Reducing Clocks in Timed Automata while Preserving Bisimulation

Shibashis Guha Chinmay Narayan S. Arun-Kumar

Indian Institute of Technology Delhi

July 8, 2014

The problem we solve

Given a timed automaton (TA), is it possible to
minimize the number of clocks of a timed automaton
while preserving timed bisimulation?

Mentioned and left open in (LLW1 '95)

The problem we solve

Given a timed automaton (TA), is it possible to
minimize the number of clocks of a timed automaton
while preserving timed bisimulation?

Mentioned and left open in (LLW1 ’95)

Contribution

We give a procedure to construct a TA with the minimum possible number of clocks while preserving timed bisimulation.

Importance

Verification and model checking of timed automata uses

region graph or zone graph,

that has a size exponential in the number of the clocks.

Importance

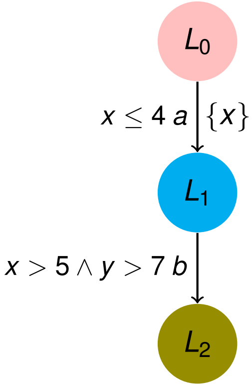
Verification and model checking of timed automata uses

region graph or zone graph,

that has a size exponential in the number of the clocks.

- smaller number of clocks implies smaller region graph or zone graph and thus easier verification

A Timed Automaton



$\bowtie \in \{<, \leq, =, >, \geq\}$

Semantics: Timed Labeled Transition System (TLTS)

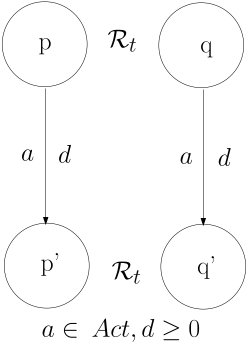
- Infinite transition graph structure
- Nodes are timed automaton states; tuple (ℓ, v)
- Two types of transitions

Discrete transitions : $a \in Act$: $(\ell, v) \xrightarrow{a} (\ell', v')$ if there is an edge $(\ell \xrightarrow{g,a,r} \ell') \in E$ and $v \models g, v' = v[r]$

Delay transitions : $d \in \mathbb{R}_{\geq 0}$: $(\ell, v) \xrightarrow{d} (\ell, v + d)$.

Timed Bisimulation

p and q are two timed states.



Related Work: Clock Reduction

Preserve timed language

- No algorithm can decide the minimality of the number of clocks and

  for the non-minimal case find a timed language equivalent automaton with fewer clocks. (Tripakis '04)

Related Work: Clock Reduction

Preserve timed language

- No algorithm can decide the minimality of the number of clocks and

  for the non-minimal case find a timed language equivalent automaton with fewer clocks. (Tripakis '04)

- Existence of a language equivalent timed automaton with smaller number of clocks is also undecidable (Finkel '06)

Related Work: Clock Reduction

Preserve timed bisimulation

- An algorithm to reduce the number of clocks is provided in (DY ’96)

  It works on a syntactical structure of the timed automaton and does not provide the minimum number of clocks

Related Work: Clock Reduction

Preserve timed bisimulation

- An algorithm to reduce the number of clocks is provided in (DY '96)

  It works on a syntactical structure of the timed automaton and does not provide the minimum number of clocks

- Checking the existence of a (C, M) automaton timed bisimilar to a given TA is decidable (LLW '95).

  C : number of clocks and M : maximum constant used in the automaton

  The problem of the current paper was *"left as an open (and interesting) problem".* conclusion of (LLW '95)

Our Approach

- Use a semantic representation of the timed automata
  Remove constraints and clocks more effectively.
- Method uses the following operations.
  1. Remove constraints that are never enabled (Stage 1)

Our Approach

- Use a semantic representation of the timed automata
  Remove constraints and clocks more effectively.
- Method uses the following operations.
  1. Remove constraints that are never enabled (Stage 1)
  2. Splitting locations may reduce the number of clocks (Stage 2)

Our Approach

- Use a semantic representation of the timed automata Remove constraints and clocks more effectively.
- Method uses the following operations.
  1. Remove constraints that are never enabled (Stage 1)
  2. Splitting locations may reduce the number of clocks (Stage 2)
  3. Remove constraints that are implied by other constraints (Stage 2)

Our Approach

- Use a semantic representation of the timed automata Remove constraints and clocks more effectively.
- Method uses the following operations.
  1. Remove constraints that are never enabled (Stage 1)
  2. Splitting locations may reduce the number of clocks (Stage 2)
  3. Remove constraints that are implied by other constraints (Stage 2)
  4. Multiple outgoing transitions from a location when considered collectively can remove some constraints (Stage 3)

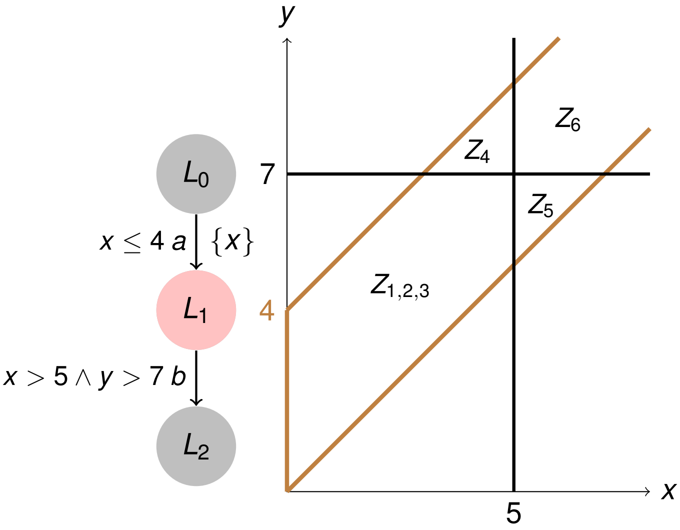
Our Approach

- Use a semantic representation of the timed automata Remove constraints and clocks more effectively.
- Method uses the following operations.
  1. Remove constraints that are never enabled (Stage 1)
  2. Splitting locations may reduce the number of clocks (Stage 2)
  3. Remove constraints that are implied by other constraints (Stage 2)
  4. Multiple outgoing transitions from a location when considered collectively can remove some constraints (Stage 3)
  5. An efficient renaming of clocks across all the locations of the timed automata (Stage 4)
- Input TA A : Through four stages we get A_4 which preserves timed bisimulation and A_4 has the least possible number of clocks.

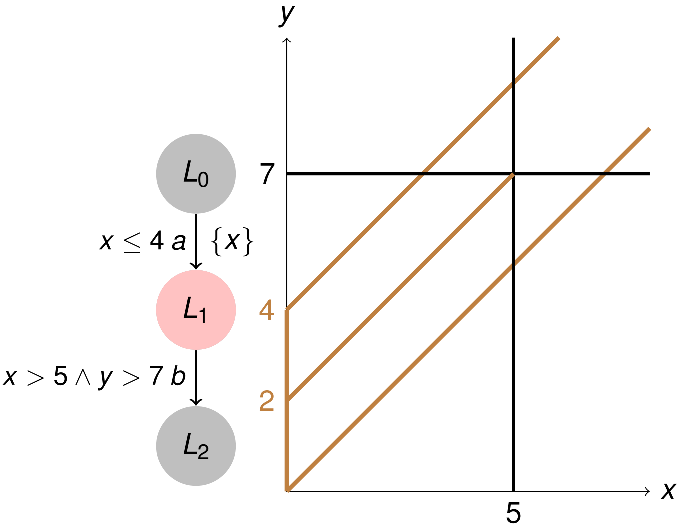
Stage 1: Construct Zone Graph

- Nodes : (l, Z), l : location, Z : zone.
- **zone**: A zone $z = \{v \in \mathbb{R}_{\geq 0}^{|C|} \mid v \models \gamma\}$, where γ is of the form $\gamma ::= \; x \smile c \mid x - y \smile c \mid g \wedge g$, where $c \in \mathbb{Z}$, $x, y \in C$ and $\smile \in \; \{\leq, <, =, >, \geq\}$.
- A zone is a convex set of valuations

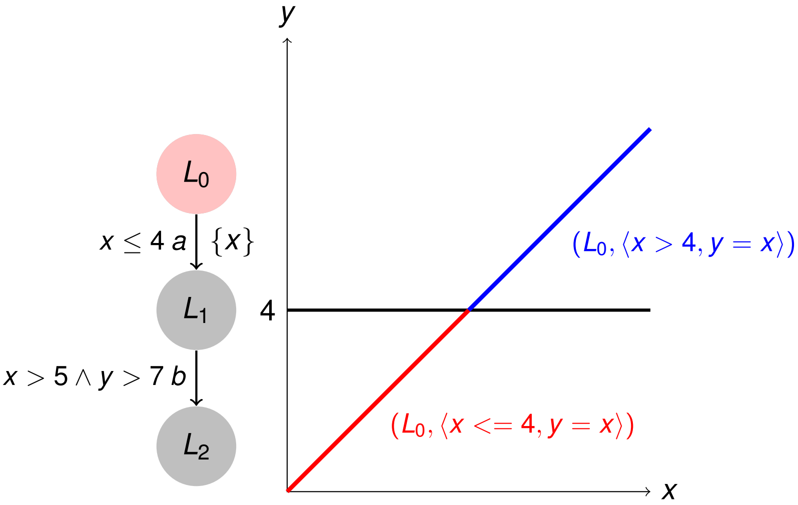
Prestabilizing Zones



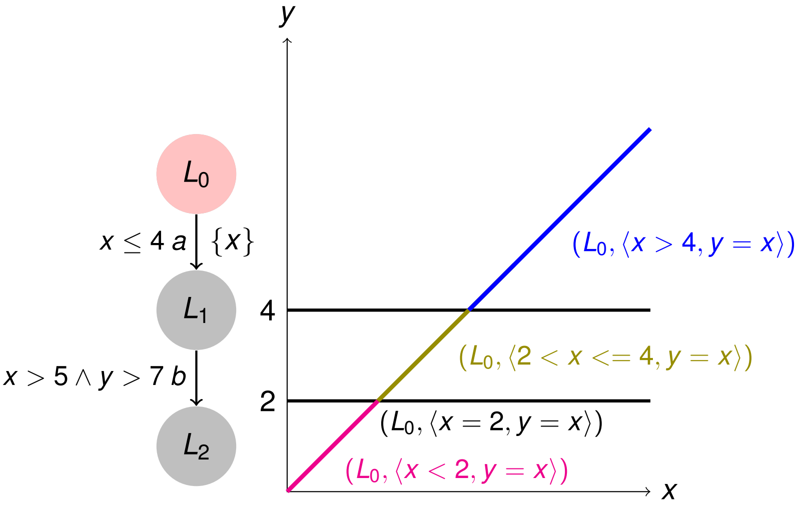
Prestabilizing Zones



Prestabilizing Zones



Prestabilizing Zones



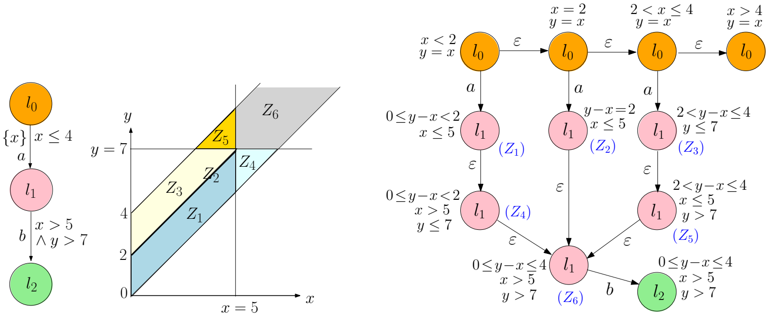
Pre-stability

An important property

Pre-stability ensures that if

- the zone Z in any node (l, Z) in the zone graph is *bounded above*,
- then it is bounded fully from above by a hyperplane $x = h$, where $x \in C$ and $h \in \mathbb{N}$.

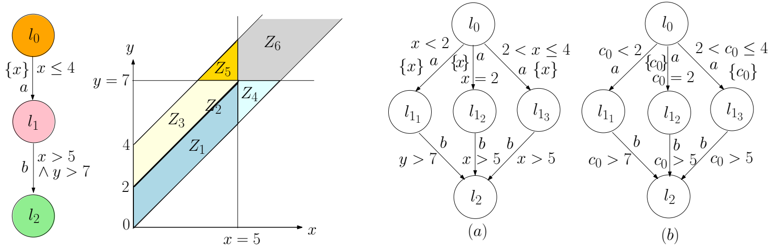
Stage 1: Construct Zone Graph



Consider an edge $l \xrightarrow{g,a,R} l'$ If for all nodes (l, Z), $Z \cap [\![g]\!] = \emptyset$, then remove the edge.

Splitting locations: Stage 2

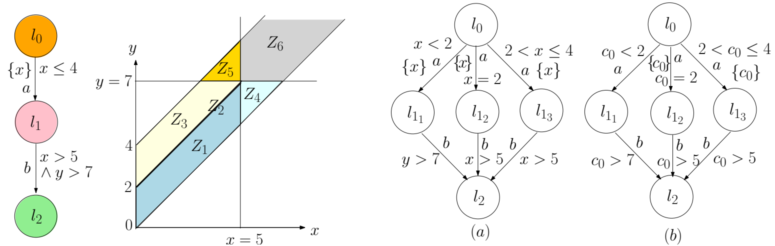
Base zones: Zones without delay predecessors



Some constraints removed from the newly created locations and clocks can be reused

Splitting locations: Stage 2

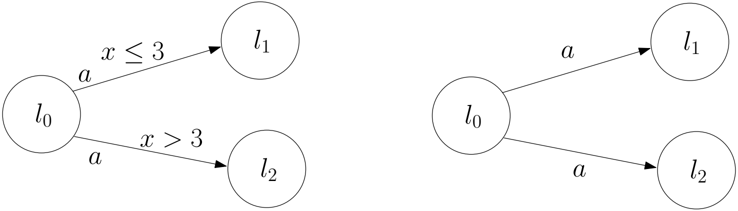
Base zones: Zones without delay predecessors



Number of newly created locations bounded by the number of zones: hence exponential in the number of clocks

No change in underlying semantics; zones distributed to different locations, hence A_2 timed bisimilar to A.

Remove constraints by considering multiple outgoing edges: Stage 3

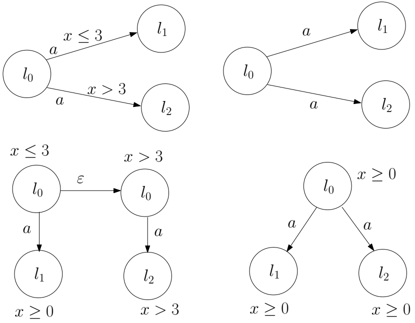


Multiple outgoing edges from the same location

Consider zones with the same set of actions enabled.

Check if removing some hyperplanes and hence some constraints from the zone graph preserves timed bisimilarity.

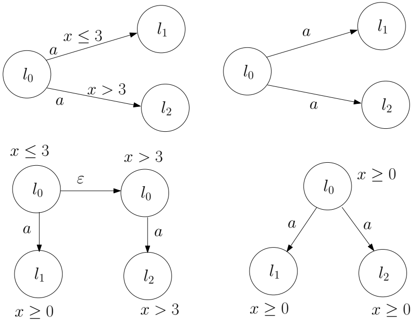
Remove constraints by considering multiple outgoing edges: Stage 3



Check if removing hyperplanes and thus merging some zones preserves timed bisimilarity.

Timed bisimilarity can be checked using the zone graph. (LZ '97, GKNA '13)

Remove constraints by considering multiple outgoing edges: Stage 3



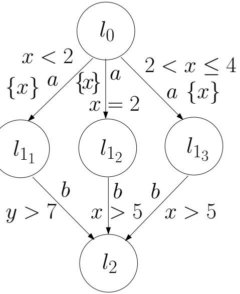
Removing constraints can lead to a reduction in the number of clocks.

Since every constraint removal checks timed bisimulation explicitly, A_3 is timed bisimilar to A.

Clock renaming: Stage 4

Finding active clocks

Active clocks ($act(l)$) : clocks whose valuations are relevant for defining the behaviour of the timed automaton from location l.



$act(l_0) = \{x_0, y_0\},\ act(l_{1_1}) = \{y_{1_1}\},\ act(l_{1_2}) = \{x_{1_2}\},\ act(l_{1_3}) = \{x_{1_3}\},\ act(l_2) = \emptyset$.

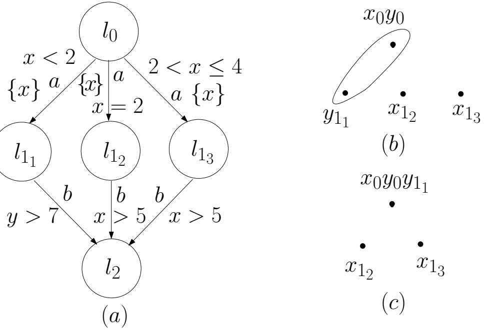
Clock renaming: Stage 4

Partitioning active clocks

Active clocks: Partition *act*(*l*) at a location *l* into equivalence classes.

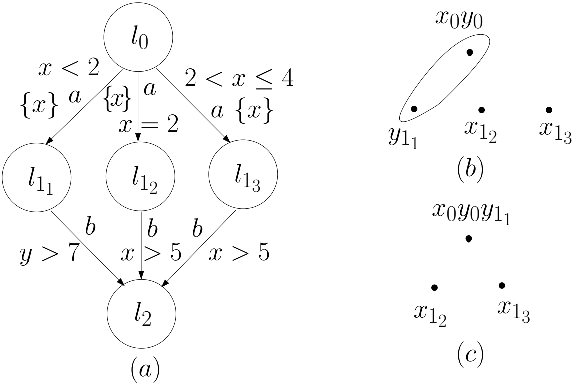
$\forall x, y \in act(l)$, $x \equiv y$ iff $\exists c \in \mathbb{N}.\ x - y = c$.

E.g. $x_0 - y_0 = 0$.



Clock renaming: Stage 4

Vertex colouring of the clock graph



This graph can be coloured using one color.

Complexity: Stage 4

- Number of vertices in clock graph:
  - Bounded by the number of locations $\times$ number of clocks,
  - Hence, exponential in the number of clocks
- Graph colouring: EXPTIME in the number of vertices
  - Thus, $2 - \mathit{EXPTIME}$ in the number of the clocks.

Minimality of clocks

Semantically every hyperplane in the zone graph of A_4 indicates a change in the behaviour and is essential for preserving bisimulation.

Minimality of clocks

Minimal bisimilar TA
For a given timed automaton D, a minimal bisimilar TA is one that is

- timed bisimilar to D and
- has the minimum number of clocks possible.

Fact
For every TA D, there exists a minimal bisimilar TA.

The timed automaton A_4 has the same number of clocks as a minimal bisimilar TA for A.

Our Result

Theorem

Given a timed automaton A,

- there exists an algorithm to construct a TA A_4

  that is timed bisimilar to A such that

  - among all the timed automata that are timed bisimilar to A
  - A_4 has the minimum number of clocks.

- Further the algorithm runs in time that is doubly exponential in the number of clocks of A.

Questions