Introduction to Model Checking of Hybrid Systems

Oleg Sokolsky

CIS 700-002

Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Model Checking

- Systematic evaluation
  - of formally specified system behaviors
  - with respect to a formally specified property
  - using state space exploration
- Need a model of the system and a property specification
  - Use hybrid systems to represent behaviors
  - Use temporal logics to represent properties
- Verification vs. falsification
  - Verification constructs a proof that every behavior of the system satisfies the property
    - When verification fails, a counterexample is produced
  - Falsification systematically searches for counterexamples

Outline

- Hybrid systems modeling with hybrid automata
- Property specification with STL
- Set-based reachability analysis
- Simulation-based analysis

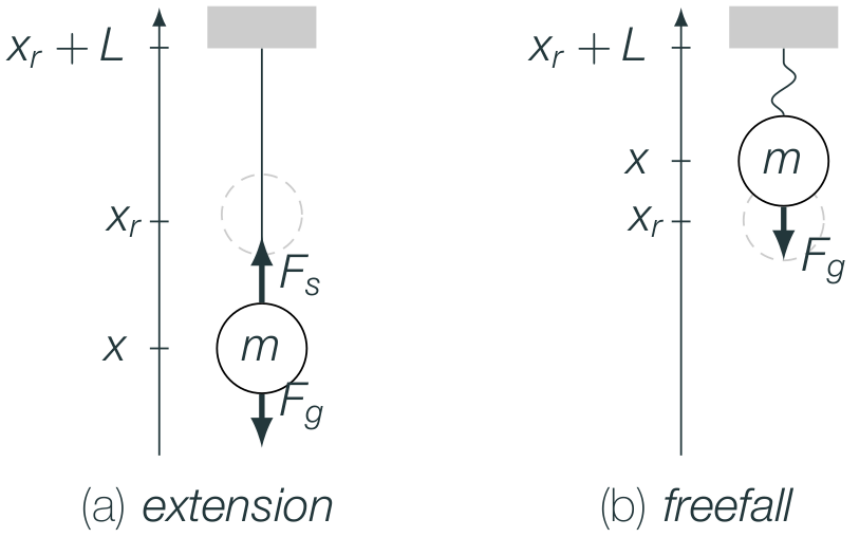
A lot of content is borrowed without permission from G. Frehse, E. Abraham, G. Fainekos

Hybrid Systems

- Combination of continuous and discrete behaviors
- Continuous behaviors "flow"
  - Values of variables gradually change with time
- Discrete behaviors "jump"
  - Variables instantaneously change values
- Hybrid systems are a **modeling device**
  - Physical entities are continuous
    - Unless quantum effects are considered
  - Discrete changes (e.g., digital computations) take time
  - Hybrid systems abstract away very fast dynamics
    - **Simpler models with sufficient accuracy**

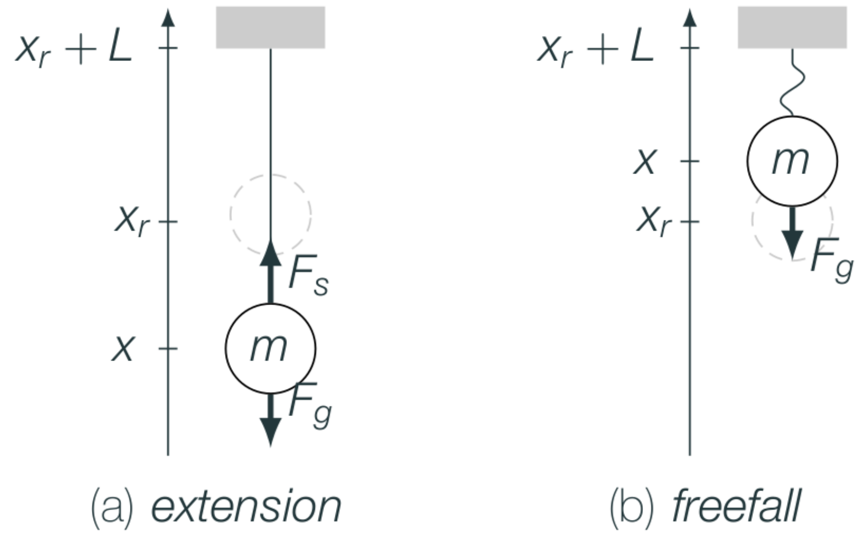
Example: Bouncing Ball

- Three phenomena:
  - String extension
  - Free fall
  - Collision
- How do we model collision?
  - Dynamics of hitting a wall are quite complex
    - Material deformation?
  - Discontinuous flows using DAEs?

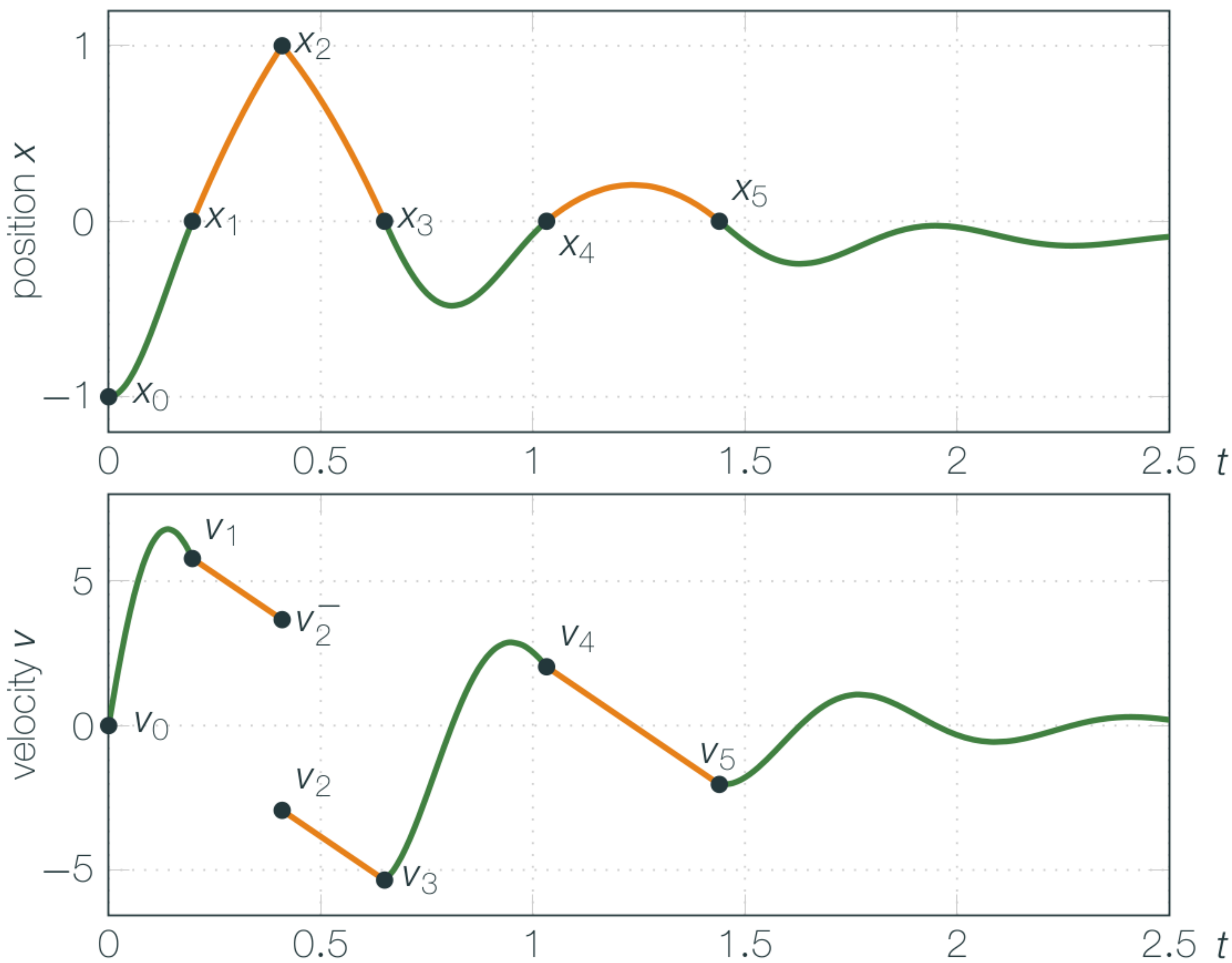


Bouncing Ball Dynamics

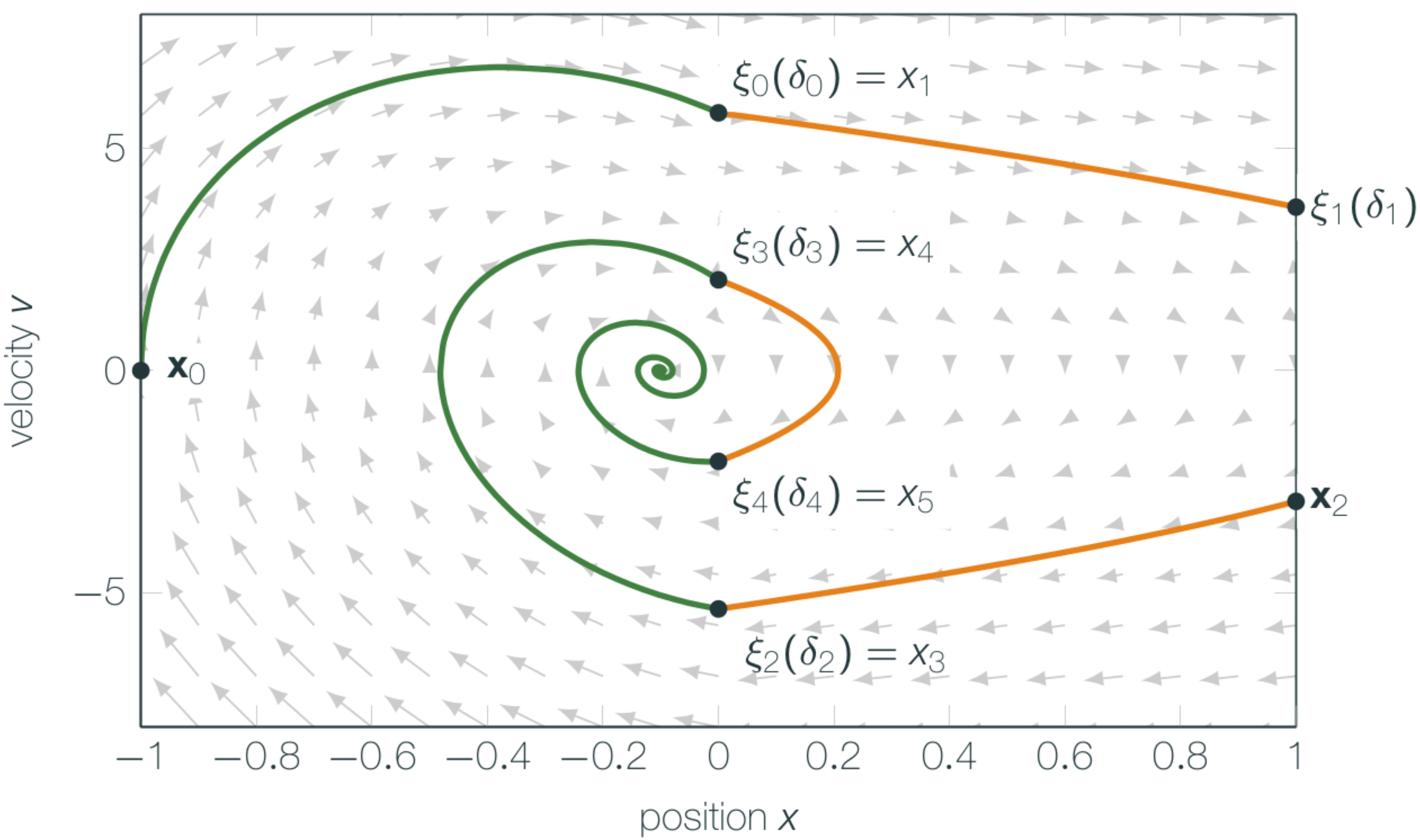
- Flows are captured by ODEs
- Jumps use "next value"
- Equations of motion
  - Free fall, $x \geq x_r$
    $$m\ddot{x} = F_g = -mg$$
    - *m* is mass, g is gravity constant
  - Extension, $x \leq x_r$
    $$m\ddot{x} = F_g + F_s = -mg + kx_r - kx - d\dot{x}$$
    - *k* is spring constant, *d* is damping
  - Collision, $x = x_r + L$
    $$\dot{x}' = -c\dot{x}$$
    - $c \in [0,1]$ is absorbtion factor



Behavior ($x_r = 0$)

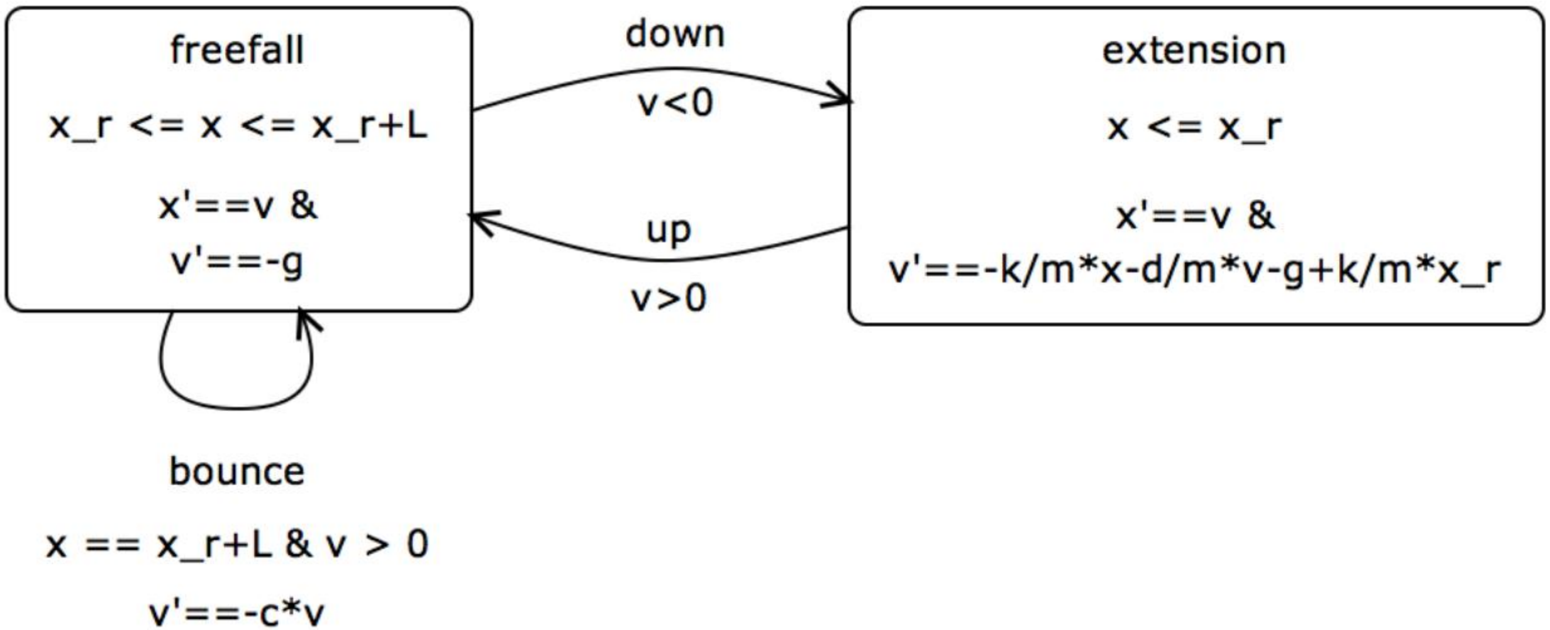


Phase Portrait



Hybrid Automata

- States (locations) pass time
  - Variables evolve according to flows
- Transitions are instantaneous and represent jumps



Hybrid Automata Syntax

- locations $\mathrm{Loc} = \{\ell_1, \ldots, \ell_m\}$ and variables $X = \{x_1, \ldots, x_n\}$ define the state space $\mathrm{Loc} \times \mathbb{R}^X$,
- transitions $\mathrm{Edg} \subseteq \mathrm{Loc} \times \mathrm{Lab} \times \mathrm{Loc}$ define location changes with synchronization labels Lab,
- invariant or staying condition $\mathrm{Inv} \subseteq \mathrm{Loc} \times \mathbb{R}^X$,
- flow relation Flow, where $\mathrm{Flow}(\ell) \subseteq \mathbb{R}^{\dot{X}} \times \mathbb{R}^X$, e.g.,

$$\dot{\mathbf{x}} = f(\mathbf{x});$$

- jump relation Jump, where $\mathrm{Jump}(e) \subseteq \mathbb{R}^X \times \mathbb{R}^{X'}$, e.g.,

$$\mathrm{Jump}(e) = \{(\mathbf{x}, \mathbf{x}') \mid \mathbf{x} \in \mathcal{G} \wedge \mathbf{x}' = r(\mathbf{x})\},$$

- initial states $\mathrm{Init} \subseteq \mathrm{Inv}$.

Hybrid Automata Semantics

- Hybrid automaton behaviors are given by a set of runs
- A run is an alternating sequence of flows and jumps

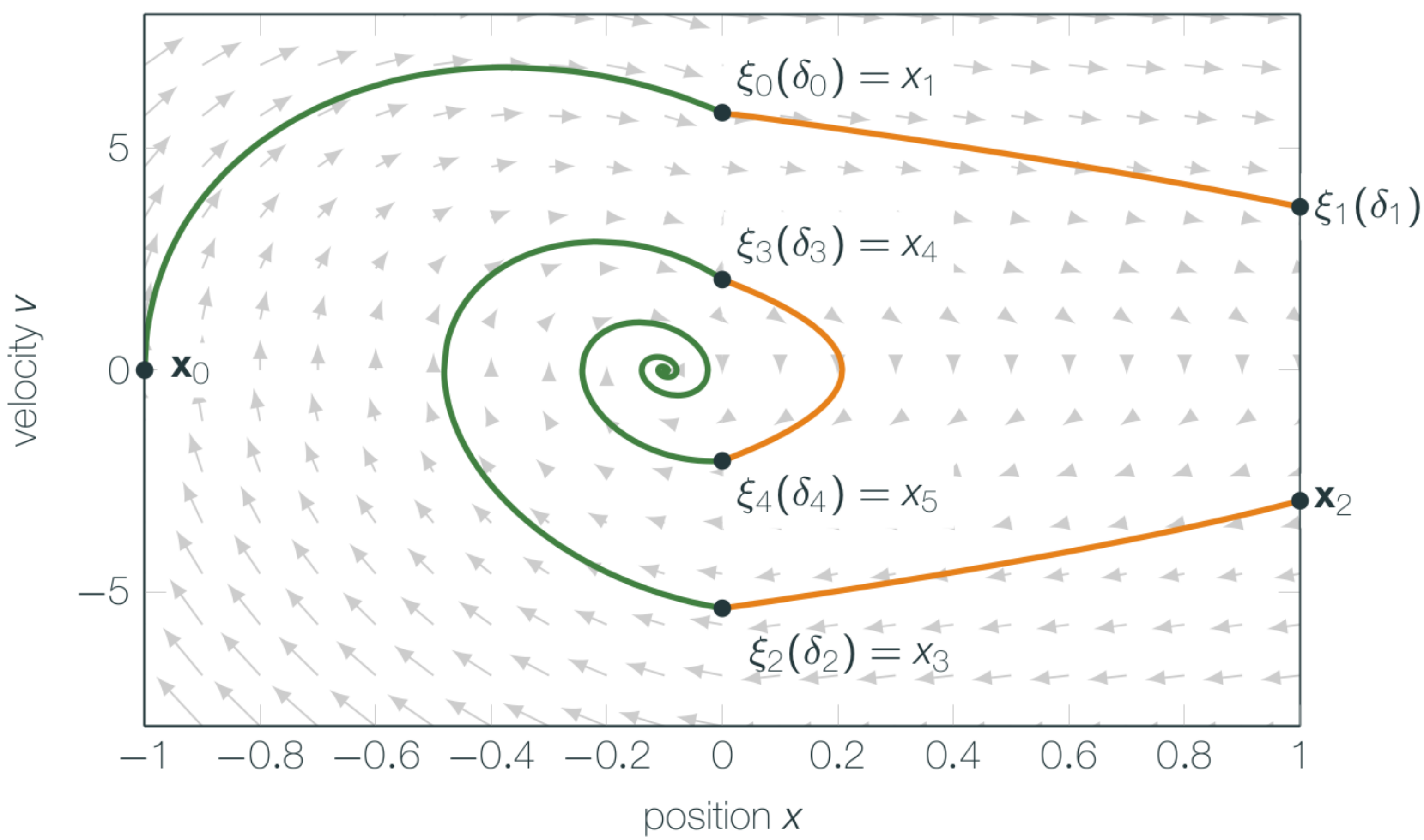
$$(\ell_0, \mathbf{x}_0) \xrightarrow{\delta_0, \xi_0} (\ell_0, \xi_0(\delta_0)) \xrightarrow{\alpha_0} (\ell_1, \mathbf{x}_1) \xrightarrow{\delta_1, \xi_1} (\ell_1, \xi_1(\delta_1)) \ldots$$

with $(\ell_0, \mathbf{x}_0) \in \mathrm{Init}$, $\alpha_i \in \mathrm{Lab} \cup \{\tau\}$, and for $i = 0, 1, \ldots$:

1. **Trajectories:** $(\dot{\xi}(t), \xi(t)) \in \mathrm{Flow}(\ell)$ and $\xi_i(t) \in \mathrm{Inv}(\ell_i)$ for all $t \in [0, \delta_i]$.
2. **Jumps:** $(\xi_i(\delta_i), \mathbf{x}_{i+1}) \in \mathrm{Jump}(e_i)$, $e_i = (\ell_i, \alpha_i, \ell_{i+1}) \in \mathrm{Edg}$, and $\mathbf{x}_{i+1} \in \mathrm{Inv}(\ell_{i+1})$.

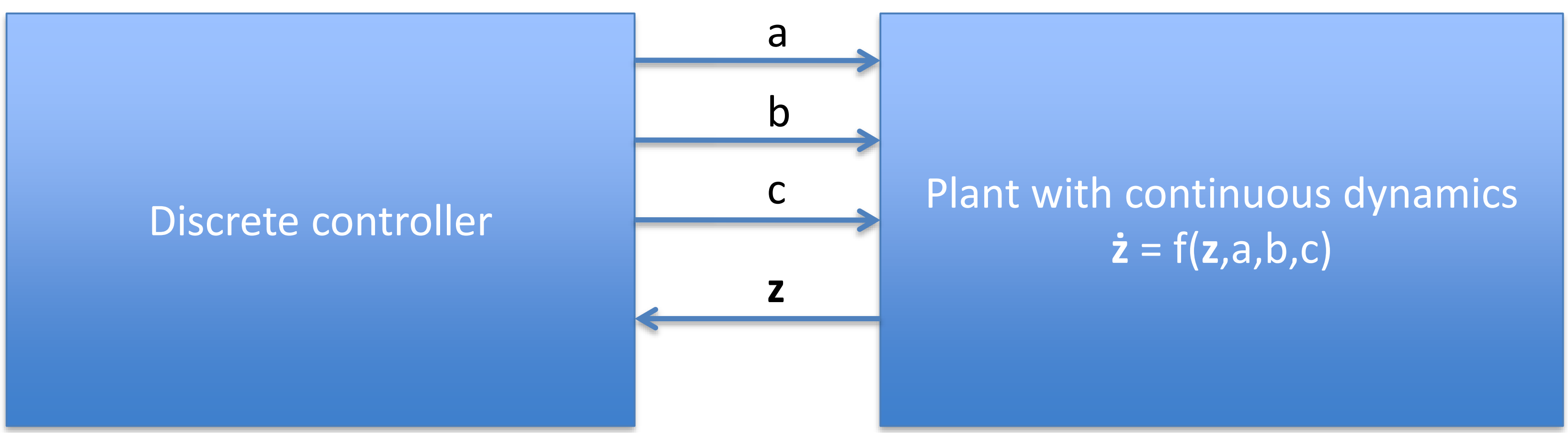
A state $(\ell, \mathbf{x})$ is **reachable** if there exists a run with $(\ell_i, \mathbf{x}_i) = (\ell, \mathbf{x})$ for some i.

Phase Portrait



Switched Systems

- Jumps in hybrid automata introduce discontinuities in trajectories
  - Often discontinuities are not needed
- A typical control system:



- Modes (locations) are implicit in the controller

Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Outline

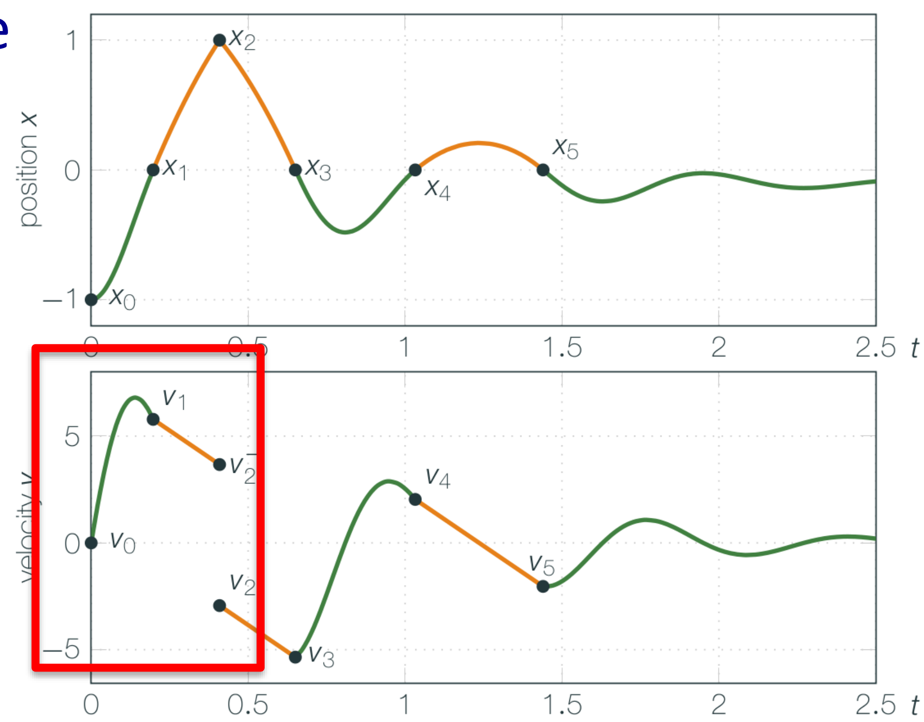
- Hybrid systems modeling with hybrid automata
- Property specification with STL
- Set-based reachability analysis
- Simulation-based analysis

Properties

- A property is a set of behaviors
  - A behavior satisfies property *P iff* it is included in the set
  - A system satisfies the property *iff* all its behaviors satisfy *P*
- Safety vs. liveness
  - Safety: something bad never happens
    - Never hit an obstacle
  - Liveness: something good eventually happens
    - Successfully complete a mission
  - An arbitrary property can be expressed as intersection of a safety and a liveness property
    - Complete a mission while avoiding obstacles

Signal Temporal Logic (STL)

- Properties are temporal relations between signal predicates
- Examples:
  - Velocity will be non-negative until a collision occurs
    - True
  - Collision will not occur
    - False



STL Syntax

- Syntax:

$$\varphi := \mathrm{true} \mid x_i \geq 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathrm{U}_I \, \varphi$$

  - X_i is a system variable
  - I is an interval [a,b]
  - U is the *until* operator

Common syntactic sugar:
$\Box_I \varphi = \mathrm{true} \ \mathrm{U}_I \neg \varphi$
$\Diamond_I \varphi = \neg \Box_I \neg \varphi$

- Examples:
  - Velocity will be non-negative until a collision occurs
    $v \geq 0 \ \mathrm{U}_{[0,\infty]} \ x \geq L$
  - Collision will not occur
    $\Box_{[0,\infty]} \ x < L$
    - Its negation is a **reachability property**

STL Semantics

- STL formulas are evaluated over execution traces
  - A trace is a set of signals
  - Signal is the value of a variable as a function of time: $\mathbb{R}^{\geq 0} \rightarrow \mathbb{R} \cup \{\perp, \top\}$
- From runs to traces

$$(\ell_0, \mathbf{x}_0) \xrightarrow{\delta_0, \xi_0} (\ell_0, \xi_0(\delta_0)) \xrightarrow{\alpha_0} (\ell_1, \mathbf{x}_1) \xrightarrow{\delta_1, \xi_1} (\ell_1, \xi_1(\delta_1)) \ldots$$

  - For $0 \leq t \leq \delta_0$, $w(t) = \xi_0(t-\delta_0-\ldots-\delta_i)$
  - For $\delta_0+\ldots+\delta_i \leq t \leq \delta_0+\ldots+\delta_{i+1}$, $w(t) = \xi_{i+1}(t-\delta_0-\ldots-\delta_i)$

Boolean STL Semantics

Syntax: $\varphi := \mathrm{true} \mid x_i \geq 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathrm{U}_I \varphi.$

Boolean Semantics:

$$
\begin{array}{lll}
w, t \models \mathrm{true} & & \\
w, t \models x_i \geq 0 & \text{iff} & x_i(t) \geq 0 \\
w, t \models \neg\varphi & \text{iff} & w, t \not\models \varphi \\
w, t \models \varphi \wedge \psi & \text{iff} & w, t \models \varphi \text{ and } w, t \models \psi \\
w, t \models \varphi \,\mathrm{U}_I\, \psi & \text{iff} & \exists t' \in t + I : w, t' \models \psi \wedge \\
& & \qquad \forall t'' \in [t, t'] : w, t'' \models \varphi
\end{array}
$$

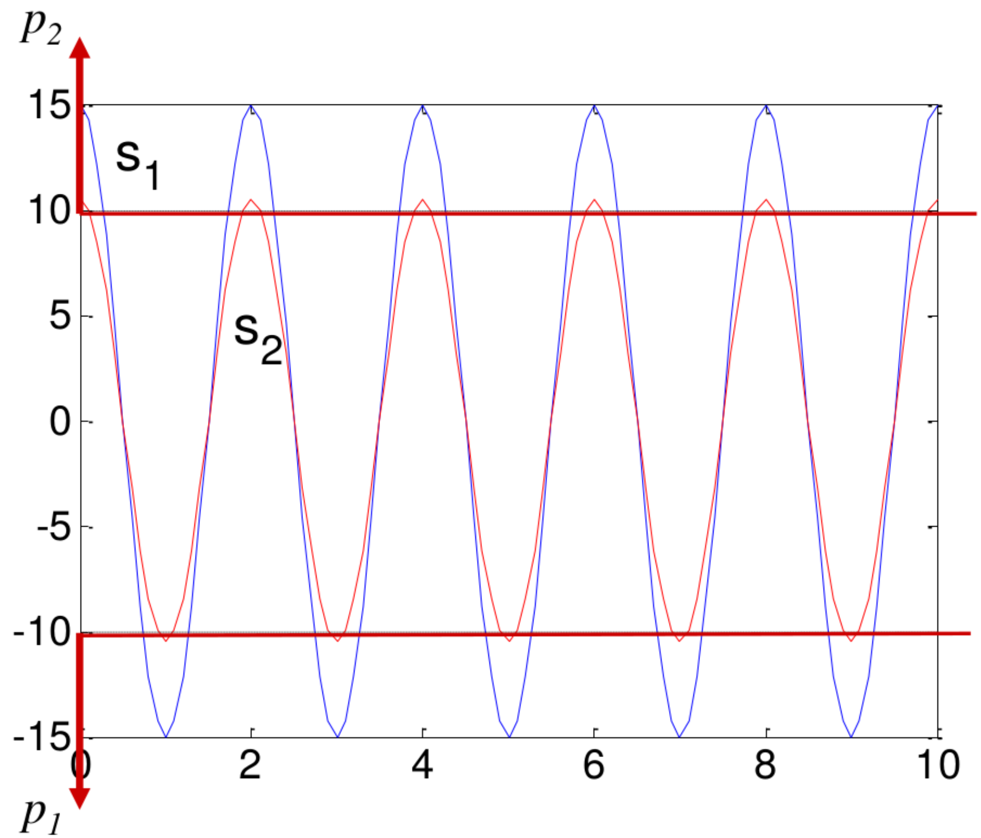
Penn Engineering

PRECISE

PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Robustness

- Whenever the signal is below -10 [p_1: x<-10], it will be above 10 within 2 seconds [p_2: x>10]
  - $\Box(p_1 \rightarrow \Diamond_{[0,2]} p_2)$
- Both s_1 and s_2 satisfy the property
- s_2 is not robust
  - Small perturbation will lead to violation



Robustness STL Semantics

Syntax: $\varphi := \mathrm{true} \mid x_i \geq 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathrm{U}_I \varphi.$

Quantitative Semantics: robustness estimation

$$
\begin{aligned}
\rho(\mathrm{true}, w, t) &= \top \\
\rho(x_i \geq 0, w, t) &= x_i(t) \\
\rho(\neg\varphi, w, t) &= -\rho(\varphi, w, t) \\
\rho(\varphi \wedge \psi, w, t) &= \min\{\rho(\varphi, w, t), \rho(\psi, w, t)\} \\
\rho(\varphi \,\mathrm{U}_I\, \psi, w, t) &= \sup_{t' \in t+I} \min\Big\{\rho(\psi, w, t'), \\
&\qquad\qquad \inf_{t'' \in [t,t']} \rho(\phi, w, t'')\Big\}
\end{aligned}
$$

Final Comments on Properties

- Checking liveness for hybrid systems is very hard
  - Need to reason about infinite behaviors
    - I.e., loops in the state space
- Bounded liveness is safety
  - A mission will be completed in 10 minutes
- Any safety property can be reduced to reachability checking
  - Using an “observer” technique – ask me after class
- Bottom line

Reachability is what we usually check

Reachability Analysis

- Two main techniques
  - Set-based reachability
  - Simulation-based reachability
- Set-based reachability
  - Overapproximation
    - “Safe” means safe
  - Primarily used for verification
- Simulation-based reachability
  - Underapproximation
    - “Unsafe” means unsafe
  - Primarily used for falsification

Outline

- Hybrid systems modeling with hybrid automata
- Property specification with STL
- Set-based reachability analysis
- Simulation-based analysis

Set-Based Reachability

- Extend numerical simulation from points to sets
- Building blocks:

One-step successors by time elapse from set of states S,

$$\mathrm{Post}_C(S) = \left\{(\ell, \xi(\delta)) \mid \exists(\ell, x) \in S : (\ell, \mathbf{x}) \xrightarrow{\delta,\xi} (\ell, \xi(\delta))\right\}.$$

One-step successors by jump from set of states S,

$$\mathrm{Post}_D(S) = \Big\{(\ell', \mathbf{x}') \mid \exists(\ell', \mathbf{x}') \in S, \exists\alpha \in \mathrm{Lab} \cup \{\tau\} : (\ell, \mathbf{x}) \xrightarrow{\alpha} (\ell', \mathbf{x}')\Big\}$$

Reachability Algorithm

- Basic algorithm:

  $R_0 = \text{Post}_C(\text{Init})$

  iterate

  $R_{i+1} = R_i \cup \text{Post}_C(\text{Post}_D(R_i))$

  until $R_{i+1} = R_i$

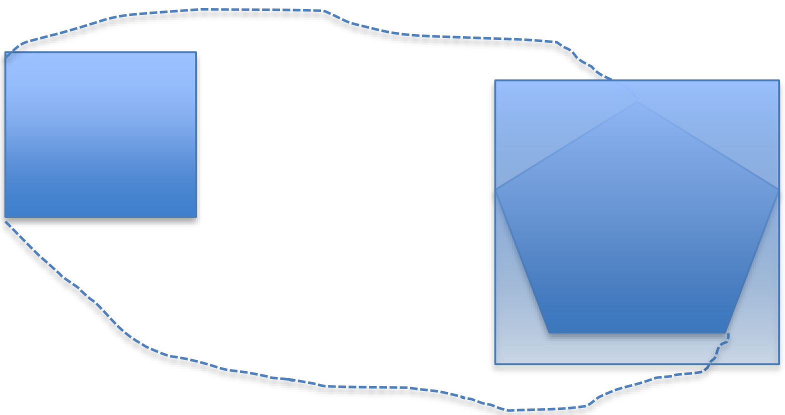
- *Deceptively* simple
  - If state space is unbounded, may not terminate
  - If jumps are nondeterministic, need to keep a queue of unprocessed states
  - Usually require an additional termination condition
    - Bound elapsed time or number of jumps

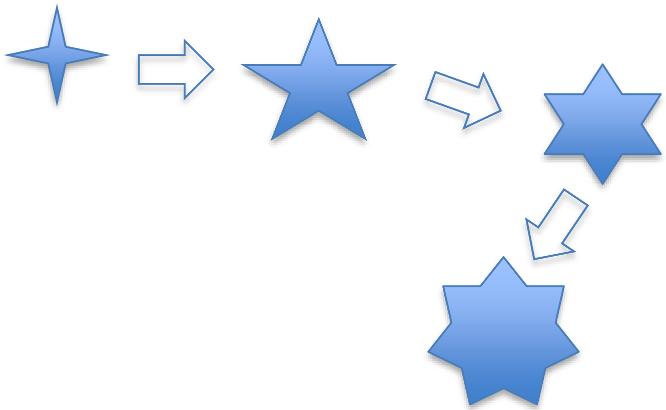
Implementing reachability

- Need to choose
  - Representation of *R*
  - Implementation of $Post_C$ and $Post_D$
- Depends on the continuous dynamics
  - In most cases, $Post_C$ cannot be computed exactly
  - Need to ensure overapproximation
- Tradeoff between accuracy and scalability

State Set Representations

- Key requirement: shape preservation

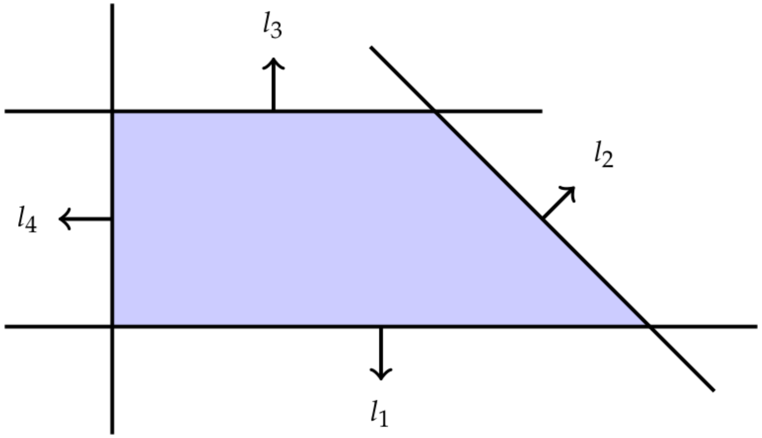




  - If the shape does not fit, need to overapproximate
  - If the shape gets complex, scalability suffers
- Common state representation for linear systems
  - Polyhedra
  - Ellipsoids

Polyhedra

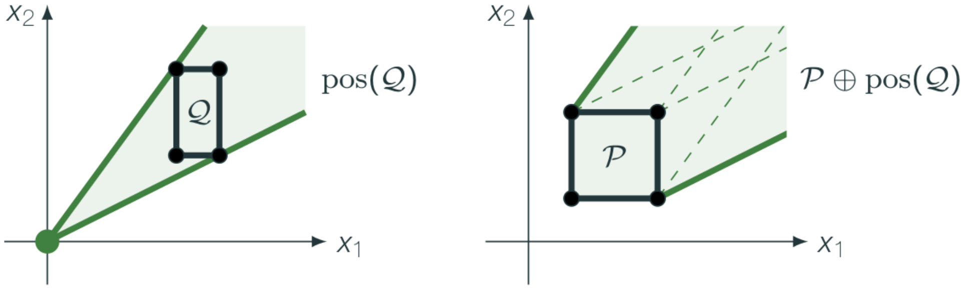
- Halfspace: $\{\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$
- Polyhedron: intersection of finitely many halfspaces



- Convex polyhedra
  - Much easier to manipulate
  - Usually introduce more conservatism
    - Or require you to have more of them

Important Classes of Hybrid Systems

- Piecewise-constant dynamics
  - Derivatives of state variables are constrained by constants
  - State set representation: convex polyhedra
- Post_C and Post_D can be computed exactly



(a) cone $\text{pos}(\mathcal{Q})$ (b) $\mathcal{P} \nearrow \mathcal{Q} = \mathcal{P} \oplus \text{pos}(\mathcal{Q})$

Intersect with invariant:

$$\text{post}_C(\ell \times P) = \ell \times \left(P \nearrow \text{Flow}(\ell)\right) \cap \text{Inv}(\ell).$$

$$\text{post}_D(\ell \times P) = \ell' \times \left(C(\mathcal{P} \cap \mathcal{G}) \oplus \mathcal{W}\right) \cap \text{Inv}(\ell')$$

Penn Engineering

PRECISE

Important Classes of Hybrid Systems

- Linear hybrid automata
  - Affine dynamics
  - Linear assignments in jumps
  - Initial states and invariants are convex polyhedra

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \qquad \mathbf{u} \in \mathcal{U}$$
$$\mathbf{x}' = C\mathbf{x} + \mathbf{w}, \qquad \mathbf{w} \in \mathcal{W}.$$

- Solutions to affine ODEs are exponential functions
  - Need to incorporate effects of inputs

$\xi(t)$ from $\xi(0) = \mathbf{x}_0$ for given input signal $\zeta(t) \in \mathcal{U}$

$$\xi_{\mathbf{x}_0,\zeta}(t) = e^{At}\mathbf{x}_0 + \int_0^t e^{A(t-s)}B\zeta(s)ds.$$

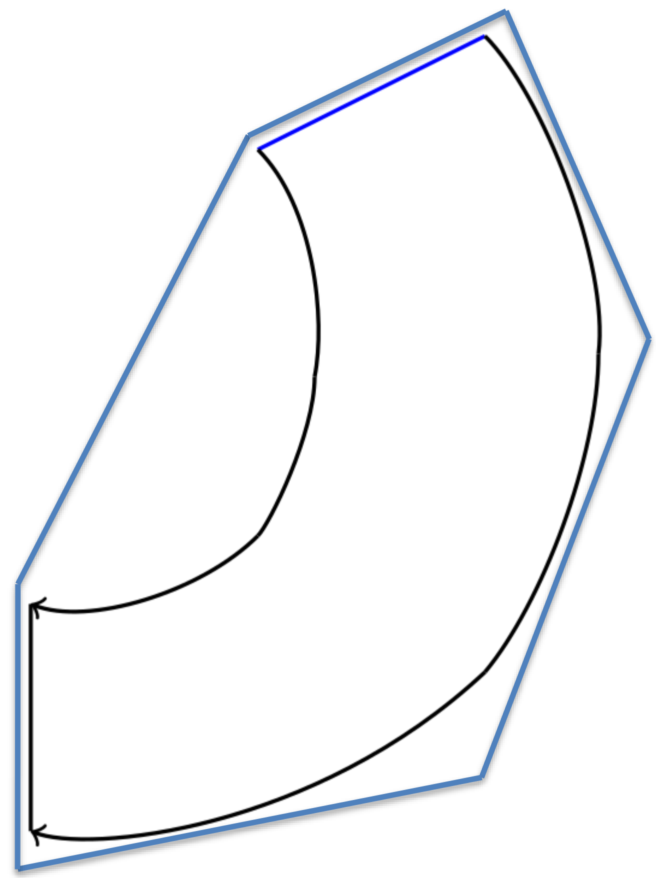
reachable states from set $\mathcal{X}_0$ for any input signal:

$$\mathcal{X}_t = e^{At}\mathcal{X}_0 \oplus \mathcal{Y}_t,$$

$$\mathcal{Y}_t = \int_0^t e^{As}\mathcal{U}ds = e^{At}\mathcal{X}_0 \oplus \lim_{\delta\to 0} \bigoplus_{k=0}^{\lfloor t/\delta \rfloor} e^{A\delta k}\delta\mathcal{U}$$

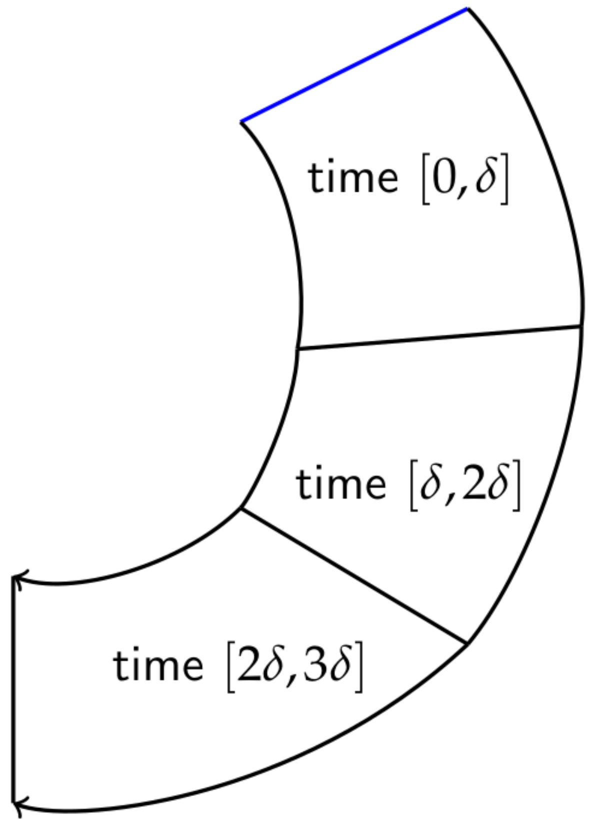
Computing Post_D

- A flow can be approximated by a single polyhedron
  - Too conservative



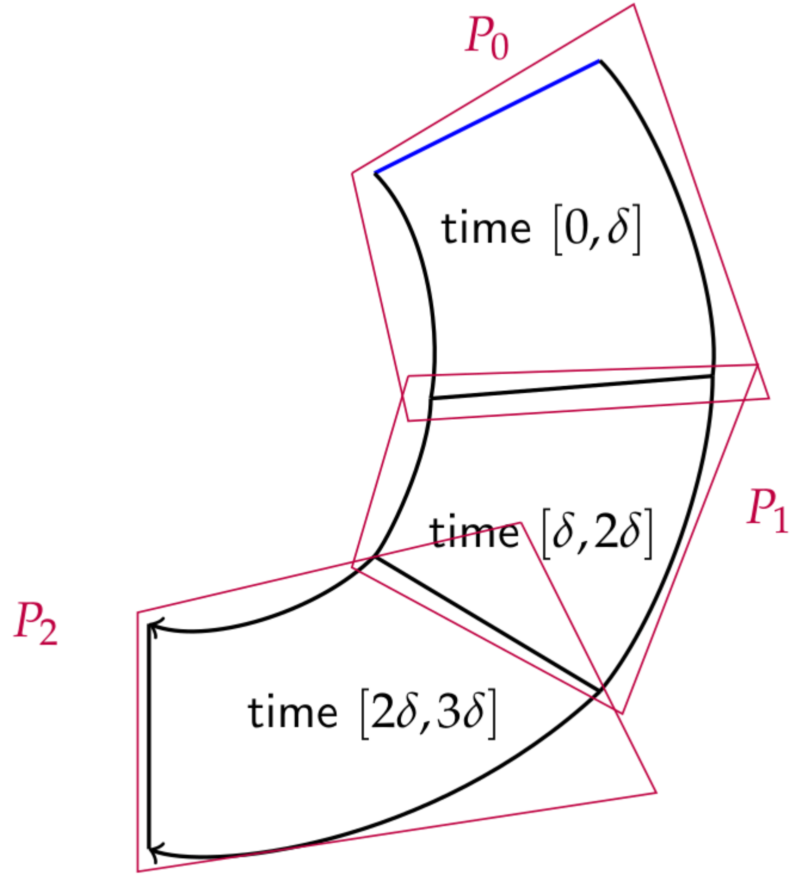
Computing $Post_D$

- A flow can be approximated by a single polyhedron
  - Too conservative
- Instead, select a time step δ
- Partition flow into time slices



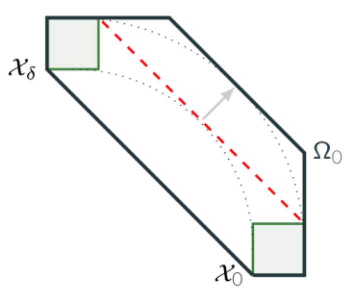
Computing Post$_D$

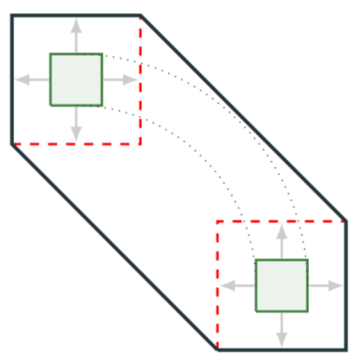
- A flow can be approximated by a single polyhedron
  - Too conservative
- Instead, select a time step δ
- Partition flow into time slices
- Approximate each slice by a polyhedron

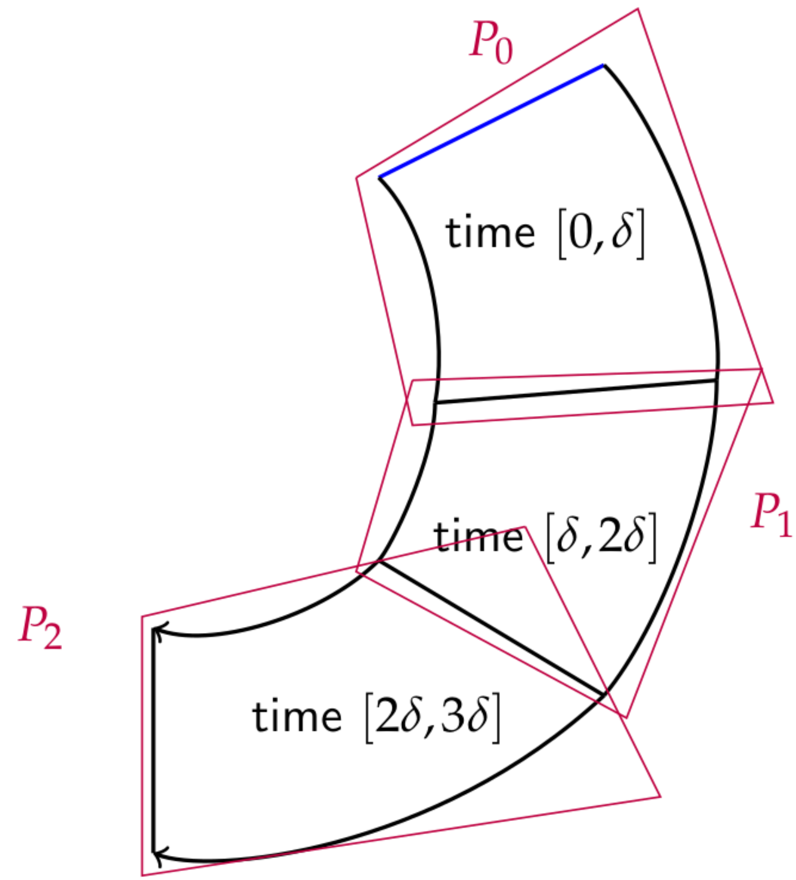


Computing Post$_D$

- A flow can be approximated by a single polyhedron
  - Too conservative
- Instead, select a time step δ
- Partition flow into time slices
- Approximate each slice by a polyhedron

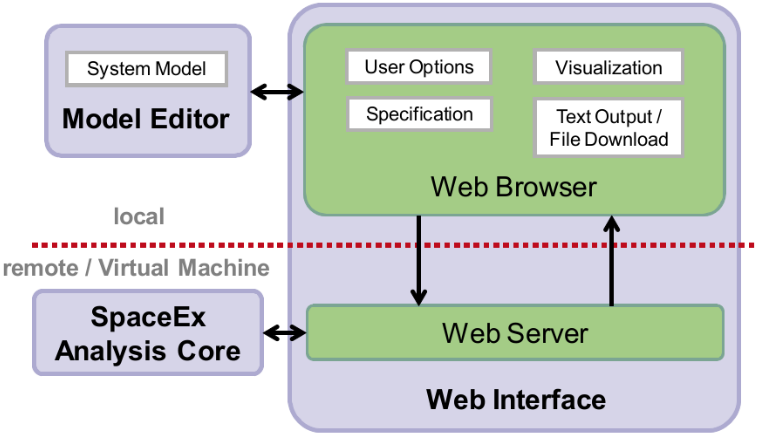


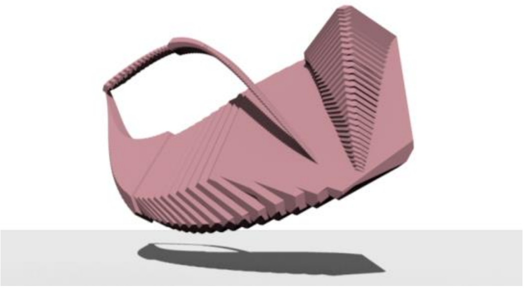


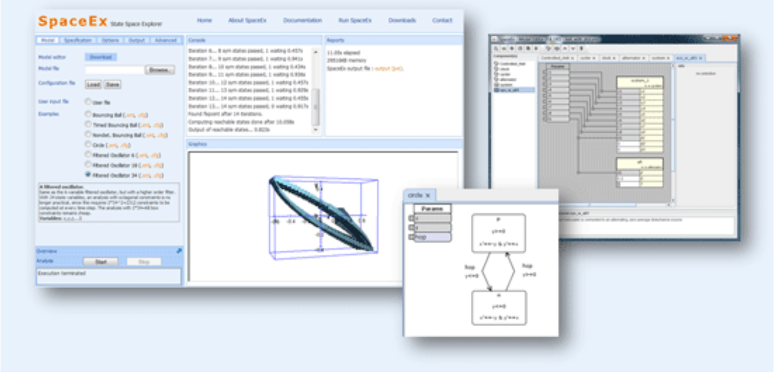


SpaceEx

- Extensible platform for verification of continuous and hybrid systems
- Based on a web interface
- Modules
  - Visual model editor
  - Analysis engine
    - APIs for new state representations and set operation implementations
  - Visualization

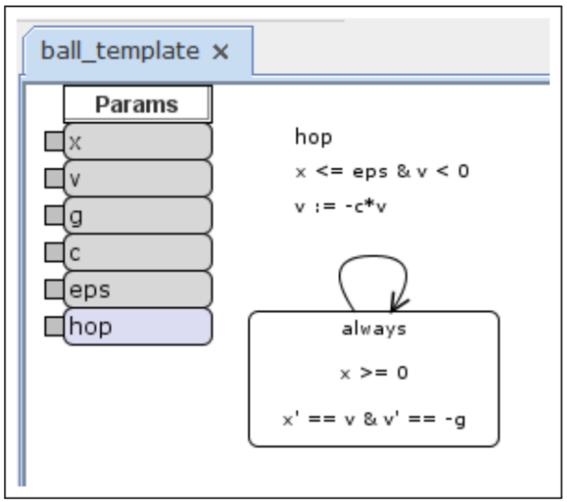




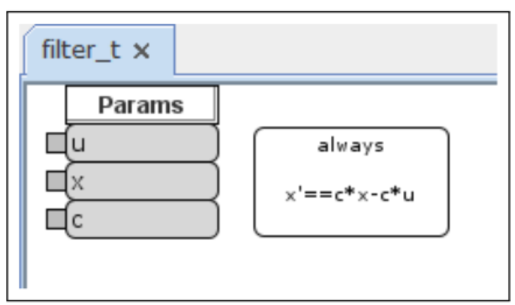


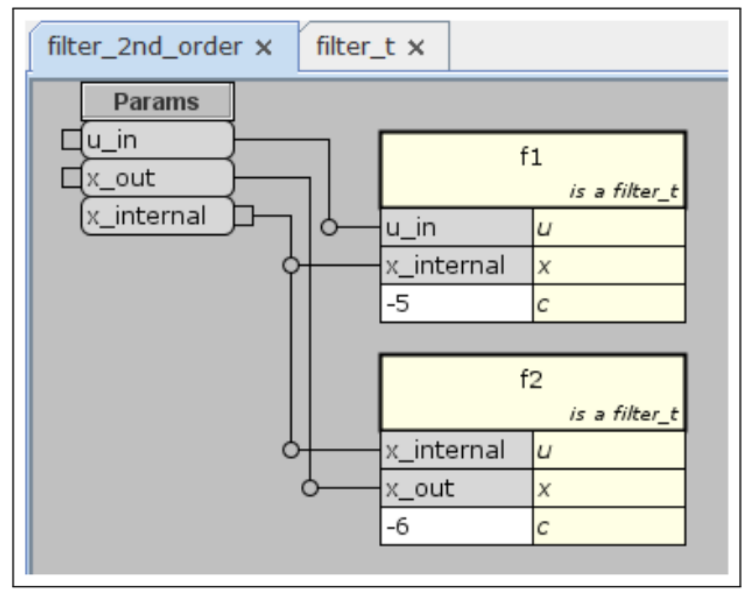
SpaceEx Models

- Networks of hybrid automata
- Automata specified as parameterized templates



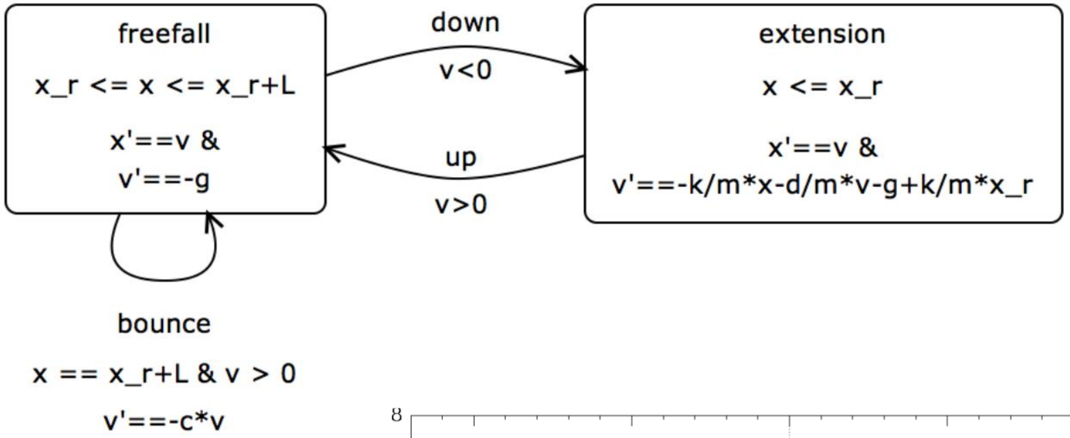
http://spaceex.imag.fr/



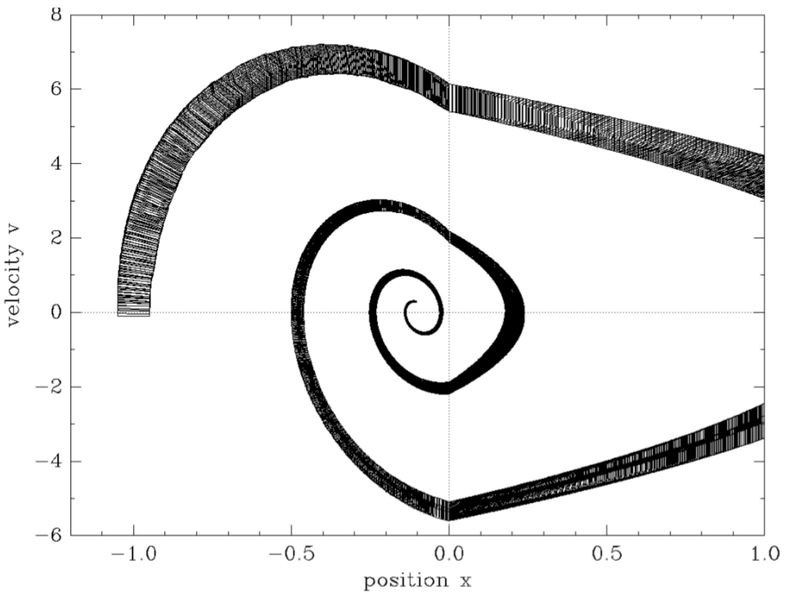


Bouncing Ball in SpaceEx

- Hybrid automaton model



- Phase portrait



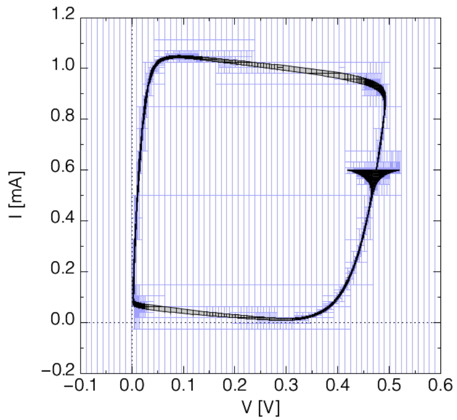
http://spaceex.imag.fr/

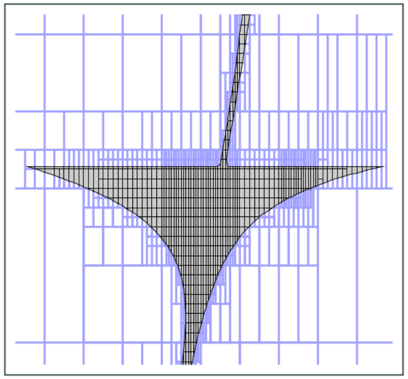
Non-Linear Hybrid Systems

- What's missing?
  - Mathematical results that yield geometric abstractions for successor computation
- Polynomial hybrid systems
  - Some support in SpaceEx
- Approximation of non-linear dynamics
  - Polynomial approximations
    - Taylor models – polynomial approximations of Taylor expansions
    - Flow* tool – next lecture
  - Hybridization
    - Local approximation with simpler dynamics

Hybridization

- Split a flow into multiple subflows with simpler dynamics
  - Time-based or space-based
- Can be done statically or adaptively
  - Trade-off between accuracy and scalability

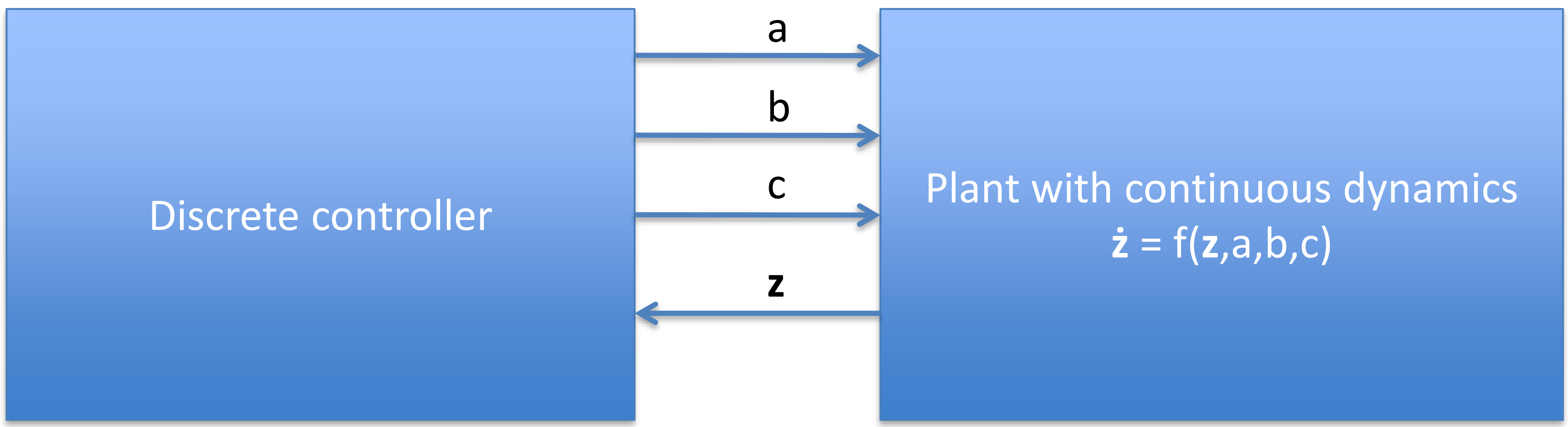




- Hyst – hybrid system model transformation tool
  - Implements static hybridization
  - Generates SpaceEx format
  - http://verivital.com/hyst/

Implicit Hybrid Systems

- Most systems are not designed as hybrid automata



  - E.g., Simulink/Stateflow, and many other design tools
- Extracting a hybrid automaton is not easy
  - Requires an accurate and complete translator
  - May not scale

Outline

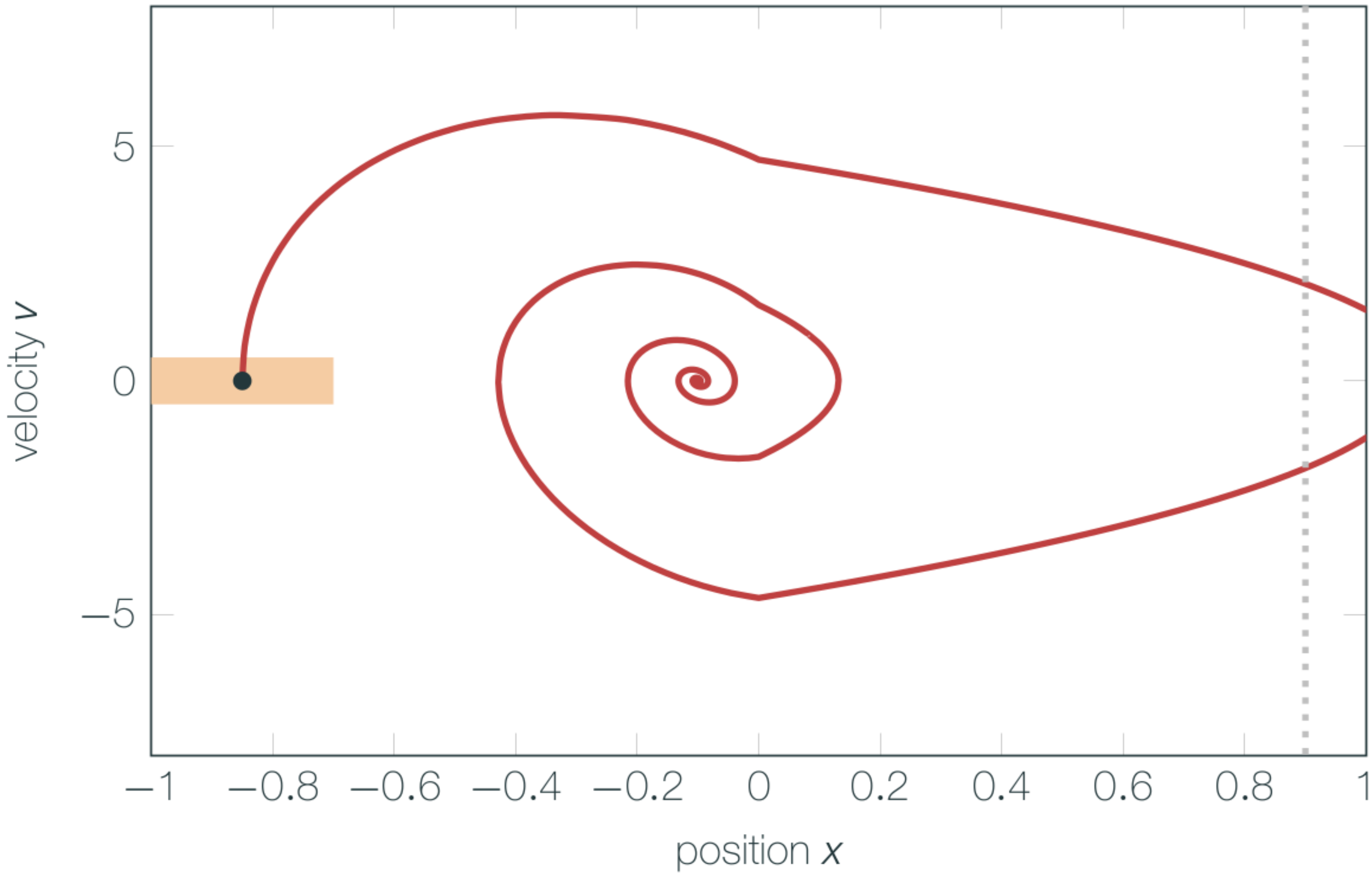
- Hybrid systems modeling with hybrid automata
- Property specification with STL
- Set-based reachability analysis
- Simulation-based analysis

Simulation-Based Analysis

- Reachability and beyond
- Works on implicit hybrid systems
  - Use simulator within the design tool
  - Or even a real implementation
- Any problem found through simulation is a real problem
  - No inherent conservatism
  - Assuming that simulations are accurate enough
- From flows to sampled traces
  - Analysis becomes monitoring of a trace
- Main challenge
  - Coverage

Simulation-Based Analysis

- Trace violates property □x≤0.9

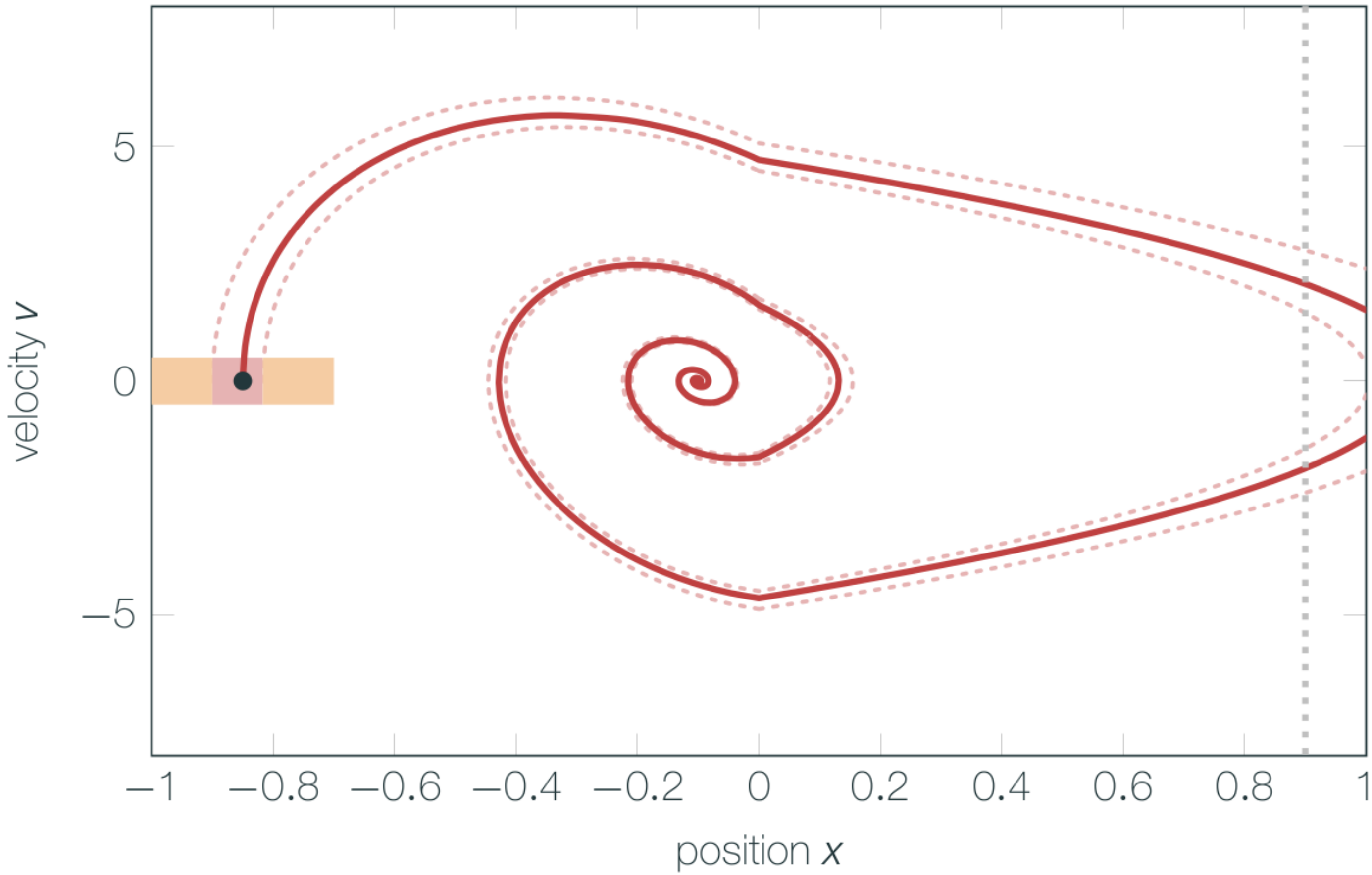


Penn Engineering

PRECISE

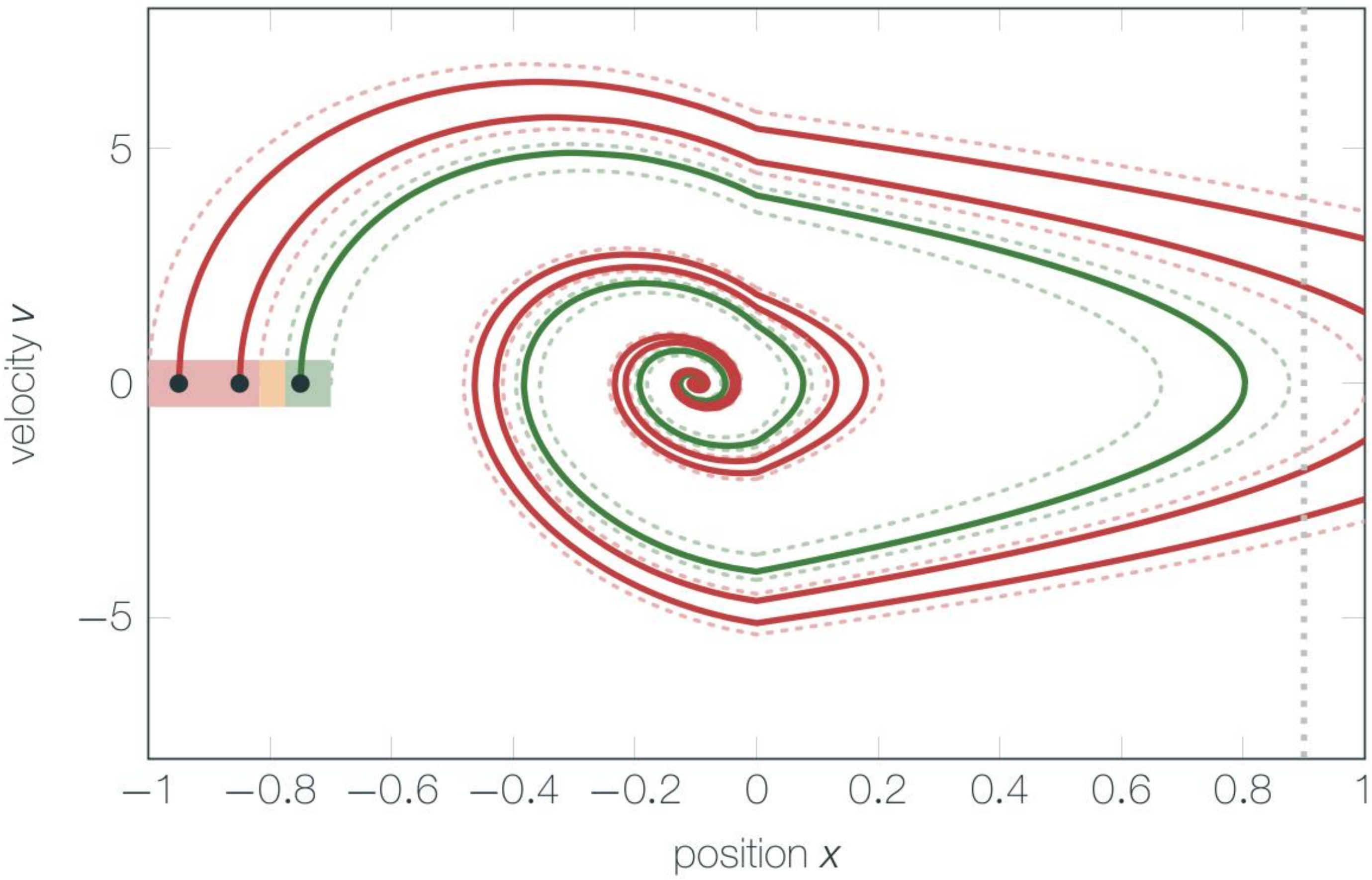
Simulation-Based Analysis

- Find equivalent initial states



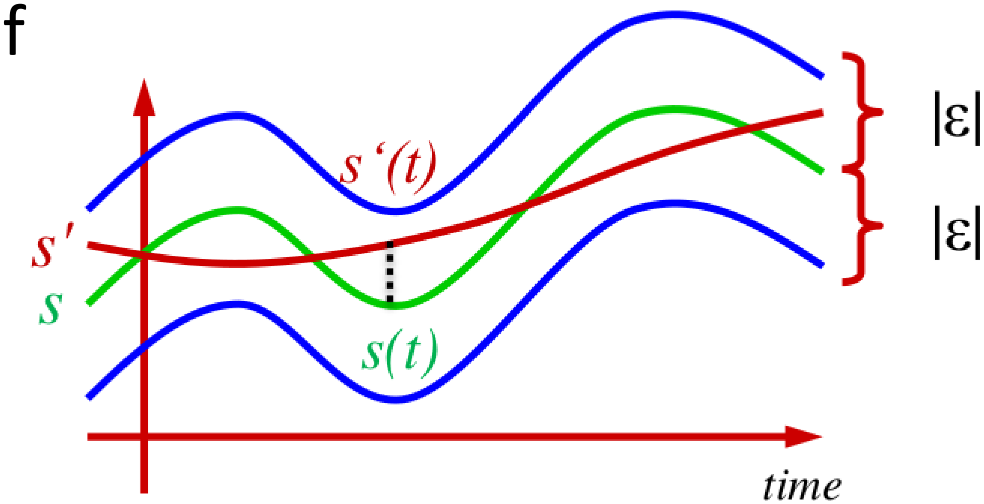
Simulation-Based Analysis

- Repeat until desired coverage is reached



The Importance of Robustness

- Allows us to extend analysis from one simulation to a set of simulations
- If a trace *s* satisfies a property φ with roustness ε
  - Every trace *s'* no more than ε away from *s* also satisfies φ
- Calculate how variation of initial state influences robustness
- Breach
  - Matlab toolbox
  - Use sensitivity from the ODE solver

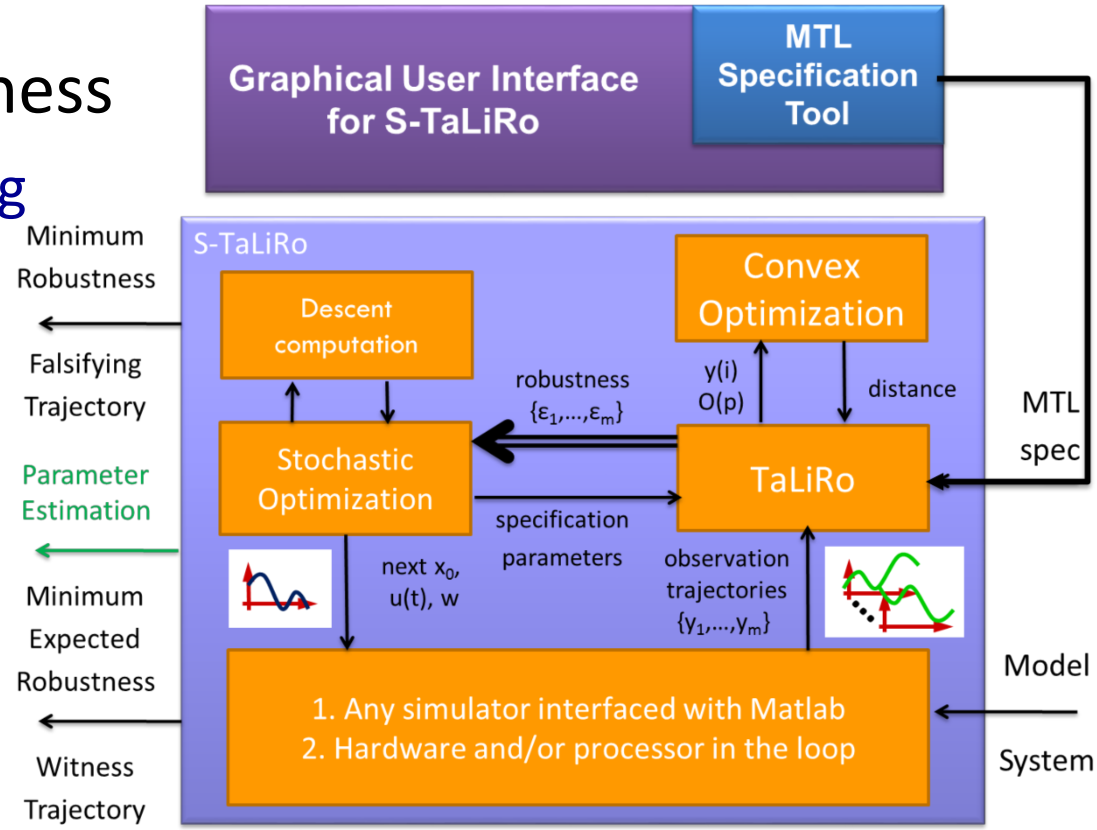


Falsification

- (Historic) goal for verification tools
  - Scale up towards exhaustive exploration
- Exhaustive verification rarely succeeds
  - But verification tools are very efficient in finding bugs
- Goal for falsification tools
  - Improve search for counterexamples
- Robustness-driven falsification
  - Select next initial state to minimize robustness
  - Tool: S-TaLiRo

S-TaLiRo

- Falsification based on robustness minimization
  - Matlab toolbox
- Calculation of robustness
  - Dynamic programming formulation
- Calculation of simulation inputs
- Closing the loop
  - Simulator
  - HW in the loop



https://sites.google.com/a/asu.edu/s-taliro/

Robustness Exploration

System:

$$dx/dt = x-y+0.1t$$

$$dy/dt = y\cos(2\pi y)-x\sin(2\pi x)+0.1t$$

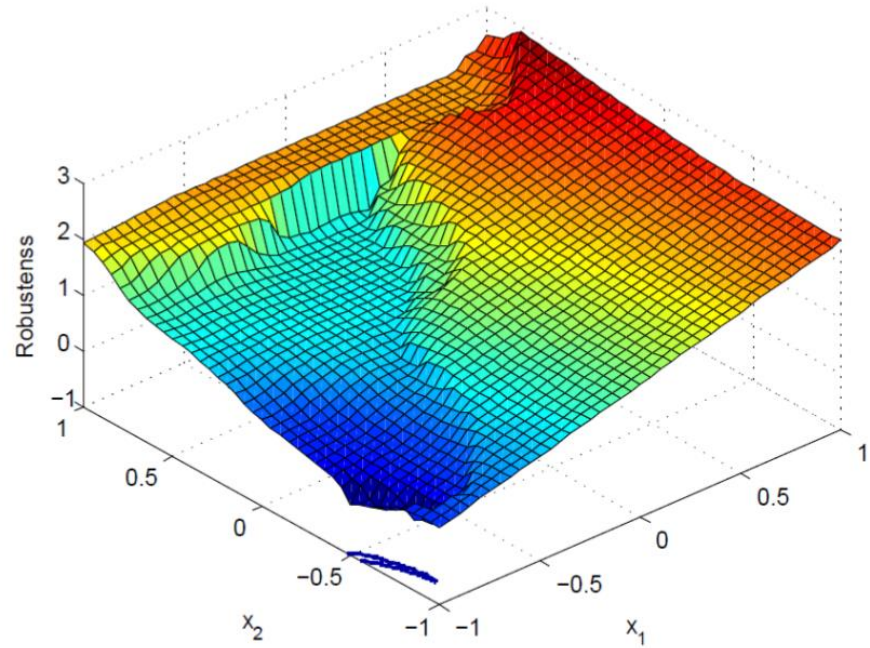
Initial conditions:

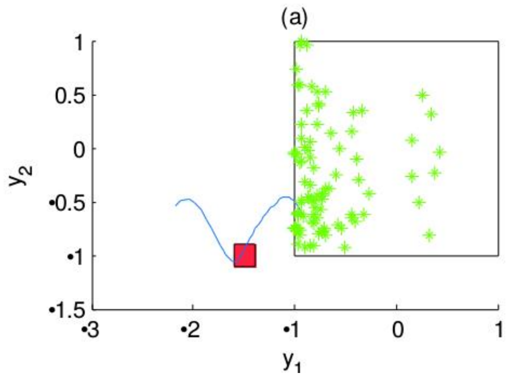
$$[-1,1]\times[-1,1]$$

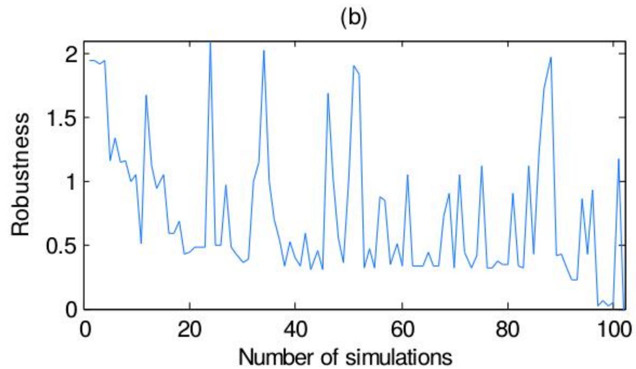
Specification:

$$G_{[0,2]} \neg a$$

where $O(a) = [-1.6,-1.4]\times[-1.1,-.9]$







Summary

- Model checking techniques allow us to analyze properties of system models
- Hybrid systems are widely used to model cyber-physical systems
  - Express rich sets of continuous and discrete behaviors
- Set-based reachability supported by many tools
  - Variety of linear and non-linear dynamics
  - Many different set representations
- Simulation-based analysis allows to check more complex properties
  - Enables effective falsification techniques for black-box systems