Modelling and Verification

Timed Automata: A Formalism for Real-time Systems

- Labelled transition systems with time
- Timed automata
- Timed and untimed bisimilarity
- Timed and untimed language equivalence
- Region graph and the reachability problem
- Networks of timed automata
- Model checking of timed automata

Need for Introducing Time Features

- Timeouts in protocols:
  - In CCS timeouts were modelled using nondeterminism.
  - Enough to prove that the protocol is safe.
  - Maybe too abstract for certain questions (What is the average time to deliver the message?).
- Many real-life systems depend on timing:
  - Real-time controllers (production lines, computers in cars, railway crossings).
  - Embedded systems (mobile phones, remote controllers, digital watch).
  - ...

Labelled Transition Systems with Time

Timed (labelled) transition system (TLTS)

TLTS is a triple $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ where

- $Proc$ is a set of states (or processes),
- $Act = N \cup \mathbb{R}^{\geq 0}$ is a set of actions (consisting of labels and time-elapsing steps), and
- for every $a \in Act$, $\xrightarrow{a} \subseteq \ Proc \times Proc$ is a binary relation on states called the transition relation.

We write

- $s \xrightarrow{a} s'$ if $a \in N$ and $(s, s') \in \xrightarrow{a}$, and
- $s \xrightarrow{d} s'$ if $d \in \mathbb{R}^{\geq 0}$ and $(s, s') \in \xrightarrow{d}$.

How Can One Describe Timed Transition Systems?

Syntax		Semantics
unknown entity	$\longrightarrow$	known entity
CCS	$\longrightarrow$	Labelled Transition Systems
???	$\longrightarrow$	Timed Transition Systems

Timed Automata [Alur, Dill'90]

Finite-state automata equipped with clocks.

How Can One Describe Timed Transition Systems?

Syntax		Semantics
unknown entity	$\longrightarrow$	known entity
CCS	$\longrightarrow$	Labelled Transition Systems
???	$\longrightarrow$	Timed Transition Systems

Timed Automata [Alur, Dill'90]

Finite-state automata equipped with clocks.

How Can One Describe Timed Transition Systems?

Syntax		Semantics
unknown entity	$\longrightarrow$	known entity
CCS	$\longrightarrow$	Labelled Transition Systems
???	$\longrightarrow$	Timed Transition Systems

Timed Automata [Alur, Dill'90]

Finite-state automata equipped with clocks.

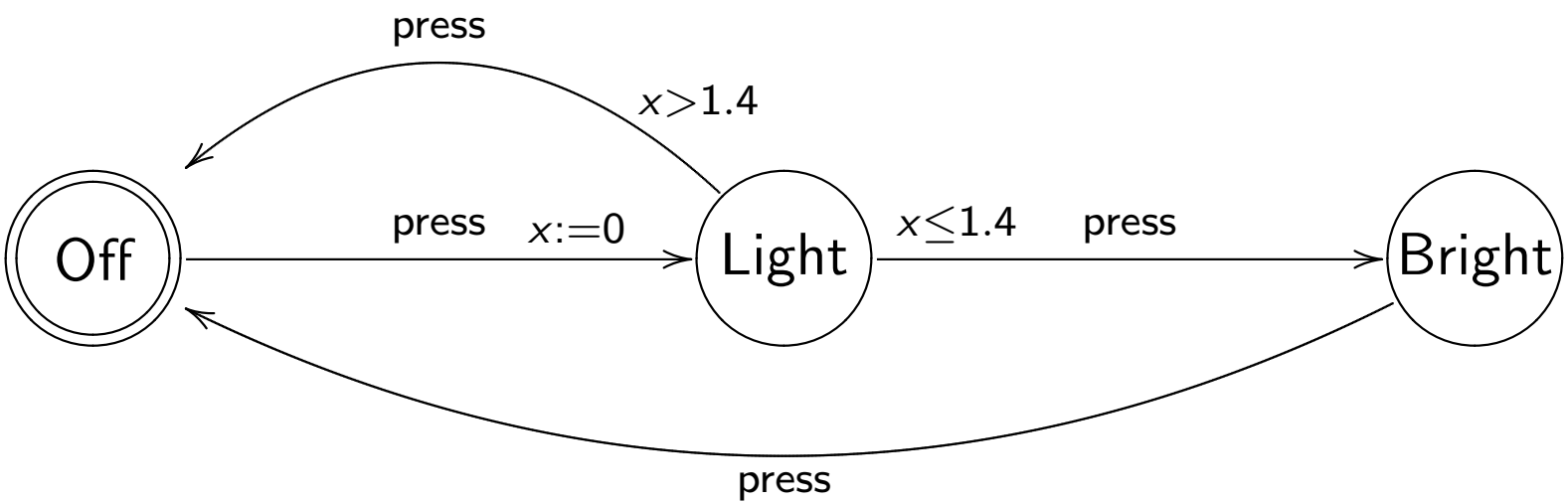
How Can One Describe Timed Transition Systems?

Syntax		Semantics
unknown entity	$\longrightarrow$	known entity
CCS	$\longrightarrow$	Labelled Transition Systems
???	$\longrightarrow$	Timed Transition Systems

Timed Automata [Alur, Dill'90]

Finite-state automata equipped with clocks.

Example: Light switch



Definition of TA: Clock Constraints

Let $C = \{x, y, \ldots\}$ be a finite set of clocks.

Set $\mathcal{B}(C)$ of clock constraints over C

$\mathcal{B}(C)$ is defined by the following abstract syntax

$$g, g_1, g_2 \ ::= x \sim n \ \mid \ x - y \sim n \ \mid \ g_1 \wedge g_2$$

where $x, y \in C$ are clocks, $n \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$.

Example: $x \leq 3 \ \wedge \ y > 0 \ \wedge \ y - x = 2$

Clock Valuation

Clock valuation

Clock valuation v is a function $v : C \to \mathbb{R}^{\geq 0}$.

Let v be a clock valuation. Then

- $v + d$ is a clock valuation for any $d \in \mathbb{R}^{\geq 0}$ and it is defined by

$$(v + d)(x) = v(x) + d \text{ for all } x \in C$$

- $v[r]$ is a clock valuation for any $r \subseteq C$ and it is defined by

$$v[r](x) = \begin{cases} 0 & \text{if } x \in r \\ v(x) & \text{otherwise.} \end{cases}$$

Clock Valuation

Clock valuation

Clock valuation v is a function $v : C \to \mathbb{R}^{\geq 0}$.

Let v be a clock valuation. Then

- $v + d$ is a clock valuation for any $d \in \mathbb{R}^{\geq 0}$ and it is defined by

$$(v + d)(x) = v(x) + d \text{ for all } x \in C$$

- $v[r]$ is a clock valuation for any $r \subseteq C$ and it is defined by

$$v[r](x) = \begin{cases} 0 & \text{if } x \in r \\ v(x) & \text{otherwise.} \end{cases}$$

Evaluation of Clock Constraints

Evaluation of clock constraints ($v \models g$)

$$
\begin{array}{ll}
v \models x < n & \text{iff } v(x) < n \\
v \models x \leq n & \text{iff } v(x) \leq n \\
v \models x = n & \text{iff } v(x) = n \\
\vdots & \\
v \models x - y < n & \text{iff } v(x) - v(y) < n \\
v \models x - y \leq n & \text{iff } v(x) - v(y) \leq n \\
\vdots & \\
v \models g_1 \wedge g_2 & \text{iff } v \models g_1 \text{ and } v \models g_2
\end{array}
$$

Syntax of Timed Automata

Definition

A timed automaton over a set of clocks C and a set of labels N is a tuple

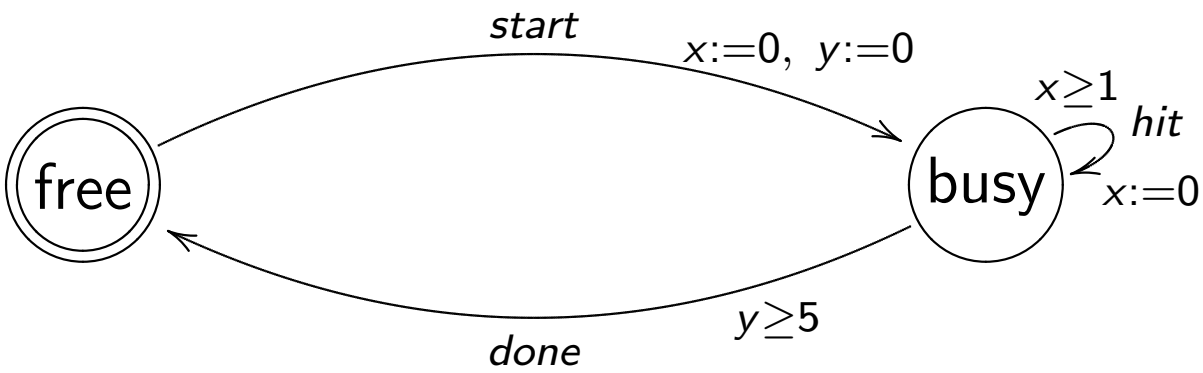
$$(L, \ell_0, E, I)$$

where

- L is a finite set of locations
- $\ell_0 \in L$ is the initial location
- $E \subseteq L \times \mathcal{B}(C) \times N \times 2^C \times L$ is the set of edges
- $I : L \to \mathcal{B}(C)$ assigns invariants to locations.

We usually write $\ell \xrightarrow{g,a,r} \ell'$ whenever $(\ell, g, a, r, \ell') \in E$.

Example: Hammer



Semantics of Timed Automata

Let $A = (L, \ell_0, E, I)$ be a timed automaton.

Timed transition system generated by A

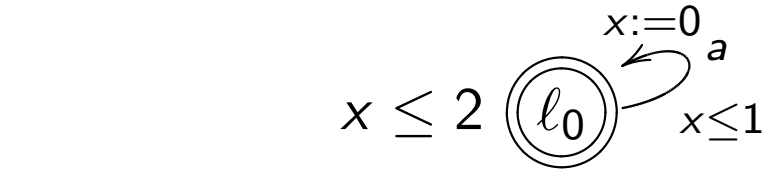
$T(A) = (Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ where

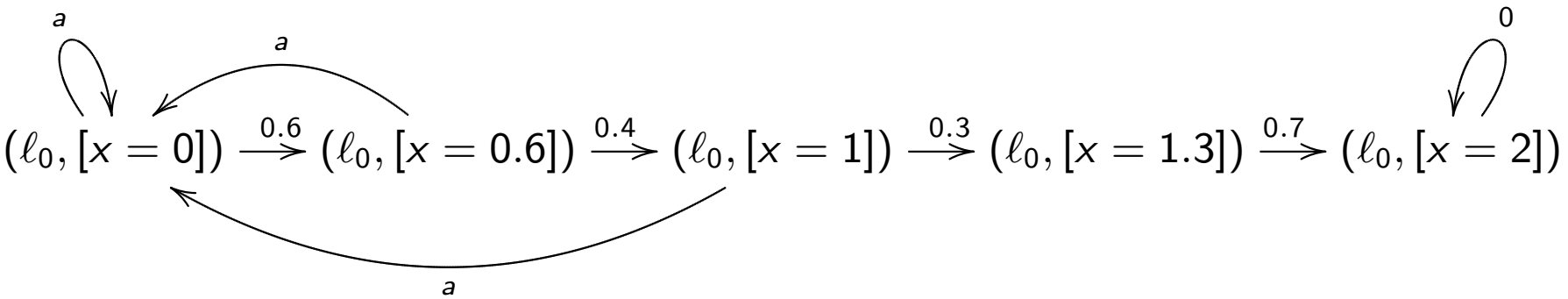
- $Proc$ is the collection of states of the form (ℓ, v) where ℓ is a location and v a valuation such that $v \models I(\ell)$,
- $Act = N \cup \mathbb{R}^{\geq 0}$ and
- $\longrightarrow$ is defined as follows:

$(\ell, v) \xrightarrow{a} (\ell', v')$ if there is $(\ell \xrightarrow{g,a,r} \ell') \in E$ s.t. $v \models g$ and $v' = v[r]$

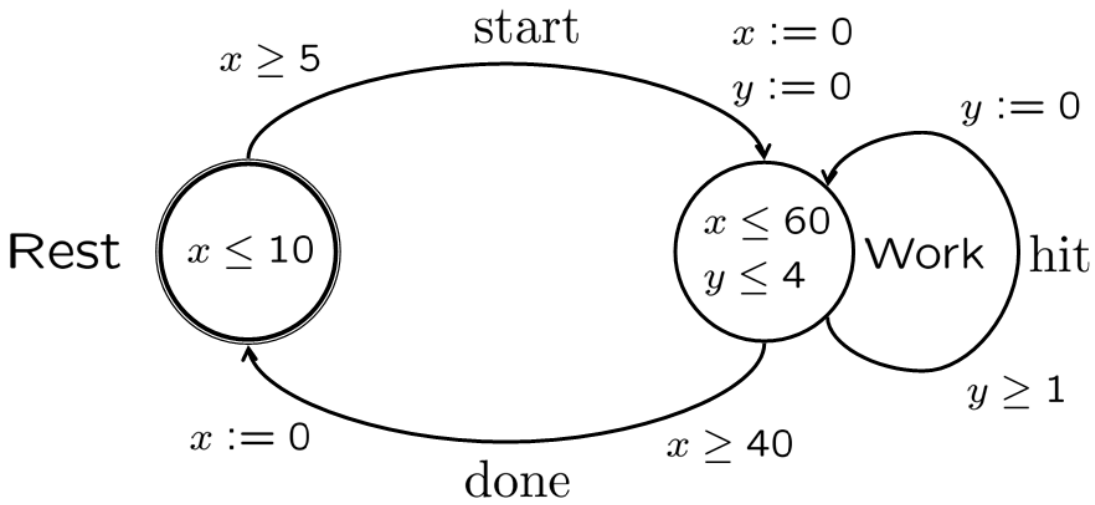
$(\ell, v) \xrightarrow{d} (\ell, v + d)$ for all $d \in \mathbb{R}^{\geq 0}$ s.t. $v \models I(\ell)$ and $v + d \models I(\ell)$

A timed automaton and a fragment of its associated TLTS





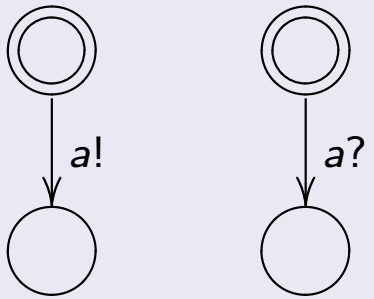
Example: A small jobshop



Can you give a fragment of its associated TLTS?

Networks of Timed Automata

Timed Automata in Parallel



Intuition in CCS

$$(\bar{a}.Nil \mid a.Nil) \setminus \{a\}$$

Let C be a set of clocks and $Chan$ a set of channels.

We let $Act = N \cup \mathbb{R}^{\geq 0}$ where

- $N = \{c! \mid c \in Chan\} \cup \{c? \mid c \in Chan\} \cup \{\tau\}$.

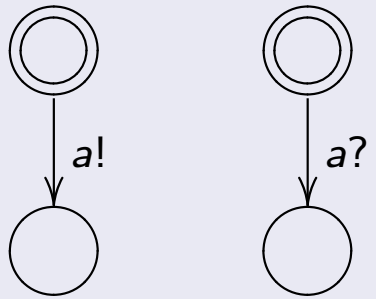
Let $A_i = (L_i, \ell_0^i, E_i, I_i)$ be timed automata for $1 \leq i \leq n$.

Networks of Timed Automata

We call $A = A_1|A_2|\cdots|A_n$ a network of timed automata.

Networks of Timed Automata

Timed Automata in Parallel



Intuition in CCS

$$(\bar{a}.Nil \mid a.Nil) \setminus \{a\}$$

Let C be a set of clocks and $Chan$ a set of channels.

We let $Act = N \cup \mathbb{R}^{\geq 0}$ where

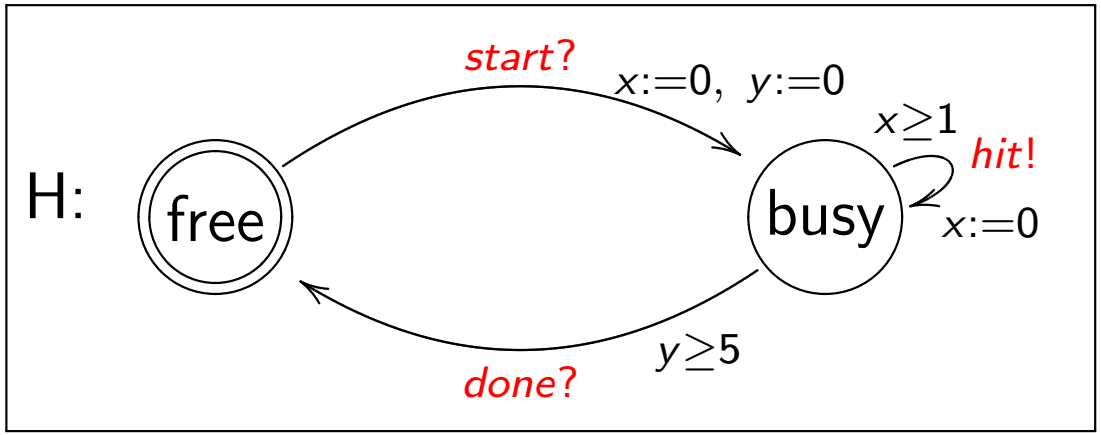
- $N = \{c! \mid c \in Chan\} \cup \{c? \mid c \in Chan\} \cup \{\tau\}$.

Let $A_i = (L_i, \ell_0^i, E_i, I_i)$ be timed automata for $1 \leq i \leq n$.

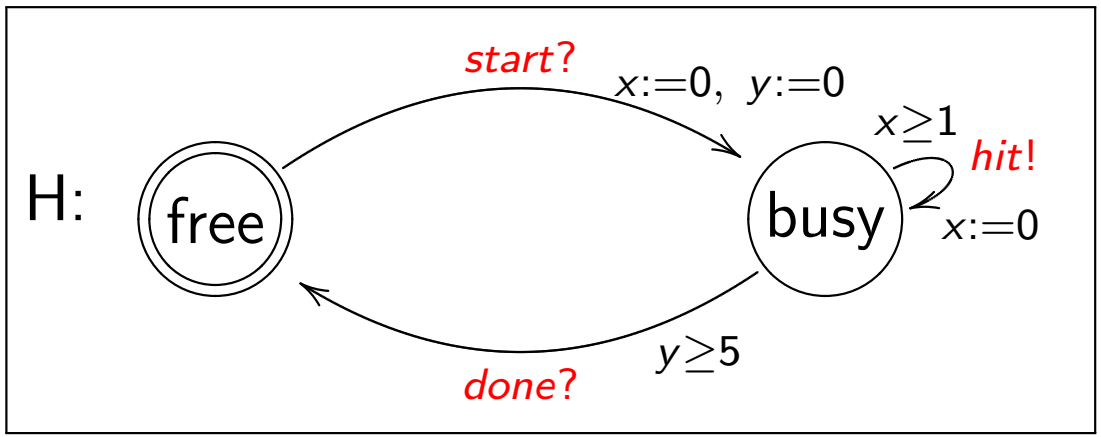
Networks of Timed Automata

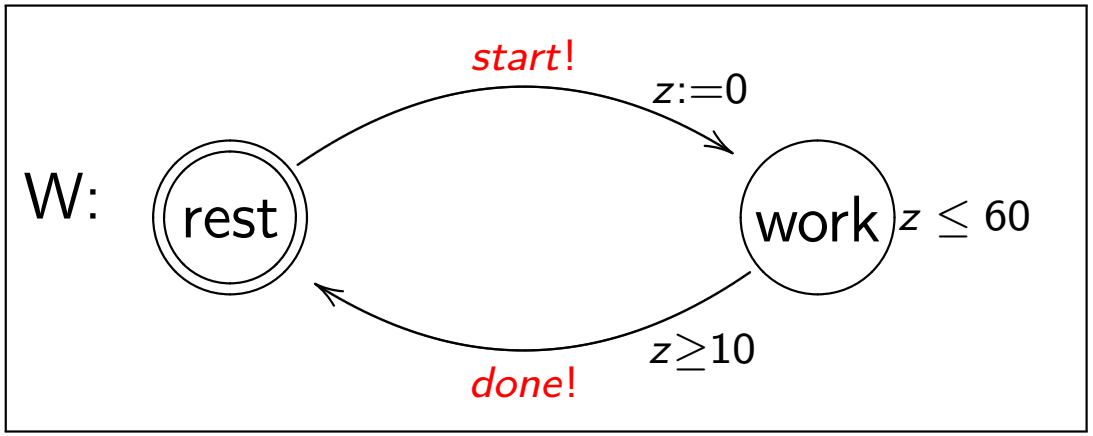
We call $A = A_1|A_2|\cdots|A_n$ a network of timed automata.

Example: Hammer, Worker, Nail

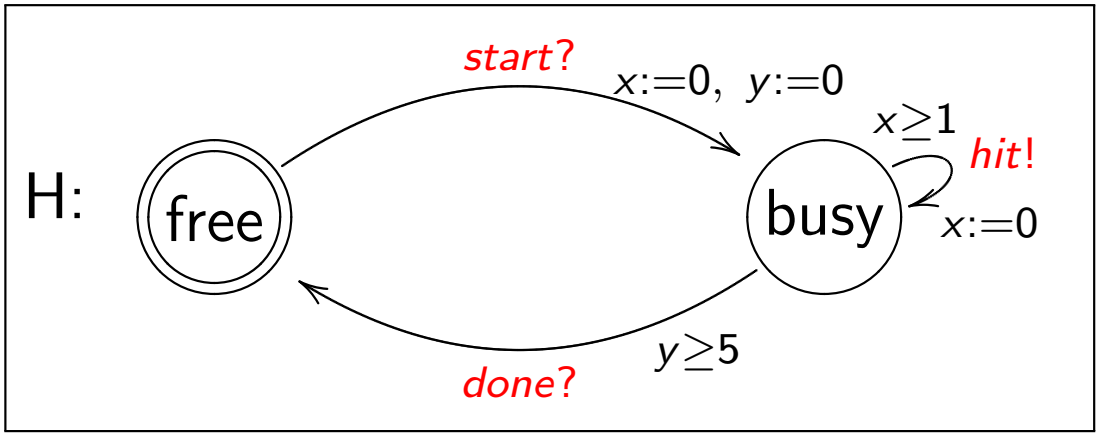


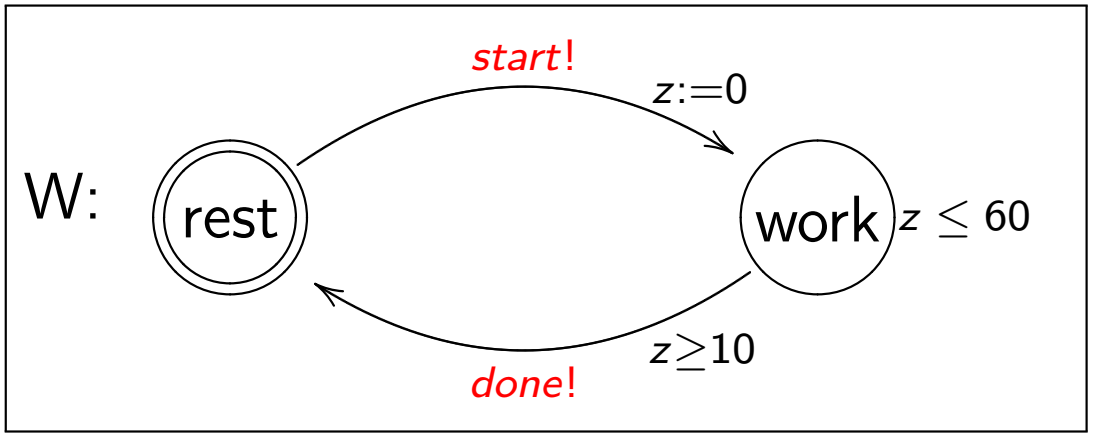
Example: Hammer, Worker, Nail

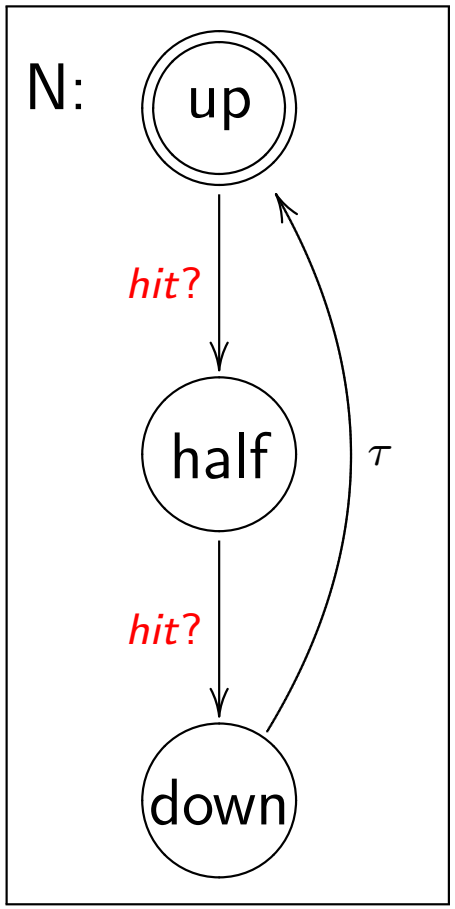




Example: Hammer, Worker, Nail







Timed Transition System Generated by $A = A_1 | \cdots | A_n$

$T(A) = (Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ where

- $Proc$ is the subset of $(L_1 \times L_2 \times \cdots \times L_n) \times (C \to \mathbb{R}^{\geq 0})$ consisting of states of the form $((\ell_1, \ell_2, \ldots, \ell_n), v)$ where ℓ_i is a location in A_i and v a clock valuation such that $v \models \bigwedge_k I_k(\ell_k)$,
- $Act = \{\tau\} \cup \mathbb{R}^{\geq 0}$, and
- $\longrightarrow$ is defined as follows:

$((\ell_1, \ldots, \ell_i, \ldots, \ell_n), v) \xrightarrow{\tau} ((\ell_1, \ldots, \ell'_i, \ldots, \ell_n), v')$ if there is $(\ell_i \xrightarrow{g,\tau,r} \ell'_i) \in E_i$ s.t. $v \models g$ and $v' = v[r]$ and $v' \models I_i(\ell'_i) \wedge \bigwedge_{k \neq i} I_k(\ell_k)$

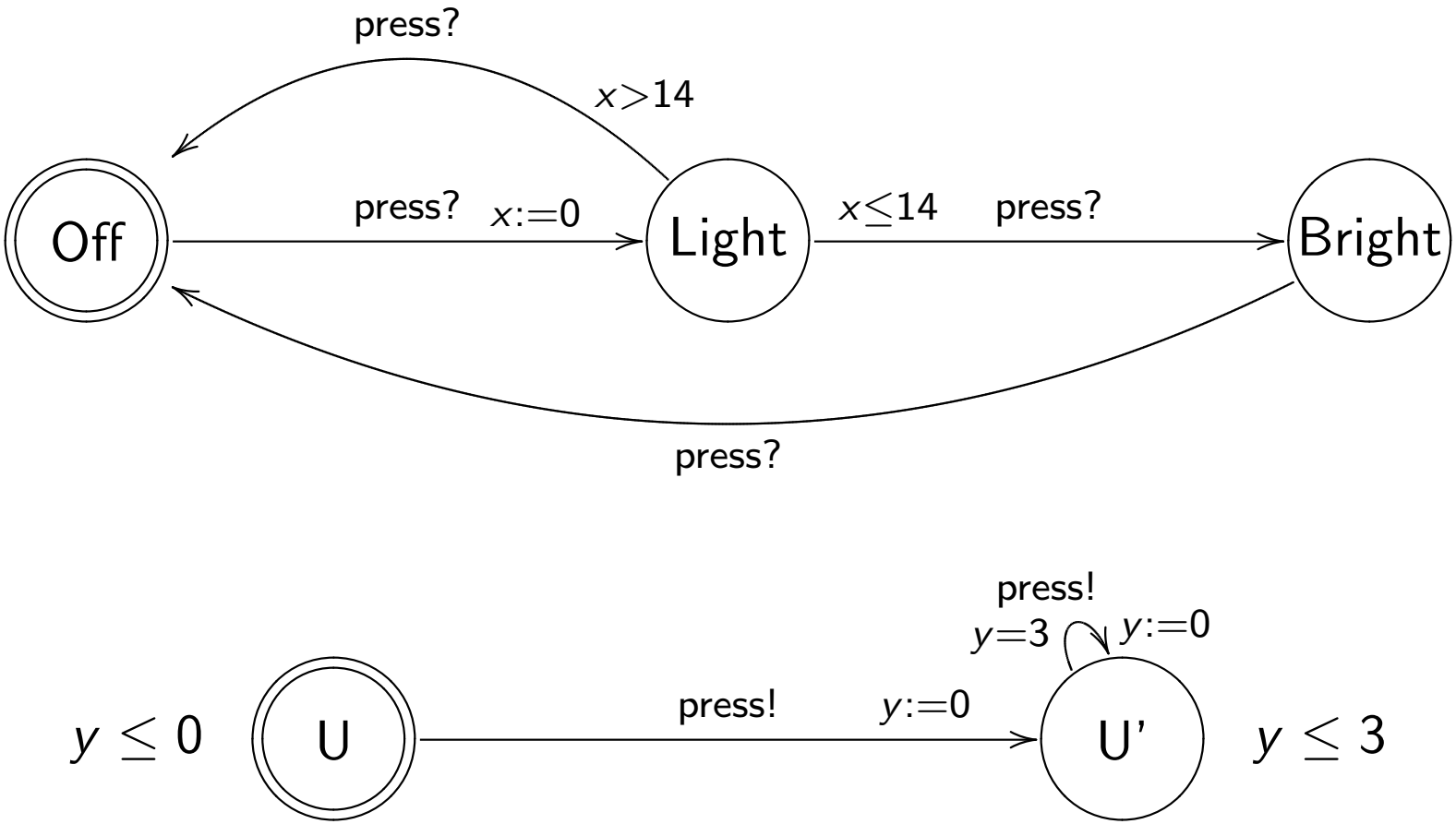
$((\ell_1, \ldots, \ell_n), v) \xrightarrow{d} ((\ell_1, \ldots, \ell_n), v + d)$ for all $d \in \mathbb{R}^{\geq 0}$ s.t. $v \models \bigwedge_k I_k(\ell_k)$ and $v + d \models \bigwedge_k I_k(\ell_k)$

Continuation

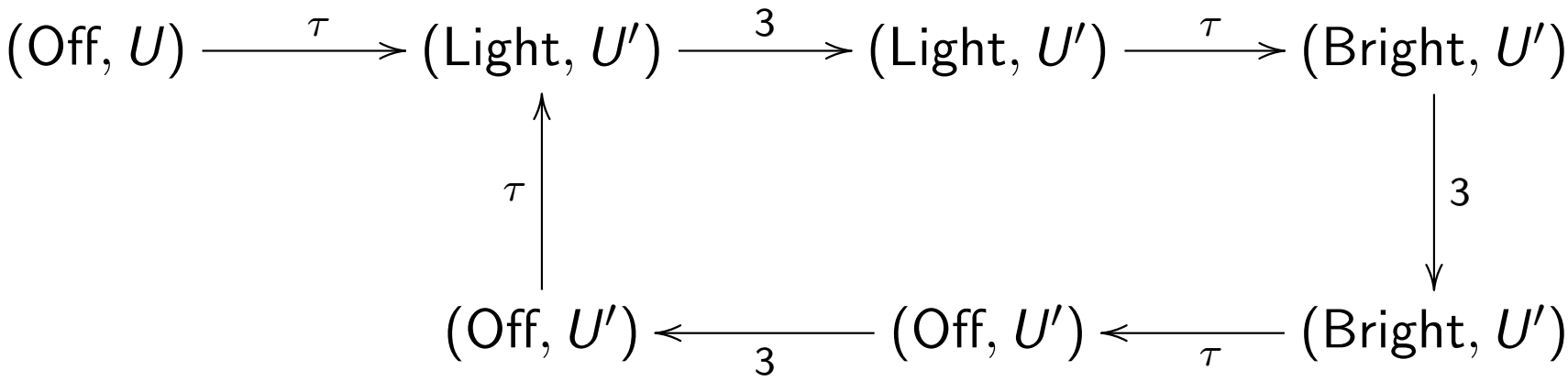
$$((\ell_1, \ldots, \ell_i, \ldots, \ell_j, \ldots, \ell_n), v) \xrightarrow{\tau} ((\ell_1, \ldots, \ell'_i, \ldots, \ell'_j, \ldots, \ell_n), v')$$

if $i \neq j$ and there are $(\ell_i \xrightarrow{g_i, a!, r_i} \ell'_i) \in E_i$ and $(\ell_j \xrightarrow{g_j, a?, r_j} \ell'_j) \in E_j$ s.t. $v \models g_i \wedge g_j$ and $v' = v[r_i \cup r_j]$ and $v' \models I_i(\ell'_i) \wedge I_j(\ell'_j) \wedge \bigwedge_{k \neq i,j} I_k(\ell_k)$

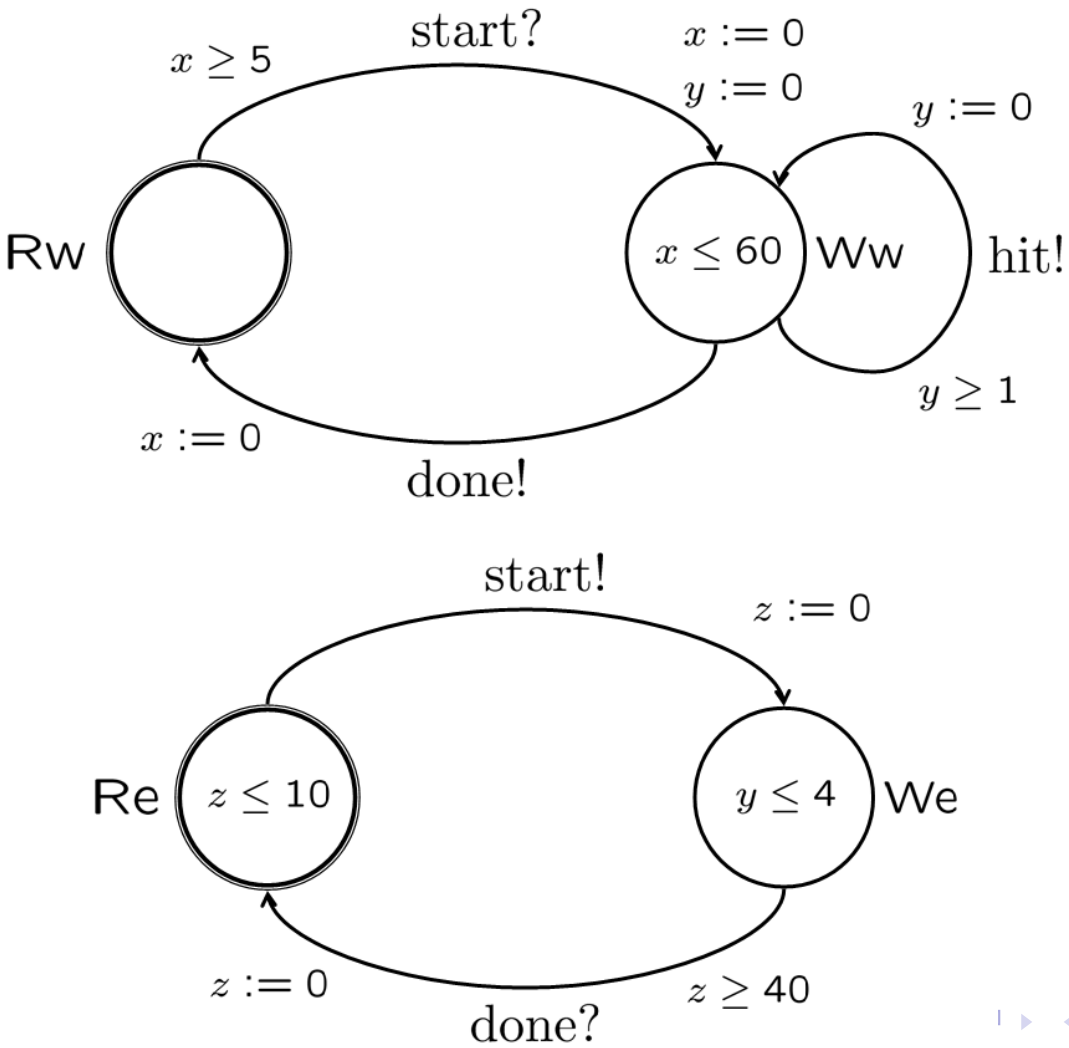
The light switch and a fast user



A fragment of the TLTS for the previous network



The lazy Worker and his demanding Employer



Timed Bisimilarity

Let A_1 and A_2 be timed automata.

Timed Bisimilarity

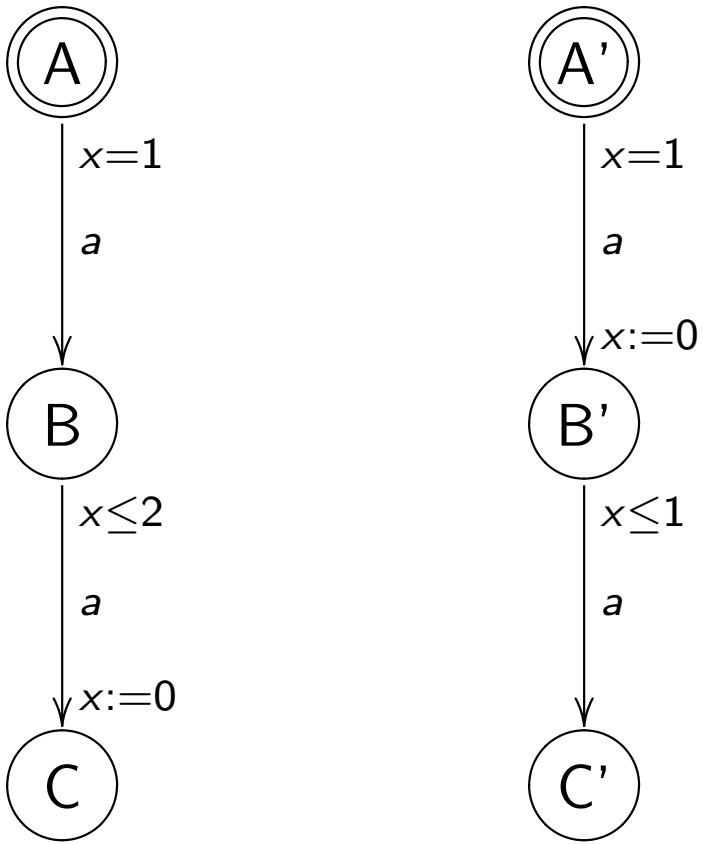
We say that A_1 and A_2 are timed bisimilar iff the transition systems $T(A_1)$ and $T(A_2)$ generated by A_1 and A_2 are strongly bisimilar.

Remark: both

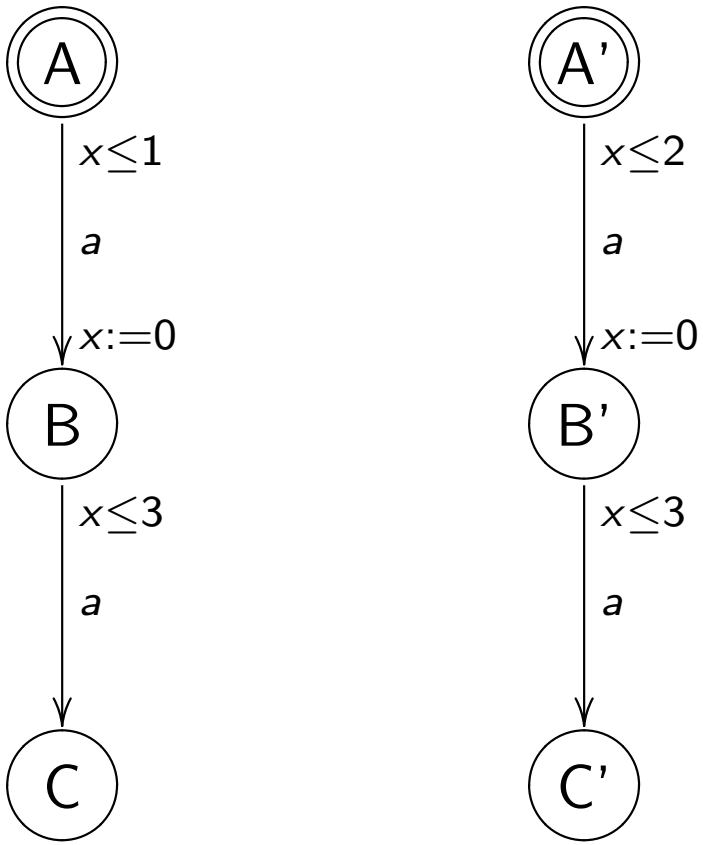
- $\xrightarrow{a}$ for $a \in N$ and
- $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$

are considered as normal (visible) transitions.

Example of Timed Bisimilar Automata



Example of Timed Non-Bisimilar Automata



Untimed Bisimilarity

Let A_1 and A_2 be timed automata. Let ϵ be a new (fresh) action.

Untimed Bisimilarity

We say that A_1 and A_2 are untimed bisimilar iff the transition systems $T(A_1)$ and $T(A_2)$ generated by A_1 and A_2 where every transition of the form $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$ is replaced with $\xrightarrow{\epsilon}$ are strongly bisimilar.

Remark:

- $\xrightarrow{a}$ for $a \in N$ is treated as a visible transition, while
- $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$ are all labelled by a single visible action $\xrightarrow{\epsilon}$.

Corollary

Any two timed bisimilar automata are also untimed bisimilar.

Untimed Bisimilarity

Let A_1 and A_2 be timed automata. Let ϵ be a new (fresh) action.

Untimed Bisimilarity

We say that A_1 and A_2 are untimed bisimilar iff the transition systems $T(A_1)$ and $T(A_2)$ generated by A_1 and A_2 where every transition of the form $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$ is replaced with $\xrightarrow{\epsilon}$ are strongly bisimilar.

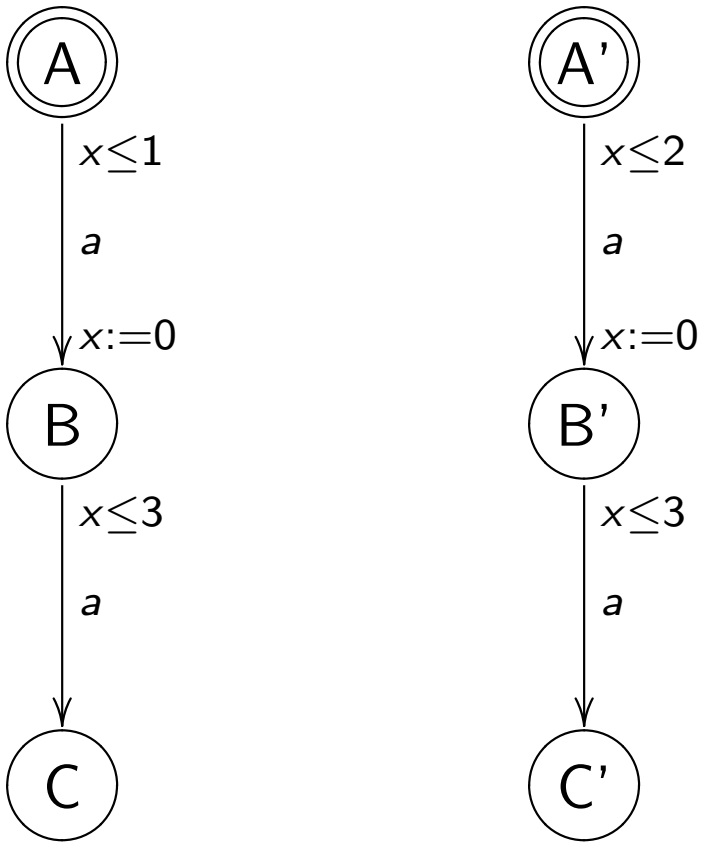
Remark:

- $\xrightarrow{a}$ for $a \in N$ is treated as a visible transition, while
- $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$ are all labelled by a single visible action $\xrightarrow{\epsilon}$.

Corollary

Any two timed bisimilar automata are also untimed bisimilar.

Timed Non-Bisimilar but Untimed Bisimilar Automata



Decidability of Timed and Untimed Bisimilarity

Theorem [Cerans'92]

Timed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).

Theorem [Larsen, Wang'93]

Untimed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).

Decidability of Timed and Untimed Bisimilarity

Theorem [Cerans'92]

Timed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).

Theorem [Larsen, Wang'93]

Untimed bisimilarity for timed automata is decidable in EXPTIME (deterministic exponential time).

Timed Traces

Let $A = (L, \ell_0, E, I)$ be a timed automaton over a set of clocks C and a set of labels N.

Timed Traces

A sequence $(t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$ where $t_i \in \mathbb{R}^{\geq 0}$ and $a_i \in N$ is called a timed trace of A iff there is a transition sequence

$$(\ell_0, v_0) \xrightarrow{d_1} . \xrightarrow{a_1} . \xrightarrow{d_2} . \xrightarrow{a_2} . \xrightarrow{d_3} . \xrightarrow{a_3} \ldots$$

in A such that $v_0(x) = 0$ for all $x \in C$ and

$$t_i = t_{i-1} + d_i \qquad \text{where } t_0 = 0.$$

Intuition: t_i is the absolute time (time-stamp) when a_i happened since the start of the automaton A.

Timed and Untimed Language Equivalence

The set of all timed traces of an automaton A is denoted by $L(A)$ and called the timed language of A.

Theorem [Alur, Courcoubetis, Dill, Henzinger'94]

Timed language equivalence (the problem whether $L(A_1) = L(A_2)$ for given timed automata A_1 and A_2) is undecidable.

We say that $a_1 a_2 a_3 \ldots$ is an untimed trace of A iff there exist $t_1, t_2, t_3, \ldots \in \mathbb{R}^{\geq 0}$ such that $(t_1, a_1)(t_2, a_2)(t_3, a_3) \ldots$ is a timed trace of A.

Theorem [Alur, Dill'94]

Untimed language equivalence for timed automata is decidable.

Timed and Untimed Language Equivalence

The set of all timed traces of an automaton A is denoted by $L(A)$ and called the timed language of A.

Theorem [Alur, Courcoubetis, Dill, Henzinger'94]

Timed language equivalence (the problem whether $L(A_1) = L(A_2)$ for given timed automata A_1 and A_2) is undecidable.

We say that $a_1 a_2 a_3 \ldots$ is an untimed trace of A iff there exist $t_1, t_2, t_3, \ldots \in \mathbb{R}^{\geq 0}$ such that $(t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$ is a timed trace of A.

Theorem [Alur, Dill'94]

Untimed language equivalence for timed automata is decidable.

Logic for Timed Automata in UPPAAL

Let ϕ and ψ be local properties (checkable locally in a given state).
Example: $(\text{H.busy} \wedge \text{W.rest} \wedge \ 20 \leq z \leq 30)$

UPPAAL can check the following formulae (subset of TCTL)

- $\text{A}[]\phi$ — invariantly ϕ
- $\text{E}\langle\rangle\phi$ — possibly ϕ
- $\text{A}\langle\rangle\phi$ — always eventually ϕ
- $\text{E}[]\phi$ — potentially always ϕ
- $\phi \text{ -> } \psi$ — ϕ always leads to ψ $\left(\text{same as } \text{A}[](\phi \implies \text{A}\langle\rangle\psi)\right)$

Legend:

- A and E are so called path quantifiers, and
- [] and $\langle\rangle$ quantify over states of a selected path.

Automatic Verification of Timed Automata

Fact

Even very simple timed automata generate timed transition systems with infinitely (even uncountably) many reachable states.

Question

Is any automatic verification approach (like bisimilarity checking, model checking or reachability analysis) possible at all?

Answer

Yes, using region graph techniques.

Key idea: infinitely many clock valuations can be partitioned into finitely many equivalence classes.

Automatic Verification of Timed Automata

Fact

Even very simple timed automata generate timed transition systems with infinitely (even uncountably) many reachable states.

Question

Is any automatic verification approach (like bisimilarity checking, model checking or reachability analysis) possible at all?

Answer

Yes, using region graph techniques.

Key idea: infinitely many clock valuations can be partitioned into finitely many equivalence classes.

Automatic Verification of Timed Automata

Fact

Even very simple timed automata generate timed transition systems with infinitely (even uncountably) many reachable states.

Question

Is any automatic verification approach (like bisimilarity checking, model checking or reachability analysis) possible at all?

Answer

Yes, using region graph techniques.

Key idea: infinitely many clock valuations can be partitioned into finitely many equivalence classes.

Preliminaries

Let $d \in \mathbb{R}^{\geq 0}$. Then

- let $\lfloor d \rfloor$ be the integer part of d, and
- let $frac(d)$ be the fractional part of d.

Any $d \in \mathbb{R}^{\geq 0}$ can be now written as $d = \lfloor d \rfloor + frac(d)$.

Example: $\lfloor 2.345 \rfloor = 2$ and $frac(2.345) = 0.345$.

Let A be a timed automaton and $x \in C$ be a clock. We define

$$c_x \in \mathbb{N}$$

as the largest constant with which the clock x is ever compared either in the guards or in the invariants present in A.

Preliminaries

Let $d \in \mathbb{R}^{\geq 0}$. Then

- let $\lfloor d \rfloor$ be the integer part of d, and
- let $frac(d)$ be the fractional part of d.

Any $d \in \mathbb{R}^{\geq 0}$ can be now written as $d = \lfloor d \rfloor + frac(d)$.

Example: $\lfloor 2.345 \rfloor = 2$ and $frac(2.345) = 0.345$.

Let A be a timed automaton and $x \in C$ be a clock. We define

$$c_x \in \mathbb{N}$$

as the largest constant with which the clock x is ever compared either in the guards or in the invariants present in A.

Intuition

Let $v, v' : C \to \mathbb{R}^{\geq 0}$ be clock valuations.
Let $\sim$ denote untimed bisimilarity of timed transition systems.

Our Aim

Define an equivalence relation $\equiv$ over clock valuations such that

1. $v \equiv v'$ implies $(\ell, v) \sim (\ell, v')$ for any location ℓ
2. $\equiv$ has only finitely many equivalence classes.

Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations v and v' are equivalent ($v \equiv v'$) iff

Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations v and v' are equivalent ($v \equiv v'$) iff

1. for all $x \in C$ such that $v(x) \leq c_x$ or $v'(x) \leq c_x$ we have

$$\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$$

Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations v and v' are equivalent ($v \equiv v'$) iff

1. for all $x \in C$ such that $v(x) \leq c_x$ or $v'(x) \leq c_x$ we have

$$\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$$

2. for all $x \in C$ such that $v(x) \leq c_x$ we have

$$frac(v(x)) = 0 \quad \text{iff} \quad frac(v'(x)) = 0$$

Clock (Region) Equivalence

Equivalence Relation on Clock Valuations

Clock valuations v and v' are equivalent ($v \equiv v'$) iff

1. for all $x \in C$ such that $v(x) \leq c_x$ or $v'(x) \leq c_x$ we have

$$\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$$

2. for all $x \in C$ such that $v(x) \leq c_x$ we have

$$frac(v(x)) = 0 \quad \text{iff} \quad frac(v'(x)) = 0$$

3. for all $x, y \in C$ such that $v(x) \leq c_x$ and $v(y) \leq c_y$ we have

$$frac(v(x)) \leq frac(v(y)) \quad \text{iff} \quad frac(v'(x)) \leq frac(v'(y))$$

Regions

Let v be a clock valuation. The $\equiv$-equivalence class represented by v is denoted by $[v]$ and defined by $[v] = \{v' \mid v' \equiv v\}$.

Definition of a Region

An $\equiv$-equivalence class $[v]$ represented by some clock valuation v is called a region.

Theorem

For every location ℓ and any two valuations v and v' from the same region ($v \equiv v'$) it holds that

$$(\ell, v) \sim (\ell, v')$$

where $\sim$ stands for untimed bisimilarity.

Regions

Let v be a clock valuation. The $\equiv$-equivalence class represented by v is denoted by $[v]$ and defined by $[v] = \{v' \mid v' \equiv v\}$.

Definition of a Region

An $\equiv$-equivalence class $[v]$ represented by some clock valuation v is called a region.

Theorem

For every location ℓ and any two valuations v and v' from the same region ($v \equiv v'$) it holds that

$$(\ell, v) \sim (\ell, v')$$

where $\sim$ stands for untimed bisimilarity.

Symbolic States and Region Graph

state (ℓ, v) $\rightsquigarrow$ symbolic state $(\ell, [v])$

Note: $v \equiv v'$ implies that $(\ell, [v]) = (\ell, [v'])$.

Region Graph

Region graph of a timed automaton A is an unlabelled (and untimed) transition system where

- states are symbolic states
- $\Longrightarrow$ on symbolic states is defined as follows:
  $(\ell, [v]) \Longrightarrow (\ell', [v'])$ iff $(\ell, v) \xrightarrow{a} (\ell', v')$ for some label a
  $(\ell, [v]) \Longrightarrow (\ell, [v'])$ iff $(\ell, v) \xrightarrow{d} (\ell, v')$ for some $d \in \mathbb{R}^{\geq 0}$

Fact

A region graph of any timed automaton is finite.

Symbolic States and Region Graph

state (ℓ, v) $\rightsquigarrow$ symbolic state $(\ell, [v])$

Note: $v \equiv v'$ implies that $(\ell, [v]) = (\ell, [v'])$.

Region Graph

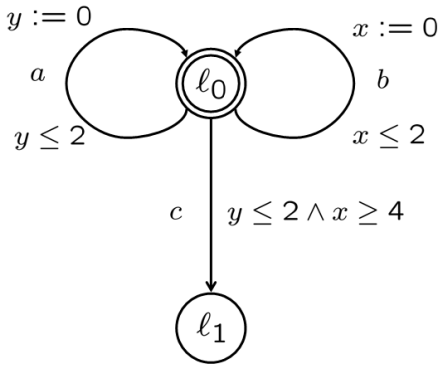
Region graph of a timed automaton A is an unlabelled (and untimed) transition system where

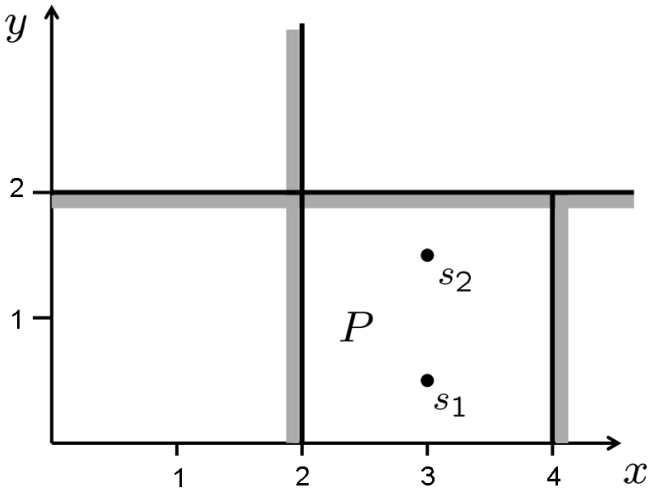
- states are symbolic states
- $\Longrightarrow$ on symbolic states is defined as follows:
  $(\ell, [v]) \Longrightarrow (\ell', [v'])$ iff $(\ell, v) \xrightarrow{a} (\ell', v')$ for some label a
  $(\ell, [v]) \Longrightarrow (\ell, [v'])$ iff $(\ell, v) \xrightarrow{d} (\ell, v')$ for some $d \in \mathbb{R}^{\geq 0}$

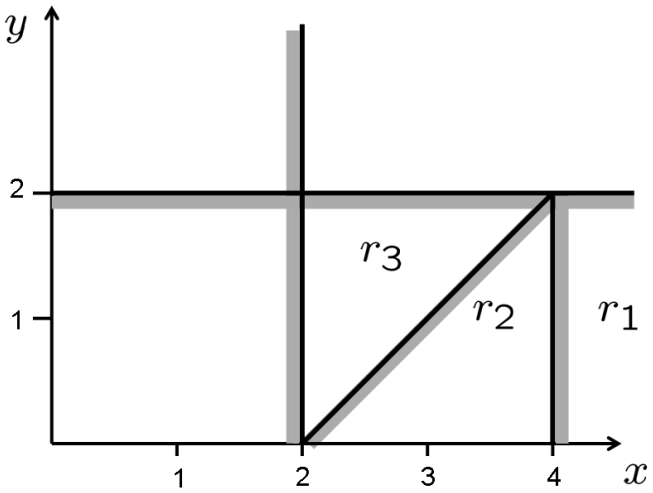
Fact

A region graph of any timed automaton is finite.

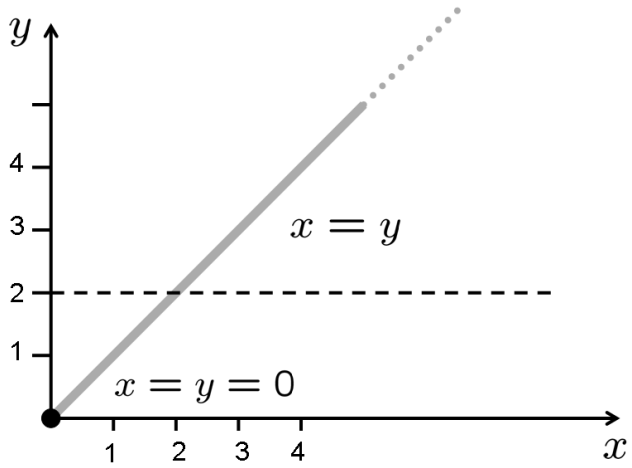
Partitioning of the valuations for a simple timed automaton

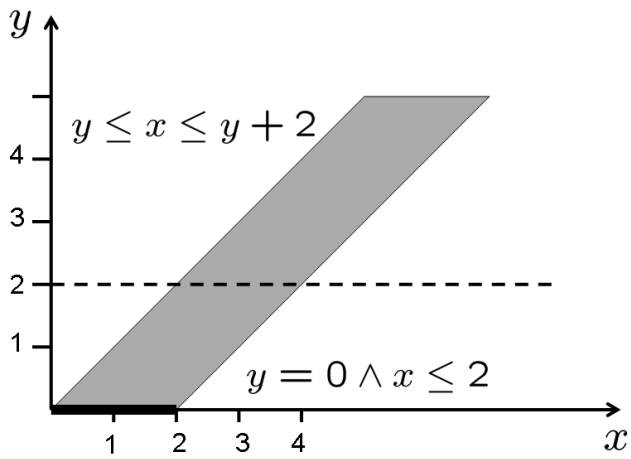


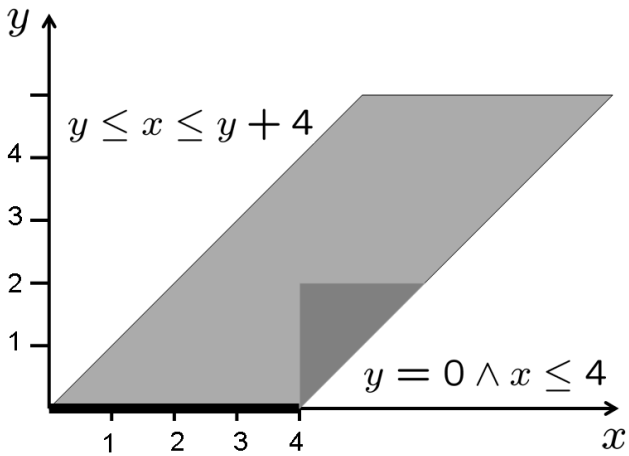




Symbolic exploration of the simple timed automaton







Application of Region Graphs to Reachability

We write $(\ell, v) \longrightarrow (\ell', v')$ whenever

- $(\ell, v) \xrightarrow{a} (\ell', v')$ for some label a, or
- $(\ell, v) \xrightarrow{d} (\ell', v')$ for some $d \in \mathbb{R}^{\geq 0}$.

Reachability Problem for Timed Automata

Instance (input): Automaton $A = (L, \ell_0, E, I)$ and a state (ℓ, v).

Question: Is it true that $(\ell_0, v_0) \longrightarrow^* (\ell, v)$ (where $v_0(x) = 0$ for all $x \in C$)?

Reduction of Timed Automata Reachability to Region Graphs

Reachability for timed automata is decidable because

$(\ell_0, v_0) \longrightarrow^* (\ell, v)$ in a timed automaton if and only if

$(\ell_0, [v_0]) \Longrightarrow^* (\ell, [v])$ in its (finite) region graph.

Application of Region Graphs to Reachability

We write $(\ell, v) \longrightarrow (\ell', v')$ whenever

- $(\ell, v) \xrightarrow{a} (\ell', v')$ for some label a, or
- $(\ell, v) \xrightarrow{d} (\ell', v')$ for some $d \in \mathbb{R}^{\geq 0}$.

Reachability Problem for Timed Automata

Instance (input): Automaton $A = (L, \ell_0, E, I)$ and a state (ℓ, v).

Question: Is it true that $(\ell_0, v_0) \longrightarrow^* (\ell, v)$ (where $v_0(x) = 0$ for all $x \in C$)?

Reduction of Timed Automata Reachability to Region Graphs

Reachability for timed automata is decidable because

$$(\ell_0, v_0) \longrightarrow^* (\ell, v) \text{ in a timed automaton if and only if}$$
$$(\ell_0, [v_0]) \Longrightarrow^* (\ell, [v]) \text{ in its (finite) region graph.}$$

Applicability of Region Graphs

Pros

Region graphs provide a natural abstraction which enables to prove decidability of e.g.

- reachability
- timed and untimed bisimilarity
- untimed language equivalence and language emptiness.

Cons

Region graphs have too large state spaces. State explosion is exponential in

- the number of clocks
- the maximal constants appearing in the guards.

Applicability of Region Graphs

Pros

Region graphs provide a natural abstraction which enables to prove decidability of e.g.

- reachability
- timed and untimed bisimilarity
- untimed language equivalence and language emptiness.

Cons

Region graphs have too large state spaces. State explosion is exponential in

- the number of clocks
- the maximal constants appearing in the guards.

Zones and Zone Graphs

Zones provide a more efficient representation of symbolic state spaces. A number of regions can be described by one zone.

Zone

A zone is described by a clock constraint $g \in \mathcal{B}(C)$.

$$[g] = \{v \mid v \models g\}$$

Region Graphs

symbolic state: $(\ell, [v])$
where v is a clock valuation

Zone Graphs

symbolic state: $(\ell, [g])$
where g is a clock constraint

A zone is usually represented (and stored in the memory) as DBM (Difference Bound Matrix).

Zones and Zone Graphs

Zones provide a more efficient representation of symbolic state spaces. A number of regions can be described by one zone.

Zone

A zone is described by a clock constraint $g \in \mathcal{B}(C)$.

$$[g] = \{v \mid v \models g\}$$

Region Graphs

symbolic state: $(\ell, [v])$
where v is a clock valuation

Zone Graphs

symbolic state: $(\ell, [g])$
where g is a clock constraint

A zone is usually represented (and stored in the memory) as DBM (Difference Bound Matrix).

Zones and Zone Graphs

Zones provide a more efficient representation of symbolic state spaces. A number of regions can be described by one zone.

Zone

A zone is described by a clock constraint $g \in \mathcal{B}(C)$.

$$[g] = \{v \mid v \models g\}$$

Region Graphs

symbolic state: $(\ell, [v])$
where v is a clock valuation

Zone Graphs

symbolic state: $(\ell, [g])$
where g is a clock constraint

A zone is usually represented (and stored in the memory) as DBM (Difference Bound Matrix).