

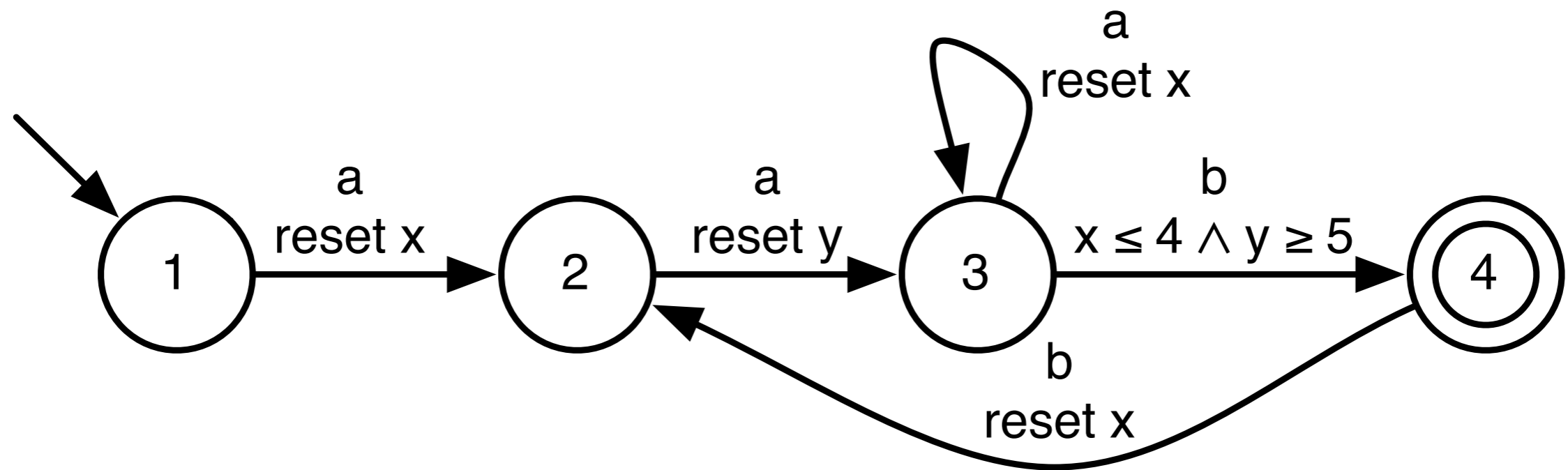
# One-clock deterministic timed automata are efficiently identifiable in the limit

Sicco Verwer, Mathijs de Weerdt, Cees Witteveen  
DWFTT 2009

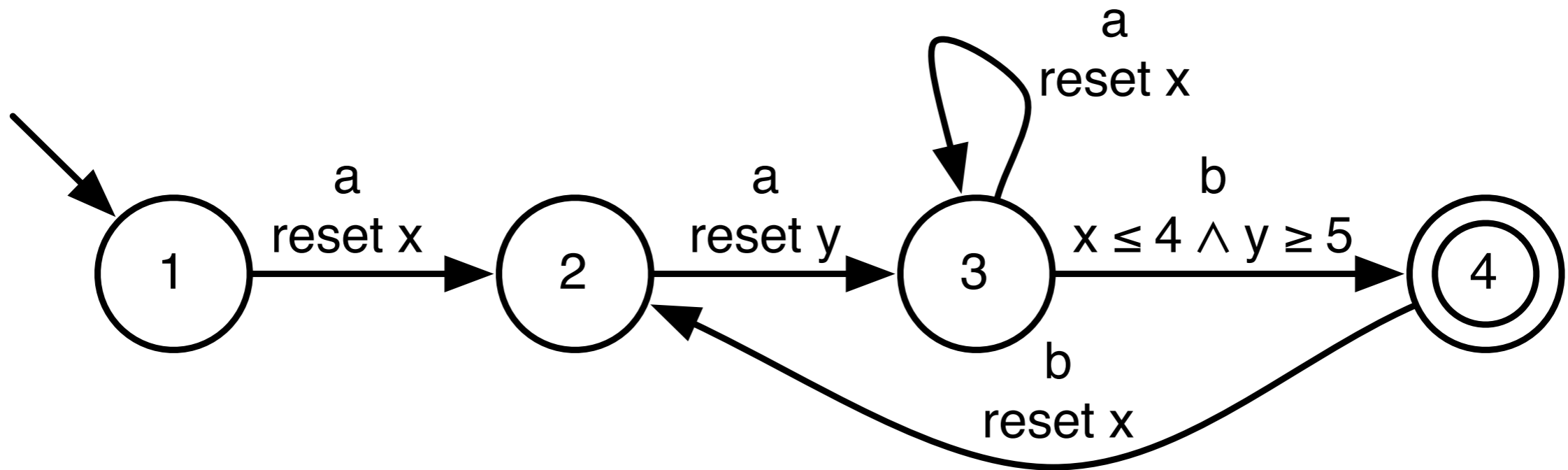
# Overview

- Deterministic timed automata (DTAs)
- Why learn DTAs?
- Efficient identification in the limit
- DTAs are not efficiently identifiable
- Learning DTAs with a single clock efficiently
- Conclusions and future work

# DTAs

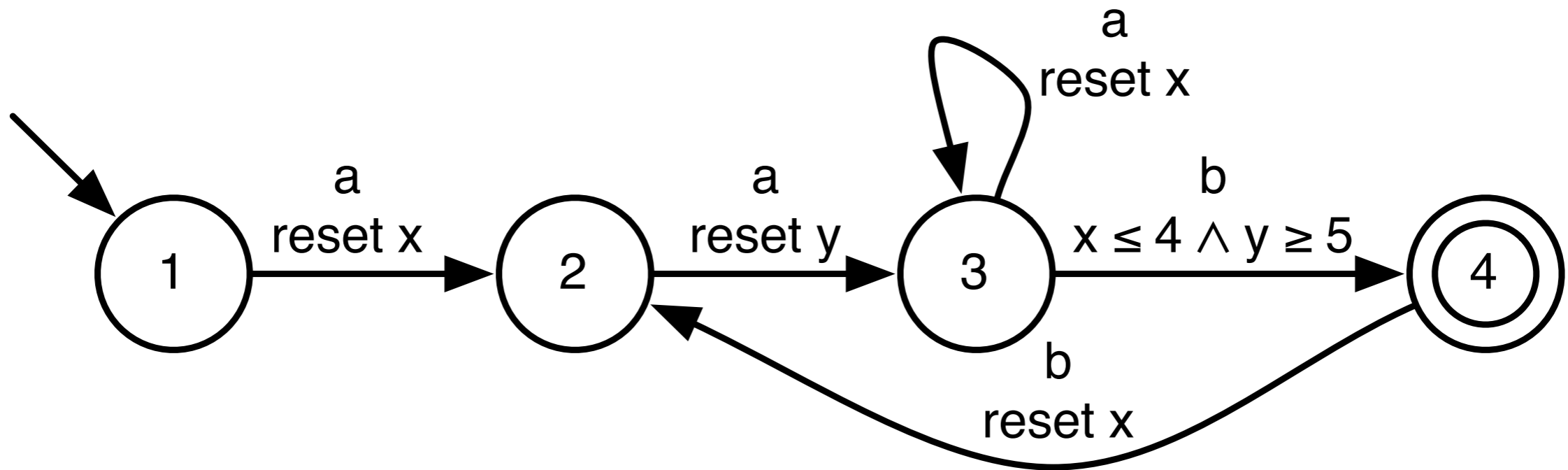


# DTAs



**accepts:** (a, 1)(a, 2)(a, 3)(b, 4) **rejects:** (a, 1)(a, 2)(a, 1)(b, 2)

# DTAs



rejects:  $(a, t)(a, t')(b, t'')$  for **any**  $t, t', t''$   
because  $x$  is reset before  $y$  in such a path

# DTAs

- A deterministic timed automaton (DTA):
  - A deterministic finite state automaton (DFA)
  - A set of **clocks**  $X$
  - A **clock guard** (constraint)  $g$  for every transition  $d$
  - A set of **clock resets**  $R$  for every transition  $d$
- Timed properties:
  - All clocks increase their values **synchronously**
  - A clock value can be **reset to 0**
  - A transition can fire if its clock guard is **satisfied**

# Why learn DTAs?

- DTAs:
  - Use an **explicit** time representation (using numbers)
  - Are **intuitive** models for many real-time systems
  - Are used to **model** and **verify** reactive systems
- In practice it is often difficult to construct DTAs by hand, but data is easy to obtain:
  - We want to **identify** them from data

# Why learn DTAs?

- Any timed system can also be represented using an **implicit** time representation, using DFAs or HMMs
  - **Exponential blowup** of the models and the data required for learning
  - **Inefficient** in the size of the timed data and the timed model
- We want to learn DTAs **directly** from timed data
  - Is it possible to do so **efficiently**?

# Applications

- Learning truck driver **behavior**
- Identifying **process** models
- Inferring **models** for ship movement
- Model based **testing**
- ...
  - Anywhere where representing time **explicitly** results in a large **reduction** in model size

# Overview

- Deterministic timed automata (DTAs)
- Why learn DTAs?
- Efficient identification in the limit
- DTAs are not efficiently identifiable
- Learning DTAs with a single clock efficiently
- Conclusions and future work

# Identification in the limit

- **Identification in the limit** views learning of languages and models as a continuous process:
  - Data is observed
  - This data is used to modify the current model
- Identification is **successful** if from some point on, the model does **not change** anymore, and if it is **correct**
- Identification is **efficient** if this point can occur after observing a **polynomial** amount of **data**, and if the model is computed using a **polynomial time** algorithm

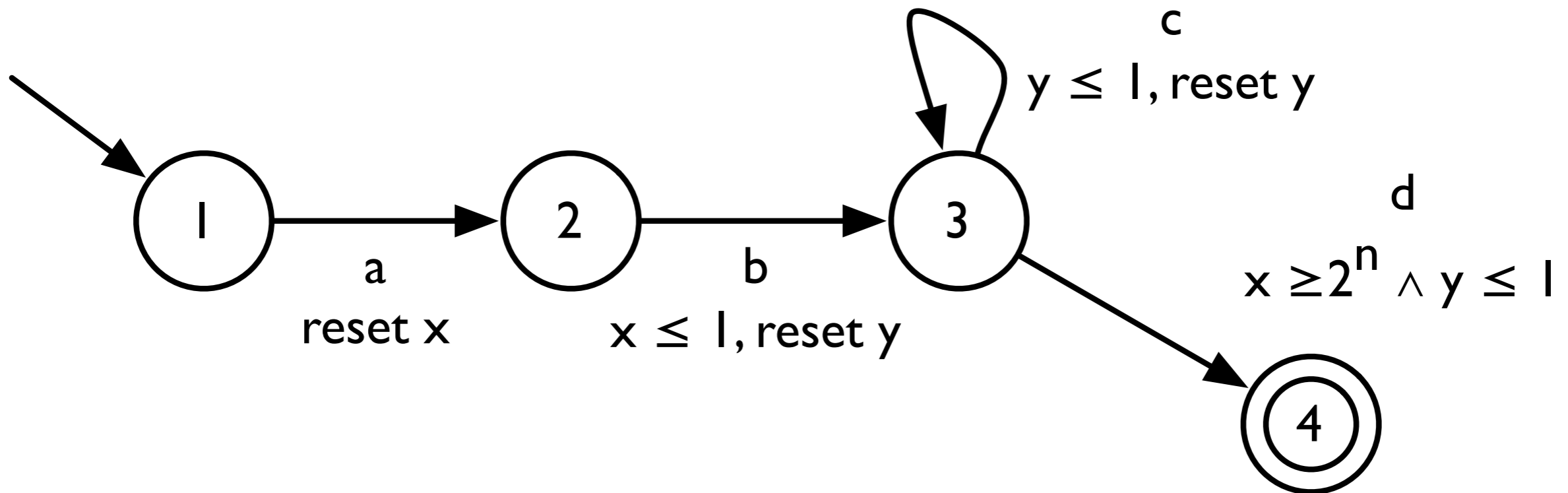
# Efficient Identification

- A language class  $C$  is **efficiently identifiable in the limit** if:
  - there exists a **polynomial time** algorithm that can identify any language  $L$  from  $C$
  - this algorithm is guaranteed to identify the correct language  $L_t$  when the input data contains a **polynomial characteristic set**:
    - a subset of size polynomial in the size of the smallest representation (automaton)  $A$  such that  $L(A) = L_t$

# Polynomial Distinguishability

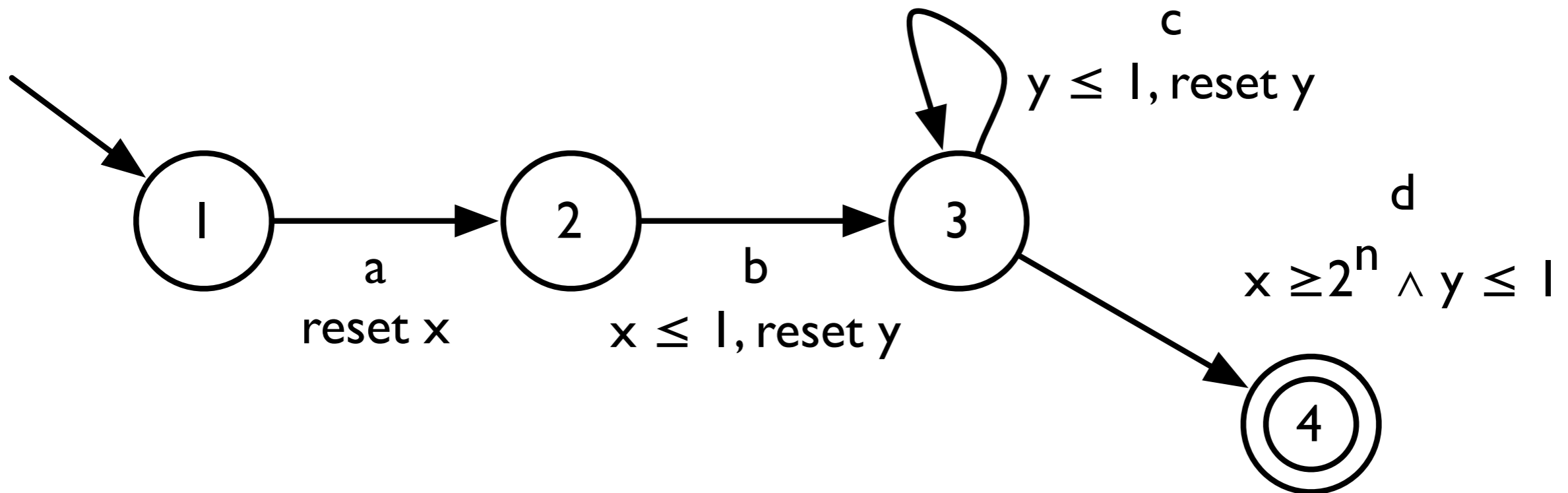
- If  $C$  is **efficiently identifiable in the limit** (from polynomial time and data), then an automaton class  $C_a$  for  $C$  is polynomially distinguishable
- A class of automata  $C_a$  is **polynomially distinguishable** if:
  - for any two automata  $A$  and  $A'$  in  $C_a$ , there exists a string in the **symmetric difference** of  $L(A)$  and  $L(A')$  of size polynomial in  $|A| + |A'|$

# DTAs are not pol. dist.



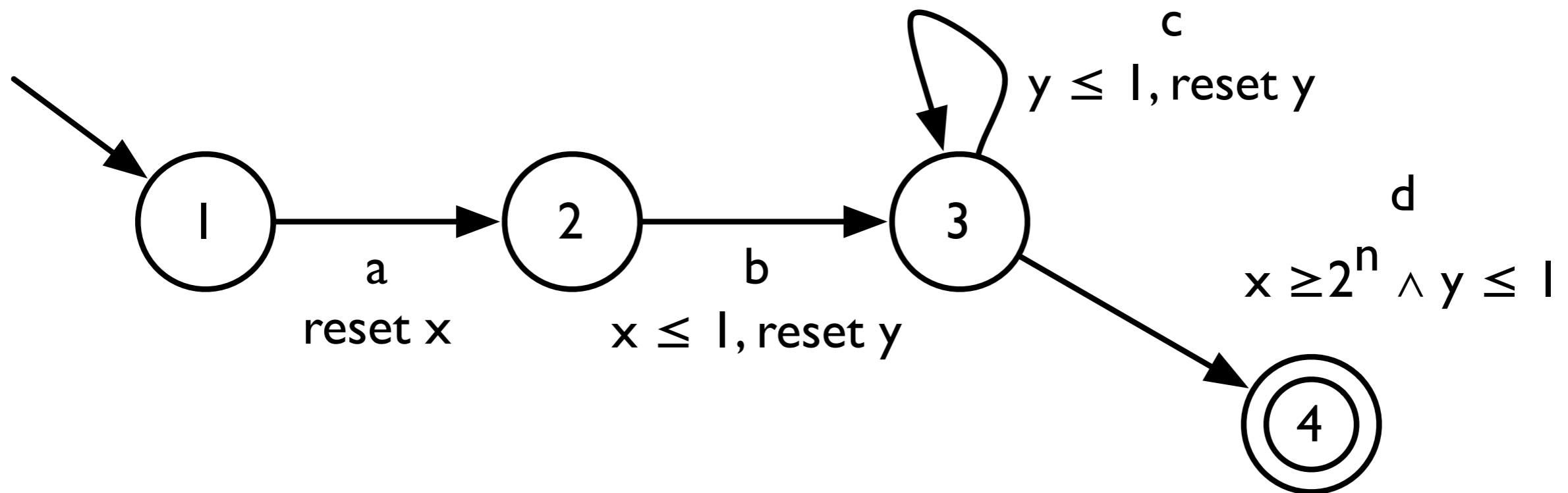
This DTA requires a timed string of exponential length in order to end in **state 4**

# DTAs are not pol. dist.



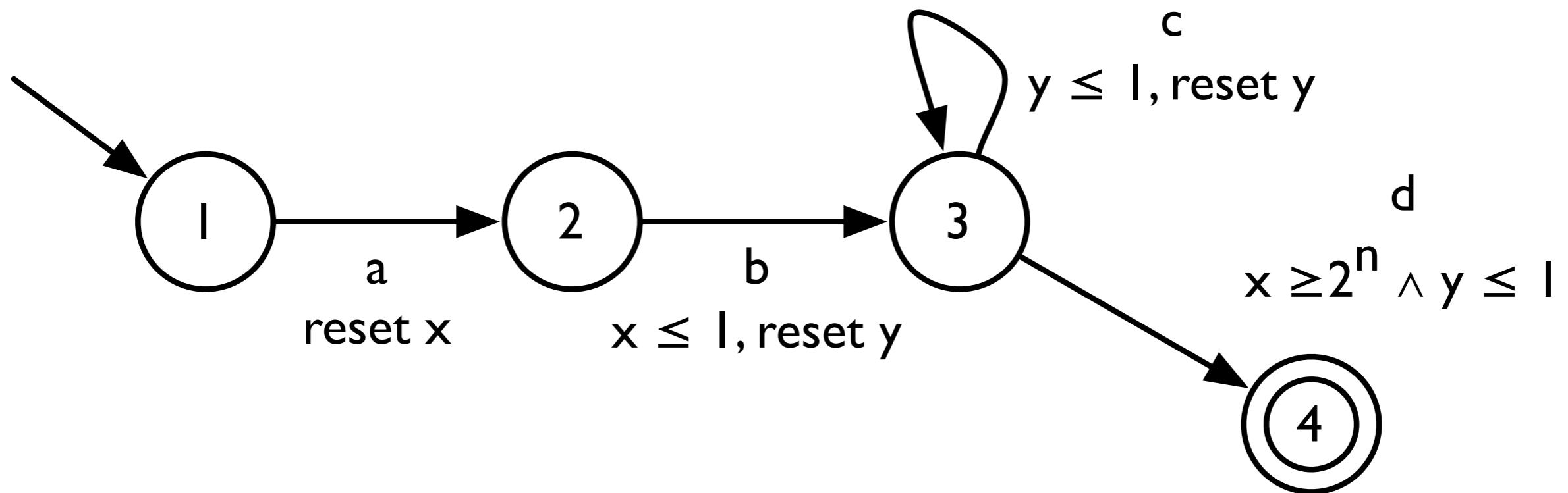
accepts **only**  $(a, t)(b, t')(c, t_1)\dots(c, t_m)(d, t'')$  where:  
all  $t', t'', t_1, \dots, t_m \leq 1$  and  $\sum t_1, \dots, t_m \geq 2^n$ ,  
hence, it **has to hold** that:  $m \geq 2^n$

# DTAs are not pol. dist.



We cannot polynomially bound the size of the **shortest string** that distinguishes these DTAs (for different  $n$ ) from a DTA accepting the empty language

# 2-DTAs are not pol. dist.



These DTAs **only** require 2 clocks!

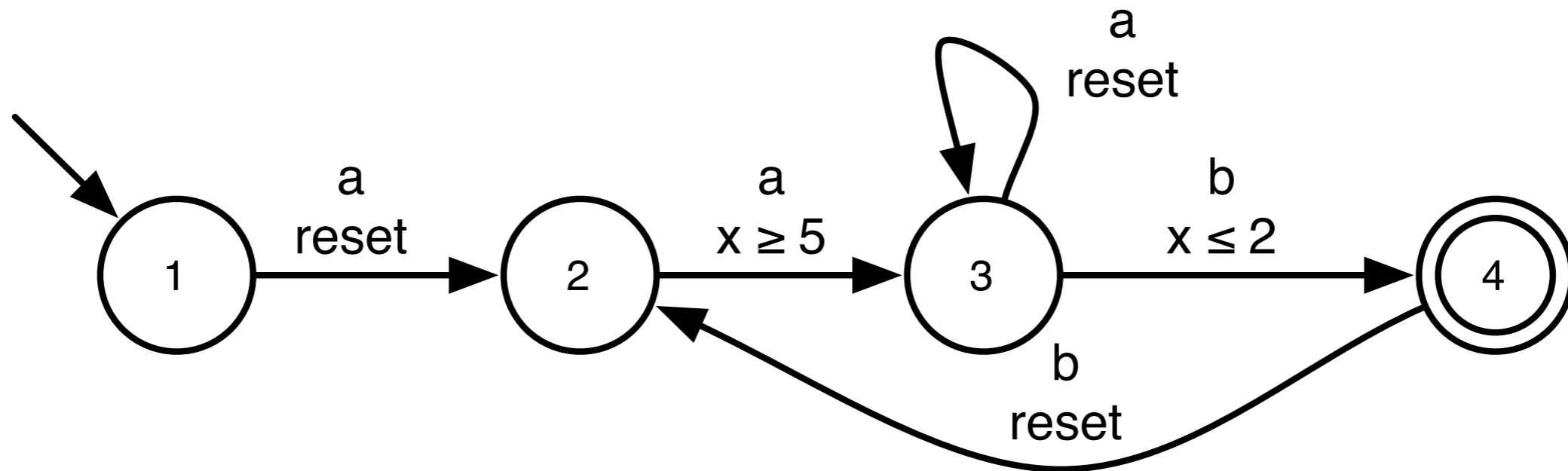
# Overview

- Deterministic timed automata (DTAs)
- Why learn DTAs?
- Efficient identification in the limit
- DTAs are not efficiently identifiable
- **Learning DTAs with a single clock efficiently**
- **Conclusions and future work**

# I-DTAs

- An I-DTA is a DTA with **one clock**  $x$
- The DTAs we used to prove the non-polynomial distinguishability of DTAs require **at least two clocks**
- **I-DTAs are polynomially distinguishable!** (ICGI 2008)
- Are they **efficiently identifiable?**

# I-DTAs

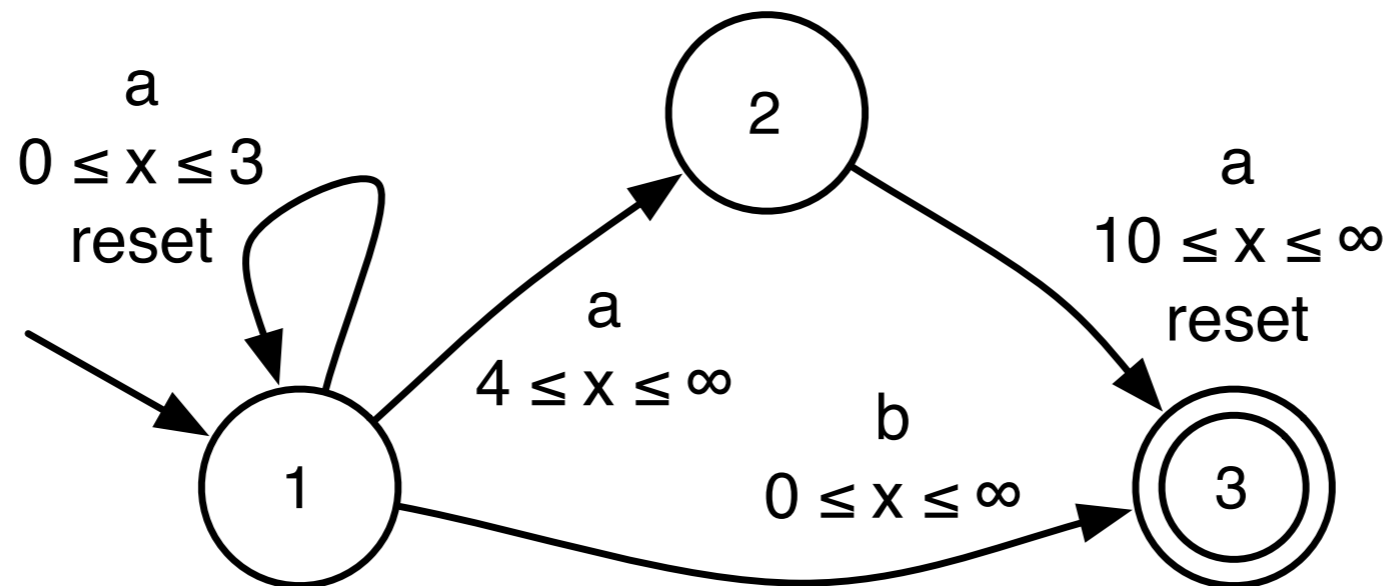


rejects:  $(a, t)(a, t')(b, t'')$  for **any**  $t, t', t''$   
because  $x$  is greater than 5  
the first time it enters state 3

# Learning I-DTAs

- Based on **RPNI**, an algorithm for learning DFA
- Learn I-DTA one **transition**  $\langle q, q', a, r, g \rangle$  at a time:
  - The source state  $q$
  - The target state  $q'$
  - The transition label  $a$
  - The cock reset  $r$
  - The clock guard  $g$

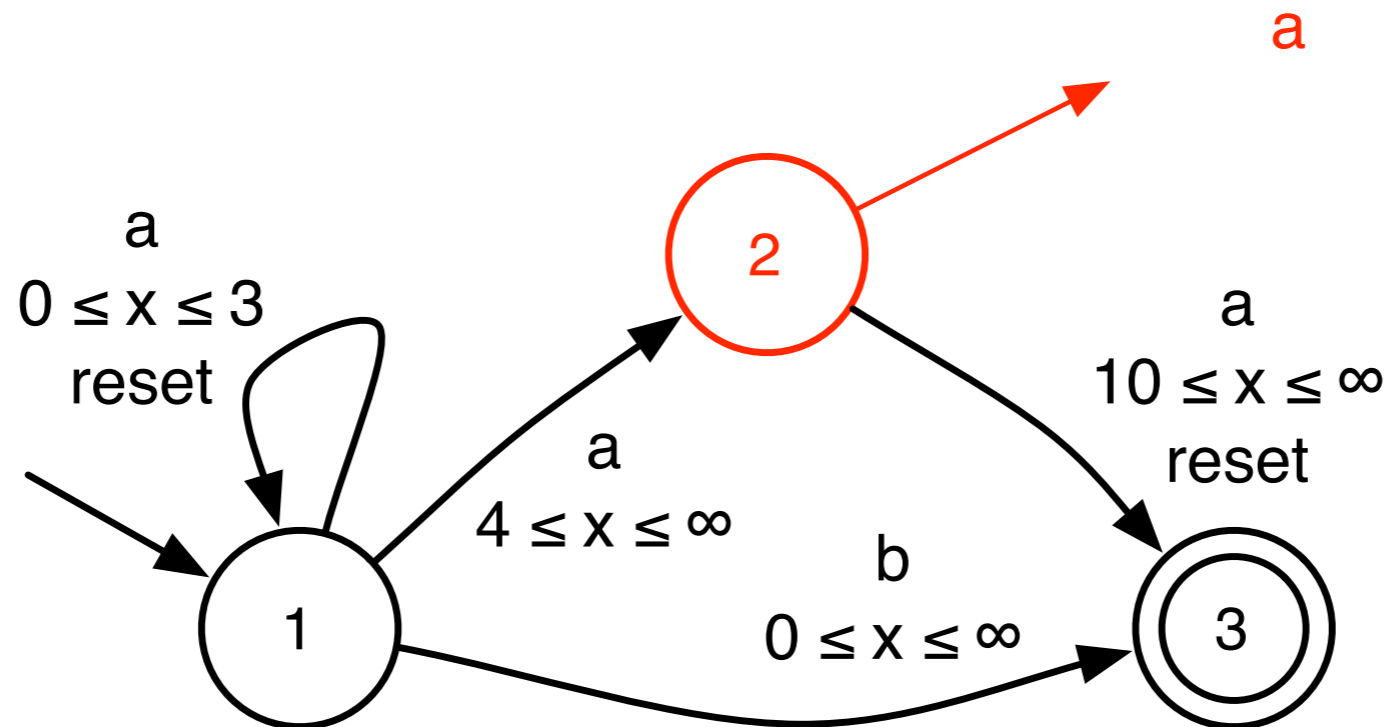
# Learning Example



$S_+ \supseteq \{(a,4)(a,6); (a,5)(b,6); (b,3)(a,2); (a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$

$S_- \supseteq \{(a,3)(a,10); (a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$

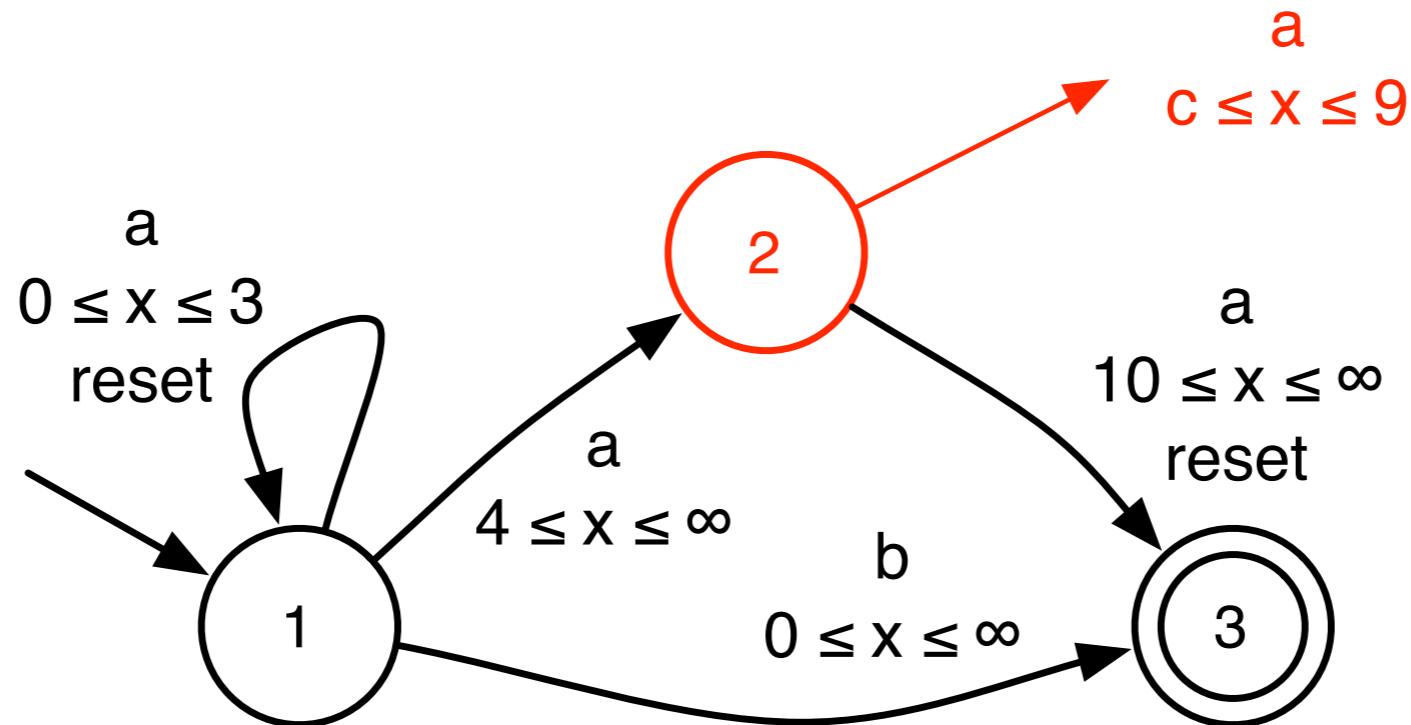
# Learning Example



Choose the source state and transition label:  
smallest number first and alphabetic order

Identify a transition for **state 2**, with **label a**

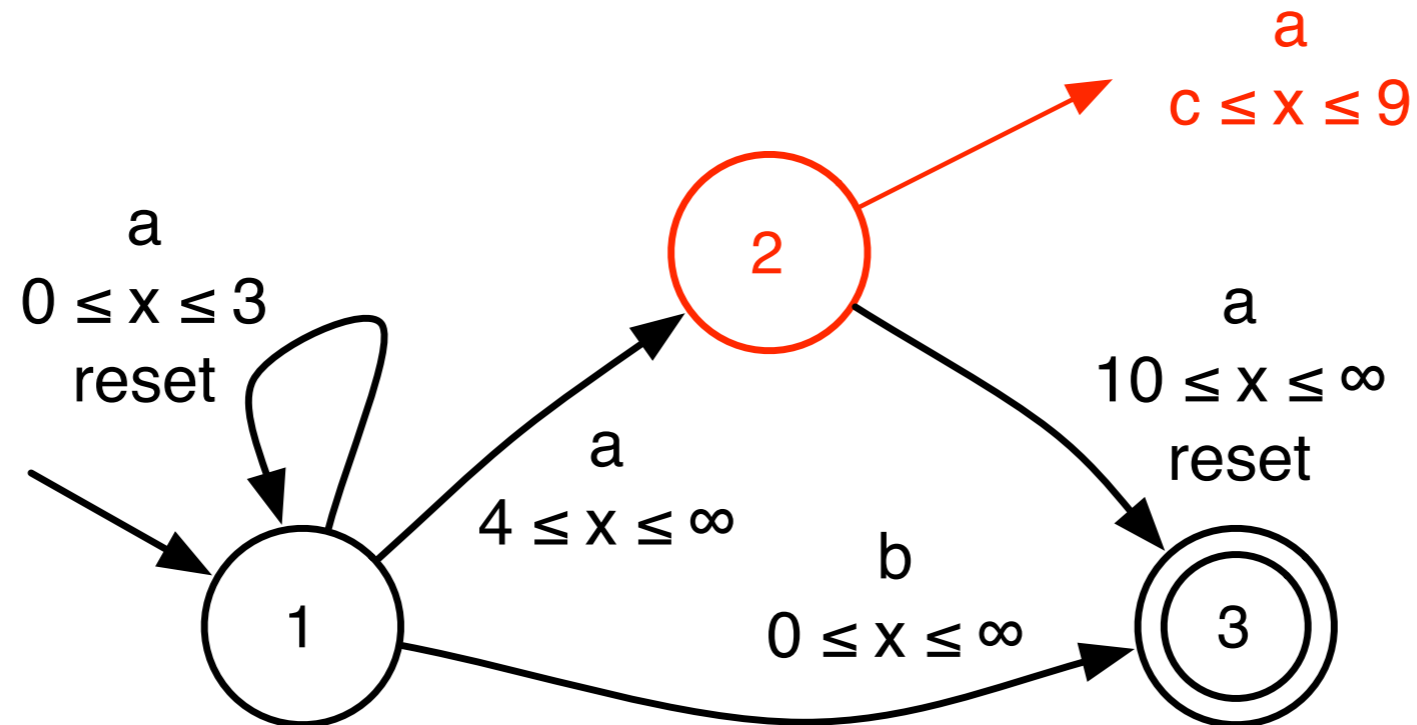
# Learning Example



Choose the maximum upper bound for  $g$ , in this case 9  
Use the data set to determine the **smallest consistent lower bound**  $c$

$$g = c \leq x \leq 9$$

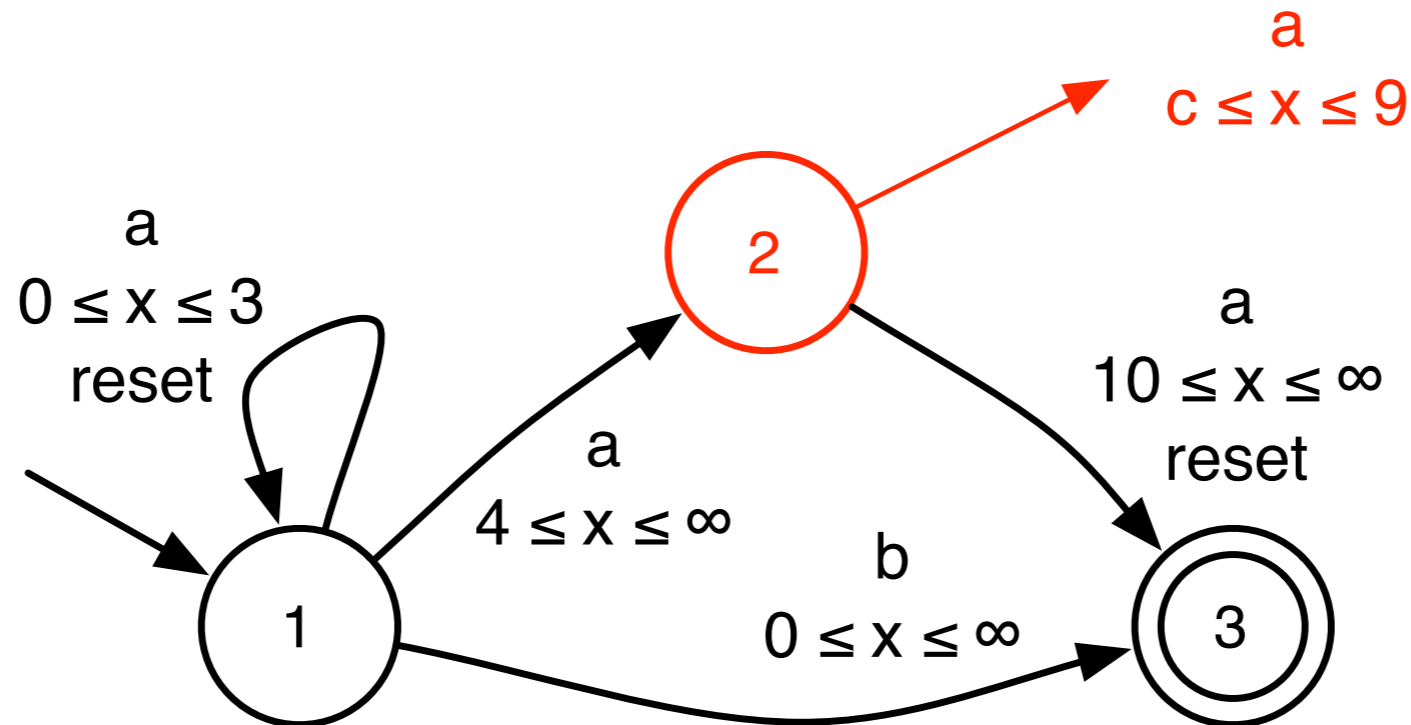
# Learning Example



The smallest reachable lower bound is 4

$$4 \leq c \leq 9$$

# Learning Example

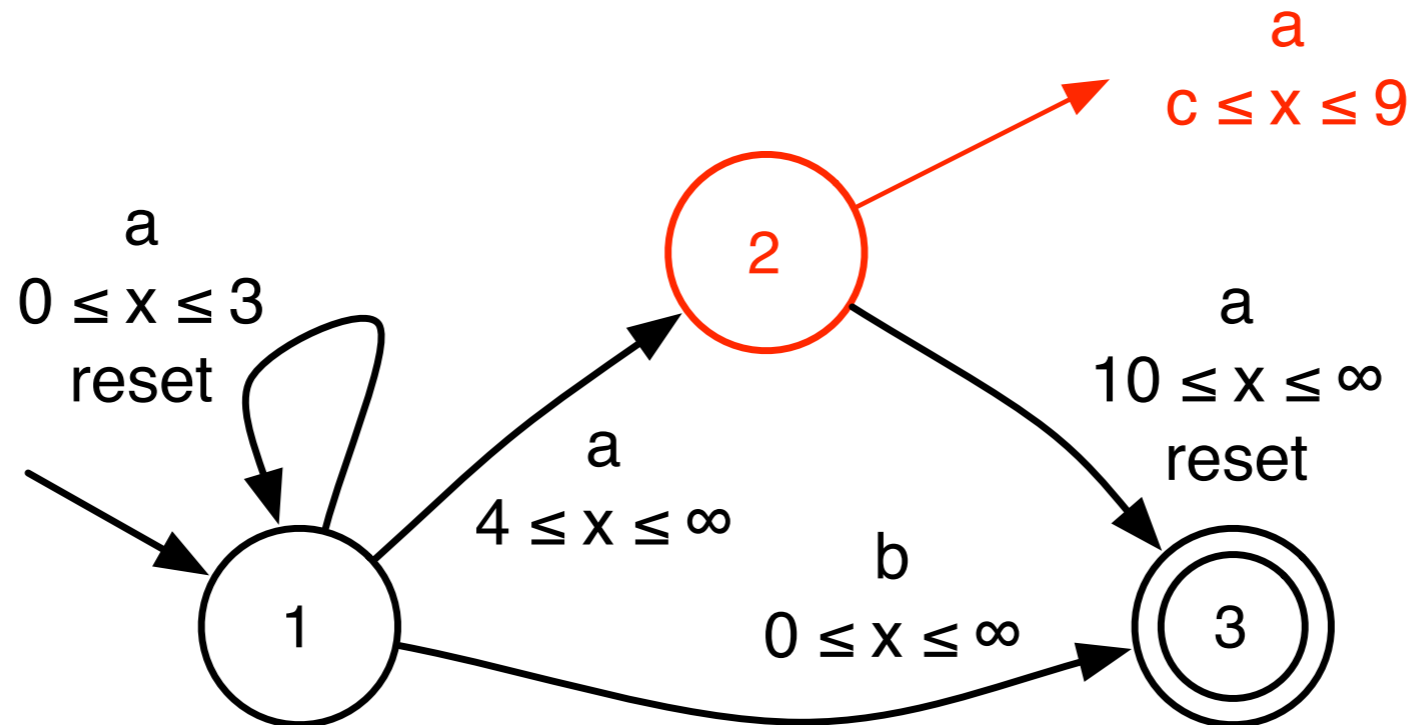


$S_+ \supseteq \{(a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$

$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$

will potentially use the transition

# Learning Example

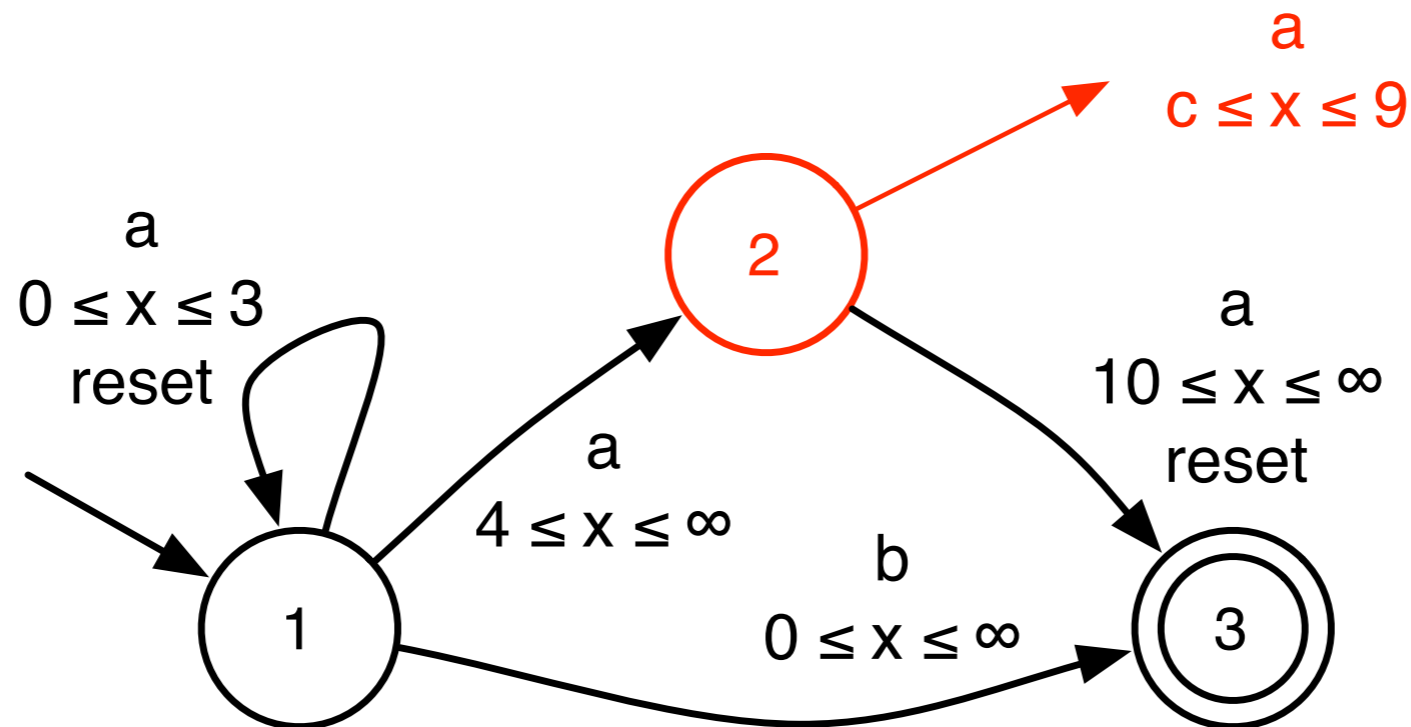


$$S_+ \supseteq \{(a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$$

$$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$$

$$c \in \{4, 5, 6, 7, 8, 9\}$$

# Learning Example



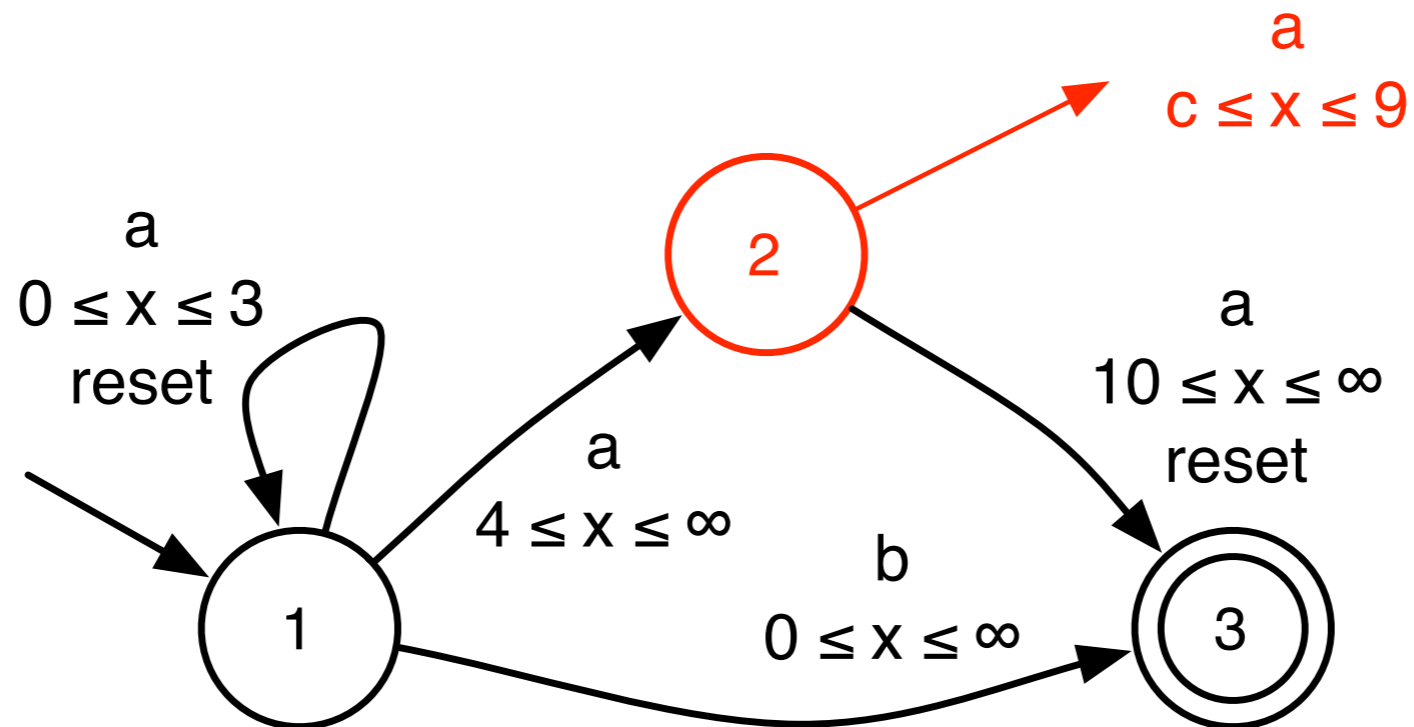
$S_+ \supseteq \{(a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$

$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$

inconsistency

$c \in \{4, 5, 6, 7, 8, 9\}$

# Learning Example

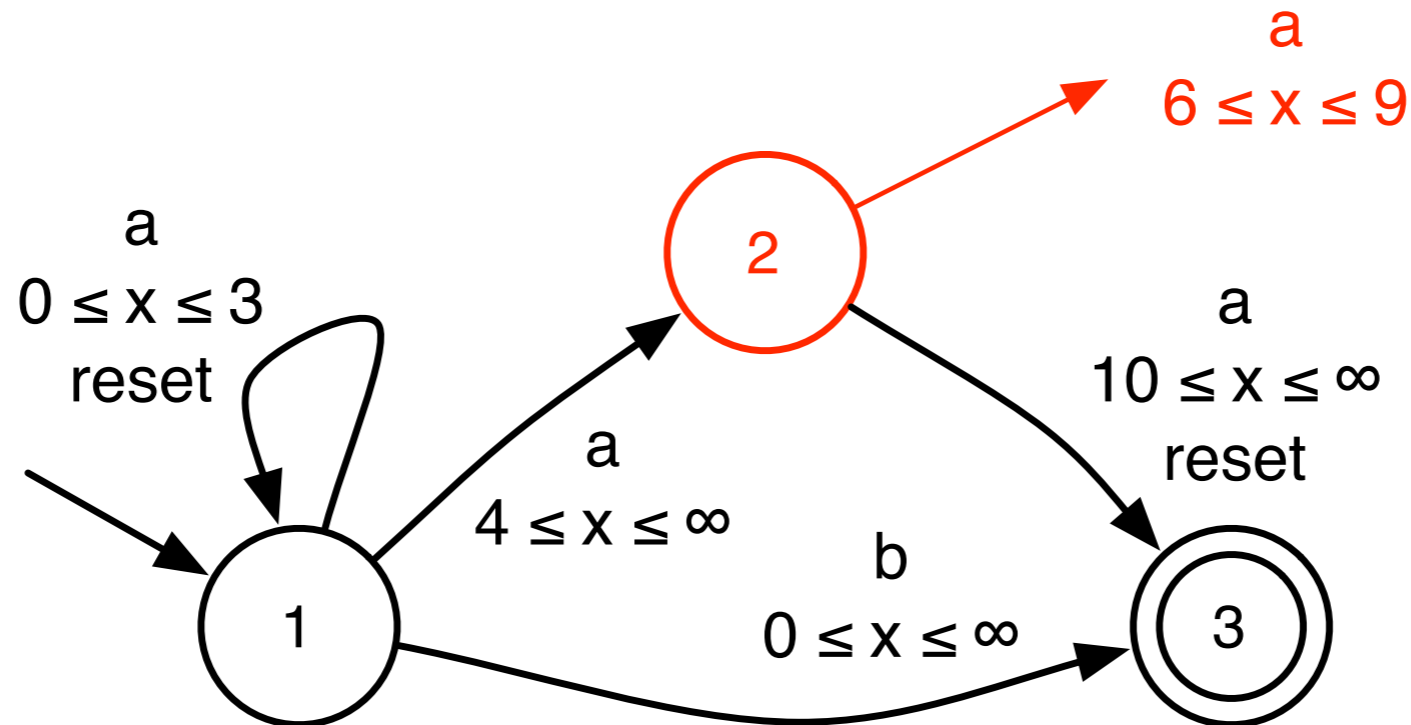


$$S_+ \supseteq \{(a,4)(a,2)(a,2)(b,3)\}$$

$$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$$

$$c \in \{6, 7, 8, 9\}$$

# Learning Example



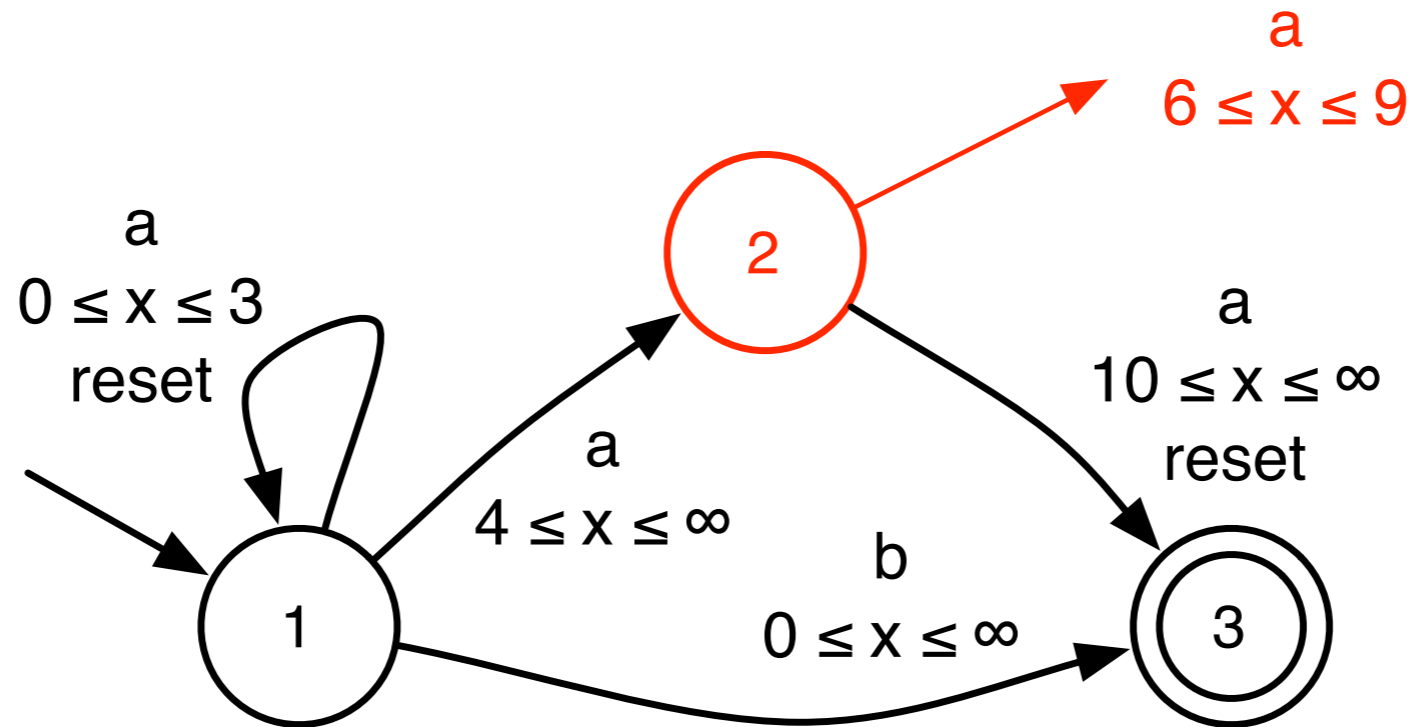
$$S_+ \supseteq \{(a,4)(a,2)(a,2)(b,3)\}$$

$$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$$

no inconsistency

$$c \in \{6, 7, 8, 9\} \rightarrow c = 6$$

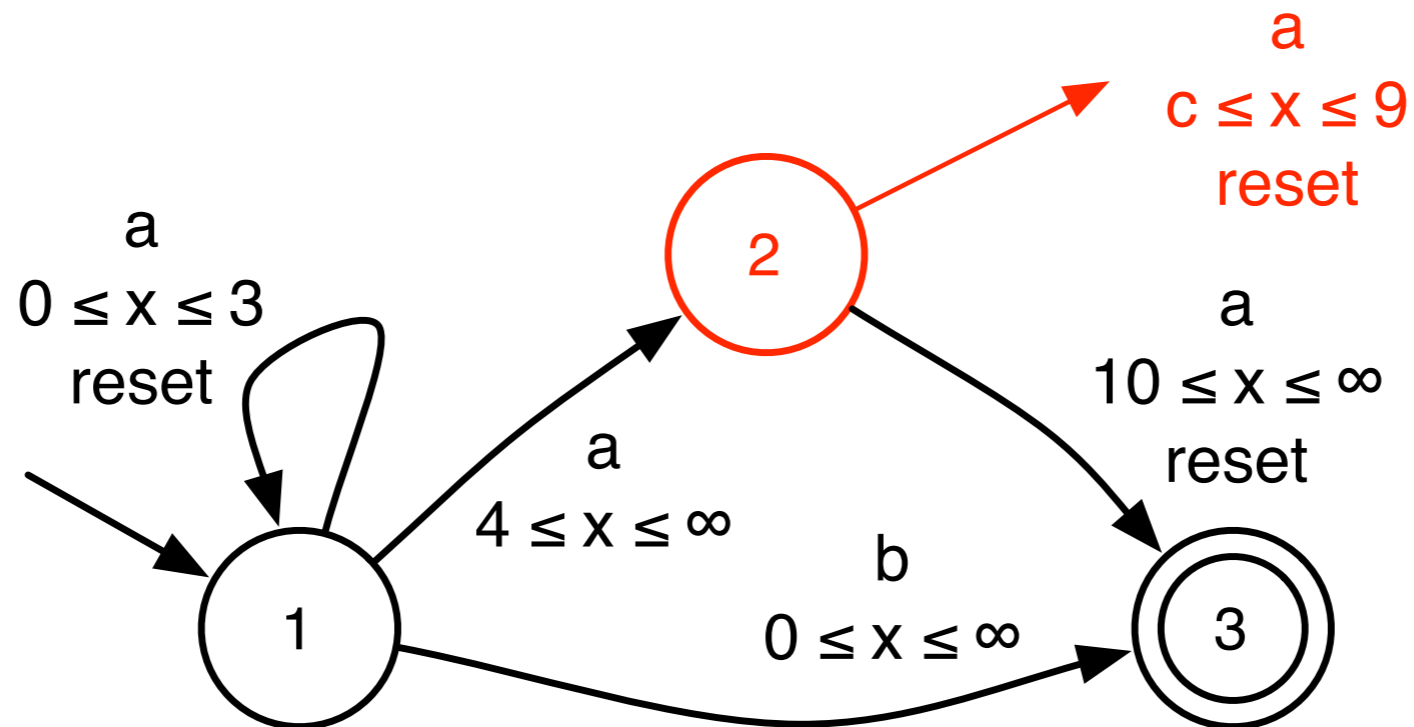
# Learning Example



The identified guard is  $g = 6 \leq x \leq 9$   
if the clock is not reset

Perform the same algorithm for identifying  $g$  if the **clock is reset**

# Learning Example

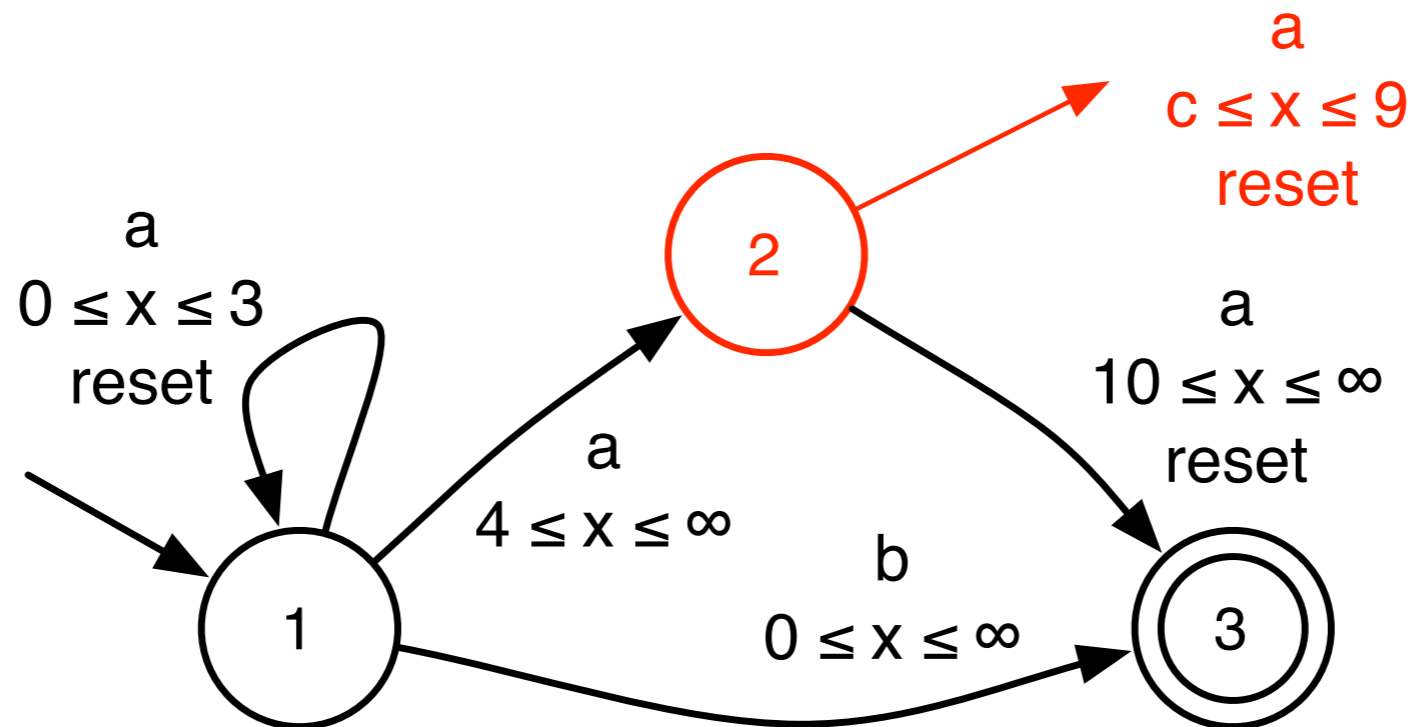


$$S_+ \supseteq \{(a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$$

$$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$$

$$c \in \{4, 5, 6, 7, 8, 9\}$$

# Learning Example



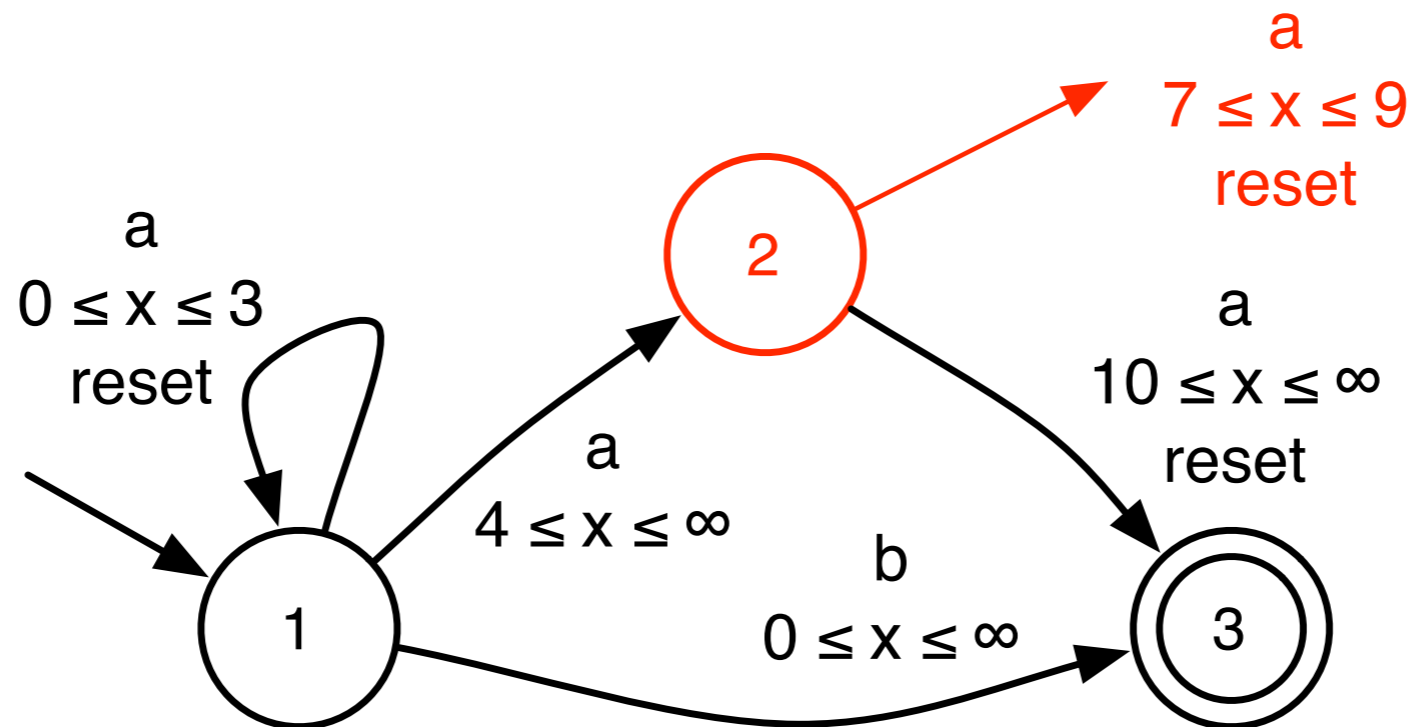
$S_+ \supseteq \{(a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$

$S_- \supseteq \{(a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$

inconsistency if the clock is reset

$c \in \{4, 5, 6, 7, 8, 9\}$

# Learning Example



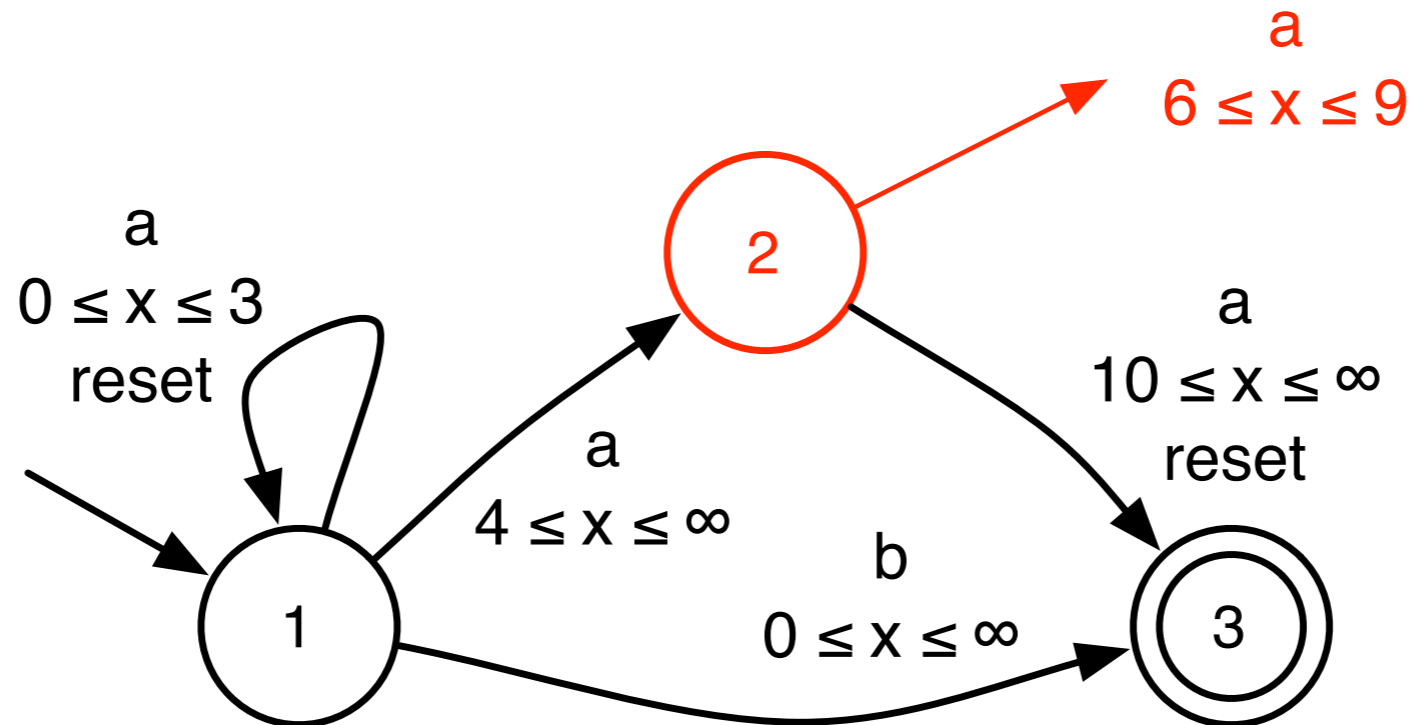
$$S_+ \supseteq \{ \}$$

$$S_- \supseteq \{ (a,4)(a,3)(a,2)(b,3); (a,5)(a,3) \}$$

no inconsistency if the clock is reset

$$c \in \{ 7, 8, 9 \} \rightarrow c = 7$$

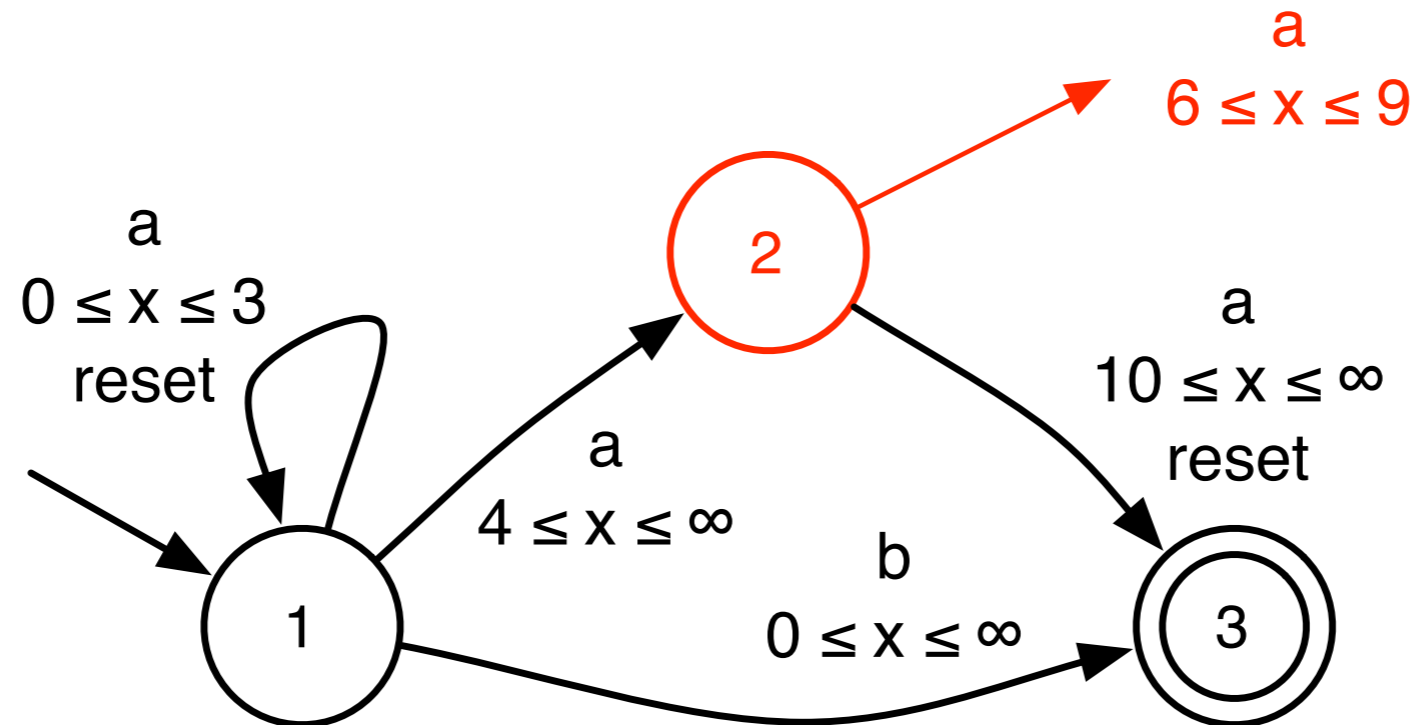
# Learning Example



The identified guard is  $g = 7 \leq x \leq 9$   
if the clock is reset

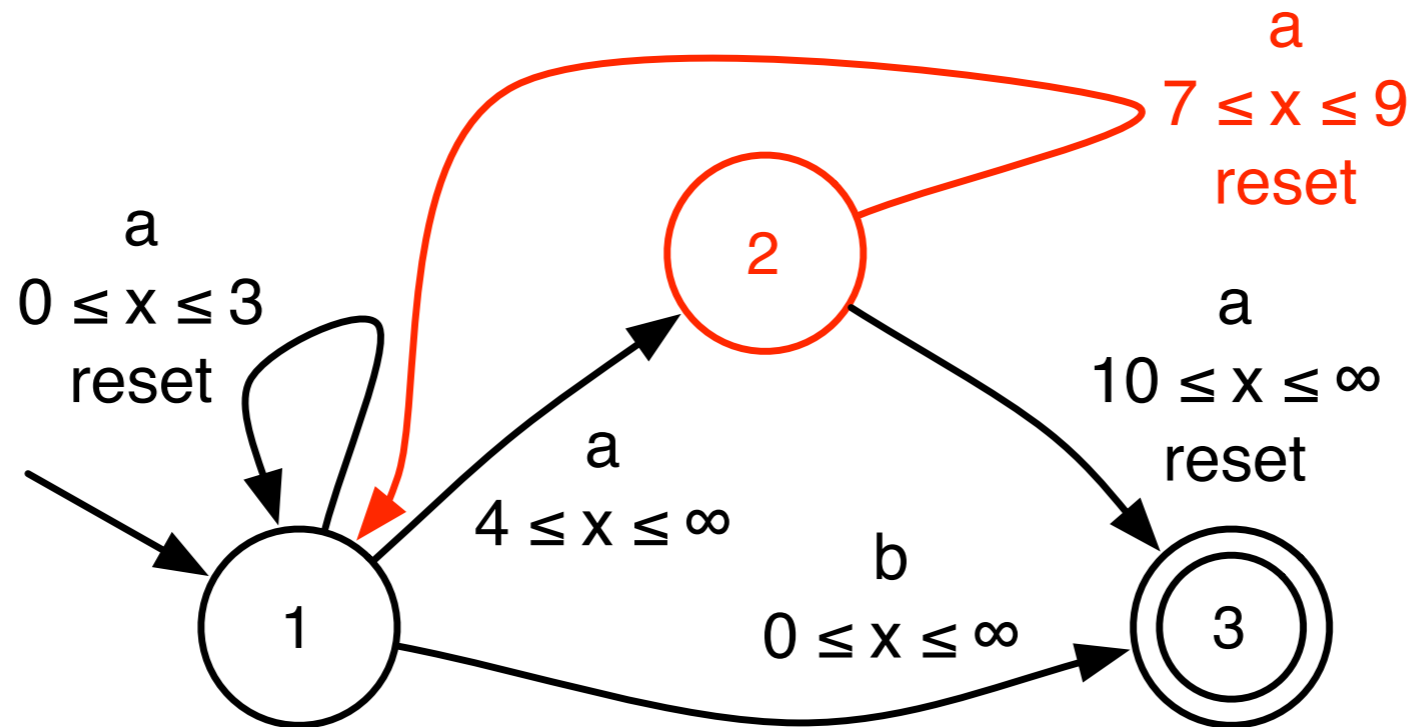
Choose the **smallest lower bound**:  
 $g = 6 \leq x \leq 9$  and the clock is not reset

# Learning Example



Choose the first consistent target state  
(identical to **RPNI**)

# Learning Example

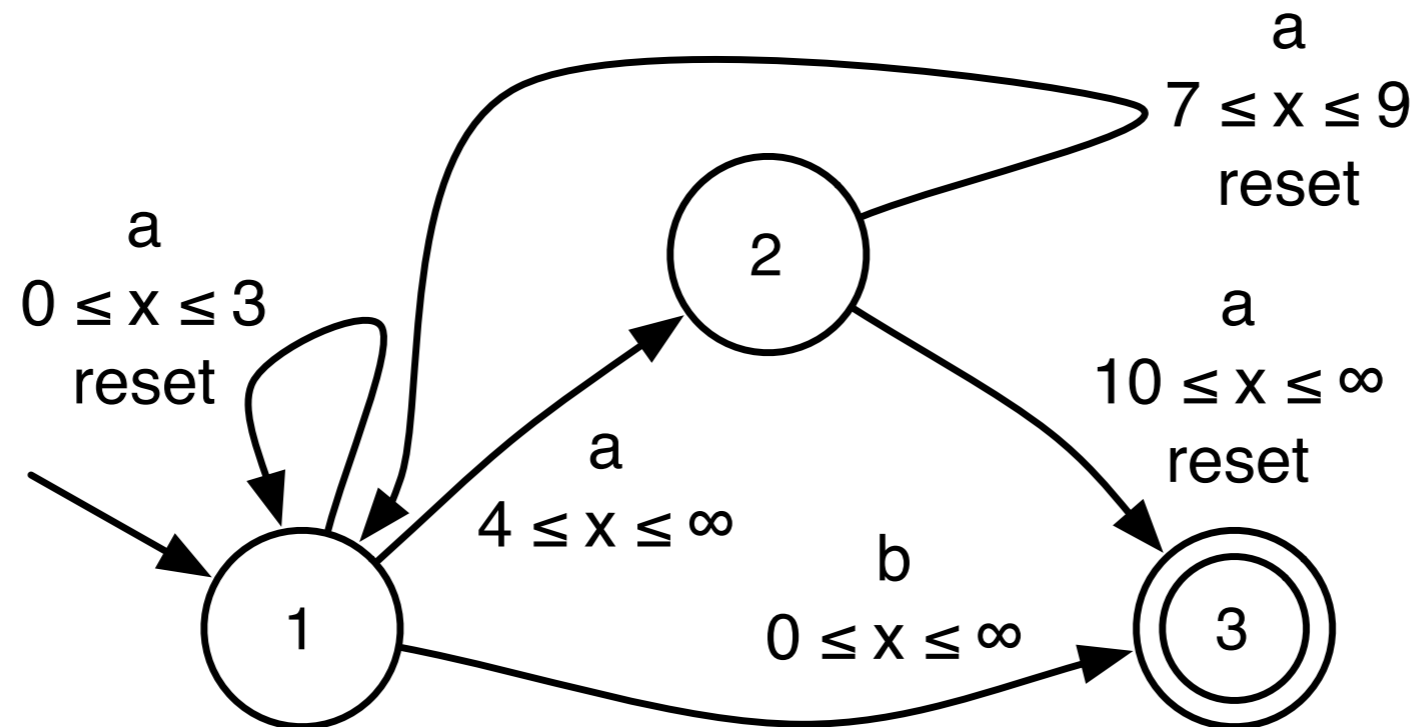


$S_+ \supseteq \{(a,4)(a,6); (a,5)(b,6); (b,3)(a,2); (a,4)(a,1)(a,3); (a,4)(a,2)(a,2)(b,3)\}$

$S_- \supseteq \{(a,3)(a,10); (a,4)(a,2)(a,2); (a,4)(a,3)(a,2)(b,3); (a,5)(a,3)\}$

no inconsistency if  $q' = 1$

# Learning Example



We identify:

$\langle q = 2, q' = 1, s = a, r = \text{false}, g = 6 \leq x \leq 9 \rangle$

and **iterate**

# Learning I-DTAs

- Identify transitions in a **fixed order**
- Use data to determine the smallest consistent **clock guard** and **clock reset**
- Use data to determine the first consistent target state, like **RPNI**
- **Iterate** until no new transitions can be identified

# Properties

- The algorithm is **polynomial time** because:
  - Identifying a single transition takes polynomial time
  - Every transition is fired by **at least one string**
- Identifying a I-DTA requires **polynomial data** because:
  - Inconsistent pairs exist for every incorrect decision
  - I-DTAs are polynomially distinguishable, hence:
    - The inconsistent pairs are of **polynomial length**
    - Reaching all possible behaviors of a I-DTA requires a **polynomial number of transitions**

# Learning I-DTAs

- Theorem:
  - I-DTAs are efficiently identifiable in the limit

# Conclusions

- DTAs are an **intuitive** representation for real-time systems
- They are more **compact** (efficient) than DFA or HMM representations of the same systems
- Unfortunately, DTAs can in general not be identified efficiently
- **I-DTAs are efficiently identifiable in the limit**, and still a lot more compact than DFAs or HMMs

# Future work

- Find classes of DTAs with **multiple clocks** that are polynomially distinguishable
- Determine whether DTA learning algorithms exist that identify DTAs efficiently **in the size of the smallest I-DTA representation**
- Learn one-clock timed automata in order to perform **model-based testing**
- Write an **evidence driven** variation of this algorithm and test it on real data