Overview

Next two weeks

Model checking real-time systems

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

Today
- Decidability and region equivalence
- Symbolic model checking for the region automaton
- Other decidability results

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

Algorithmic Verification

Comp4151
Lecture 11-B
Ansgar Fehnker

Overview

Model Checking Approaches

- Explicit State Model Checking
- Symbolic Model Checking
- Bounded Model Checking
- Automatic Abstraction Refinement
- Correctness of software, hardware and protocols
- Correctness for finite state systems
**Recap**

**Biphasic Mark Protocol**

- Protocol and model correct if certain locations are not reachable
- Problems
  - The state space is infinite: $S = \{ (x,y) \mid x \in \text{Loc}, y \in \text{Inv}, x \in C \rightarrow R_{E} \}$
  - The transition relation is infinite: $R \subseteq S \times \Sigma \cup R_{E} \times S$

**Problems**

- Decidability
  - Does there exist an algorithm that decides for any TA $A$ and a location $l$, if $l$ is reachable in $A$ or not.

**Preliminaries**

**Constraints**

Given a set of clocks $C$ let $\psi(C)$ be defined by

$$\psi = \phi \land \neg q \land x \leq n.$$ 

where $x, y \in C, a \in N, n \in \mathbb{N}$

**Finite Control Graph**

Timed automata $(\text{Loc}, \Sigma, E, \text{Inv})$ has a finite set of locations Loc and a finite set of edges $E$.

**Infinite Transition System**

The underlying timed transition system $(S_{E}, R_{E})$ has an infinite set of states and an infinite number of transitions.

**Introduction**

**Region Automaton**

- Proposed by Alur and Dill [AD94, AD91]
- Provides a finite abstraction
- Used for many other decidability results

**Reachability**

- Check if a given location in a given TA is reachable from the initial state

**Decidability**

- Does there exist an algorithm that decides for any TA $A$ and a location $l$, if $l$ is reachable in $A$ or not.

**Approach**

Given a TA $(\text{Loc}, \Sigma, E, \text{Inv})$ with underlying TTS $(S_{E}R_{E})$

- Define an equivalence relation $\approx$ on clock valuations such that
  - Given states $(\ell, \nu), (\ell', \nu')$ of TTS with $(\ell, \nu) \approx (\ell', \nu')$, we have location $l$ is reachable from $(\ell, \nu)$ iff $l'$ is reachable from $(\ell', \nu')$
  - The number of equivalence classes $S_{E}/\approx$ is finite

**Problems**

- How to define such an equivalence relation?
- How to represent an equivalence class?

---

*Main problem*  
*Secondary problem*
Region Equivalence

First observation
- All clocks are compared only to integer values

Equivalence
- Define a integer grid on clock valuations

\[(l, v) = (l', v') \iff \forall x \in \mathbb{C}, \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \]  (1)

Region Equivalence

Second observation
- Need to distinguish between valuations above and below diagonals

Equivalence
- Define a integer grid on clock valuations
- Divide each cell along its diagonals
- Diagonals satisfy \( \text{frac}(v(x)) = \text{frac}(v'(x)) \)

\[(l, v) = (l', v') \iff \] (2)
Region Equivalence

Third Observation

- It matters whether the value of a clock is an integer

Equivalence

- Define a integer grid on clock valuations
- Divide each cell along its diagonals $\text{frac}(v(x)) = \text{frac}(v(y))$
- Divide the cells into vertices, edges, diagonals, and open simplices

$(\ell, y) = (\ell', y')$ if and only if
  
  - $\ell = \ell'$ and $\forall x \in C. \, \lfloor v(x) \rfloor = \lfloor v(x) \rfloor$
  - $\forall x, y \in C. \, \text{frac}(v(x)) \leq \text{frac}(v(y)) \Leftrightarrow \text{frac}(v'(x)) \leq \text{frac}(v'(y))$
  - $\forall x \in C. \, \text{frac}(v(x)) = 0 \Leftrightarrow \text{frac}(v'(x)) = 0$

(7)

Region Equivalence

Forth Observation

- The value of a clock is irrelevant once it exceeds the biggest constant

Equivalence

- Define a integer grid on clock valuations
- Divide each cell along its diagonals $\text{frac}(v(x)) = \text{frac}(v(y))$
- Divide the cells into vertices, edges, diagonals, and open simplices
- Bound the partition using the biggest constant in guards and invariants

$(\ell, y) = (\ell', y')$ if and only if
  
  - $\ell = \ell'$ and $\forall x \in C. \, \lfloor v(x) \rfloor \leq \lfloor y(x) \rfloor$
  - $\forall x, y \in C. \, v(x) \leq v(y) \Rightarrow \text{frac}(v(x)) \leq \text{frac}(v(y))$
  - $\forall x \in C. \, \lfloor v(x) \rfloor \leq \lfloor y(x) \rfloor \Rightarrow \text{frac}(v(x)) = 0 \Leftrightarrow \text{frac}(v'(x)) = 0$

There are countable but infinitely many equivalence classes.
Overview

Next two weeks

Model checking real-time systems

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

Today
- Decidability and region equivalence
- Symbolic model checking for the region automaton
- Other decidability results

Symbolic Semantics

Use clock equivalence to define a finite region automaton.

1st step: represent regions symbolically:

- Given a set of clocks C with maximal constants \(c_i\), we represent a region as a triple \(H=(h(C_0,...,C_i), C)\) with:
  - \(h: C \rightarrow \text{Nat}\) that assigns to each clock a natural number \(\leq c_i\)
  - \(C_0,...,C_i\) and \(C\) define a partition of the set of clocks.
- \(C_0\) and \(C\) may be empty.

Let \(\mathcal{H}\) be the finite set of all possible \(H\) given the set of clocks and the maximal constants.

Symbolic Semantics

A clock valuation \(v \in (h[C_0,...,C_i], C)\) if

- \(\lfloor v(x) \rfloor = h(x)\) for \(x \in C_i\)
- \(\lceil v(x) \rceil > c_i\) for \(x \in C_i\)
- \(\text{frac}(v(x)) = \text{frac}(v(y))\) for \(x, y \in C_i\)
- \(\text{frac}(v(x)) < \text{frac}(v(y))\) for \(x, y \in C_i\), \(i < j\)

Equivalent clock valuations

- \(v' = (h[C_0,...,C_j], C')\) implies \((v) = (v')\)

Symbolic Semantics

Use clock equivalence to define a finite region automaton.

2nd step: define symbolic operations on regions

Reset

Given a region \(H=(h[C_0,...,C_i], C)\) and \(x \in C_i\)

- if \(i=0\) then \(\text{reset}(H) = (h[C_0,...,C_i], C)\) with \(H = (h(x=0), C_0,...,C_i, C)\)
- if \(0 < i\) and \(C_i = (x)\) then \(\text{reset}(H) = (h[C_0,...,C_{i-1}, x=0], C)\) with \(H = (h(x=0), C_0,...,C_i, C)\)
- otherwise \(\text{reset}(H) = (h[C_0,...,C_i], C)\) with \(H = (h(x=0), C_0,...,C_i, C)\)

Given a region \(H=(h[C_0,...,C_i], C)\) and \(x \in C_i\)

- \(\text{reset}(H) = (h[C_0,...,C_i], C)\) with \(H = (h(x=0), C_0,...,C_i, C)\)
- \(\text{reset}(H) = (h[C_0,...,C_i], C)\) with \(H = (h(x=0), C_0,...,C_i, C)\)
Symbolic Semantics

Use clock equivalence to define a finite region automaton.

2\textsuperscript{nd} step: define symbolic operations on regions

\textbf{Delay}

Given a region \( H = \{ h'_i \mid C_{h_i} \subseteq C_h \} \)

- If \( C_h = \emptyset \) then
  \( \text{delay}(H) = \{ h'_i \mid C_{h_i} \subseteq C_h \} \) with \( C_{h'_i} = C_{h_i} \setminus \{ x \mid h(x) = c_i \} \) and \( C_{h'_i} = C_{h'_i} \cup \{ x \mid h(x) = c_i \} \)

- If \( C_h = \emptyset \) and \( k>0 \) then
  \( \text{delay}(H) = \{ h'_i \mid C_{h_i} \subseteq C_h \} \) with \( h(x) = h(x)+k \) if \( x \in C_i \) and \( h(x) = h(x) \) otherwise.

- Otherwise (if all clocks in \( C_h \))
  \( \text{delay}(H) = H \)

Symbolic Semantics

Reminder

The operational semantics of a timed automaton \( \mathcal{A} = (\text{Loc}, \Sigma, \mathcal{E}, \mathcal{I}, \mathcal{F}) \) is given as a transition system \( TS(\mathcal{A}) \) with

- set of states \( S = \{ (i,v) \mid i \in \text{Loc}, v \models \text{Env}(i) \} \)

- initial state \( x_0 = (i_0,0) \)

- transition relation \( R \subseteq S \times \Sigma \times R_{\mathcal{A}} \times S \) that contains the following

- discrete transitions \( (i,v) \xrightarrow{(a,v')} (i',v') \) if there exist \( (i_0,v_0) \vdash E \) s.t. \( v_0 \models a \) and \( v_0[v_0/a] = v' \)

- delay transitions \( (i,v) \xrightarrow{(a,v') \tau} (i',v') \) for \( d = \mathcal{D}_{\mathcal{A}} \) for all \( 0 \leq d' \leq d \) holds \( v + d' \models \text{Env}(i) \)

\textbf{Infinitely many states and transitions!}
Symbolic Semantics

Definition

The region semantics of a timed automaton \( A = (\text{Loc}, \Sigma, \delta, \mathcal{I}, m_0) \) is given as a transition system \( R(A) \) with

- set of states \( S = \{ (l, H) | l \in \text{Loc}, H \in \mathcal{P} \} \)
- initial state \( s_0 = (l_0, (0, C, \emptyset)) \)
- transition relation \( R \subseteq S \times (\Sigma \cup \{\delta\}) \times S \) that contains the following

  discrete transitions
  \( (l, H) \xrightarrow{\sigma} (l', H') \) if \( \exists (l, \sigma, r, l', H') \in \delta \) such that \( H' = \text{reset}(H, r) \)

  delay transitions
  \( (l, H) \xrightarrow{\delta} (l', H') \) if \( H' \models \text{Inv}(l) \) and \( H' = \text{delay}(H) \)

Finitely many states and transitions!

Example

Show that \( A || B \) can not reach the critical section in location \((\text{cs}_1, \text{cs}_2)\).

Model Checking

Forward reachability
- Start with the initial state \((l_0, (0, C, \emptyset))\) of the region automaton
- Explore the state space using the transition relation until either
  - A fix-point has been reached, or
  - The target location has been reached.
- Search orders are DFS, BFS, random DFS, …

Backward reachability
- Start with all regions in the target location
- Explore the state space using the inverse transition relation until either
  - A fix-point has been reached
  - The initial state \((l_0, (0, C, \emptyset))\) has been reached

Useful theorem

Given a location \( l \) of timed automaton \( A \), it is reachable in \( TS(A) \) iff it is reachable in \( R(A) \).

Sketch of proof

\( \overset{=>}{=} \)

Given an execution \((l_0, H_0) \xrightarrow{\ldots} (l_1)\) there exist a symbolic execution \((l_0, H_0) \xrightarrow{\ldots} (l_1)\) with \( v \in H \)

\( \overset{<=}{=} \)

Given a symbolic execution \((l_0, H_0) \xrightarrow{\ldots} (l_1)\) there exist execution \((l_0, H_0) \xrightarrow{\ldots} (l_1)\) with \( v \in H \)
Decidability for Timed Automata

Other positive results

- TCTL model checking for timed automata is decidable
  \[ φ \land A \land \neg φ \land φ \lor \neg φ \land Z \text{ in } [A \land Φ \lor Φ \lor E \text{ in } Φ \lor Φ] \]
- Emptiness of untimed language is decidable
  - Is the language accepted by an TA empty? (reachability, Büchi-like acceptance)
- Un-timed language inclusion
- Timed bisimulation is decidable
  - Two TAs are bisimilar if 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 determinizable 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.

Diagonal Constraints

Another useful theorem (almost forgotten)

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

Summary

Results

- The reachability problem for timed automata is decidable
- Finite symbolic semantics for the region automaton
- The region-construction useful to prove decidability of other problems.

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 maximal constants.
- The reachability problem is P-space complete.

Next week: Efficient model checking of timed automata