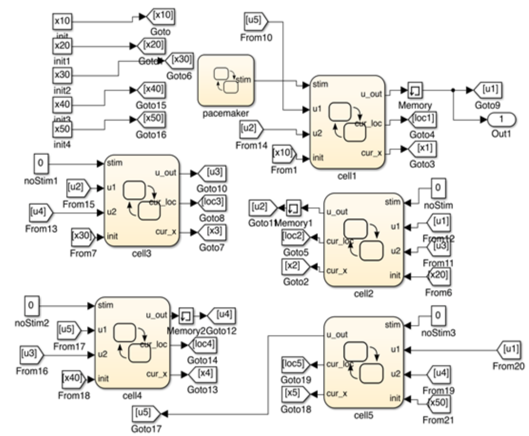
Composition

Sayan Mitra
Verifying cyberphysical systems
mitras@illinois.edu

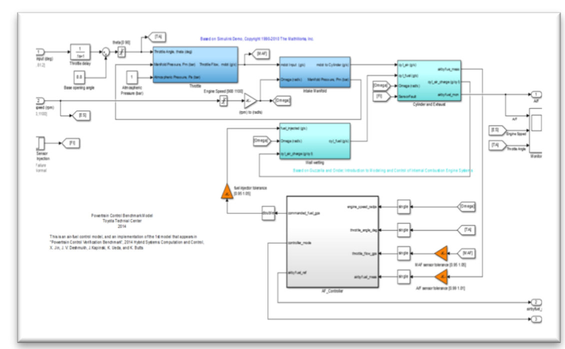
- HW3 out
- Read chapter 5

What is composition?

- Complex models and systems are built by putting together **components** or **modules**
- **Composition** is the mathematical operation of *putting together*
- Leads to precise definition of module **interfaces**
- What properties are preserved under composition?



model of a network of oscillators [Huang et. al 14]



Powertrain model from Toyota [Jin et al. 15]

- Give an example of how you’ve built something more complex from simple components
- Throughout the lecture, think if your notion of composition is captured by what we define

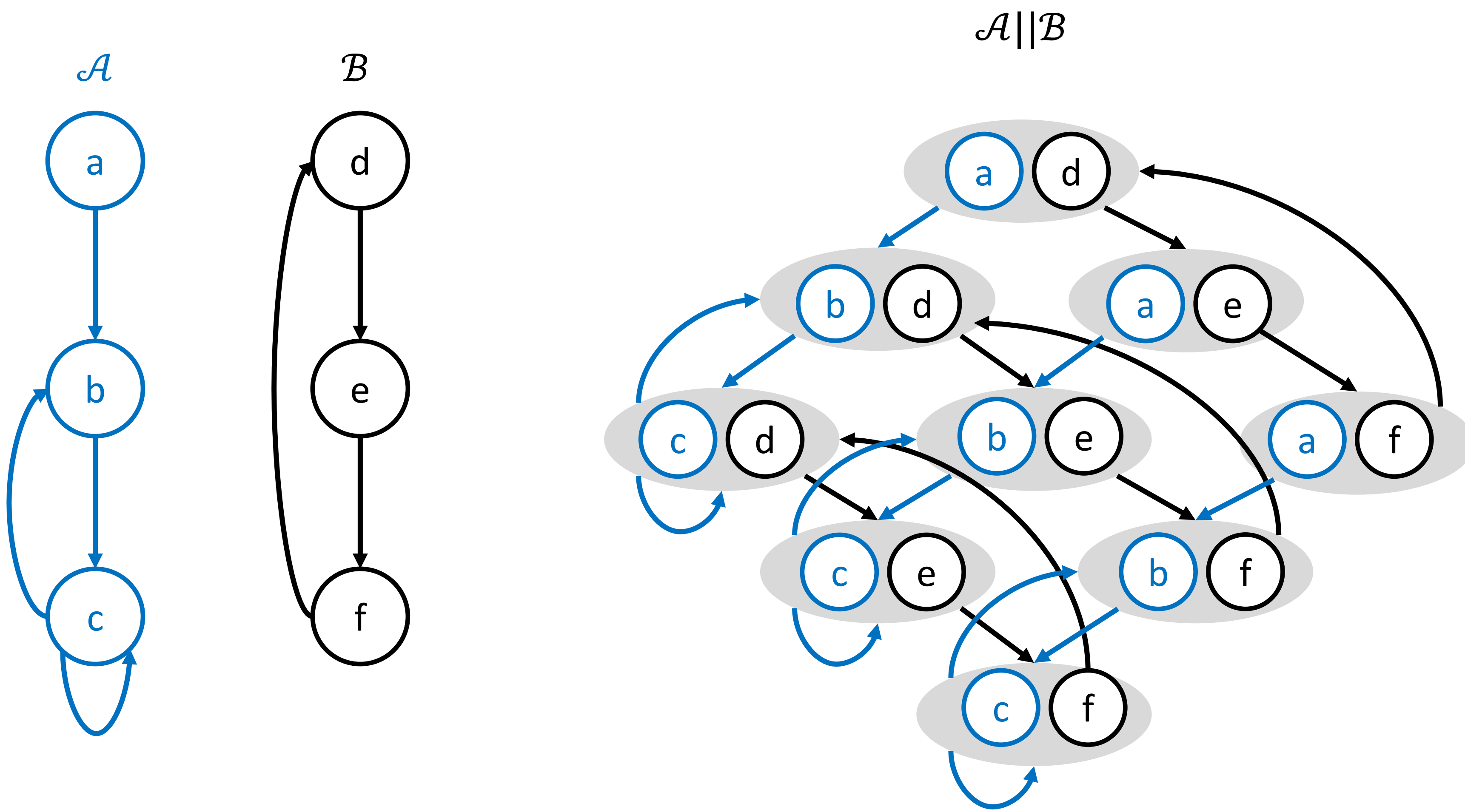
Outline

- Composition operation
  - Input/output interfaces
- I/O automata
- hybrid I/O automata
- Examples
- Properties of composition

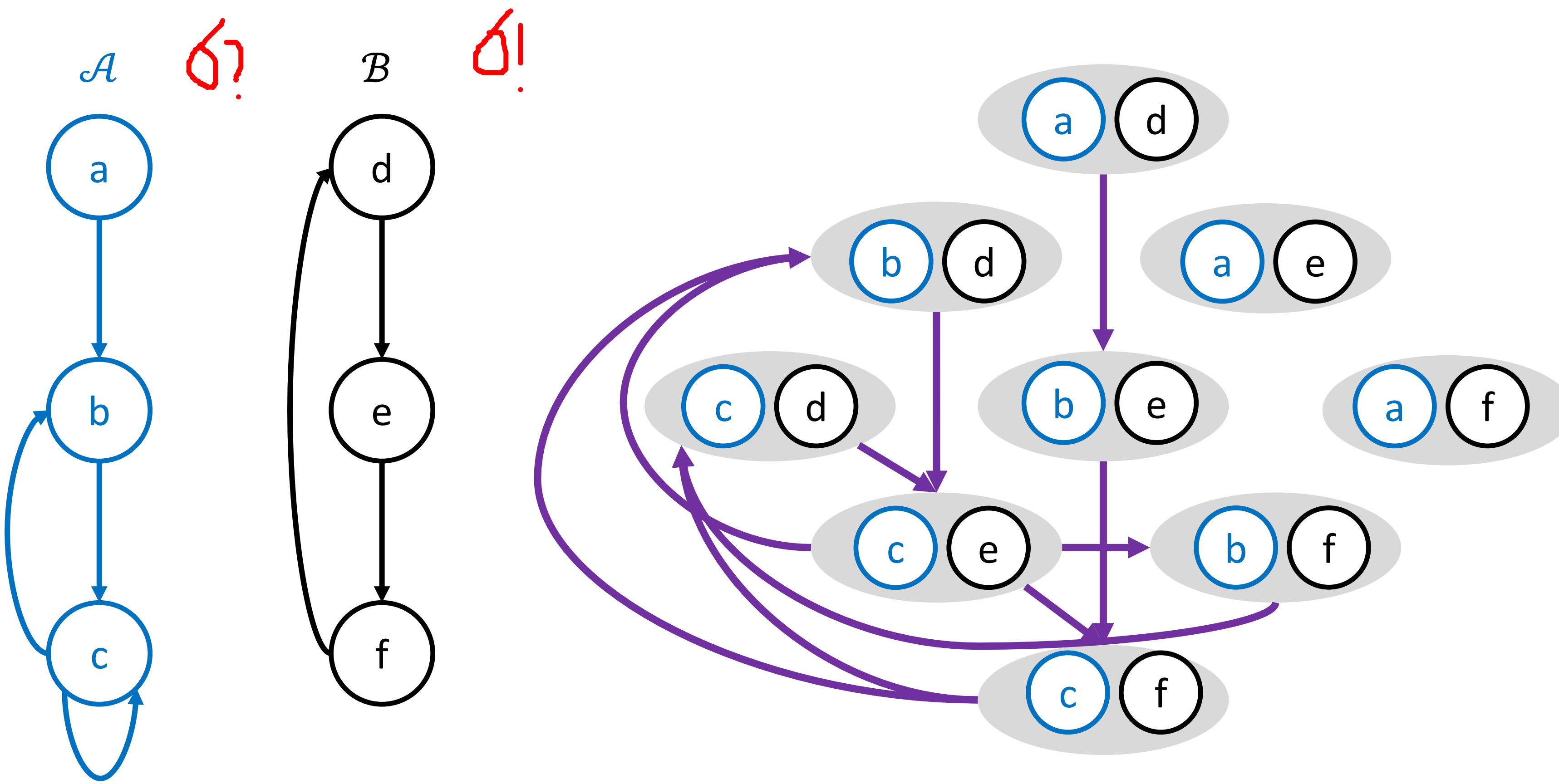
Composition of automata

- Complex systems are built by "putting together" simpler subsystems
- Recall $\mathcal{A} = \langle X, \Theta, A, \boldsymbol{D} \rangle$
- $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$
  - $\mathcal{A}_1, \mathcal{A}_2$ are the *component automata* and
  - $\mathcal{A}$ is the *composed* automaton
  - $||$ symbol for the composition operator

Composition: asynchronous modules



composition: modules synchronize



Composition of (discrete) automata

- More generally, some transitions of $\mathcal{A}$ and $\mathcal{B}$ may synchronize, while others may not synchronize
- Further, some transitions may be **controlled** by $\mathcal{A}$ which when occurs **forces** the corresponding transition of $\mathcal{B}$
- Thus, we will partition the set of actions A of $\mathcal{A} = \langle X, \Theta, A, \boldsymbol{D} \rangle$ into
  - H: **internal** (do not synchronize)
  - O: **output** (synchronized and controlled by $\mathcal{A}$)
  - I: **input** (synchronized and controlled by some other automaton)
- $A = H \cup O \cup I$
- This gives rise to **I/O automata** **[Lynch, Tuttle 1996]**

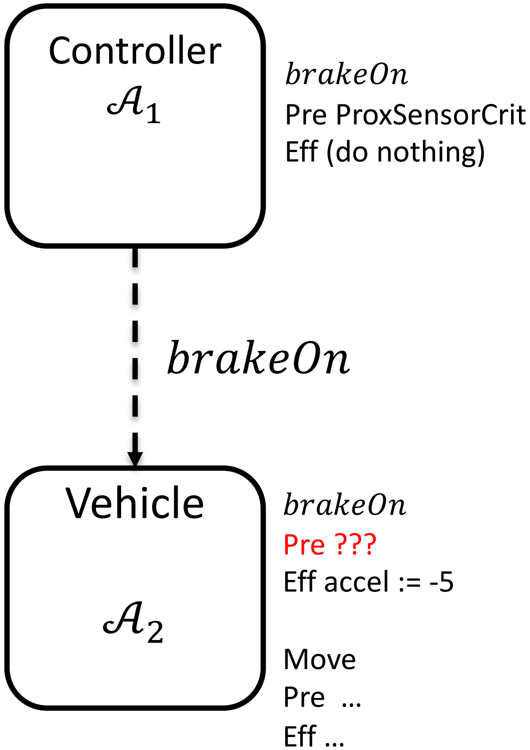
Reactivity: Input enabling

- Consider a shared action **brakeOn** controlled by $\mathcal{A}_1$ and listend-to or read by $\mathcal{A}_2$
- Input enabling ensures that when $\mathcal{A}_1$ and $\mathcal{A}_2$ are composed then $\mathcal{A}_2$ can react to **brakeOn**

Definition. An **input/output automaton** is a tuple $\mathcal{A} = \langle X, \Theta, A, \mathcal{D} \rangle$ where

- X is a set of names of variables
- $\Theta \subseteq val(X)$ is the set of initial states
- $A = I \cup O \cup H$ is a set of names of actions
- $\mathcal{D} \subseteq val(X) \times A \times val(X)$ is the set of transitions and $\mathcal{A}$ satisfies the input enabling condition:

E1. For each $\boldsymbol{x} \in val(X), a \in I$ there exists $\boldsymbol{x}' \in val(X)$ such that $\boldsymbol{x} \rightarrow_a \boldsymbol{x}'$



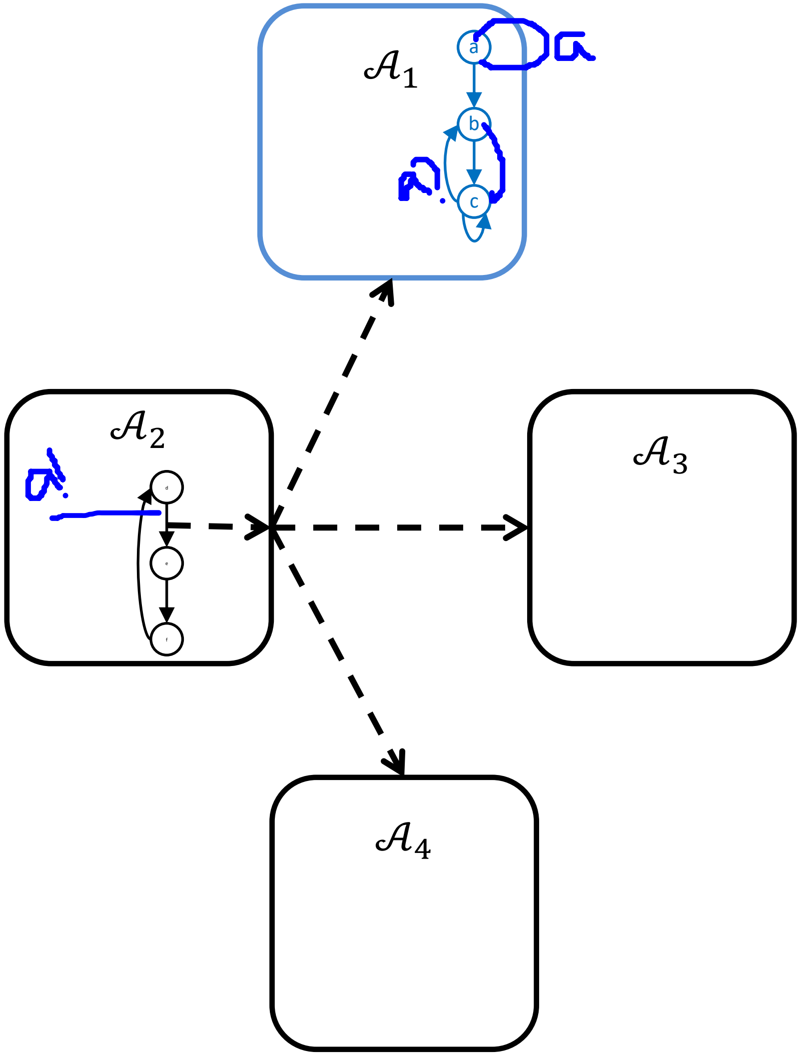
Compatibility IOA

A pair of I/O automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible if

$H_i \cap A_j = \emptyset$ no unintended interactions

$O_i \cap O_j = \emptyset$ no duplication of authority

Extended to collection of automata in the natural way



Composition of I/O automaton

Definition. For compatible automata $\mathcal{A}_1$ and $\mathcal{A}_2$ their composition $\mathcal{A}_1 \,||\, \mathcal{A}_2$ is the structure $\boldsymbol{\mathcal{A}} = (X, \Theta, A, \mathcal{D})$

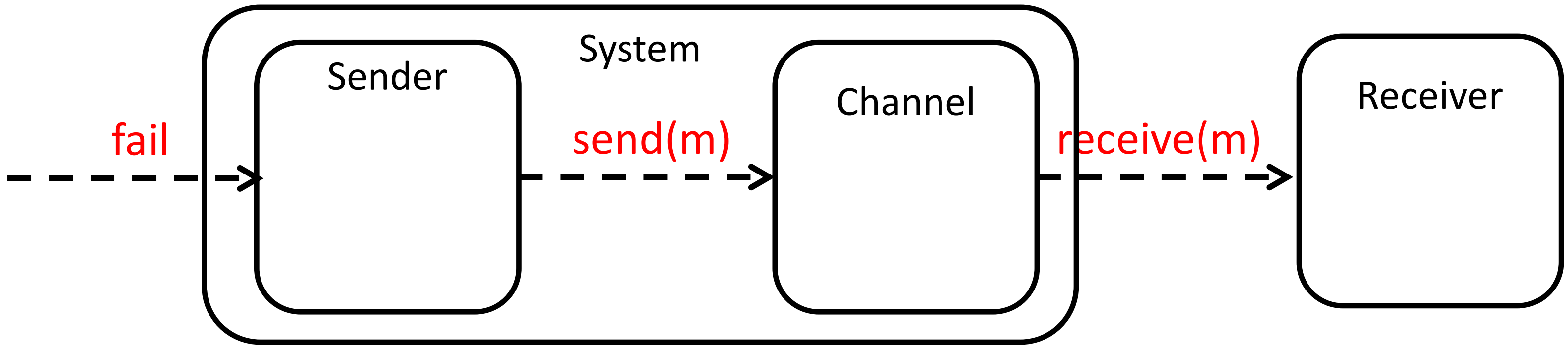
- $X = X_1 \cup X_2$
- $\Theta = \{ \boldsymbol{x} \in val(X) | \; \forall\, i \; \in \{1,2\}\colon \; \boldsymbol{x}\lceil X_{\boldsymbol{i}} \in \Theta_i\}$
- $H = H_1 \; \cup H_2$
- $\mathrm{O} = O_1 \; \cup O_2$ $\qquad\qquad \}\;\; A = H \cup O \cup I$
- $\mathrm{I} = I_1 \; \cup I_2 \setminus O$
- $(\boldsymbol{x}, a, \boldsymbol{x}') \in \; \mathcal{D}$ *iff for* $i \in \{1,2\}$
    - $a \in A_i$ *and* $(\boldsymbol{x}\lceil X_i, a, \boldsymbol{x}'\lceil X) \in \; \mathcal{D}_i$
    - $a \notin A_i \; \boldsymbol{x}\lceil X_i = \boldsymbol{x}\lceil X_i$

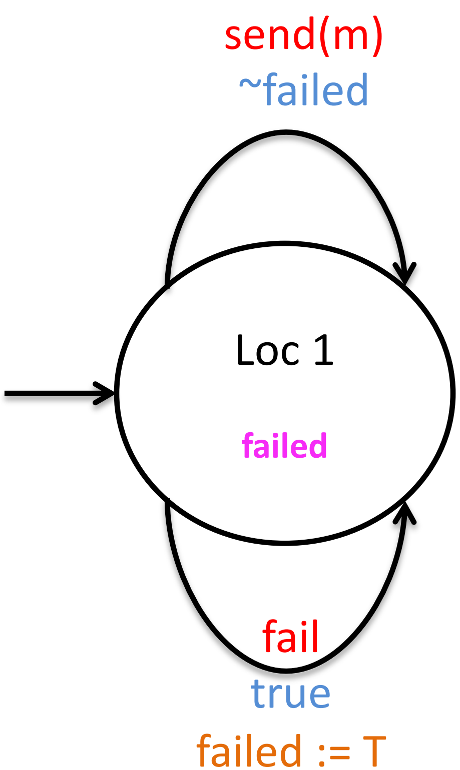
Theorem. The class of IO-automata is closed under composition. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible I/O automata then $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$ is also an I/O automaton.

Proof. Only 2 things to check

- Input, output, and internal actions are disjoint---by construction
- $\mathcal{A}$ satisfies **E1**. Consider any state $\boldsymbol{x} \in val(X_1 \cup X_2)$ and any input action a $\in I_1 \cup I_2 \setminus O$ such that a is enabled in $\boldsymbol{x}$.
- Suppose, w.lo.g. a $\in I_1$
- We know by **E1** of $\mathcal{A}_1$ that there exists $\boldsymbol{x_1'} \in val(X_1)$ such that $\boldsymbol{x}\lceil X_1 \rightarrow_a \boldsymbol{x_1'}$
- $a \notin O_2, I_2, H_2$ (by compatibility)
- Therefore, $\boldsymbol{x} \rightarrow_a (\boldsymbol{x_1'}, x\lceil X_2)$ is a valid transition of $\mathcal{A}$ (by definition of composition)

Example: Sending process and channel





Automaton Sender(u)
  variables internal
    failed:Boolean := F
  output send(m:M)
  input fail
  transitions:
    output send(m)
      pre ~failed
      eff
    input fail
      pre true
      eff failed := T

FIFO channel & Simple Failure Detector

Automaton Sender(u)
variables internal
failed:Boolean := F
output send(m:M)
input fail
transitions:
output send(m)
pre ~failed
eff
input fail
pre true
eff failed := T

Automaton Channel(M)
variables internal queue: **Queue**[M] := {}
actions input send(m:M)
output receive(m:M)
transitions:
input send(m)
pre true
eff queue := append(m, queue)
output receive(m)
pre head(queue)=m
eff queue := queue.tail

Automaton System(M)
variables queue: **Queue**[M] := {}, failed: Bool
actions input fail
output send(m:M), receive(m:M)
transitions:
output send(m)
pre ~failed
eff queue := append(m, queue)
output receive(m)
pre head(queue)=m
eff queue := queue.tail
input fail
pre true
eff failed := true

COMPOSING HYBRID SYSTEMS

Hybrid IO Automaton

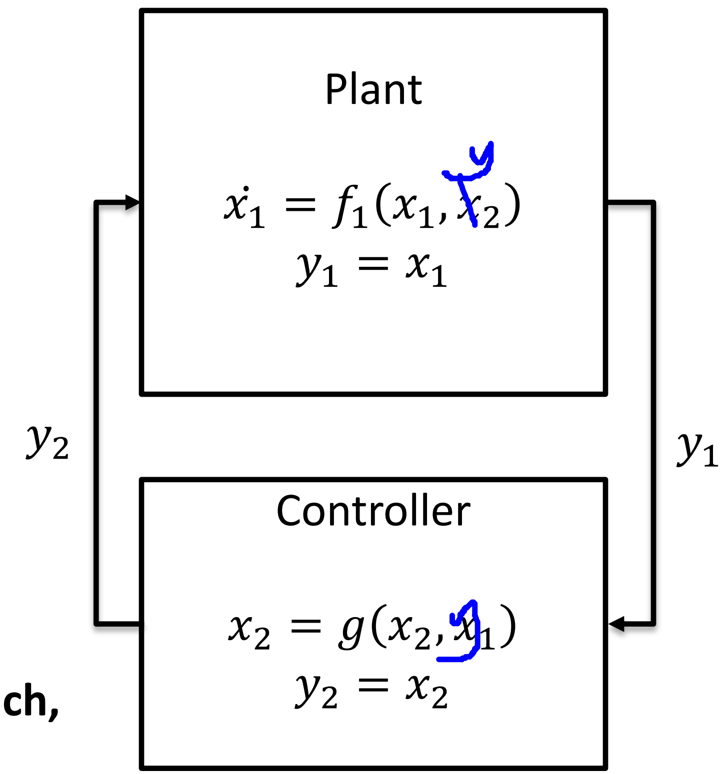
In addition to interaction through shared actions hybrid input/output automata (HIOA) will allow interaction through shared variables

Recall a hybrid automaton $\mathcal{A} = \langle V, \Theta, A, \boldsymbol{D}, \boldsymbol{T} \rangle$

We will partition the set of variables V of $\mathcal{A}$ into

- X: **internal or state variables** (do not interact)
- Y: **output** variables
- U: **input** variables

- $V = X \cup Y \cup U$



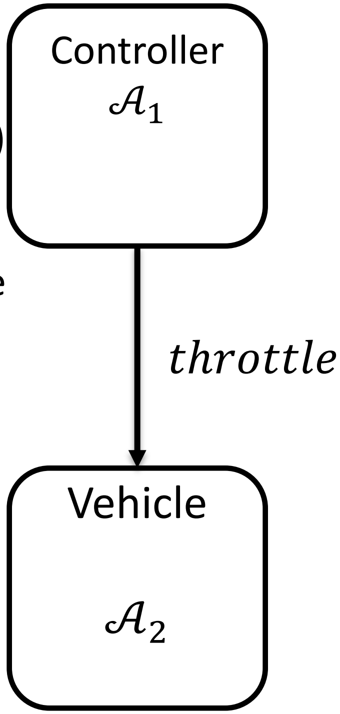
This gives rise to **hybrid I/O automata (HIOA) [Lynch, Segala, Vaandrager 2002]**

Reactivity: Input trajectory enabling

Consider a shared variable **throttle** controlled by $\mathcal{A}_1$ and listened-to or read by $\mathcal{A}_2$

Input trajectory enabling ensures that when $\mathcal{A}_1$ and $\mathcal{A}_2$ are composed then $\mathcal{A}_2$ can react to any signal generated by $\mathcal{A}_1$

If the trajectories of $\mathcal{A}_2$ are defined by ordinary differential equations, then input enabling is guaranteed if $\mathcal{A}_1$ only generates piece-wise continuous signals (throttle)



Definition. An **hybrid input/output automaton** is a tuple $\mathcal{A} = \langle V, \Theta, A, \boldsymbol{\mathcal{D}}, \boldsymbol{T} \rangle$ where

- $V = X \cup U \cup Y$ is a set of variables
- $\Theta \subseteq val(X)$ is the set of initial states
- $A = I \cup O \cup H$ is a set of actions
- $\mathcal{D} \subseteq val(X) \times A \times val(X)$ is the set of transitions
- $\boldsymbol{T}$ is a set of trajectories for V closed under prefix, suffix, and concatenation

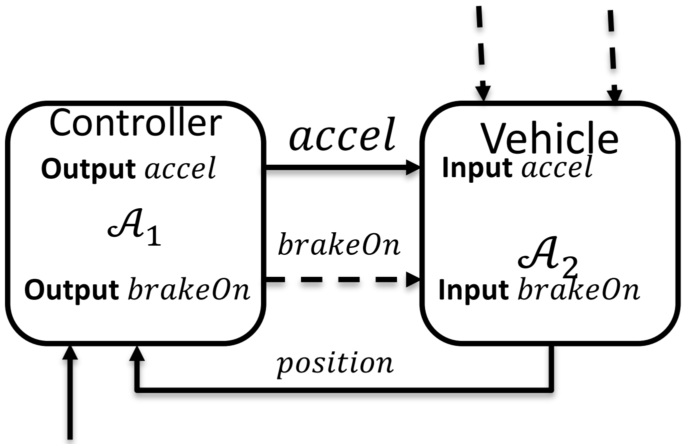
E1. For each $\boldsymbol{x} \in val(X), a \in I$ there exists $\boldsymbol{x}' \in val(X)$ such that $\boldsymbol{x} \rightarrow_a \boldsymbol{x}'$

E2. For each $\boldsymbol{x} \in val(X), \mathcal{A}$ should be able to react to any trajectory η of U.

i.e, $\exists\, \tau \in \boldsymbol{T}$ with $\tau.fstate = \boldsymbol{x}$ such that $\tau \downarrow U$ is a prefix of η and either (a) $\tau \downarrow U = \eta$ or (b) τ is closed and some $a \in H \cup O$ is enabled at $\tau.lstate$.

Compatibility of hybrid automata

- For the interaction of hybrid automata $\mathcal{A}_1$ and $\mathcal{A}_2$ to be well-defined we need to ensure that they have the right *interfaces*



- compatibility conditions

Compatibility HIOA

A pair of hybrid I/O automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible if

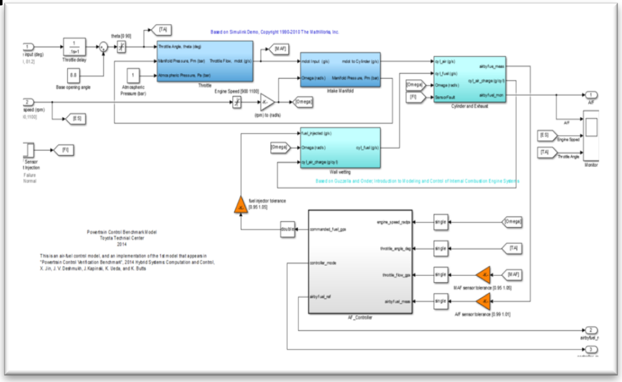
$H_i \cap A_j = \emptyset$ no unintended discrete interactions

$O_i \cap O_j = \emptyset$ no duplication of discrete authority

$X_i \cap V_j = \emptyset$ no unintended continuous interactions

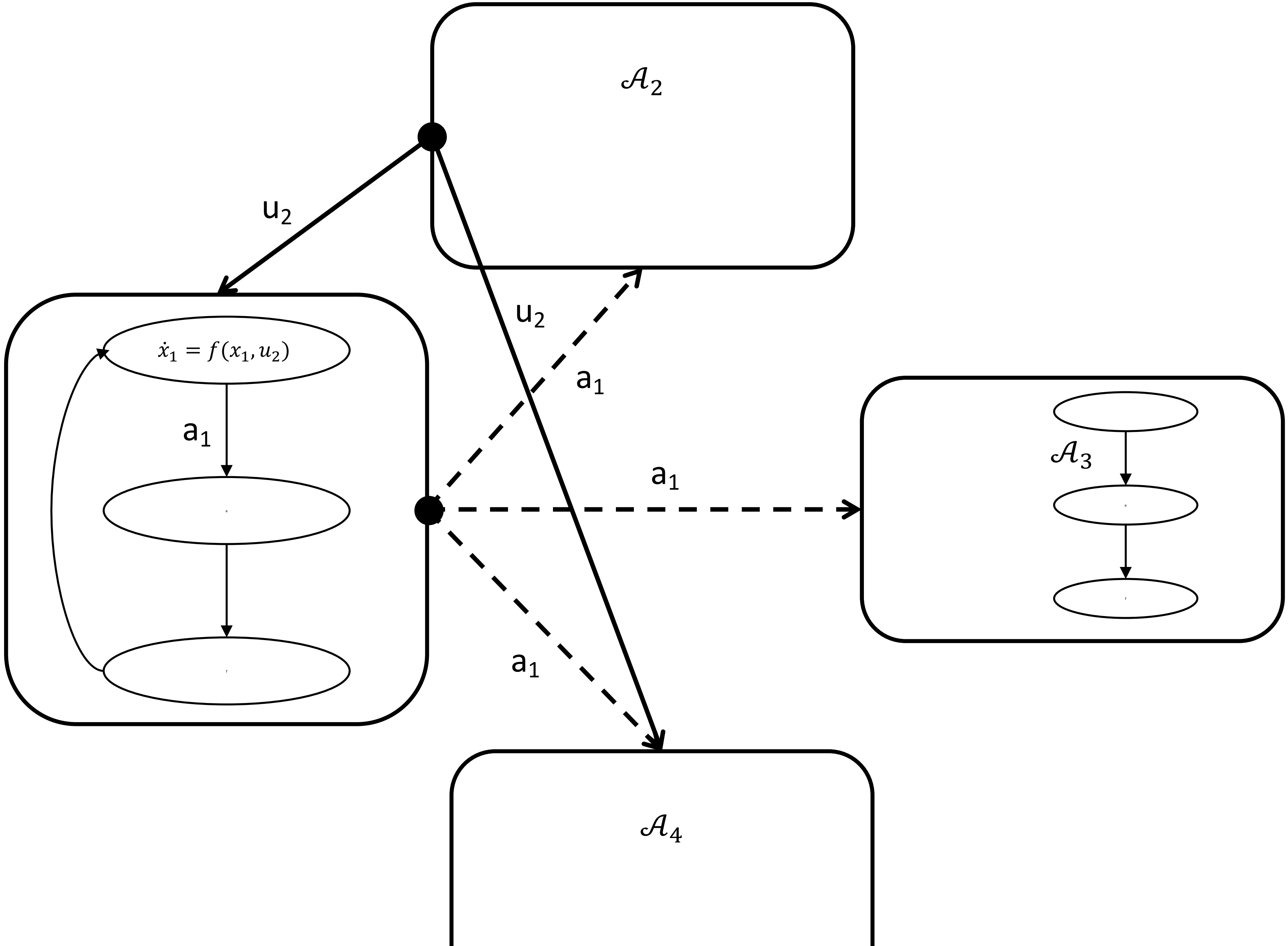
$Y_i \cap Y_j = \emptyset$ no duplication of continuous authority

Extended to collection of automata in the natural way and captures most common notions of composition in, for example, Matlab/Simulink



Composition

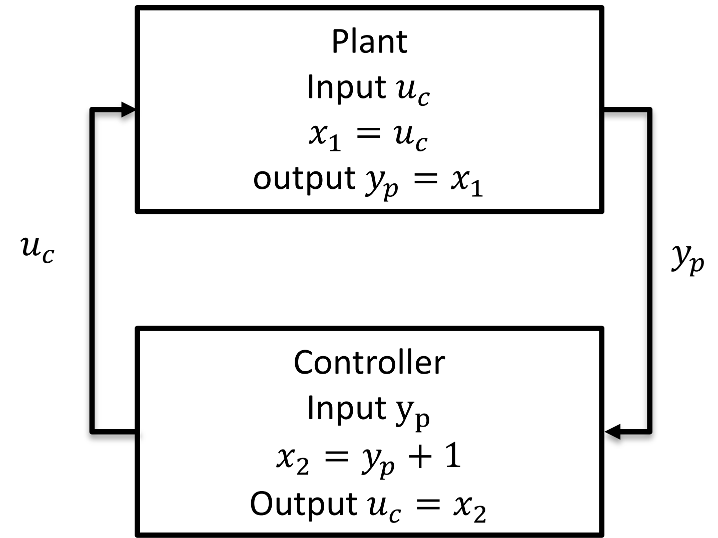
- For compatible $\mathcal{A}_1$ and $\mathcal{A}_2$ their composition $\mathcal{A}_1 \,||\, \mathcal{A}_2$ is the structure $\boldsymbol{\mathcal{A}}$= $(V, \Theta, A, \mathcal{D}, \mathcal{T})$
- Variables $V = X \cup Y \cup U$
  - $X = X_1 \cup X_2$, $Y = Y_1 \cup Y_2$, $U = U_1 \cup U_2 \setminus Y$
- $\Theta = \{ \boldsymbol{x} \in val(X) | \forall i \in \{1,2\}: \boldsymbol{x} \lceil X_i \in \Theta_i \}$
- Actions $A = H \cup O \cup I$
  - $H = H_1 \cup H_2$, $O = O_1 \cup O_2$, $I = I_1 \cup I_2 \setminus O$,
- $(\boldsymbol{x}, a, \boldsymbol{x}') \in \mathcal{D}$ *iff for* $i \in \{1,2\}$
  - $a \in A_i$ *and* $(\boldsymbol{x} \lceil X_i, a, \boldsymbol{x}' \lceil X_i) \in \mathcal{D}_i$
  - $a \notin A_i$ $\boldsymbol{x} \lceil X_i = \boldsymbol{x} \lceil X_i$
- $\mathcal{T}$: set of **trajectories** for V
  - $\tau \in \mathcal{T}$ iff $\forall i \in \{1,2\}$, $\tau \downarrow V_i \in \mathcal{T}_i$



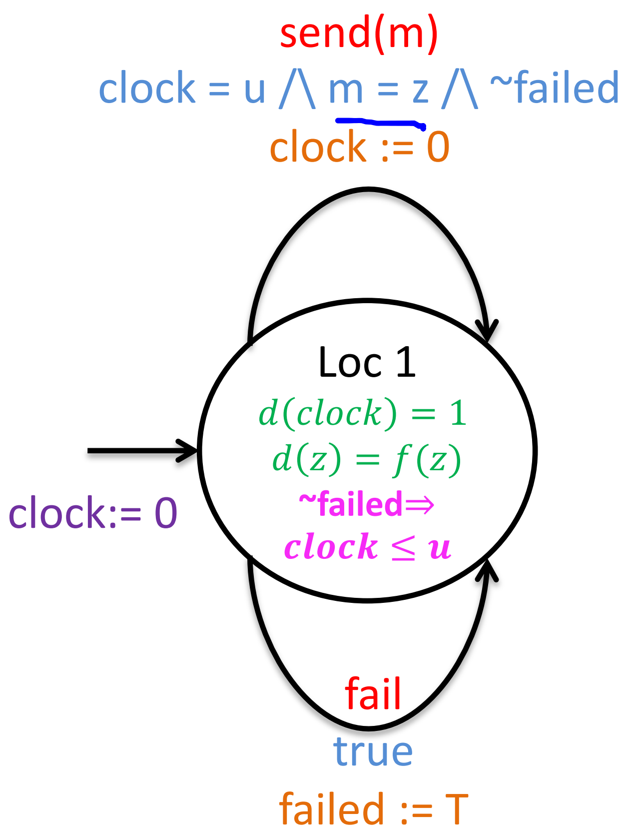
$\mathcal{A}_2$
u_2
u_2
$\dot{x}_1 = f(x_1, u_2)$
a_1
a_1
a_1
$\mathcal{A}_3$
a_1
$\mathcal{A}_4$

Closure under composition

- Conjecture. The class of HIOA is closed under composition. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible HIOA then $\mathcal{A}_1||\mathcal{A}_2$ is also a HIOA.
- Can we ensure that input trajectory enabled condition is satisfied in the composed automaton?
- No, in general
  - Additional conditions are needed



Example 2: Periodically Sending Process



Automaton PeriodicSend(u)

variables internal

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

signature output send(m:Reals)

input fail

transitions:

output send(m)

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

eff clock := 0

input fail

pre true

eff failed := T

trajectories:

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

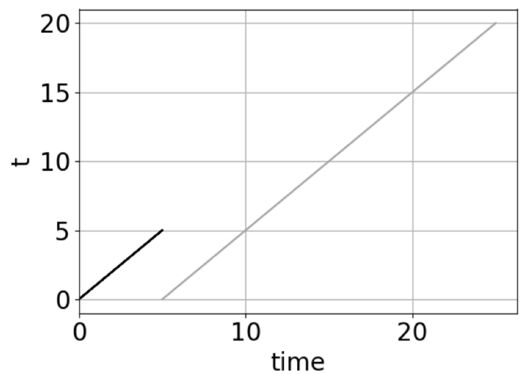
invariant failed \/ clock≤u

Time bounded channel & Simple Failure Detector

Automaton Timeout(u,M)
variables: suspected: Boolean := F,
clock: Reals := 0
signature input receive(m:M)
output timeout
transitions:
input receive(m)
pre true
eff clock := 0; suspected := false;
output timeout
pre ~suspected /\ clock = u
eff suspected := true
trajectories:
evolve d(clock) = 1
invariant $clock \leq u \vee suspected$

Automaton Channel(b,M)
variables internal queue: **Queue**[M,Reals] := <
clock: Reals := 0
signature input send(m:M)
output receive(m:M)
transitions:
input send(m)
pre true
eff queue := append(<m, clock+b>, queue)
output receive(m)
pre $head(queue)[1]=m \wedge$
$head(queue)[2]=clock$
eff queue := queue.tail
trajectories:
evolve d(clock) = 1
invariant $\forall \langle m,d \rangle \in queue: d \geq clock$

Example 3: Oscillator and pulse generator



On
$d(now) = 1$
$u = 1$
$inv\ now \leq T_{on}$

pre $now \geq T_{on}$ **eff** $now := 0$ ↓

↑ **pre** $now \geq T_{off}$ **eff** $now := 0$

Off
$d(now) = 1$
$u = 0$
$inv\ now \leq T_{off}$

pulseGen

Mode
$$d(x_1) = -x_1(x_1^2 + 0.9x_1 + 0.9) - x_2 + u$$
$$d(x_2) = x_1 - 2x_2$$

Oscillator

Composed automaton

On,Mode

$d(now) = 1$

$d(x_1) = -x_1(x_1^2 + 0.9x_1 + 0.9) - x_2 + 1$

$d(x_2) = x_1 - 2x_2$

$now \leq T_{On}$

pre: $now \geq T_{On}$
eff: $now \coloneqq 0$

pre: $now \geq T_{Off}$
eff: $now \coloneqq 0$

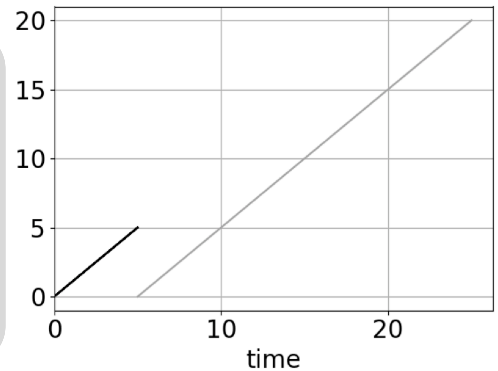
Off,Mode

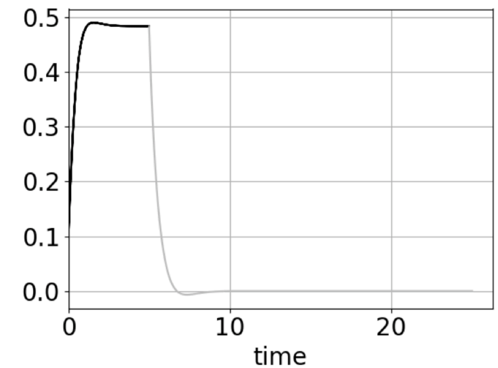
$d(now) = 1$

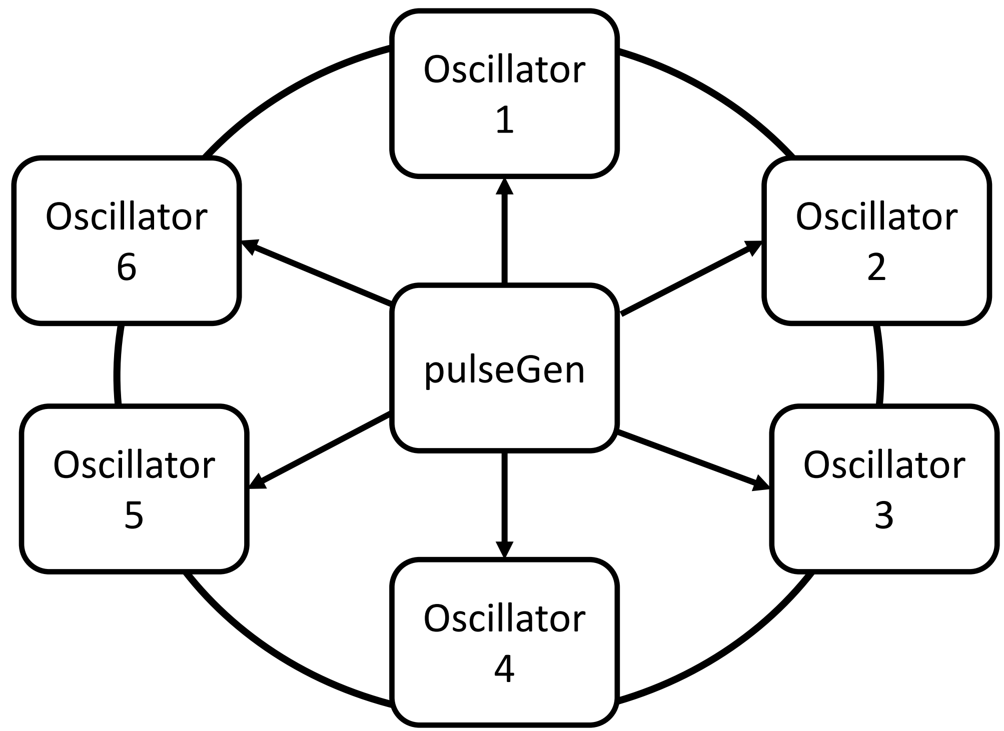
$d(x_1) = -x_1(x_1^2 + 0.9x_1 + 0.9) - x_2 + 0$

$d(x_2) = x_1 - 2x_2$

$now \leq T_{Off}$







Cardiac oscillator network models, Grosu et al. CAV, HSCC 2007-2015

Restriction operation on exections

- Sometimes it is useful to restrict our attention to only some subset of variables and actions in an execution
- Recall the **restriction** operations $x\lceil V \; and \;\; \tau \downarrow V$
- Let $\alpha = \tau_0 a_1 \tau_1 a_2$ be an execution fragment of a hybrid automaton with set of variables V and set of actions A. Let A' be a set of actions and V' be a set of variables.

- **Restriction** of α to (A', V'), written as $\alpha\lceil(A', V')$ is the sequence defined inductively as:
  - $\alpha\lceil(A', V')$ = $\tau \downarrow V'$ if $\alpha = \tau$
  - $\alpha \; a \; \tau \lceil(A', V')$ =
    - $\alpha \lceil(A', V') \; a \; (\tau \downarrow V')$ if a $\in A'$
    - $\alpha \lceil(A', V') \; concat \;\; (\tau \downarrow V')$ if a $\notin A'$

- From the definition it follows $\alpha.lstate \lceil V' = \alpha\lceil(A', V').lstate$ for any A', V'

Properties of Compositions

Proposition. Let $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$. α is an execution fragment of $\mathcal{A}$ iff $\alpha \lceil (A_i, V_i), i \in \{1,2\}$ are both execution fragments of $\mathcal{A}_i$.

- Proof of the forward direction. Fix α and i. We prove this by induction on the length of α.

- Base case: $\alpha = \tau$. $\alpha \lceil (A_i, V_i) = \tau \downarrow V_i$ by definition of composition $\tau \downarrow V_i \in T_i$. So, $\tau \downarrow V_i \in Frag_i$

- $\alpha = \alpha' \, a \, \tau \lceil (A_i, V_i)$ and $a \in A_i$ and by induction hypothesis $\alpha' \lceil (A_i, V_i) \in Frag_i$. Let $\alpha' \lceil (A_i, V_i)$. lstate = v. By the definition of composition: $\tau \downarrow V_i \in T_i$.
  - It remains to show that $v \lceil V_i \rightarrow_a (\tau \downarrow V_i).fstate$. Since $a \in A_i$, by the definition of composition: $\alpha' \lceil (A_i, V_i)$. lstate $\rightarrow_a \tau \downarrow V_i.fstate$

- $\alpha = \alpha' \, a \, \tau \lceil (A_i, V_i)$ and $a \notin A_i$ and by induction hypothesis $\alpha' \lceil (A_i, V_i) \in Exec_i$. Let τ' be the last trajectory in that execution.
  - Since $a \notin A_i$, by the definition of composition:$\tau'.lstate = \tau \downarrow V_i.fstate$. By concatenation closure of T_i, it follows that $\tau' concat \, \tau \downarrow V_i \in \mathrm{T}_i$. Therefore $\alpha \lceil (A_i, V_i) \in Exec_i$.

properties of executions of composed automata

- α is an *execution* iff $\alpha\lceil(A_i, V_i), i \in \{1,2\}$ are both executions.
- α is *time bounded* iff $\alpha\lceil(A_i, V_i), i \in \{1,2\}$ are both time bounded.
- α is *admissible* iff $\alpha\lceil(A_i, V_i), i \in \{1,2\}$ are both admissible.
- α is *closed* iff $\alpha\lceil(A_i, V_i), i \in \{1,2\}$ are both closed.
- α is *non-Zeno* iff $\alpha\lceil(A_i, V_i), i \in \{1,2\}$ are both time non-Zeno.

Summary

- Composition operation
  - I/O interfaces: actions and variables
  - Reactivity/input enabling
  - (non) Closure under composition
- Properties of executions preserved under composition
- Inductive invariants

Example Inductive Invariance Proof

S: $\forall$ <m,d> $\in$ **x.**queue: **x**.clock $\leq$ d $\leq$ **x**.clock+b
Is an invariant for the timed channel.

Proof. Use the theorem.

- Check start condition. Holds vacuously as **x.**queue $=$ {} [Def of initial states]
- Check trajectory condition. 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 $\leq$ d [By Assumption]
  - Suppose **x'.**clock $>$ d
  - **x'.**clock - **x.**clock $>$ d - **x.**clock
  - t $>$ d - **x.**clock, then there exists t' $\in$ τ.dom and t' $<$ t where τ(t').clock $=$ d
  - By **invariant** τ.ltime $=$ t' which is a contradiction
  - Also, since d $\leq$ **x**.clock+b, d$\leq$ **x'**.clock+t+b
- Check transitions:
  - $\boldsymbol{x} \rightarrow_{\boldsymbol{receive}(\boldsymbol{m})} \boldsymbol{x}'$. Follows from assumption $\boldsymbol{x} \in \boldsymbol{S}$.
  - $\boldsymbol{x} \rightarrow_{\boldsymbol{send}(\boldsymbol{m})} \boldsymbol{x}'$ **…**