GLOBAL INITIATIVE OF ACADEMIC NETWORKS

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी • Indian Institute of Technology Guwahati •

University of Illinois at Urbana-Champaign

Lecture 5: Abstractions and Composition

Sayan Mitra

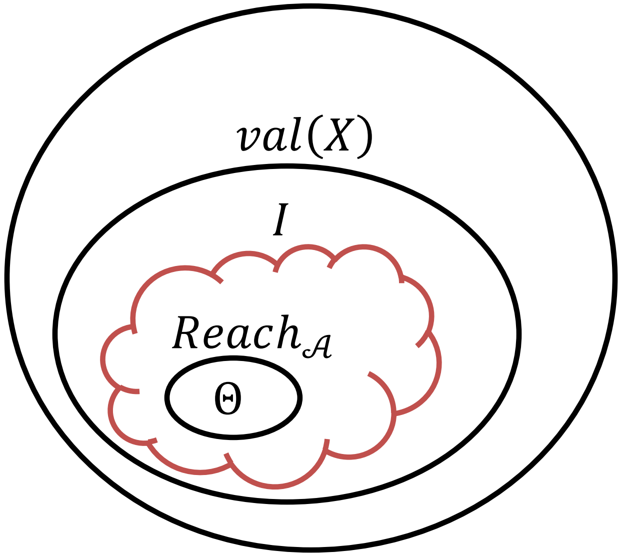
Plan

- Invariants and reachability
  - A simple technique for safety verification
- Abstraction
  - Motivation with discrete examples
  - Forward simulations for hybrid automata
- Composition

Invariants

Hybrid automaton $\mathcal{A} = \langle X, \Theta, A, D, T \rangle$

Recall, $I \subseteq val(X)$ is an invariant if $Reach_{\mathcal{A}} \subseteq I$

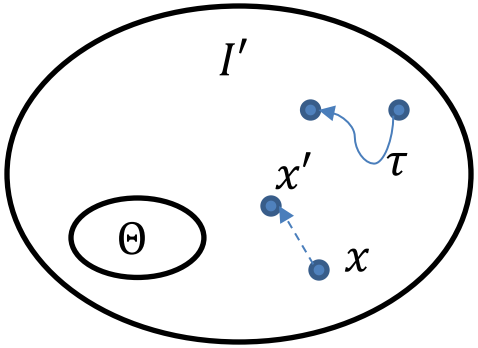


A set $I' \subseteq val(X)$ is inductive iff

(a) $\Theta \subseteq I'$

(b) $\forall \boldsymbol{x} \rightarrow^{a} \boldsymbol{x}', \boldsymbol{x} \in I' \Rightarrow \boldsymbol{x}' \in I'$

(c) $\forall \tau \in \mathrm{T}, \tau.fstate \in I' \Rightarrow \tau.lstate \in I'$



Inductive invariants

Proposition. Inductive sets are invariants.

Proof. Consider an inductive set I', we will show that $Reach_{\mathcal{A}} \subseteq I'$. Pick any $\boldsymbol{x} \subseteq Reach_{\mathcal{A}}$. There exists an closed execution α with $\alpha.lstate = \boldsymbol{x}$. We will do induction on length of α.

Base case. $\alpha = \tau_0$ and τ_0 is a point trajectory. Then by (a) $\tau_0(0) \in \Theta \subseteq I'$.

Inductive case 1. $\alpha = \alpha'\tau'$ and τ' is closed. By inductive hypothesis we know, $\alpha'.lstate = \tau'.fstate \in I'$. From (c), $\tau'.lstate \in I'$.

Inductive case 2. $\alpha = \alpha' a' \tau'$ and τ' is a point traj. By inductive hypothesis we know, $\alpha'.lstate \in I'$. From (b), $\tau'.fstate \in I'$.

Inductive invariants 2

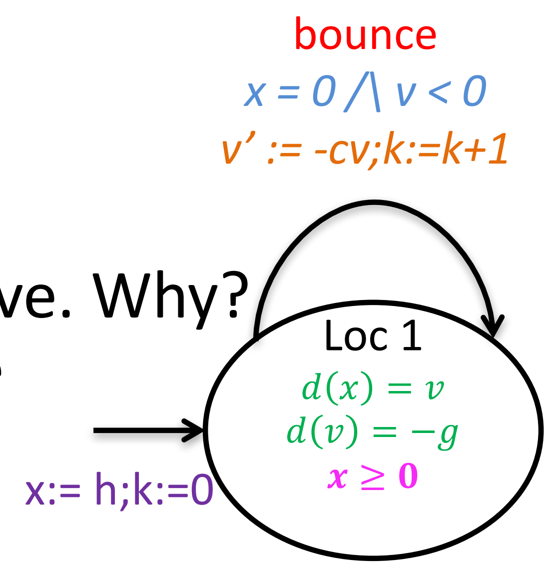
Is $Reach_{\mathcal{A}}$ an invariant? Is it an inductive invariant?

Not all invariants are inductive

Recall bouncing ball (height x)

Is $x \geq 0$ an inductive invariant?

$x \leq hc^{2k}$ is an invariant but not inductive. Why?
$v^2 - \left(Hc^{2k} - x\right)2g = 0$ is an inductive invariant. Check this.



Summary

If I is an inductive invariant of $\mathcal{A}$ and $I \cap U = \emptyset$ the $\mathcal{A}$ is safe w.r.t U.

Checking a given inductive invariant is (relatively) easy recall, (a)$\Theta \subseteq I'$ (b)$\forall \boldsymbol{x} \to^{a} \boldsymbol{x}', \boldsymbol{x} \in I' \Rightarrow \boldsymbol{x}' \in I'$, (c)$\forall \tau \in \mathrm{T}, \tau.fstate \in I' \Rightarrow \tau.lstate \in I'$

Finding inductive invariants not so easy; requires iterative strengthening.

This is parallel to Lyapunov analysis. Checking if a given $V: val(X) \to R$ is a Lyapunov function is relatively easy; finding a Lypunov function is harder.

Abstractions and Simulations

Consider models that have the same external interface (input/output variables and actions)

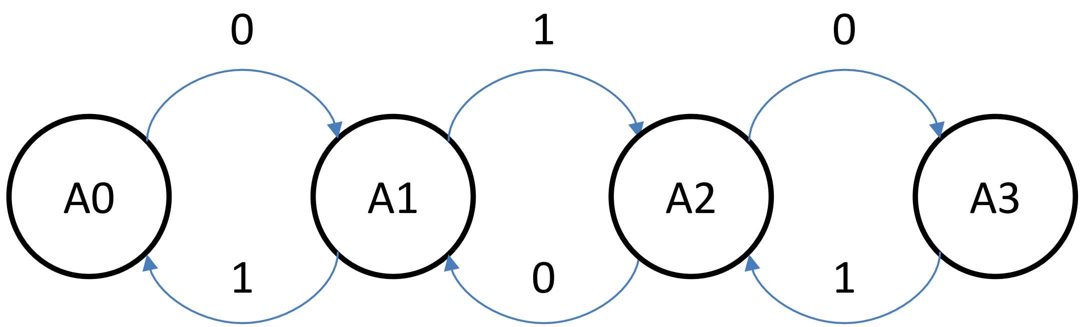
We would like to approximate one (hybrid) automaton H_1 with another one H_2

H_2 should be “simpler” and preserve some properties of H_1 (and not others)

Verifying H_2 we can then infer properties of H_1

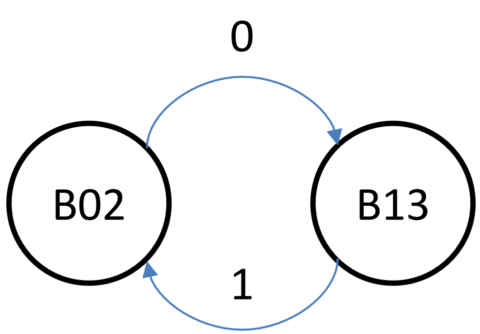
Finite state examples

A



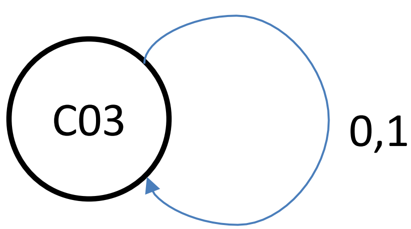
$Traces_A = (01)^*$

B



$Traces_B = 01^*$

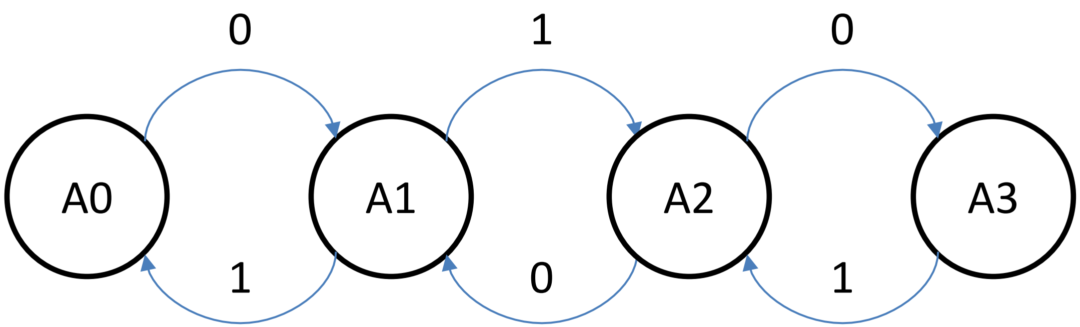
C



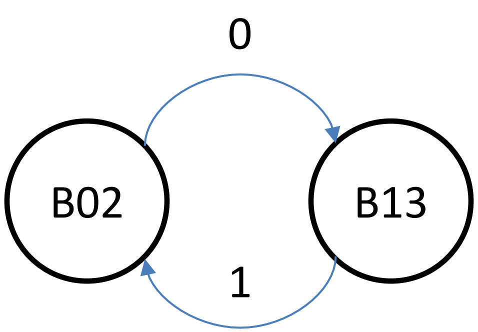
$Traces_C = \{0,1\}^*$

Finite state examples

A

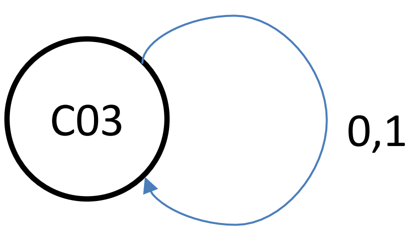


B



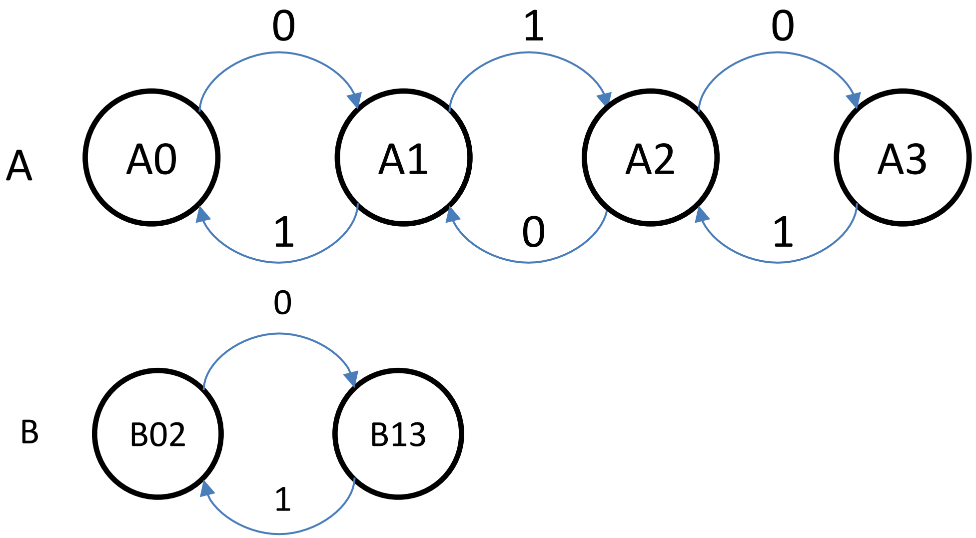
B **simulates** A and vice versa.
A and B are **bisimilar**.

C



C simulates both A and B.
C is an abstraction of both A and B.

How to prove B simulates A?

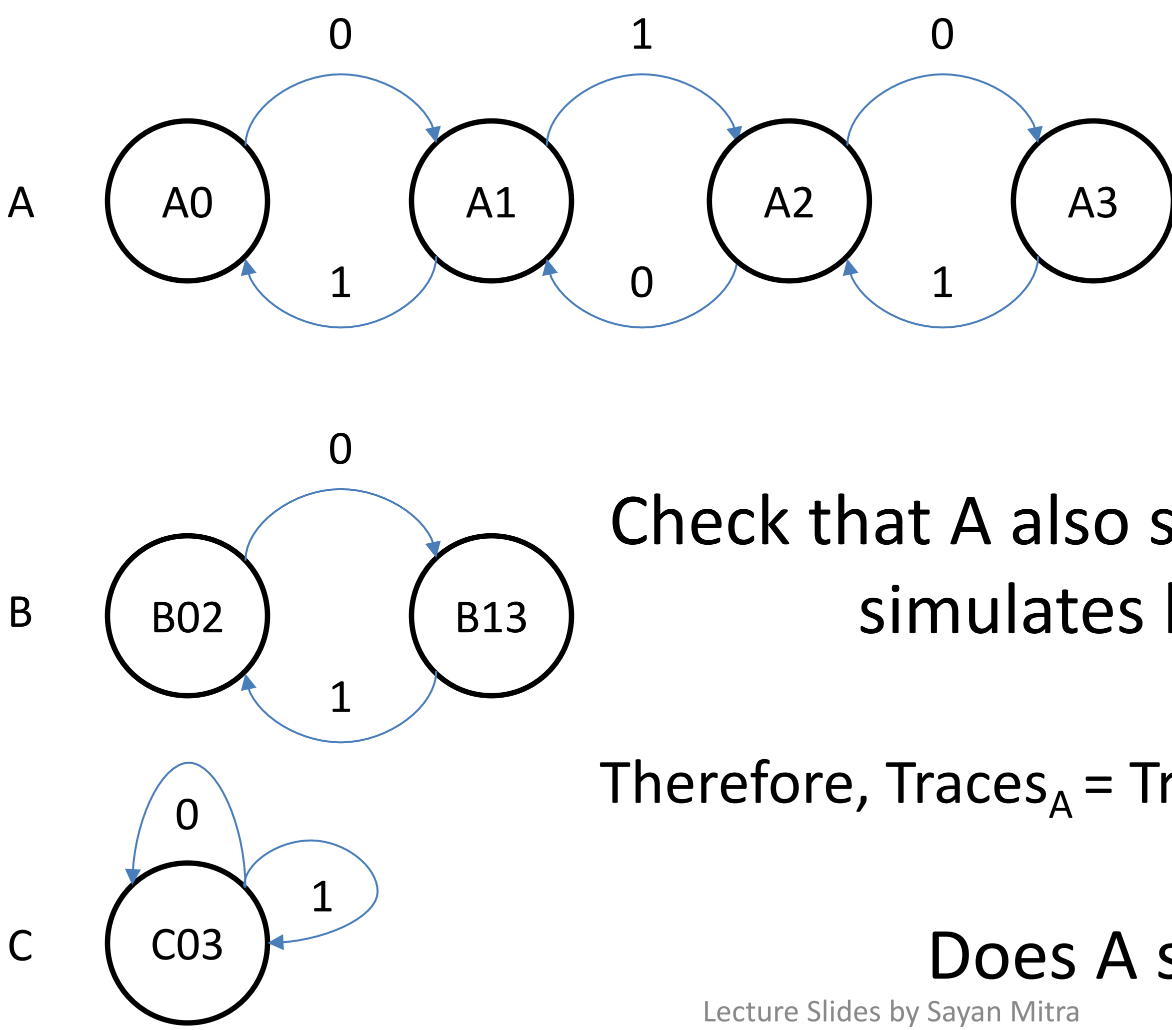


Show there exists a **simulation relation** from states of A to states of B. Say, $R = ((A0, B02), (A2, B02), (A1, B13), (A3, B13))$

Show that for every transition $Ai \rightarrow_A Ai'$ and $(Ai, Bj) \in R$ there exists Bj' such that

1. $Bj \rightarrow_B Bj'$
2. $(Ai', Bj') \in R$
3. $Trace(Bj \rightarrow_B Bj') = Trace(Ai \rightarrow_A Ai')$

Finite state examples

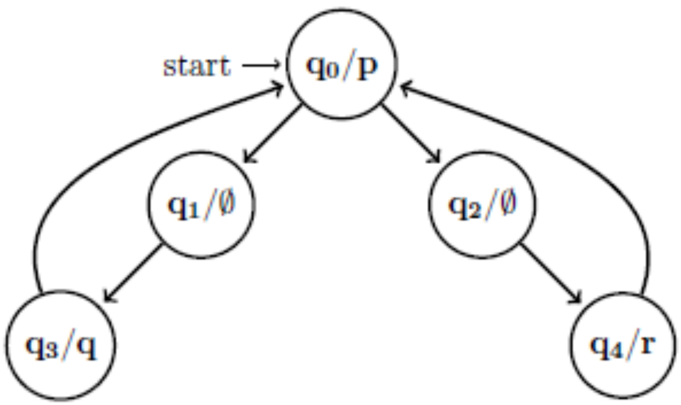


Check that A also simulates B and that C simulates both A and B.

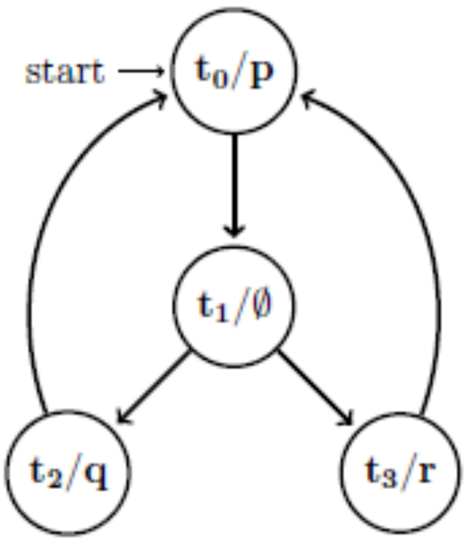
Therefore, $\text{Traces}_A = \text{Traces}_B \subseteq Traces_C$?

Does A simulate C?

Two Machines with identical sets of Traces but are not Bisimilar



(a) State Machine 1

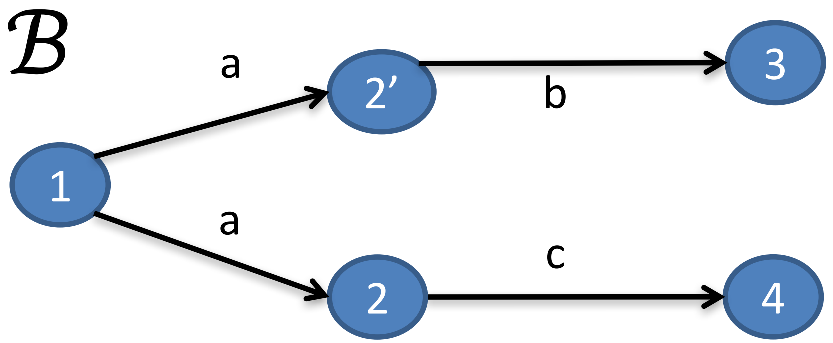


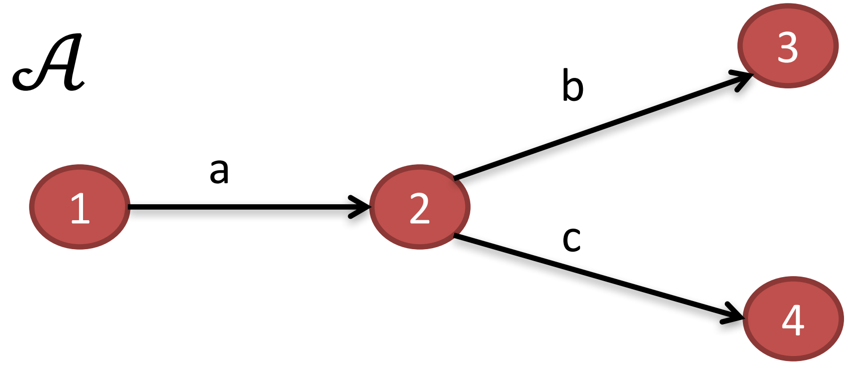
(b) State Machine 2

$$\mathcal{R} = \{\{q_0, t_0\},\{q_1, t_1\}, \{q_2, t_1\}, \{q_3, t_2\}, \{q_4, t_3\}\}.$$

A Simulation Example

- $\mathcal{A}$ is an implementation of $\mathcal{B}$
- Is there a forward simulation from $\mathcal{A}$ to $\mathcal{B}$?
- Consider the forward simulation relation
- $\mathcal{A}$: $2 \rightarrow_c 4$ cannot be simulated by $\mathcal{B}$ from 2' although (2,2') are related.





Simulations (Same actions related states)

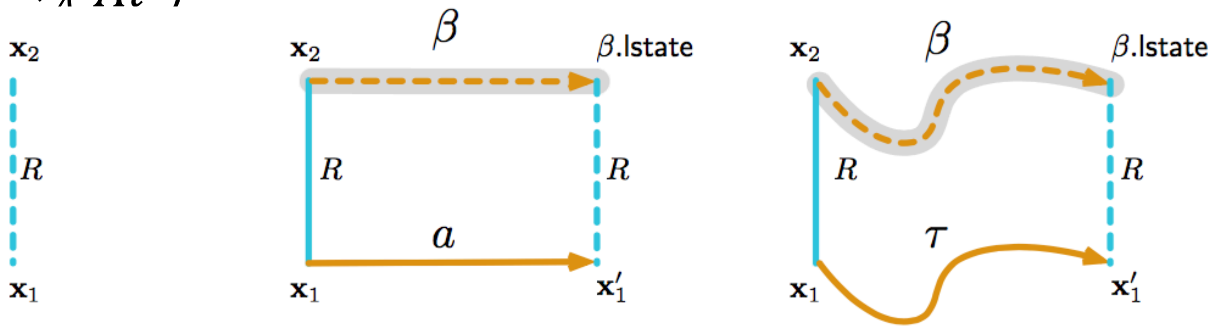
Forward simulation relation from $\mathcal{A}_1$ **to** $\mathcal{A}_2$ is a relation R $\subseteq$ $val(X_1) \times val(X_2)$ such that

1. For every $\mathbf{x_1} \in \Theta_1$ there exists $\mathbf{x_2} \in \Theta_2$ such that $\mathbf{x_1}$ R $\mathbf{x_2}$
2. For every $\mathbf{x_1} \rightarrow_{\boldsymbol{a_1}} \mathbf{x_1}' \in \mathcal{D}$ and $\mathbf{x_2}$ such that $\mathbf{x_1}$ R $\mathbf{x_2}$, there exists $\mathbf{x_2}'$ such that
   - $\mathbf{x_2} \rightarrow_{\boldsymbol{a_1}} \mathbf{x_2}'$ and
   - $\mathbf{x_1}'$ R $\mathbf{x_2}'$
3. For every $\boldsymbol{\tau_1} \in \mathcal{T}_1$ and $\mathbf{x_2}$ such that $\tau_1.fstate$ R $\mathbf{x_2}$, there exists $\tau_2 \in \mathcal{T}_2$ that
   - $\mathbf{x_2} = \tau_2.fstate$ and
   - $\mathbf{x_1}'$ R $\tau_2.lstate$
   - $\tau_2.\mathrm{dom} = \tau_1.dom$

Theorem. If there exists a forward simulation relation from hybrid automaton $\mathcal{A}_1$ to $\mathcal{A}_2$ then for every execution of $\mathcal{A}_1$ there exists a corresponding execution of $\mathcal{A}_2$.

Simulation relations for hybrid automata

- Recall condition 3 in definition of simulation relation: $Trace(Bj \rightarrow_B Bj') = Trace(Ai \rightarrow_A Ai')$



- Hybrid automata have transitions and trajectories
- Different types of simulation depending on different notions for "Trace"
  - Match for all variable values, action names, and time duration of trajectories (abstraction)
  - Match variables but not time (time abstract simulation)
  - Match a subset (external) of variables and actions (trace inclusion)
  - Match single action/trajectory of A with a sequence of actions and trajectories of B

Timer simulates Ball (w.r.t. timing of bounce actions)

Automaton Ball(c,v_0,g)
  variables:
    x: Reals := 0
    v: Reals := v_0
  actions: bounce
    transitions:
      bounce
        pre $x = 0 \wedge v < 0$
        eff v := -cv
    trajectories:
        evolve d(x) = v; d(v) = -g
        invariant $x \geq 0$

Automaton Timer(c, v_0, g)
  variables: analog
  timer: Reals := $2v_0/g$,
  n:Naturals=0;
  actions: bounce
    transitions:
      bounce
        pre *timer = 0*
        eff n:=n+1; timer := $\frac{2v_0}{gc^n}$
    trajectories:
        evolve d(timer) = -1
        invariant timer ≥ 0

Some nice properties of Forward Simulation

Let $\mathcal{A}, \mathcal{B},$ and $\mathcal{C}$ be comparable TAs. If R_1 is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R_2 is a forward simulation from $\mathcal{B}$ to $\mathcal{C}$, then $R_1 \circ R_2$ is a forward simulation from $\mathcal{A}$ to $\mathcal{C}$

$\mathcal{A}$ implements $\mathcal{C}$

The **implementation relation** is a preorder of the set of all (comparable) hybrid automata

(A preorder is a reflexive and transitive relation)

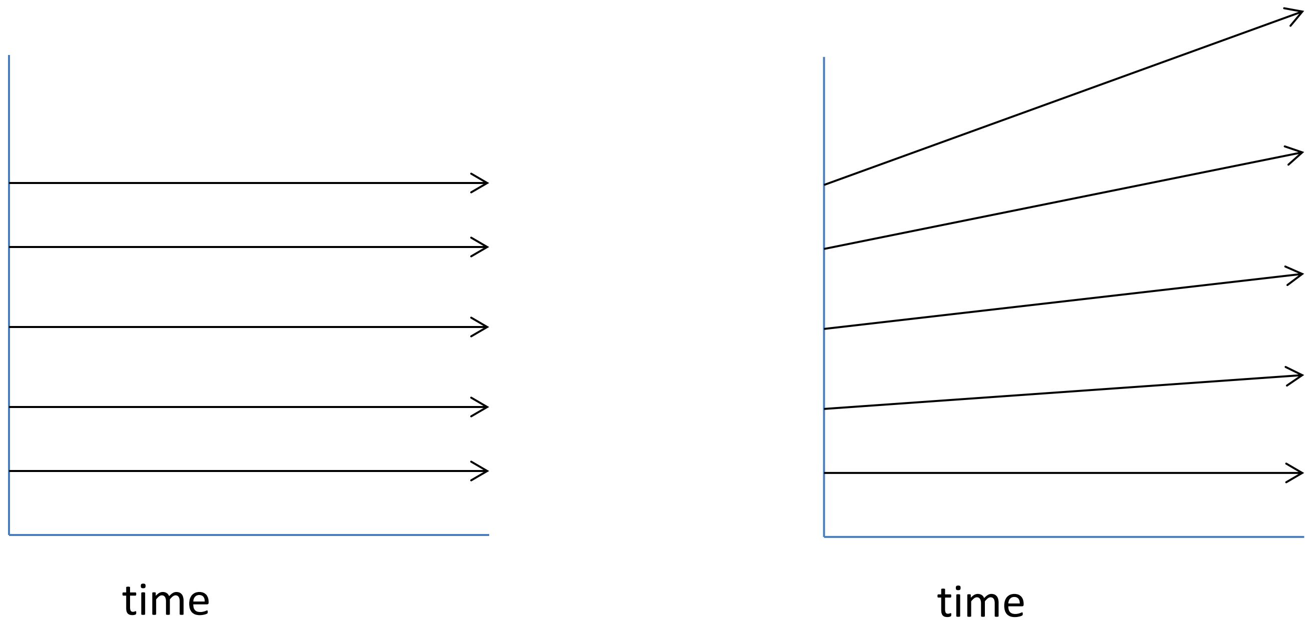
If R is a forward simulation from $\mathcal{A}$ to $\mathcal{B}$ and R^{-1} is a forward simulation from $\mathcal{B}$ to $\mathcal{A}$ then R is called a **bisimulation** and $\mathcal{B}$ are $\mathcal{A}$ **bisimilar**

Bisimilarity is an **equivalence relation**

(reflexive, transitive, and symmetric)

Remark on Simulations and Stability

Stability not preserved by ordinary simulations and bisimulations [Prabhakar, et. al 15]



Stability Preserving Simulations and Bisimulations for Hybrid Systems, Prabhakar, Dullerud, Viswanathan IEEE Trans. Automatic Control 2015

Backward Simulations

Backward simulation relation from $\mathcal{A}_1$ **to** $\mathcal{A}_2$ is a relation R $\subseteq$ $Q_1 \times Q_2$ such that

1. If $\mathbf{x_1} \in \Theta_1$ and $\mathbf{x_1}$ R $\mathbf{x_2}$ then $\mathbf{x_2} \in \Theta_2$ such that
2. If $\mathbf{x'_1}$ R $\mathbf{x'_2}$ and x_1—a→ x_1' then
   - $\mathbf{x_2} \mathbf{-\beta\rightarrow} \mathbf{x_2'}$ and
   - $\mathbf{x_1}$ R $\mathbf{x_2}$
   - **Trace($\boldsymbol{\beta}$)** = $\mathbf{a_1}$
3. For every $\boldsymbol{\tau} \in \mathcal{T}$ and $\mathbf{x_2} \in Q_2$ such that $\mathbf{x_1'}$ R $\mathbf{x_2'}$, there exists $\mathbf{x_2}$ such that
   - $\mathbf{x_2} \mathbf{-\beta\rightarrow} \mathbf{x_2'}$ and
   - $\mathbf{x_1}$ R $\mathbf{x_2}$
   - **Trace($\boldsymbol{\beta}$)** = $\boldsymbol{\tau}$

Theorem. If there exists a backward simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$ then ClosedTraces$_1 \subseteq$ ClosedTraces$_2$

COMPOSITION

Lecture Slides by Sayan Mitra
mitras@illinois.edu

Composition of Hybrid Automata

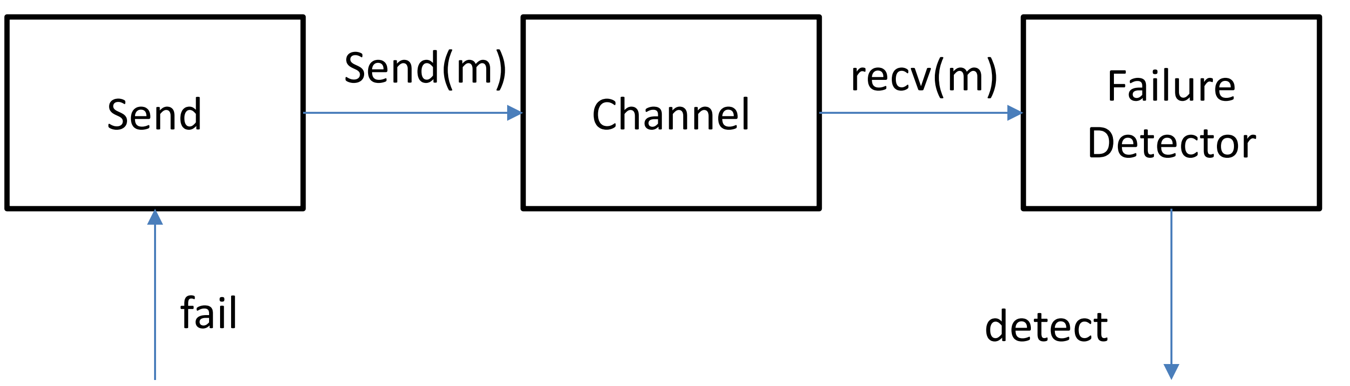
The parallel **composition** operation on automata enable us to construct larger and more complex models from simpler automata modules

$\mathcal{A}_1$ to $\mathcal{A}_2$ are **compatible** if $\mathrm{X}_1 \cap \mathrm{X}_2 = \mathrm{H}_1 \cap \mathrm{A}_2 = \mathrm{H}_2 \cap \mathrm{A}_1 = \emptyset$

Variable names are disjoint; Action names of one are disjoint with the internal action names of the other

Modeling a Simple Failure Detector System

- Periodic send
- Channel
- Timeout



Composition

- For compatible $\mathcal{A}_1$ and $\mathcal{A}_2$ their composition $\mathcal{A}_1 \mid\mid \mathcal{A}_2$ is the structure $\boldsymbol{\mathcal{A}}$= $(X, Q, \Theta, E, H, \mathcal{D}, \mathcal{T})$
- $X = X_1 \cup X_2$ (disjoint union)
- $Q \subseteq val(X)$
- $\Theta = \{ \boldsymbol{x} \in Q \mid \forall\, i \in \{1,2\}\text{: } \boldsymbol{x}.Xi \in \Theta_i\}$
- $H = H_1 \cup H_2$ *(disjoint union)*
- $E = E_1 \cup E_2$ *and* A= E ∪ H
- $(\boldsymbol{x}, a, \boldsymbol{x}') \in \mathcal{D}$ *iff*
  - $a \in H_1$ *and* $(\boldsymbol{x}.X_1, a, \boldsymbol{x}'.X_1) \in \mathcal{D}_1$ *and* $\boldsymbol{x}.X_2 = \boldsymbol{x}.X_2$
  - $a \in H_2$ *and* $(\boldsymbol{x}.X_2, a, \boldsymbol{x}'.X_2) \in \mathcal{D}_2$ *and* $\boldsymbol{x}.X_1 = \boldsymbol{x}.X_1$
  - *Else,* $(\boldsymbol{x}.X_1, a, \boldsymbol{x}'.X_1) \in \mathcal{D}_1$ *and* $(\boldsymbol{x}.X_2, a, \boldsymbol{x}'.X_2) \in \mathcal{D}_2$
- $\mathcal{T}$*:* set of **trajectories** for X
  - $\tau \in \mathcal{T}$ iff $\forall\, i \in \{1,2\}$, $\tau.Xi \in \mathcal{T}_i$

Theorem . $\boldsymbol{\mathcal{A}}$ is also a hybrid automaton.

Example: Send || TimedChannel

Automaton Channel(b,M)
variables: internal
queue: **Queue**[M,Reals] := {}
clock1: Reals := 0
actions: external send(m:M), receive(m:M)
transitions:
send(m)
pre true
eff queue := append(<m, clock1+b>, queue)
receive(m)
pre $\text{head}(\text{queue})[1] = m$
eff queue := queue.tail
trajectories:
evolve d(clock1) = 1
stop when $\exists$ m, d, <m,d> ∈ queue
/\ clock=d

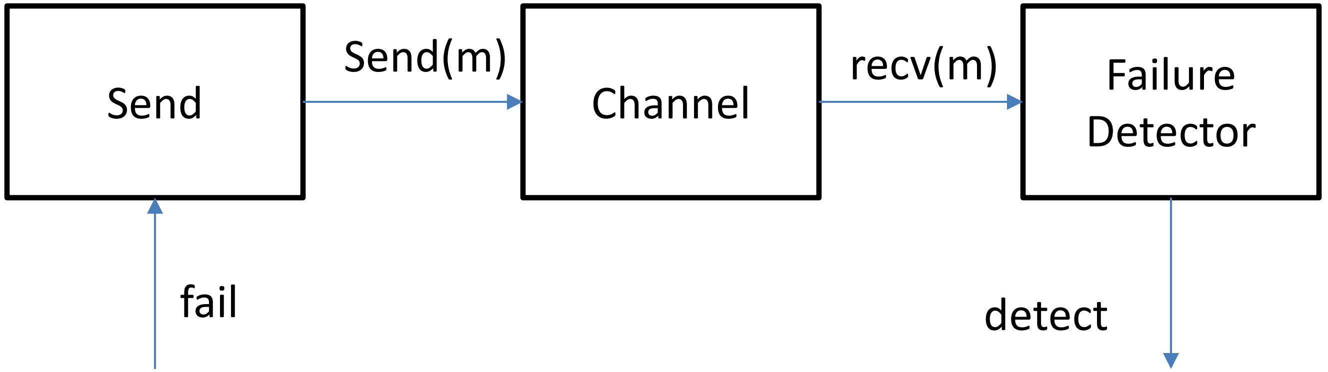
Automaton PeriodicSend(u, M)
variables: internal clock: Reals := 0
actions: external send(m:M)
transitions:
send(m)
pre clock = u
eff clock := 0
trajectories:
evolve d(clock) = 1
stop when clock=u

Composed Automaton

Automaton SC(b,u)
  variables: internal queue: **Queue**[M,Reals] := {}
      clock_s, clock_c: Reals := 0
  actions: external send(m:M), receive(m:M)
  transitions:
    send(m)
    pre clock_s = u
    eff queue := append(<m, clock_c+b>, queue); clock_s := 0
    receive(m)
    pre head(queue)[1] = m
    eff queue := queue.tail
  trajectories:
    evolve d(clock_c) = 1; d(clock_s) = 1
    stop when
      (∃ m, d, <m,d> ∈ queue /\ clock_c=d)
      \/ (clock_s=u)

Modeling a Simple Failure Detector System

- Periodic send || Channel
- Periodic send || Channel || Timeout

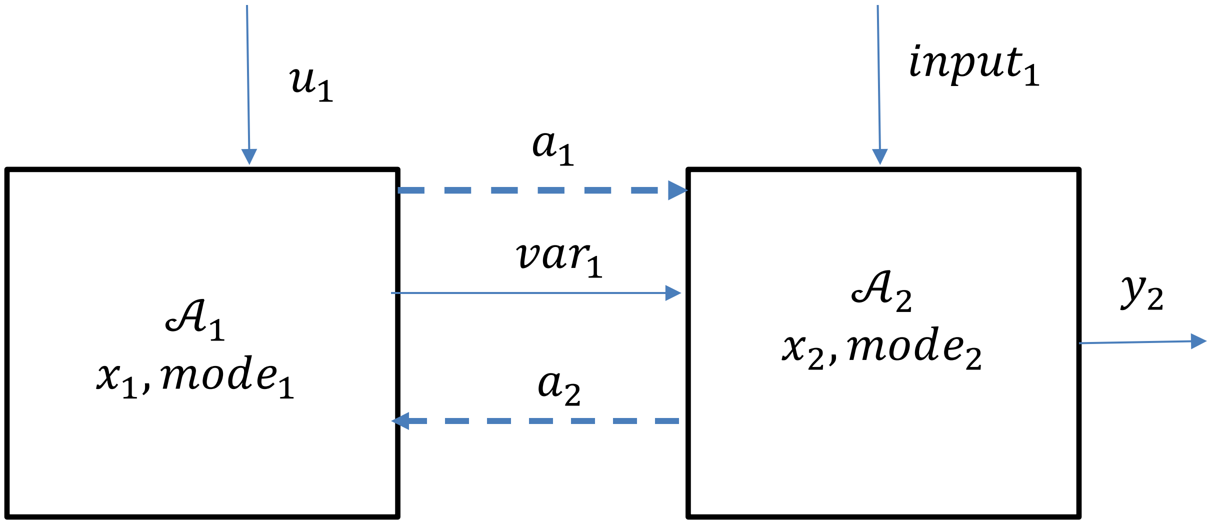


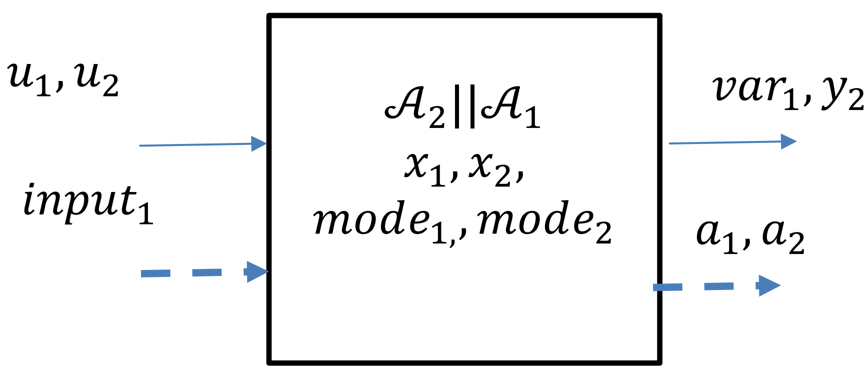
Time bounded channel & Simple Failure Detector

Automaton Timeout(u,M)
  variables: internal 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
  trajectories:
    evolve d(clock) = 1
    stop when clock = u /\ ~suspected

Lecture Slides by Sayan Mitra
mitras@illinois.edu

General composition





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

$(1-\rho) \leq d(\text{clock}$

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

Automaton Channel(b,M)

deadline

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 ∃ m, d, <m,d> ∈ queue /\ clock=d

Some properties about composed automata

- Let $\boldsymbol{\mathcal{A}}= \mathcal{A}_1 \; || \; \mathcal{A}_2$ and let α be an execution fragment of $\boldsymbol{\mathcal{A}}$.
  - Then $\alpha_i = \alpha|(A_i, X_i)$ is an execution fragment of $\mathcal{A}_i$
  - α is time-bounded iff both α_1 and α_2 are time-bounded
  - α is admissible iff both α_1 and α_2 are admissible
  - α is closed iff both α_1 and α_2 are closed
  - α is non-Zeno iff both α_1 and α_2 are non-Zeno
  - α is an execution iff both α_1 and α_2 are executions
- $\text{Traces}_{\boldsymbol{\mathcal{A}}} = \{ \boldsymbol{\beta} \mid \quad \boldsymbol{\beta}| E_i \in \text{Traces}\, \mathcal{A}_i \}$
- See examples in the TIOA monograph

Substitutivity

- **Theorem.** Suppose $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{B}$ have the same external interface and $\mathcal{A}_1$, $\mathcal{A}_2$ are compatible with $\mathcal{B}$. If $\mathcal{A}_1$ implemens $\mathcal{A}_2$ then $\mathcal{A}_1 || \mathcal{B}$ implements $\mathcal{A}_2 || \mathcal{B}$
- Proof sketch.
- Define the simulation relation:

Substutivity

- **Theorem.** Suppose $\mathcal{A}_1$ $\mathcal{A}_2$ $\mathcal{B}_1$ and $\mathcal{B}_2$ are HAs and $\mathcal{A}_1$ $\mathcal{A}_2$ have the same external actions and $\mathcal{B}_1$ $\mathcal{B}_2$ have the same external actions and $\mathcal{A}_1$ $\mathcal{A}_2$ is compatible with each of $\mathcal{B}_1$ and $\mathcal{B}_2$
- If $\mathcal{A}_1$ implements $\mathcal{A}_2$ and $\mathcal{B}_1$ implements $\mathcal{B}_2$ then $\mathcal{A}_1 \,||\, \mathcal{B}_1$ implements $\mathcal{A}_2||\mathcal{B}_2$.

- Proof. $\mathcal{A}_1 \,||\, \mathcal{B}_1$ implements $\mathcal{A}_2||\mathcal{B}_1$

  $\mathcal{A}_2||\mathcal{B}_1$ implements $\mathcal{A}_2||\mathcal{B}_2$

  By transitivity of implementation relation

  $\mathcal{A}_1 \,||\, \mathcal{B}_1$ implements $\mathcal{A}_2||\mathcal{B}_2$

- **Theorem.** $\mathcal{A}_1 \mid\mid \mathcal{B}_2$ implements $\mathcal{A}_2 || \mathcal{B}_2$ and $\mathcal{B}_1$ implements $\mathcal{B}_2$ then $\mathcal{A}_1 \mid\mid \mathcal{B}_1$ implements $\mathcal{A}_2 || \mathcal{B}_2$.

Summary

- Implementation Relation
  - Forward and Backward simulations
- Composition
- Substitutivity