Algorithmic Verification

Comp4151
Lecture 12-A
Ansgar Fehnker

Outline

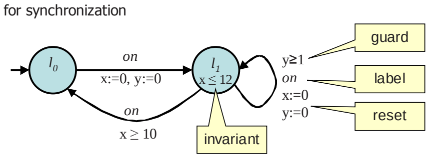
Model checking real-time systems

Themes
- Decidability
- Efficient implementations and data structures
- Application examples

Today
- Efficient reachability
- Zone semantics
- Bounded model checking

Recap

Timed automata
- A finite control graph with locations and edges
- Instantaneous transitions along edges, delays while in location
- Real-valued clocks, that increase at the same rate
- Constraints on clocks as guard on edges
- Clock resets to measure time between transitions
- Invariants in locations to enforce progress
- Labels for synchronization



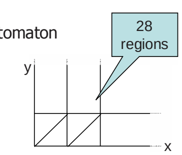
Recap

The reachability problem for timed automata is decidable
Finite partition in regions of equivalent clock valuations
Region automaton with finite semantics

However

Reachability is linear in the size of the region automaton
The size of the region automaton is
- linear in the number of locations,
- exponential in the number of clocks, and
- exponential in the encoding of the constants.

The reachability problem is Pspace complete.

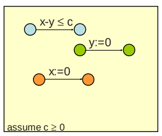


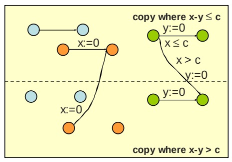
Recap

Diagonal Constraints

Decidability was shown for diagonal-free TA. But

- A timed automaton with diagonal constraints is timed bisimilar to an TA without diagonal constraints





[Berard,Diekert,Gastin,Petit 1998]

Decidability for Timed Automata

Other positive results

- TCTL model checking for timed automata is decidable
  - $\phi ::= p \mid \alpha \mid \neg \phi \mid \phi \vee \phi \mid z \text{ in } \phi \mid \mathbf{A}[\phi\, \mathbf{U}\, \phi] \mid E\, [\phi\, \mathbf{U}\, \phi]$
- Emptiness of untimed language is decidable
  - Is the language accepted by an TA empty? (reachability, Buechi-like acceptance)
- Un-timed language inclusion
- Timed bisimulation is decidable
  - Two TAs are bismilar iff they perform the same actions in bisimilar states they reach bisimilar states.
- Untimed bisimulation is decidable

Decidability for Timed Automata

Negative Results

- The universality problem is undecidable.
  - Does an TA accept all timed words?
- Timed language inclusion is undecidable.
- Timed automata are not determinzable nor complementable
- The following leads to undecidability:
  - Decrementing clocks
  - Incrementing clocks
  - Linear expressions as guards
  - Guards that compare clocks with irrational constants
  - Stop-watches (i.e. clocks that can have rates 0 or 1)
- However there are subclasses of TA such that make of these problems decidable.

Recap

Last Monday

- Discrete time (tick semantics) vs real-time (Continuous Time)
- Timed automata for modelling real-time
- Can be used for continuous time models of unreliable digital clocks.

Last Thursday

- Decidability via partitioning into regions of equivalent states
- Symbolic region semantics
- Region automaton tends to be (too) large.

Zones

An observation

- Guards and invariants are comparisons of clocks and clock differences with constants
- Resets are projections
- All clocks proceed at the same rate
  - Delays do not affect difference-constraints
  - A clock can exceed any bound by delay
- A system of clock and difference constraints defines a union of regions

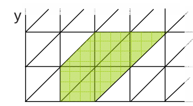
The zone approach

- Propagate clock and difference constraints, rather than regions.

Zones

Zone

- A zone is solution set to a conjunction of constraints of the form
  $x \le n \mid x < n \mid x \ge n \mid x > n \mid x - y \le m \mid x - y < m$
  $n \in$ Nat, $m \in$ Int
- Let $\mathcal{Z}$ be the set of all zones



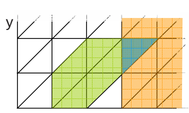
$x \le 4$
$x \ge 1$
$y \le 2$
$y \ge 0$
$y - x \le 0$
$x - y \le 2$

- A region is a zone
- A zone is a convex union of regions
- All invariants and guards are zone

Operations on zones

Conjunction

- Let Z, Z′ be two zones then Z ∧ Z′ := Z ∩ Z′

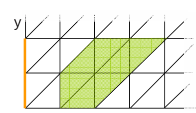


$x \le 4$, $x \ge 1$, $y \le 2$, $y \ge 0$, $y - x \le 0$, $x - y \le 2$	∧	$x \ge 3$, $y \ge 0$	=	$x \le 4$, $x \ge 3$, $y \le 2$, $y \ge 0$, $y - x \le 0$, $x - y \le 2$

Operations on zones

Reset

- Let Z, $x \in C$ then reset(Z,x) ={ v[x:=0] | v ∈ Z}
- Removing all constraints involving x, add $x \le 0$ and $x \ge 0$

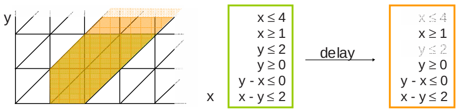


$x \le 4$, $x \ge 1$, $y \le 2$, $y \ge 0$, $y - x \le 0$, $x - y \le 2$	reset x →	$x \le 0$, $x \ge 0$, $y \le 2$, $y \ge 0$, $y - x \le 0$, $x - y \le 2$

Operations on zones

Delay

- Let Z, $x \in C$ then $\text{delay}(Z) = \{ v+d \mid v \in Z, d \in \mathbf{R}_{\geq 0} \}$
- Removing all upper bounds on clocks



Zone Semantics

Definition

The *symbolic zone semantics* of a timed automaton $A = (Loc, l_0, \Sigma, E, Inv)$ is given as a transition system with

- set of states $S = \{ (l, Z) \mid l \in Loc,\ Z \in \mathcal{Z} \}$
- initial state $s_0 = (l_0, \mathbf{0})$

Zone Semantics

- transition relation $R \subseteq S \times \Sigma \cup \{ \boldsymbol{\delta} \} \times S$ that contains the following

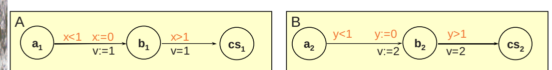
discrete transitions

$(l,Z) \xrightarrow{\sigma} (l',Z')$ if there exist $(l,g,\sigma,r,l') \in E$ s.t. $Z \cap g \neq \emptyset$, $Z' \cap \text{Inv}(l') \neq \emptyset$ and $Z' = \text{reset}(Z \cap g, r) \cap \text{Inv}(l')$

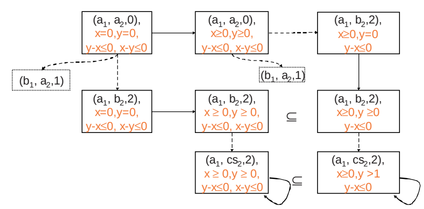
delay transitions

$(l,Z) \xrightarrow{\delta} (l,Z')$ $Z' = \text{delay}(Z) \cap \text{Inv}(l)$

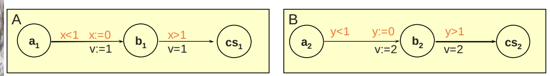
Example



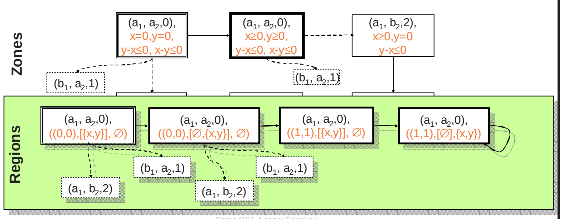
Show that A || B can not reach the critical section in location (cs_1, cs_2)



Example



Show that A || B can not reach the critical section in location (cs_1,cs_2)



Reachability with Zones

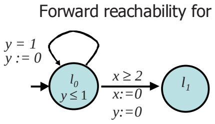
Forward Reachability

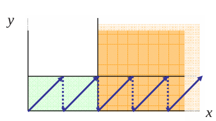
```
Pass := {}, Wait := {(l0,Z0)}
while Wait ≠ {} do
   select (l,Z) from Wait
   if l=lf
    return "lf reachable"
    fi
   if Z'⊄ Z forall (l',Z') in Pass then
      add (l,Z) to Pass
      forall (m,Z') such that (l,Z) → (m,Z'):
         add (m,Z') to Wait
  fi
od
return "lf not reachable"
```

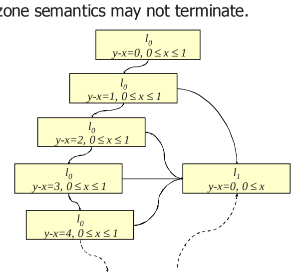

Reachability with Zones

Observation

Forward reachability for the zone semantics may not terminate.







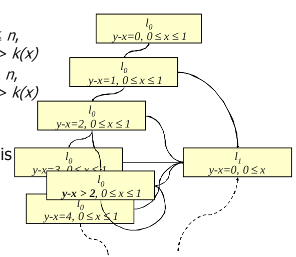
Reachability with Zones

k-normalization

Given a *closed* zone Z

- Remove all constraints $x < n$, $x \leq n$, $x\text{-}y < m$ and $x\text{-}y \leq m$ where $n,m > k(x)$
- Replace all constraints $x > n$, $x \geq n$, $x\text{-}y > m$ and $x\text{-}y \geq m$ where $n,m > k(x)$ with x > $k(x)$ and $x\text{-}y > k(x)$

Number of k-normalized zones is finite.

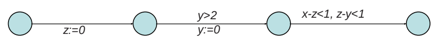


Reachability with Zones

Another observation

- For TAs with diagonal constraints the soundness result is lost.
- Location may be reachable as an artefact of k-normalization.
- Discovered in 2002 by Patricia Bouyer (*Timed Automata may cause some troubles, Untameable timed automata*)

oops



without normalization	$x-y = 0$ $y-z = 0$ $z-x = 0$	$x-y = 0$ $z-x \leq 0$ $z-y \leq 0$	$x-y > 2$ $y-z \leq 0$ $z-x \leq 0$ $x > 2$	not reachable
with normalization	$x-y = 0$ $y-z = 0$ $z-x = 0$	$x-y = 0$ $z-x \leq 0$ $z-y \leq 0$	$x-y > \mathbf{1}$ $y-z \leq 0$ $z-x \leq 0$ $x > \mathbf{1}$	reachable

oops

- There are automata (> 3 clocks) such that no sound k-normalization exists

Reachability with Zones

k-Normalization with Difference Constraints

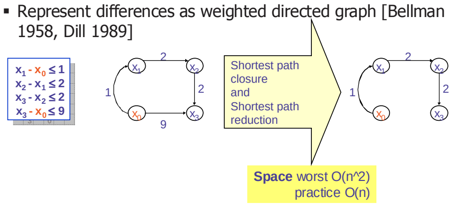
- Given a set of difference constraints *G* that are used in the timed automaton normalize a zone *Z* as follows:
  - Collect all constraints *g* that are either satisfied by all or no valuations in the un-normalized zone *Z*.
  - Split the zone for each constraint in *G* that intersects with the un-normalized zone *Z*.
  - Apply k-normalization to thus obtained zones
  - Add all difference constraints (or their negations) that were collected in the first step to the zone.
- This solves the problem.

Efficient operations on zones

Difference Bound Matrices

A compact representation of a minimal set of constraints

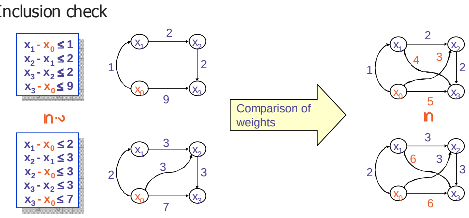
- Given a set of constraints on clocks introduce a special clock $\mathbf{x_0}$ constant to zero
- Represent differences as weighted directed graph [Bellman 1958, Dill 1989]



Efficient operations on zones

Difference Bound Matrices

Inclusion check

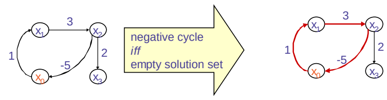


Efficient operations on zones

Difference Bound Matrices

Emptiness

$x_1 - x_0 \leq 1$
$x_2 - x_1 \leq 3$
$x_3 - x_2 \leq 2$
$x_0 - x_2 \leq -5$



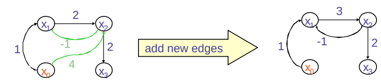
Efficient operations on zones

Difference Bound Matrices

Conjunction

$x_1 - x_0 \leq 1$
$x_2 - x_1 \leq 2$
$x_3 - x_2 \leq 2$

$x_2 - x_1 \leq -1$
$x_2 - x_0 \leq 4$

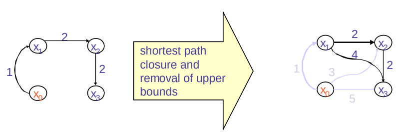


Efficient operations on zones

Difference Bound Matrices

Delay

$x_1 - x_0 \leq 1$
$x_2 - x_1 \leq 2$
$x_3 - x_2 \leq 2$
$x_3 - x_0 \leq 9$

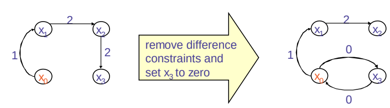


Efficient operations on zones

Difference Bound Matrices

Reset

$x_1 - x_0 \leq 1$
$x_2 - x_1 \leq 2$
$x_3 - x_2 \leq 2$
$x_3 - x_0 \leq 9$

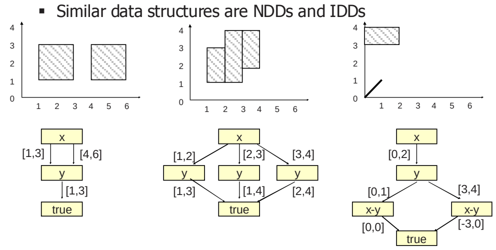


Now we've got everything for an efficient implementation.

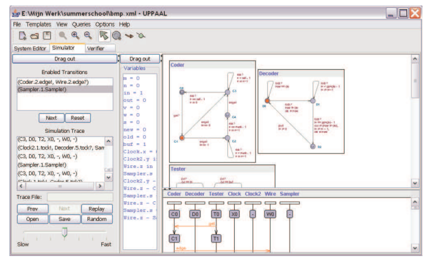
Other Data Structures

Clock difference diagrams

- CDDs are BDD like structures
- Similar data structures are NDDs and IDDs



Example



Timed Automata Verification

Uppaal

- Supports zone semantics and CDDs
- On-the-fly forward reachability
- Uses optimizations such as
  - bit-state hashing
  - convex-hull approximation
  - active clock reduction
- Simplified specification language
  - A[] *p*
  - E<> *p* *Safety*
  - A<> *p*
  - E[] *p* *Liveness*
  - *p* imply *q* (read "*p* leads to *q*, i.e *p* ⇒ A<> *q*)

Timed Automata Verification

Liveness in Uppaal

Checking *p* → *q*

- Compute the zone automaton, split zones such that they agree on validity of *p* and *q*.
- *p* → *q* does not hold if one can find the following
  - A loop such that after reaching *p*, *q* will never hold.
  - An unbounded zone, such that there exist an infinite delay, such that after reaching *p*, *q* can not be reached.
  - A deadend zone such that after reaching *p*, *q* can not be reached.

Uppaal contains optimizations such that the splitting is not physically done.

Timed Automata Verification

Uppaal

Kronos

- Uses zone semantics
- Model checking TCTL
  - Product automaton computed in advance
- On-the fly forward reachability
- Untimed language inclusion
- Optimizations such as
  - active clock reduction
  - convex-hull approximation

Timed Automata Verification

Uppaal

Kronos

Fully symbolic

- RED
  - Region Encoding Diagram, encodes region automaton as BDD
- DDD
  - Difference Decision Diagrams,
- TMV
  - quantifier elimination and deciding of constraints from real-valued to
    boolean variables, BDDs, SAT solving, full TCTL support.
- Mathsat
  - Bounded model checking using hybrid SAT.

...

Summary

Timed Autoamata

- Framework for modelling systems with real time
- Underlying infinite state transition systems
- Decidability via region automaton construction
- Efficiency via zones and DBMs
- Alternatives to DBMs exists
- First tool using SAT-like techniques