

# Tools and Algorithms for Deciding Timed Relations

B Tech Project, 2012-2013

Mihir Mehta

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

`cs1090197@cse.iitd.ac.in`

December 2012

# Overview

- ▶ Bisimilarity and related notions
- ▶ Kanellakis and Smolka's algorithm
- ▶ Fernandez' algorithm
- ▶ Paige and Tarjan's algorithm
- ▶ Timed Automata
- ▶ Equivalences on Timed Automata
- ▶ Code written so far

# Bisimilarity and related notions

- ▶ Labeled Transition System: This is a triple  $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$  where
  - ▶  $Proc$  is a set of states (also called processes or configurations.)
  - ▶  $Act$  is a set of actions (also called labels.)
  - ▶  $\xrightarrow{a} \subseteq Proc \times Proc$  is a transition relation.
- ▶ CCS expression: Defined by the following grammar:
  - ▶  $P ::= K$
  - ▶  $P ::= \alpha.P$
  - ▶  $P ::=_{i \in I} P_i$
  - ▶  $P ::= P|Q$
  - ▶  $P ::= P[f]$
  - ▶  $P ::= P\mathbf{L}$

# Bisimilarity and related notions

- ▶ Equivalence for CCS processes.
- ▶ Trace equivalence:  $Traces(P) = Traces(Q)$
- ▶ Unsatisfactory (differences in deadlock behaviour.)
- ▶ Strong bisimulation: A binary relation  $R$  is a *strong bisimulation* if and only if, for all  $(s_1, s_2) \in R$  and  $a \in Act$ .  
$$\forall s'_1 (s_1 \xrightarrow{a} s'_1 \Rightarrow \exists s'_2. (s_2 \xrightarrow{a} s'_2 \wedge (s'_1, s'_2) \in R)) \wedge$$
$$\forall s'_2 (s_2 \xrightarrow{a} s'_2 \Rightarrow \exists s'_1. (s_1 \xrightarrow{a} s'_1 \wedge (s'_1, s'_2) \in R))$$
- ▶ 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  $\sim$ .

# Bisimilarity and related notions

- ▶ Better notion of equivalence than trace equivalence: picks up differences in the deadlock behaviour of processes under study.
- ▶ Failing: does not account for the invisible nature of  $\tau$  transitions in CCS processes.
- ▶ Weak bisimulation: A binary relation  $R$  is a *weak bisimulation* if and only if, for all  $(s_1, s_2) \in R$  and  $a \in \text{Act}$ .  
$$\forall s'_1 (s_1 \xrightarrow{a} s'_1 \Rightarrow \exists s'_2. (s_2 \xRightarrow{a} s'_2 \wedge (s'_1, s'_2) \in R)) \wedge$$
$$\forall s'_2 (s_2 \xrightarrow{a} s'_2 \Rightarrow \exists s'_1. (s_1 \xRightarrow{a} s'_1 \wedge (s'_1, s'_2) \in R))$$
- ▶ It can be shown that the union of all weak bisimulations over the set of states is a weak bisimulation. This binary relation is called *weak bisimilarity*, denoted by  $\approx$ .
- ▶ Better suited to CCS processes, as it ignores  $\tau$  transitions, thus disregarding hidden behaviour within a process.

# Kanellakis and Smolka's algorithm

- ▶ This is a naive algorithm for determining the bisimilarity relation for the set of processes in a labelled transition system.
- ▶ This relies on the notion of a splitter.
- ▶ Let  $\pi = \{B_0, \dots, B_k\}$ ,  $k \geq 0$  be a partition of the set of states  $Pr$  in a labeled transition system.
- ▶ A splitter for a block  $B_i \in \pi$  is a block  $B_j \in \pi$  such that for some action  $a \in Act$ , some states in  $B_i$  have  $a$ -labelled transitions whose targets lie in  $B_j$  while other states in  $B_i$  do not.
- ▶ This suggests a refinement of  $\pi$ : replace block  $B_i$  with
$$B_i^1 = B_i \cap T_a^{-1}[B_j]$$
$$B_i^2 = B_i - B_i^1$$
- ▶ Refinements of this kind constitute the steps of this algorithm.

# Kanellakis and Smolka's algorithm

Initialise  $\pi$  to  $Pr$ ;

**while** *there exist splitters among the elements of  $\pi$*  **do**

    Pick a splitter  $B$ ;

**foreach**  $B_j \in \pi$  **do**

**foreach**  $a \in Act$  **do**

            Split  $B_j$  with respect to  $B$  for action  $a$ ;

**end**

**end**

**end**

Return  $\pi$ ;

- ▶ The time complexity of this algorithm is  $O(mn)$ , since there can be at most  $n$  iterations, and all  $m$  edges are scanned in each iteration.

# Fernandez' algorithm

- ▶ More efficient algorithm ( $O(m \log n)$ ).
- ▶ Relies on Paige and Tarjan's technique of three-way splitting.
- ▶ Splitters can now be 'simple' or 'compound'.
- ▶ Stability: A partition  $\pi$  is said to be stable with respect to a compound block  $S$  if  $S$  is not a splitter for any block in  $\pi$  for any action.
- ▶ For a compound block  $S$ , having a constituent simple block  $B$  satisfying  $n(B) \leq 0.5 * n(S)$ , and with respect to which  $\pi$  is stable, we can split a block  $B_i$  on an action  $a$  as follows:

$$B_i^1 = (B_i \cap T_a^{-1}[B]) - T_a^{-1}[S - B]$$

$$B_i^2 = (B_i \cap T_a^{-1}[S - B]) - T_a^{-1}[B]$$

$$B_i^3 = B_i \cap T_a^{-1}[B] \cap T_a^{-1}[S - B]$$



# Fernandez' algorithm

Initialise  $\pi = \{Pr\}$ ;

Initialise  $W = \{Pr\}$  **while**  $W$  is not empty **do**

    Choose a splitter  $B$  from  $W$ , removing it;

**if**  $B$  is a simple splitter **then**

        Perform a two-way split on each action with respect to  $B$   
        and update  $W$ ;

**else**

        Perform a three-way split on each action with respect to  $B$   
        and update  $W$ ;

**end**

**end**

# Paige and Tarjan's Algorithm

- ▶ Technique of three way splitting came from here.
- ▶ Special case of Fernandez' algorithm when there's only one kind of action.

# Timed Automata

- ▶ Formally, a timed automaton 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.
- ▶  $I : L \rightarrow B(C)$  assigns invariants to each edge location.
- ▶  $B(C)$  is the set of clock constraints over  $C$ . An element of  $B(C)$  can be an equality, a slack inequality, a strict inequality, or an AND combination of such constraints.

# Equivalences on Timed Automata

- ▶ *Timed bisimulation*: A binary relation  $R$  is a timed bisimulation if and only if, for all  $(s_1, s_2) \in R$ ,  $a \in \text{Act}$ ,  $d \in \mathbb{R}_{\geq 0}$   
 $\forall s'_1 (s_1 \xrightarrow{a} s'_1 \Rightarrow \exists s'_2. (s_2 \xrightarrow{a} s'_2 \wedge (s'_1, s'_2) \in R)) \wedge$   
 $\forall s'_2 (s_2 \xrightarrow{a} s'_2 \Rightarrow \exists s'_1. (s_1 \xrightarrow{a} s'_1 \wedge (s'_1, s'_2) \in R)) \wedge$   
 $\forall s'_1 (s_1 \xrightarrow{d} s'_1 \Rightarrow \exists s'_2. (s_2 \xrightarrow{d} s'_2 \wedge (s'_1, s'_2) \in R)) \wedge$   
 $\forall s'_2 (s_2 \xrightarrow{d} s'_2 \Rightarrow \exists s'_1. (s_1 \xrightarrow{d} s'_1 \wedge (s'_1, s'_2) \in R))$
- ▶ It can be shown that the union of all timed bisimulations over the set of (location, valuation) pairs is a timed bisimulation. This binary relation is called *timed bisimilarity*, denoted by  $\sim$ .

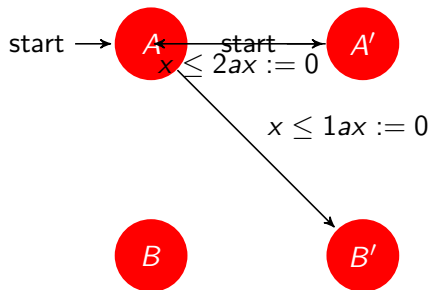
# Equivalences on Timed Automata

- ▶ *Time abstracted bisimulation*: A binary relation  $R$  is a time abstracted bisimulation if and only if, for all  $(s_1, s_2) \in R$ ,  $a \in \text{Act}$ ,  $d \in \mathbb{R}_{\geq 0}$   
 $\forall s'_1 (s_1 \xrightarrow{a} s'_1 \Rightarrow \exists s'_2. (s_2 \xrightarrow{a} s'_2 \wedge (s'_1, s'_2) \in R)) \wedge$   
 $\forall s'_2 (s_2 \xrightarrow{a} s'_2 \Rightarrow \exists s'_1. (s_1 \xrightarrow{a} s'_1 \wedge (s'_1, s'_2) \in R)) \wedge$   
 $\forall s'_1 (s_1 \xrightarrow{d} s'_1 \Rightarrow \exists (s'_2, d'). (s_2 \xrightarrow{d'} s'_2 \wedge (s'_1, s'_2) \in R)) \wedge$   
 $\forall s'_2 (s_2 \xrightarrow{d} s'_2 \Rightarrow \exists (s'_1, d'). (s_1 \xrightarrow{d'} s'_1 \wedge (s'_1, s'_2) \in R))$
- ▶ It can be shown that the union of all time abstracted bisimulations over the set of (location, valuation) pairs is a time abstracted bisimulation. This binary relation is called *time abstracted bisimilarity*, denoted by  $\sim_u$ .

# Equivalences on Timed Automata

- ▶ *Equivalence of valuations:* Two valuations  $v$  and  $v'$  of a timed automaton are said to be equivalent ( $v \equiv v'$ ) if and only if:
  - ▶ For each  $x \in C$ , either both  $v(x)$  and  $v'(x)$  are greater than  $c_x$  or  $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$
  - ▶ For each  $x \in C$  such that  $v(x) \leq c_x$ ,  $\text{frac}(v(x)) = 0$  if and only if  $\text{frac}(v'(x)) = 0$ .
  - ▶ For all  $x, y \in C$  such that  $v(x) \leq c_x$  and  $v(y) \leq c_y$ , we have  $\text{frac}(v(x)) \leq \text{frac}(v(y))$  if and only if  $\text{frac}(v'(x)) \leq \text{frac}(v'(y))$ .
- ▶ Under this notion of equivalence, an equivalence class is known as a *region*. The equivalence class containing a valuation  $v$  is denoted by  $[v]_{\equiv}$ .

# Equivalences on Timed Automata



# Equivalences on Timed Automata

- ▶ *Region graph*: the region graph of a timed automaton  $A$  with clock set  $C$  and action set  $Act$  is an LTS

$$T_r(A) = (Proc, Act \cup \{\varepsilon\}, \{\xrightarrow{a} \mid a \in Act \cup \{\varepsilon\}\})$$

where  $Proc = \{(l, [v]_{\equiv}) \mid l \in L, v : C \rightarrow R_{\geq 0}\}$  (these states are called symbolic states)

The transitions are defined as follows:

- ▶ For each  $a \in A$ ,  $(l, [v]_{\equiv}) \xrightarrow{a} (l', [v']_{\equiv})$  iff  $(l, v) \xrightarrow{a} (l', v')$
- ▶  $(l, [v]_{\equiv}) \xrightarrow{d} (l, [v']_{\equiv})$  if for some  $d \in R_{\geq 0}$ ,  $(l, v) \xrightarrow{d} (l, v')$



# Equivalences on Timed Automata

- ▶ *Extended Clock Constraints*: These are described by the grammar

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

- ▶ The set of extended clock constraints is denoted by  $B^+(C)$
- ▶ *Zones*: A zone is a set of valuations represented by an extended clock constraint.

$$Z = \{v \mid v \in g_Z\}$$

- ▶ *Symbolic state*: For a timed automaton  $A$ , a symbolic state is an ordered pair of a location and a zone, that is,  $(l, Z)$ .

# Equivalences on Timed Automata

- ▶ *Reachability graph*: The reachability graph of a timed automaton is a directed graph in which the nodes are symbolic states and the edges are given by the *reachability relation*:
  - ▶  $(l, Z')$  is reachable from  $(l, Z)$  if there exists a valuation belonging  $Z'$  which can be obtained from  $Z$  by applying a time delay which does not violate the invariant at  $l$ .
  - ▶  $(l', Z')$  is reachable from  $(l, Z)$  if  $l$  has an action transition to  $l'$  and there exists a valuation in  $Z'$  which violates neither the guard of the transition nor the invariant at  $l'$  after performing the clock resets required by the transition.

## Code written so far

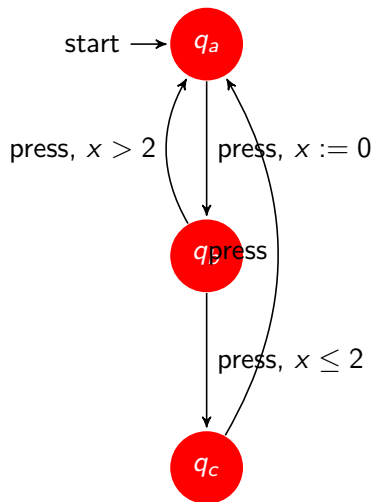
- ▶ *Fernandez' algorithm*: We expect to need this algorithm while building the software, so we built a module implementing it in Ocaml (primary language for this project.)
- ▶ *Grammar*: We implemented a lexer and parser in Ocaml to build a representation of a timed automaton from a specification in a text file.

## Timed automaton example

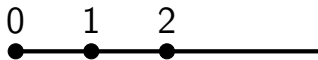
```
states 3 trans 6 clocks 1 X
state: 0 invar: TRUE trans: TRUE => RESET X ; goto 1
state: 1 invar: TRUE trans: X <= 2 => RESET ; goto 0 X
> 2 => RESET ; goto 2
state: 2 invar: TRUE trans: TRUE => RESET ; goto 0
```

*Example taken from Reactive Systems*

# Timed automaton example



# Timed automaton example



Regions in this graph:

- ▶  $[x = 0]_{\equiv}, [x = 1]_{\equiv}, [x = 2]_{\equiv}$
- ▶  $[0 < x < 1]_{\equiv}, [1 < x < 2]_{\equiv}$
- ▶  $[2 < x]_{\equiv}$

# References

- ▶ Reactive Systems: Modeling, Specification and Verification -  
Luca Aceto, Anna Ingolfssdottir, Kim Larsen, Jiri Srba.