ECE/CS 584: Hybrid Automaton Modeling Framework Executions, Reach set, Invariance

Lecture 03

Sayan Mitra

Announcements

- Project proposals due in a week
  - 2 pages with goals, description & milestones
- Allerton Conference special session on Verification of CPS
  - October 4th, 1:30 pm at Allerton House
  - Free!

Plan for Today

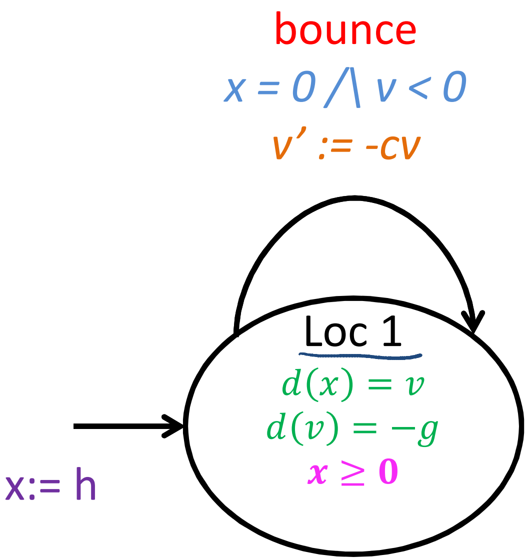
- Examples of hybrid models
- Executions, reach sets, invariants

Hybrid Automata (a.k.a Timed Automata Kaynar, et al. 2005)

$\boldsymbol{\mathcal{A}} = (X, Q, \Theta, E, H, \mathcal{D}, \mathcal{T})$

- X**:** set of **internal or state variables**
- $Q \subseteq val(X)$ set of **states**
- $\Theta \subseteq Q$ set of **start states**
- *E,H* sets of **internal** and **external actions**, A= E ∪ H
- $\mathcal{D} \subseteq Q \times A \times Q$
- $\mathcal{T}$*:* set of **trajectories** for X which is closed under **prefix, suffix, and concatenation**

Bouncing Ball



Automaton Bouncingball(c,h,g)

variables: analog x: Reals := h, v: Reals := 0

states: True

actions: external bounce

transitions:

bounce

pre $x = 0 \wedge v < 0$ ← guard

eff v := -cv ← reset map

$v \in [-c_1 v, -c_2 v]$

trajectories:

evolve d(x) = v; d(v) = -g

invariant $x \geq 0$

$d(x) \in [av, bv]$

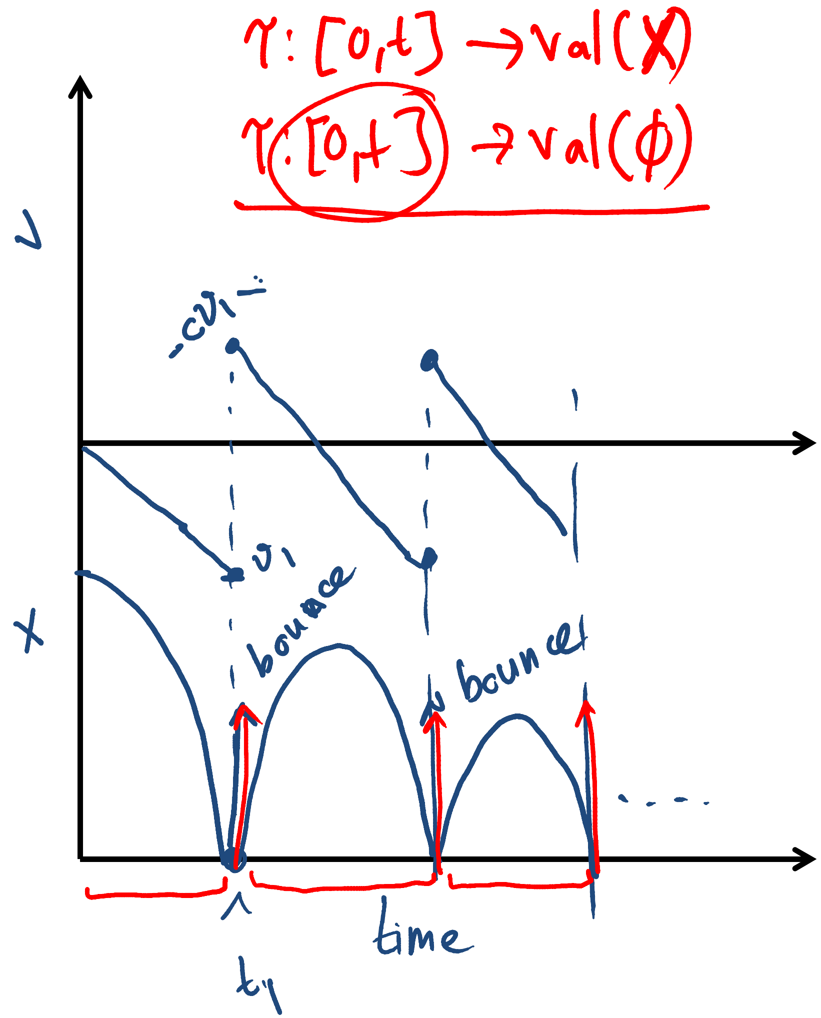
Graphical Representation used in many articles

TIOA Specification Language (close to PHAVer & UPPAAL's language)

Semantics: Executions and Traces

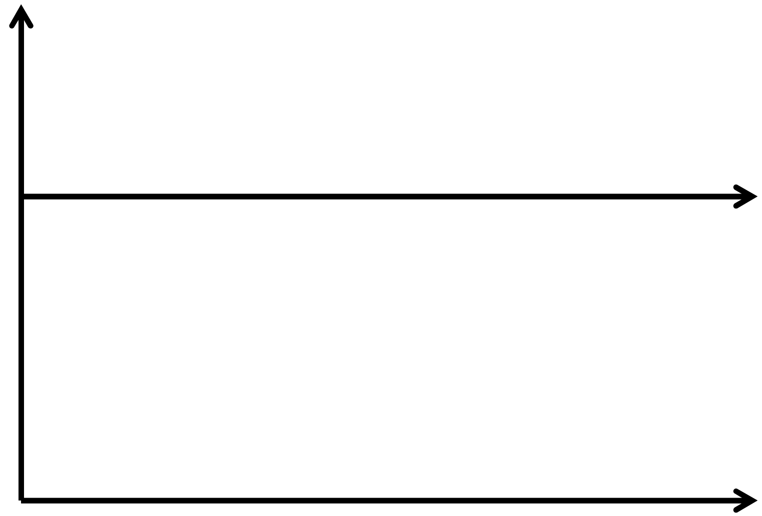
- An **execution fragment** of $\mathcal{A}$ is an (possibly infinite) alternating (A, X)-sequence $\alpha = \tau_0\, a_1\, \tau_1 a_2 \tau_2$ ... where
  - $\forall$ i $\tau_i.lstate \xrightarrow{a_{i+1}} \tau_{i+1}.fstate$
- If τ_0.fstate ∈ Θ then its an **execution**
- $\text{Execs}_{\mathcal{A}}$ set of all executions

- The **trace** of an execution: external part of the execution. Alternating sequence of **external** actions and trajectories of the **empty set** of variables

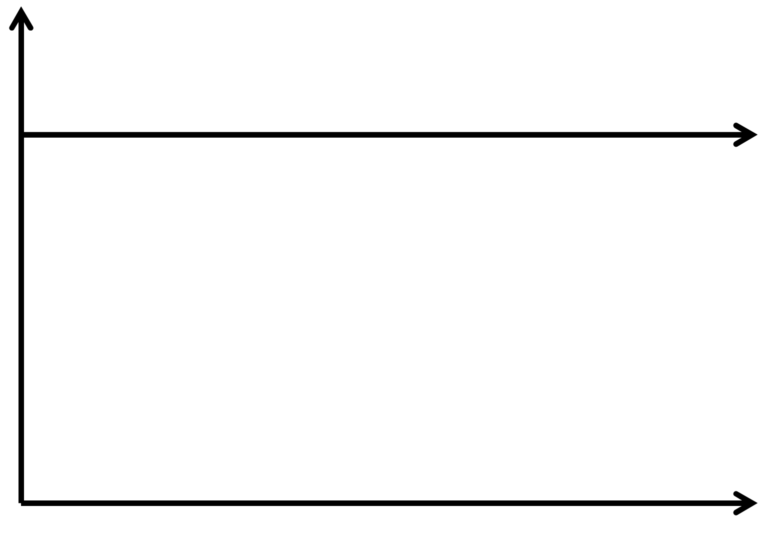


Traces

- With E = {} H = {bounce}
  - Internal bounce



- With E = {bounce} H = {}
  - External bounce

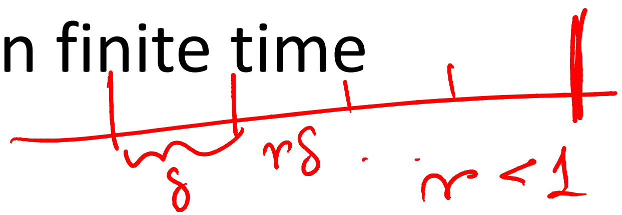


Special kinds of executions

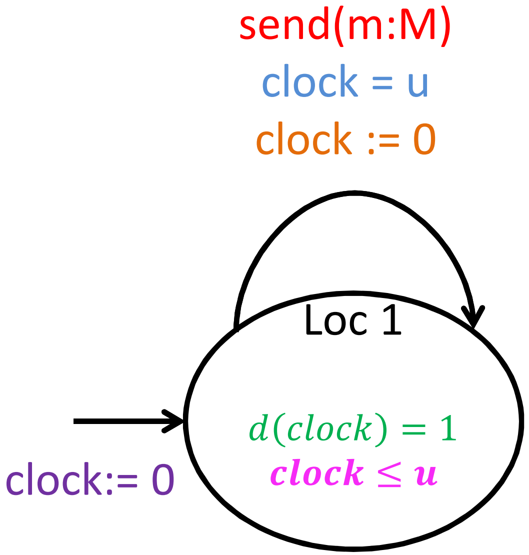
- **Infinite**: Infinite sequence of transitions and trajectories
- **Closed**: Finite with final trajectory with closed domain

  $\alpha = \tau_0 a_1 \ldots \tau_n$ $\quad \tau_n : [0,t] \to val(x)$

  $[0,\infty)$
- **Admissable**: Infinite duration
  - May or may not be infinite
- **Zeno**: Infinite but not admissable
  - Infinite number of transitions in finite time



Periodically Sending Process



Automaton PeriodicSend(u, M)

variables: analog clock: Reals := 0

states: True

actions: external send(m:M)

transitions:

send(m)

pre clock = u

eff clock := 0

trajectories:

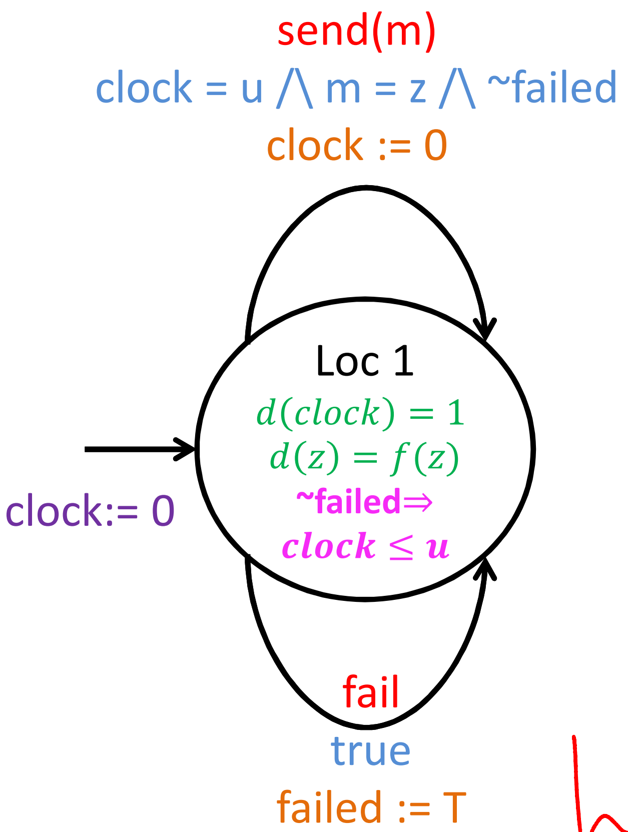
evolve d(clock) = 1

stop when clock=u

Graphical Representation used in many articles

TIOA Specification Language (close to PHAVer & UPPAAL's language)

Another Example: Periodically Sending Process



Automaton PeriodicSend(u)

variables: analog

clock: Reals := 0, z:Reals, failed:Boolean := F

actions: external send(m:Reals), fail

transitions:

send(m)

pre clock = u /\ m = z /\ ~failed

eff clock := 0

fail

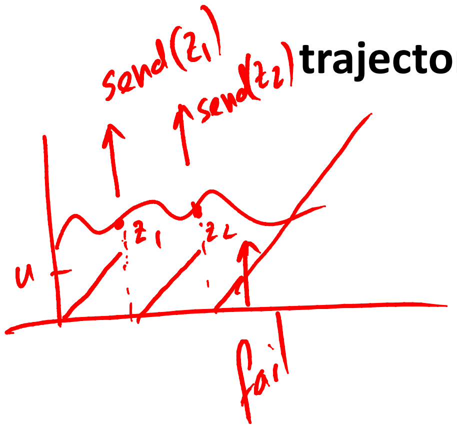
pre true

eff failed := T

trajectories:

evolve d(clock) = 1, d(z) = f(z)

stop when ~failed /\ clock=u



Inv : $S \subseteq Q$

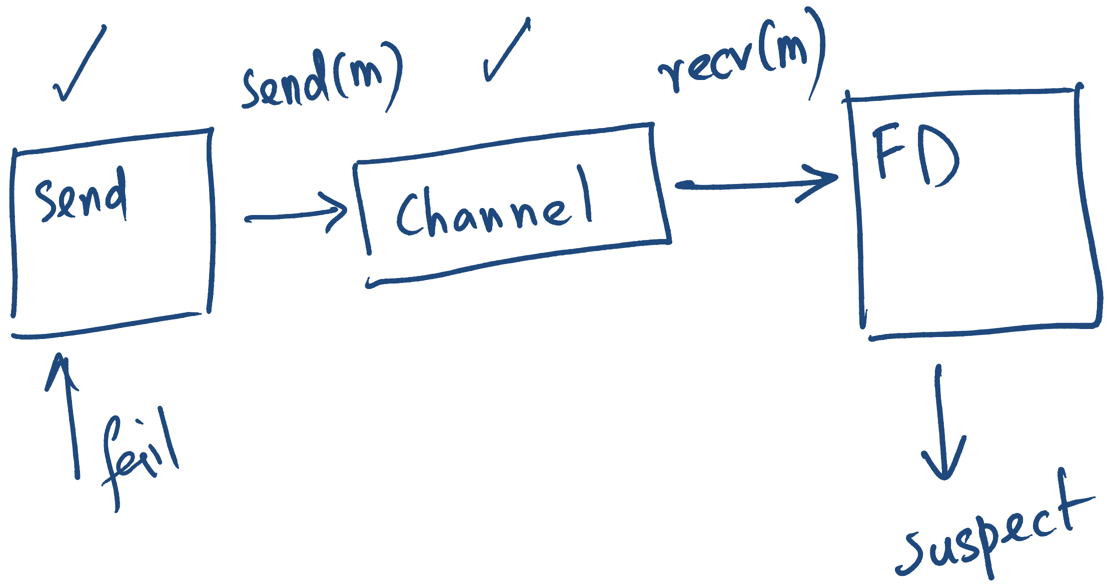
$\forall t' \in \gamma.\mathrm{dom} \quad \gamma(t') \in S$

Stop : $S \subseteq Q$

$\forall t' \in \gamma.\mathrm{dom} \quad \gamma(t') \in S \Rightarrow t' = \gamma.\mathrm{ltime}$

Modeling a Simple Failure Detector System

- Periodic send
- Channel
- Timeout



Time bounded channel & Simple Failure Detector

Automaton Timeout(u,M)
variables: suspected: Boolean := F,
clock: Reals := 0
actions: external receive(m:M), timeout
transitions:
receive(m)
pre true
eff clock := 0; suspected := false;
timeout
pre ~suspected /\ clock = u
eff suspected := true

$(1-\rho) \leq d(clock$

trajectories:
evolve d(clock) = 1
stop when clock = u /\ ~suspected

deadline

Automaton Channel(b,M)
variables: queue: **Queue**[M,Reals] := {}
clock: Reals := 0
actions: external send(m:M), receive(m:M)
transitions:
send(m)
pre true
eff queue := append(<m, clock+b>, queue)
receive(m)
pre head(queue)[1] = m
eff queue := queue.tail
trajectories:
evolve d(clock) = 1
stop when $\exists$ m, d, $\langle m,d \rangle \in$ queue
/\ clock=d

Reachable States and Invariants

- A **state** **v** $\in$ Q is reachable if there exists an execution α with α.lstate = v.
- $\boldsymbol{Reach}_{\mathcal{A}}$ Set of all reachable states
- An S $\subseteq Q$ is an **invariant** if $\boldsymbol{Reach}_{\mathcal{A}} \subseteq$ S
  - Generalizes the idea of conservation
- So, any invariant necessarily contains the set Θ of start states

- Examples:
  - Bouncing ball: $h \geq x \geq 0$
  - $0 < v^2 \leq 2g(h-x)$
  - Periodic send: ~failed $\Rightarrow clock \leq u$

Example Inductive Invariance Proof

- Invariant. For **x** ∈ Reach_{TC} :
  ∀ <m,d> ∈ x.queue: $\boxed{\mathbf{x}.\text{clock} \leq d \leq \mathbf{x}.\text{clock+b}}$ (1)
- Proof. Fix **x** ∈ Reach_{TC}.
- ∃ α ∈ Exec_{TC} with α.lstate = **x.** Fix α = $\tau_0\ a_1\ \tau_1 a_2 \ldots \tau_n$. [Def. Reach_{TC}]
- Induction on the length of the execution
- Base case: If we set **x** = τ_0. fstate then (1) should hold
  – Holds vacuously as x.queue = {} [Def of initial states]

$x \xrightarrow{\tau} x'$

(1) t' (1)

- Inductive step 1: Consider any τ, let **x =** τ.fstate and **x'** = τ.lstate and τ.ltime = t. Assume **x** satisfies (1) and show that **x'** also.
  – x.queue = x'.queue [**trajectory Def**], Fix <m,d> in x.queue
  – x.clock ≤ d [By Assumption]
  – Suppose x'.clock > d
  – x'.clock - x.clock > $\boxed{\text{d - x.clock}}$
  – t > d - x.clock, then there exists t' ∈ τ.dom and t' < t where τ(t').clock = d
  – By **stop when** τ.ltime = t' which is a contradiction
  – Also, since d ≤ **x**.clock+b, d≤ **x'**.clock+t+b

$x \xrightarrow{send(m)} x'$

$x \xrightarrow{recv(m)} x'$

- Inductive step 2: Consider **x—send(m)→x'**
- Inductive step 3: Consider **x—receive(m)→x'** follows from Assumption.

Inductive Invariants

- An invariant S is inductive if for any $v \in S$
  - If v—a→ v' then $v' \in S$
  - If v—τ→ v' then $v' \in S$
- Proof rule for establishing an inductive invariant S
- Theorem: For any set of states S if
  1. for any $v \in \Theta$ start state, $v \in S$
  2. If $v \in S$ and v—a→ v' then $v' \in S$
  3. If $v \in S$ and v—τ→ v' then $v' \in S$

  Then $\text{Reach}_{\mathcal{A}} \subseteq S$

Pre and Post Computations

- For a given set of states $Q' \subseteq Q$, and action $a \in A$
  - $Post\_trans(Q', a) = \{ v' \mid \exists\, v \in Q', v \xrightarrow{a} v' \}$
  - $Post\_trans(Q', A') = \{ v' \mid \exists\, v \in Q', a \in A,\ v \xrightarrow{a} v' \}$
  - $Post\_taj(Q') = \{ v' \mid \exists\, v \in Q', \tau \in T, v \xrightarrow{\tau} v' \}$
  - $Post(Q') = Post\_trans(Q', A) \cup Post\_taj(Q')$
  - $Pre\_trans(Q', A') = \{ v \mid \exists\, v' \in Q', a \in A,\ v \xrightarrow{a} v' \}$
  - $Pre\_taj(Q') = \{ v \mid \exists\, v's \in Q', \tau \in T, v \xrightarrow{\tau} v' \}$
  - $Pre(Q') = Pre\_trans(Q', A) \cup Pre\_taj(Q')$

Characteristics of Timed Automata

- Guards, Transition relations, Invariants, DAEs written in some language
- These objects define the Transitions and Trajectories
- Transitions and trajectories define executions and traces
- Decidability of verification problem will depend on the choice of the language
- Nondeterministic
  - Transition choice
  - Transition relation
  - Branching trajectories
- External interface
  - External actions
  - Further partitioned into I/O actions
  - External variables available in the hybrid I/O automaton model

- Special cases
  - Deterministic HA
  - Rectangular HA
  - (Alur-Dill) Timed Automata
  - X = Finitely many variables with finite types → Finite State Machine with Labeled transitions
  - X = n real valued variables {x1, …, xn} and A = {} D = {} → Dynamical System

Summary & Roadmap

- Hybrid Automata
- Syntax
- Executions
- Reach sets, Invariance
- Abstractions, Simulations and Composition