# Cyber-Physical Systems (CSC.T431)

## Timed Model (2)

Instructor: Takuo Watanabe (Department of Computer Science)

# Agenda

- Timed Model (2)

Course Support & Material

- Slides: OCW-i
- Course Web: https://titech-cps.github.io
- Course Slack: titech-cps.slack.com

# Timed Automata

- Consider a property $\varphi$ over the state variables of a timed process $TP$.
- To check whether $\varphi$ is an invariant of $TP$, we would like to perform a reachability analysis as in synchronous/asynchronous models.

- Problem: real-valued clock variables
  - Uncountably many variations of timed actions seem to make the on-the-fly DFS algorithm for invariant verification impossible.

# Restrictions on the Use of Clock Variables

- To make algorithmic analysis possible, we restrict the use of clock variables.

- Assignment
  - For any clock variable $x$, the only possible assignment is the form of $x := 0$.
- Guards and Clock Invariants
  - For any clock variable $x$, atomic expressions involving $x$ should be the form of $x \leq k$ or $x \geq k$ where $k$ is an integer constant.
    - Note: $x = k$ can be expressed as $x \leq k \wedge x \geq k$, and $x < k$ can be expressed as $\neg(x \geq k)$.

# Timed Automata

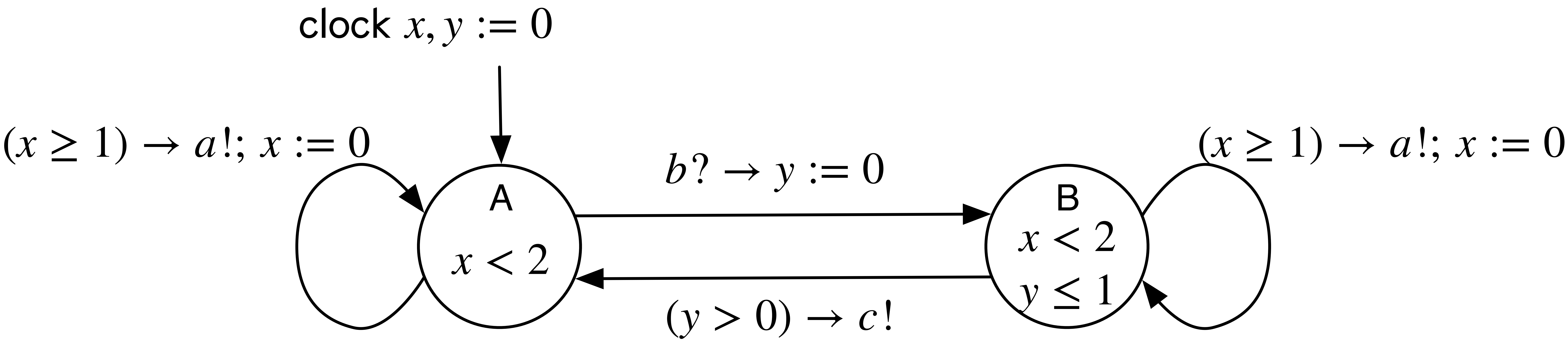
- A timed process $TP$ is said to be a *timed automaton* if for every clock variable $x$,
  1. the only assignment to $x$ occurring the update description of any of the tasks of $TP$ is of the form $x := 0$, and
  2. each atomic expression involving $x$ occurring either in the clock invariant of $TP$, or in the guards or update descriptions of any of the tasks of $TP$, is of the form $x \leq k$ or $x \geq k$, where $k$ is an integer constant.
- For the convenience of the analysis, we use a pair $(s, v)$ to denote a state of a timed automaton, where $s$ and $v$ are valuations of non-clock (discrete) variables and clock variables respectively.

# Timed Automata

- Timed automata are closed under parallel composition.
  - If $TP_1$ and $TP_2$ are timed automata, then $TP_1 \,|\, TP_2$ is also a timed automaton.

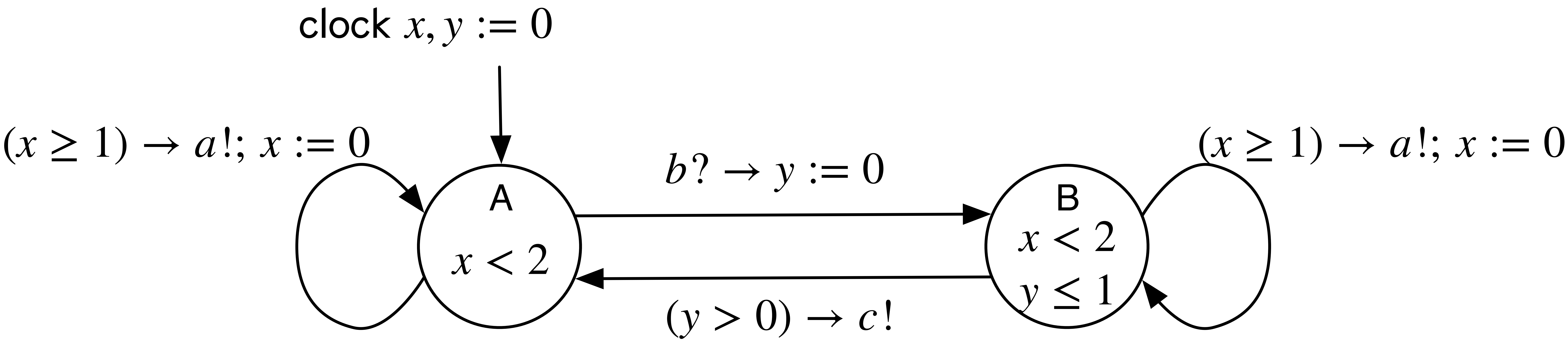
- Finite Timed Automata:
  - A timed automation where all variables / channels other than clock variables have finite types (such as Boolean, enumerate types)

# Region Equivalence



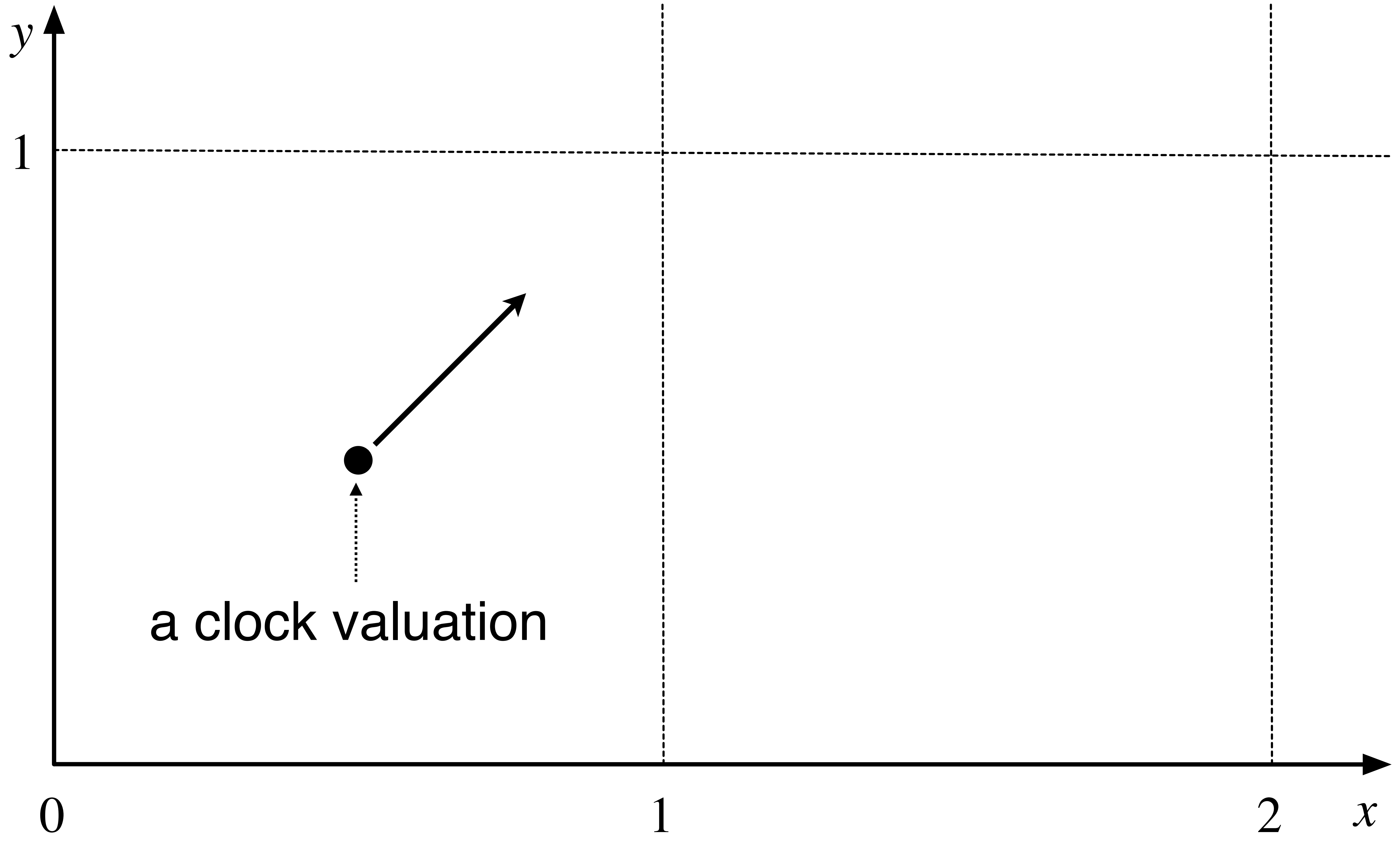

- $b$ : input channels, $a$, $c$ : output channel
- The process issues $a$ periodically with a period in $[1, 2)$.
- Whenever the process receives $b$, it issues $c$ with a delay in $(0,1]$.

# Region Equivalence



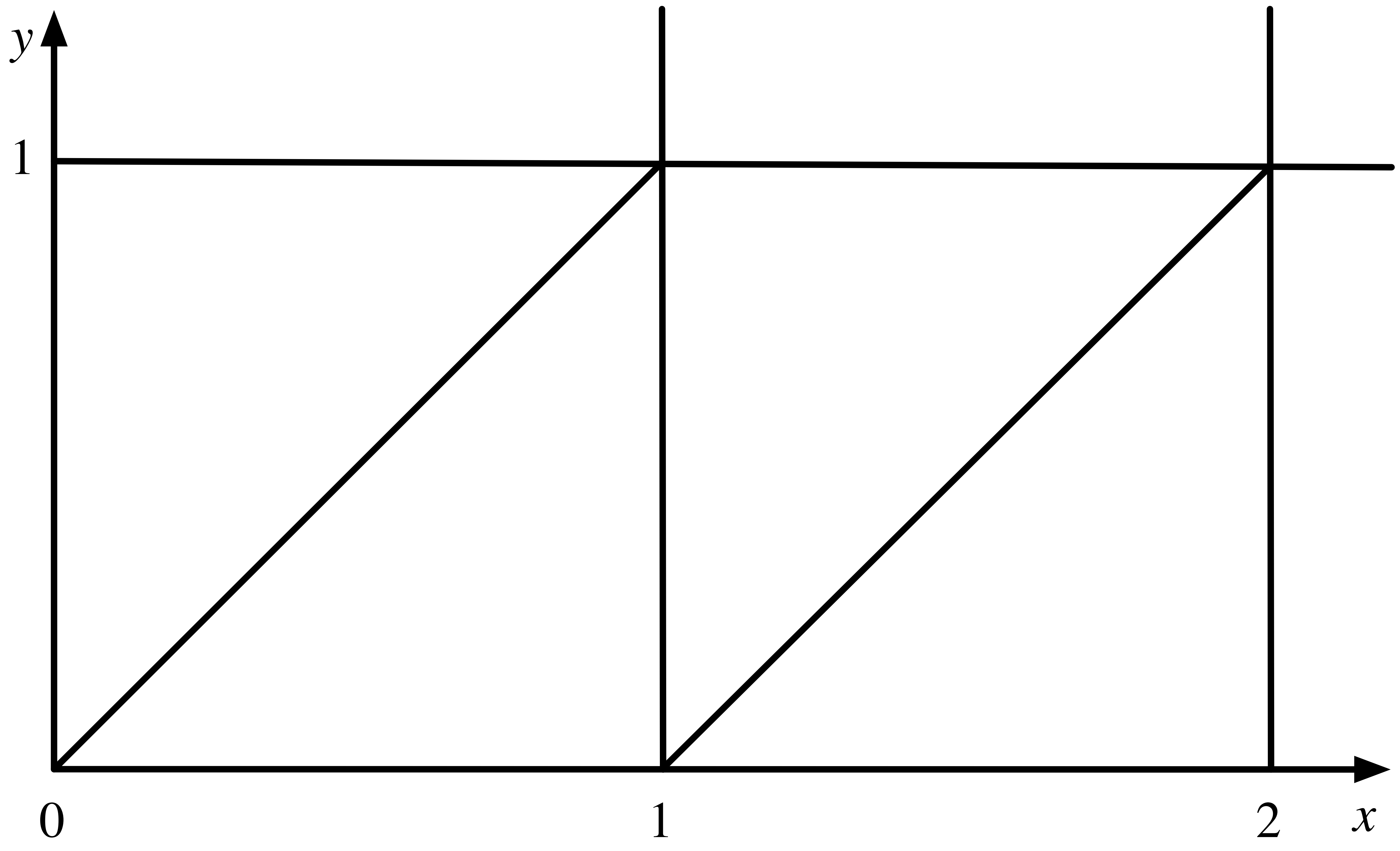

- The state space is (uncountably) infinite.
- Idea: *Region Equivalence*
    - Partitions the clock valuations into finitely many equivalence classes so that equivalent states behave similarly.

# Regions



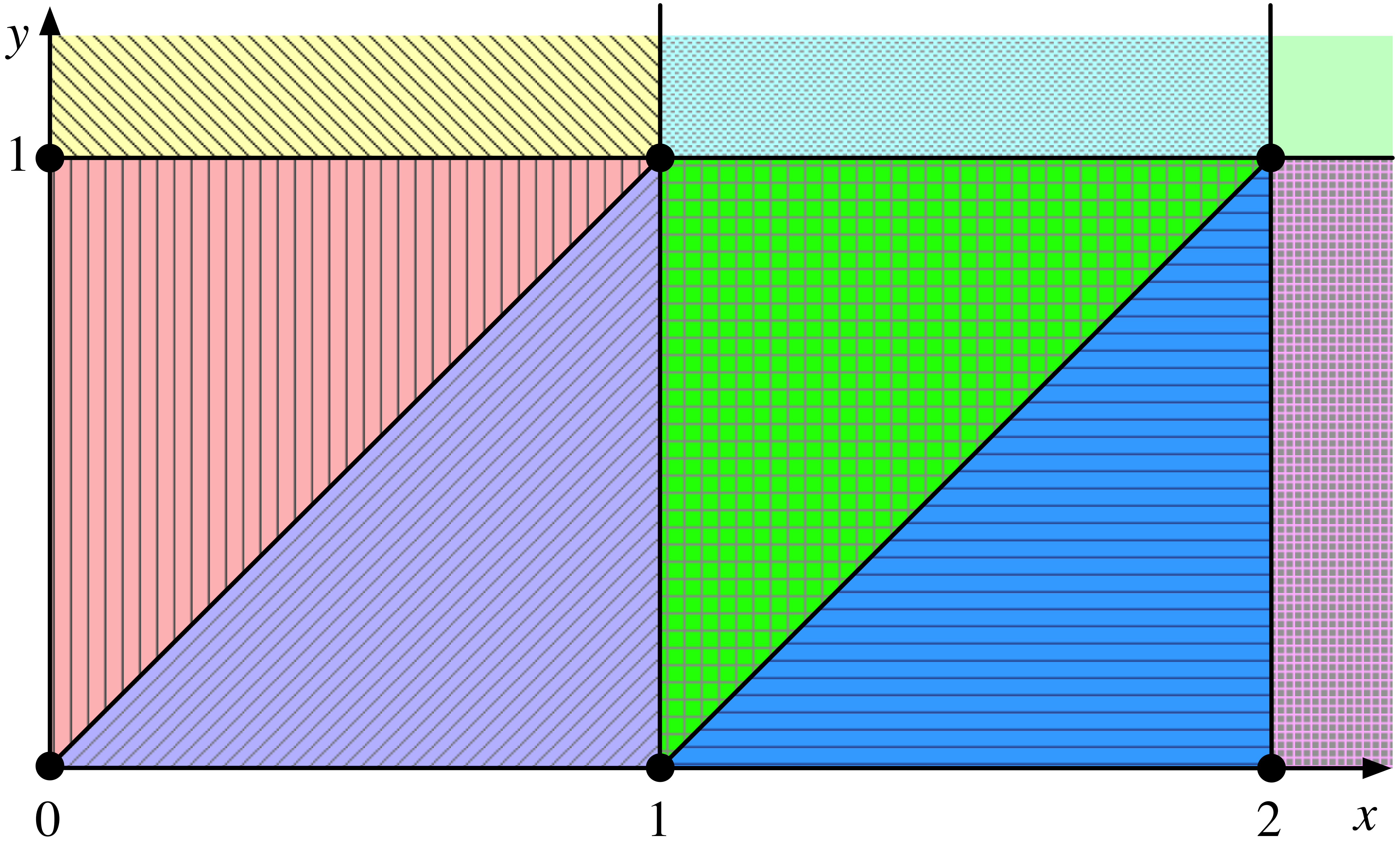

- A clock valuation in the example can be expressed as a point in the first quadrant of the two-dimensional x/y-plane.
- A clock valuation evolves along the diagonal direction.

# Regions



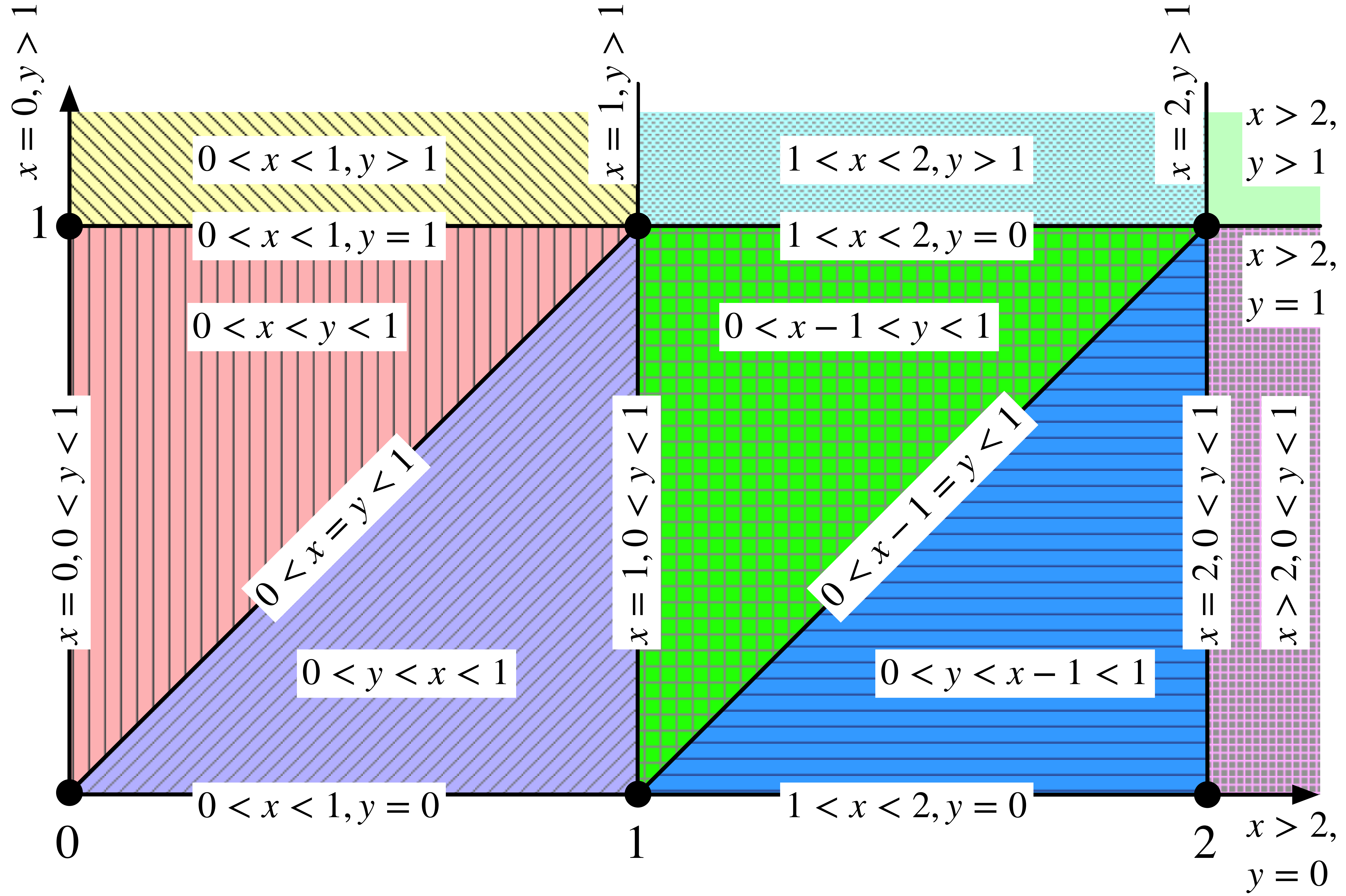

- In mode A, B
  - If $x < 1$, $a$ cannot happen.
  - If $1 \leq x < 2$, $a$ may happen.
- In mode B
  - If $y = 0$, $c$ cannot happen
  - If $0 < y \leq 1$, $c$ may happen
  - If $x - 1 < y$, then $y$ reaches 1 before $x$ reaches 2
- ... similar arguments may be applied to partition the quadrant.

# Regions



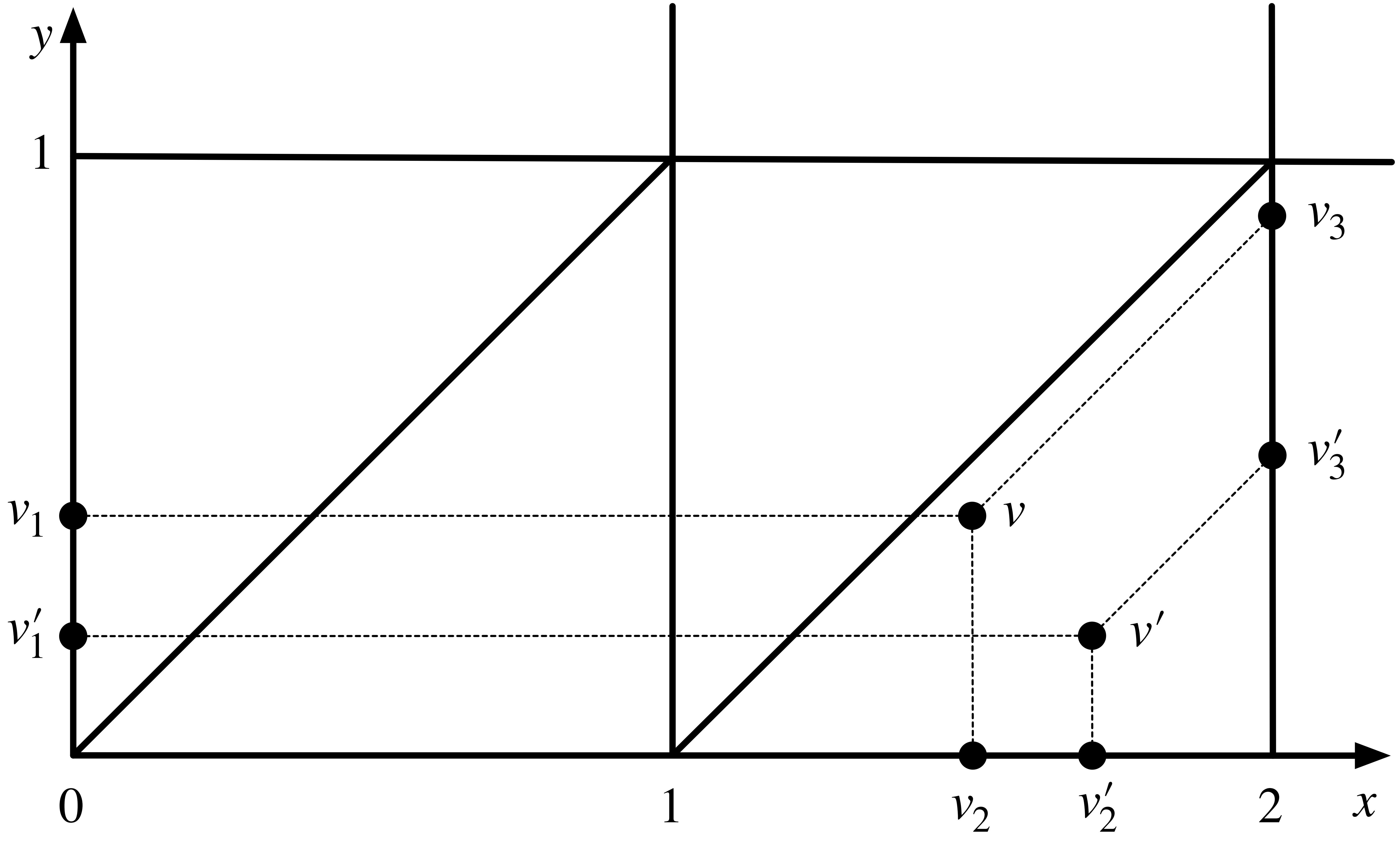

- We can divide the quadrant into 28 partitions:
  - 6 grid points
  - 14 open line segments
    - 9 bounded line segments
    - 5 unbounded line segments
  - 8 open regions
    - 4 triangular regions
    - 4 unbounded regions

# Regions



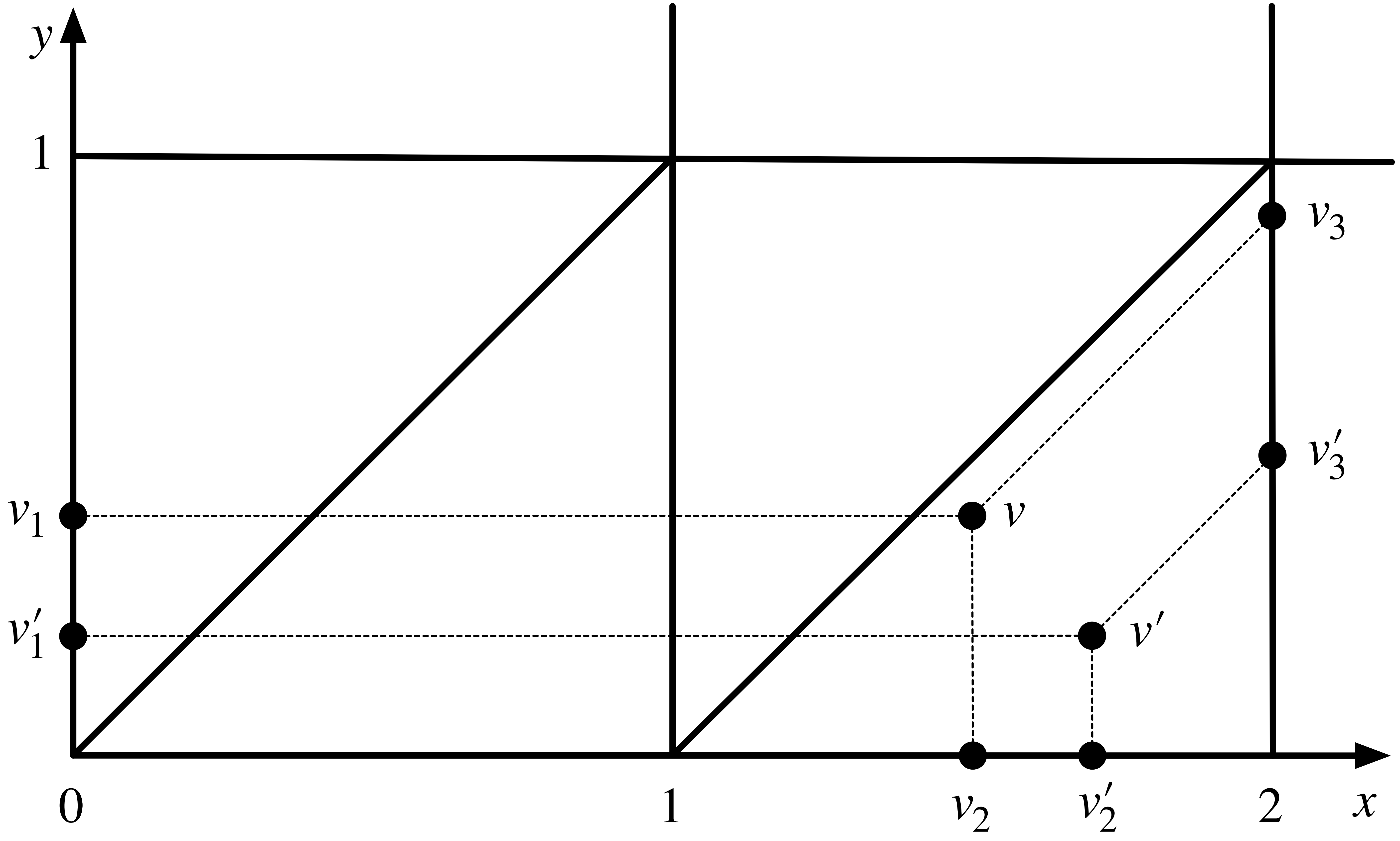

- We can divide the quadrant into 28 partitions:
  - 6 grid points
  - 14 open line segments
    - 9 bounded line segments
    - 5 unbounded line segments
  - 8 open regions
    - 4 triangular regions
    - 4 unbounded regions

# Region Equivalence




- Two clock valuations *region-equivalent* if they belong to the same partition.
  - Ex. $v$ and $v'$ are region-equivalent.
  - A mode-switch from B to A maps $v$ and $v'$ to $v_2$ and $v_2'$ which are in the same partition.
  - Timed-action eventually maps $v$ and $v'$ to $v_3$ and $v_3'$ which are also in the same partition.

# Region Equivalence




- Two region-equiv. states behave the same.
  - For example, if a mode-switch is enabled in $(\mathsf{A}, v)$, then it is also enabled in $(\mathsf{A}, v')$.
  - The resulting clock valuations after the mode-switch belong to the same partition.

# Region Equivalence

- A possible execution:

$$(\mathsf{A}, 0, 0) \xrightarrow{0.6} (\mathsf{A}, 0.6, 0.6) \xrightarrow{b?} (\mathsf{B}, 0.6, 0) \xrightarrow{0.5} (\mathsf{B}, 1.1, 0.5) \xrightarrow{c!} (\mathsf{A}, 1.1, 0.5)$$
$$\xrightarrow{0.2} (\mathsf{A}, 1.3, 0.7) \xrightarrow{a!} (\mathsf{A}, 0, 0.7) \xrightarrow{1.25} (\mathsf{A}, 1.25, 1.95) \xrightarrow{0.61} (\mathsf{A}, 1.86, 2.56)$$

- Another possible *similar* execution:

$$(\mathsf{A}, 0, 0) \xrightarrow{0.1} (\mathsf{A}, 0.1, 0.1) \xrightarrow{b?} (\mathsf{B}, 0.1, 0) \xrightarrow{0.91} (\mathsf{B}, 1.01, 0.91) \xrightarrow{c!} (\mathsf{A}, 1.01, 0.91)$$
$$\xrightarrow{0.05} (\mathsf{A}, 1.06, 0.96) \xrightarrow{a!} (\mathsf{A}, 0, 0.96) \xrightarrow{1.25} (\mathsf{A}, 1.25, 2.21) \xrightarrow{0.61} (\mathsf{A}, 1.86, 2.82)$$

- From a pair of region-equivalent states, at every step of execution, the state pair can remain region-equivalent.

# Region Equivalence

## Notation: $k_x$

- Let $x$ be a clock variable of a timed automaton $TP$. We use $k_x$ to denote the largest integer constant that $x$ is compared with in the atomic constraints that appear in a guard, update description, or a clock-invariant in $TP$.
- Ex. In the previous example, $k_x = 2$ and $k_y = 1$.

# Region Equivalence

## Definition

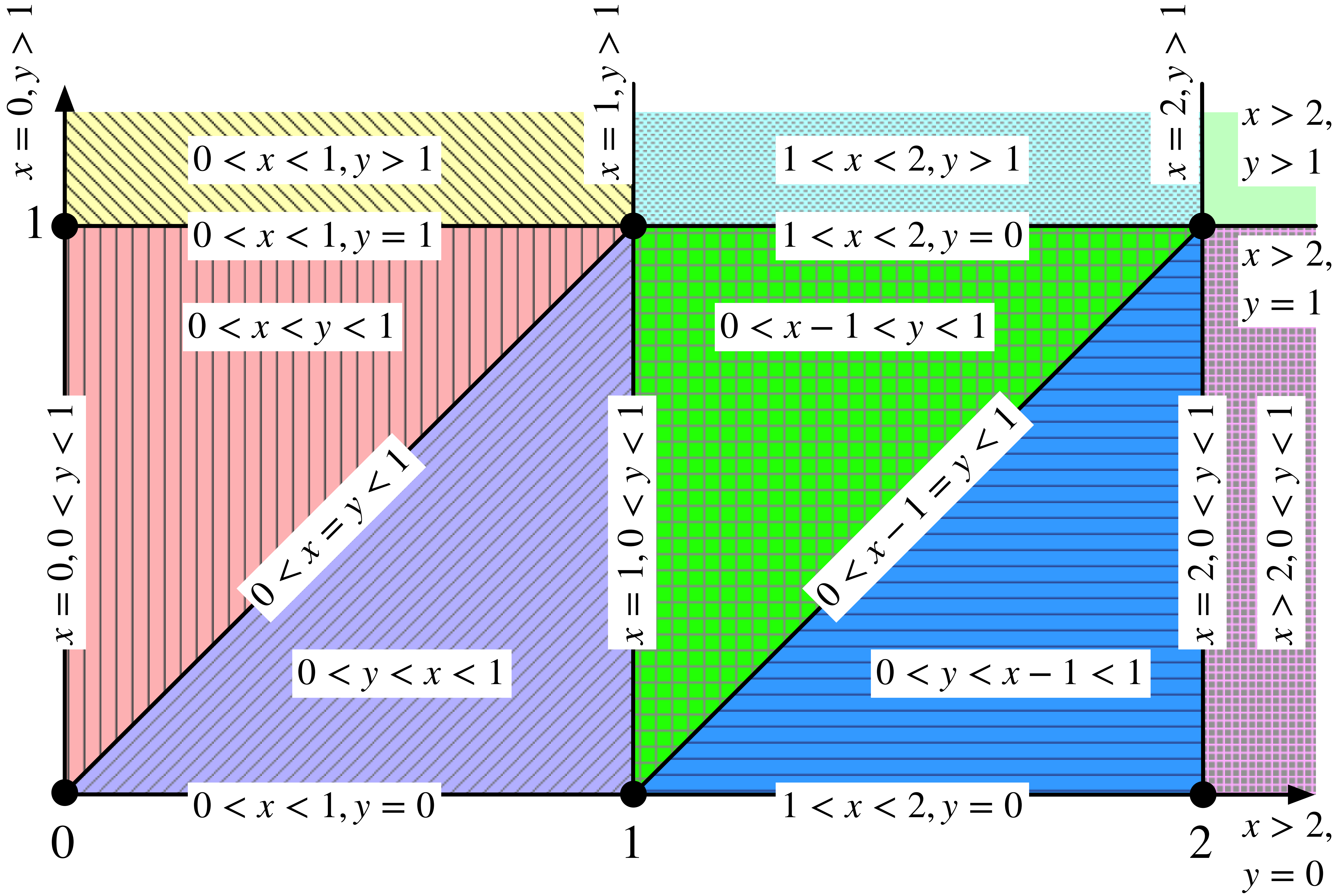
- Let $TP$ be a timed automaton. Two clock valuations $v$ and $v'$ of $TP$ are region-equivalent if:
- for eveay clock variable $x$ and for every integer $0 \leq d \leq k_x$, $v(x) = d$ if and only if $v'(x) = d$ and $v(x) < d$ if and only if $v'(x) < d$, and
- for every pair of clock variable x and y such that $v(x) \leq k_x$ and $v(y) \leq k_y$, $\mathsf{frac}(v(x)) \leq \mathsf{frac}(v(y))$ if and only if $\mathsf{frac}(v'(x)) \leq \mathsf{frac}(v'(y))$.
  - For a real number $r$, $\mathsf{frac}(r)$ denotes the fractional part of $r$, i.e., $\mathsf{frac}(r) = r - \lfloor r \rfloor$.
- Two states $(t, v)$ and $(t', v')$ of $TP$ are region-equivalent if $t = t'$ and $v$ and $v'$ are region equivalent.

# Region Equivalence

## Theorem

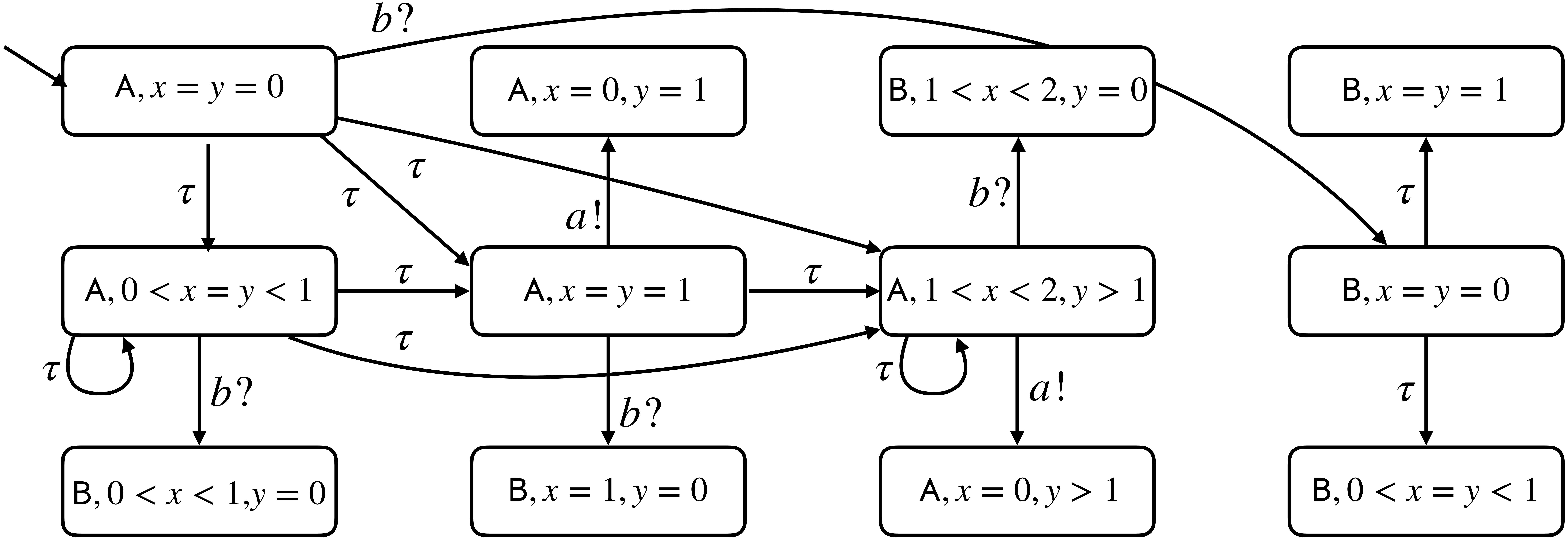
- Let $TP$ be a timed automaton. Consider two states $s$ and $t$ of $TP$ that are region-equivalent. We have the following.
  (1) If $s \stackrel{\alpha}{\rightarrow} s'$ is an input, or output, or internal action of $TP$, then there exists a state $t'$ such that $t \stackrel{\alpha}{\rightarrow} t'$ holds and $s'$ and $t'$ are region-equivalent.
  (2) For every real-valued time duration $\delta > 0$ such that $s \stackrel{\delta}{\rightarrow} s + \delta$ is a timed action of $TP$, there exists a duration $\delta' > 0$ such that $t \stackrel{\delta'}{\rightarrow} t + \delta'$ is a timed action of $TP$ and $s + \delta$ and $t + \delta'$ are region-equivalent.

# Search using Clock Regions




- The infinite clock valuations are partitioned into finitly many regions.
- Now we can adopt the on-the-fly DFS algorithm for reachability.
- Let us start from the initial region, written as $[\mathsf{A}, x = y = 0]$.

# Search using Clock Regions



# Executions on Regions

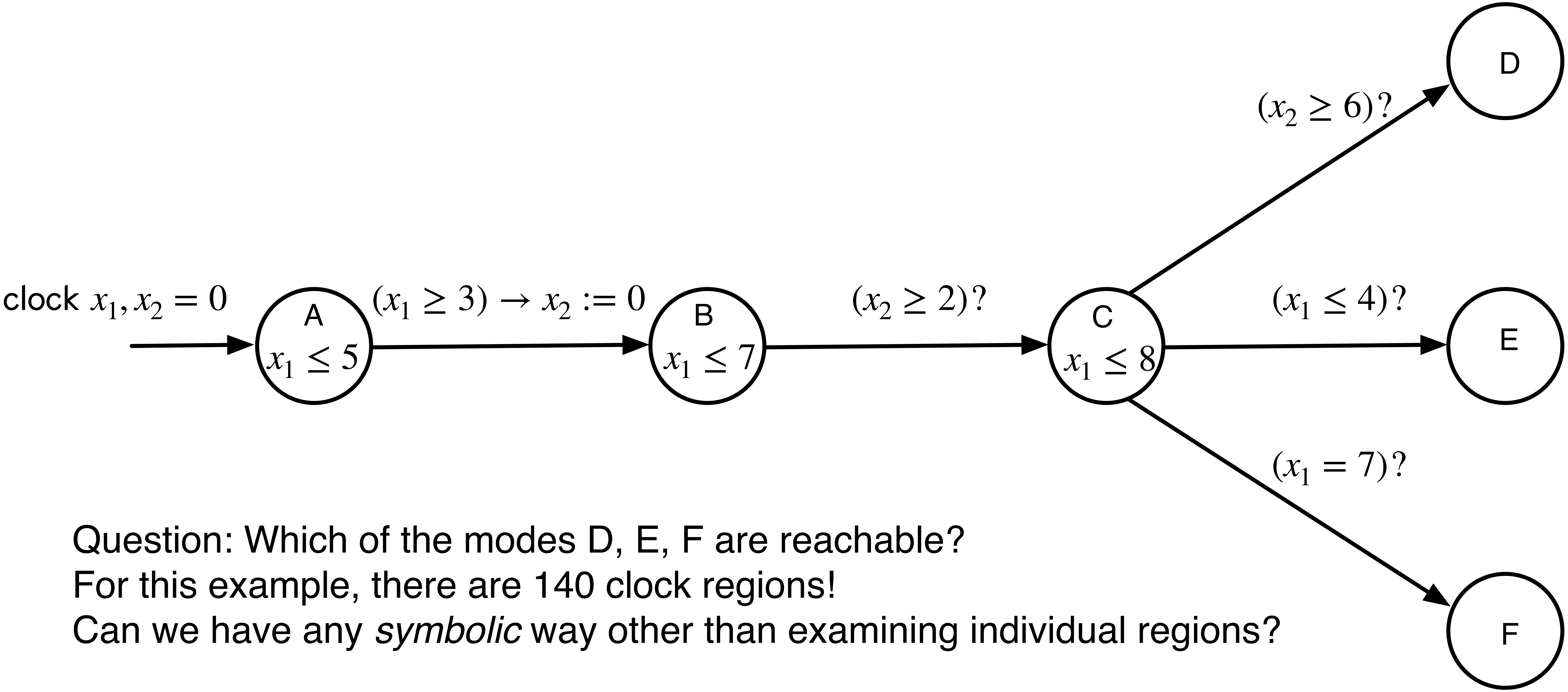
- An execution

$$(\mathsf{A}, 0, 0) \xrightarrow{0.6} (\mathsf{A}, 0.6, 0.6) \xrightarrow{b?} (\mathsf{B}, 0.6, 0) \xrightarrow{0.5} (\mathsf{B}, 1.1, 0.5) \xrightarrow{c!} (\mathsf{A}, 1.1, 0.5)$$
$$\xrightarrow{0.2} (\mathsf{A}, 1.3, 0.7) \xrightarrow{a!} (\mathsf{A}, 0, 0.7) \xrightarrow{1.25} (\mathsf{A}, 1.25, 1.95) \xrightarrow{0.61} (\mathsf{A}, 1.86, 2.56)$$

  and other *similar* executions can be expressed as:

$$[\mathsf{A}, x = y = 0] \xrightarrow{\tau} [\mathsf{A}, 0 < x = y < 1] \xrightarrow{b?} [\mathsf{B}, 0 < x < 1, y = 0]$$
$$\xrightarrow{\tau} [\mathsf{B}, 0 < x - 1 < y < 1] \xrightarrow{c!} [\mathsf{A}, 0 < x - 1 < y < 1]$$
$$\xrightarrow{\tau} [\mathsf{A}, 0 < x - 1 < y < 1] \xrightarrow{a!} [\mathsf{A}, x = 0, 0 < y < 1]$$
$$\xrightarrow{\tau} [\mathsf{A}, 1 < x < 2, y > 1] \xrightarrow{\tau} [\mathsf{A}, 1 < x < 2, y > 1]$$

# Timing Analysis Example




Question: Which of the modes D, E, F are reachable?
For this example, there are 140 clock regions!
Can we have any *symbolic* way other than examining individual regions?

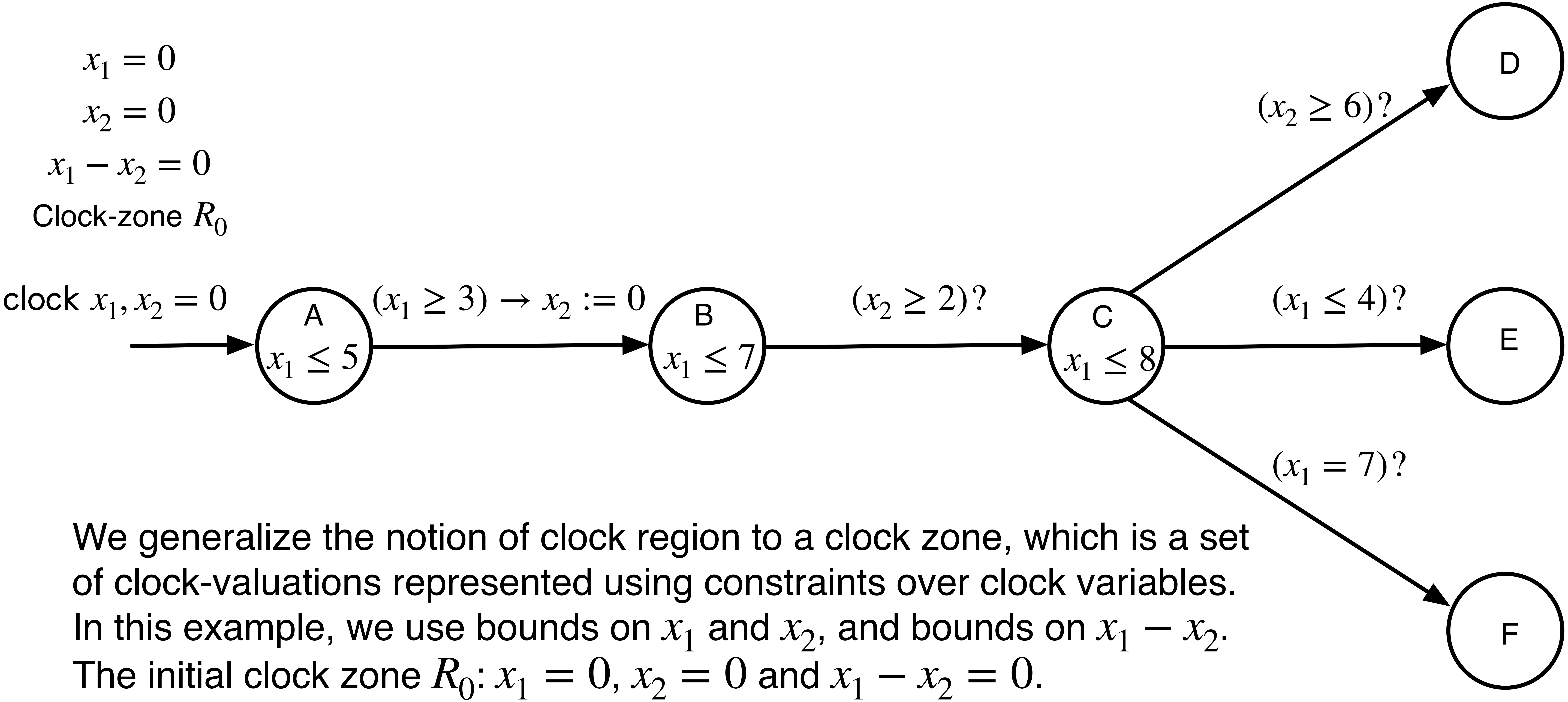
# Timing Analysis Example

$x_1 = 0$

$x_2 = 0$

$x_1 - x_2 = 0$

Clock-zone $R_0$




We generalize the notion of clock region to a clock zone, which is a set of clock-valuations represented using constraints over clock variables. In this example, we use bounds on $x_1$ and $x_2$, and bounds on $x_1 - x_2$. The initial clock zone $R_0$: $x_1 = 0$, $x_2 = 0$ and $x_1 - x_2 = 0$.

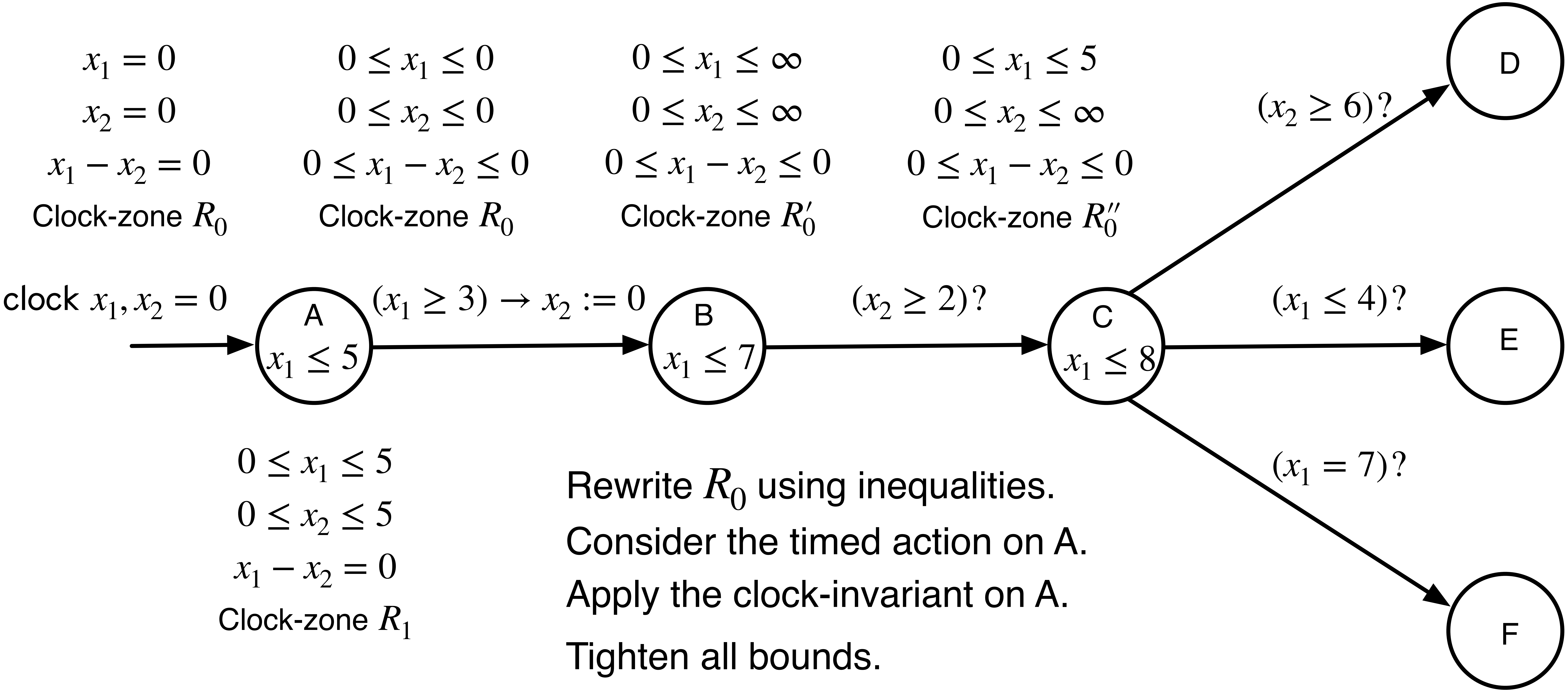
# Timing Analysis Example

$x_1 = 0$
$x_2 = 0$
$x_1 - x_2 = 0$
Clock-zone $R_0$

$0 \leq x_1 \leq 0$
$0 \leq x_2 \leq 0$
$0 \leq x_1 - x_2 \leq 0$
Clock-zone $R_0$

$0 \leq x_1 \leq \infty$
$0 \leq x_2 \leq \infty$
$0 \leq x_1 - x_2 \leq 0$
Clock-zone $R_0'$

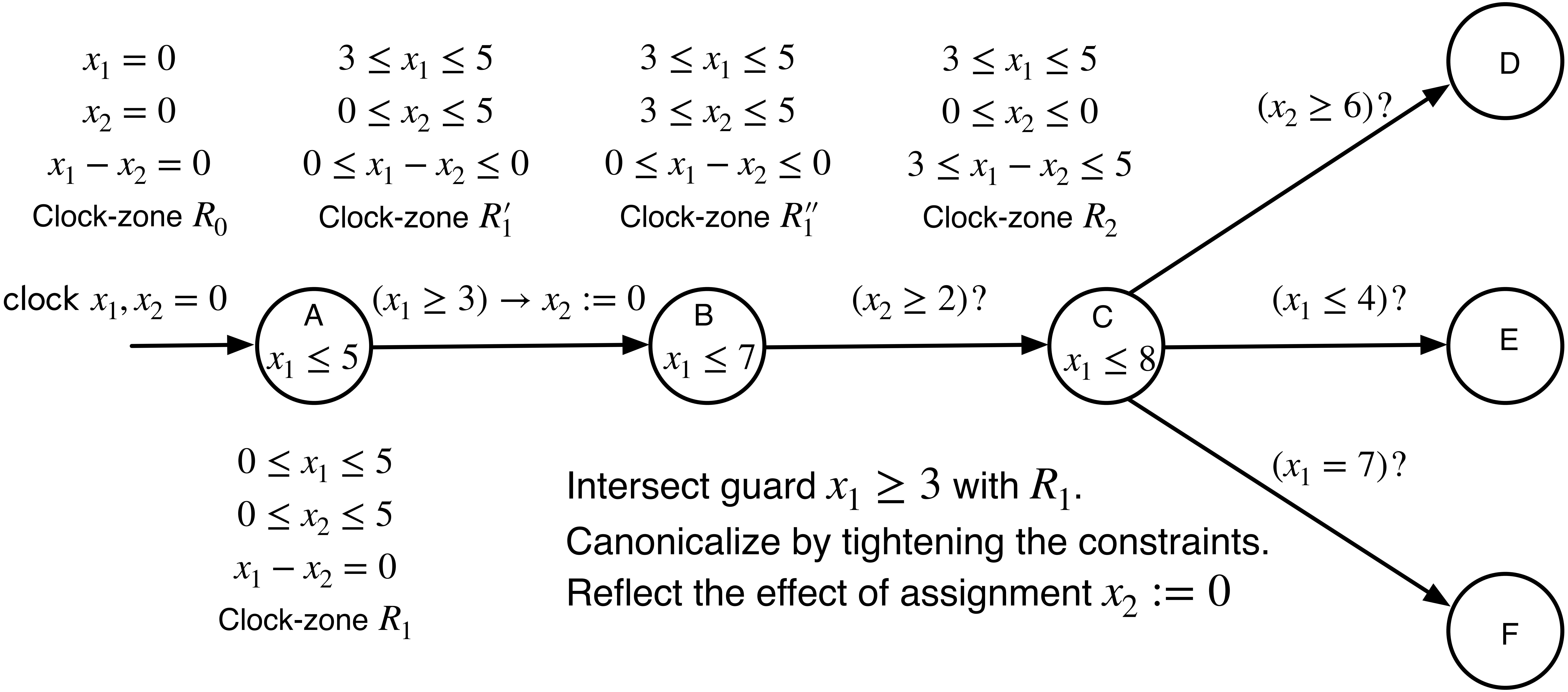
$0 \leq x_1 \leq 5$
$0 \leq x_2 \leq \infty$
$0 \leq x_1 - x_2 \leq 0$
Clock-zone $R_0''$




$0 \leq x_1 \leq 5$
$0 \leq x_2 \leq 5$
$x_1 - x_2 = 0$
Clock-zone $R_1$

Rewrite $R_0$ using inequalities.
Consider the timed action on A.
Apply the clock-invariant on A.
Tighten all bounds.

# Timing Analysis Example

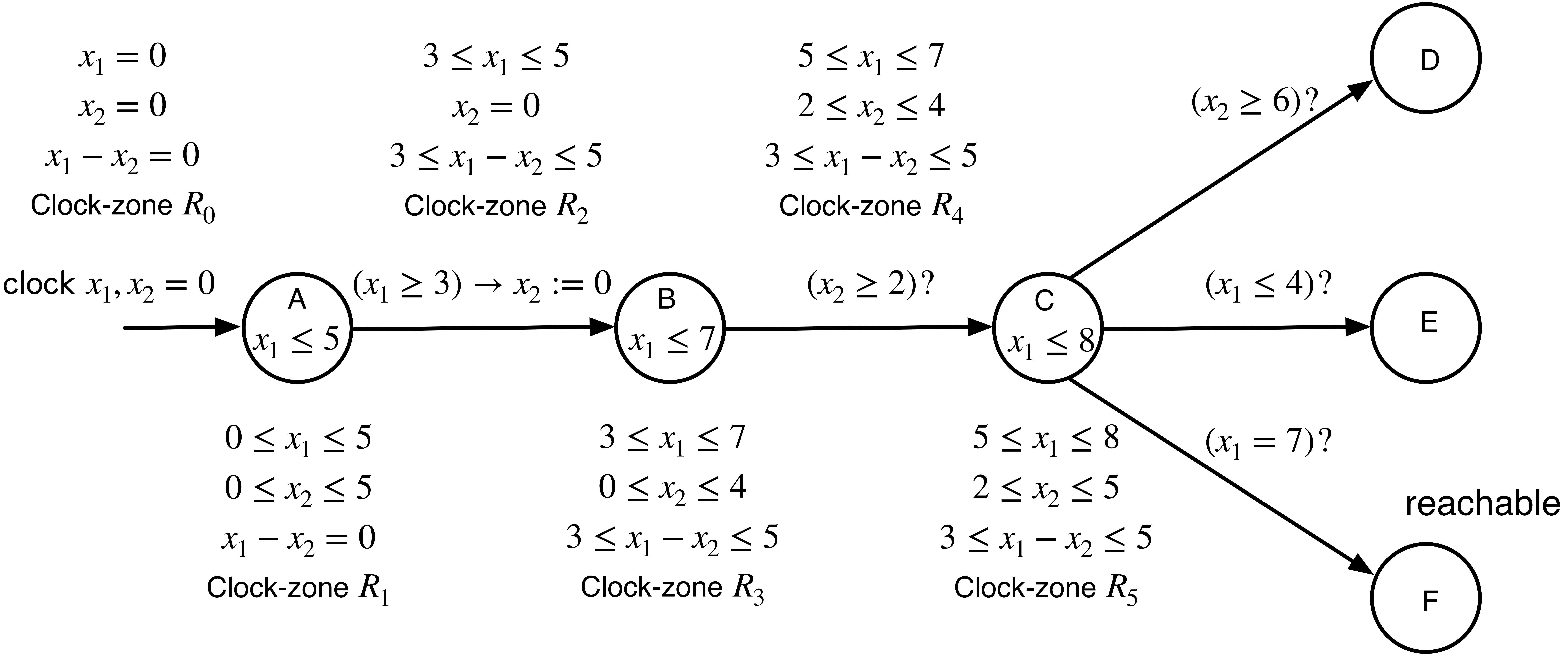



Intersect guard $x_1 \geq 3$ with $R_1$.

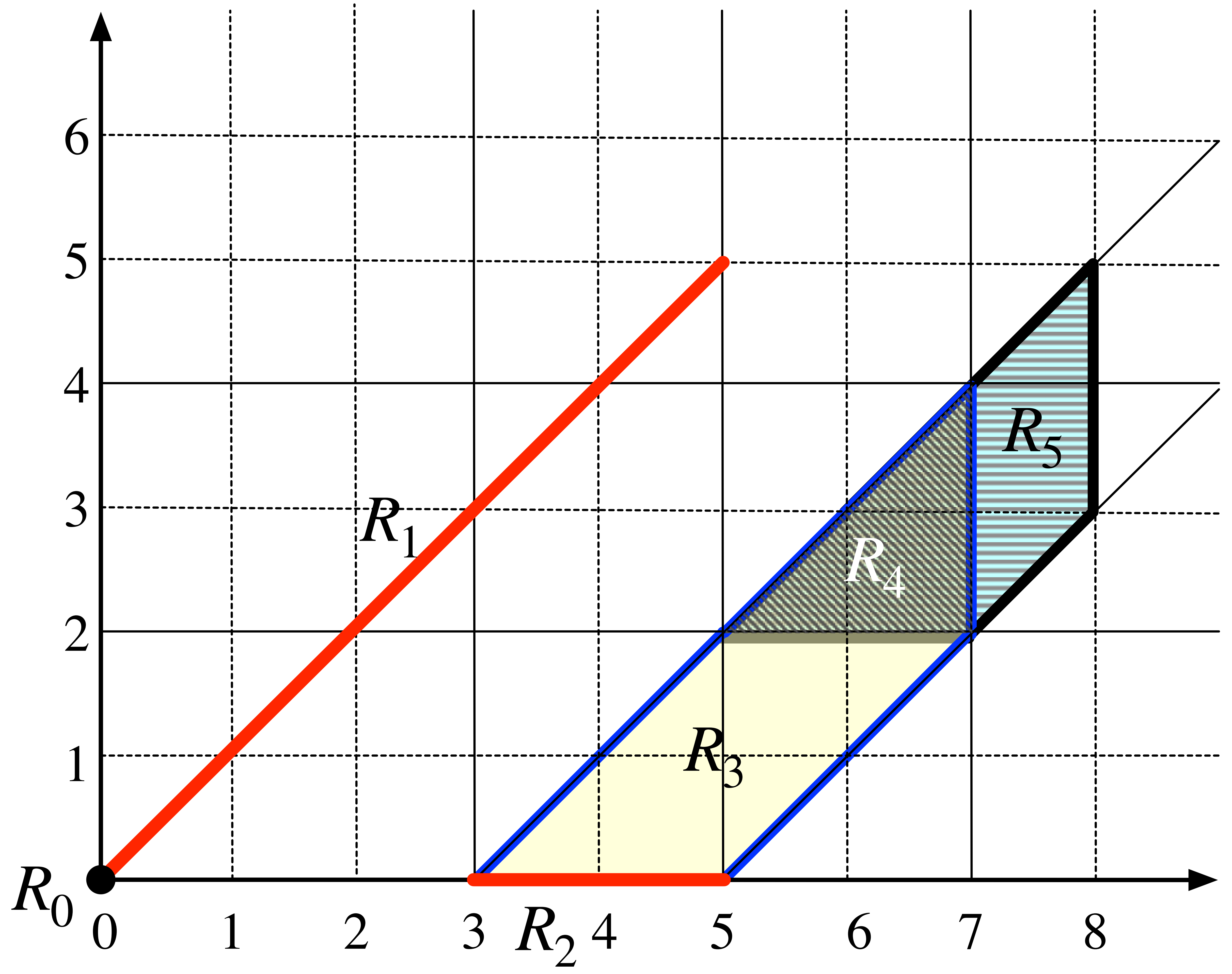
Canonicalize by tightening the constraints.

Reflect the effect of assignment $x_2 := 0$

# Timing Analysis Example



# Clock Zones



# Difference Bounds Matrix

- Clock variables: $x_1, x_2, \ldots, x_m$, with dummy $x_0 = 0$
- *Difference Bounds Matrix* (DBM)
  - A $(m+1)$-square matrix $R$ with the $(i, j)$-th entry gives the upper bound on $x_i - x_j$.
  - $R$ represents a clock zone with constraint $\bigwedge_{0 \leq i,j \leq m} (x_i - x_j) \leq R_{ij}$
    - Note: $R_{ij}$ may be $\infty$
  - Since $x_0 = 0$, the column 0 (entries $R_{i0}$) gives the upper bounds on $x_i$ and row 0 (entries $R_{0j}$) gives the upper bounds on $-x_j$ (thus the lower bounds of $x_j$).

Examples

$$R_1 = \begin{pmatrix} 0 & 0 & 0 \\ 5 & 0 & 0 \\ 5 & 0 & 0 \end{pmatrix}$$

$$R_5 = \begin{pmatrix} 0 & -5 & 2 \\ 8 & 0 & 5 \\ 5 & -3 & 0 \end{pmatrix}$$

# Canonicalization

- A DBM $R$ is said to be canonical if and only if

$$\text{for all } 0 \leq i, j, l \leq m, R_{ij} \leq R_{il} + R_{lj}.$$

- Algorithm to canonicalize a DBM $R$:
  - Ex. $(1 \leq x_1 \leq 3) \wedge (x_2 \geq 0) \wedge (0 \leq x_3 \leq 3)$
    $\wedge (x_2 - x_3 = 1) \wedge (x_2 - x_1 \geq 2)$

$$\begin{pmatrix} 0 & -1 & 0 & 0 \\ 3 & 0 & -2 & \infty \\ \infty & \infty & 0 & 1 \\ 3 & \infty & -1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & -1 & -3 & -2 \\ 2 & 0 & -2 & -1 \\ 4 & 3 & 0 & 1 \\ 3 & 2 & -1 & 0 \end{pmatrix}$$

Input: $(m+1)$-dim DBM $R$
Output: Canonicalized $R$ or Empty if $R$ is empty
for $l = 0$ to $m$ {
  for $i = 0$ to $m$ {
    for $j = 0$ to $m$ {
      $R[i, j] := \min(R[i, j], R[i, l] + R[l, j]);$
    };
    if $R[i, i] < 0$ then return Empty
  }
};
return R

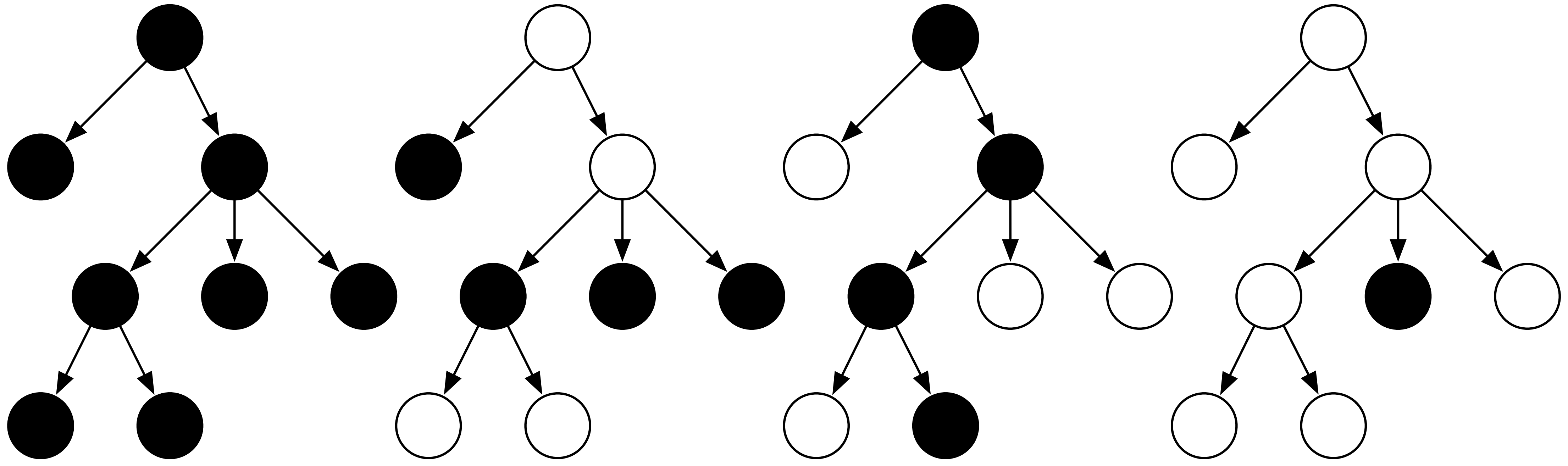
# Operations on DBMs

- Atomic constraints: an atomic constraint $x_i \leq k$ can be represented by $R$ with $R_{ii} = 0$ for all $0 \leq i \leq m$, $R_{i0} = k$, $R_{0j} = 0$ for all $0 \leq j \leq m$, and ∞ for all other entries.
- Intersections: The intersection of the clock zones represented by $R$ and $R'$ can be represented by a DBM whose $(i, j)$-th entry is $\min(R_{i,j}, R'_{i,j})$.
- Time elapse: To compute the set of clock valuations that can be reached from the clock zone represented by $R$ using timed actions, simply set ∞ to $R_{i0}$ for all $1 \leq i \leq m$.
- Subset test: The clock zones represented by a DBM $R$ is a subset of the clock zone represented by $R'$ if and only if $R_{i,j} \leq R'_{i,j}$ for every $0 \leq i, j \leq m$.

# Model Checking Timed Automata

- Properties to be verified are written as formulae of TCTL (Timed Computational Tree Logic)
  - $A\Box P$ : $P$ always holds in all possible paths
  - $A\Diamond P$ : $P$ eventually holds in all possible paths
  - $E\Box P$ : $P$ always holds at least in a path
  - $E\Diamond P$ : $P$ eventually holds at least in a path

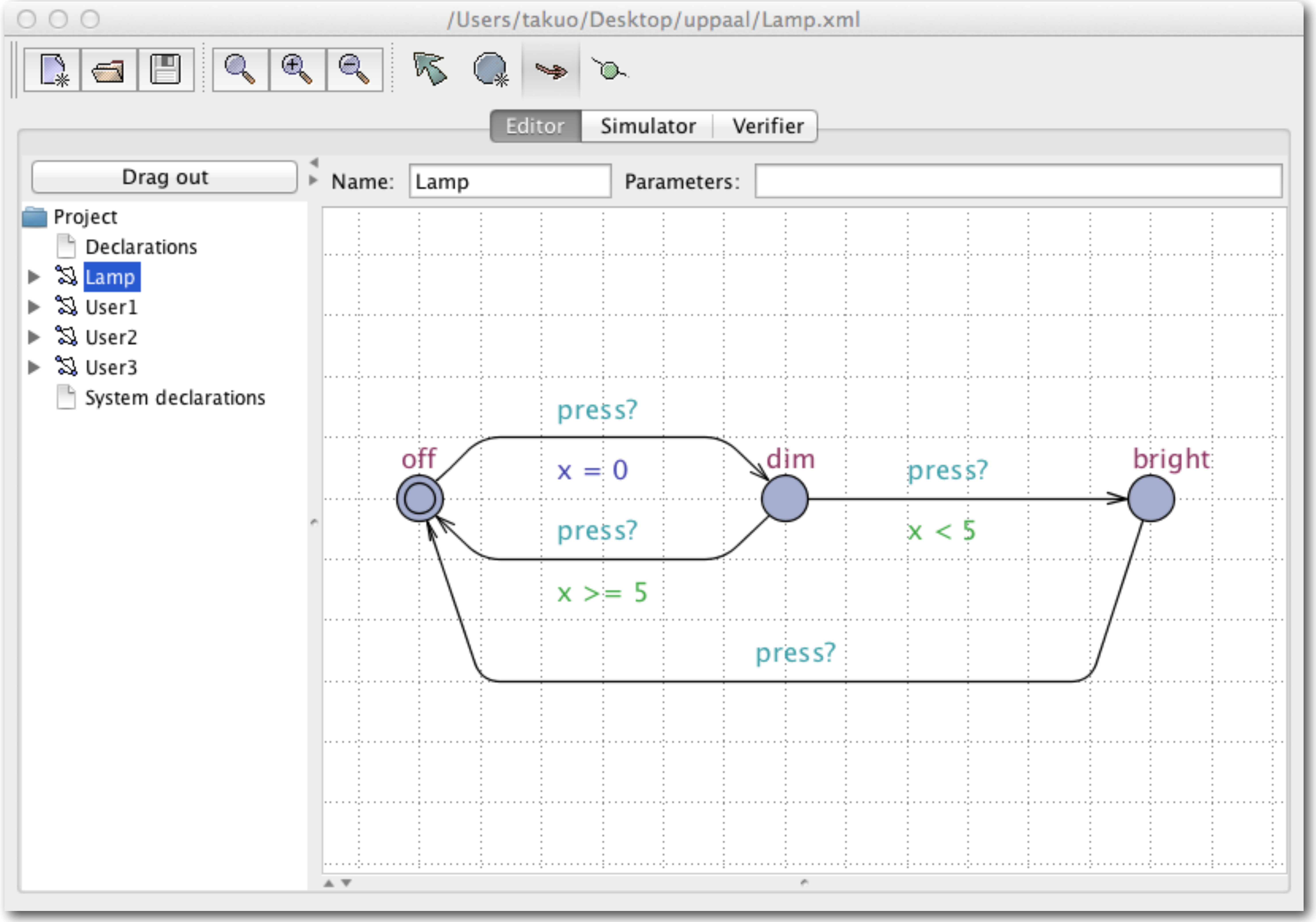
$A\Box P$ $\quad$ $A\Diamond P$ $\quad$ $E\Box P$ $\quad$ $E\Diamond P$



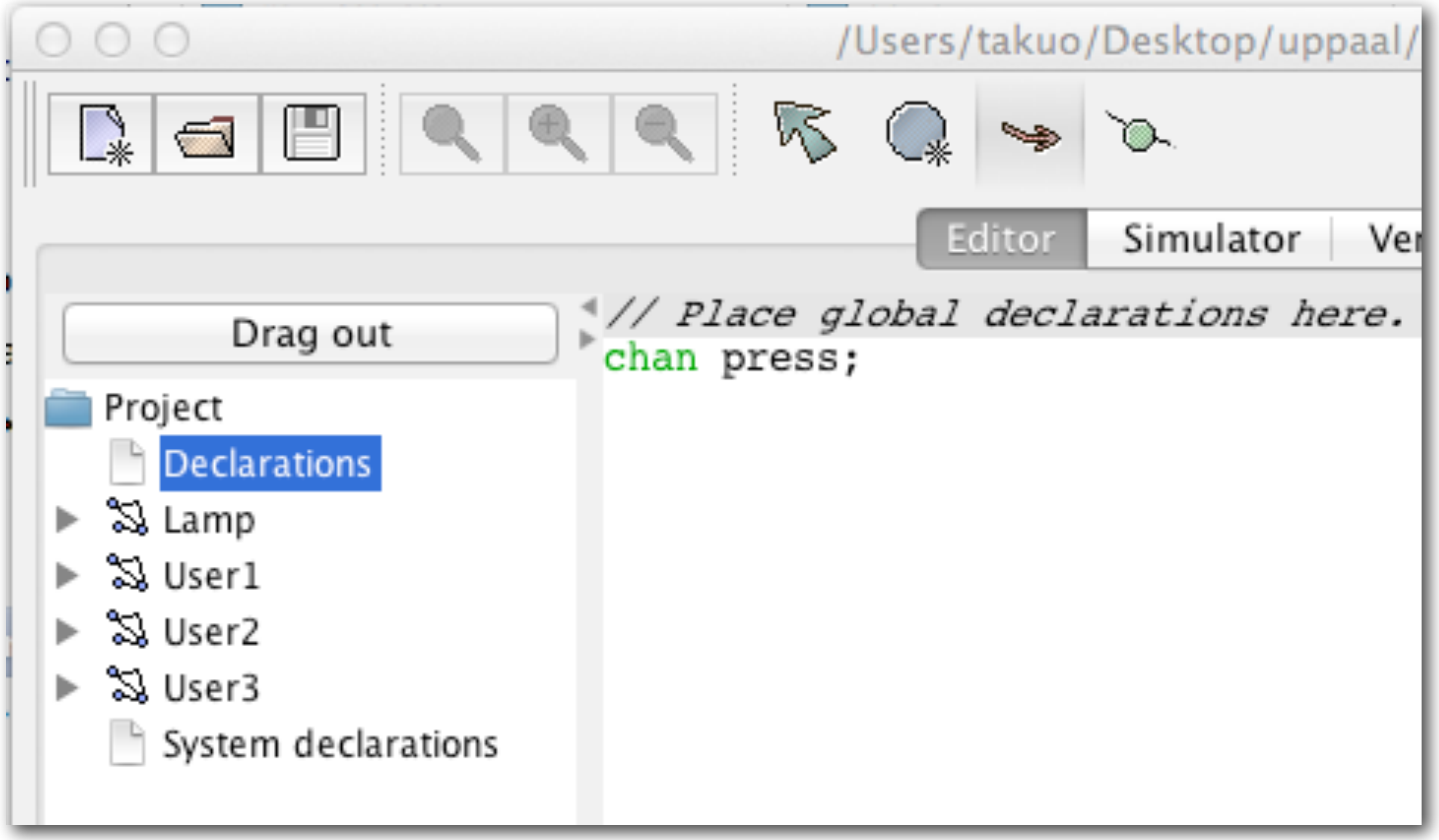
# UPPAAL

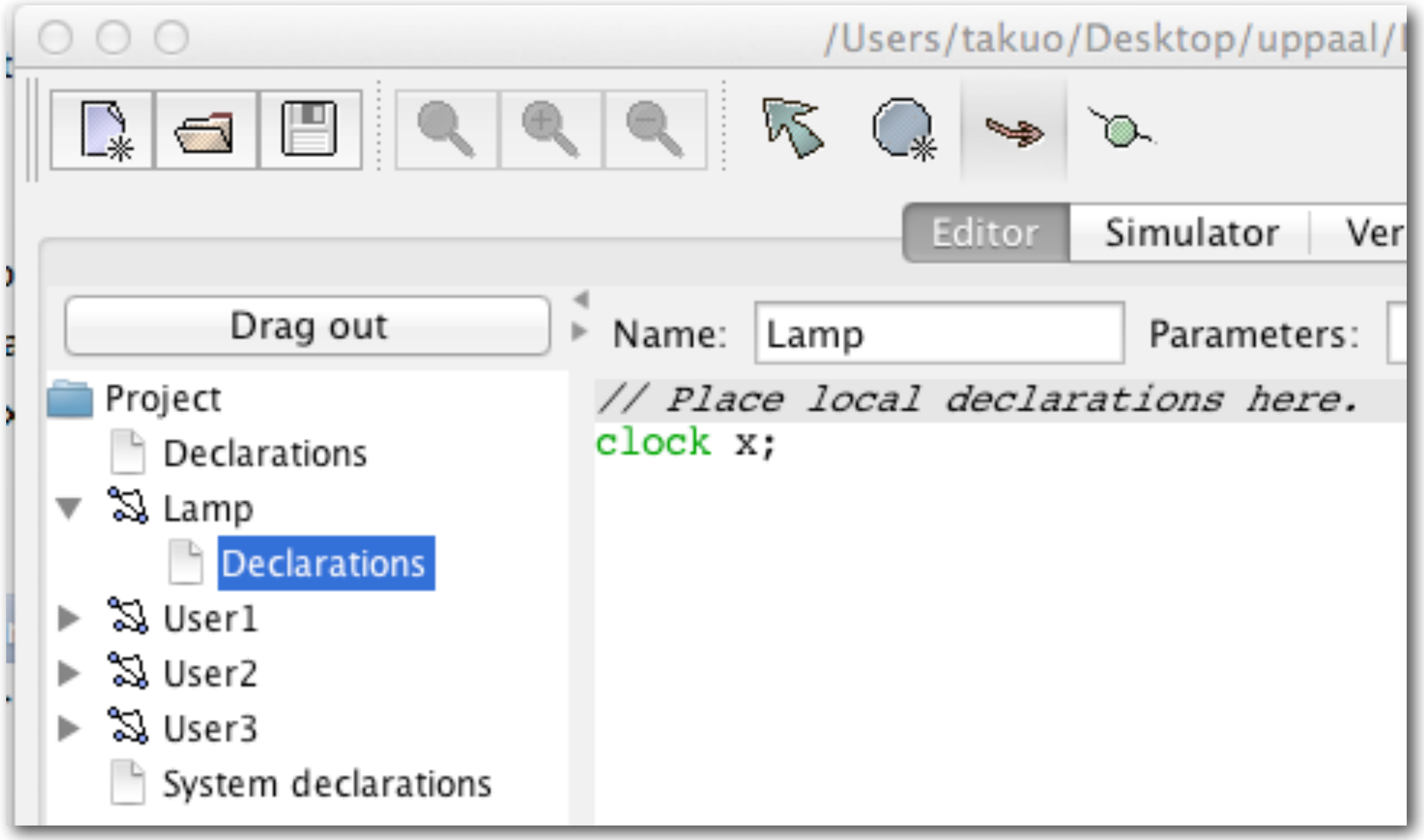
- An integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata.
- http://uppaal.org
- Developed at Uppsala University & Aalborg University

# UPPAAL: Template Editor

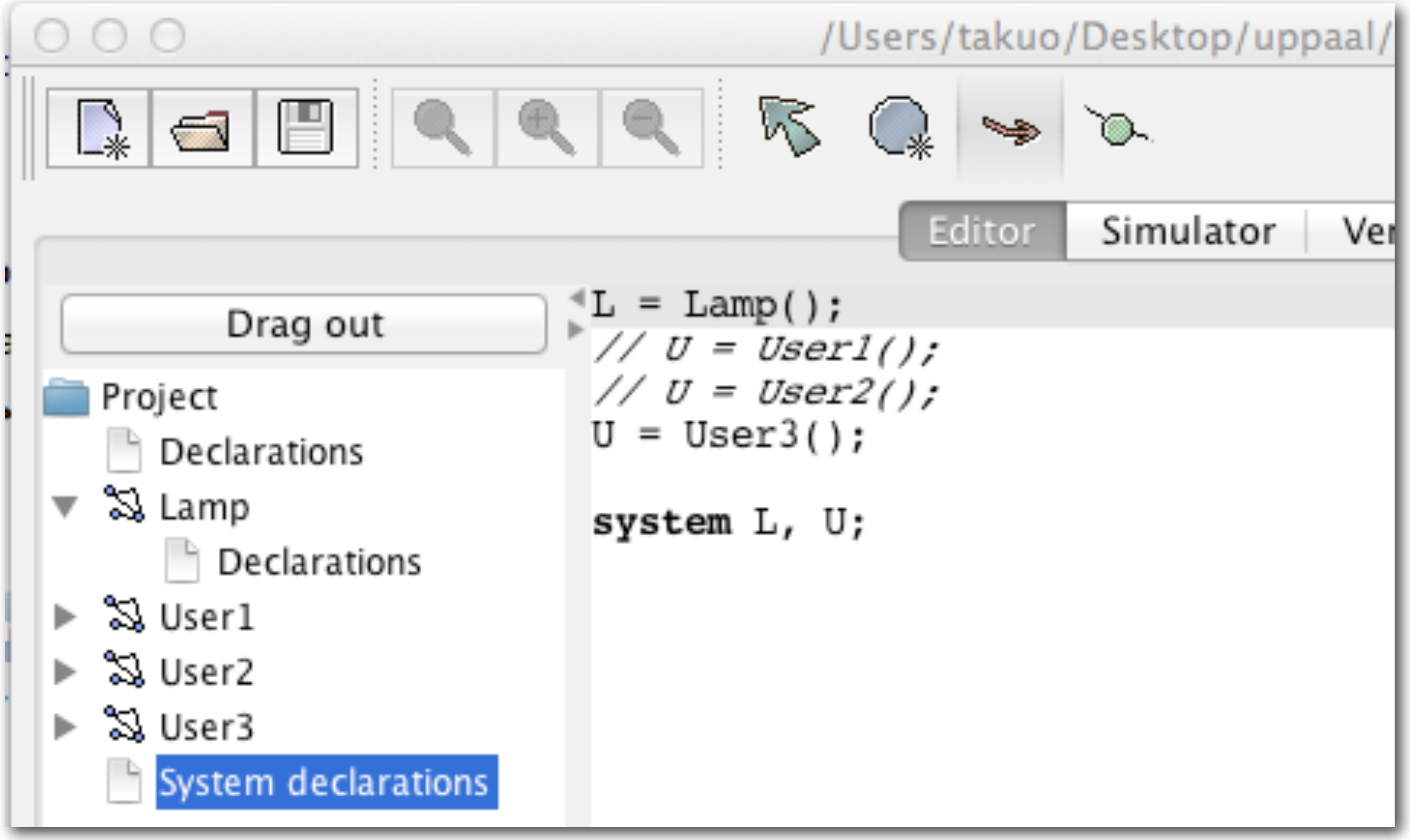


# Global & Local Declarations

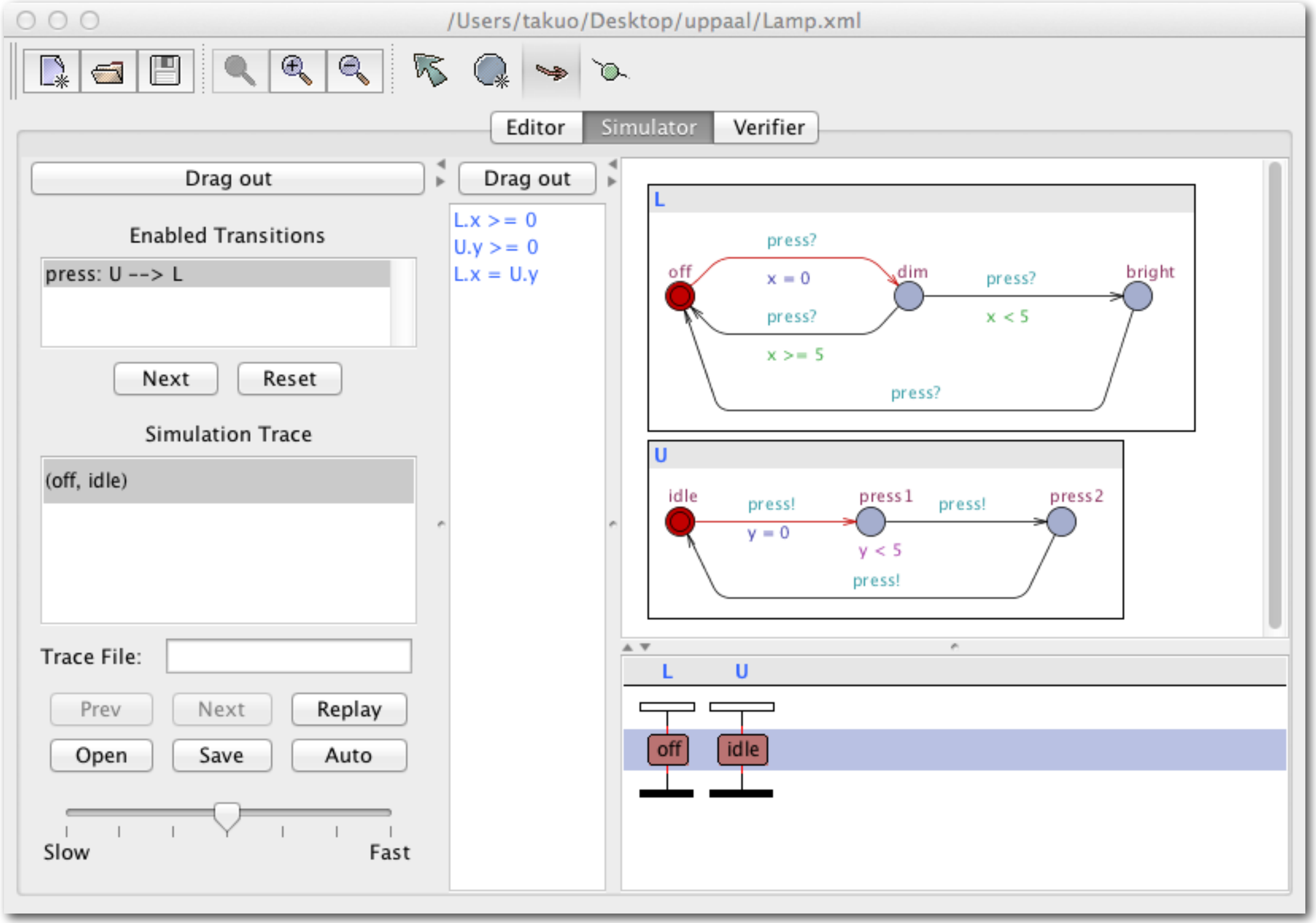


# System Declarations

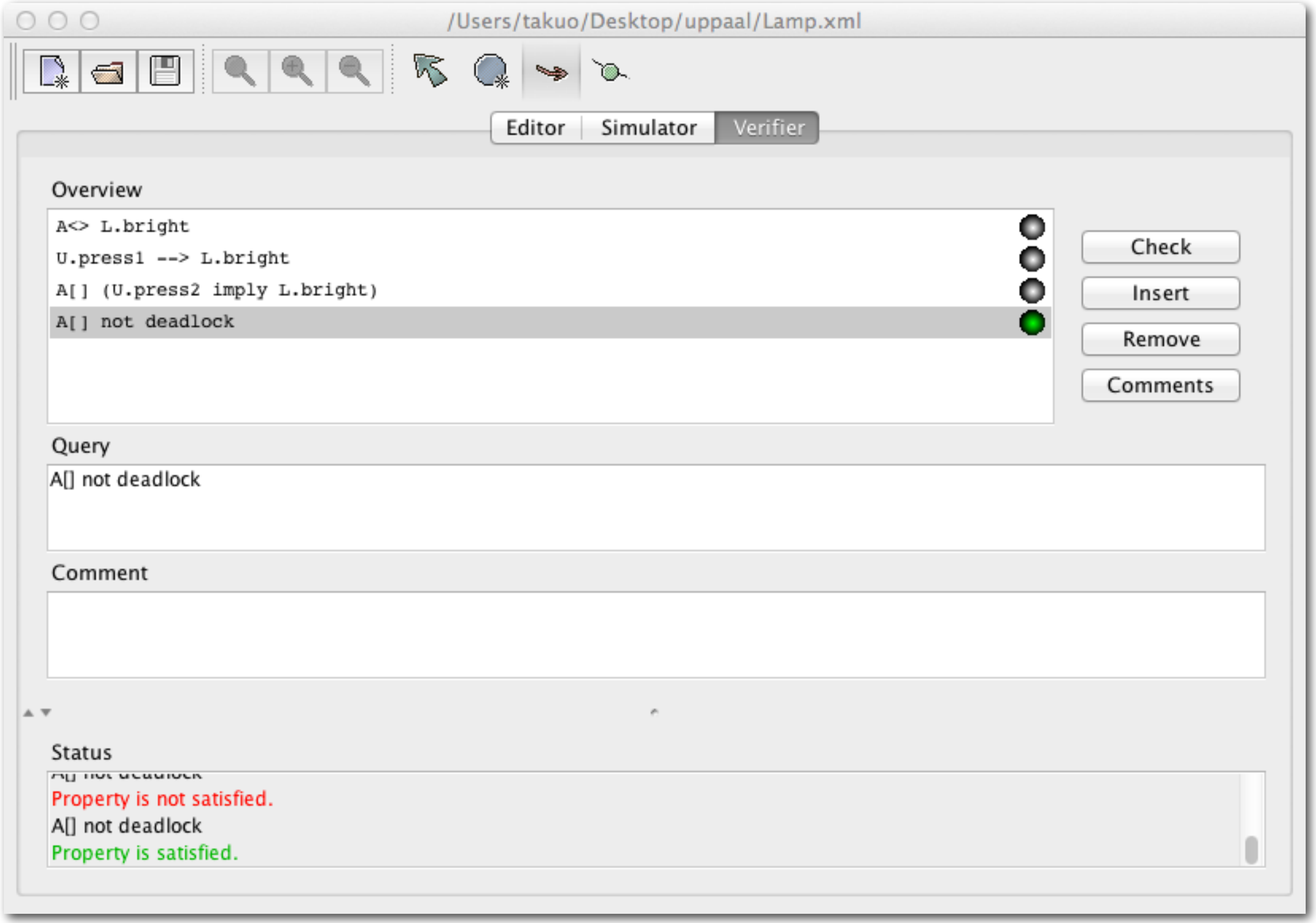


# Simulator



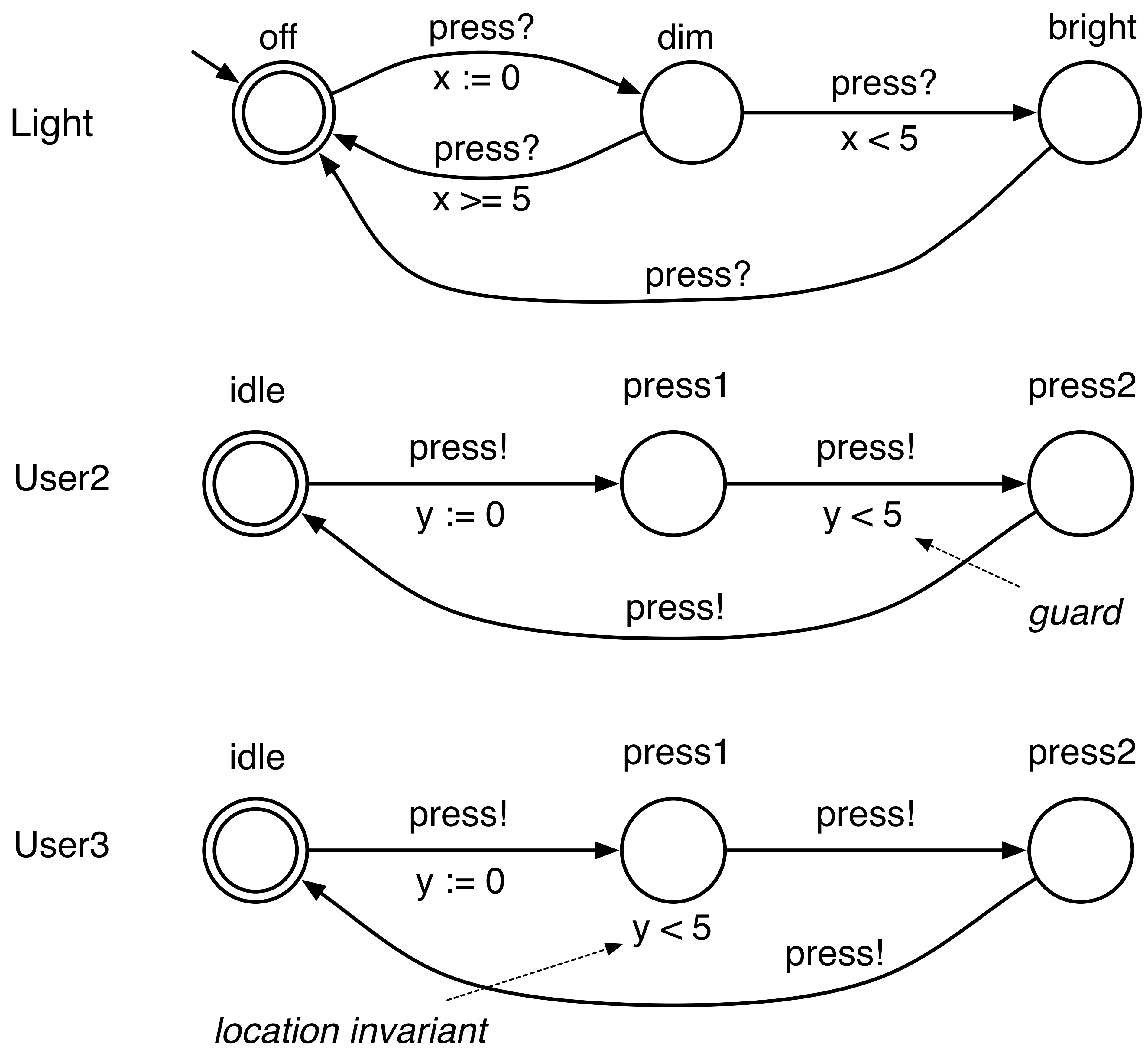
/Users/takuo/Desktop/uppaal/Lamp.xml
Editor
Simulator
Verifier
Drag out
Enabled Transitions
press: U --> L
Next
Reset
Simulation Trace
(off, idle)
Trace File:
Prev
Next
Replay
Open
Save
Auto
Slow
Fast
Drag out
L.x >= 0
U.y >= 0
L.x = U.y
L
off
press?
x = 0
dim
press?
bright
press?
x < 5
press?
x >= 5
press?
U
idle
press!
y = 0
press1
y < 5
press!
press2
press!
L
U
off
idle

# Verifier



/Users/takuo/Desktop/uppaal/Lamp.xml
Editor
Simulator
Verifier
Overview
A<> L.bright
U.press1 --> L.bright
A[] (U.press2 imply L.bright)
A[] not deadlock
Check
Insert
Remove
Comments
Query
A[] not deadlock
Comment
Status
Property is not satisfied.
A[] not deadlock
Property is satisfied.

# Ex. Light & Users




| TCTL formula | Lamp ll User1 | Lamp ll User2 | Lamp ll User3 |
|---|---|---|---|
| `A<> L.bright` | × | × | × |
| `U.press1 --> L.bright` | — | × | ○ |
| `A[] (U.press2 imply L.bright)` | — | ○ | ○ |
| `A[] not deadlock` | ○ | × | ○ |

# Summary

- Timed Model (2)
    - Timed Automata
    - Region, Region Equivalence
    - Clock Zone
    - Difference Bounds Matrix (DBM)
    - Model Checking using UPPAAL