ECE/CS 584: Verification of Embedded Computing Systems Model Checking Timed Automata

Sayan Mitra

Lecture 09

Reachability analysis: Integer Timed Automaton

Sayan Mitra
Verifying cyberphysical systems
mitras@illinois.edu

This course so far

- A modeling framework
  - Discrete and continuous dynamics
  - Compositional (modular) modeling
- General proof techniques for proving invariants

Next

- Focus on specific classes of Hybrid Automata for which safety properties (invariants) can be verified completely automatically
  - Alur-Dill's Timed Automata[1] (Today)
  - Rectangular initializaed hybrid automata
  - Linear hybrid automata
  - …
- Later we will look at other types of properties like stability, liveness, etc.
- We will introduce notions of abstractions and invariance are still going to be important

[1] Rajeev Alur et al. The Algorithmic Analysis ofHybrid Systems. Theoretical Computer Science, colume 138, pages 3-34, 1995.

Today

- Algorithmic analysis of (Alur-Dill’s) Timed Automata[1]
  - A restricted class of what we call hybrid automata in this course with only clock variables

[1] Rajeev Alur and David L. Dill. A theory of timed automata. Theoretical Computer Science, 126:183-235, 1994.

Clocks and Clock Constraints

- A clock variable x is a continuous (analog) variable of type real such that along any trajectory τ of x, for all t ∈ $\tau.dom, (\tau \downarrow x)(t) = t$.

- For a set X of clock variables, the set Φ(X) of integral clock constraints are expressions defined by the syntax:
  g ::= x $\leq q \mid x \geq q \mid \neg\, g \mid g_1 \wedge g_2$
  where $x \in X$ and $q \in \mathbb{Z}$

- Examples: x = 10; x ∈ [2, 5); true are valid clock constraints
- What do clock constraints look like?

- Semantics of clock constraints $[g]$

Integral Timed Automata

- **Definition.** A integral timed automaton is a HIOA $\mathcal{A} = \langle V, \Theta, A, \mathcal{D}, \mathcal{T} \rangle$ where
  - V = X ∪ $\{l\}$, where X is a set of n clocks and l is a discrete state variable of finite type Ł
  - A is a finite set
  - $\mathcal{D}$ is a set of transitions such that
    - The guards are described by clock constrains $\Phi(X)$
    - $\langle x, l \rangle - a \rightarrow \langle x', l' \rangle$ implies either $x' = x$ or $x = 0$
  - $\mathcal{T}$ set of clock trajectories for the clock variables in X

Example: Light switch

Math Formulation

automaton Switch

  variables

  internal x, y:Real := 0, loc: {on,off} := off

  transitions

  internal push

    pre x $\geq$ 2

    eff if loc = on **then** x := 0

      else x,y := 0; loc := off

  internal pop

    pre y = 15 $\bigwedge$ loc = off

    eff x := 0

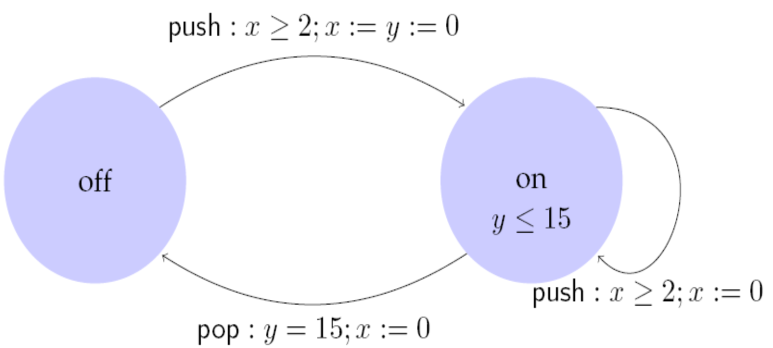
  trajectories

    invariant loc = off => y $\leq$ 15

    evolve d(x) = 1; d(y) = 1

Description

Switch can be turned on whenever at least 2 time units have elapsed since the last turn on. Switches off automatically 15 time units after the last on.



Control State (Location) Reachability Problem

- Given an ITA, check if a particular (discrete) control state is reachable from the initial states
- Why is control state reachability (CSR) good enough?

- This problem is decidable [Alur Dill]

- Key idea:
  - Construct a finite automaton that is a time-abstract ***bisimilar*** to the ITA (behaves identically with respect to control state reachability)
  - Check reachability of FSM

An equivalence relation with a finite quotient

- Under what conditions do two states x_1 and x_2 of the automaton $\mathcal{A}$ behave identically with respect to control state reachability (CSR)?
  - When do they satisfy the same set of clock constraints?
  - When would they continue to satisfy the same set of clock constraints?

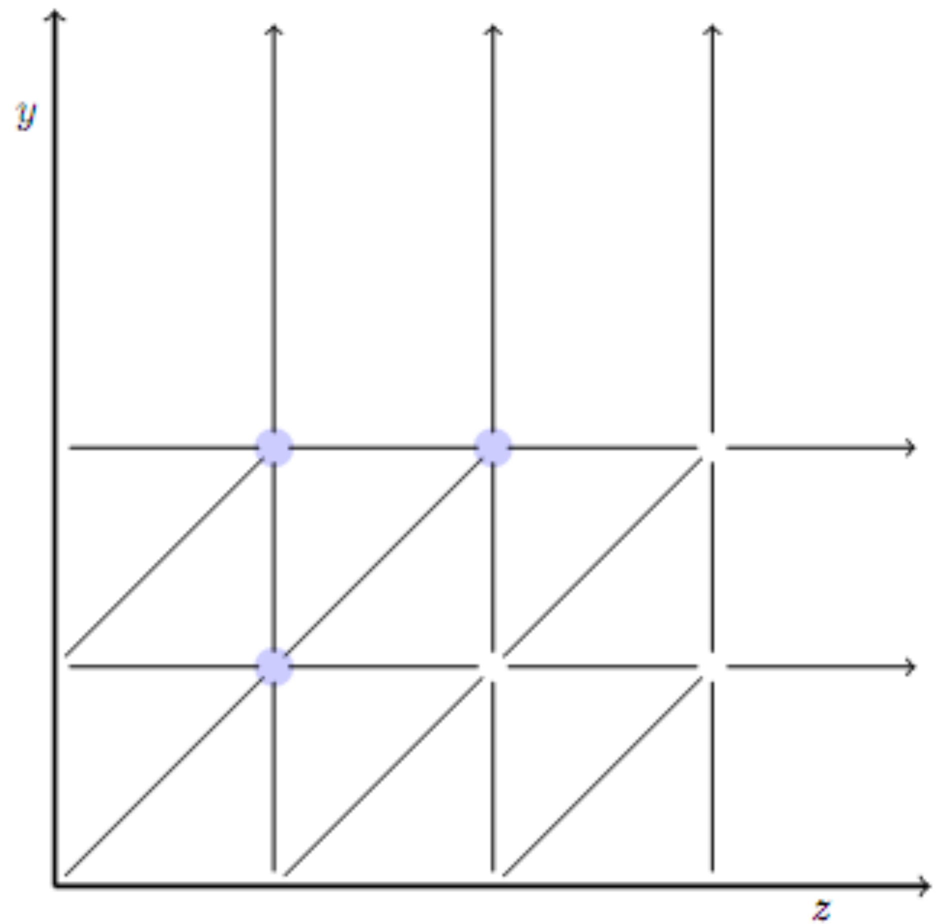
- $x_1.loc = x_2.loc$ and
- x_1 and x_2 satisfy the same set of clock constraints
  - For each clock y int($x_1.y$) = int($x_2.y$) or int$(x_1.y) \geq c_{\mathcal{A}y}$ and int$(x_2.y) \geq c_{\mathcal{A}y}$. ($c_{\mathcal{A}y}$ **is the maxium clock guard** of y)
  - For each clock y with $x_1.y \leq c_{\mathcal{A}y}$, frac$(x_1.y) = 0$ iff frac$(x_2.y) = 0$
  - For any two clocks y and z with $x_1.y \leq c_{\mathcal{A}y}$ and $x_1.z \leq c_{\mathcal{A}z}$, frac($x_1.y$) $\leq$ frac($x_1.z$) iff frac($x_2.y$) $\leq$ frac($x_2.z$)
- Lemma. This is a **equivalence relation** on *val(V)* the states of $\mathcal{A}$

- The partition of *val(V)* induced by this relation is are called **clock regions**

What do the clock regions look like?

Example of
Two Clocks

X = {y,z}
$c_{\mathcal{A}y}$ = 2
$c_{\mathcal{A}z}$ = 3



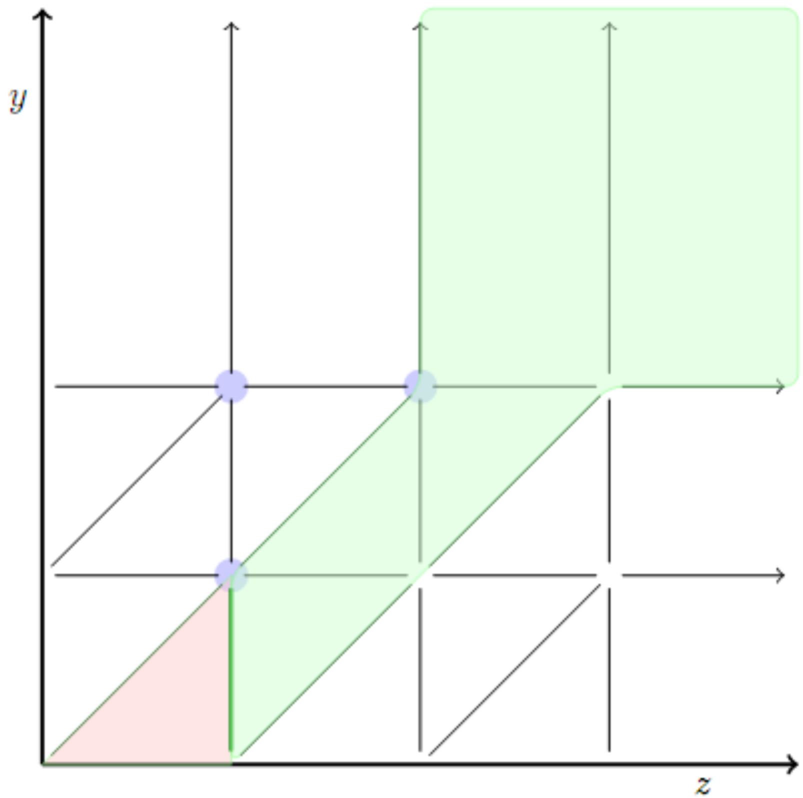
Complexity

- Lemma. The number of clock regions is bounded by $|X|!\, 2^{|X|} \prod_{z \in X} (2c_{\mathcal{A}z} + 2)$.

Region Automaton

- ITA (clock constants) defines the clock regions
- Now we add the “appropriate transitions” between the regions to create a finite automaton which gives a visits the same set of states (but timing information is lost) ITA with respect to control state reachability
  - **Time successors**: Consider two clock regions γ and γ', we say that γ' is a time successor of γ if there exits a trajectory of ITA starting from γ that ends in γ’
  - **Discrete transitions**: Same as the ITA

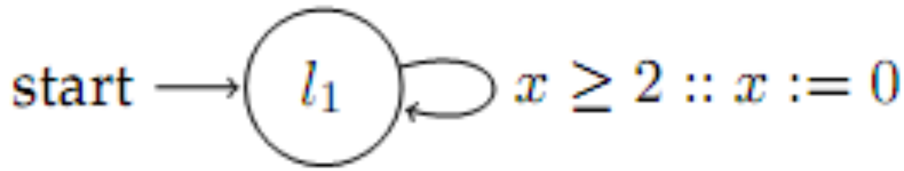
Time Successors



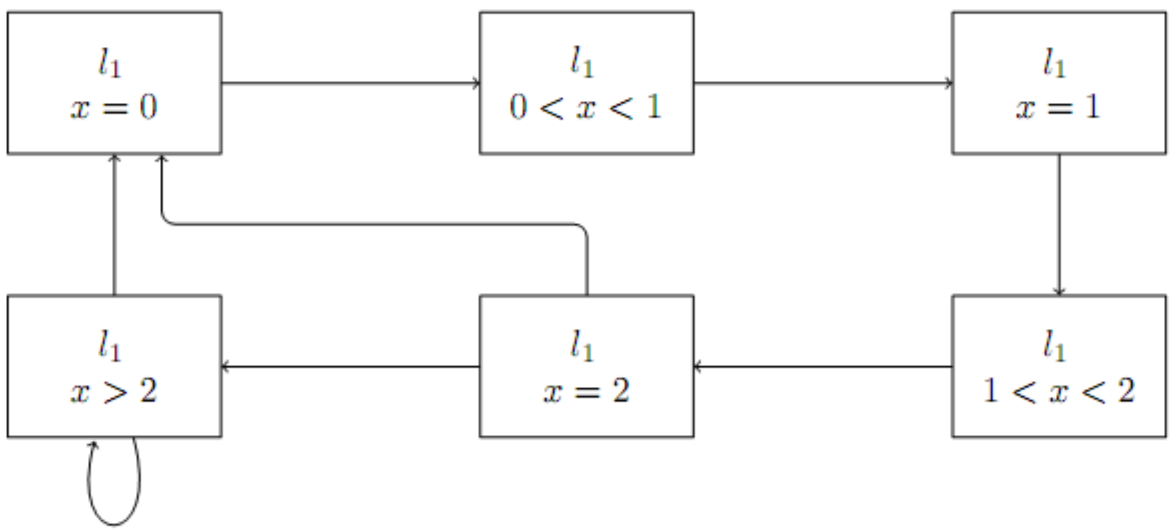
The clock regions in blue are time successors of the clock region in red.

Example 1: Region Automata

ITA

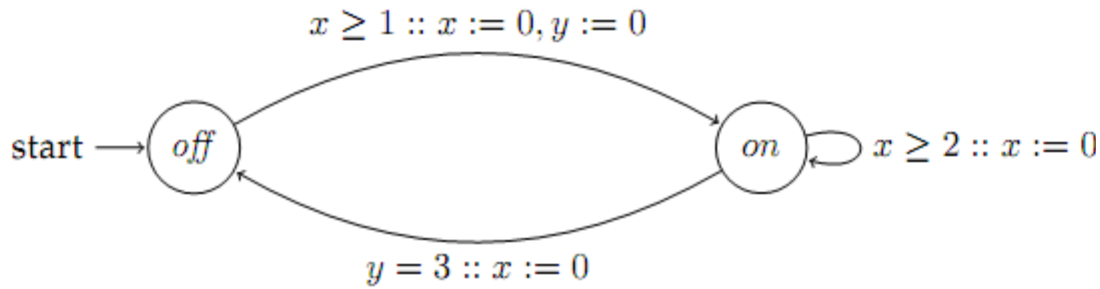


Corresponding FA

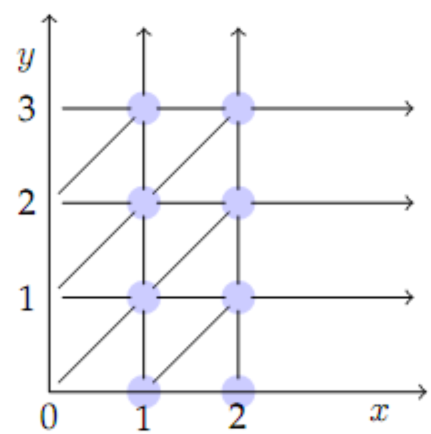


Example 2

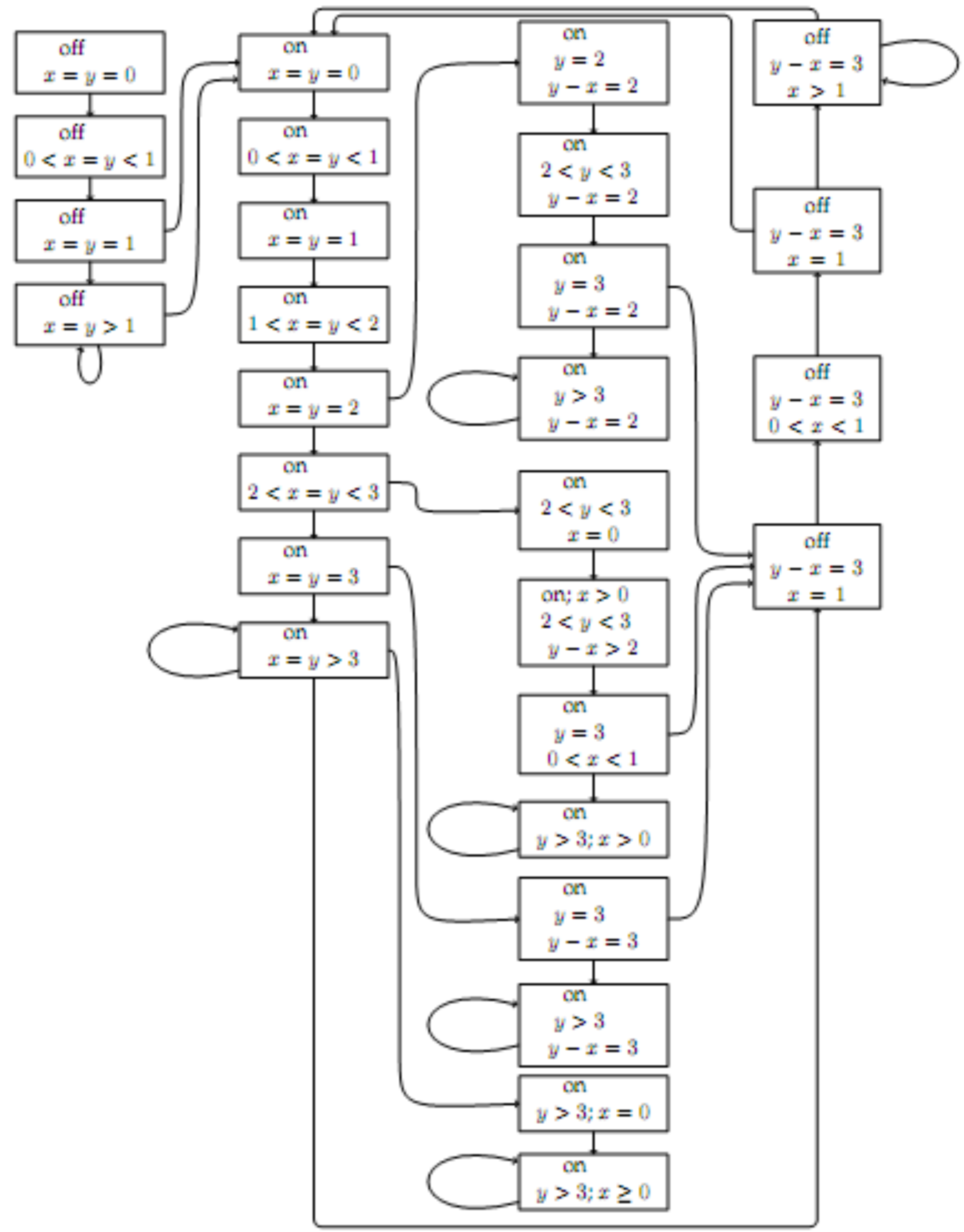
ITA



Clock Regions



Corresponding FA



$|X|!\, 2^{|X|} \prod_{z \in X} (2c_{\mathcal{A}z} + 2)$

Drastically increasing with the number of clocks

Summary

- ITA: (very) Restricted class of hybrid automata
  - Clocks, integer constraints
  - No clock comparison, linear
- Control state reachability
- Alur-Dill's algorithm
  - Construct finite bisimulation (region automaton)
  - Idea is to lump together states that behave similarly and reduce the size of the model
- UPPAAL model checker based on similar model of timed automata