CIS 721 - Real-Time Systems

Lecture 22: UPPAAL Logic

Mitch Neilsen
neilsen@ksu.edu

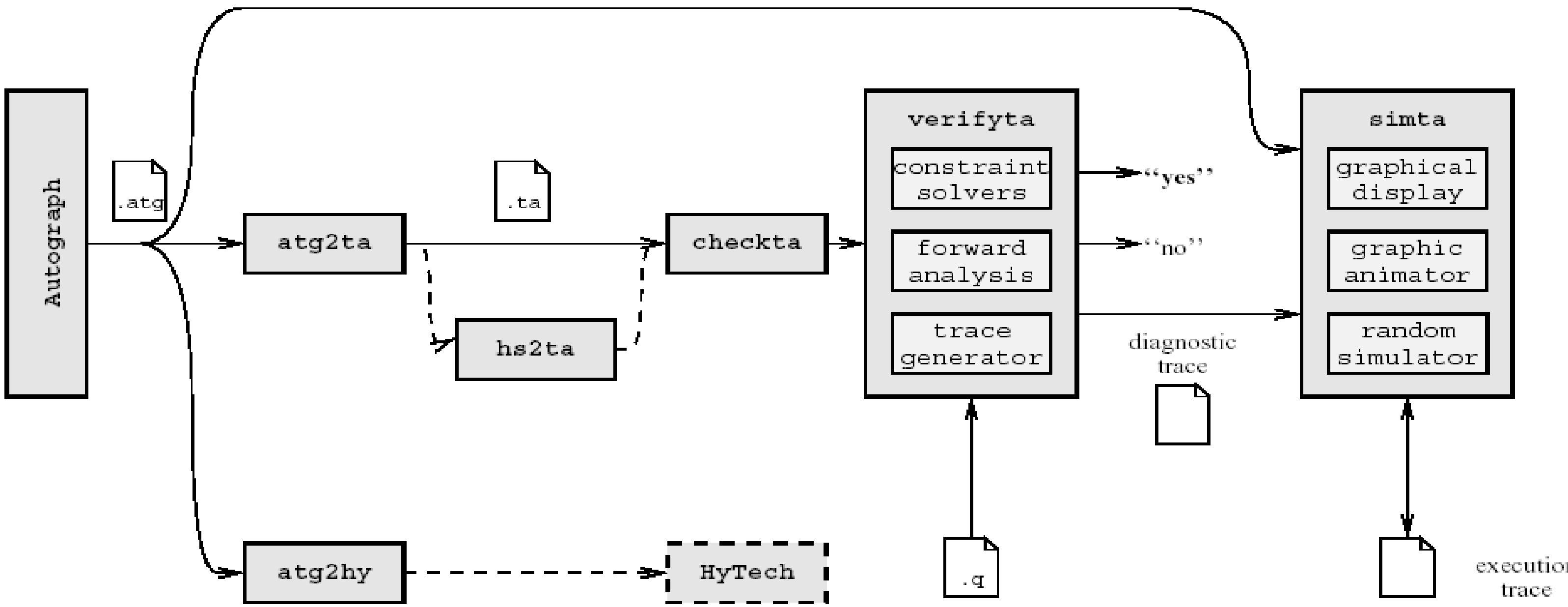
Outline

- Real-Time Verification and Validation Tools
  - Promela and SPIN
    - Simulation
    - Verification
  - **Real-Time Extensions:**
    - RT-SPIN – Real-Time extensions to SPIN
    - **UPPAAL – Toolbox for validation and verification of real-time systems**

UPPAAL Components

- **UPPAAL** consists of three main parts:
  - a **description language**,
  - a **simulator**, and
  - a **model checker**.
- The **description language** is a non-deterministic guarded command language with data types. It can be used to describe a system as a *network of timed automata* using either a graphical (*.atg, *.xml) or textual (*.xta) format.
- The **simulator** enables examination of *possible* dynamic executions of a system during the early modeling stages.
- The **model checker** exhaustively checks *all* possible states.

UPPAAL Tools (earlier version)



- **checkta** – syntax checker
- **simta** – simulator
- **verifyta** – model checker

UPPAAL Specification Language

A[] p

E<> p

A = on all paths, [ ] = always

E = on some path, <> = eventually

(AG p) – all paths, always (globally)

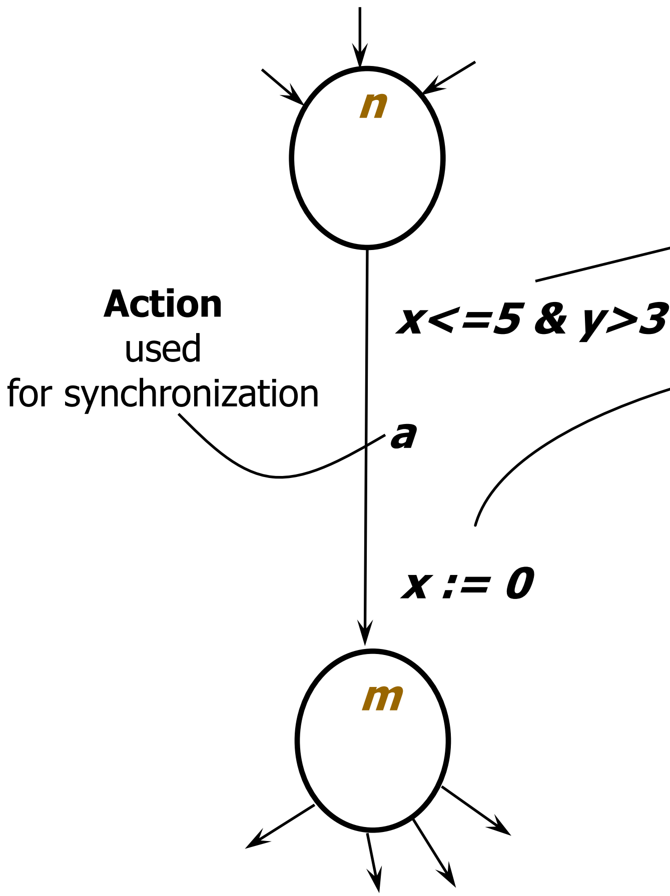
(EF p) – some path, eventually (finally)

process location   data guards   clock guards

p::= a.l | g_d | g_c | p and p |
p or p | not p | p imply p |
(p)

Timed Automata

(Alur & Dill, 1990)



Clocks: *x, y*

Guard
Boolean combination of comp with integer bounds

Reset
Action perfumed on clocks

State

(***location*** , ***x***=v , ***y***=u) where v,u are in $\mathbf{R}$

Transitions

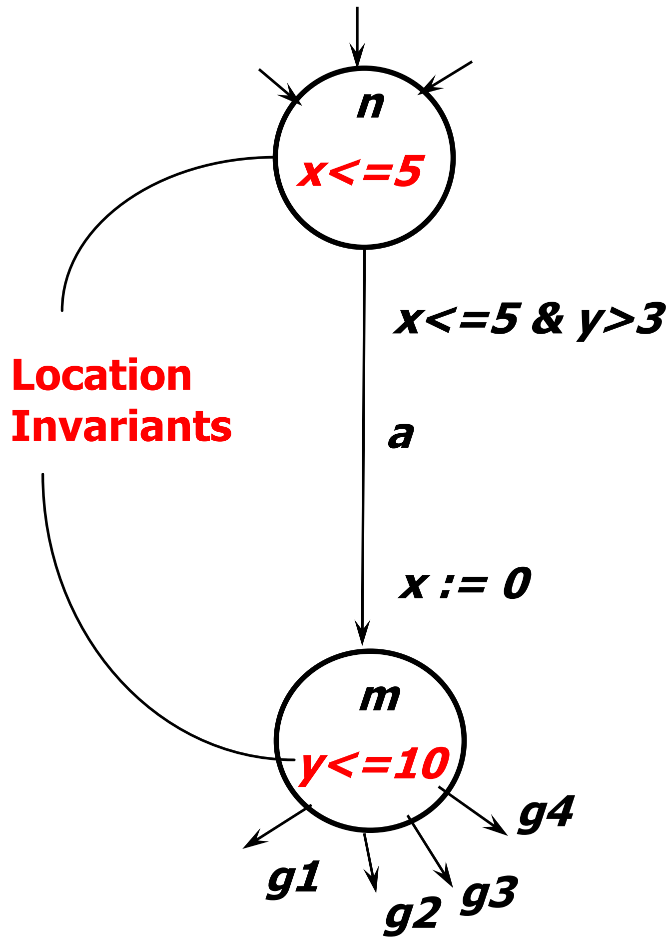
$(\, n\,,\ x{=}2.4\,,\ y{=}3.1415\,) \xrightarrow{a} (\, m\,,\ x{=}0\,,\ y{=}3.1415\,)$

$(\, n\,,\ x{=}2.4\,,\ y{=}3.1415\,) \xrightarrow{e(1.1)} (\, n\,,\ x{=}3.5\,,\ y{=}4.2415\,)$

Timed Safety Automata =

(Henzinger et al, 1992)

Timed Automata + Invariants



Clocks: ***x, y***

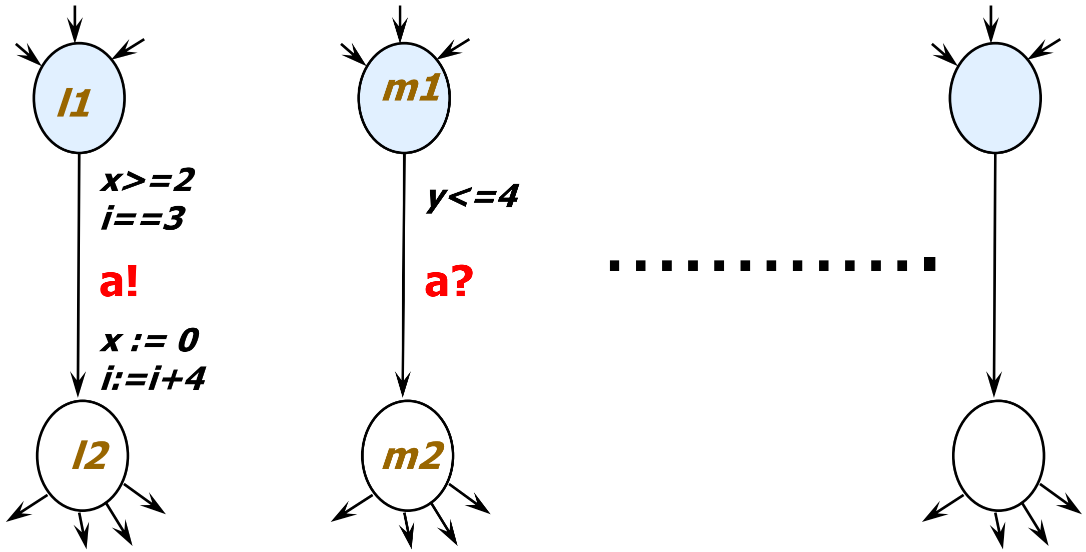
Transitions

(***n*** , ***x***=2.4 , ***y***=3.1415) $\cancel{\xrightarrow{e(3.2)}}$

(***n*** , ***x***=2.4 , ***y***=3.1415) $\xrightarrow{e(1.1)}$ (***n*** , ***x***=3.5 , ***y***=4.2415)

Networks of Timed Automata

+ Integer Variables + Arrays (in UPPAAL)



Declarations in UPPAAL

```
clock x1, …, xn;
int i1, …, im;
chan a1, …, ao;
const c1 n1, …, cp np;
```

Examples:

```
clock x, y;
int i, J0; int[0,1] k[5];
const delay 5, true 1, false 0;
```

A simple program

```
Int x

Process P
        do
        :: x<2000 → x:=x+1
        od

Process Q
        do
        :: x>0 → x:=x-1
        od

Process R
        do
        :: x=2000 → x:=0
        od

fork P; fork Q; fork R
```

What are possible values for x?

Questions/Properties:

E<>(x>1000)
E<>(x>2000)
A[](x<=2000)
E<>(x<0)
A[](x>=0)

Possible

Always

Appendix B: BNF for q-format

$$
\begin{array}{lcl}
\mathit{Prop} & \rightarrow & \texttt{E<>}\ \mathit{StateProp} \mid \texttt{A[]}\ \mathit{StateProp} \\
\mathit{StateProp} & \rightarrow & \mathit{AtomicProp} \mid \texttt{(}\ \mathit{StateProp}\ \texttt{)} \\
 & & \mid \texttt{not}\ \mathit{StateProp} \\
 & & \mid \mathit{StateProp}\ \texttt{or}\ \mathit{StateProp} \\
 & & \mid \mathit{StateProp}\ \texttt{and}\ \mathit{StateProp} \\
 & & \mid \mathit{StateProp}\ \texttt{imply}\ \mathit{StateProp} \\
\mathit{AtomicProp} & \rightarrow & \mathit{Id}\texttt{.}\mathit{Id} \mid \mathit{Id}\ \mathit{RelOp}\ \mathit{Nat} \\
 & & \mid \mathit{Id}\ \mathit{RelOp}\ \mathit{Id}\ \mathit{Op}\ \mathit{Nat} \\
\mathit{RelOp} & \rightarrow & \texttt{<} \mid \texttt{<=} \mid \texttt{>=} \mid \texttt{>} \mid \texttt{==} \\
\mathit{Op} & \rightarrow & \texttt{+} \mid \texttt{-} \\
\mathit{Id} & \rightarrow & \mathit{Alpha} \mid \mathit{Id}\ \mathit{AlphaNum} \\
\mathit{Nat} & \rightarrow & \mathit{Num} \mid \mathit{Num}\,\mathit{Nat} \\
\mathit{Alpha} & \rightarrow & \texttt{A} \mid \texttt{...} \mid \texttt{Z} \mid \texttt{a} \mid \texttt{...} \mid \texttt{z} \\
\mathit{Num} & \rightarrow & \texttt{0} \mid \texttt{...} \mid \texttt{9} \\
\mathit{AlphaNum} & \rightarrow & \mathit{Alpha} \mid \mathit{Num} \mid \texttt{\_}
\end{array}
$$

Verification (example.xta)

```
int x:=0;
process P{
state S0;
init S0;
trans S0 -> S0{guard x<2000; assign x:=x+1; };
}
process Q{
state S1;
init S1;
trans S1 -> S1{guard x>0; assign x:=x-1; };
}
process R{
state S2;
init S2;
trans S2 -> S2{guard x==2000; assign x:=0; };
}
p1:=P();
q1:=Q();
r1:=R();
system p1,q1,r1;
```

Int x

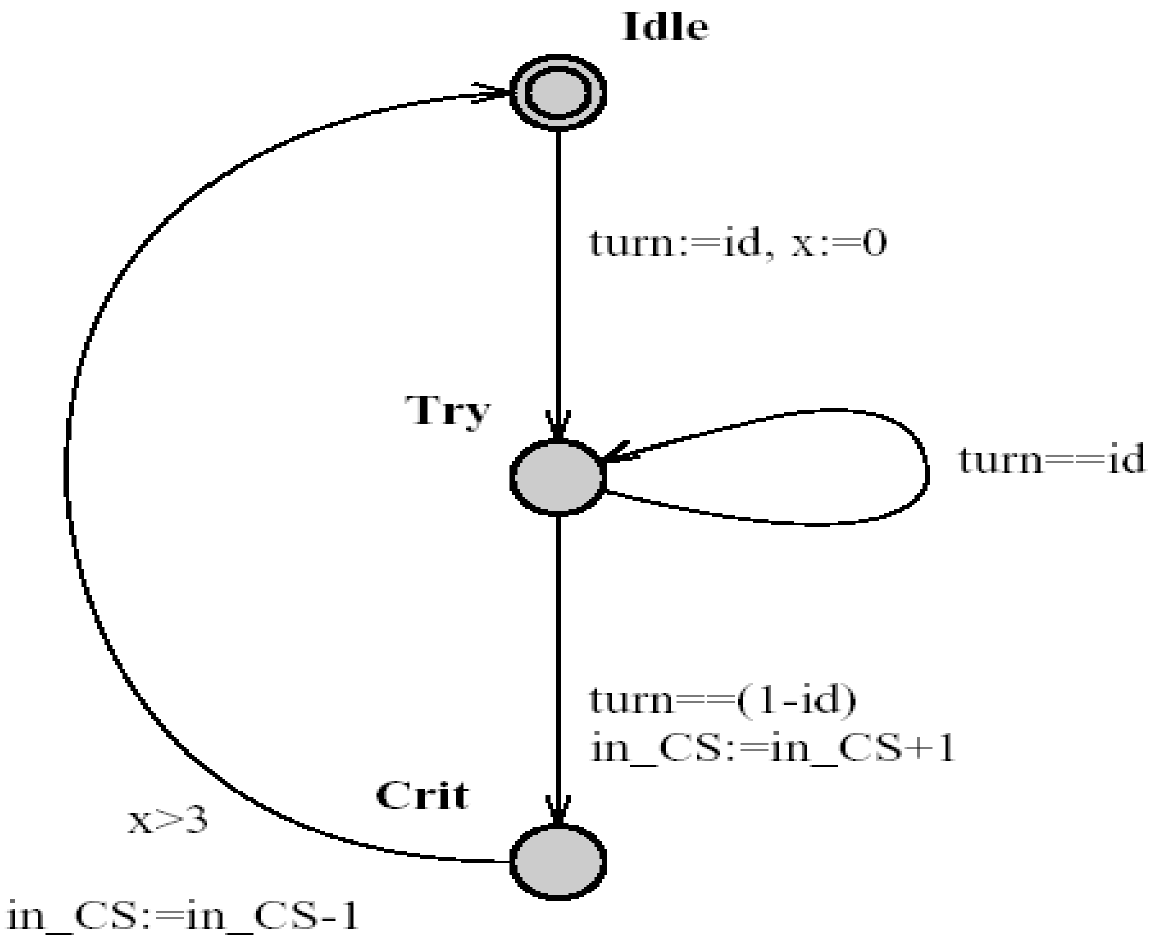
Process P
 do
 :: x<2000 → x:=x+1
 od

Process Q
 do
 :: x>0 → x:=x-1
 od

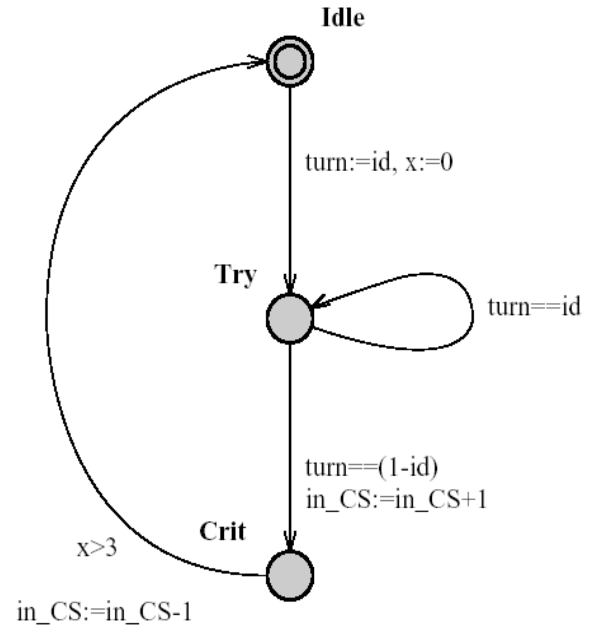
Process R
 do
 :: x=2000 → x:=0
 od

fork P; **fork** Q; **fork** R

Example: Mutual Exclusion



Example (mutex2.xta)



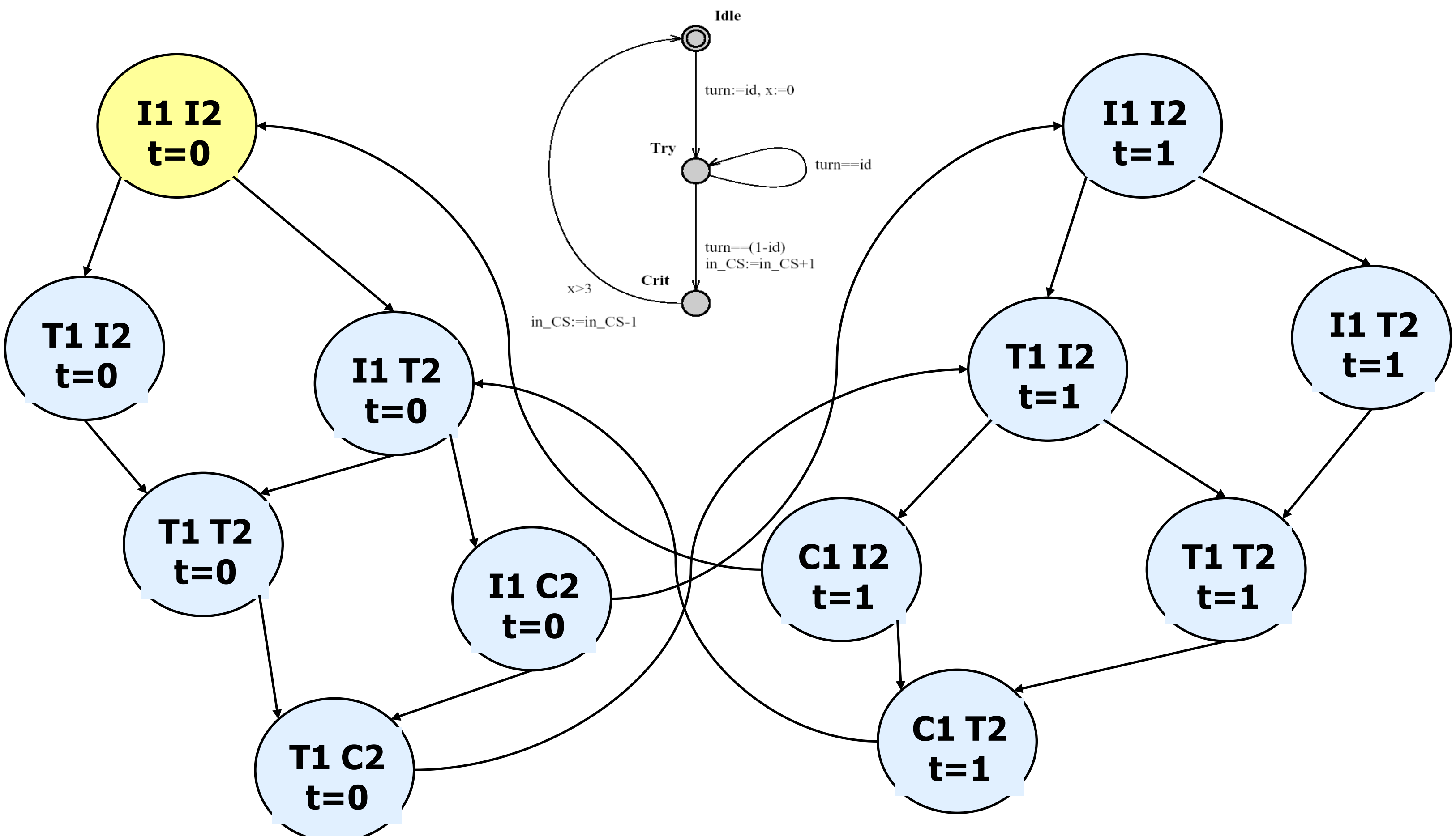

```
//Global declarations
int turn;
int in_CS;

//Process template
process P(const id){
clock x;
state Idle, Try, Crit;
init Idle;
trans Idle -> Try{assign turn:=id, x:=0; },
Try -> Crit{guard turn==(1-id); assign in_CS:=in_CS+1; },
Try -> Try{guard turn==id; },
Crit -> Idle{guard x>3; assign in_CS:=in_CS-1; };
}

//Process assignments
P1:=P(1);
P2:=P(0);

//System definition.
system P1, P2;
```

From UPPAAL$_{-time}$ Models to *Kripke Structures*



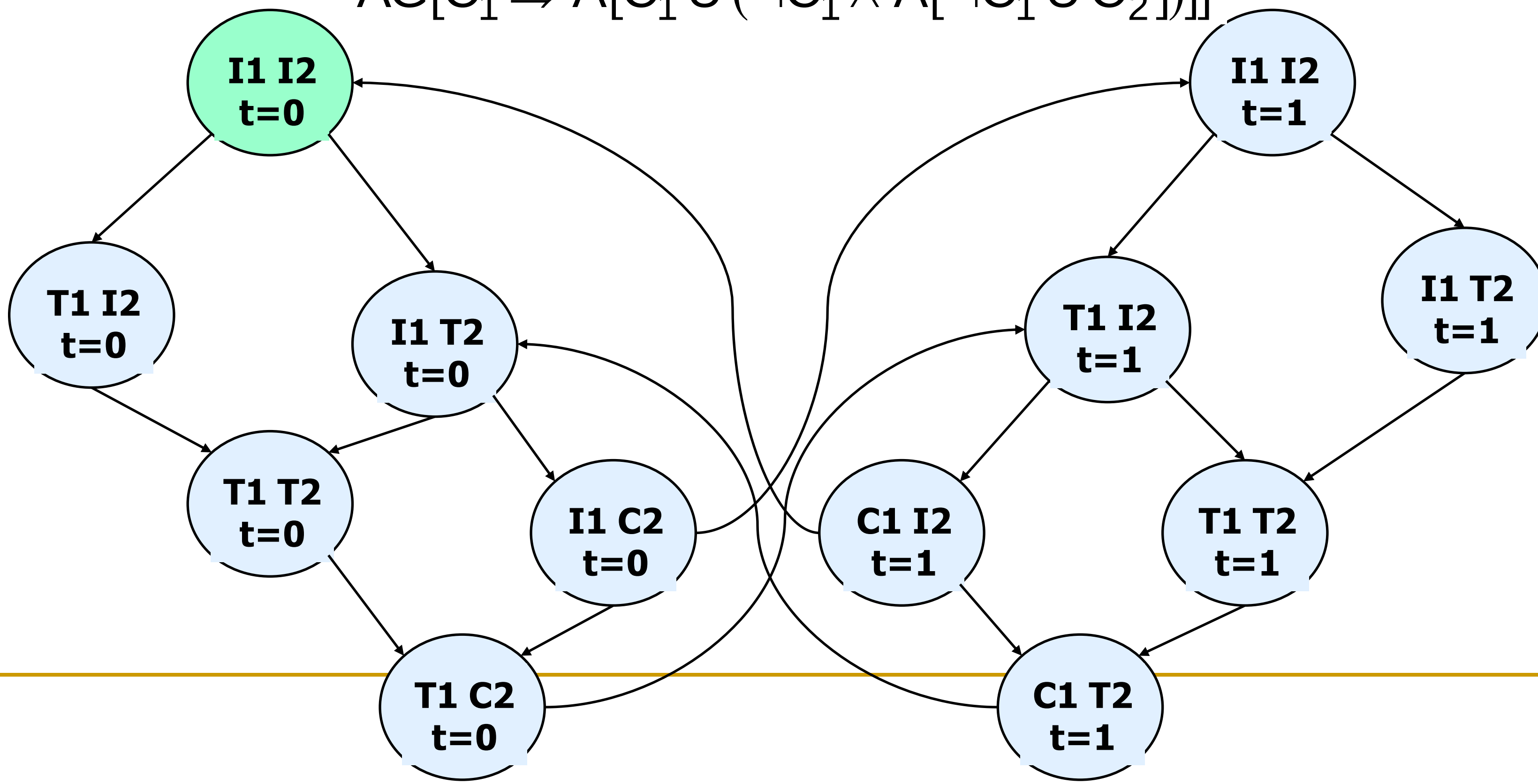
Properties of MUTEX example ?

$$AG\ \neg(C_1 \wedge C_2)$$

$$AG[\ T_1 \Rightarrow AF(C_1)]$$

$$EG[\neg C_1]$$

$$AG[C_1 \Rightarrow A[C_1\ U\ (\neg C_1 \wedge A[\neg C_1\ U\ C_2\ ])]]$$



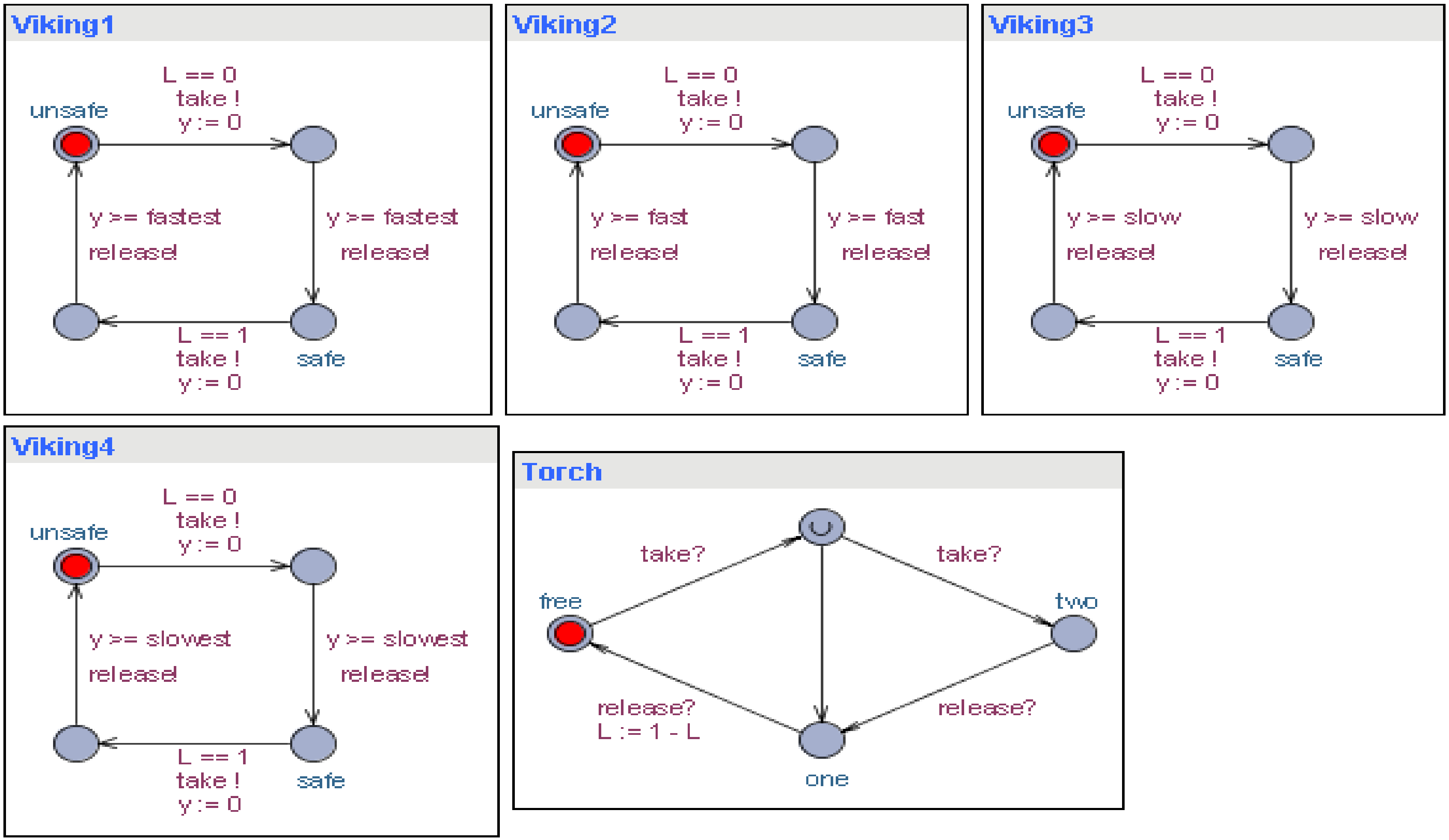
Example: Vikings' Problem

- Four vikings are about to cross a damaged bridge in the middle of the night.
- The bridge can only carry two of the vikings at the time and to find the way over the bridge the vikings need to bring a torch.
- The vikings need at least 5, 10, 20 and 25 minutes (one-way) respectively to cross the bridge.

- Does a schedule exist which gets all four vikings over the bridge within 60 minutes? What is the minimum time required to get all four vikings over the bridge?

Example: Vikings' Problem Model

- Model the four vikings as four processes, and the torch as a single process (see bridge.xml).



Example: Vikings' Problem

- Four vikings are about to cross a damaged bridge in the middle of the night.
- The bridge can only carry two of the vikings at the time and to find the way over the bridge the vikings need to bring a torch.
- The vikings need 5, 10, 20 and 25 minutes (one-way) respectively to cross the bridge.

- **Question:** What is the minimum time required for all four vikings to safely cross the bridge?
- **Answer:** 60 minutes: use E<>(Viking1.safe and Viking2.safe and Viking3.safe and Viking4.safe) with the additional Option: **Diagnostic Trace: Fastest.**

Urgent (U) vs Committed (C) Locations

- **Urgent locations -** Urgent locations freeze time; *i.e.* time is not allowed to pass when a process is in an urgent location. Semantically, urgent locations are equivalent to:
  - adding an extra clock, **x**, that is reset on every incoming edge, and
  - adding an invariant **x <= 0** to the location.
- **Committed locations -** Like urgent locations, committed locations freeze time. Furthermore, if any process is in a committed location, the next transition must involve an edge from one of the committed locations.
- Committed locations are useful for creating atomic sequences and for encoding synchronization between more than two components. Notice that if several processes are in a committed location at the same time, then they will interleave.

Computation Tree Logic (CTL)

CTL Models

A CTL-model is a triple $\mathcal{M} = (S, R, Label)$ where

- S is a non-empty set of states,
- $R \subseteq S \times S$ is a total relation on S, which relates to $s \in S$ its possible successor states,
- $Label : S \longrightarrow 2^{AP}$, assigns to each state $s \in S$ the atomic propositions $Label(s)$ that are valid in s.

Computation Tree Logic, CTL
(Clarke and Emerson, 1980)

Syntax

$$\boxed{\phi \quad ::= \quad p \mid \neg\,\phi \mid \phi \vee \phi \mid \mathsf{EX}\,\phi \mid \mathsf{E}\,[\phi\,\mathsf{U}\,\phi] \mid \mathsf{A}\,[\phi\,\mathsf{U}\,\phi].}$$

- EX (pronounced "for some path next")
- E (pronounced "for some path")
- A (pronounced "for all paths") and
- U (pronounced "until").

Example
(from UPPAAL2k: Small Tutorial)

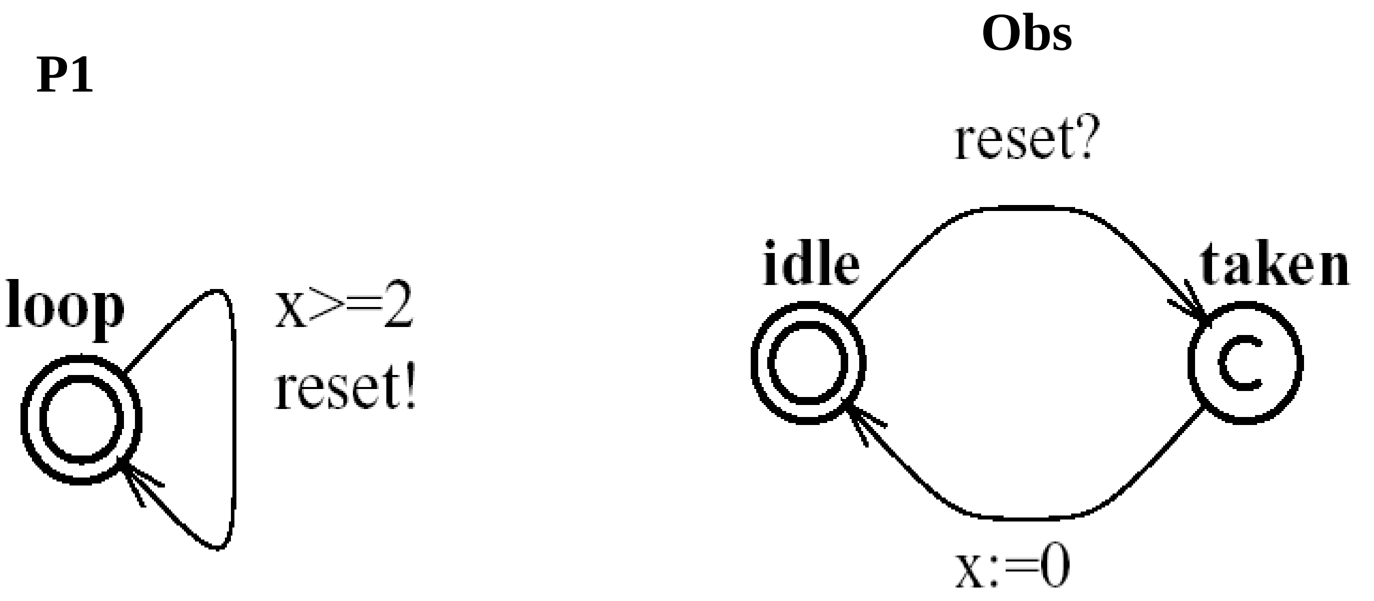
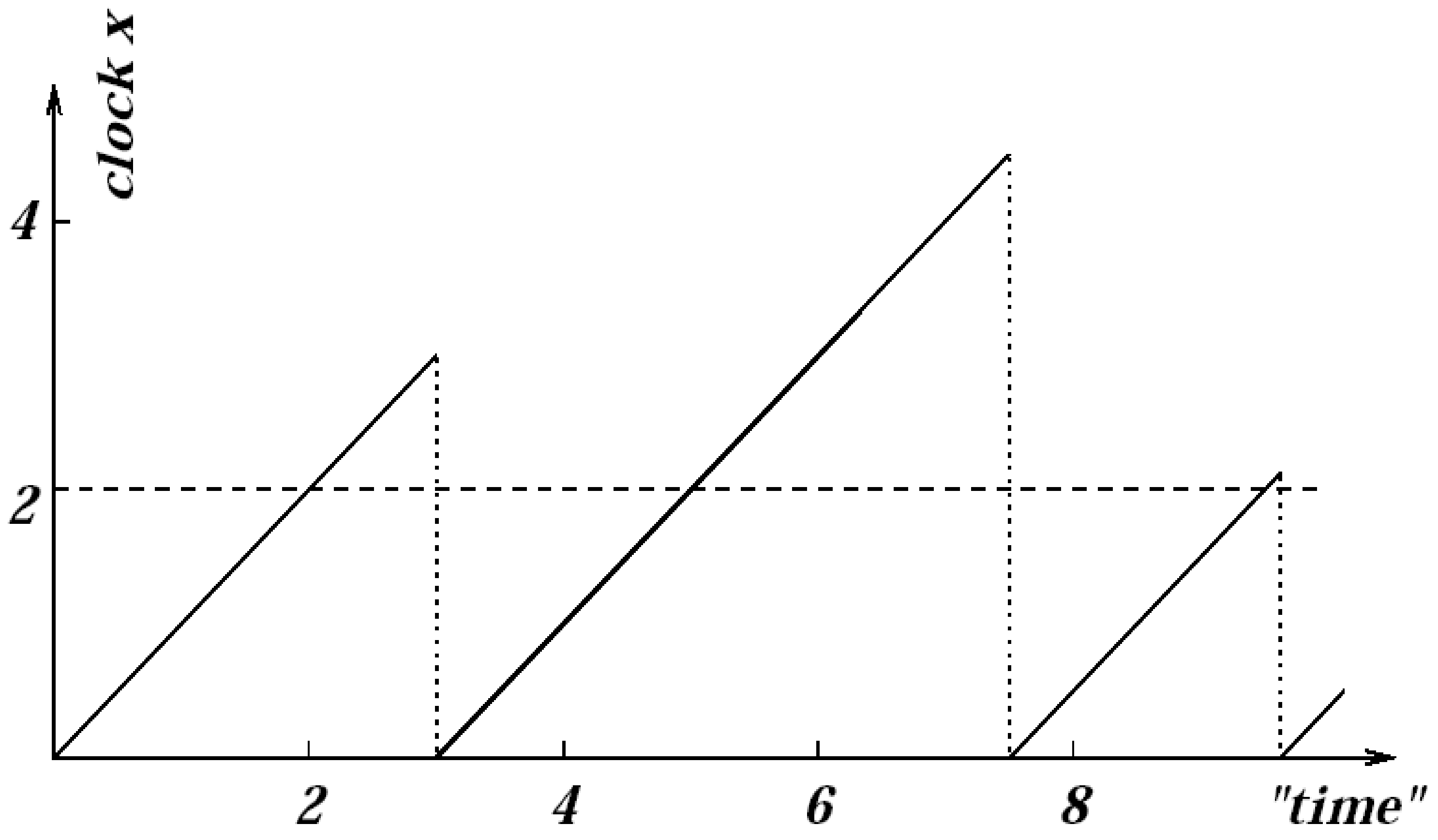


Figure 5: First example with the observer.

Example (cont.)



Example (cont.)

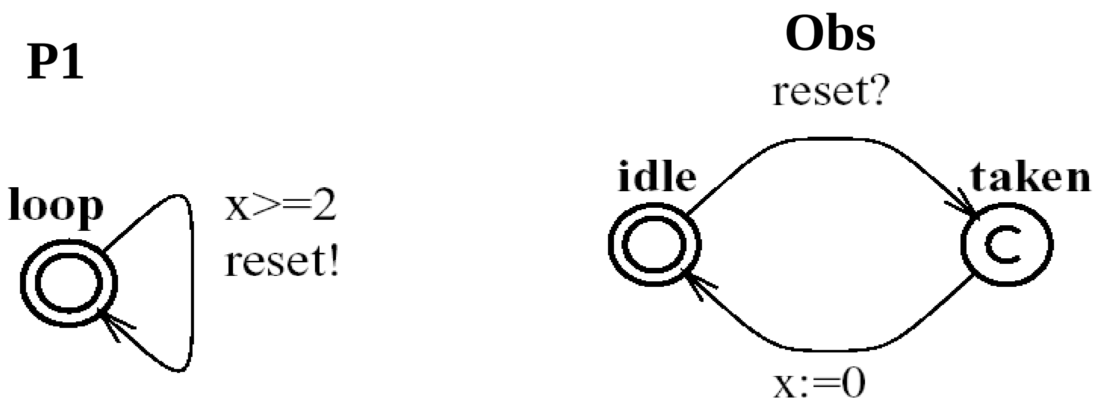


Figure 5: First example with the observer.

- Verification:
  - A[](Obs.taken imply x>=2)
  - E<>(Obs.idle and x>3) - for some path E, there is eventually <> a state in which Obs is in the idle state and x > 3.
  - E<>(Obs.idle and x>3000)

Example (cont.)

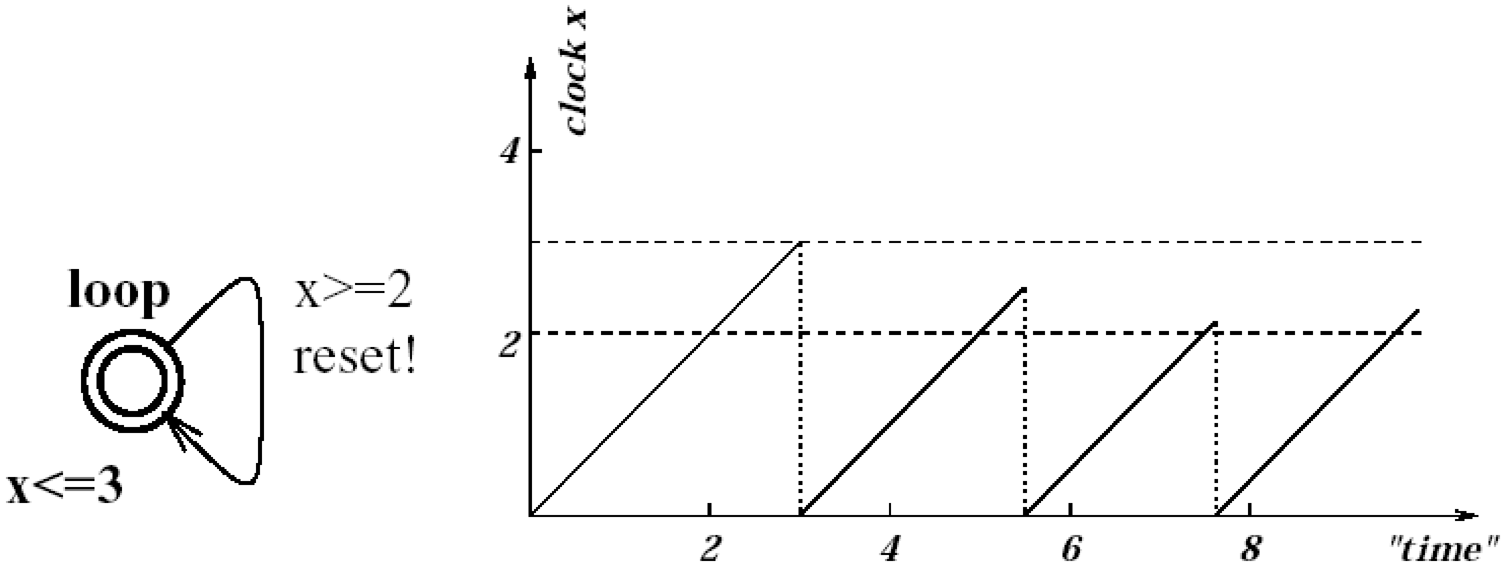


Figure 7: Adding an invariant: the new behaviour.

Example (cont.)

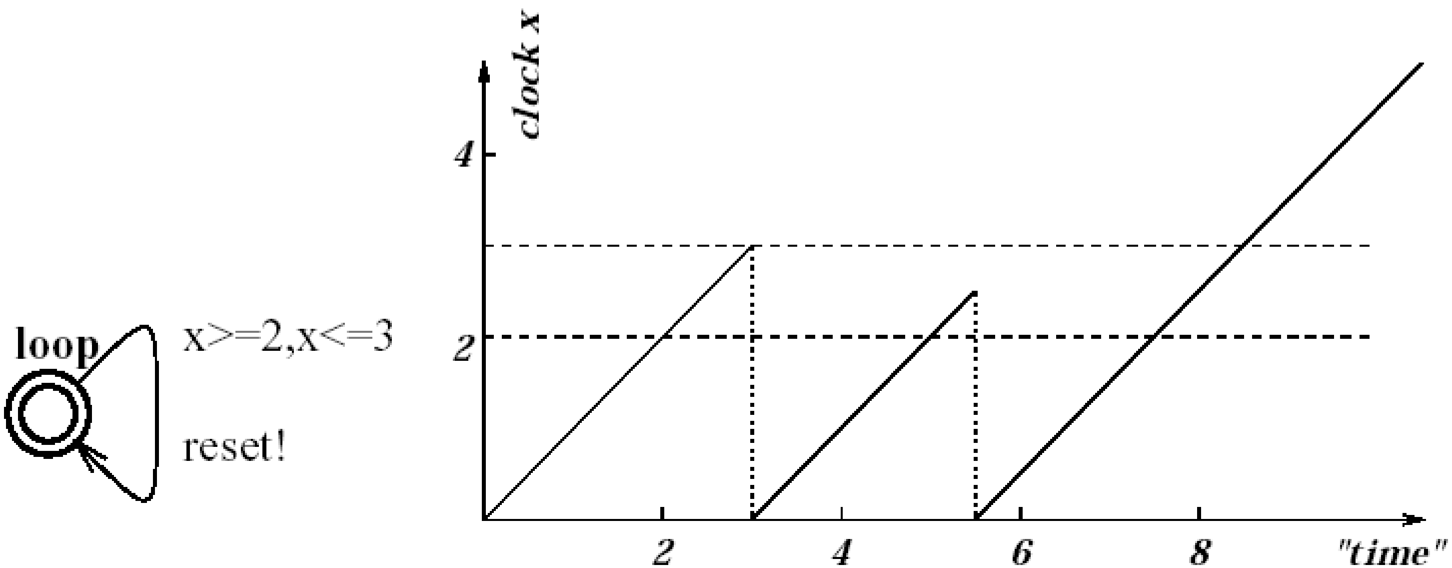
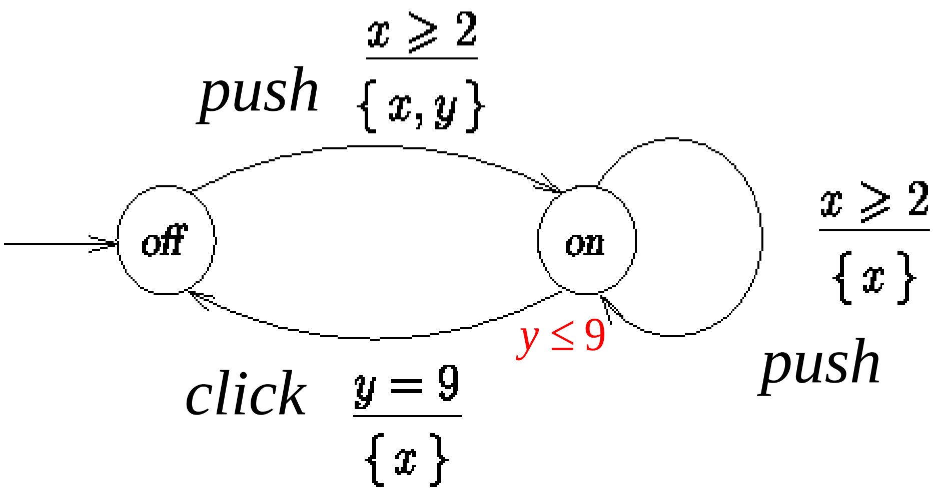


Figure 8: No invariant and a new guard: the new behaviour.

Timed CTL (TCTL)

Light Switch



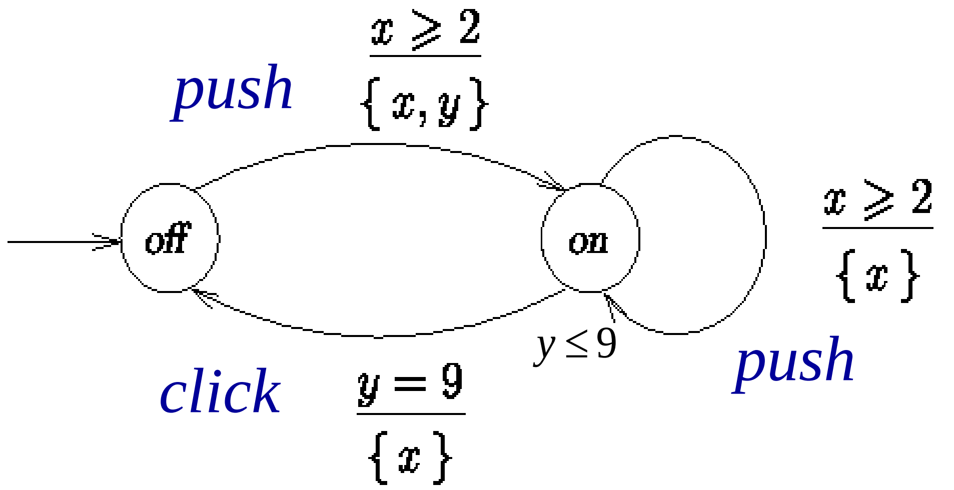
- Switch may be turned on whenever at least 2 time units has elapsed since last “turn off”
- Light automatically switches off after 9 time units.

Semantics

- *clock valuations*: $V(C) \quad v : C \to R_{\geq 0}$
- *state*: $(l, v) \quad where \quad l \in L \;\; and \;\; v \in V(C)$
- Semantics of timed automata is a *labeled transition system* $(S, \to)$ where
  $$S = \{ \; (l, v) \;\; | v \in V(C) \;\; and \;\; l \in L \; \}$$

- *action transition*
  $$(l, v) \xrightarrow{a} (l', v') \;\; iff$$
  $$g(v) \;\; and \;\; v' = v[r] \;\; and \;\; Inv(l')(v')$$
- *delay Transition*
  $$(l, v) \xrightarrow{d} (l, v + d) \; iff$$
  $$Inv(l)(v + d') \;\; whenever \;\; d' \leq d \in R_{\geq 0}$$

Semantics: Example



$$(\mathit{off}, x = y = 0) \xrightarrow{3.5} (\mathit{off}, x = y = 3.5) \xrightarrow{\mathit{push}}$$
$$(\mathit{on}, x = y = 0) \xrightarrow{\pi} (\mathit{on}, x = y = \pi) \xrightarrow{\mathit{push}}$$
$$(\mathit{on}, x = 0, y = \pi) \xrightarrow{3} (\mathit{on}, x = 3, y = \pi + 3) \xrightarrow{9-(\pi+3)}$$
$$(\mathit{on}, x = 9 - (\pi + 3), y = 9) \xrightarrow{\mathit{click}} (\mathit{off}, x = 0, y = 9) \ldots$$

TCTL = CTL + Time

$$\phi ::= p \mid \alpha \mid \neg\phi \mid \phi \vee \phi \mid z \textbf{ in } \phi \mid \mathsf{E}[\phi \,\mathsf{U}\, \phi] \mid \mathsf{A}[\phi \,\mathsf{U}\, \phi]$$

α – constraints over clocks
z – formula clocks

Derived Operators

$\boxed{A\,[\phi\, U_{\leqslant 7}\, \psi]} \quad = \quad z \text{ in } A\,[(\phi \;\wedge\; z \leqslant 7)\, U\, \psi].$

Along any path, ϕ holds continuously until ψ becomes valid within 7 time units.

$\boxed{EF_{<5}\,\phi} \quad = \quad z \text{ in } EF\,(z < 5 \;\wedge\; \phi)$

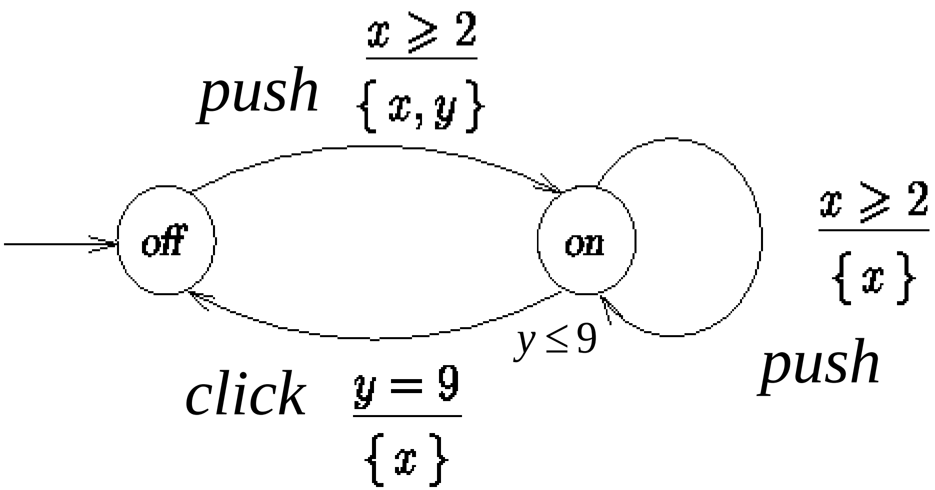
The property ϕ may become valid within 5 time units.

Light Switch (cont.)

`A[] (x <= y)` $\quad \mathsf{AG}(x \leq y)$

`P.on --> P.off` $\quad \mathsf{AG}(on \Rightarrow \mathsf{AF}\, off)$

$\mathsf{AG}(on \Rightarrow \mathsf{AF}_{\leq 9}\, off)$



$\mathsf{A}[off \;\mathsf{U}\; x \geq 2]$

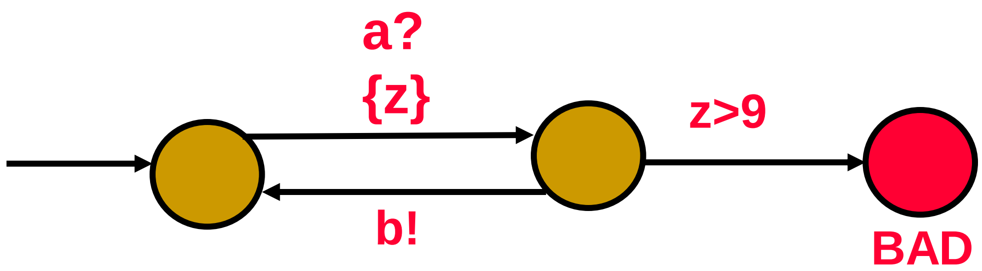
$\mathsf{A}[off \;\mathsf{U}\; x \geq 3]$

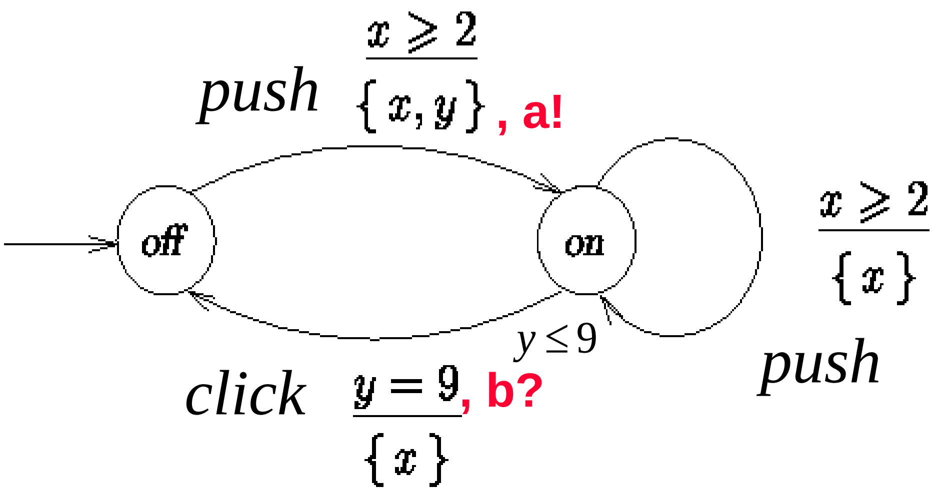
$\mathsf{E}[off \;\mathsf{U}\; x \geq 3]$

$\mathsf{A}[x \leq 2 \;\mathsf{U}\; on]$

$\mathsf{E}[x \leq 2 \;\mathsf{U}\; on]$

Light Switch (Add Observer)





$\mathsf{AG}(x \leq y)$

$\mathsf{AG}(on \Rightarrow \mathsf{AF}\, off)$

$\boxed{\mathsf{AG}(on \Rightarrow \mathsf{AF}_{\leq 9}\, off)}$

$\mathsf{A}[off \;\mathsf{U}\; x \geq 2]$

$\mathsf{A}[off \;\mathsf{U}\; x \geq 3]$

$\mathsf{E}[off \;\mathsf{U}\; x \geq 3]$

$\mathsf{A}[x \leq 2 \;\mathsf{U}\; on]$

$\mathsf{E}[x \leq 2 \;\mathsf{U}\; on]$

Timeliness Properties

$$\boxed{\mathsf{AG}\,[\mathit{send}(m) \;\Rightarrow\; \mathsf{AF}_{<5}\,\mathit{receive}\,(r_m)]}$$

receive(m) always occurs within 5 time units after *send(m)*

$$\boxed{\mathsf{EG}\,[\mathit{send}(m) \;\Rightarrow\; \mathsf{AF}_{=11}\,\mathit{receive}\,(r_m)]}$$

receive(m) may occur exactly 11 time units after *send(m)*

$$\boxed{\mathsf{AG}\,[\mathsf{AF}_{=25}\,\mathit{putbox}]}$$

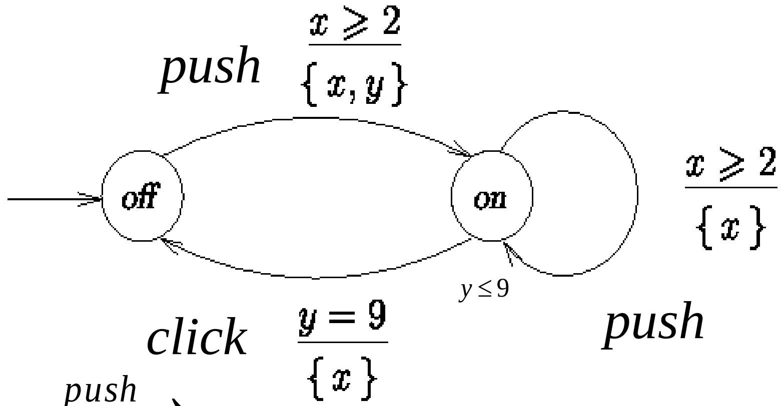
putbox occurs periodically (exactly) every 25 time units

(note: other *putbox*'s may occur in between)

Paths

A *path* is an infinite sequence $s_0\, a_0\, s_1\, a_1\, s_2\, a_2\, \ldots$ of states alternated by transition labels such that $s_i \xrightarrow{a_i} s_{i+1}$ for all $i \geqslant 0$.

Example Path:



$$(\mathit{off}, x = y = 0) \xrightarrow{3.5} (\mathit{off}, x = y = 3.5) \xrightarrow{push}$$

$$(on, x = y = 0) \xrightarrow{\pi} (on, x = y = \pi) \xrightarrow{push}$$

$$(on, x = 0, y = \pi) \xrightarrow{3} (on, x = 3, y = \pi + 3) \xrightarrow{9-(\pi+3)}$$

$$(on, x = 9 - (\pi + 3), y = 9) \xrightarrow{click} (\mathit{off}, x = 0, y = 9) \ldots$$

Elapsed Time in Path

$$\boxed{\begin{aligned} \Delta(\sigma,0) &= 0 \\ \Delta(\sigma,i{+}1) &= \Delta(\sigma,i) + \begin{cases} 0 & \text{if } a_i = * \\ a_i & \text{if } a_i \in \mathbb{R}^+. \end{cases} \end{aligned}}$$

Example:

$$\sigma = (\mathit{off}, x = y = 0) \xrightarrow{3.5} (\mathit{off}, x = y = 3.5) \xrightarrow{push}$$

$$(on, x = y = 0) \xrightarrow{\pi} (on, x = y = \pi) \xrightarrow{push}$$

$$(on, x = 0, y = \pi) \xrightarrow{3} (on, x = 3, y = \pi + 3) \xrightarrow{9-(\pi+3)}$$

$$(on, x = 9 - (\pi + 3), y = 9) \xrightarrow{click} (\mathit{off}, x = 0, y = 9) \ldots$$

$\Delta(\sigma,1)=3.5$, $\Delta(\sigma,6)=3.5+9=12.5$

TCTL Semantics

$$
\begin{array}{ll}
s, w \models p & \text{iff } p \in \mathit{Label}(s) \\
s, w \models \alpha & \text{iff } v \cup w \models \alpha \\
s, w \models \neg\,\phi & \text{iff } \neg\,(s, w \models \phi) \\
s, w \models \phi \,\vee\, \psi & \text{iff } (s, w \models \phi) \,\vee\, (s, w \models \psi) \\
s, w \models z \text{ in } \phi & \text{iff } s, \textsf{reset } z \text{ in } w \models \phi \\
s, w \models \mathsf{E}\,[\phi\,\mathsf{U}\,\psi] & \text{iff } \exists\, \sigma \in P^{\infty}_{\mathcal{M}}(s).\, \exists\,(i, d) \in \mathit{Pos}(\sigma). \\
& \quad (\sigma(i, d), w{+}\Delta(\sigma, i) \models \psi \,\wedge \\
& \quad (\forall\,(j, d') \ll (i, d).\, \sigma(j, d'), w{+}\Delta(\sigma, j) \models \phi \,\vee\, \psi)) \\
s, w \models \mathsf{A}\,[\phi\,\mathsf{U}\,\psi] & \text{iff } \forall\, \sigma \in P^{\infty}_{\mathcal{M}}(s).\, \exists\,(i, d) \in \mathit{Pos}(\sigma). \\
& \quad ((\sigma(i, d), w{+}\Delta(\sigma, i)) \models \psi \,\wedge \\
& \quad (\forall\,(j, d') \ll (i, d).\, (\sigma(j, d'), w{+}\Delta(\sigma, j)) \models \phi \,\vee\, \psi)).
\end{array}
$$

s - (location, clock valuation)

w - formula clock valuation

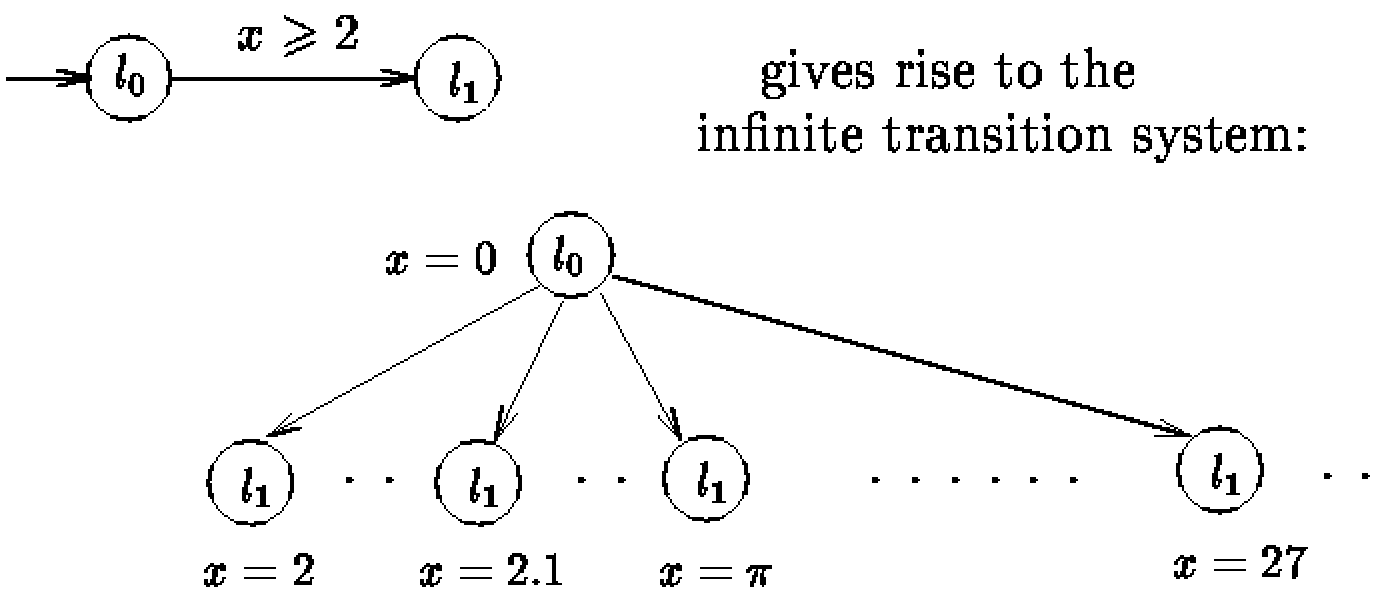
$P^{\infty}_{\mathrm{M}}(s)$ - set of paths from *s*

Pos(σ) - positions in σ

Δ(σ,i) - elapsed time

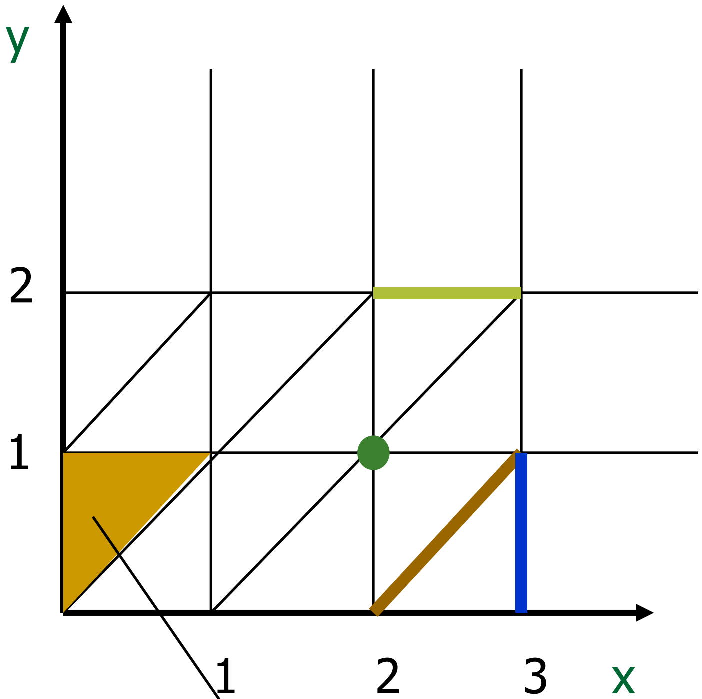
(i,d) <<(i',d') iff (i<j) or ((i=j) and (d<d'))

Infinite State Space?



Regions

Finite partitioning of state space



Definition

$w \approx w'$ iff w and w' satisfy the exact same conditions of the form

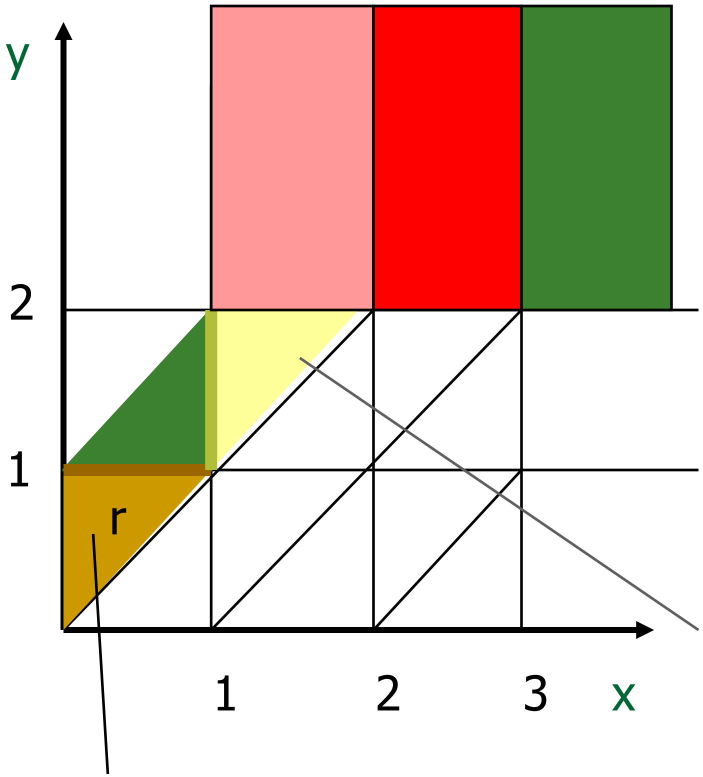
$x_i \leq n$ and $x_i - x_j \leq n$

where $n \leq \max$

An equivalence class (i.e. a *region*)
in fact there are only a *finite* number of regions!

Regions

Finite partitioning of state space



Definition

$w \approx w'$ iff w and w' satisfy the exact same conditions of the form

$x_i \leq n$ and $x_i - x_j \leq n$

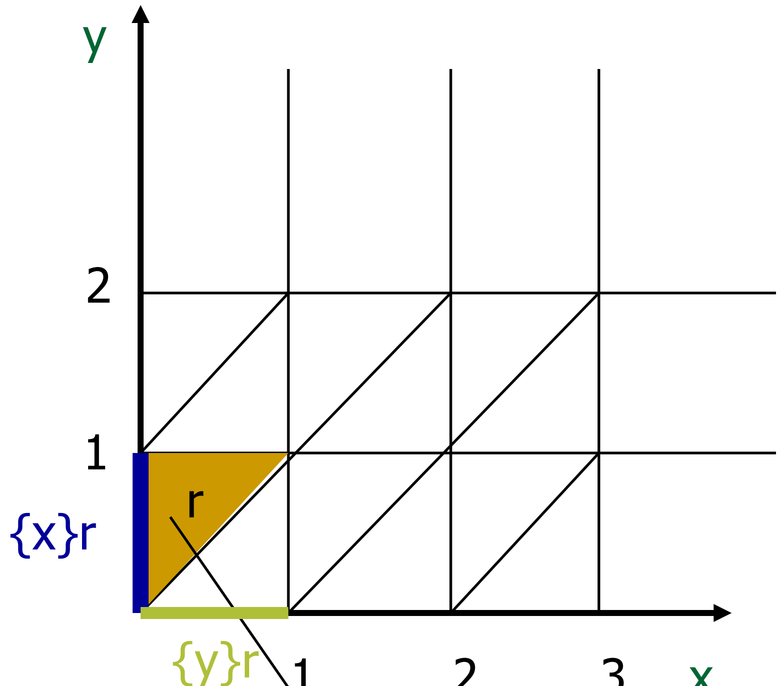
where $n \leq \max$

Successor regions, Succ(r)

An equivalence class (i.e. a *region*)

Regions

Finite partitioning of state space



Reset regions

An equivalence class (i.e. a *region*) r

Definition

$w \approx w'$ iff w and w' satisfy the exact same conditions of the form

$x_i \leq n$ and $x_i - x_j \leq n$

where $n \leq \max$

THEOREM

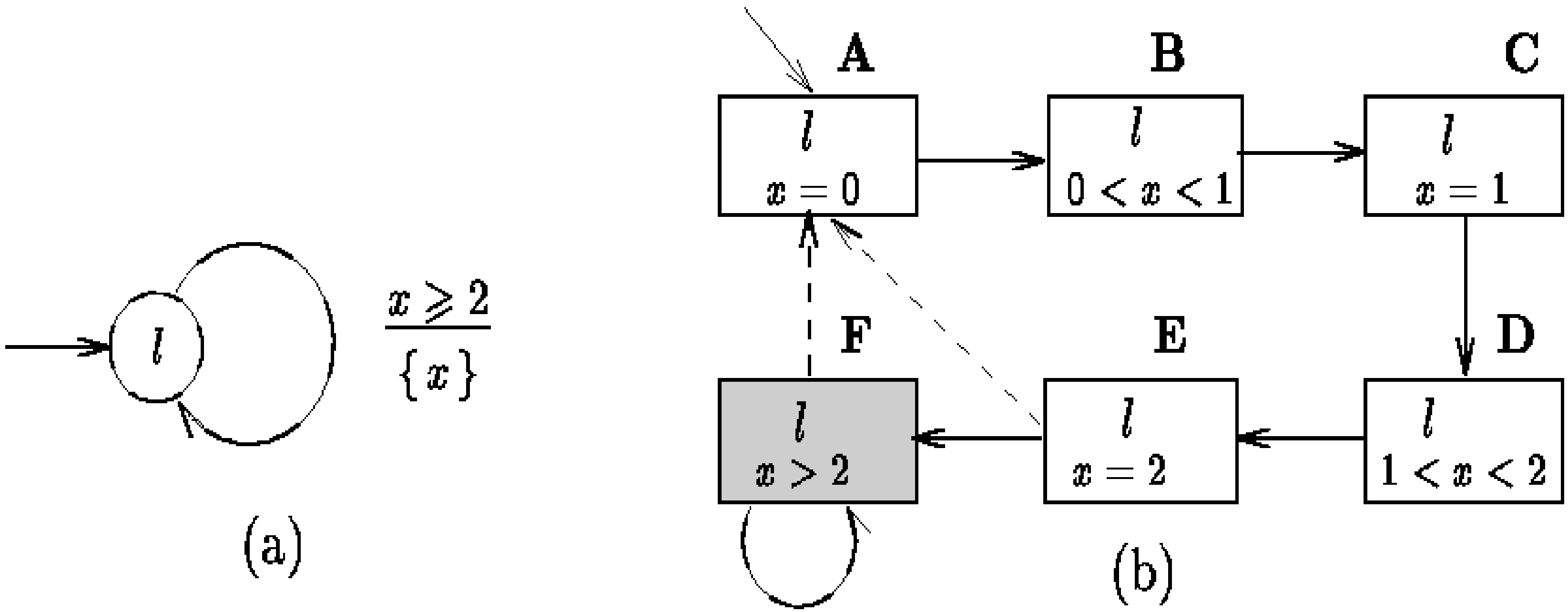
Whenever $uv \approx u'v'$ then

$[(l,u),v]$ sat ϕ

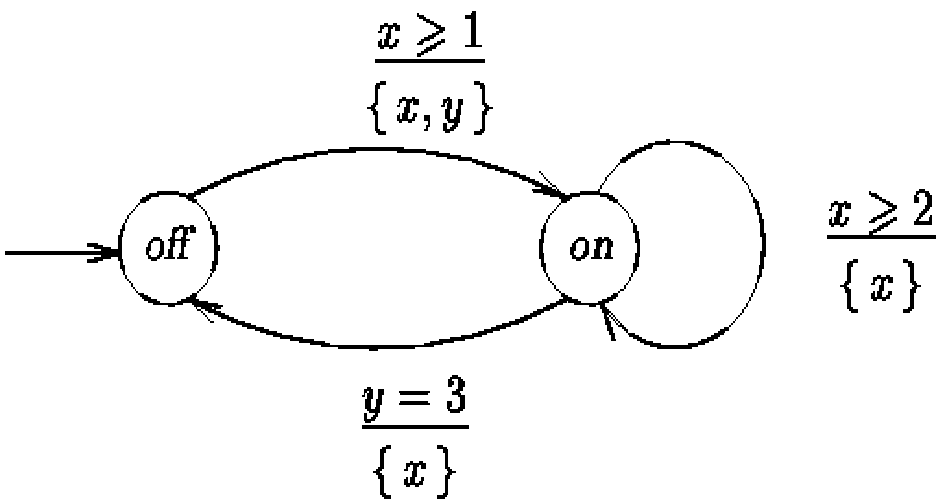
$\Leftrightarrow$

$[(l,u'),v']$ sat ϕ

Region graph of simple timed automata

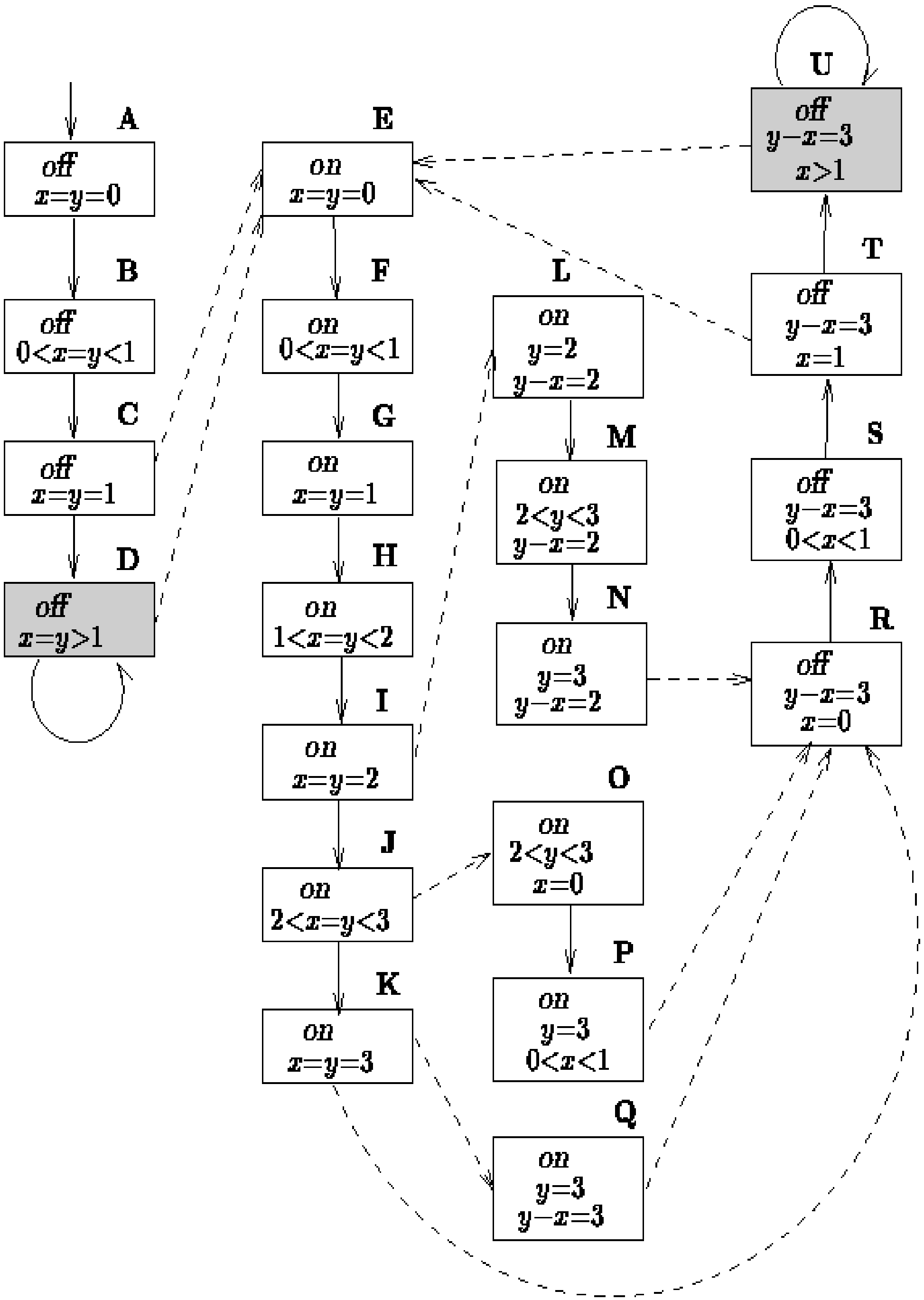


Modified light switch



$inv(off) = \mathrm{true}$

$inv(on) = y \leqslant 3$



Reachable part of region graph

Properties

$$AG(x \leq y)$$

$$AG(on \Rightarrow AF\,off\,)$$

$$AG(on \Rightarrow AF_{\leq 3}off\,)$$

Roughly speaking....

Model checking a timed automata against a TCTL-formula amounts to model checking its region graph against a CTL-formula

Problem to be solved

The worst-case time complexity of model checking TCTL*-formula ϕ over timed automaton $\mathcal{A}$, with the clock constraints of ϕ and $\mathcal{A}$ in Ψ is:*

$$\mathcal{O}\left(|\phi| \times \left(n! \times 2^n \times \prod_{x\in\Psi} c_x \times |L|^2\right)\right).$$

☺ (i) linear in the length of the formula ϕ

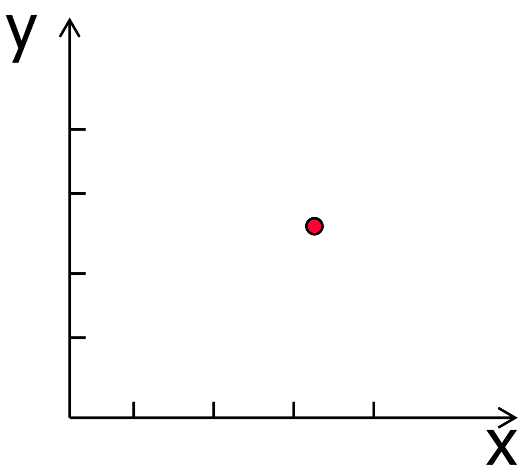
😐 (ii) exponential in the number of clocks in $\mathcal{A}$ and ϕ

☹ (iii) exponential in the maximal constants with which clocks are compared in $\mathcal{A}$ and ϕ.

Model Checking TCTL is PSPACE-hard

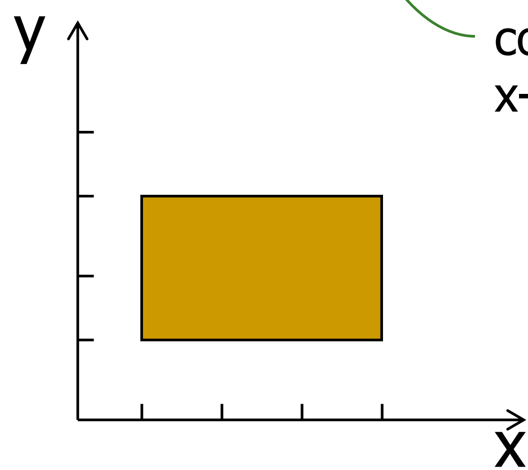
Zones: From Infinite to Finite

State
(n, $x=3.2$, $y=2.5$)

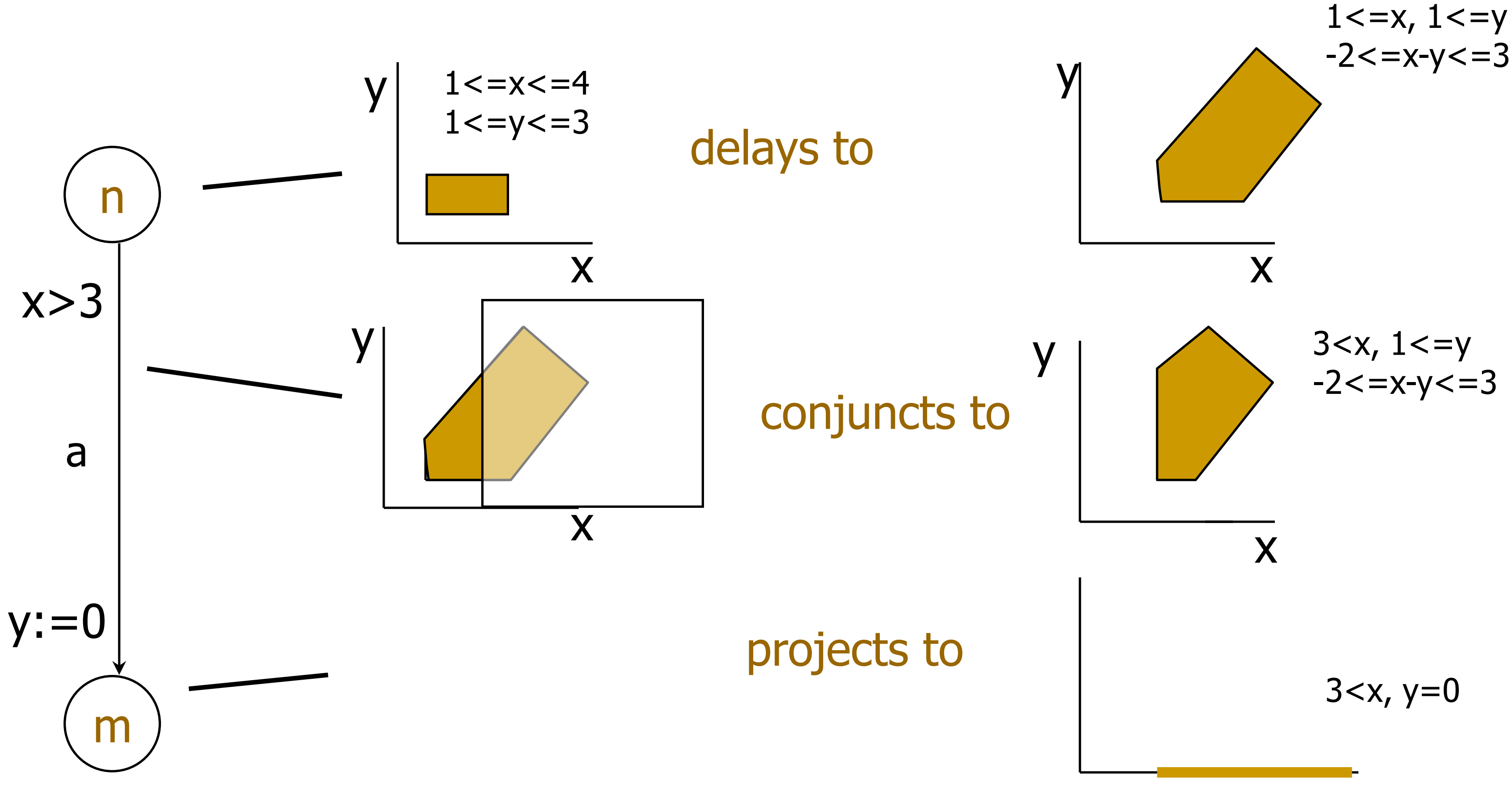


Symbolic state (set)
(n, $1 \le x \le 4, 1 \le y \le 3$)

Zone:
conjunction of
x-y<=n, x<=>n



Symbolic Transitions



Thus (n,1<=x<=4,1<=y<=3) =a => (m, 3<x, y=0)

Summary

- **Next Time:** UPPAAL Internals