# Tools and Algorithms for Deciding Relations on Timed Automata

B Tech project, supervised by S Arun-Kumar, verification group

Mihir Mehta

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

cs1090197@cse.iitd.ac.in

May 12, 2013

# Outline

Automata without timing and relations on them

Timed automata and relations on them

Algorithms

# Labeled transition systems

**Definition**

*Labelled Transition System*: A labelled transition system (LTS) [1] is an automaton which is described by

- $S$, a set of *states*
- $Act$, a set of *actions*
- $\rightarrow \subseteq S \times Act \times S$, a *transition relation*.
- optionally, $I \subseteq S$ ,a set of initial states. If there is exactly one initial state, then the LTS is said to be *rooted*.

# Relations on LTS I

## Definition

*Strong bisimulation*:A binary relation $R$ on the states of an LTS is a strong bisimulation if and only if, for all $(s_1, s_2) \epsilon R$ and $a \epsilon Act$.
$\forall s_1'(s_1 \xrightarrow{a} s_1' \Rightarrow \exists s_2'.(s_2 \xrightarrow{a} s_2' \wedge (s_1', s_2') \epsilon R)) \wedge$
$\forall s_2'(s_2 \xrightarrow{a} s_2' \Rightarrow \exists s_1'.(s_1 \xrightarrow{a} s_1' \wedge (s_1', s_2') \epsilon R))$

## Definition

It can be shown that the union of all strong bisimulations over the set of states is a strong bisimulation. This binary relation is called *strong bisimilarity*, denoted by $\sim$.

# Outline

Automata without timing and relations on them

Timed automata and relations on them

Algorithms

# Timed Automata

## Definition

*Timed Automaton*: A timed automaton [2] over a finite set of clocks $C$ and a finite set of actions $Act$ is a 4-tuple $(L, l_0, E, I)$.

- $L$ is a finite set of locations.
- $l_0$ is the initial location.
- $E \subseteq L \times B(C) \times Act \times 2^C \times L$ is a finite set of edges.
- $I : L \to B(C)$ assigns invariants to each edge location.
- $B(C)$ is the set of clock constraints over C.

# Relations on timed automata I

## Definition

*Strong time abstracted bisimulation*: A binary relation $R$ is a strong time abstracted bisimulation (STaB) if and only if, for all $(s_1, s_2) \in R$ , $a \in Act$, $d \in R_{\geq 0}$

$$\forall s_1'(s_1 \xrightarrow{a} s_1' \Rightarrow \exists s_2'.(s_2 \xrightarrow{a} s_2' \wedge (s_1', s_2') \in R))\wedge$$
$$\forall s_2'(s_2 \xrightarrow{a} s_2' \Rightarrow \exists s_1'.(s_1 \xrightarrow{a} s_1' \wedge (s_1', s_2') \in R))\wedge$$
$$\forall s_1'(s_1 \xrightarrow{d} s_1' \Rightarrow \exists (s_2', d').(s_2 \xrightarrow{d'} s_2' \wedge (s_1', s_2') \in R))\wedge$$
$$\forall s_2'(s_2 \xrightarrow{d} s_2' \Rightarrow \exists (s_1', d').(s_1 \xrightarrow{d'} s_1' \wedge (s_1', s_2') \in R))$$

It can be shown that the union of all strong time abstracted bisimulations over the set of (location, valuation) pairs is a strong time abstracted bisimulation. This binary relation is called *strong time abstracted bisimilarity*.

## Definition

# Relations on timed automata II

*Time abstracted delay bisimulation*: A binary relation $R$ is a time abstracted delay bisimulation (TadB) if and only if, for all $(s_1, s_2) \epsilon$ $R$ , $a \epsilon Act$, $d \epsilon R_{\geq 0}$

$$\forall s_1'(s_1 \xrightarrow{a} s_1' \Rightarrow \exists(s_2', d).(s_2 \xrightarrow{d}\xrightarrow{a} s_2' \wedge (s_1', s_2') \epsilon R))\wedge$$
$$\forall s_2'(s_2 \xrightarrow{a} s_2' \Rightarrow \exists(s_1', d).(s_1 \xrightarrow{d}\xrightarrow{a} s_1' \wedge (s_1', s_2') \epsilon R))\wedge$$
$$\forall s_1'(s_1 \xrightarrow{d} s_1' \Rightarrow \exists(s_2', d').(s_2 \xrightarrow{d'} s_2' \wedge (s_1', s_2') \epsilon R))\wedge$$
$$\forall s_2'(s_2 \xrightarrow{d} s_2' \Rightarrow \exists(s_1', d').(s_1 \xrightarrow{d'} s_1' \wedge (s_1', s_2') \epsilon R))$$

It can be shown that the union of all time abstracted delay bisimulations over the set of (location, valuation) pairs is a time abstracted delay bisimulation. This binary relation is called *time abstracted delay bisimilarity*.

Definition

# Relations on timed automata III

*Time abstracted observational bisimulation*: A binary relation $R$ is a time abstracted observational bisimulation (TaoB) if and only if, for all $(s_1, s_2) \in R$ , $a \in Act$, $d \in R_{\geq 0}$

$$\forall s_1'(s_1 \xrightarrow{a} s_1' \Rightarrow \exists(s_2', d, d').(s_2 \xrightarrow{d}\xrightarrow{a}\xrightarrow{d'} s_2' \wedge (s_1', s_2') \in R))\wedge$$
$$\forall s_2'(s_2 \xrightarrow{a} s_2' \Rightarrow \exists(s_1', d, d').(s_1 \xrightarrow{d}\xrightarrow{a}\xrightarrow{d'} s_1' \wedge (s_1', s_2') \in R))\wedge$$
$$\forall s_1'(s_1 \xrightarrow{d} s_1' \Rightarrow \exists(s_2', d').(s_2 \xrightarrow{d'} s_2' \wedge (s_1', s_2') \in R))\wedge$$
$$\forall s_2'(s_2 \xrightarrow{d} s_2' \Rightarrow \exists(s_1', d').(s_1 \xrightarrow{d'} s_1' \wedge (s_1', s_2') \in R))$$

It can be shown that the union of all time abstracted observational bisimulations over the set of (location, valuation) pairs is a time abstracted observational bisimulation. This binary relation is called *time abstracted observational bisimilarity*.

# Outline

Automata without timing and relations on them

Timed automata and relations on them

Algorithms

# Creating the zone valuation graph

Initialise the queue $Q$ with a single element $(null, null, l_0)$;
Initialise the graph *zone_graph* with a single node $(l_0, v_0\uparrow)$ with an $\epsilon$ self-loop;
**while** *$Q$ is not empty* **do**
  Dequeue $(l_{parent}, t, l_{child})$ from $Q$;
  **if** $l_{parent} \neq null$ **then**
    **foreach** *zone $Z_{parent}$ of $l_{parent}$* **do**
      Add new zones to the zones of $l_{child}$ so that all zones reachable from $Z_{parent}$ are represented;
      Abstract if necessary;
      Update edges from $Z_{parent}$ to the new zones of $l_{child}$ **if** *new zones are created in $l_{child}$ or $l_{parent}$ is null* **then**
        **foreach** *outgoing transition $t'$ of $l_{child}$* **do**
          Enqueue $(l_{child}, t',$ `t'.target`$)$ in $Q$;
        **end**
      **end**
    **end**
  **end**

# Zone valuation graph example


Figure: Timed automaton. Here, the states are $\{0\}$, the actions are $\{a\}$, and the clocks are $\{X, Y\}$.

# Zone valuation graph example

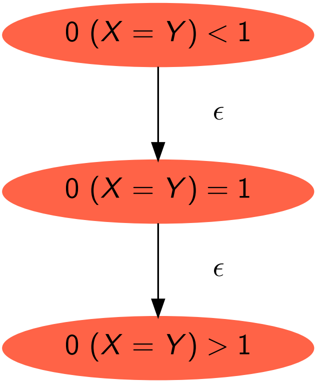



Figure: Zones after one iteration.

# Zone valuation graph example

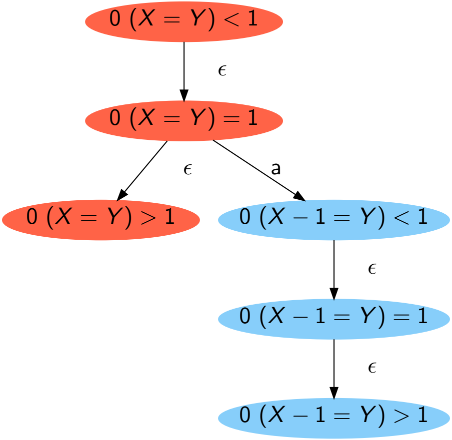



Figure: Zones after two iterations.

# Zone valuation graph example

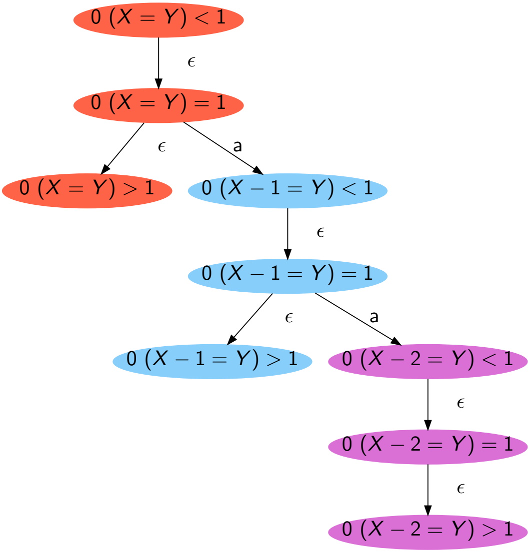



Figure: Zones after three iterations without abstraction.

# Zone valuation graph example

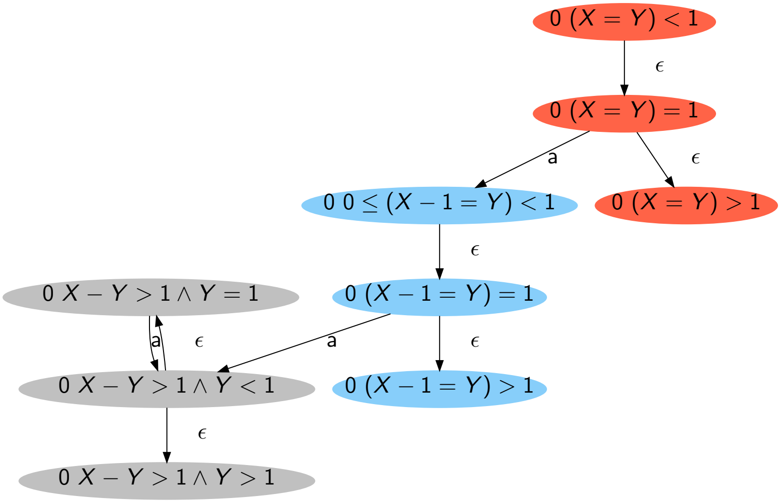



Figure: Zones after three iterations with abstraction.

# Zone valuation graph example

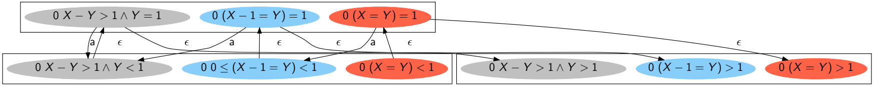



Figure: Zone graph with bisimilarity classes.

# Verifying relations on pairs of timed automata.

- Origin for this algorithm - [3]
- General method to compute $(\rho, \sigma)$-bisimilarities on two LTS, starting from their initial locations.
- Can be adapted for a certain class of timed and time abstracted relations by using zone valuation graphs.
- For every relation $R$ satisfying this property, functions $f_P$ and $f_Q$ must exist such that the proposition $s_P R s_Q$ resolves to one of these:
  - yes
  - no
  - if and only if

$$\forall(s'_P, L'_Q) \in f_P(s_P) : \exists s'_Q \in L'_Q : s'_P R s'_Q \quad \wedge$$
$$\forall(L'_P, s'_Q) \in f_Q(s_Q) : \exists s'_P \in L'_P : s'_P R s'_Q$$

# Verifying relations on pairs of timed automata.

- For STaB, we define $f_P$ and $f_Q$ as

$$
\begin{aligned}
f_P(s_P) = & \{(s'_P, L'_Q) | s_P \xrightarrow{a} s'_P, L_Q = \{s'_Q | s_Q \xrightarrow{a} s'_Q\}\} \\
& \cup \{(s'_P, L'_Q) | s_P \xrightarrow{\epsilon} s'_P, L_Q = \{s'_Q | s_Q \xrightarrow{\epsilon} s'_Q\}\} \\
f_Q(s_Q) = & \{(L'_P, s'_Q) | s_Q \xrightarrow{a} s'_Q, L_P = \{s'_P | s_P \xrightarrow{a} s'_P\}\} \\
& \cup \{(L'_P, s'_Q) | s_Q \xrightarrow{\epsilon} s'_Q, L_P = \{s'_P | s_P \xrightarrow{\epsilon} s'_P\}\}
\end{aligned}
$$

- For TadB, we define $f_P$ and $f_Q$ as

$$
\begin{aligned}
f_P(s_P) = & \{(s'_P, L'_Q) | s_P \xrightarrow{a} s'_P, L_Q = \{s'_Q | s_Q \xrightarrow{\epsilon}\xrightarrow{a} s'_Q\}\} \\
& \cup \{(s'_P, L'_Q) | s_P \xrightarrow{\epsilon} s'_P, L_Q = \{s'_Q | s_Q \xrightarrow{\epsilon} s'_Q\}\} \\
f_Q(s_Q) = & \{(L'_P, s'_Q) | s_Q \xrightarrow{a} s'_Q, L_P = \{s'_P | s_P \xrightarrow{\epsilon}\xrightarrow{a} s'_P\}\} \\
& \cup \{(L'_P, s'_Q) | s_Q \xrightarrow{\epsilon} s'_Q, L_P = \{s'_P | s_P \xrightarrow{\epsilon} s'_P\}\}
\end{aligned}
$$

- For TaoB, we define $f_P$ and $f_Q$ as

$$
\begin{aligned}
f_P(s_P) = & \{(s'_P, L'_Q) | s_P \xrightarrow{a} s'_P, L_Q = \{s'_Q | s_Q \xrightarrow{\epsilon}\xrightarrow{a}\xrightarrow{\epsilon} s'_Q\}\} \\
& \cup \{(s'_P, L'_Q) | s_P \xrightarrow{\epsilon} s'_P, L_Q = \{s'_Q | s_Q \xrightarrow{\epsilon} s'_Q\}\} \\
f_Q(s_Q) = & \{(L'_P, s'_Q) | s_Q \xrightarrow{a} s'_Q, L_P = \{s'_P | s_P \xrightarrow{\epsilon}\xrightarrow{a}\xrightarrow{\epsilon} s'_P\}\} \\
& \cup \{(L'_P, s'_Q) | s_Q \xrightarrow{\epsilon} s'_Q, L_P = \{s'_P | s_P \xrightarrow{\epsilon} s'_P\}\}
\end{aligned}
$$

# Verifying relations on pairs of timed automata.

```
begin
    if lookup(yes_table, s_P, s_Q) then
        return true;
    else
        if lookup(yes_table, s_P, s_Q) then
            return false;
        else
            insert(yes_table, s_P, s_Q);
            Set v_P;
            foreach (s'_P, L'_Q) in f_P(s_P) do
                Reset v_P;
                foreach s'_Q in L'_Q do
                    if CheckStatesRelation(P, Q, s'_P, s'_Q, yes_table, no_table) then
                        Set v_P;
                    end
                end
            end
            Set v_Q;
            foreach (s'_Q, L'_P) in f_Q(s_Q) do
                Reset v_Q;
                foreach s'_P in L'_P do
                    if CheckStatesRelation(P, Q, s'_P, s'_Q, yes_table, no_table) then
                        Set v_Q;
                    end
                end
            end
            if v_P ∧ v_Q then
                return true;
            else
                remove(yes_table, s_P, s_Q);
                insert(yes_table, s_P, s_Q);
                return false;
            end
        end
    end
end
```

**Procedure** CheckStatesRelation($P$, $Q$, $s_P$, $s_Q$, *yes_table*, *no_table*)

## Verifying relations on pairs of timed automata.

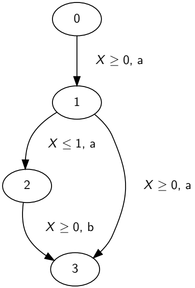
**begin**

Create zone valuation graphs $G_P$, $G_Q$ of $T_P$, $T_Q$;
Find the zone $s_P$ in $G_P$ which contains the initial state of $T_P$;
Find the zone $s_Q$ in $G_Q$ which contains the initial state of $T_Q$;
Initialise *yes_table* and *no_table* to empty tables;
**return** `CheckStatesRelation`($G_P$, $G_Q$, $s_P$, $s_Q$, *yes_table*, *no_table*);

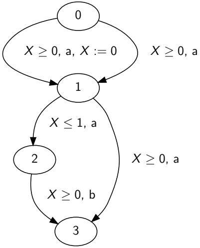
**end**

**Procedure** CheckAutomataRelation($T_P$, $T_Q$)

# Verifying relations: example



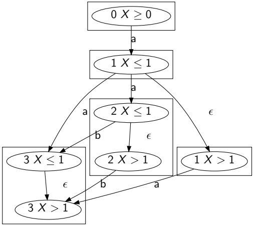

(a) First



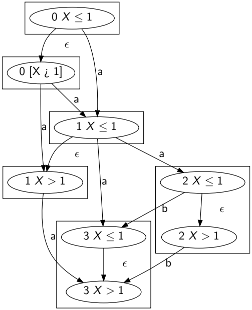

(b) Second

Figure: Timed automata.

# Verifying relations: example




(a) First




(b) Second

Figure: Zone valuation graphs.

# References I


Keller, R.M.:
Formal verification of parallel programs.
Commun. ACM **19**(7) (July 1976) 371–384

Alur, R., Dill, D.L.:
A theory of timed automata.
Theoretical Computer Science **126** (1994) 183–235

Arun-Kumar, S.:
On bisimilarities induced by relations on actions.
In: Software Engineering and Formal Methods, 2006. SEFM 2006. Fourth IEEE International Conference on, IEEE (2006) 41–49