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

Sayan Mitra

Lecture 09

What we have seen so far

- A very general modeling framework (Lynch et al.'s Hybrid Automata)
  - Complex discrete dynamics
  - Possibly nonlinear continuous dynamics
  - Distributed
- General proof techniques for the above model
  - Inductive invariants for proving safety
  - Simulation relations for trace inclusion
- Introduction to a General-purpose theorem prover (PVS) and examples of mechanizing proofs for state machines
  - How to model state machines in PVS
  - How to construct invariant proofs
  - Can be partially automated but requires a lot of manual work

Next

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

Today

- Algorithmic analysis of (Alur-Dill's) Timed Automata
  - A restricted class of what we call hybrid automata in this course with only clock variables
- Reference: 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 ≤ q | $x \geq q$ | ¬ g | $g_1 \wedge g_2$

  where $x \in X$ and $q \in \mathbb{Z}$
- Examples: x = 10; x ∈ [2, 5); true are valid clock constraints
- Semantics of clock constraints $[g]$

Integral Timed Automata

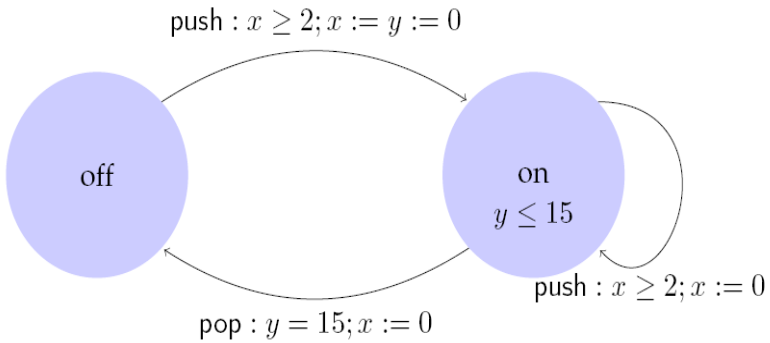
- **Definition.** A **integral timed automaton** is a HIOA A = $\langle V, Q, \Theta, A, \mathcal{D}, \mathcal{T} \rangle$ where
  - V = X ∪ $\{l\}$, where X is a set of n clocks and l is a discrete state variable of finite type Ł
  - A is a finite set
  - $\mathcal{D}$ is a set of transitions such that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

Example: Light switch

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

automaton Switch

- **internal** push; pop
- **variables**
  internal x, y:Real := 0, loc:{on,off} := off
- **transitions**
- **internal** push
  pre x ≥ 2
  eff if loc = on **then** y := 0 **fi;** x := 0; loc := off
- **internal** pop
  pre y = 15 /\ loc = off
  eff x := 0
- **trajectories**
  invariant loc = on \/ loc = off
  stop when y = 15 /\ loc = off
  evolve d(x) = 1; d(y) = 1



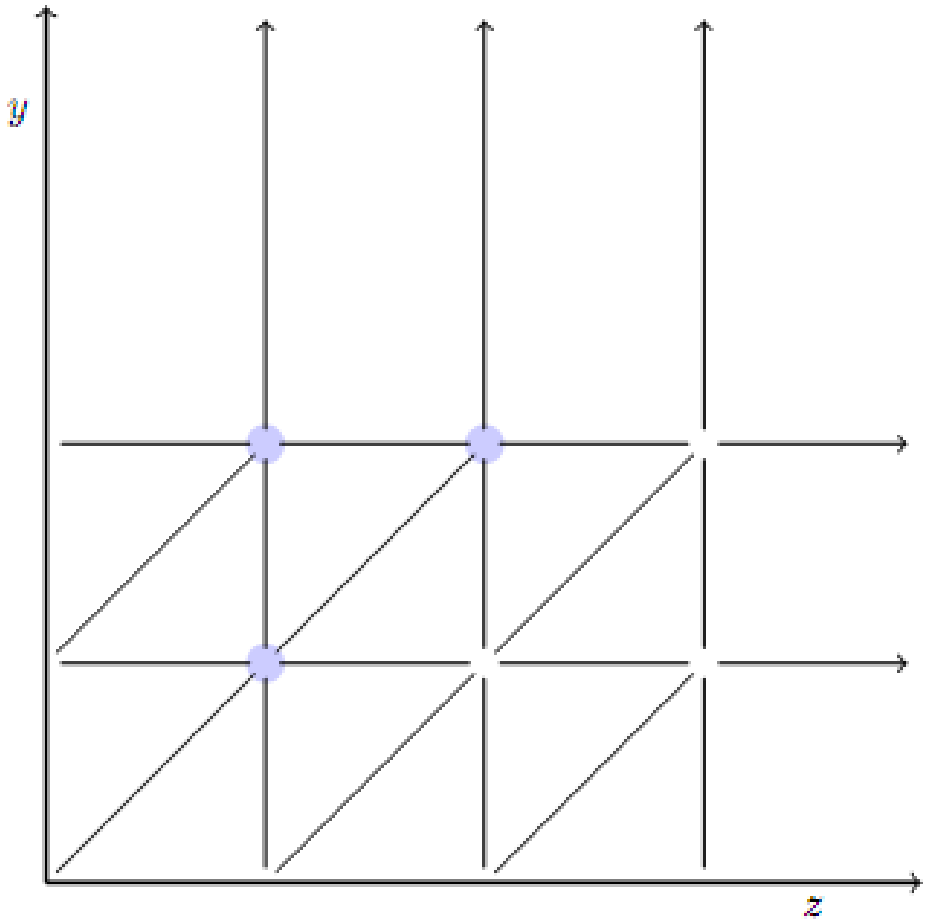
Control State (Location) Reachability Problem

- Given an ITA, check if a particular location is reachable from the initial states
- This problem is decidable
- Key idea:
  - Construct a Finite State Machine that is a time-abstract bisimilar to the ITA
  - Check reachability of FSM

A Simulation Relation with a finite quotient

- When two states $\mathbf{x_1}$ and $\mathbf{x_2}$ in Q behave identically?
- $\mathbf{x_1}.loc = \mathbf{x_2}.loc$ and
- $\mathbf{x_1}$ and $\mathbf{x_2}$ satisfy the same set of clock constraints
  - For each clock y $\text{int}(\mathbf{x_1}.y) = \text{int}(\mathbf{x_2}.y)$ or $\text{int}(\mathbf{x_1}.y) \geq c_{\mathcal{A}y}$ and $\text{int}(\mathbf{x_2}.y) \geq c_{\mathcal{A}y}$
  - For each clock y with $\mathbf{x_1}.y \leq c_{\mathcal{A}y}$, $\text{frac}(\mathbf{x_1}.y) = 0$ iff $\text{frac}(\mathbf{x_2}.y) = 0$
  - For any two clocks y and z with $\mathbf{x_1}.y \leq c_{\mathcal{A}y}$ and $\mathbf{x_1}.y \leq c_{\mathcal{A}z}$, $\text{frac}(\mathbf{x_1}.y) \leq \text{frac}(\mathbf{x_1}.z)$ iff $\text{frac}(\mathbf{x_2}.y) \leq \text{frac}(\mathbf{x_2}.z)$
- **Lemma.** This is a **equivalence relation** on Q
- The partition of Q induced by this relation is are called **clock regions**

What do the clock regions look like?



X = {y,z}

$c_{\mathcal{A}y}$ = 2

$c_{\mathcal{A}z}$ = 3

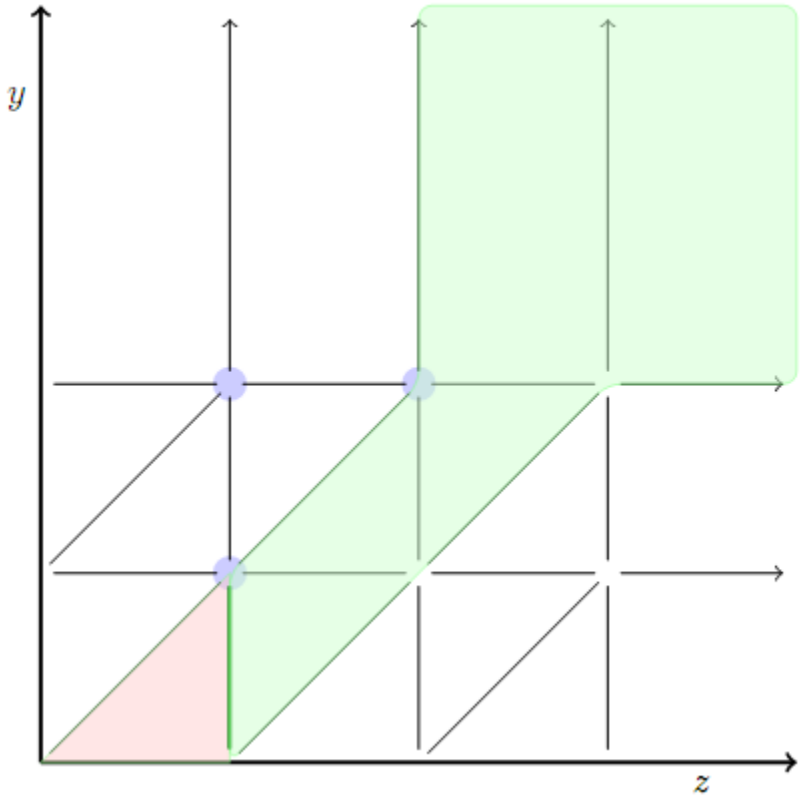
Complexity

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

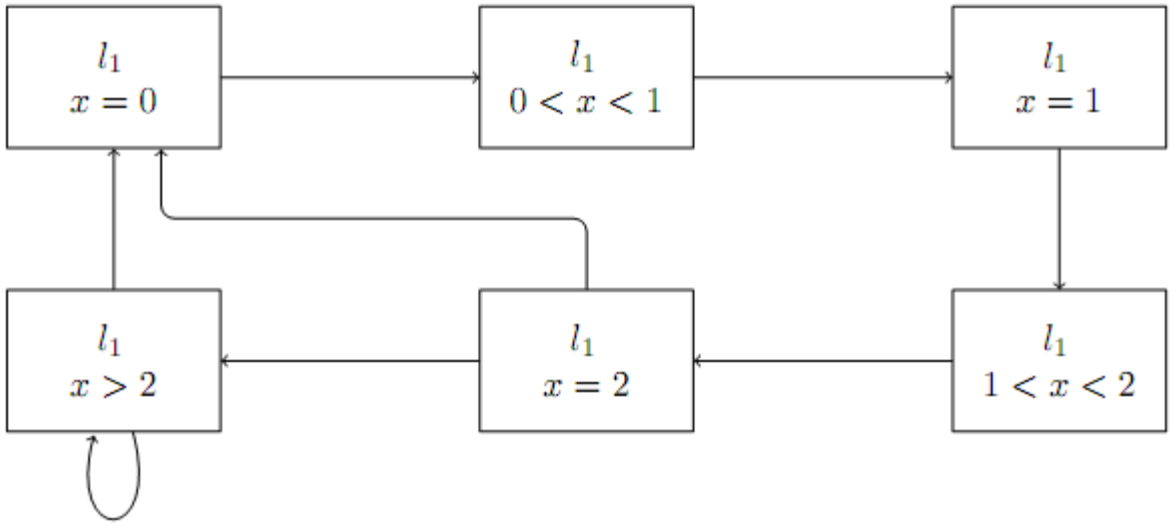
Region Automaton

- ITA (clock constants) defines the clock regions
- Now we add the "appropriate transitions" between the regions to create a finite automaton which gives a **time abstract bisimulation** of the 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**

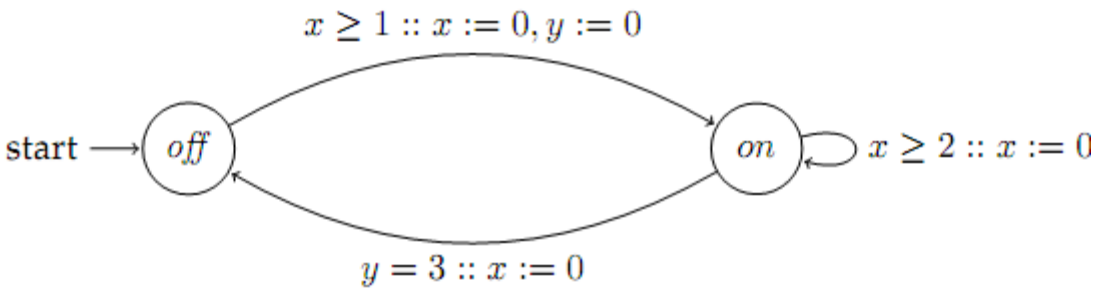
Time Successors

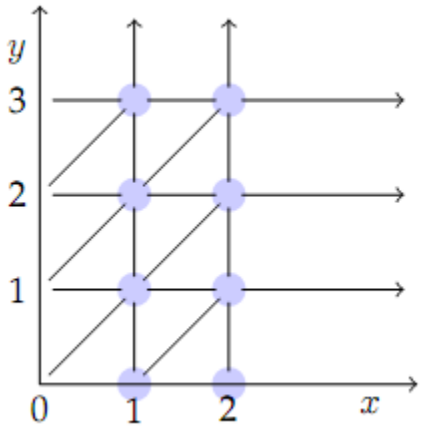


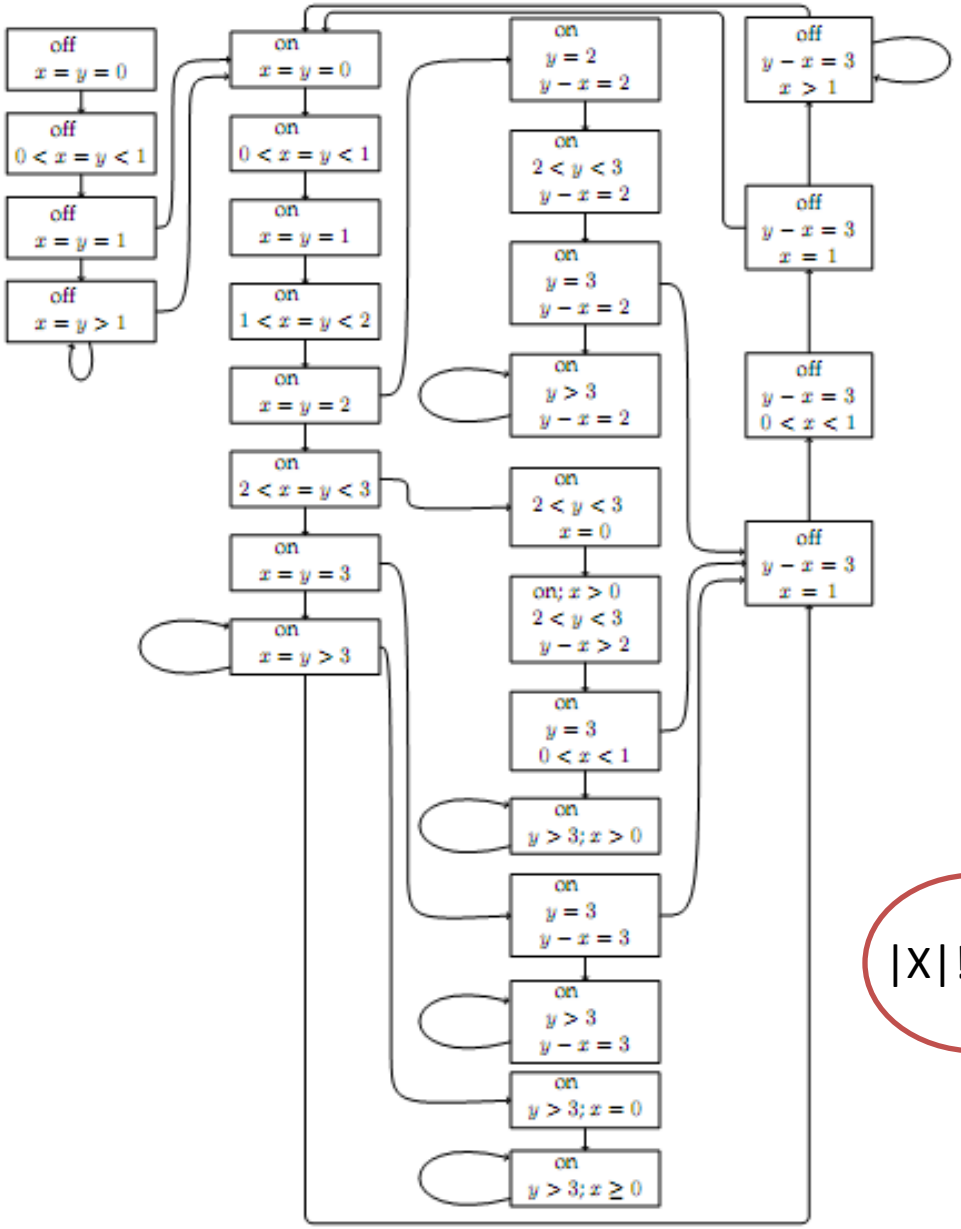
Example 1: Region Automata



Example 2







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

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