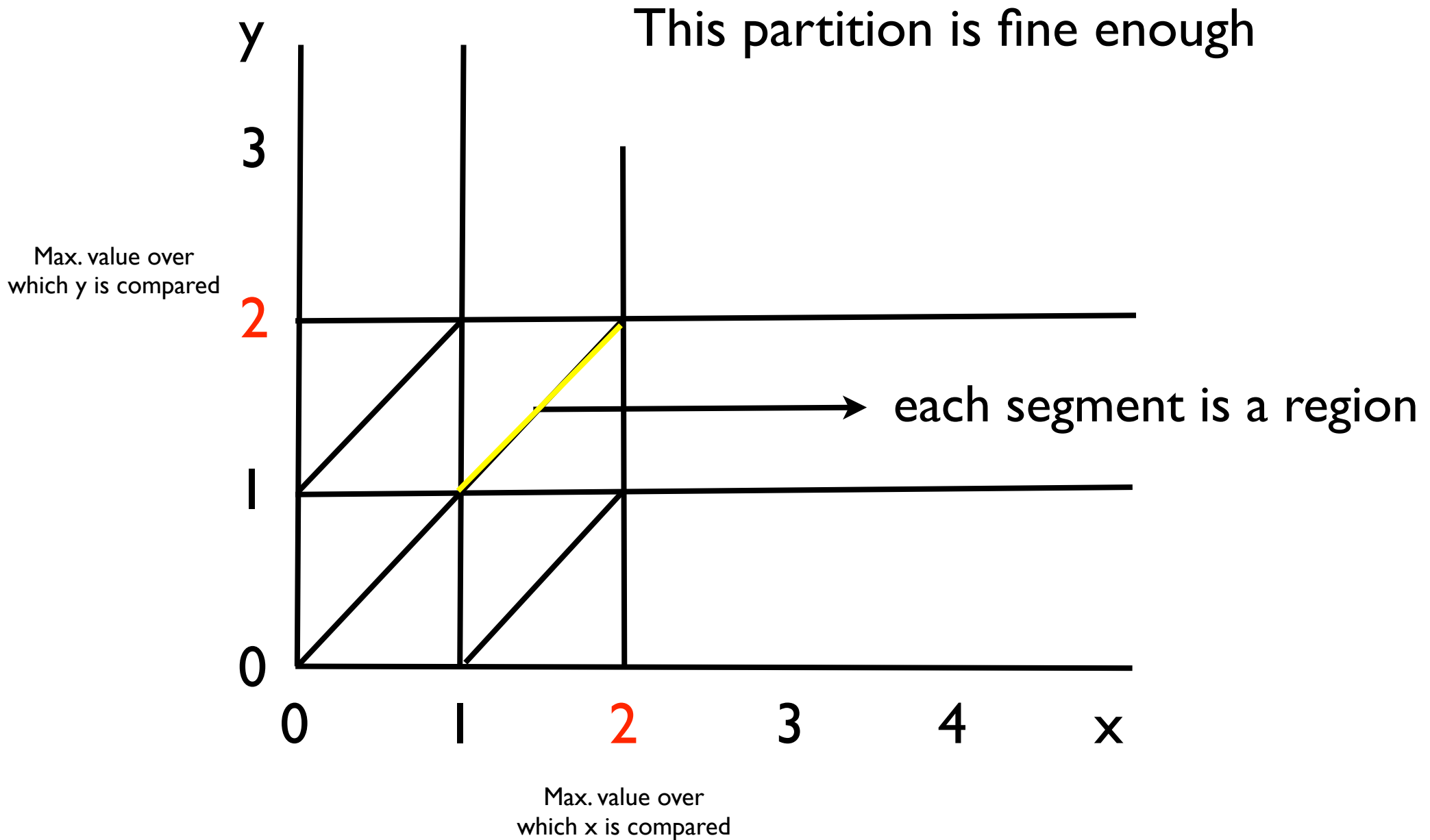
# Continuous state space



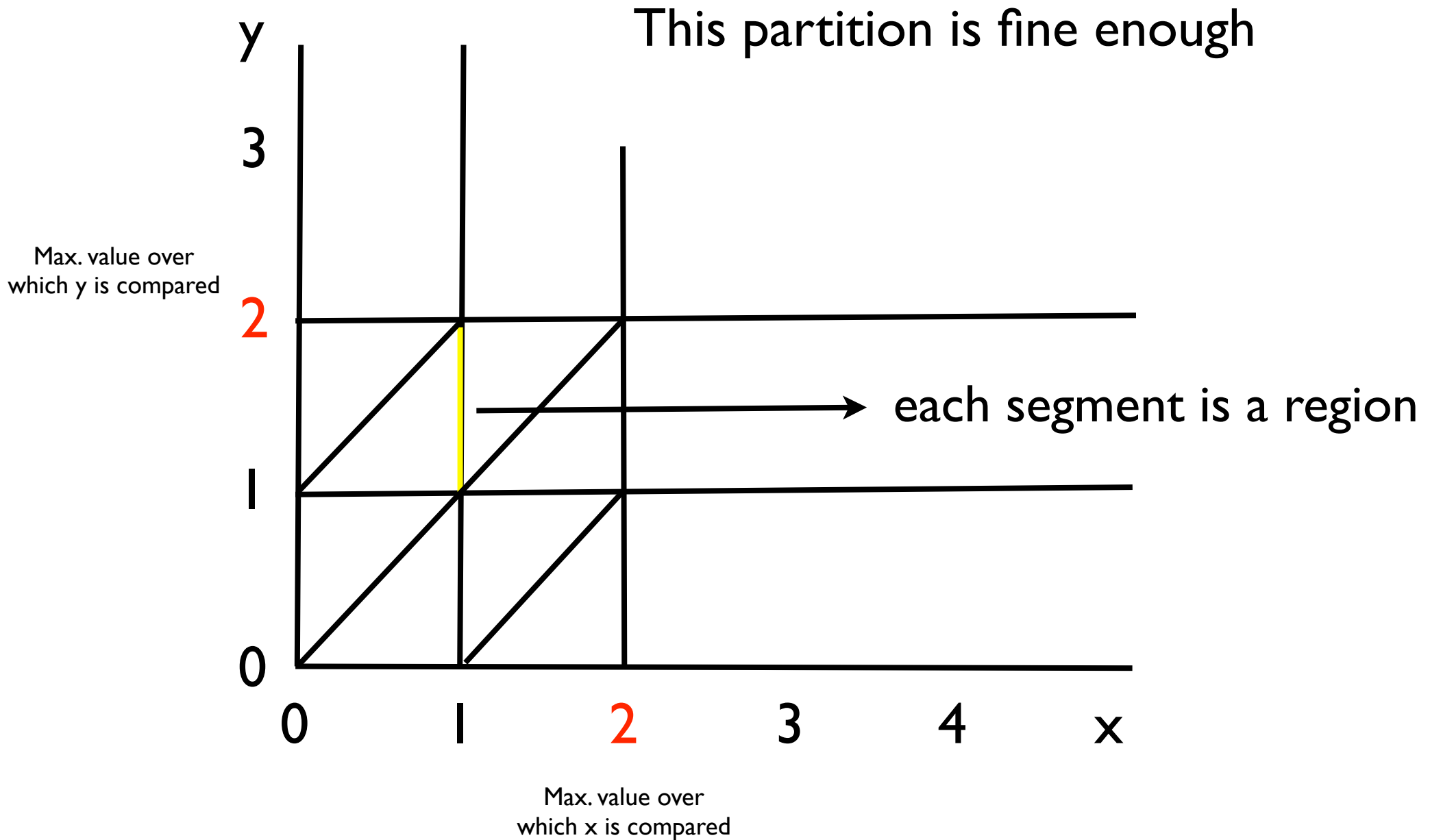
# Continuous state space



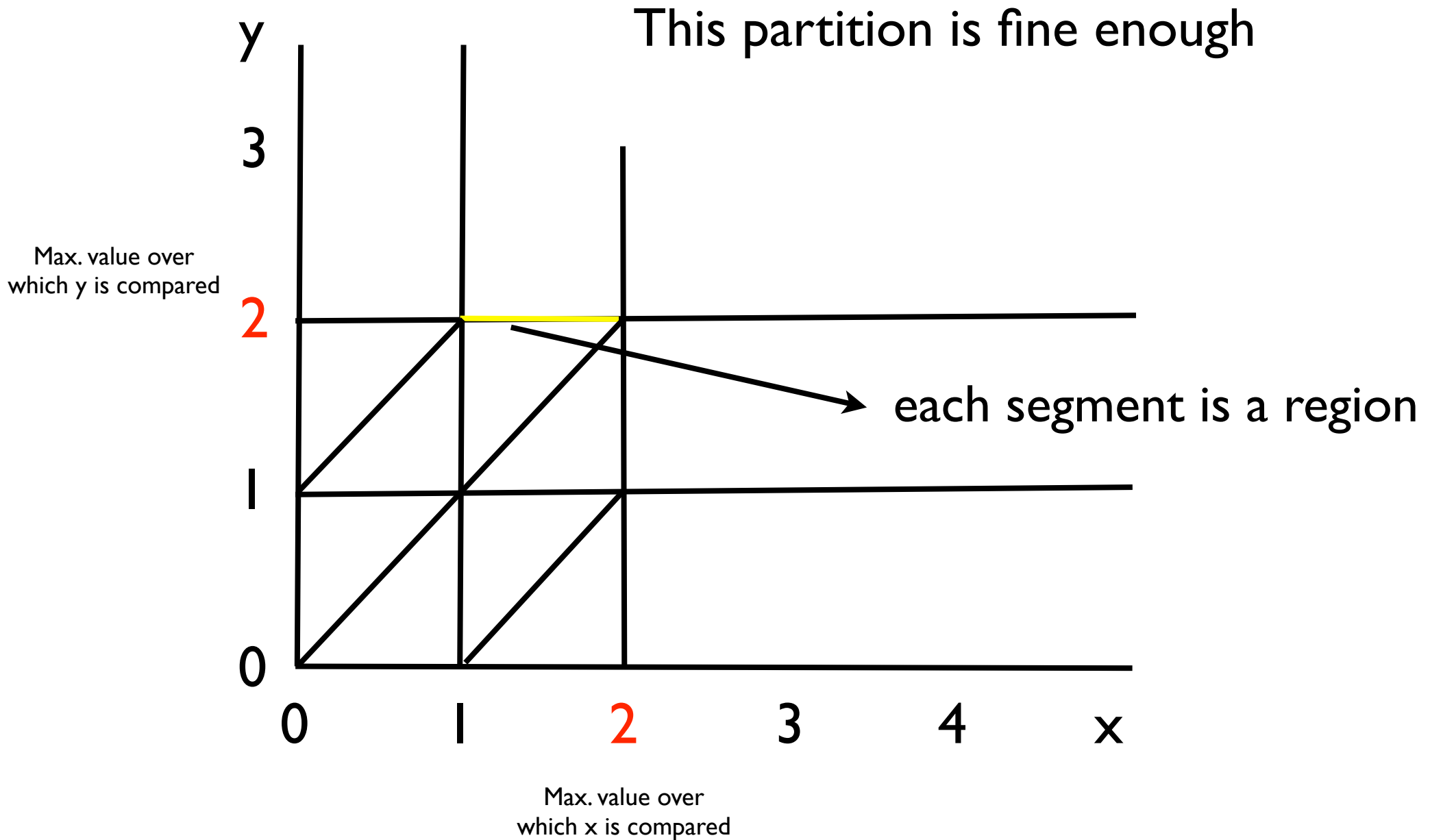
# Continuous state space



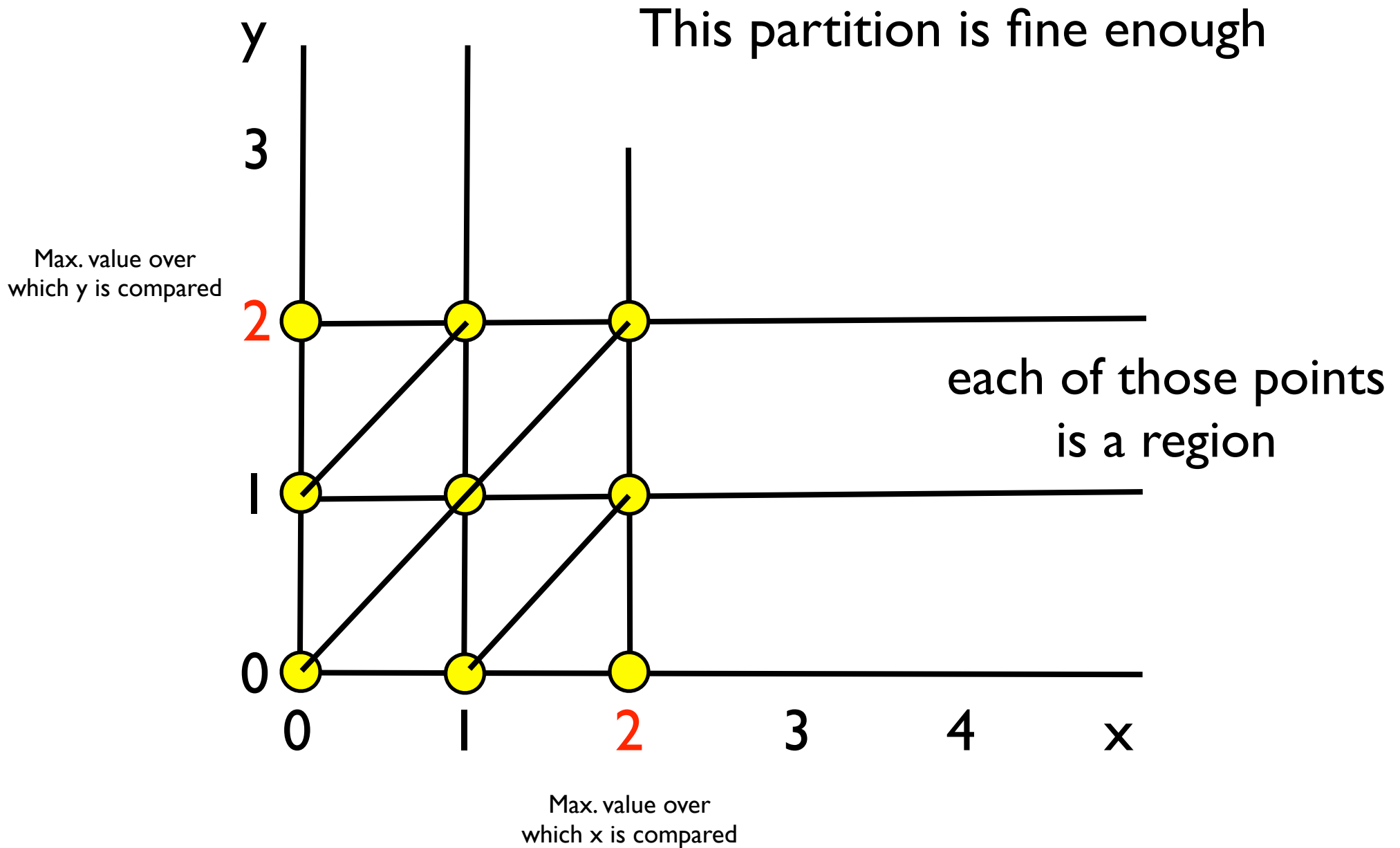
# Continuous state space



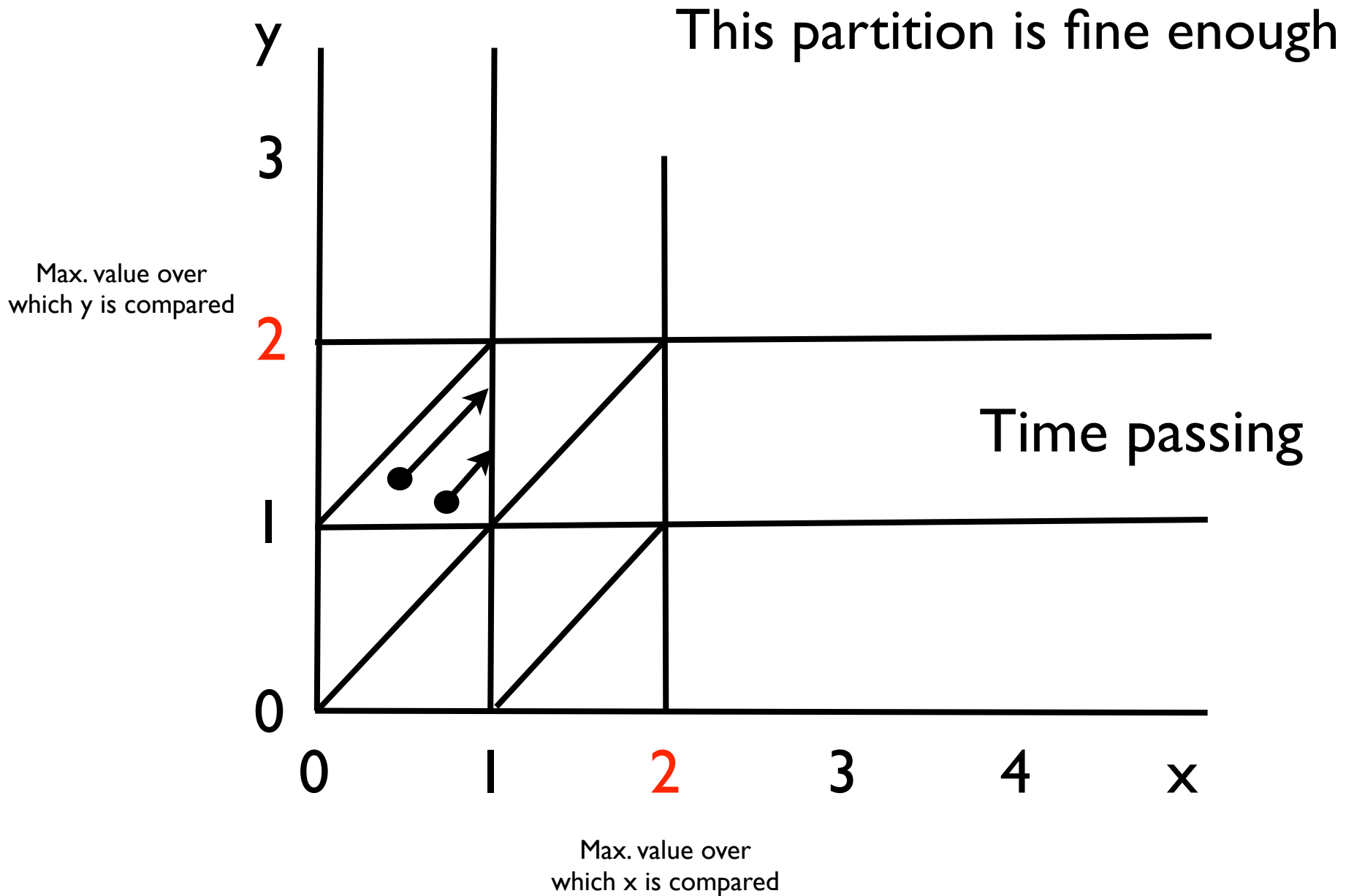
# Continuous state space



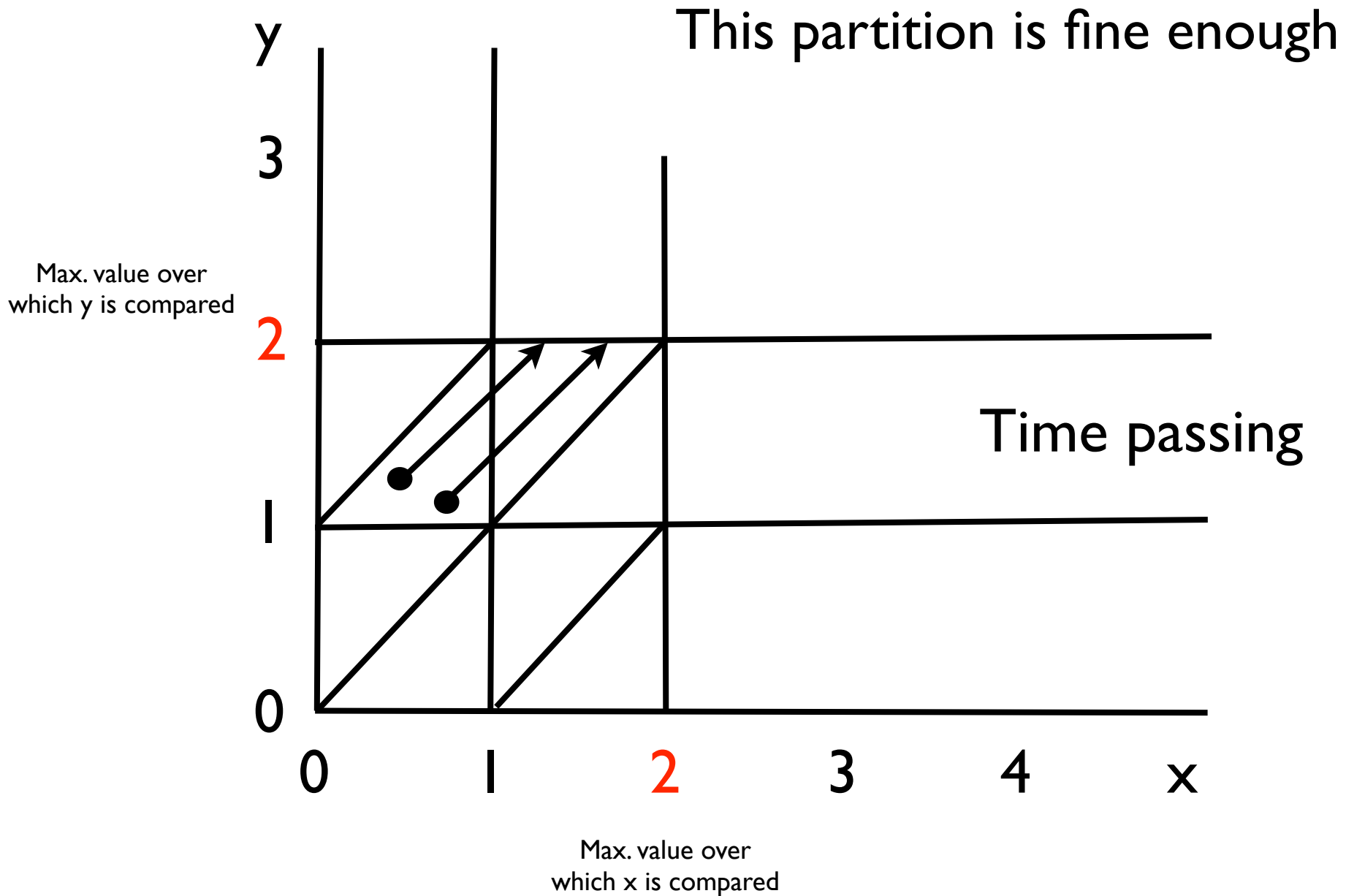
# Continuous state space



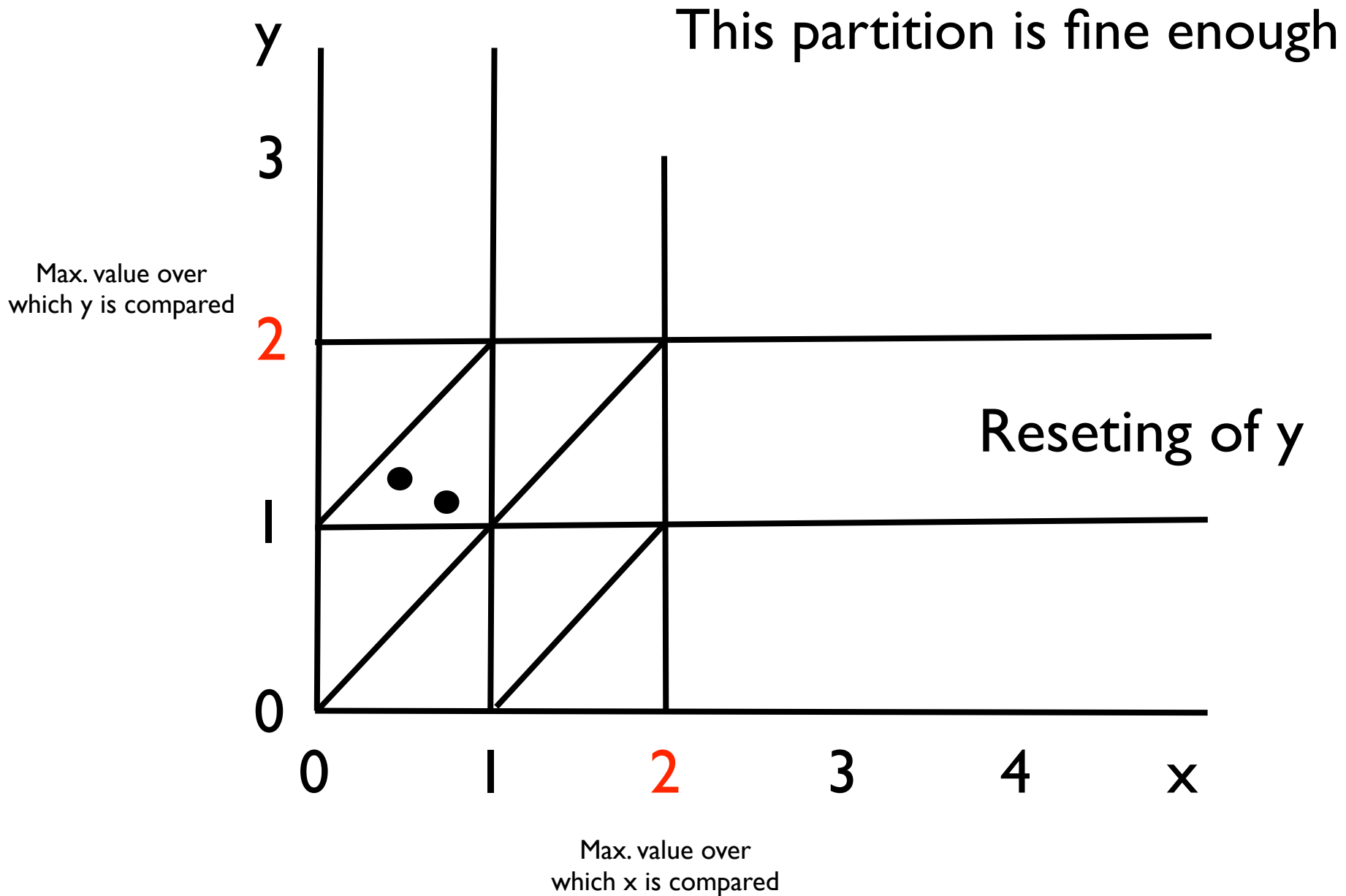
# Continuous state space



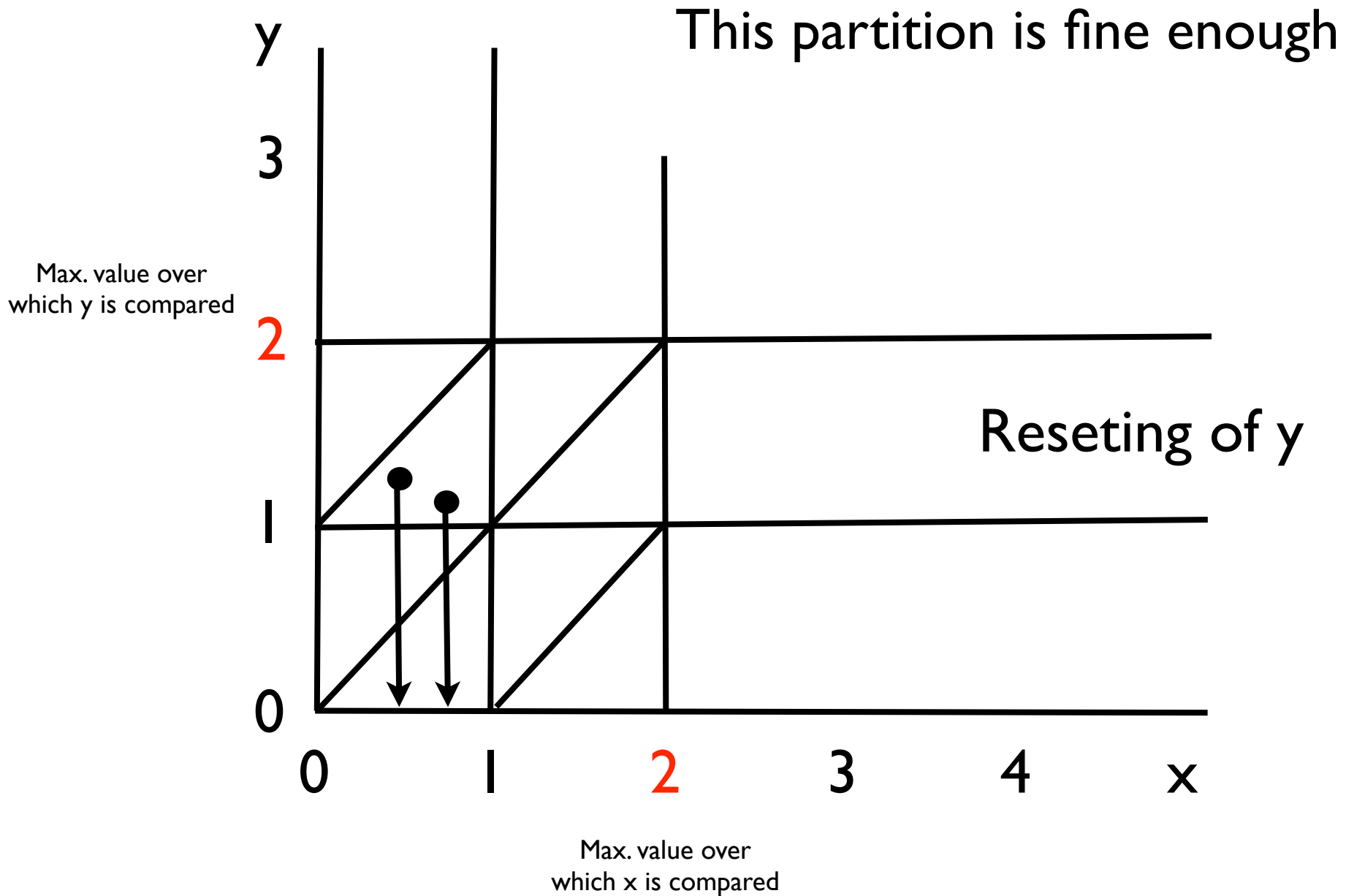
# Continuous state space



# Continuous state space

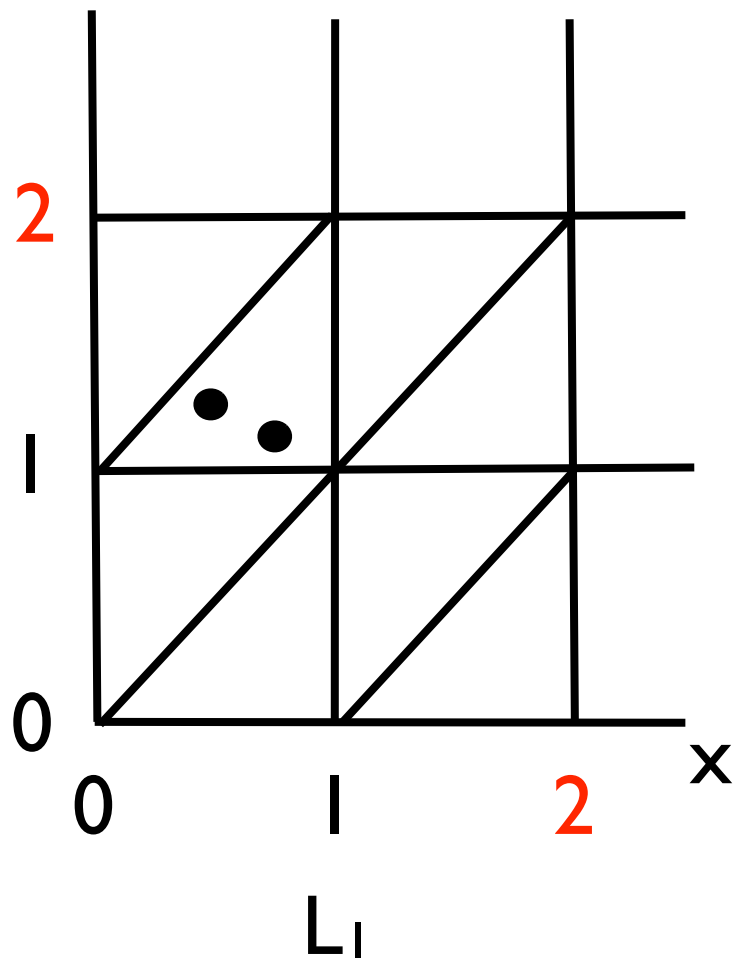


# Continuous state space

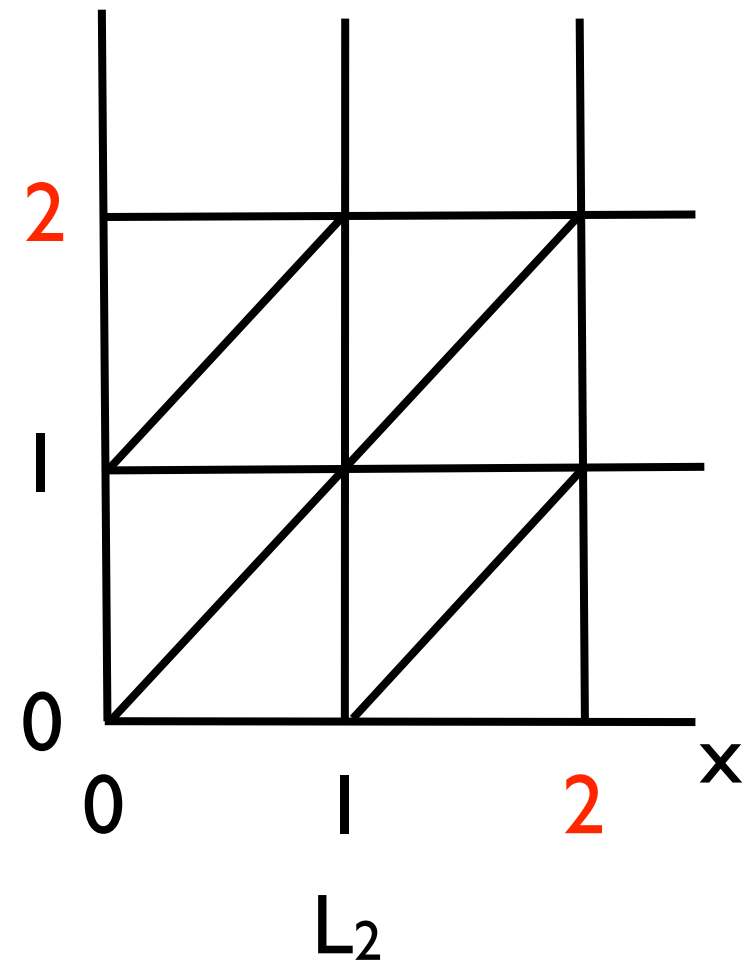


# Continuous state space

This partition is fine enough

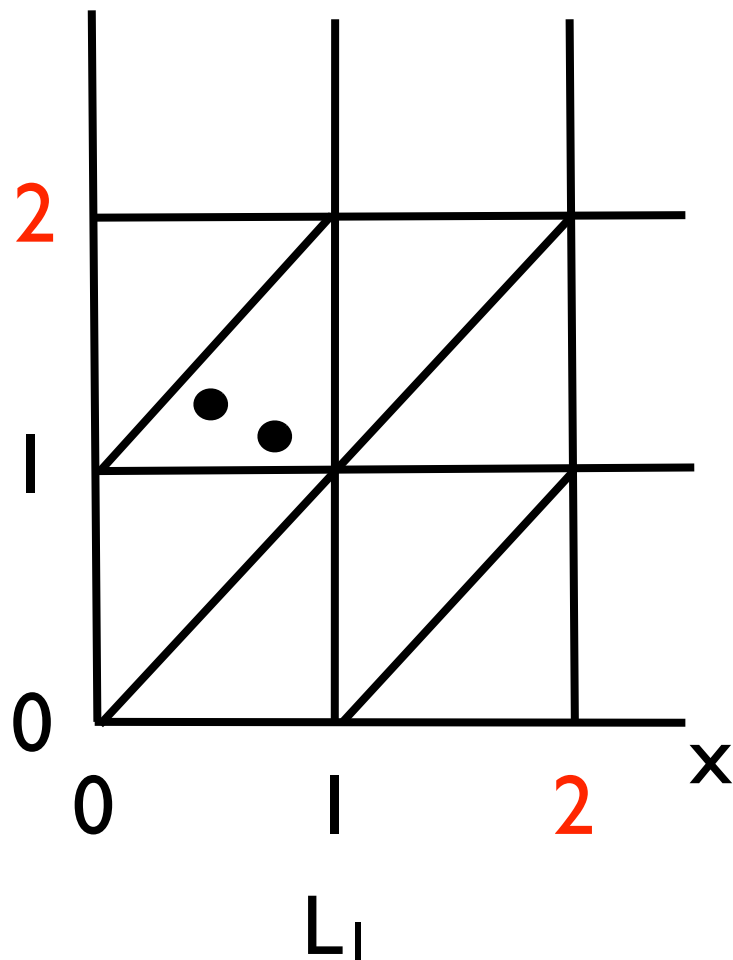


discrete transition  
from  $L_1$  to  $L_2$

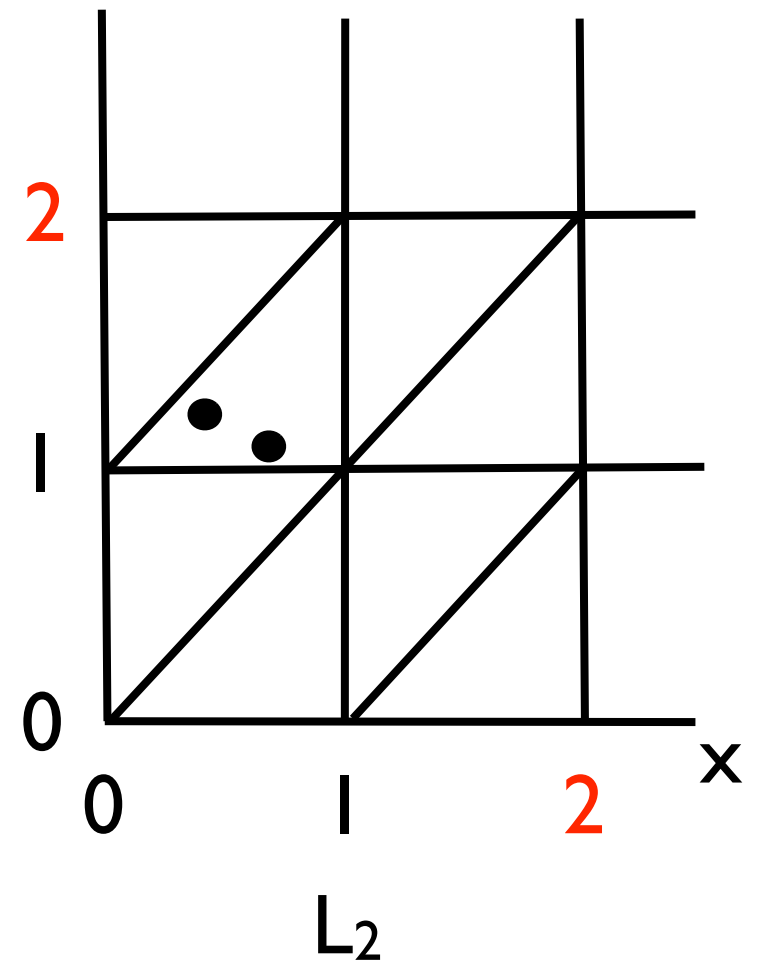


# Continuous state space

This partition is fine enough

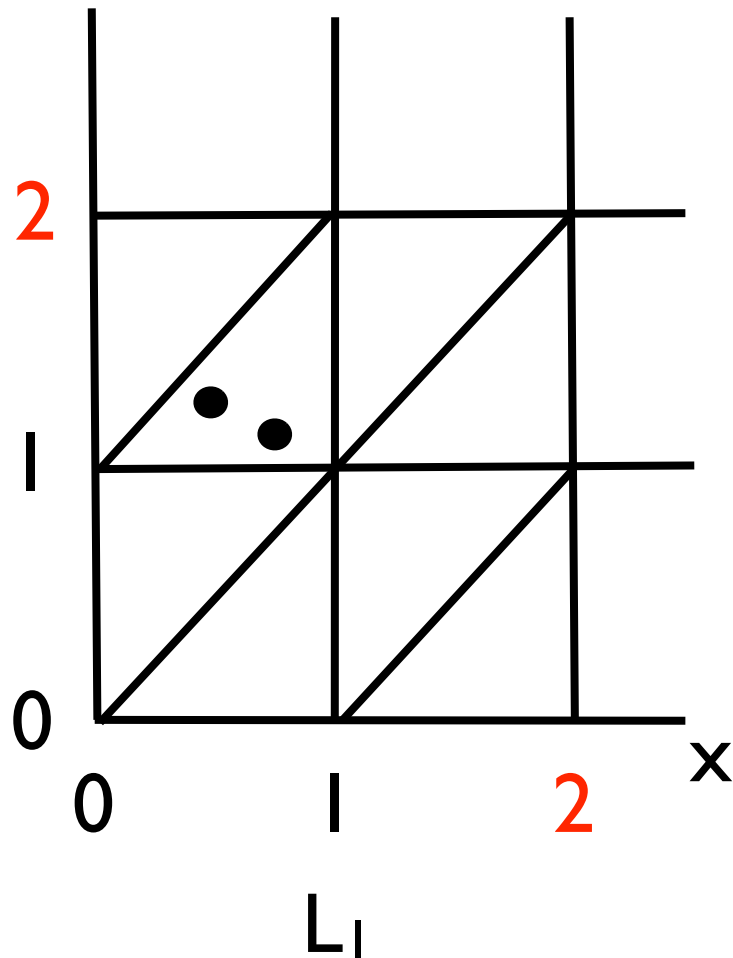


discrete transition  
from  $L_1$  to  $L_2$

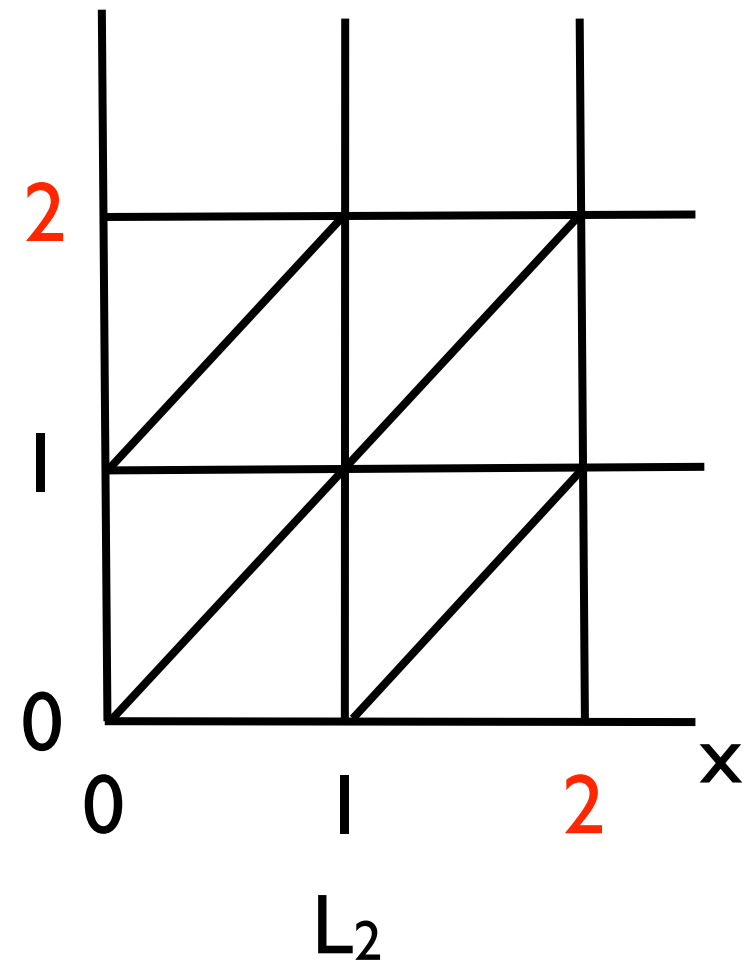


# Continuous state space

This partition is fine enough

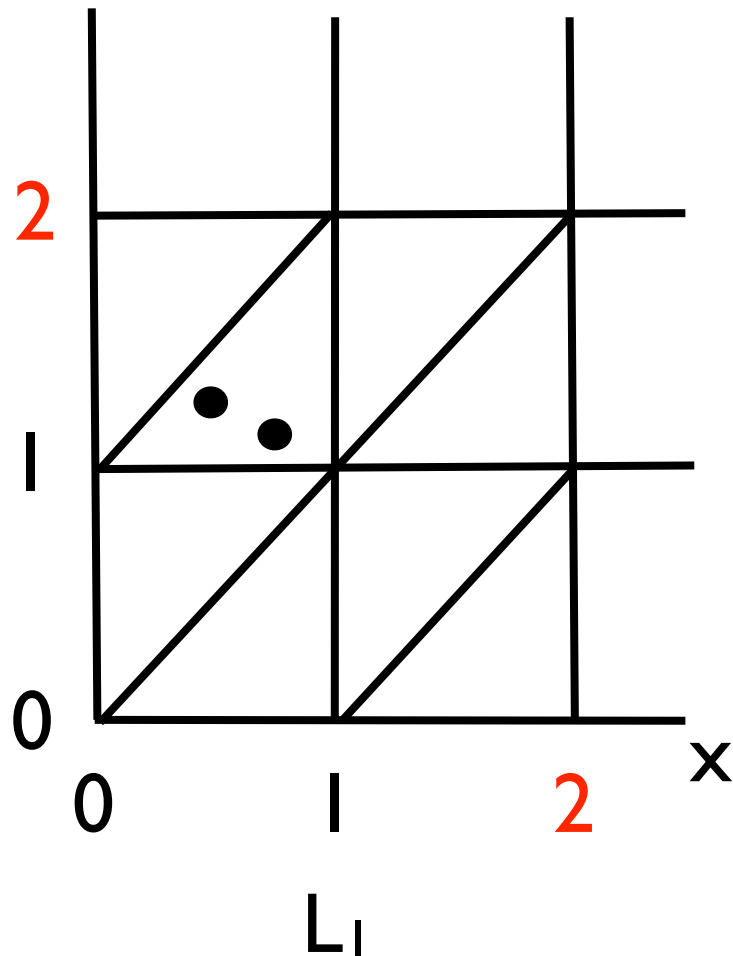


discrete transition  
from  $L_1$  to  $L_2$  with  
reset of  $y$

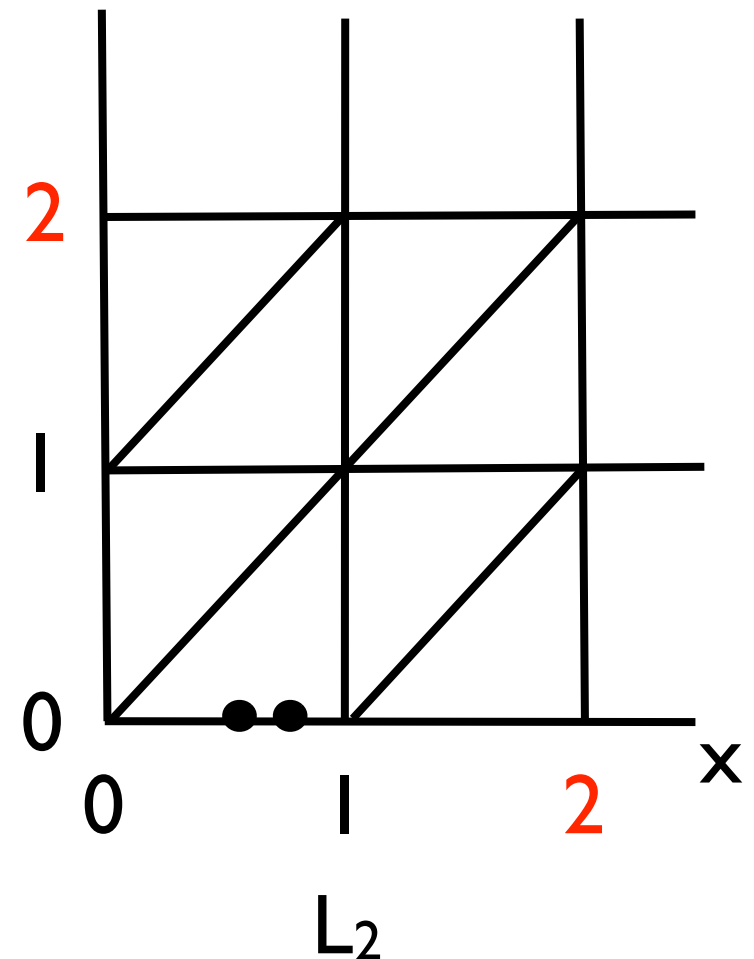


# Continuous state space

This partition is fine enough



discrete transition  
from  $L_1$  to  $L_2$  with  
reset of  $y$



# Region equivalence: formal definition

- For each variable  $x \in \text{Cl}$ , let  $c_x$  be the largest constant with which  $x$  is compared in the TA.

Two valuations  $v_1, v_2: \text{Cl} \rightarrow \mathbb{R}_{\geq 0}$  are **region equivalent**, noted  $v_1 \approx v_2$  **iff**

- **same integer parts:**  
for all  $x \in \text{Cl}$ ,  $\text{int}(v_1(x)) = \text{int}(v_2(x))$ , or  $v_1(x) > c_x$  and  $v_2(x) > c_x$ .
- **same fractional ordering:**  
for all  $x, y \in \text{Cl}$  with  $v_1(x) \leq c_x$  and  $v_1(y) \leq c_y$ ,  
 $\text{frac}(v_1(x)) \leq \text{frac}(v_1(y))$  iff  $\text{frac}(v_2(x)) \leq \text{frac}(v_2(y))$
- **same null fractional parts:**  
for all  $x, y \in \text{Cl}$  with  $v_1(x) \leq c_x$  and  $v_1(y) \leq c_y$ ,  
 $\text{frac}(v_1(x)) = 0$  iff  $\text{frac}(v_2(x)) = 0$
- **Theorem: a Region** is a set of valuations that are **time abstract bisimilar**.

# Region equivalence quotient of the time-abstract LTS

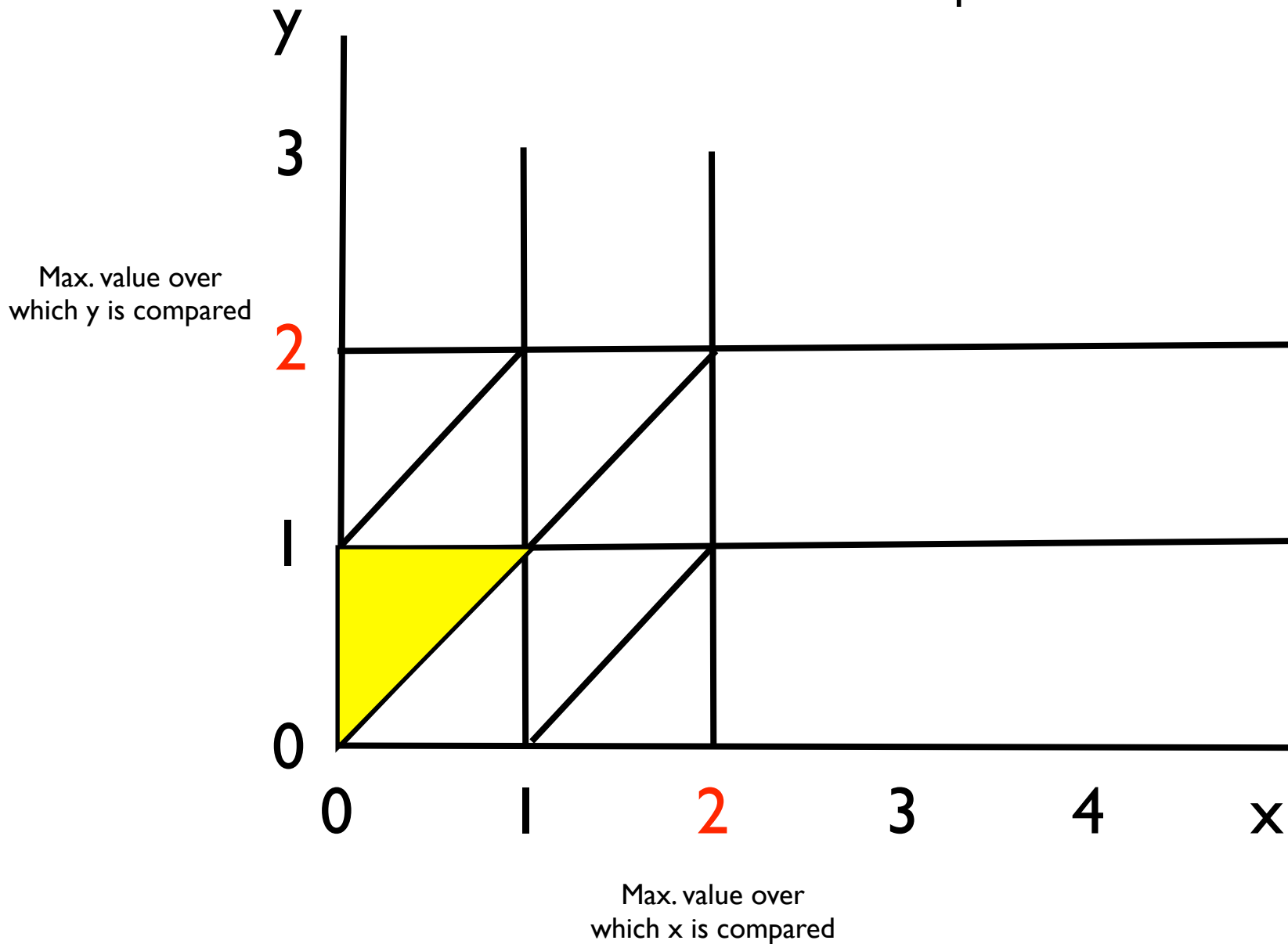
- The following theorem is the **foundation** for the automatic verification of timed automata.
- **Theorem.** Let  $A$  be a timed automaton, let  $L$  be its time-abstract labeled transition system, let  $L_{\approx}$  be its quotient by the region equivalence  $\approx$ , then  $L_{\approx}$  is **finite** and  $L_{\approx}$  is **trace equivalent** to  $L$ .

# Post operations in the time abstract LTS

- To construct **“region based” bisimulation quotient** of the time-abstract LTS of a TA (or to compute on it), we must be able to compute the transition relation between regions.
- We consider the two types of transitions that we find in the time-abstract LTS of a TA:
  - **Discrete transitions** that are associated to transition edges in the timed automaton. Let  $(q_1, a, \Phi, \Delta, q_2) \in E$ :
    - (1) Note that given a region  $r$  and a guard  $\Phi$ , all valuations  $v_1, v_2 \in r$  is such that  $v_1 \models \Phi$  iff  $v_2 \models \Phi$ .
    - (2) The effect of resetting a clock on a region  $r$  gives a region  $r'$ .
  - **Delay transitions.** Given a region  $r$ , we can compute the set of regions  $r'$  that contains  $v+t$  for some  $v \in r$  and some  $t \in \mathbb{R}$ .

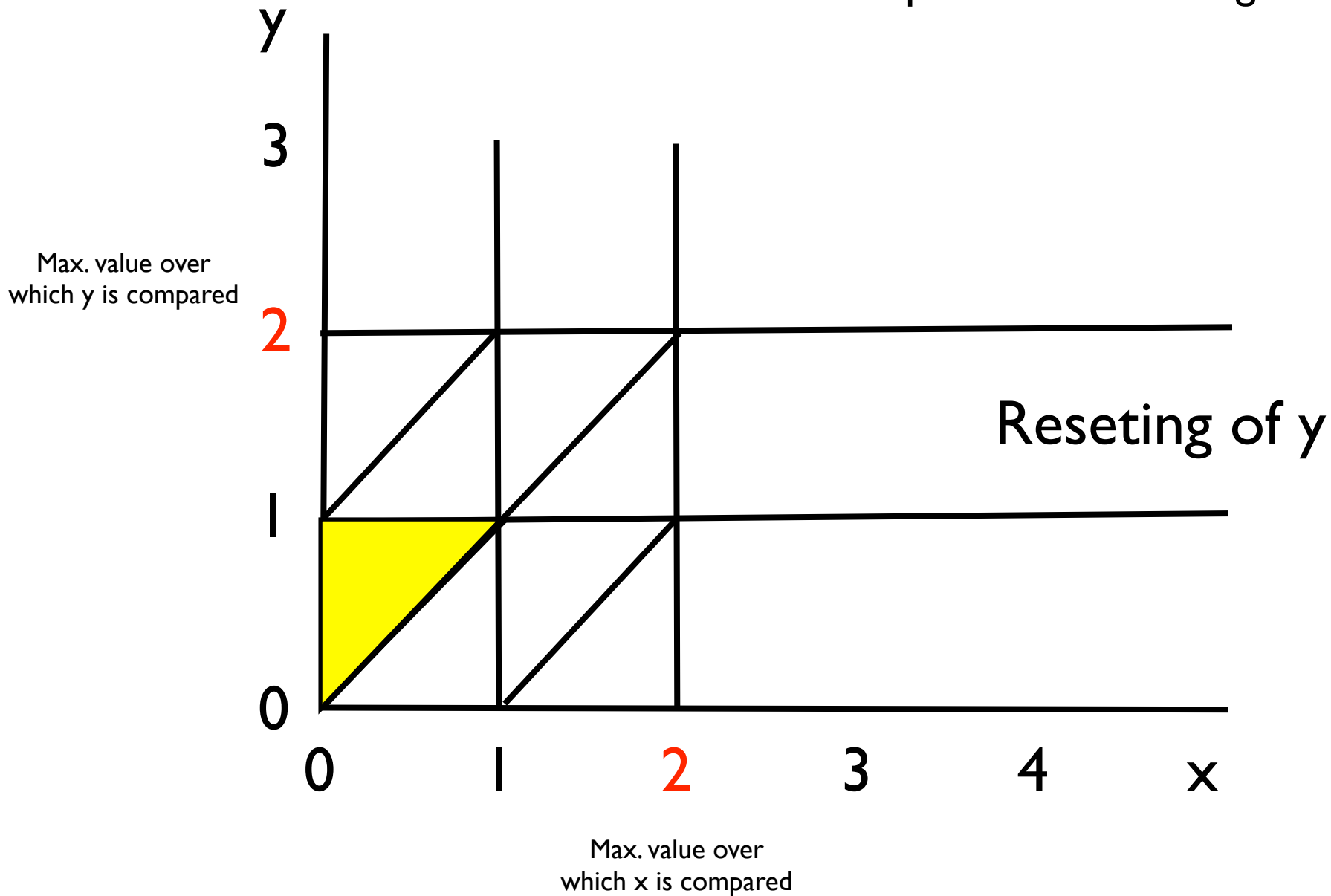
# Continuous state space

The transitions of the time-abstract transition system  
can be rephrased on the regions



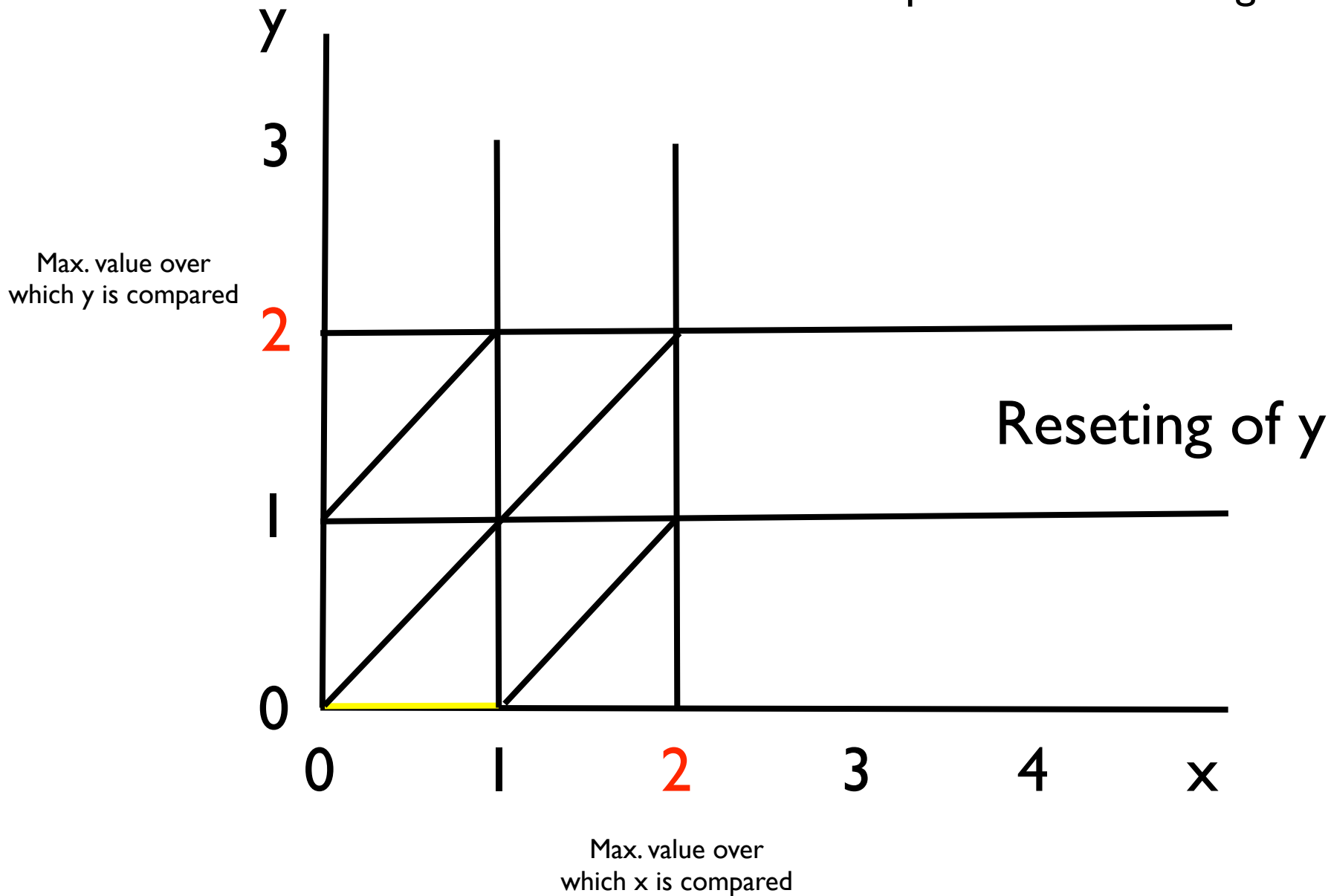
# Continuous state space

The transitions of the time-abstract transition system  
can be rephrased on the regions



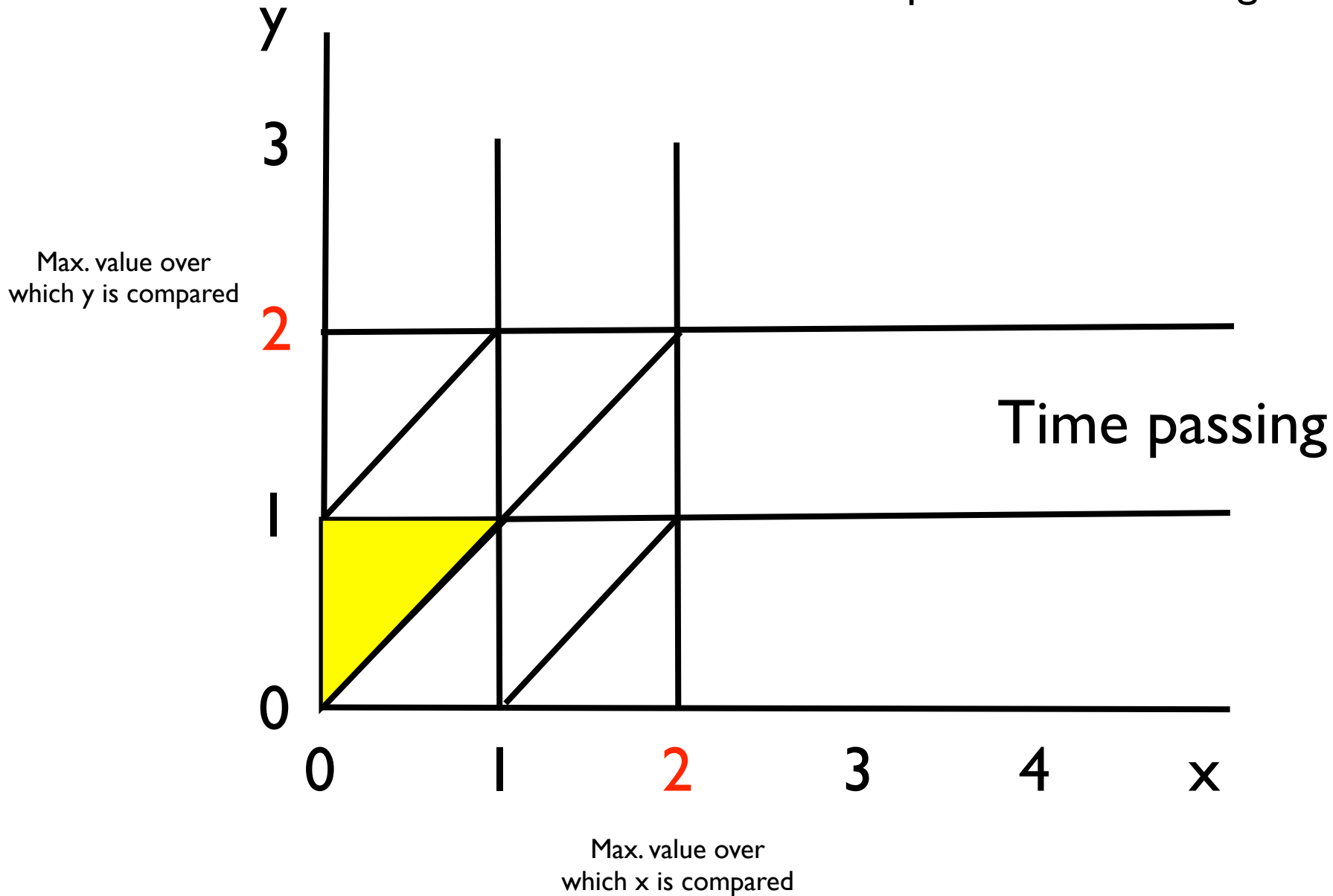
# Continuous state space

The transitions of the time-abstract transition system can be rephrased on the regions



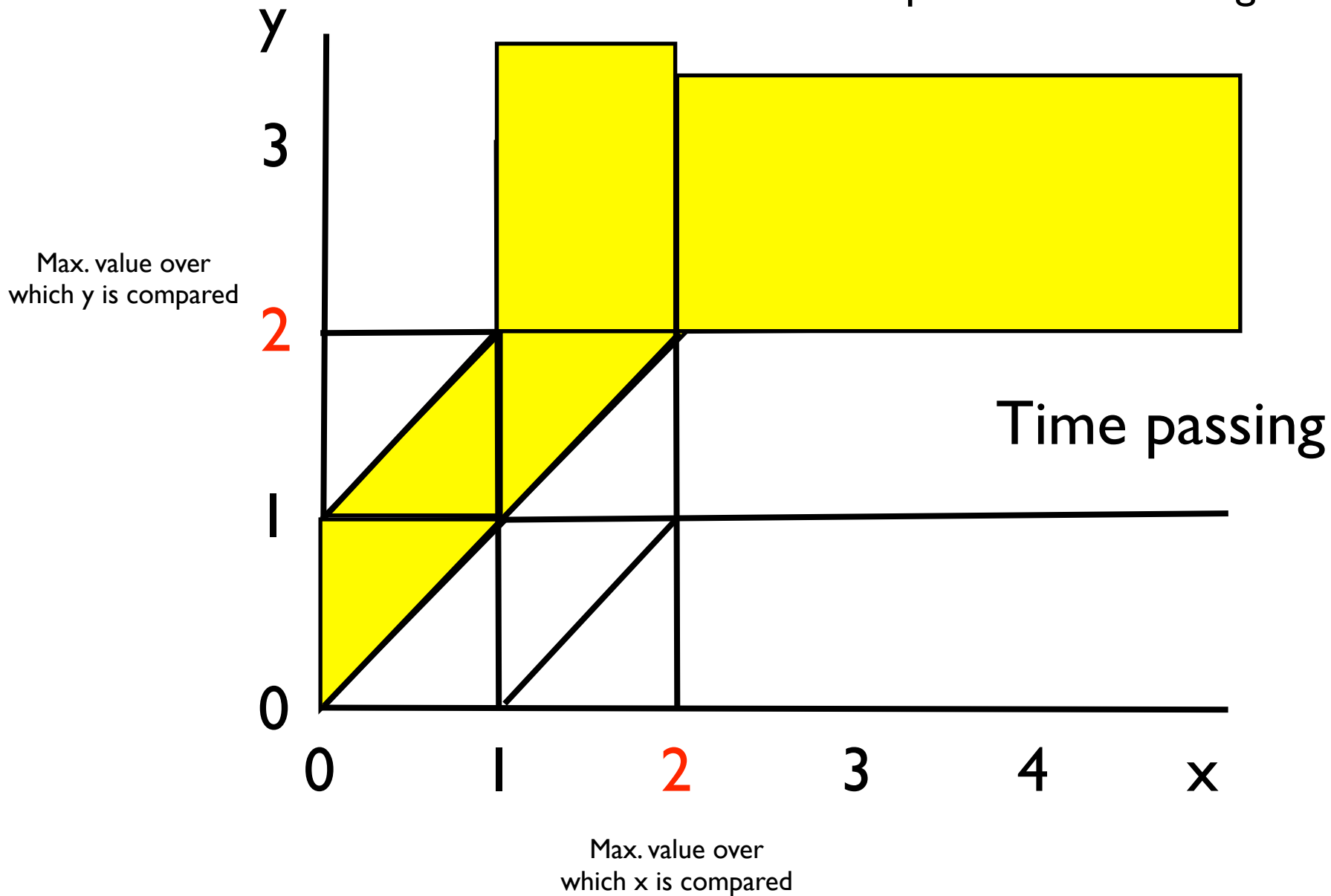
# Continuous state space

The transitions of the time-abstract transition system  
can be rephrased on the regions



# Continuous state space

The transitions of the time-abstract transition system  
can be rephrased on the regions

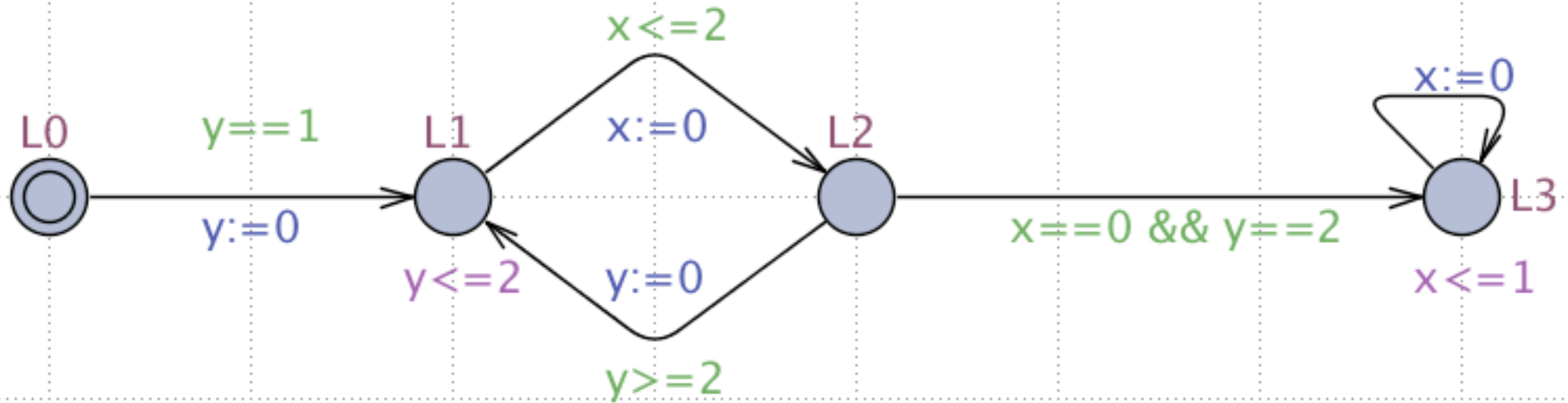


# TA and the backward approach

- The **region bisimulation** is **stable** also for the operations that explore the state space in a **backward fashion** (Pre, Apre)
- ... and so, we can also use backward algorithms to verify TA.

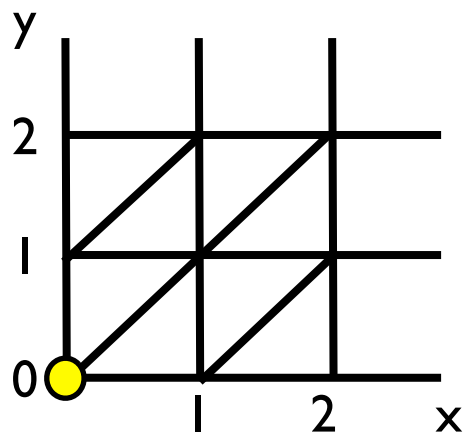
On the use of the **region  
equivalence** to verify  
reachability properties of TA

Forward reachability analysis on a  
simple example

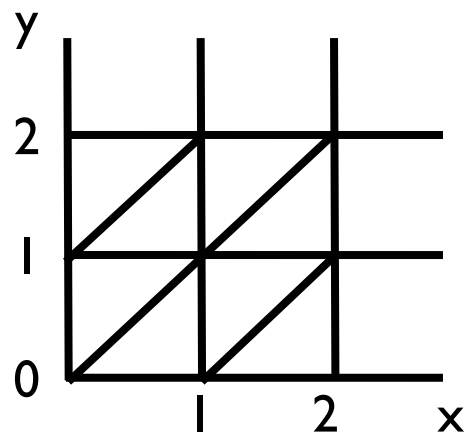


Question: Can L3 be reached ?

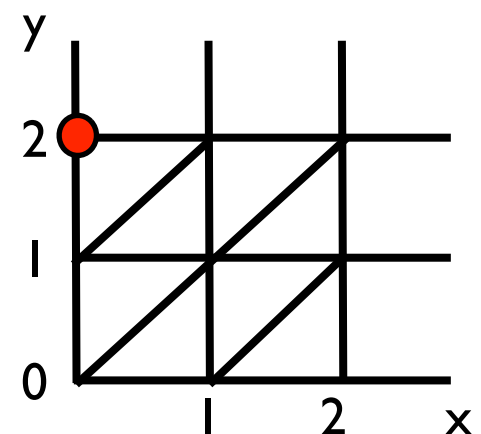
$L_0$

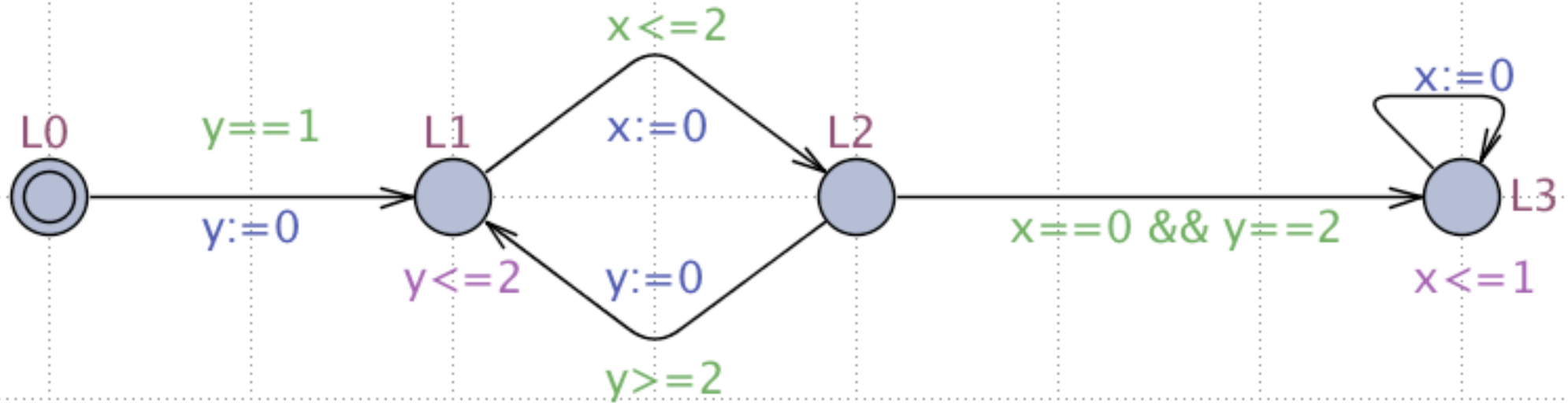


$L_1$

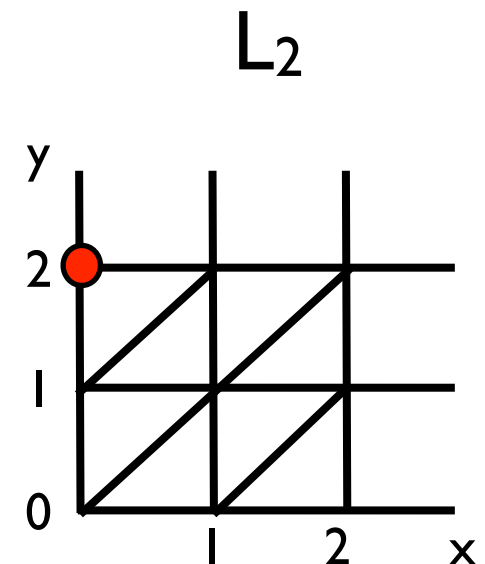
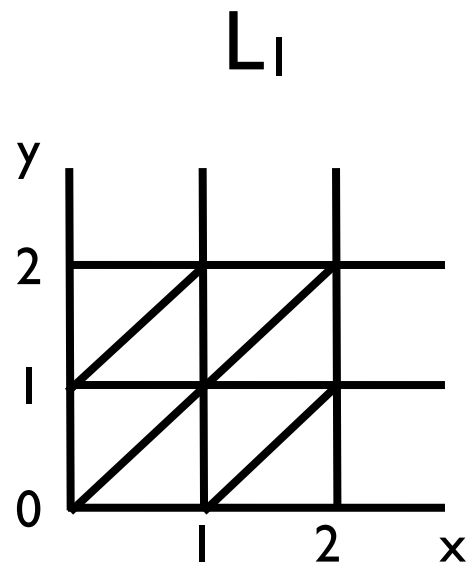
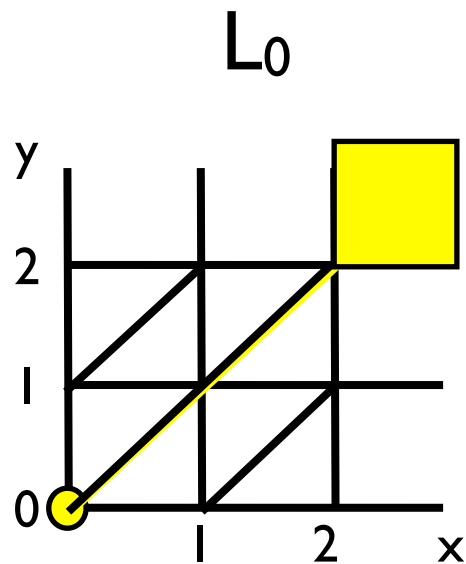


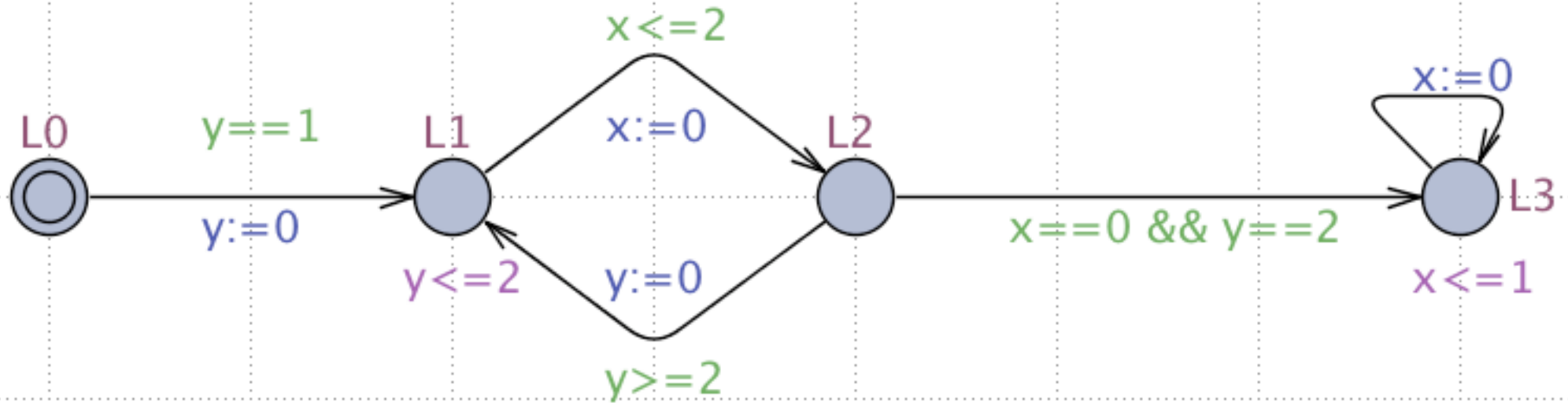
$L_2$



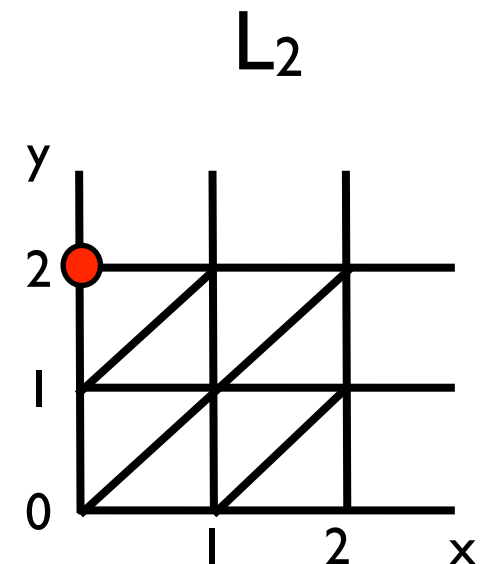
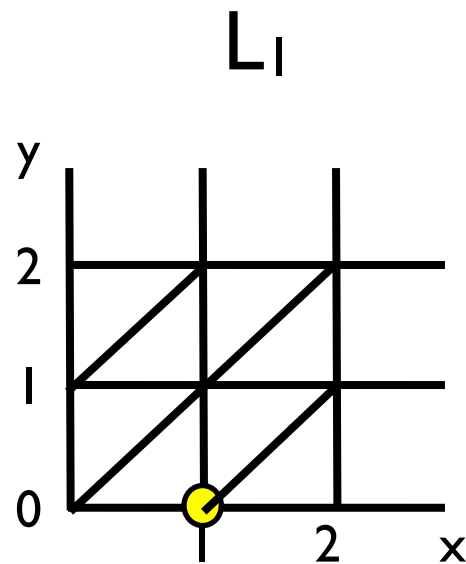
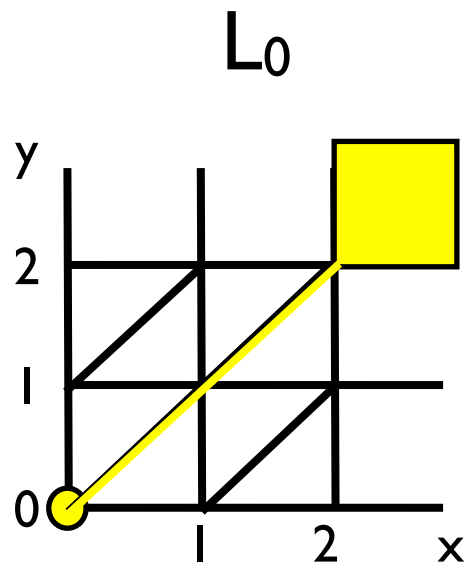


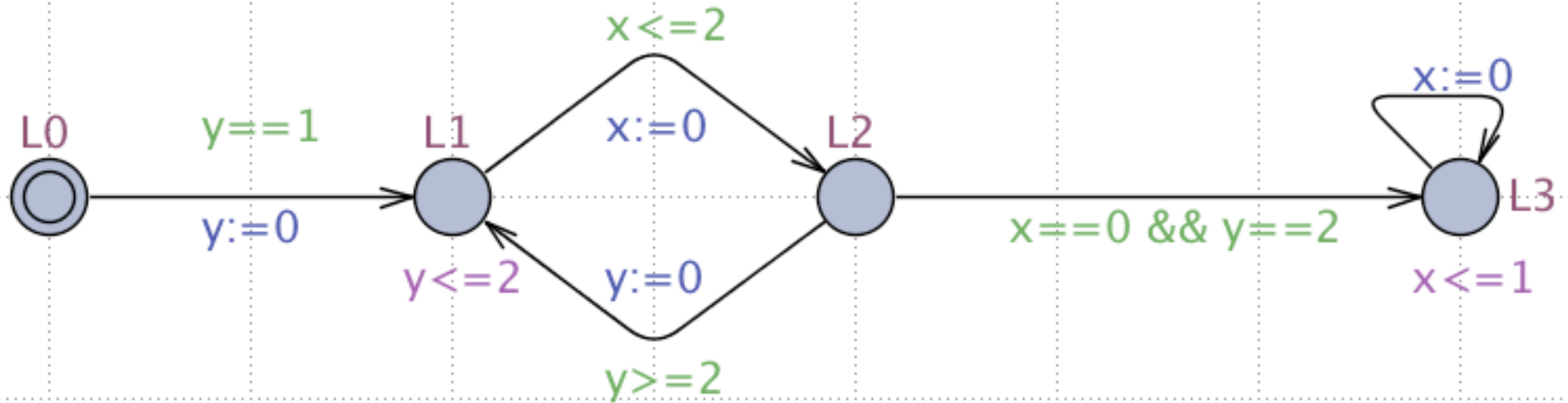
Question: Can L3 be reached ?



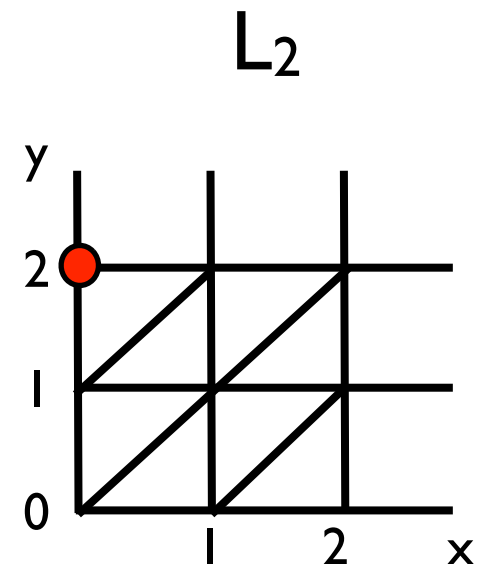
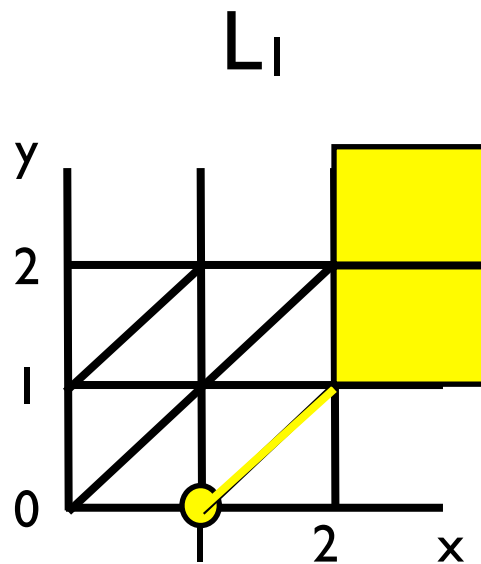
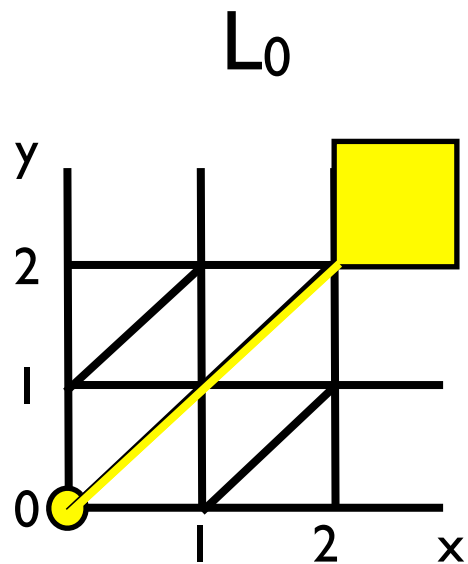


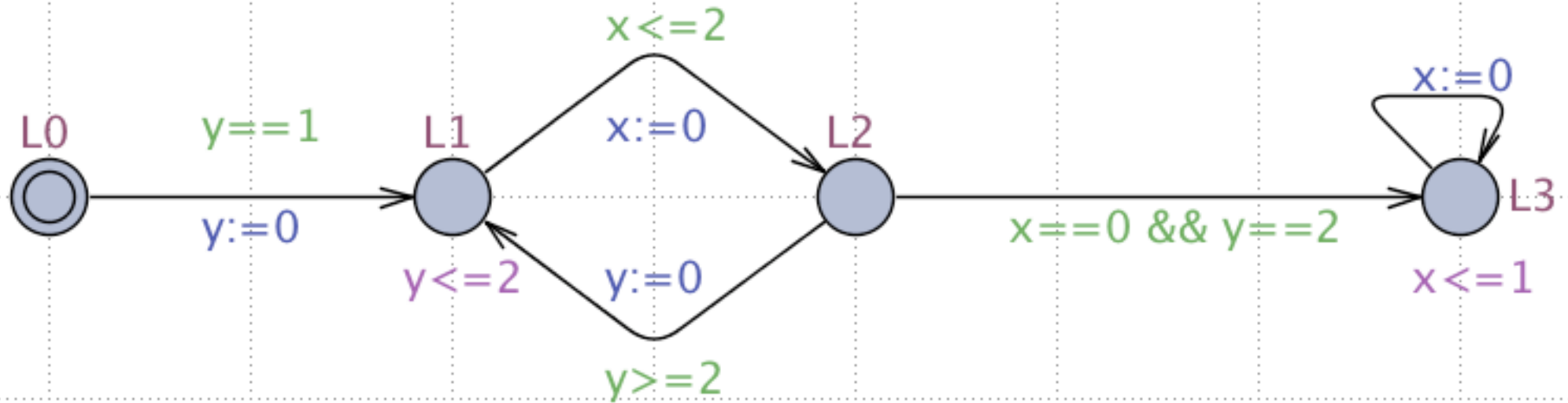
Question: Can L3 be reached ?



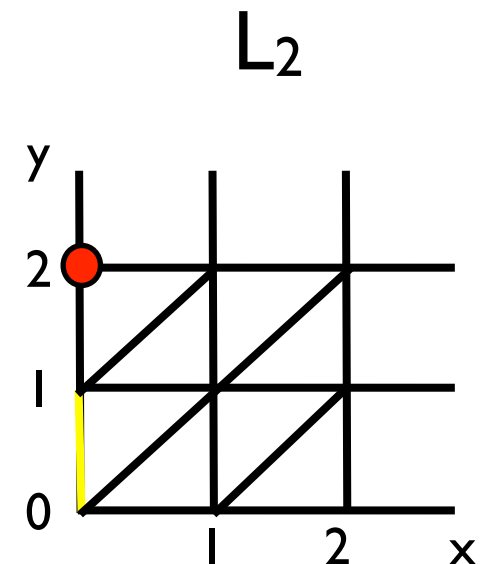
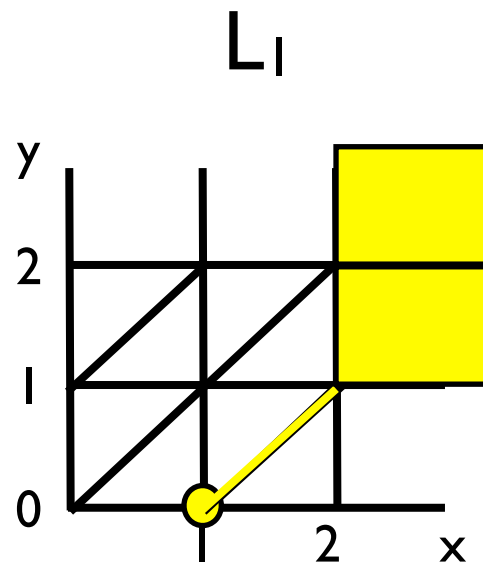
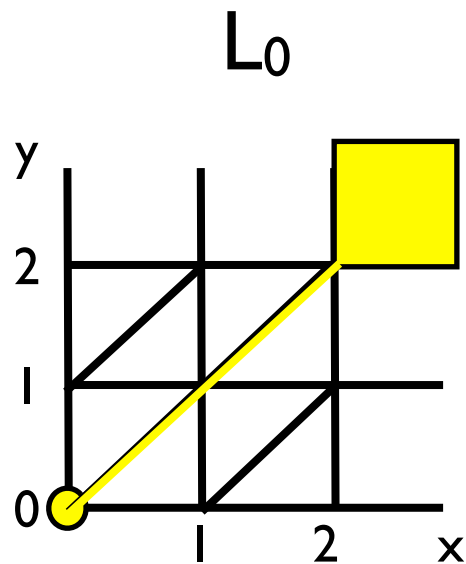


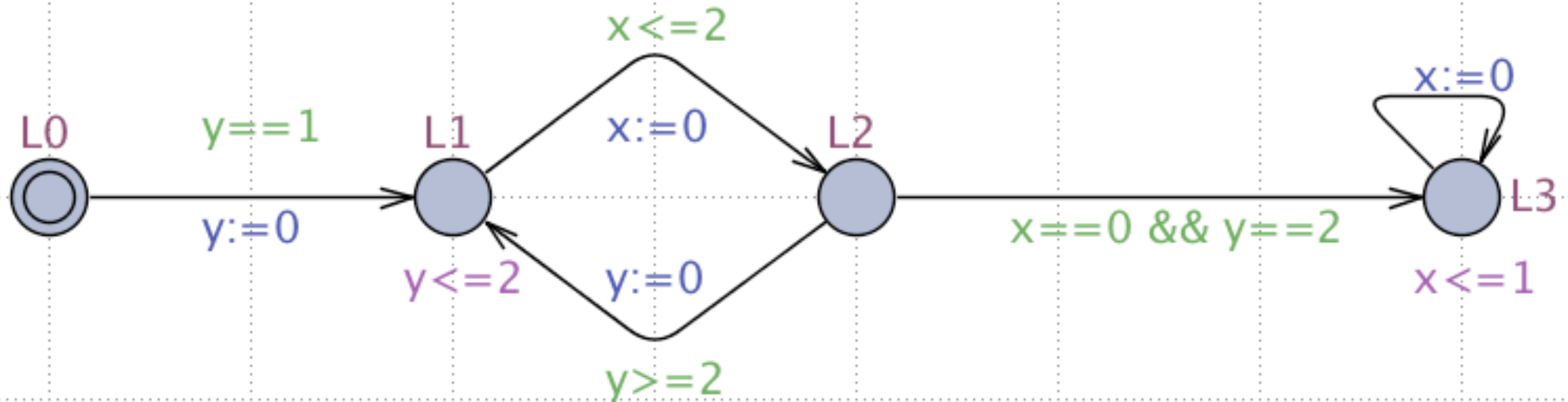
Question: Can L3 be reached ?



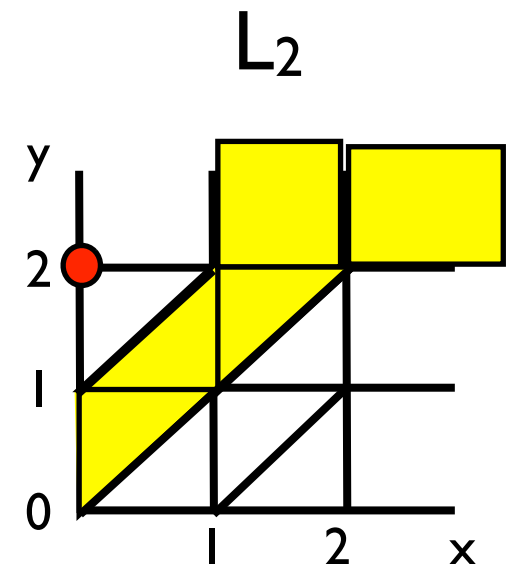
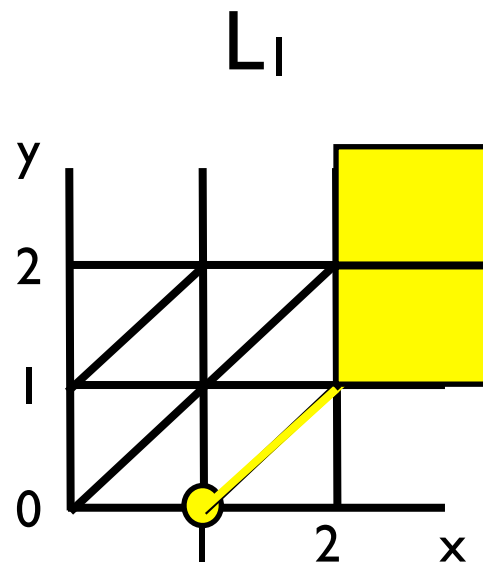
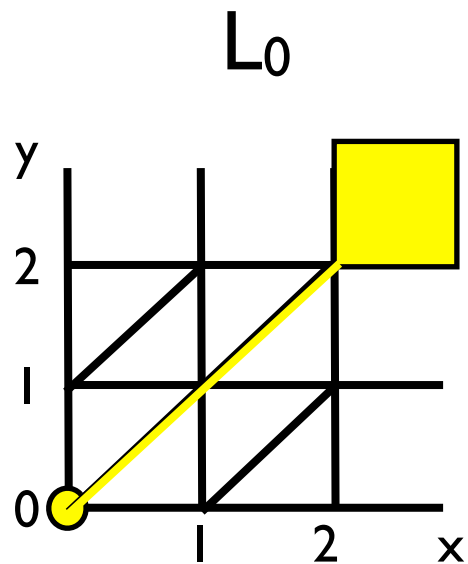


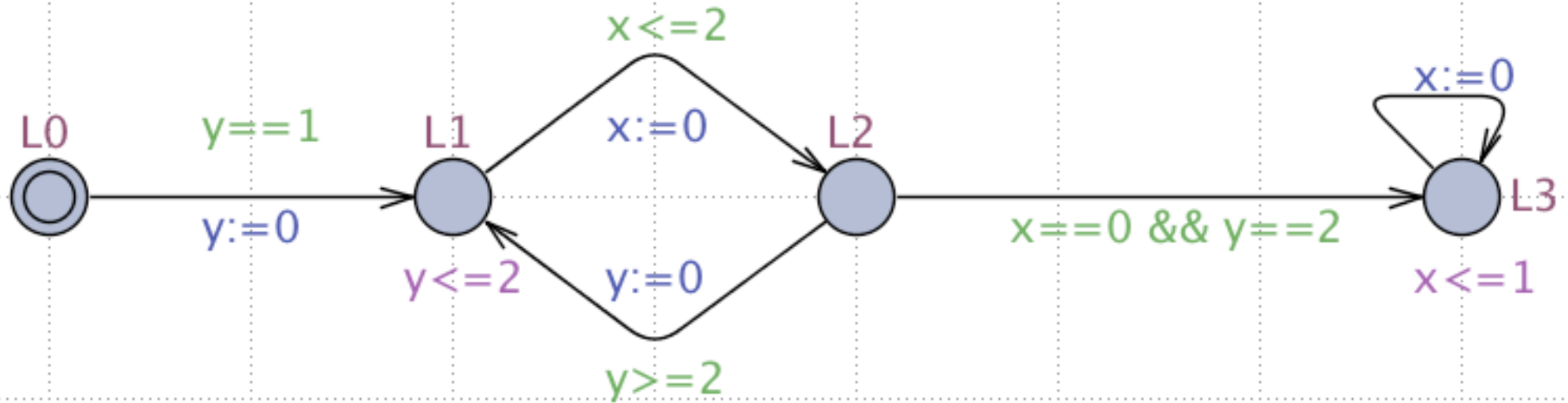
Question: Can L3 be reached ?



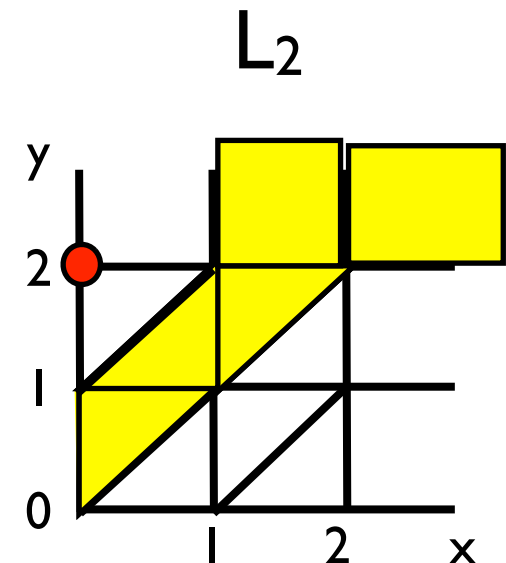
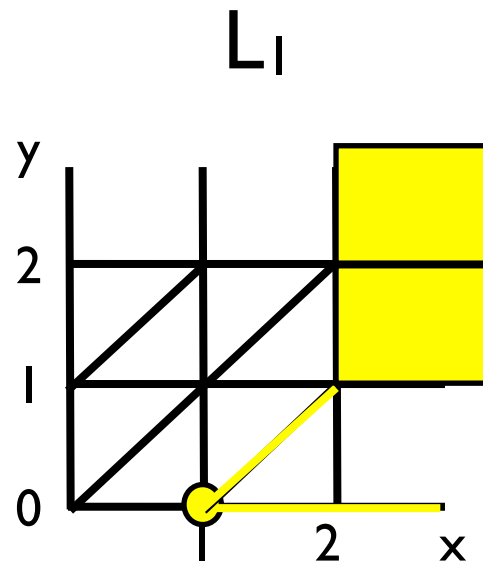
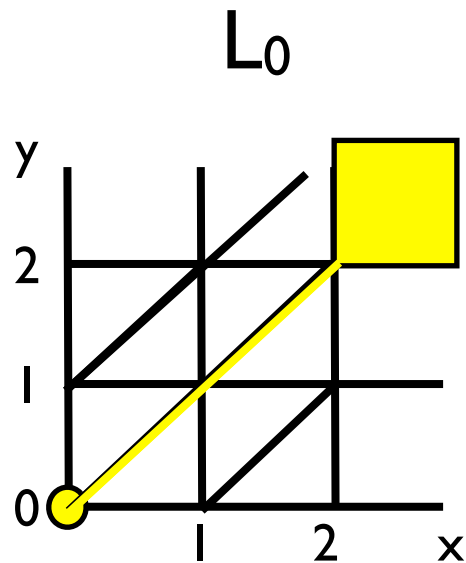


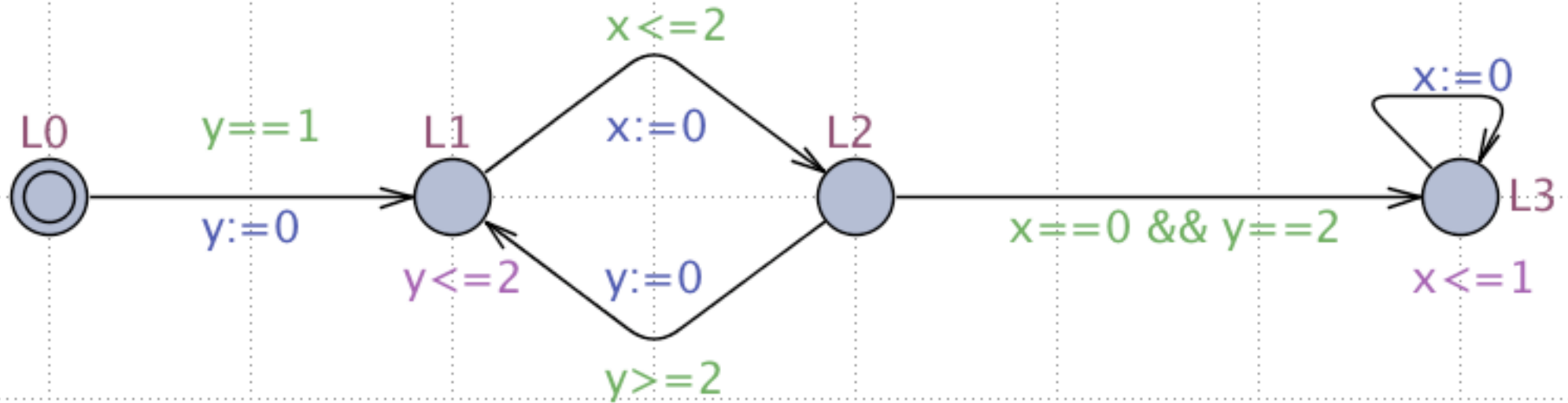
Question: Can L3 be reached ?



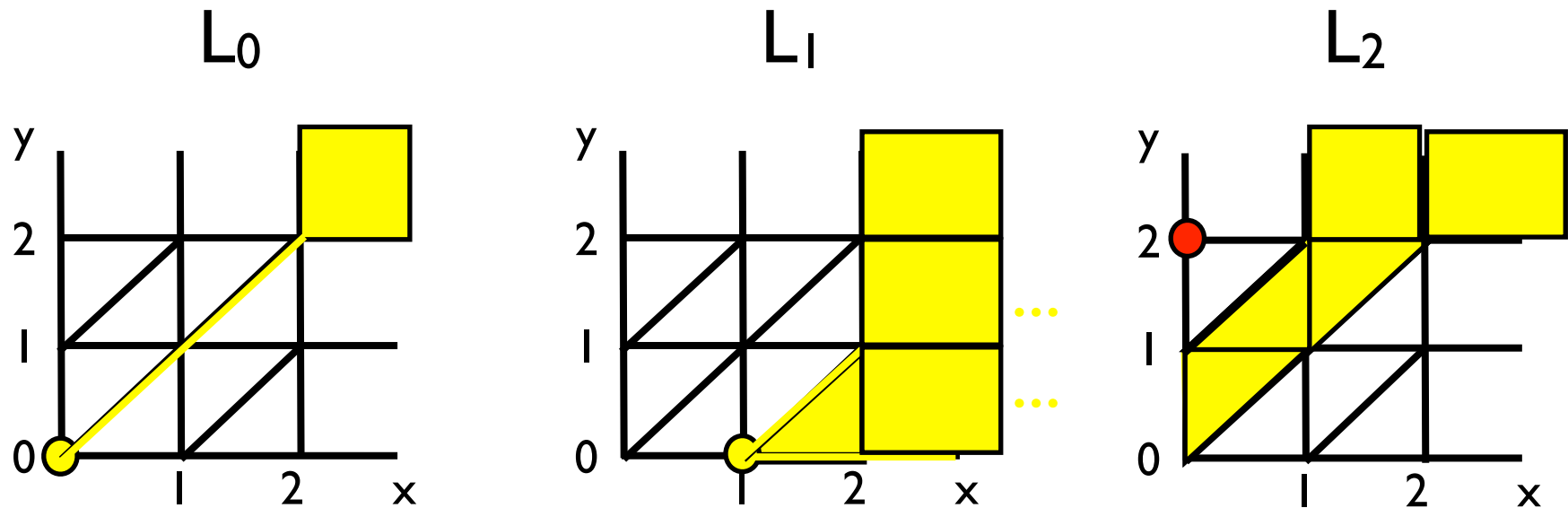


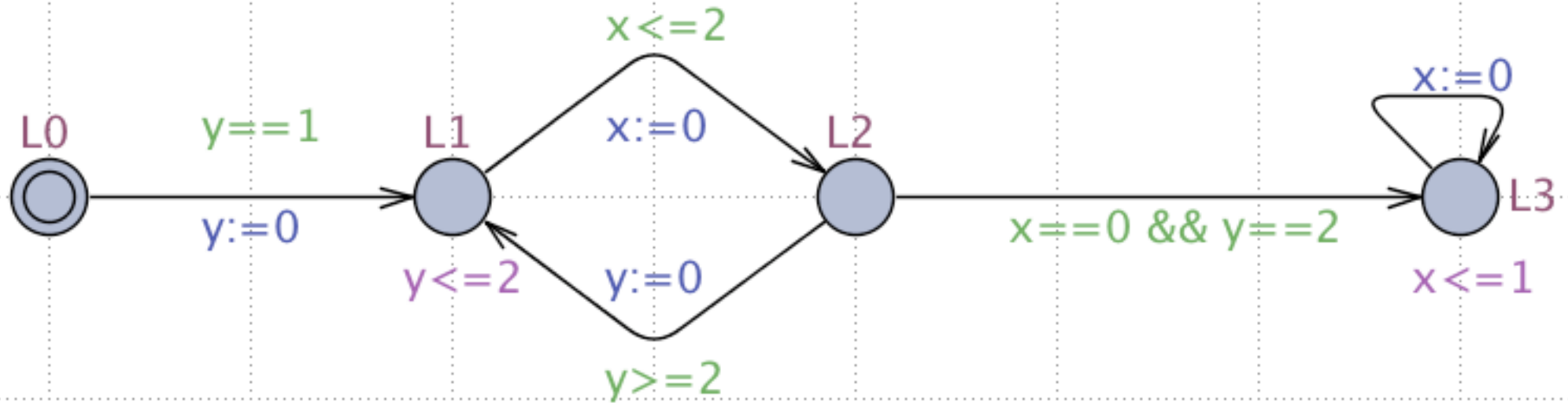
Question: Can L3 be reached ?



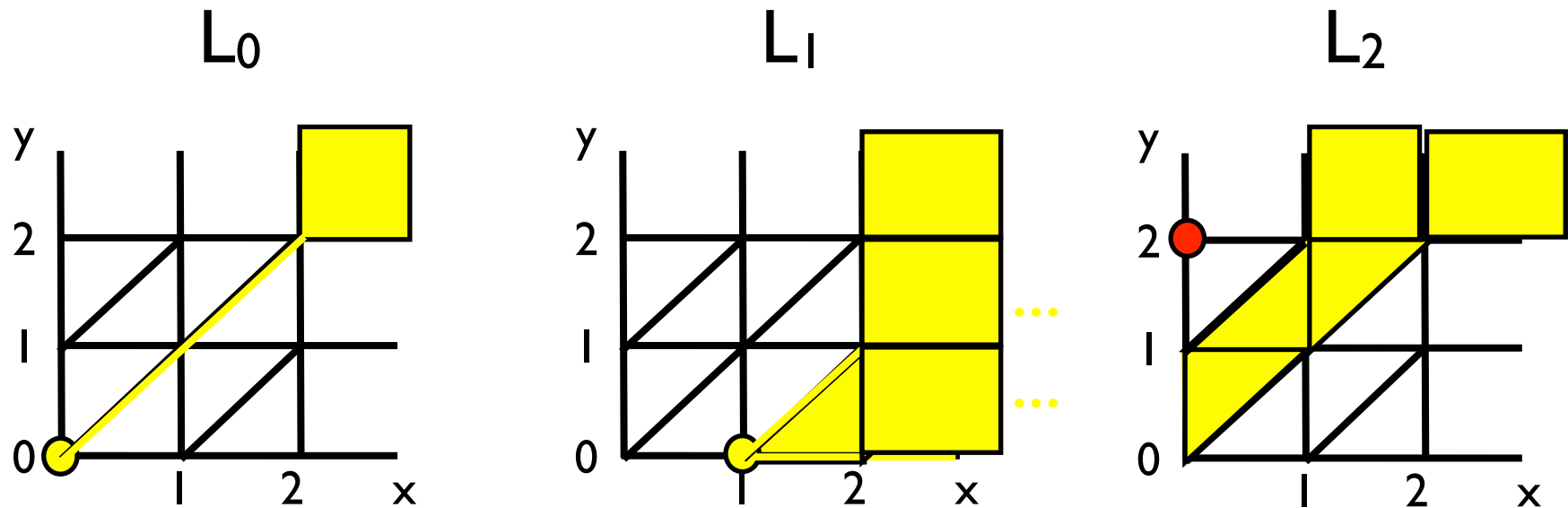


Question: Can L3 be reached ?





Question: Can L3 be reached ?

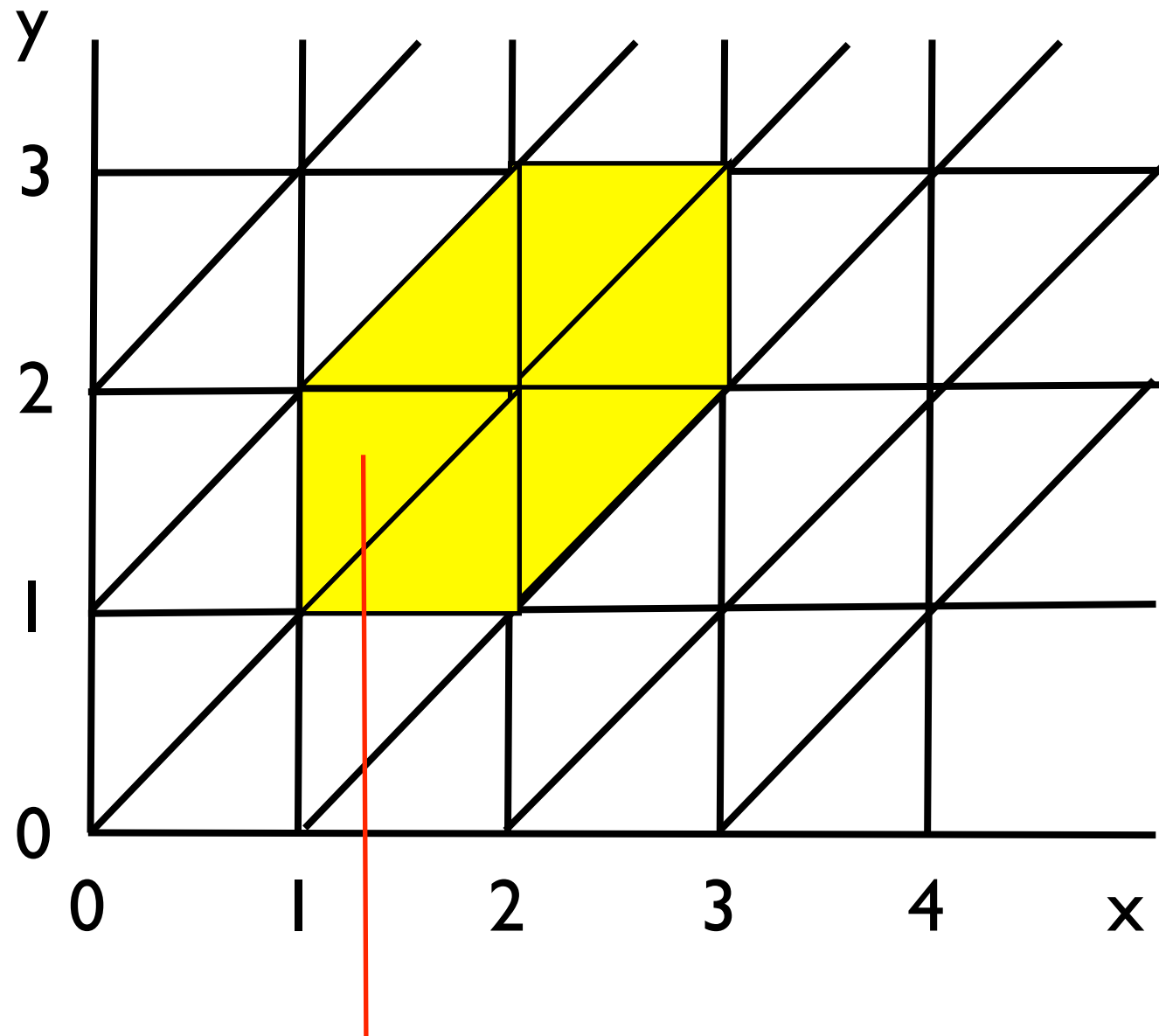


Taking the transition from L<sub>1</sub> to L<sub>2</sub> does not add any new states, so we have reached a fixed point.

# Zones

- The region equivalence gives rise to a **finite quotient** and guarantees that our fixed point algorithms are **terminating**.
- Nevertheless, the number of region is **exponential** in the **number of clocks** as well as in the binary encoding of **constants**.
- To mitigate this state explosion phenomenon, we can use efficient data-structure to represent **convex unions of regions**. **Zones** are such a data-structure.
- Note that the reachability problem for timed automata is **complete for PSpace**, so it is believed that the state explosion is unavoidable in the worst case.

# Zones

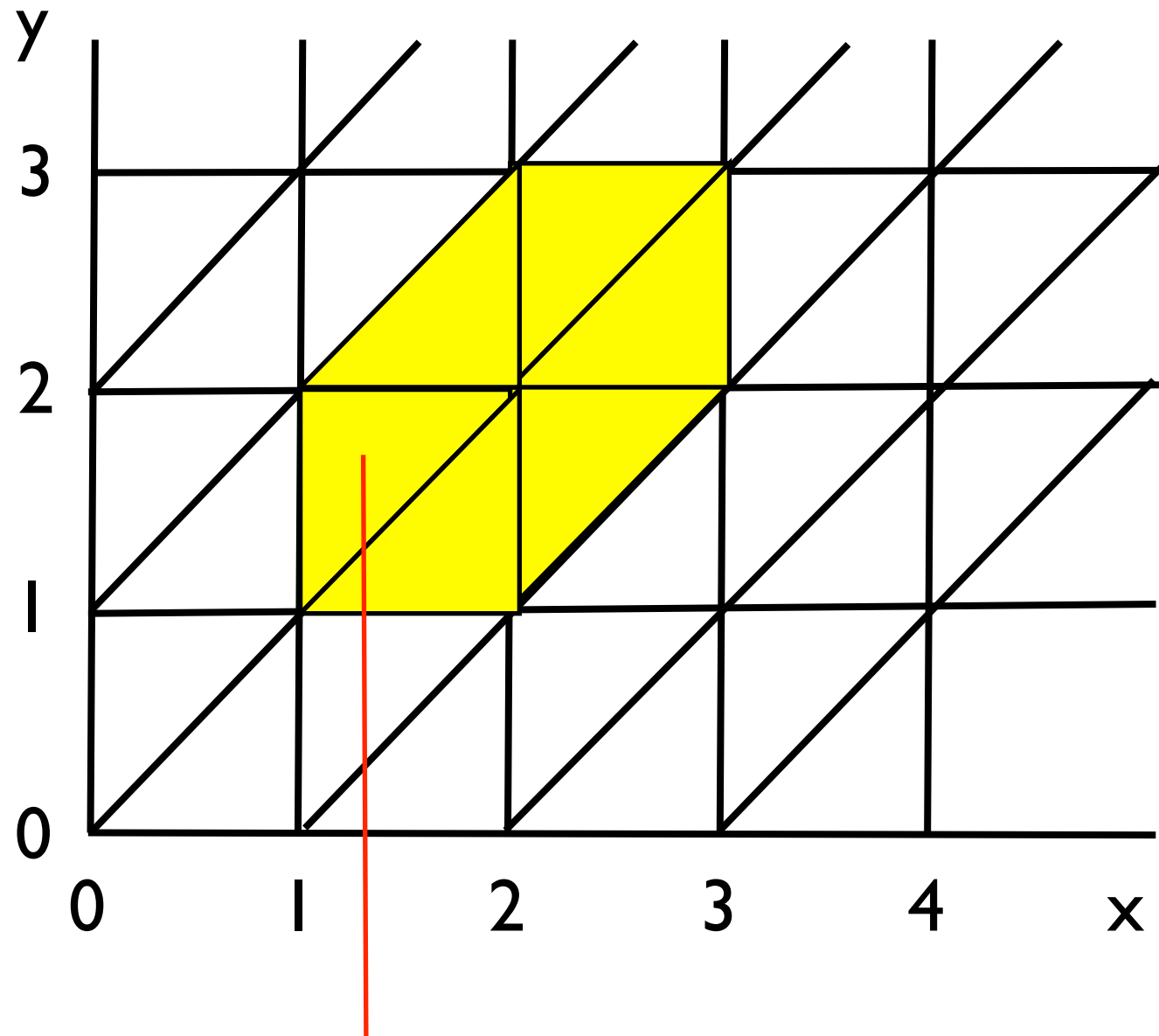


$$x \geq 1 \wedge x \leq 3 \wedge y \leq 3 \wedge y \geq 1 \wedge x - y \geq -2 \wedge y - x \geq 1$$

# Zones

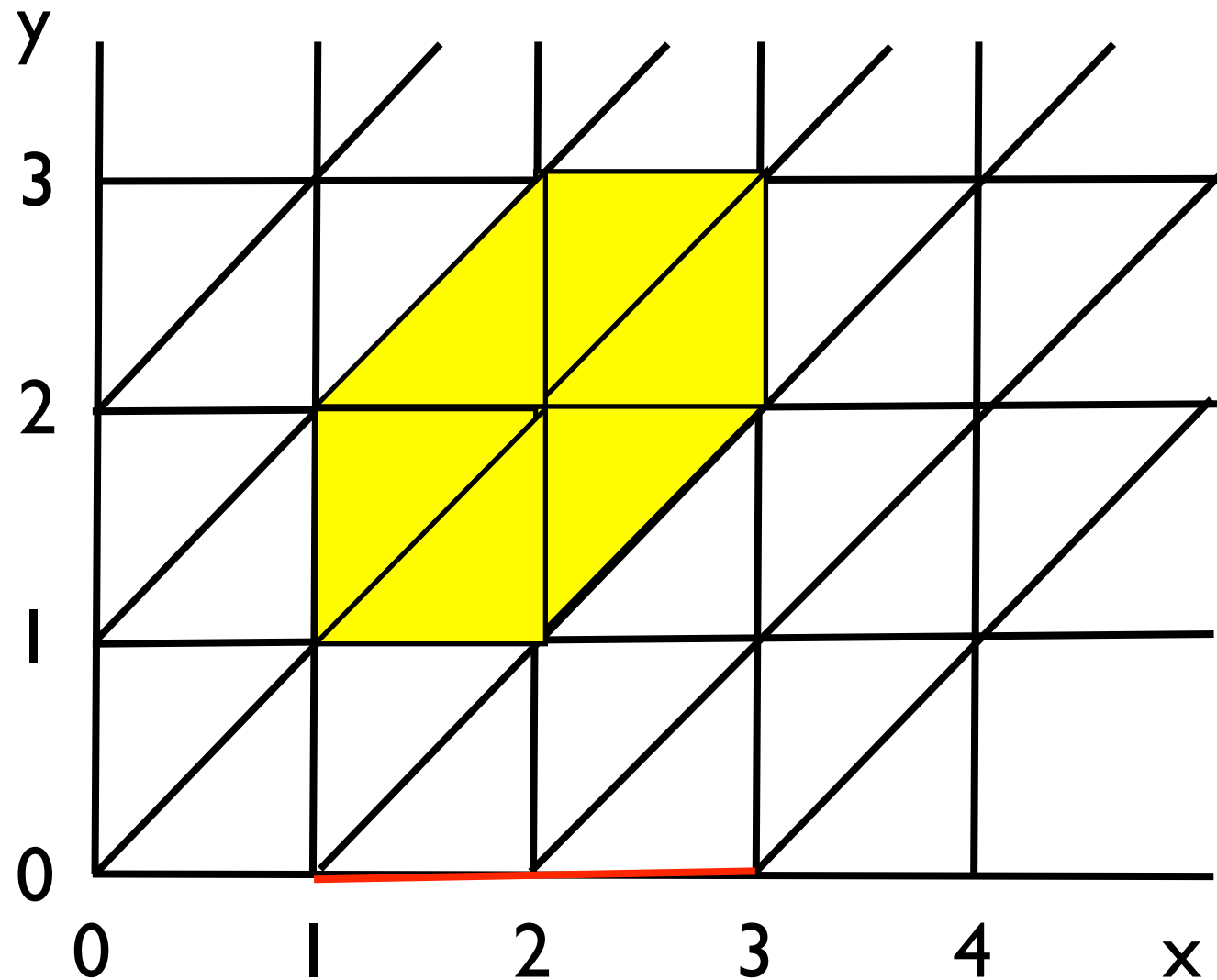
- Let  $\mathbb{C}$  be a finite set of clocks, a **zone** is defined by a set of constraints of the form:
  - (1)  $x - y \sim c$  where  $x, y \in \mathbb{C}$  and  $c \in \mathbb{Z}$ .
  - (2)  $x \sim c$  where  $x \in \mathbb{C}$  and  $c \in \mathbb{Z}$ .and  $\sim \in \{ \leq, <, =, >, \geq \}$ .
- **Zones** are closed under the **resetting operation**, the **forward** and **backward time passing** operations, and intersection.
- Unfortunately, zones are **not** closed under union nor complementation. So implementations need to maintain lists of zones.
- Zones can be canonically represented by **DBM** (=Difference Bound Matrices).

# Zones



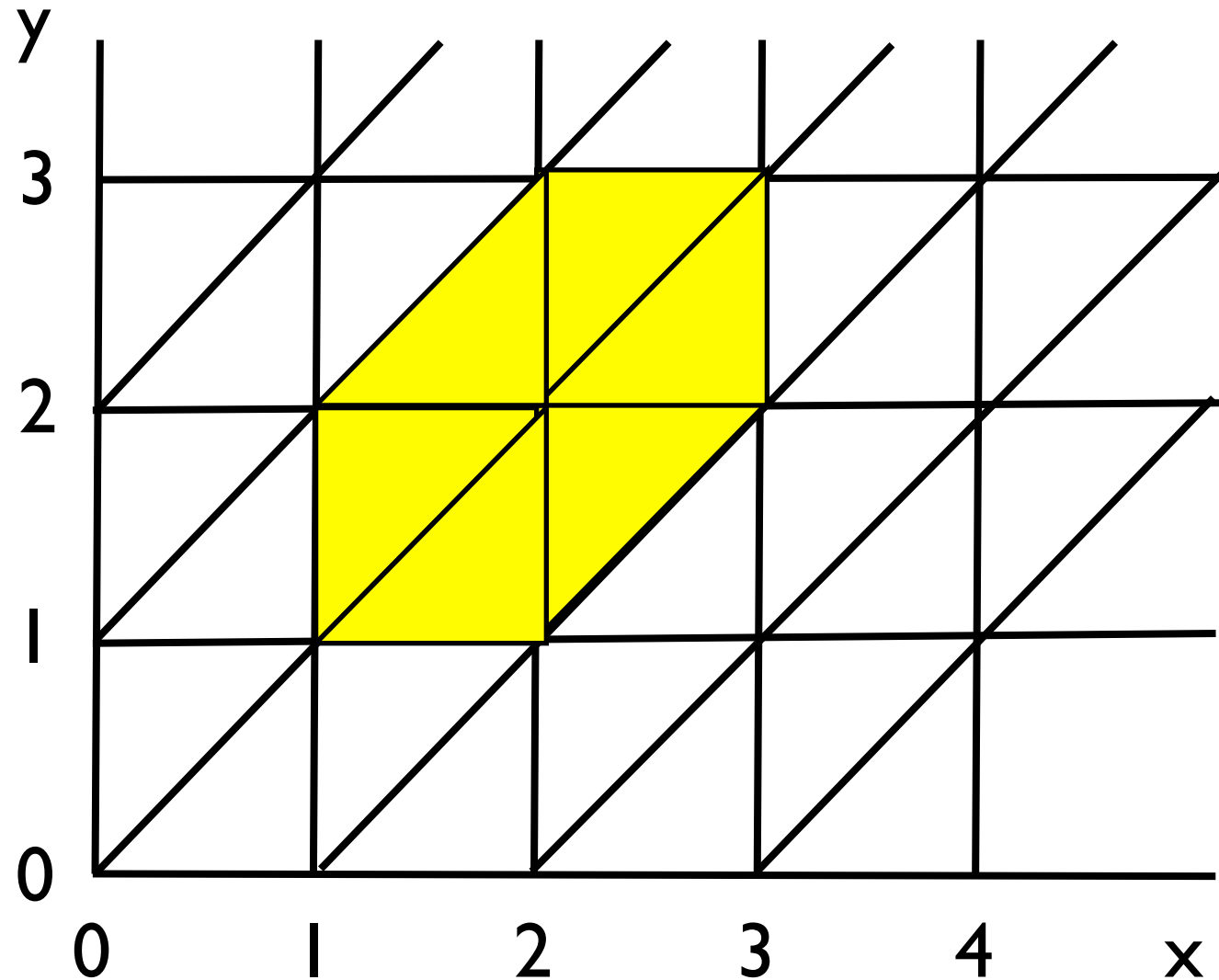
$$x \geq 1 \wedge x \leq 3 \wedge y \leq 3 \wedge y \geq 1 \wedge x - y \geq -2 \wedge y - x \geq 1$$

# Zones



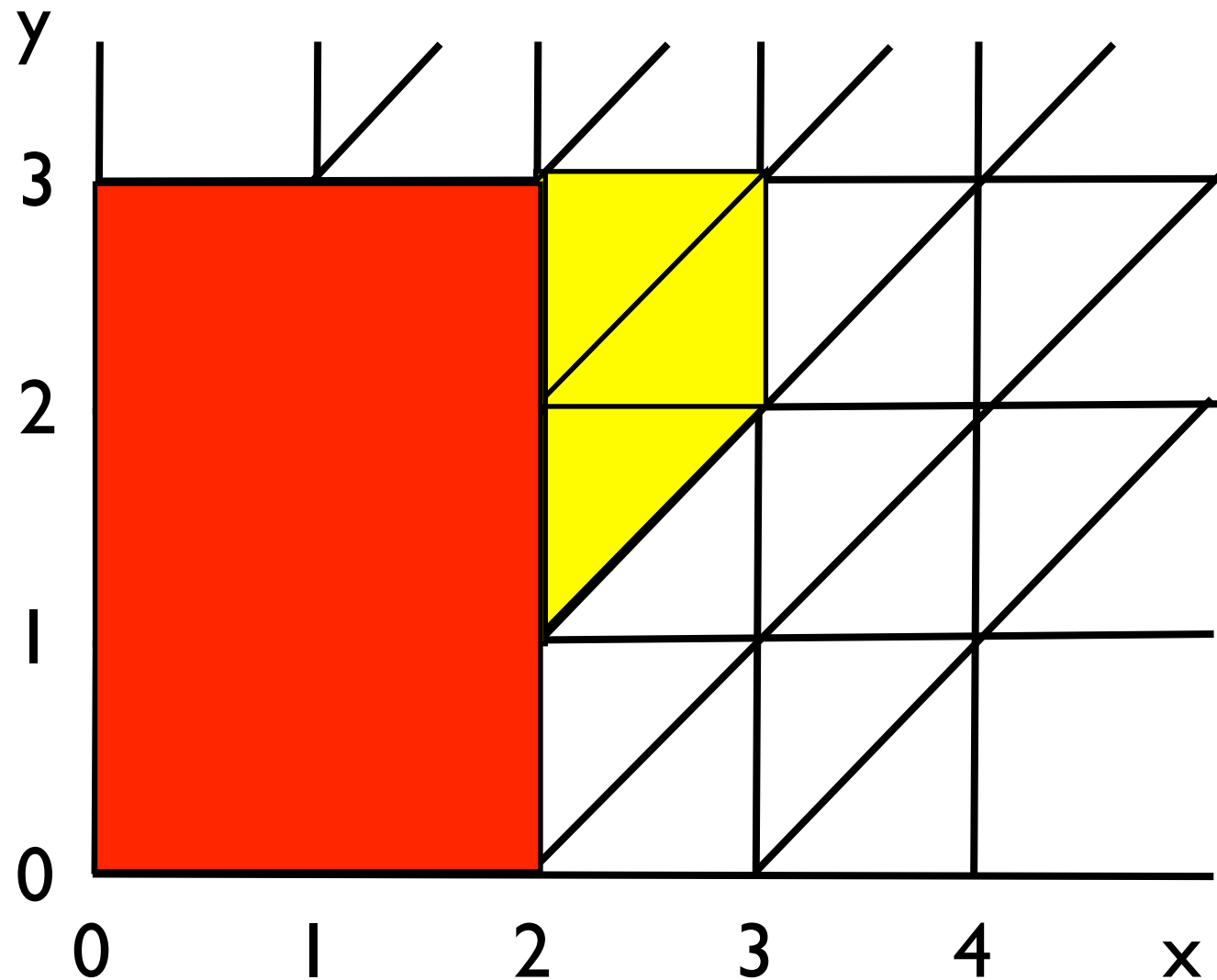
Reseting of  $y$ :  $x \geq 1 \wedge x \leq 3 \wedge y=0$

# Zones



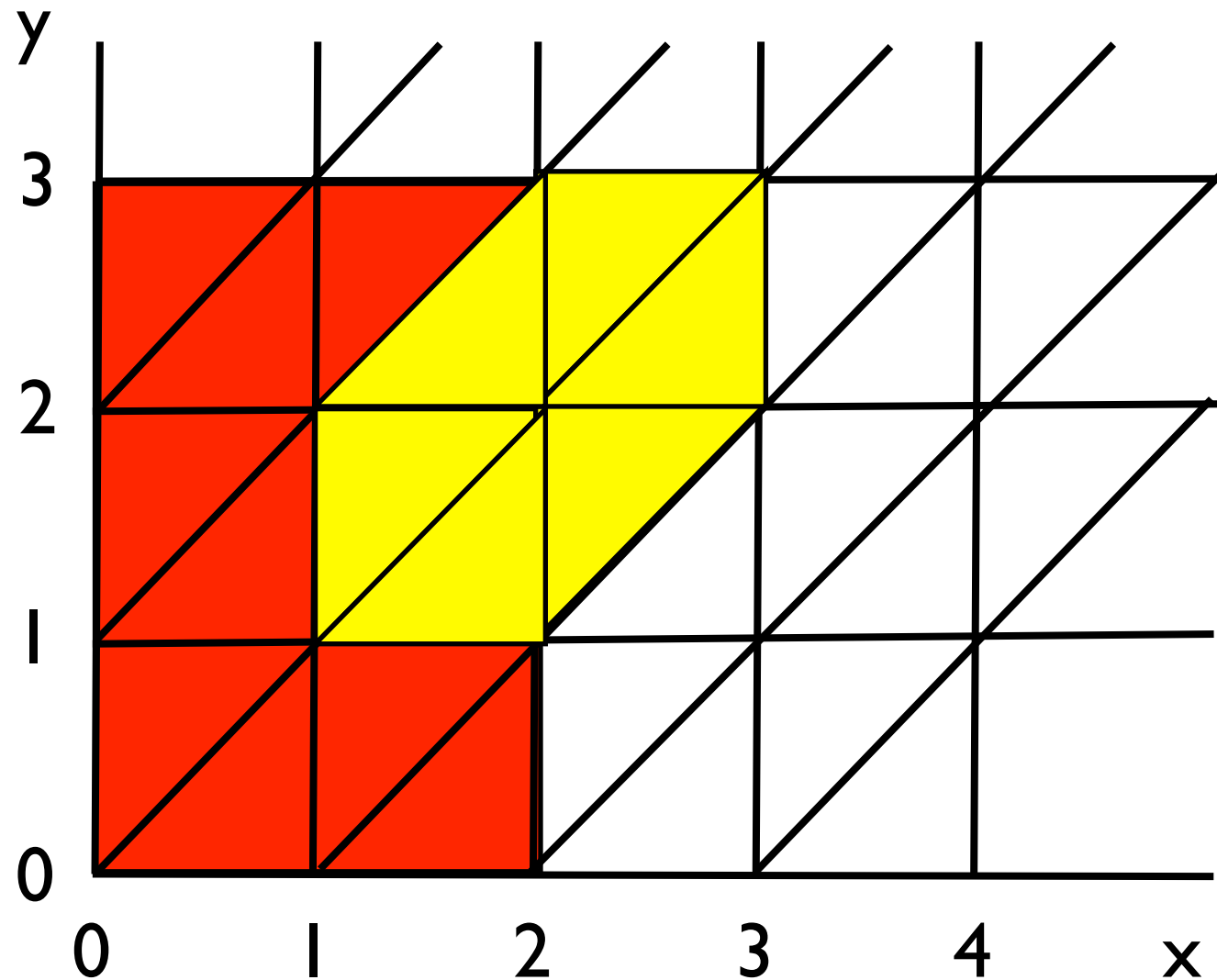
Conjunction with the guard:  $x \leq 2 \wedge y \leq 3$

# Zones



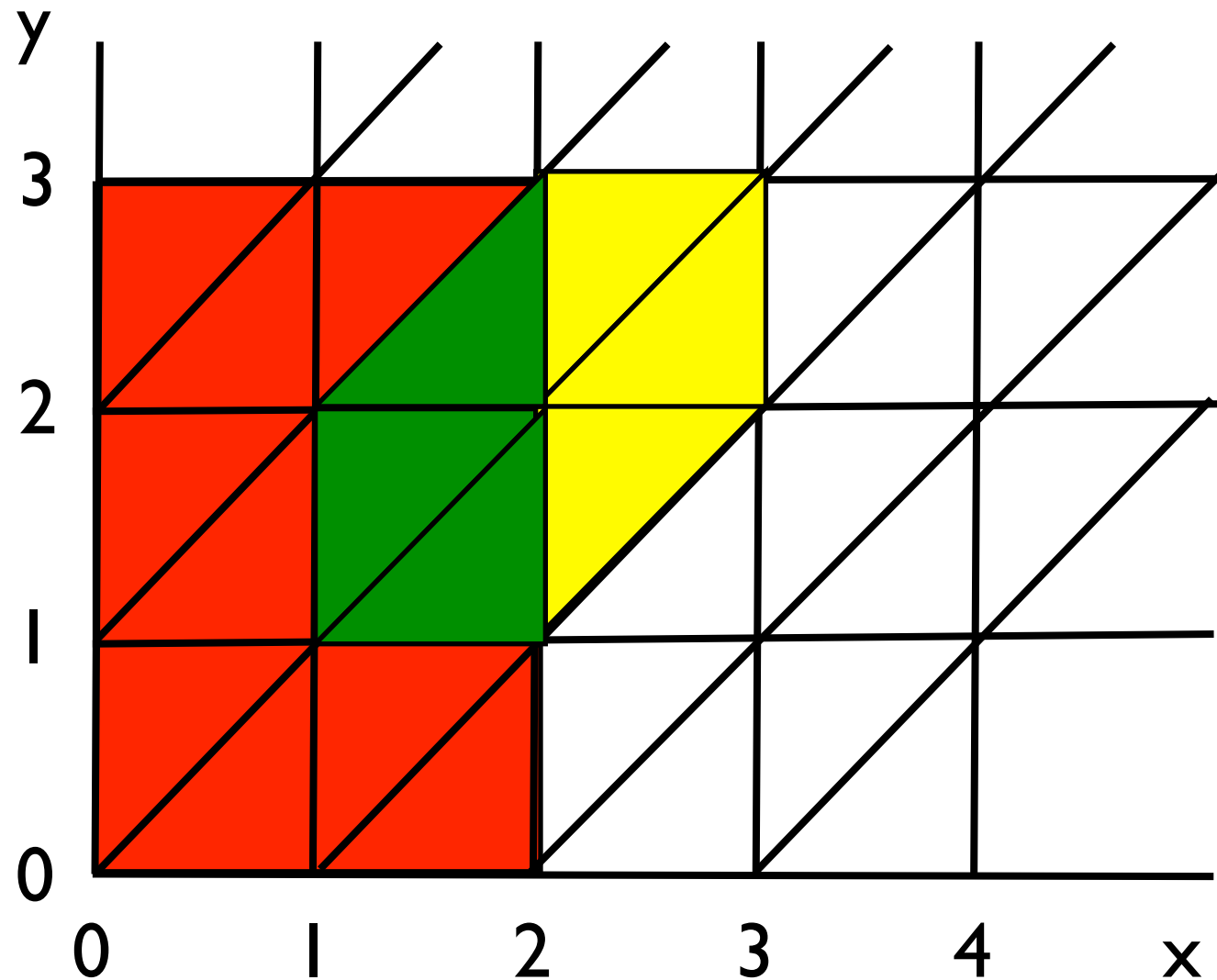
Conjunction with the guard:  $x \leq 2 \wedge y \leq 3$

# Zones



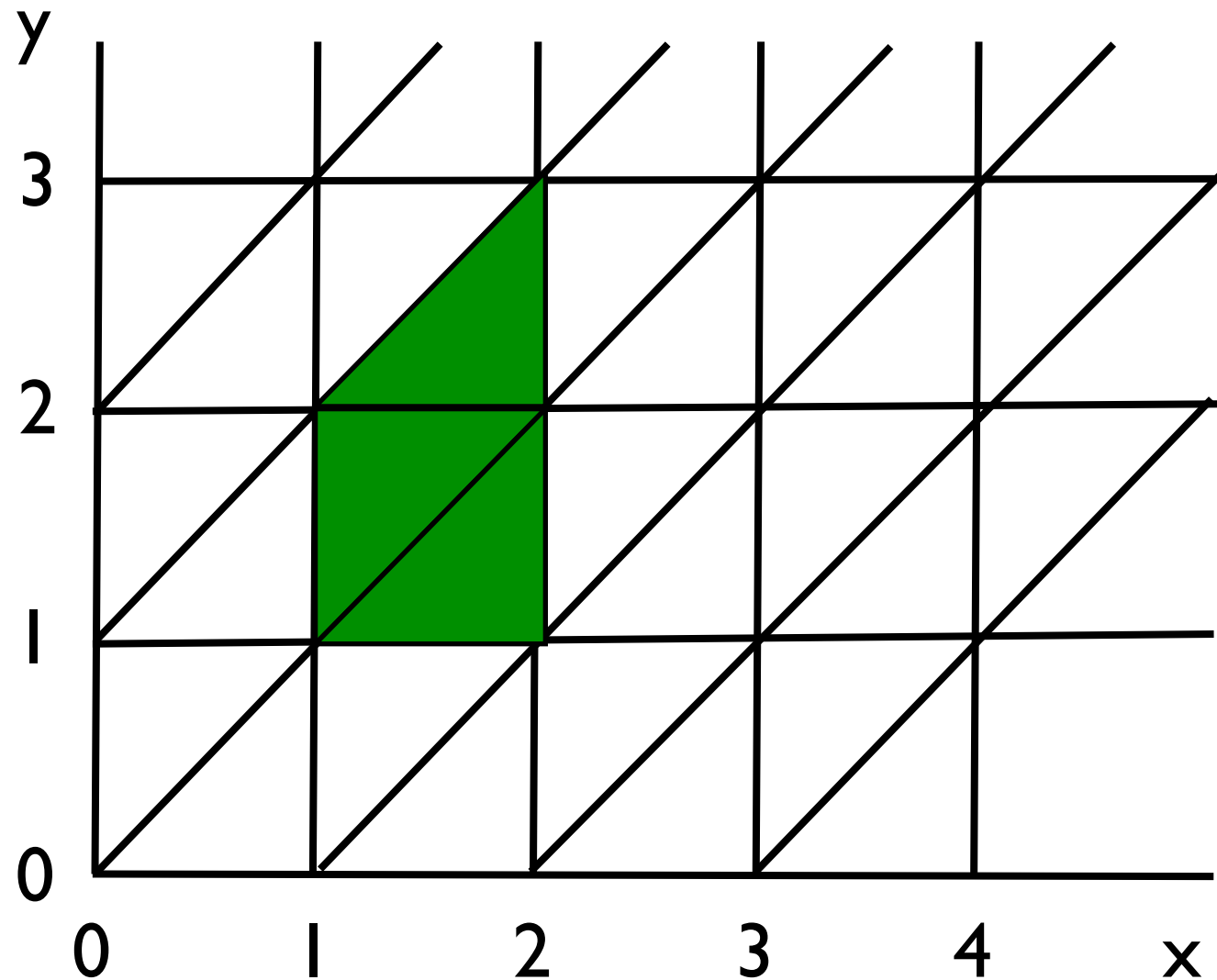
Conjunction with the guard:  $x \leq 2 \wedge y \leq 3$

# Zones



Conjunction with the guard:  $x \leq 2 \wedge y \leq 3$

# Zones



$$x \geq 1 \wedge x \leq 2 \wedge y \geq 1 \wedge y \leq 3 \wedge x - y \geq -1$$

# Decidability results for TA

# Decidability results

- As a direct consequence of our previous developments, we have that:
- The **reachability verification problem** for TA w.r.t. a set **Goal** which is defined as a union of regions is **decidable**.
- The **safety verification problem** for TA w.r.t. a set **Safe** which is defined as a union of regions is **decidable**.
- The **Büchi verification problem** for TA w.r.t. a set **Goal** which is defined as a union of regions is **decidable**.

# Decidability results

- As the reachability verification problem for TA is decidable then the **timed language emptiness problem** (finite word case) is **decidable** for TA.
- As the Büchi verification problem for TA is decidable then the **timed language emptiness problem** (infinite word case) is **decidable** for TA.

Hint to establish the result: construct a set Goal that ensures non-zenoness and the Büchi acceptance condition of the TA. Show that this set is a finite union of regions. As an intermediary step you will need a generalized Büchi condition.

# Undecidability results for TA

# Undecidability results for TA

- The **timed universality problem**, i.e. does a TA accepts all possible timed words on a alphabet, is **undecidable**.
- The **language inclusion problem** between timed automata is **undecidable**. (direct consequence of the previous undecidability result).