# Modeling Cyberphysical Systems

Sayan Mitra

Verifying cyberphysical systems

mitras@illinois.edu

# Map of CPS models

Discrete transition systems, automata

Dynamical systems
Differential inclusions

Markov chains

Probabilistic automata, Markov decision processes (MDP)

Continuous time, continuous state MDPs

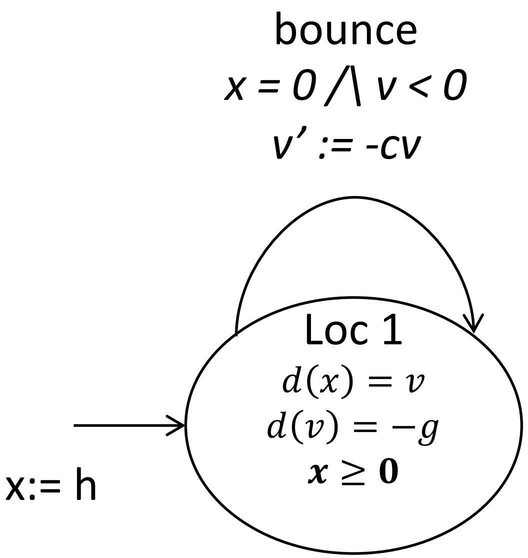
**Hybrid systems**

Stochastic Hybrid systems

# Outline

- Hybrid automata
- Executions
- Urgency
- Zeno
- Hybrid stability

# Bouncing Ball: Hello world of CPS




Graphical Representation used in many articles

**automaton** Bouncingball(c,h,g)

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

    **actions:** bounce

    **transitions:**

        bounce

            **pre** *x = 0 /\ v < 0*

            **eff** v := -cv

    **trajectories:**

            Loc1

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

            **invariant** $\boldsymbol{x \geq 0}$

mode invariant, not to be confused with invariants of the automaton

# Recall from Lecture 1. language defines an automaton

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

- $X$ is a set of names of variables; each variable $x \in X$ is associated with a type, $type(x)$
  - A **valuation** for $X$ maps each variable in X to its type
  - Set of all valuations:$val(X)$ this is sometimes identified as the **state space** of the automaton
- $\Theta \subseteq val(X)$ is the set of **initial** or **start states**
- $A$ is a set of names of **actions** or **labels**
- $\mathcal{D} \subseteq val(X) \times A \times val(X)$ is the set of **transitions**
  - a transition is a triple $(u, a, u')$
  - We write it as $u \rightarrow_a u'$

```
automaton DijkstraTR(N:Nat, K:Nat), where K > N
  type ID: enumeration [0,...,N-1]
  type Val: enumeration [0,...,K]
  actions
     update(i:ID)
  variables
     x:[ID -> Val]
  transitions
     update(i:ID)
       pre i = 0 /\ x[i] = x[N-1]
       eff x[i] := (x[i] + 1) % K

      update(i:ID)
        pre i >0 /\ x[i] ~= x[i-1]
        eff x[i] := x[i-1]
```

# Trajectories

Given a set of variables $X$ and a time interval $J$ which can be of the form $[0, T], [0, T) or\ [0, \infty)$, a ***trajectory*** for $X$ is a function $\tau\colon\ J \to val(X)$

We will specify $\tau$ as solutions of differential equations

The **first state** of a trajectory $\tau.fstate := \tau(0)$

If $\tau$ is right closed then the **limit state** of a trajectory $\tau.lstate = \tau(T)$

If $\tau$ is finite then **duration** of $\tau$ is $\tau.dur = T$

The domain of $\tau.dom = J$

A **point trajectory** is a trajectory with $\tau.dom = [0,0]$

Operations on trajectories: prefix, suffix, concatenation

A **prefix** $\tau'$ of a trajectory $\tau\colon [0, T] \to val(X)$, is a function $\tau'\colon [0, T'] \to val(X)$ such that $T' \leq T$ and $\tau'(t) = \tau(t)$ for all $t \in \tau'.dom$

# Hybrid Automaton

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

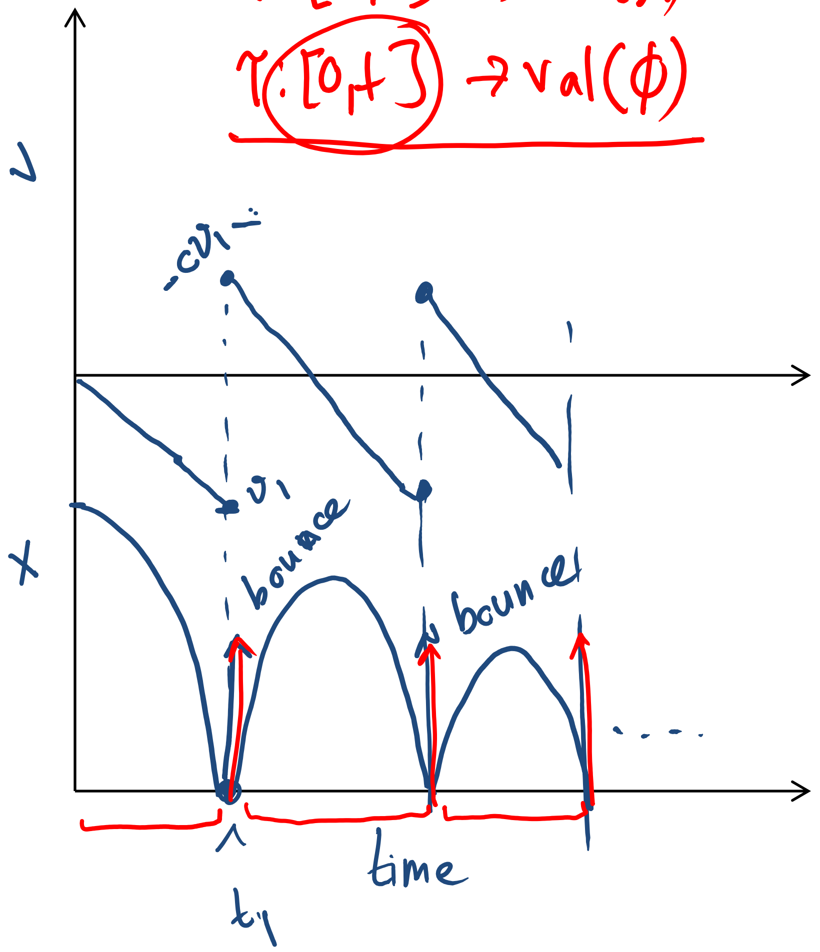
- $X$: set of **state variables**
  - $Q \subseteq val(X)$ set of **states**
- $\Theta \subseteq Q$ set of **start states**
- set of **actions**, A= E ∪ H
- $\mathcal{D} \subseteq Q \times A \times Q$
- $\mathcal{T}$: set of **trajectories** for X which is closed under **prefix, suffix, and concatenation**

# Semantics: Executions and Traces

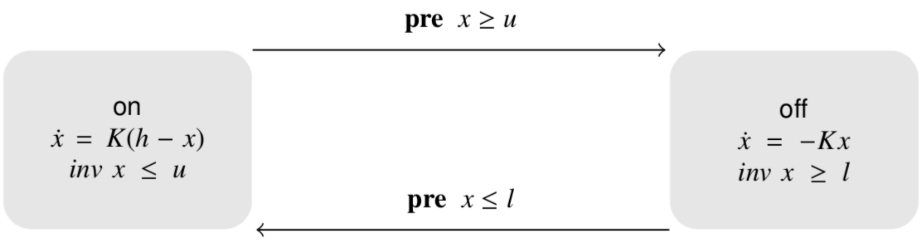
- An *execution fragment* of $\mathcal{A}$ is an (possibly infinite) alternating (A, X)-sequence $\alpha = \tau_0\, a_1\, \tau_1 a_2 \tau_2\ \ldots$ where
  - $\forall$ i, $\tau_i.lstate \xrightarrow{a_{i+1}} \tau_{i+1}.fstate$
- If $\tau_0$.fstate $\in \Theta$ then $\alpha$ is an **execution**
- **Execs**$_{\mathcal{A}}$ set of all executions
- The first state of an execution $\alpha$ is $\alpha.fstate = \tau_0.fstate$
- If the execution $\alpha$ is **finite and closed** $\tau_0\, a_1\, \tau_1 a_2 \tau_2\ \ldots \tau_k$ then $\alpha.lstate = \tau_k.lstate$
- A state $\boldsymbol{x}$ is reachable if there exists an execution $\alpha$ with $\alpha.lstate = \boldsymbol{x}$

$\tau : [0,t] \to val(X)$

$\tau : [0,t] \to val(\phi)$



# Thermostat variations




**automaton** Thermostat(*u*, *l*, *K*, *h* : Real) where *u* > *l*
**type** *Status* enumeration [*on*, *off* ]
**actions**
turnOn; turnOff;

**variables**
*x*: Real := *l* *loc*: *Status* := *on*
**transitions**
turnOn
**pre** *x*≤*l* ∧ *loc*=*off*
**eff** *loc* := *on*

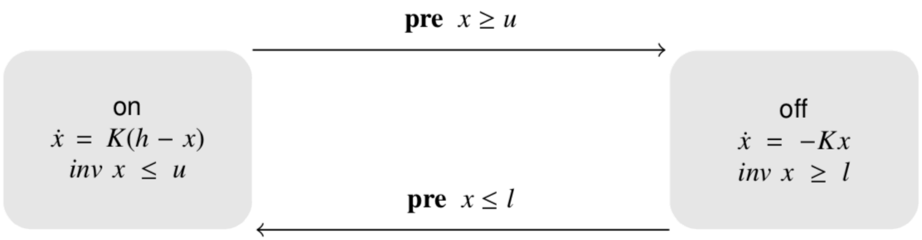
turnOff
**pre** *x*≥*u* ∧ *loc*=*on*
**eff** *loc* := *off*

**trajectories**
modeOn
**evolve** *d*(*x*) = *K*(*h* − *x*)
**invariant** *loc* = *on* ∧ *x* ≤ *u*

modeOff
**evolve** *d*(*x*) = −*Kx*
**invariant** *loc* = *off* ∧ *x* ≥ *l*

Determinism vs nondeterminism
mode invariants

# Urgency




An ***urgent*** transition (or action) is an action that has to occur as soon as it is enabled

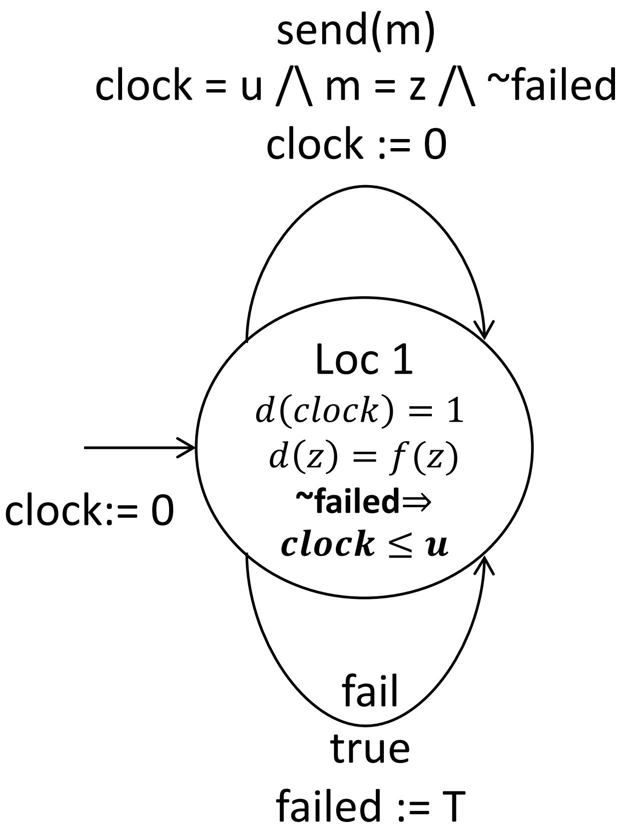
Our language does have special syntax for creating urgent transitions, but we can use mode invariants to accomplish this

```
automaton Thermostat(u, l, K, h,d : Real) where u > l
  type Status enumeration [on, off ]
  actions
    turnOn; turnOff;

  variables
    x: Real := l loc: Status := on
  transitions
    turnOn                              turnOff
    pre x≤l ∧ loc=off                     pre x≥u ∧ loc=on
    eff loc := on                         eff loc := off

trajectories
    modeOn                              modeOff
    evolve d(x) = K(h − x)                evolve d(x) = −Kx
    invariant loc = on ∧ x ≤ u + d        invariant loc = off ∧ x ≥ l - d
```

# Another Example: Periodically Sending Process




**Automaton** PeriodicSend(u)

**variables: analog**

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

**actions:** send(m:Reals), fail

**transitions:**

send(m)

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

**eff** clock := 0

fail

**pre** true

**eff** failed := T

**trajectories:**

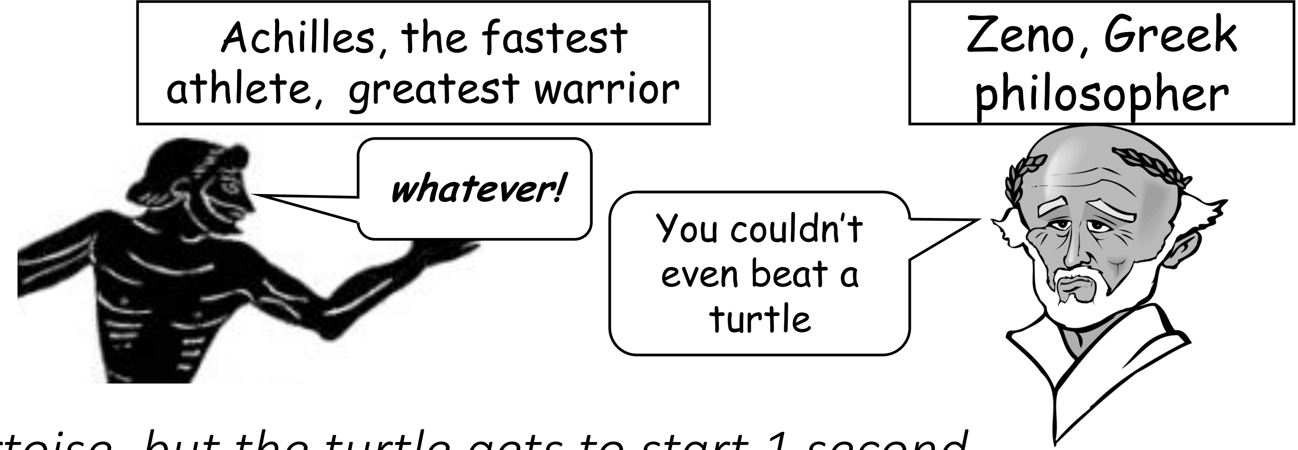
**evolve** d(clock) = 1, d(z) = f(z)

**invariant** failed \/ clock<=u

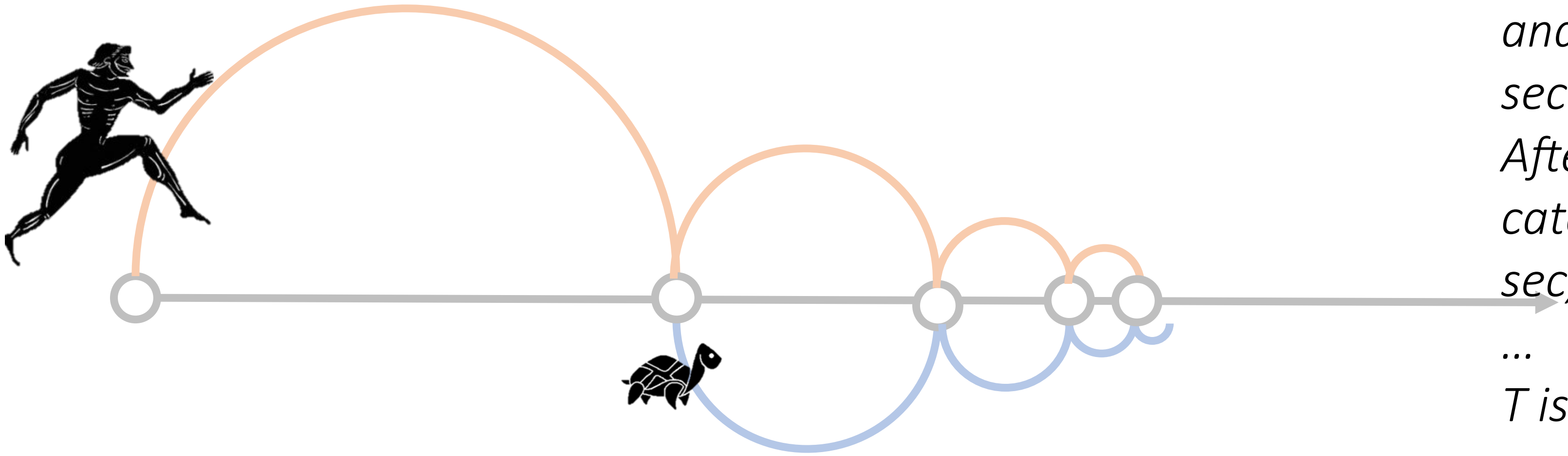
# Special kinds of executions

- **Infinite**: Infinite sequence of transitions and trajectories $\tau_0\ a_1\ \tau_1 a_2 \tau_2\ \ldots$
- **Closed**: Finite with final trajectory with closed domain $\tau_0\ a_1\ \tau_1 a_2 \tau_2\ \ldots \tau_k$ and $\tau_k.dom = [0, T]$
- **Admissible**: Infinite duration
  - May or may not be infinite
  - $\tau_0\ a_1\ \tau_1 a_2 \tau_2\ \ldots$
  - $\tau_0\ a_1\ \tau_1 a_2 \tau_2\ \ldots \tau_k$ with $\tau_k.dom = [0, \infty)$
- **Zeno**: Infinite but not admissible
  - Infinite number of transitions in finite time

# Zeno’s Paradox




*Achilles runs 10 times faster than than the tortoise, but the turtle gets to start 1 second earlier. Can Achilles ever catch Turtle?*

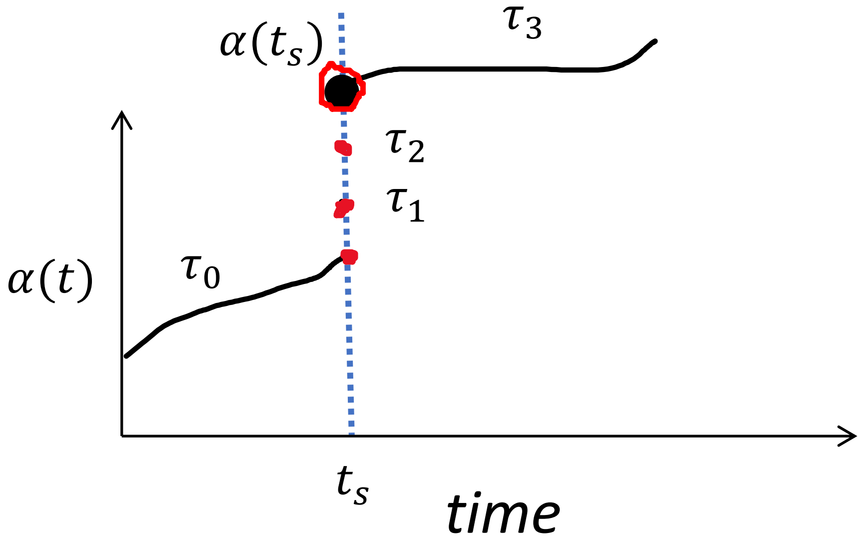


*After 1/10th of a second, Achilles reaches where the Turtle (T) started, and T has a head start of 1/10th second.*
*After another 1/100th of a second, A catches up to where T was at t=1/10 sec, but T has a head start of 1/100th*
*…*
*T is always ahead …*

*Lesson: Mixing discrete transitions with continuous motion can be tricky!*

# Defining stability of hybrid automata

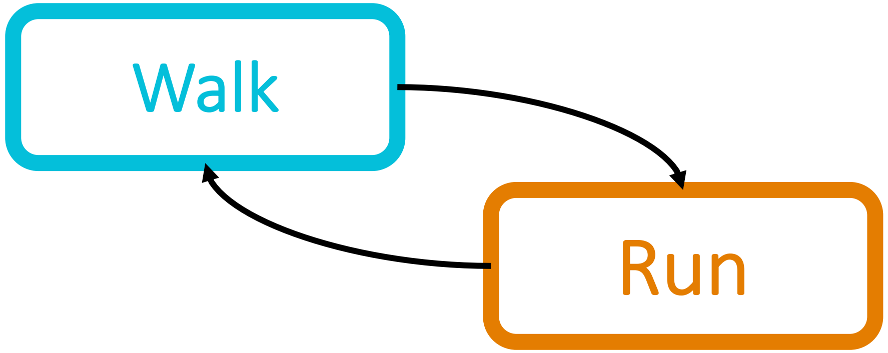
- Given an admissible execution $\alpha = \tau_0\ a_1\ \tau_1 a_2 \tau_2\ \ldots$
- We would like to view an execution as $\alpha: [0, \infty) \rightarrow val(X)$
- But, how can we define $\alpha(t)$?
- define $\alpha(t_s) = \alpha'.lstate$ where $\alpha'$ is the longest **prefix** of $\alpha$ with $\alpha'.ltime = t_s$

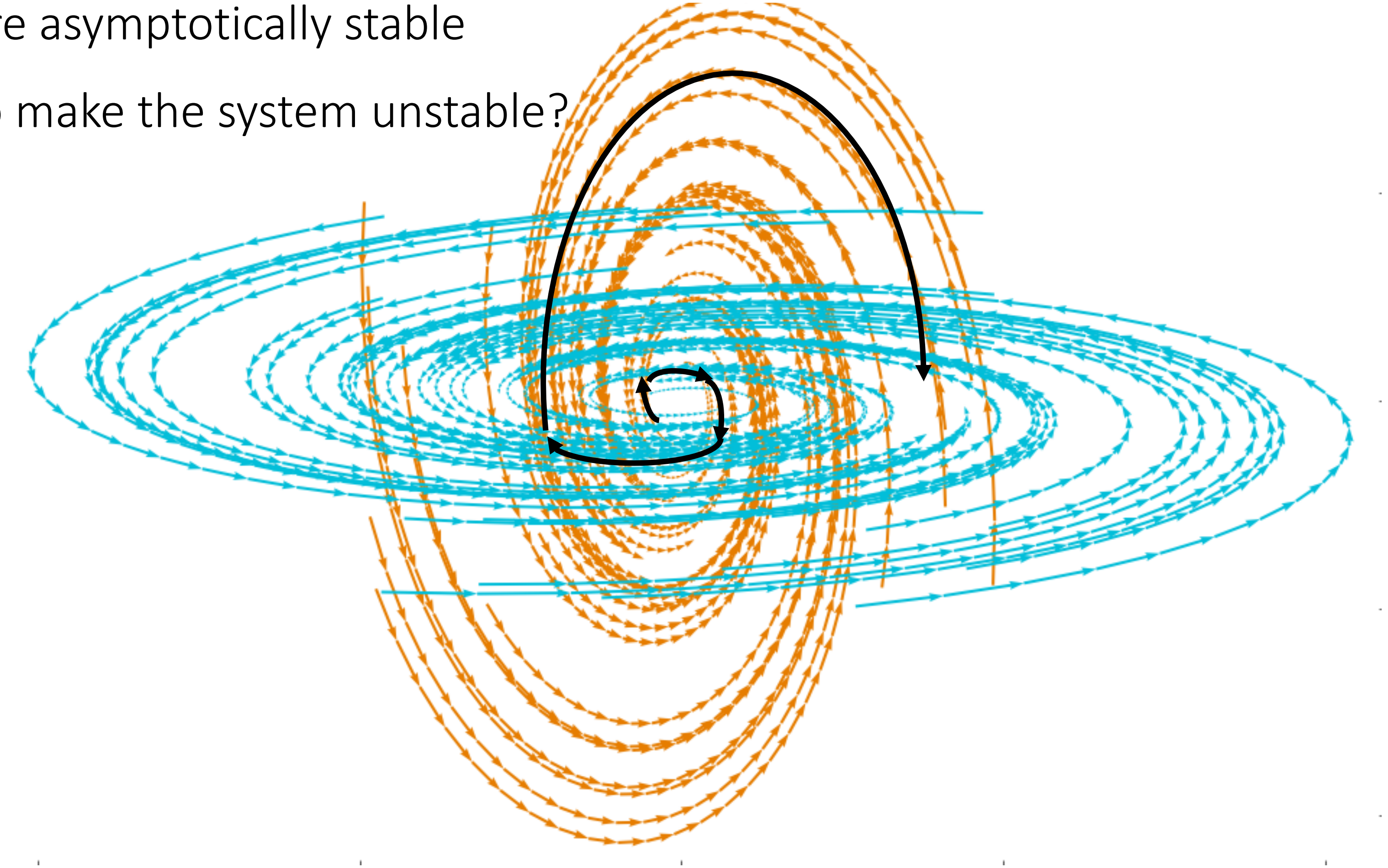


# Hybrid Instability

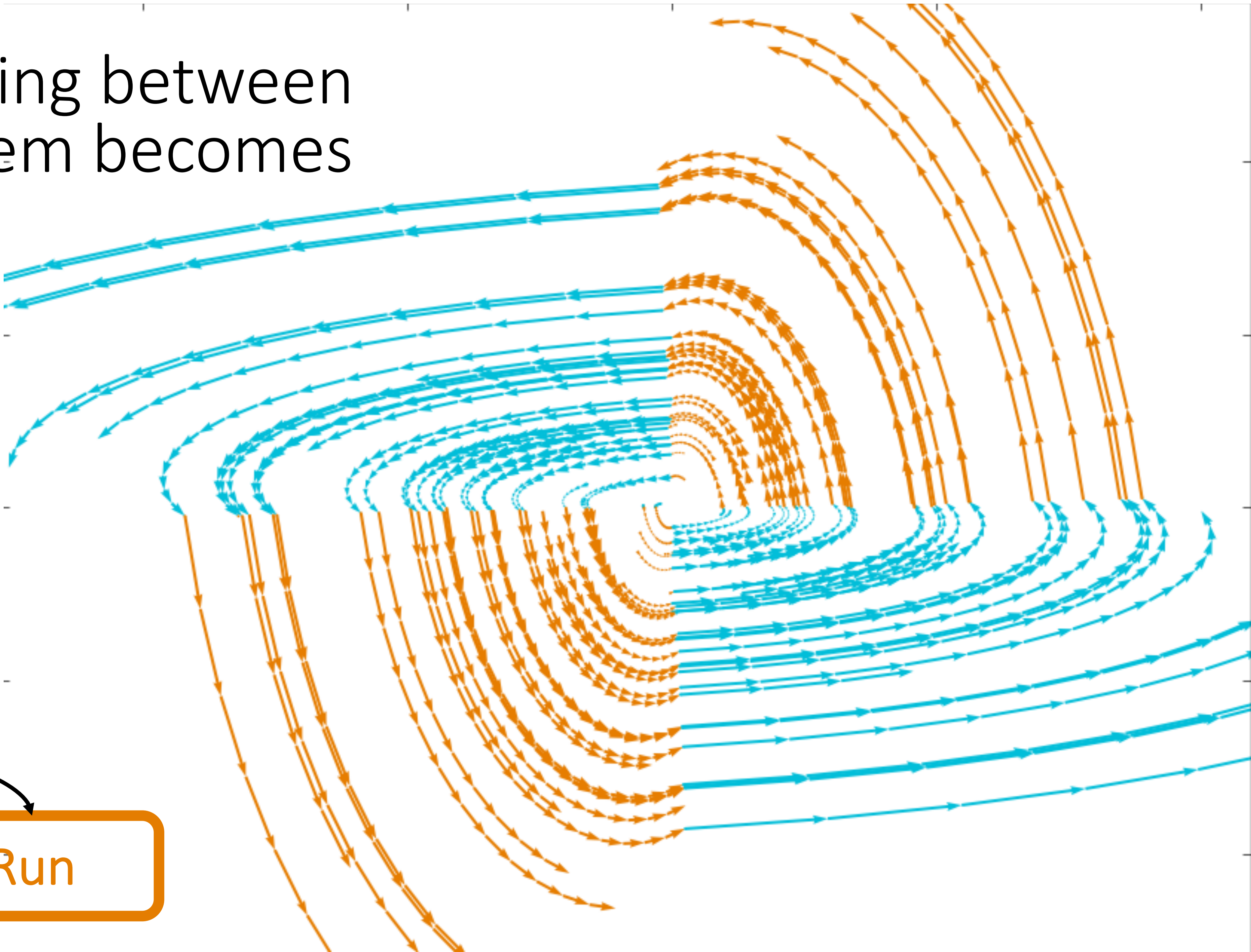
Each of the modes of a walking robot are asymptotically stable

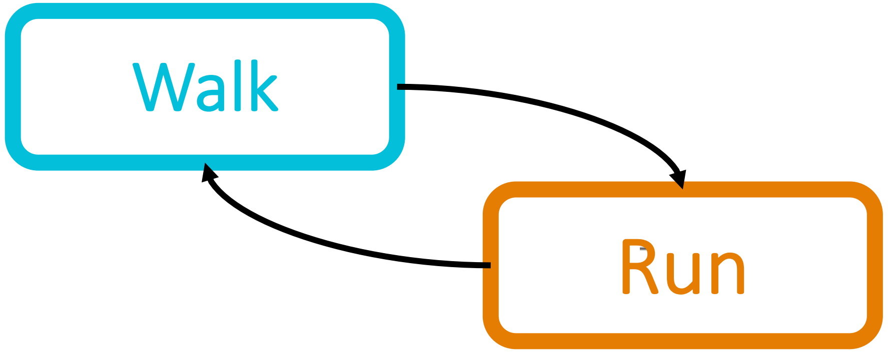
Is it possible to switch between them to make the system unstable?



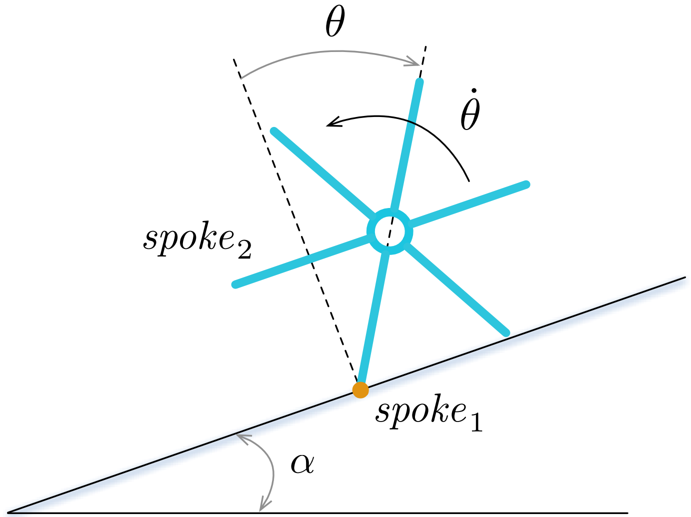



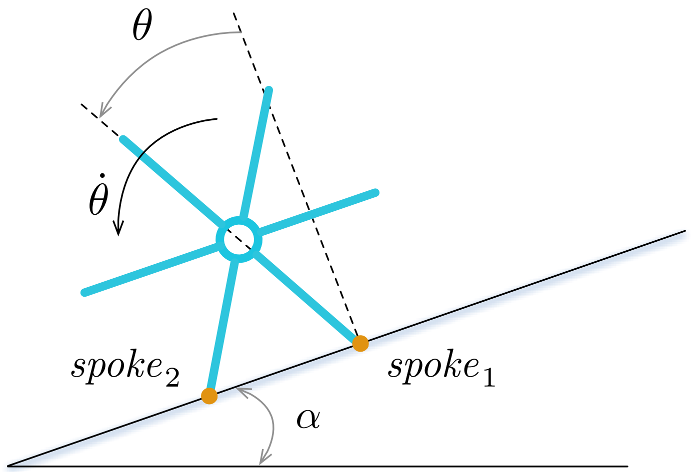
# Yes! By switching between them the system becomes unstable

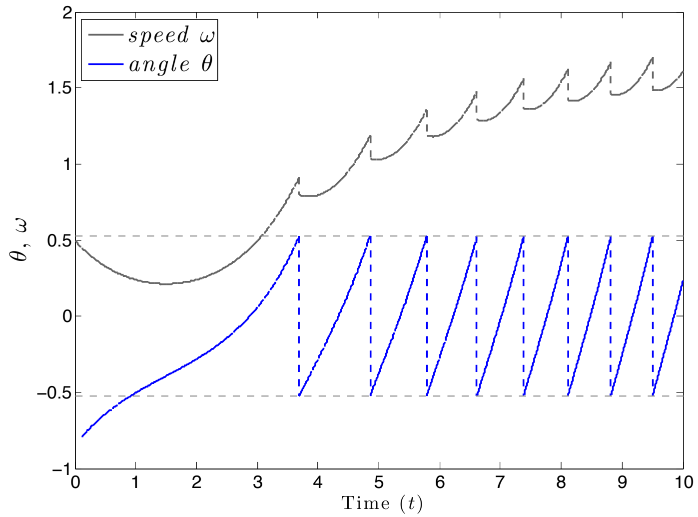




# Rimless wheel




**automaton** RimlessWheel($\alpha, \mu$: Real, $n$: Nat)
const $\beta$: Real := 2 $\pi/n$
**type** Spokes: enumeration [1,…,n]
**actions**
impact
**variables**
pivot: Spokes :=1
$\theta$:Real := 0
$\omega$: Real := 0
**transitions**
impact
pre $\theta \geq \beta/2$
**eff** pivot := pivot + 1 mod n
$\theta := -\beta/2$
$\omega := \mu\omega$

**trajectories**
swing
**evolve**
$\mathrm{d}(\theta) = \omega$
$d(\omega) = \sin(\theta + \alpha)$
**invariant** $\theta \leq \frac{\beta}{2}$

# Invariants and reachability

- A state $\boldsymbol{x}$ of automaton $\mathcal{A}$ is ***reachable*** if there exists an execution $\alpha$ with $\alpha.lstate = \boldsymbol{x}$
- $Reach_{\mathcal{A}}(\Theta)$ is the set of all reachable state from $\Theta$
- $Reach_{\mathcal{A}}(\Theta, T)$ is the set of states reachable within time $T$
- $Reach_{\mathcal{A}}(\Theta, k)$ is the set of states reachable within $k$ transitions
- $Reach_{\mathcal{A}}(\Theta, T, k)$ is the set of states reachable up to time $k$ transitions and time $T$
- An invariant $I \subseteq val(X)$ is a set of states that contains $Reach_{\mathcal{A}}(\Theta)$