Formal Verification of Cyber-Physical Systems

Pavithra Prabhakar
IMDEA Software Institute

Cyber-Physical Systems Summer School
EIT ICT Labs - PERSYVAL Lab
Grenoble, France
July 09, 2013
Cyber-Physical Systems

Computation
Control
Communication

Safety-criticality
Recalls due to Software Bugs

February 6, 2010:

Toyota recalls 133,000 Prius vehicles in the US and 52,000 in Europe to fix problems with its anti-lock brake software.

1990-2000:

200,000 devices affected due to safety recalls of pacemakers and implantable cardioverter defibrillators due to firmware problems.
Grand Challenge: Development of high-confidence Cyber-Physical Systems
Model-based Design
Model-based Design
Model-based Design

- Model the plant

Diagram:

- Plant
- Controller
Model-based Design

- Model the plant
- Synthesize the controller
Model-based Design

- Model the plant
- Synthesize the controller
- Simulate/Verify
Model-based Design

- Model the plant
- Synthesize the controller
- Simulate/Verify
- Implement
Model-based Design

- Model the plant
- Synthesize the controller
- Simulate/Verify
- Implement

Elimination of errors early in the design, resulting in more robust control system, fewer iterations in the development cycle and reduced development time and cost.
Reliable design

Plant

Controller
Reliable design

- Automated synthesis: Correct-by-construction
Reliable design

- Automated synthesis: Correct-by-construction
- Automated methods for detecting presence/absence of errors
Reliable design

- Automated synthesis: Correct-by-construction
- Automated methods for detecting presence/absence of errors
  - Simulation/Testing
Reliable design

- Automated synthesis: Correct-by-construction
- Automated methods for detecting presence/absence of errors
  - Simulation/Testing
  - Verification
Verification of Cyber-Physical Systems
Hybrid Systems

Systems consisting of mixed discrete-continuous behaviors
Highlights

- Modeling and specification
- Overview of safety verification
- Overview of stability verification
- Complexity and computability
- Approximation techniques
- Tools
Models
Models

- Hybrid Automata [Henzinger et al.]
Models

- Hybrid Automata [Henzinger et al.]
  - Continuous dynamics: Differential equations
Models

- Hybrid Automata [Henzinger et al.]
  - Continuous dynamics: Differential equations
  - Discrete dynamics: Finite state automata
Models

- Hybrid Automata [Henzinger et al.]
  - Continuous dynamics: Differential equations
  - Discrete dynamics: Finite state automata
- Extensions of process algebra and petri-nets, differential dynamic logic
Autonomous Car Controller

\[ \omega = -\frac{\pi}{4} \]

\[ r = 2 \]
Autonomous Car Controller

\[ \omega = -\frac{\pi}{4} \]

\[ r = 2 \]

\[ x = \frac{1}{2} \]

\[ x' = x \]

Go Ahead
\[ \dot{x} = rsin(\gamma) \]
\[ \dot{\gamma} = 0 \]
\[ -1 \leq x \leq 1 \]

Turn Right
\[ \dot{x} = rsin(\gamma) \]
\[ \dot{\gamma} = \omega \]
\[ -2 \leq x \leq -1 \]

Out of the Road!
\[ x \leq -2 \]
\[ x' = x \]

\( -\frac{\pi}{4} \leq \gamma \leq \frac{\pi}{4} \)
\[ -1 \leq x \leq 1 \]
\[ x' = x \]
Autonomous Car Controller

\[ \omega = -\frac{\pi}{4} \]

\[ r = 2 \]

\[ -\frac{\pi}{4} \leq \gamma \leq \frac{\pi}{4} \]

\[ -1 \leq x \leq 1 \]

\[ x = -1 \]

\[ x' = x \]

\[ x \leq -2 \]

\[ x' = x \]

Go Ahead \[ \dot{x} = r \sin(\gamma) \]
\[ \dot{\gamma} = 0 \]
\[ -1 \leq x \leq 1 \]

Turn Right \[ \dot{x} = r \sin(\gamma) \]
\[ \dot{\gamma} = \omega \]
\[ -2 \leq x \leq -1 \]

Guard

Out of the Road!
Autonomous Car Controller

\[ \omega = -\frac{\pi}{4} \]

\[ r = 2 \]

\[ \gamma \]

\[ -1 \leq x \leq 1 \]

\[ -2 \leq x \leq -1 \]

Flow

- Go Ahead
  - \( \dot{x} = r \sin(\gamma) \)
  - \( \dot{\gamma} = 0 \)
- Turn Right
  - \( \dot{x} = r \sin(\gamma) \)
  - \( \dot{\gamma} = \omega \)

Guard

- Go Ahead
  - \( x = -1 \)
  - \( x' = x \)
- Turn Right
  - \( x \leq -2 \)
  - \( x' = x \)
- Out of the Road!
Autonomous Car Controller

\[ \omega = -\frac{\pi}{4} \]

\[ r = 2 \]

\[ \gamma \]

\[ -2 \leq x \leq -1 \]

\[ -1 \leq x \leq 1 \]

\[ 0 \]

\[ 1 \]

\[ 2 \]
Autonomous Car Controller

\[ \omega = -\frac{\pi}{4} \]

\[ r = 2 \]

\[ \gamma \]

Flow:

Go Ahead

\[ \dot{x} = rsin(\gamma) \]

\[ \dot{\gamma} = 0 \]

\[ -1 \leq x \leq 1 \]

Guard

Turn Right

\[ \dot{x} = rsin(\gamma) \]

\[ \dot{\gamma} = \omega \]

\[ -2 \leq x \leq -1 \]

Invariant

Out of the Road!

\[ x \leq -2 \]

\[ x' = x \]

Reset

\[ x = -1 \]

\[ x' = x \]
Properties
Properties

Safety
Properties

Safety

- Every execution of the system is error free
Properties

Safety

• Every execution of the system is error free
• The car does not go out of the road
Properties

Safety

- Every execution of the system is error free
- The car does not go out of the road

Stability
Properties

Safety

• Every execution of the system is error free

• The car does not go out of the road

Stability

• Small perturbations in the initial state or input lead to only small perturbations in the eventual behavior of the system
Properties

Safety

- Every execution of the system is error free
- The car does not go out of the road

Stability

- Small perturbations in the initial state or input lead to only small perturbations in the eventual behavior of the system
- Small perturbations in the initial orientation of the car will still keep it inside the road
Overview of safety verification
Finite-state systems & model-checking
Finite-state systems & model-checking

- Exhaustive state space exploration
Finite-state systems & model-checking

- Exhaustive state space exploration
- Compute one step-successors
Finite-state systems & model-checking

- Exhaustive state space exploration
- Compute one step-successors
- Terminates in a finite number of steps
Finite-state systems & model-checking

- Exhaustive state space exploration
- Compute one step-successors
- Terminates in a finite number of steps
Finite-state systems & model-checking

- Exhaustive state space exploration
- Compute one step-successors
- Terminates in a finite number of steps
Finite-state systems & model-checking

- Exhaustive state space exploration
- Compute one step-successors
- Terminates in a finite number of steps
Finite-state systems & model-checking

- Exhaustive state space exploration
- Compute one step-successors
- Terminates in a finite number of steps
A simple class of hybrid systems

\begin{align*}
0 < x < 2 \\
y = 0
\end{align*}

\begin{align*}
\dot{x} &= 1 \\
y &= 2
\end{align*}

\begin{align*}
x + 2y &= 2 \\
y &= 0
\end{align*}

\begin{align*}
x &\geq 0 \\
y &\geq 0
\end{align*}

\begin{align*}
x + 2y &\leq 2
\end{align*}
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity

\[
\begin{align*}
0 < x < 2 & \quad \quad y = 0 \\
\dot{x} &= 1 & \quad \quad \dot{y} &= 2 \\
\dot{x} &= 1 & \quad \quad \dot{y} &= -2 \\
x &\geq 0 & \quad \quad y &\geq 0 \\
x + 2y &\leq 2 & \quad \quad x + 2y &\leq 2
\end{align*}
\]
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity

\[
\begin{align*}
0 < x < 2 & \quad y = 0 \\
\dot{x} &= 1 \\
\dot{y} &= 2 \\
x + 2y &= 2 \\
x &\geq 0 \\
y &\geq 0 \\
x + 2y &\leq 2
\end{align*}
\]
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity

\[
\begin{align*}
0 < x &< 2 \\
y &= 0 \\
\dot{x} &= 1 \\
y &= 2 \\
x &\geq 0 \\
y &\geq 0 \\
x + 2y &\leq 2
\end{align*}
\]

\[
\begin{align*}
x + 2y &= 2 \\
y &= 0 \\
\dot{x} &= 1 \\
y &= -2 \\
x &\geq 0 \\
y &\geq 0 \\
x + 2y &\leq 2
\end{align*}
\]
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity

\[ 0 < x < 2 \]
\[ y = 0 \]
\[ x + 2y \leq 2 \]

\[ \dot{x} = 1 \]
\[ \dot{y} = 2 \]
\[ x \geq 0 \]
\[ y \geq 0 \]

\[ \dot{x} = 1 \]
\[ \dot{y} = -2 \]
\[ x \geq 0 \]
\[ y \geq 0 \]
\[ x + 2y \leq 2 \]

\[ 0,0 \quad 2,0 \]

\[ 0,1 \]
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity

\[
\begin{align*}
0 < x < 2 & \\
y = 0 & \\
\dot{x} = 1 & \\
y = 0 & \\
\dot{y} = 2 & \\
x + 2y = 2 & \\
x + 2y \leq 2 & \\
\dot{x} = 1 & \\
\dot{y} = -2 & \\
x \geq 0 & \\
y \geq 0 & \\
x + 2y \leq 2 & \\
\end{align*}
\]
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity
A simple class of hybrid systems

Continuous dynamics - constant derivative

Invariants and guards - linear

Resets - identity

\[ \begin{align*}
0 < x < 2 & \quad y = 0 \\
\dot{x} &= 1 & \dot{y} &= 2 \\
x + 2y &= 2
\end{align*} \]
A simple class of hybrid systems

- Continuous dynamics - constant derivative
- Invariants and guards - linear
- Resets - identity
Safety analysis - Reach set computation
Safety analysis - Reach set computation

- One-step successor:
  - states reached by time evolution or discrete transition
Safety analysis - Reach set computation

- One-step successor:
  - states reached by time evolution or discrete transition

\[
\text{Succ}_C(X_0, x) := \exists t, x_0 \in X_0, \quad x = x_0 + at, \\
\forall 0 \leq t' \leq t, x_0 + at' \in Inv
\]
Safety analysis - Reach set computation

- One-step successor:
  - states reached by time evolution or discrete transition

\[ \text{Succ}_C(X_0, x) := \exists t, x_0 \in X_0, \]
\[ x = x_0 + at, \]
\[ \forall 0 \leq t' \leq t, x_0 + at' \in \text{Inv} \]

\[ \text{Succ}_D(X_0, x) := \exists x_0 \in X_0, \]
\[ x_0 \in \text{Guard}, (x_0, x) \in \text{Reset} \]

Wednesday, July 17, 2013
Safety analysis - Reach set computation

- One-step successor:
  - states reached by time evolution or discrete transition

- first order logic formula with addition

\[ \text{Succ}_C(X_0, x) := \exists t, x_0 \in X_0, \]
\[ x = x_0 + at, \]
\[ \forall 0 \leq t' \leq t, x_0 + at' \in \text{Inv} \]

\[ \text{Succ}_D(X_0, x) := \exists x_0 \in X_0, \]
\[ x_0 \in \text{Guard}, (x_0, x) \in \text{Reset} \]
Safety analysis - Reach set computation

- One-step successor:
  - states reached by time evolution or discrete transition

- first order logic formula with addition

- polyhedral set

\[ \text{Succ}_C(X_0, x) := \exists t, x_0 \in X_0, \]
\[ x = x_0 + at, \]
\[ \forall 0 \leq t' \leq t, x_0 + at' \in \text{Inv} \]

\[ \text{Succ}_D(X_0, x) := \exists x_0 \in X_0, \]
\[ x_0 \in \text{Guard}, (x_0, x) \in \text{Reset} \]
Safety analysis - Reach set computation

\[ \text{Reach}^0(X_0) = X_0 \]
\[ \text{Reach}^{i+1}(X_0) = \text{Reach}^i(X_0) \cup \text{Succ}_D(\text{Succ}_C(\text{Reach}^i(X_0))) \]
Safety analysis - Reach set computation

\[
\begin{align*}
\text{Reach}^0(X_0) &= X_0 \\
\text{Reach}^{i+1}(X_0) &= \text{Reach}^i(X_0) \cup \text{Succ}_D(\text{Succ}_C(\text{Reach}^i(X_0)))
\end{align*}
\]

- Iterate till termination - need a check for termination - equivalence between sets
Safety analysis - Reach set computation

\[ \text{Reach}^0(X_0) = X_0 \]
\[ \text{Reach}^{i+1}(X_0) = \text{Reach}^i(X_0) \cup \text{Succ}_D(\text{Succ}_C(\text{Reach}^i(X_0))) \]

- Iterate till termination - need a check for termination - equivalence between sets
Safety analysis - Reach set computation

\[
\begin{align*}
\text{Reach}^0(X_0) &= X_0 \\
\text{Reach}^{i+1}(X_0) &= \text{Reach}^i(X_0) \cup \text{Succ}_D(\text{Succ}_C(\text{Reach}^i(X_0)))
\end{align*}
\]

- Iterate till termination - need a check for termination - equivalence between sets
- Terminates for some subclasses - Timed automata [Lecture on Friday - David & Larsen]
Safety analysis - Reach set computation

\[ \text{Reach}^0(X_0) = X_0 \]
\[ \text{Reach}^{i+1}(X_0) = \text{Reach}^i(X_0) \cup \text{Succ}_D(\text{Succ}_C(\text{Reach}^i(X_0))) \]

- Iterate till termination - need a check for termination - equivalence between sets

- Terminates for some subclasses - Timed automata [Lecture on Friday - David & Larsen]

- Reach set up to a given bound on the number of discrete transitions can be computed
Polyhedral Linear Systems
Polyhedral Linear Systems

- Generalized to polyhedral hybrid systems
Polyhedral Linear Systems

- Generalized to polyhedral hybrid systems
- Invariants, Guards - linear constraints
Polyhedral Linear Systems

- Generalized to polyhedral hybrid systems
  - Invariants, Guards - linear constraints
  - Resets - linear map
Polyhedral Linear Systems

- Generalized to polyhedral hybrid systems
  - Invariants, Guards - linear constraints
  - Resets - linear map
  - Dynamics - linear constraint over dotted variables
Polyhedral Linear Systems

- Generalized to polyhedral hybrid systems
  - Invariants, Guards - linear constraints
  - Resets - linear map
  - Dynamics - linear constraint over dotted variables
- Reachability analysis tools: HYTECH, PHAVER
Safety verification primitives
Safety verification primitives

- Effective representation of one step continuous successor
Safety verification primitives

- Effective representation of one step continuous successor
- Intersection with guards and resets
Safety verification primitives

- Effective representation of one step continuous successor
- Intersection with guards and resets
- Emptiness checking after intersection with the unsafe set
Safety verification primitives

• Effective representation of one step continuous successor
• Intersection with guards and resets
• Emptiness checking after intersection with the unsafe set

• Sets represents by polyhedral sets or formulas over first-order logic
• Emptiness checking reduces to satisfiability problem of the logic
  • Satisfiability of first order logic with addition and multiplication is decidable.
A more complex class
A more complex class

Linear Dynamical System \( \dot{x}(t) = ax(t) \)
A more complex class

Linear Dynamical System \[ \dot{x}(t) = ax(t) \]

Closed form solution \[ x(t) = e^{at}x(0) \]
A more complex class

Linear Dynamical System \[ \dot{x}(t) = ax(t) \]

Closed form solution \[ x(t) = e^{at} x(0) \]

\[ \frac{d}{dt} x(t) = ae^{at} x(0) = ax(t) \]
A more complex class

Linear Dynamical System \[ \dot{x}(t) = ax(t) \]

Closed form solution \[ x(t) = e^{at} x(0) \]

\[ \frac{d}{dt} x(t) = ae^{at} x(0) = ax(t) \]

Linear Dynamical System

\[ \dot{x}(t) = A\ddot{x}(t), \ddot{x}_0 \in X \subseteq \mathbb{R}^n \]
A more complex class

Linear Dynamical System: \( \dot{x}(t) = ax(t) \)

Closed form solution: \( x(t) = e^{at} x(0) \)

Linear Dynamical System: \( \dot{\bar{x}}(t) = A\bar{x}(t), \bar{x}_0 \in X \subseteq \mathbb{R}^n \)

Closed form solution: \( \bar{x}(t) = e^{At} \bar{x}(0) \)
A more complex class

Linear Dynamical System
\[ \dot{x}(t) = ax(t) \]

Closed form solution
\[ x(t) = e^{at}x(0) \]

\[ \frac{d}{dt}x(t) = ae^{at}x(0) = ax(t) \]

Linear Dynamical System
\[ \dot{\bar{x}}(t) = A\bar{x}(t), \bar{x}_0 \in X \subseteq \mathbb{R}^n \]

Closed form solution
\[ \bar{x}(t) = e^{At}\bar{x}(0) \]

\[ e^y = 1 + y + \frac{y^2}{2!} + \frac{y^3}{3!} + \cdots \]
A more complex class

Linear Dynamical System
\[ \dot{x}(t) = ax(t) \]

Closed form solution
\[ x(t) = e^{at}x(0) \]

\[ \frac{d}{dt}x(t) = ae^{at}x(0) = ax(t) \]

Linear Dynamical System
\[ \dot{x}(t) = A\bar{x}(t), \bar{x}_0 \in X \subseteq \mathbb{R}^n \]

Closed form solution
\[ \bar{x}(t) = e^{At}\bar{x}(0) \]

\[ e^y = 1 + y + \frac{y^2}{2!} + \frac{y^3}{3!} + \cdots \]

\[ e^B = 1 + B + \frac{B^2}{2!} + \frac{B^3}{3!} + \cdots \]
One-step successor for linear systems
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision

1. Divide into $\Delta$ time steps
2. Evaluate the function at $\Delta$ time steps to obtain a piecewise linear approximation
One-step successor for linear systems

• Satisfiability is not known for first-order logic with exponentiation.
• Approximation of the successor states
• Assumption: Matrix exponential can be computed with arbitrary precision

1. Divide into $\Delta$ time steps
2. Evaluate the function at $\Delta$ time steps to obtain a piecewise linear approximation
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision

1. Divide into \( \Delta \) time steps
2. Evaluate the function at \( \Delta \) time steps to obtain a piecewise linear approximation
3. Compute a bound on the error of approximation and expand
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision

1. Divide into $\Delta$ time steps
2. Evaluate the function at $\Delta$ time steps to obtain a piecewise linear approximation
3. Compute a bound on the error of approximation and expand
One-step successor for linear systems

- Satisfiability is not known for first-order logic with exponentiation.
- Approximation of the successor states
- Assumption: Matrix exponential can be computed with arbitrary precision
One-step successor for linear systems
One-step successor for linear systems

- Complexity of further verification
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
  - Number of intervals
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
    - Number of intervals
    - The data structure enclosing each step
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
    - Number of intervals
    - The data structure enclosing each step
- Data structures
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
    - Number of intervals
    - The data structure enclosing each step
- Data structures
  - Polyhedra [Dang, Maler], [Chutinan, Krogh], Ellipsoids [Kurzhanski, Varaiya], Zonotopes [Girard, Guernic]
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
    - Number of intervals
    - The data structure enclosing each step
- Data structures
  - Polyhedra [Dang, Maler], [Chutinan, Krogh], Ellipsoids [Kurzhanski, Varaiya], Zonotopes [Girard, Guernic]
- Varying time step algorithms
One-step successor for linear systems

- Complexity of further verification
  - Intersection computation and emptiness checking
  - Size and shape of the sets
    - Number of intervals
    - The data structure enclosing each step
- Data structures
  - Polyhedra [Dang, Maler], [Chutinan, Krogh], Ellipsoids [Kurzhanski, Varaiya], Zonotopes [Girard, Guernic]
- Varying time step algorithms
  - Approximate flow computation [Viswanathan], SpaceEx [Frehse et al.]
One-step successor for linear systems

Dynamic time-step computation:
Given an error bound find the length of the next time step
One-step successor for linear systems

Dynamic time-step computation:
Given an error bound find the length of the next time step
One-step successor for linear systems

Dynamic time-step computation:
Given an error bound find the length of the next time step
One-step successor for linear systems

Dynamic time-step computation:
Given an error bound find the length of the next time step
One-step successor for linear systems

Dynamic time-step computation:
Given an error bound find the length of the next time step
One-step successor for linear systems

Dynamic time-step computation:
Given an error bound find the length of the next time step
What about non-linear systems?

\[ \dot{x} = f(x) \]

\[ x \in X_0 \subseteq \mathbb{R}^n \]
What about non-linear systems?

\[ \dot{x} = f(x) \]

\[ x \in X_0 \subseteq \mathbb{R}^n \]

Closed form of the solutions do not exist in general!
What about non-linear systems?

\[
\dot{x} = f(x) \\
x \in X_0 \subseteq \mathbb{R}^n
\]

Closed form of the solutions do not exist in general!

- **Hybridization** [Puri, Borkar, Varaiya], [Asarin,Dang,Girard]
  - Finite partition of the state-space.
  - Approximate dynamics using the right hand side of the differential equation.
Hybridization - Rectangular approximation

\[
\begin{align*}
\dot{x}_1 &= f_1(x_1, x_2) \\
\dot{x}_2 &= f_2(x_1, x_2)
\end{align*}
\]
Hybridization - Rectangular approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]
Hybridization - Rectangular approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]

Find a rectangular approximation of \( f(x) \) in each cell.
Hybridization - Rectangular approximation

\[
\dot{x}_1 = f_1(x_1, x_2) \\
\dot{x}_2 = f_2(x_1, x_2)
\]

Find a rectangular approximation of \( f(x) \) in each cell

\[
\begin{array}{c|c}
(a, c) & (b, d) \\
\hline
\dot{x}_1 \in [l_1, u_1] \\
\dot{x}_2 \in [l_2, u_2] \\
\end{array}
\]
Hybridization - Rectangular approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]

Find a rectangular approximation of \( f(x) \) in each cell

Maximize \( f_1(x_1, x_2) \)
\[
\begin{align*}
    a & \leq x_1 \leq b \\
    c & \leq x_2 \leq d
\end{align*}
\]
Hybridization - Linear Approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]
Hybridization - Linear Approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]
Hybridization - Linear Approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]

Find a linear function which interpolates \( f(x) \) at the vertices \((a, c)\) and \((b, d)\).
Hybridization - Linear Approximation

\[ \dot{x}_1 = f_1(x_1, x_2) \]
\[ \dot{x}_2 = f_2(x_1, x_2) \]

Find a linear function which interpolates \( f(x) \) at the vertices (\( a, c \)) and (\( b, d \)).
Hybridization - Linear Approximation

\[
\dot{x}_1 = f_1(x_1, x_2) \\
\dot{x}_2 = f_2(x_1, x_2)
\]

Find a linear function which interpolates \( f(x) \) at the vertices

Bounded error approximation in a finite time interval by choosing small enough cells, for Lipschitz continuous functions
What about approximations for infinite time?
What about approximations for infinite time?

- Finite time
- Bounded error approximation
- Construct finer abstraction by reducing the error bound
What about approximations for infinite time?

- Finite time
- Bounded error approximation
- Construct finer abstraction by reducing the error bound
- Infinite time systems
What about approximations for infinite time?

- Finite time
  - Bounded error approximation
  - Construct finer abstraction by reducing the error bound
- Infinite time systems
  - Hybridization
What about approximations for infinite time?

- Finite time
  - Bounded error approximation
  - Construct finer abstraction by reducing the error bound
- Infinite time systems
  - Hybridization
  - Predicate Abstraction [Alur et al], [Tiwari]
What about approximations for infinite time?

- Finite time
  - Bounded error approximation
  - Construct finer abstraction by reducing the error bound
- Infinite time systems
  - Hybridization
  - Predicate Abstraction  [Alur et al], [Tiwari]
- In general, no bound on the error
What about approximations for infinite time?

- Finite time
  - Bounded error approximation
  - Construct finer abstraction by reducing the error bound

- Infinite time systems
  - Hybridization
  - Predicate Abstraction  [Alur et al], [Tiwari]

- In general, no bound on the error
- Refine based on a counter-example
Abstraction
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Abstraction
Abstraction
Abstraction
Abstraction
Abstraction
Abstraction

Wednesday, July 17, 2013
For every trajectory of the robot, there is a corresponding path in the abstract graph.
Abstraction
Abstraction
Abstraction

Right abstractions hard to find!
Refine the abstraction
Refine the abstraction
Refine the abstraction
Refinement

Refine the abstraction
Refine the abstraction
Refine the abstraction
Counter-Example Guided Abstraction Refinement

Diagram:

1. Abstract → Model-Check
2. Model-Check → Validate
3. Validate → Refine
4. Refine → Abstract

Wednesday, July 17, 2013
Counter-Example Guided Abstraction Refinement

Concrete System

Abstract -> Refine

Refine -> Validate

Validate -> Model-Check

Model-Check -> Abstract
Counter-Example Guided Abstraction Refinement

Concrete System → Abstract System → Model-Check → Validate → Refine → Abstract System → Concrete System
Counter-Example Guided Abstraction Refinement

Concrete System → Abstract System → Model-Check → Abstract System → Refine → Validate → Property
Concrete System satisfies the property!!
Counter-Example Guided Abstraction Refinement

Concrete System → Abstract System → Model-Check

- Yes: Concrete System satisfies the property!!
- No: Abstract System → Refine
  - Abstract Counterexample → Refine
  - Validate

Wednesday, July 17, 2013
Counter-Example Guided Abstraction Refinement

Concrete System $\rightarrow$ Abstract System $\rightarrow$ Model-Check $\rightarrow$ Validate $\rightarrow$ Concrete System satisfies the property!!

Concrete System $\rightarrow$ Abstract System $\rightarrow$ Refine $\rightarrow$ Abstract Counterexample $\rightarrow$ Validate $\rightarrow$ Concrete counterexample!!
Counter-Example Guided Abstraction Refinement

Concrete System

Abstract System

Abstract System

Model-Check

Property

Concrete System satisfies the property!!

Yes

No

Abstract Counterexample

No

Analysis

Refine

No

Validate

Concrete counterexample!!

Yes
Counter-Example Guided Abstraction Refinement

Concrete System satisfies the property!!
Related Work

• **Software Verification** [Kurshan et al. 93], [Clarke et al. 00], [Ball et al. 02]
  
  • SLAM, BLAST

• **Discrete CEGAR for hybrid systems** [Alur et al. 03], [Clarke et al. 03]

• **Hybrid CEGAR for hybrid systems** [P., Duggirala, Mitra, Viswanathan], [Dierks, Kupferschmid, Larsen]
Summary of safety verification

Complexity of Continuous Dynamics

Complexity of Verification

FSM
Summary of safety verification

Complexity of Continuous Dynamics

FSM

\[ \dot{x} = 1 \]

TIMED
Summary of safety verification

$\dot{x} = 1$

Exponential

FSM

TIMED

Complexity of Continuous Dynamics

Complexity of Verification
Summary of safety verification

\[ \dot{x} \in [a, b] \]

Complexity of Verification

- Exponential
- \( \dot{x} = 1 \)
- FSM
- TIMED
- RECTANGULAR

Complexity of Continuous Dynamics
Summary of safety verification

\[
\dot{x} = 1, \quad x \in [a, b]
\]

FSM

Exponential

TIMED

Undecidable

RECTANGULAR
Summary of safety verification

- FSM
- TIMED
- RECTANGULAR
- LINEAR (SpaceEx)
- NON-LINEAR (Flow*)

Complexity of Continuous Dynamics

Exponential

Undecidable

One step successor computation hard

\[ \dot{x} = 1 \]

\[ \dot{x} \in [a, b] \]

\[ \dot{x} = Ax \]

\[ \dot{x} = f(x) \]
Overview of stability verification
Stability

Small changes to the initial state of the system result in small changes to the behavior of the system

• The controlled behavior of the car depends gracefully on small variations to its starting orientation
Stability in a pendulum
Stability in a pendulum
Stability in a pendulum
Stability in a pendulum

Stable Equilibrium
Stability in a pendulum

Stable Equilibrium
Stability in a pendulum

Stable Equilibrium
Stability in a pendulum

Stable Equilibrium
Stability in a pendulum

Stable Equilibrium

Unstable Equilibrium
Lyapunov stability
Lyapunov stability
Lyapunov stability
Lyapunov stability
Lyapunov stability

\[ \forall \epsilon > 0 \exists \delta > 0 \left[ (\sigma(0) \in B_\delta(0)) \Rightarrow \forall t (\sigma(t) \in B_\epsilon(0)) \right] \]
Lyapunov stability

\[ \forall \epsilon > 0 \exists \delta > 0 \left[ (\sigma(0) \in B_\delta(0)) \Rightarrow \forall t (\sigma(t) \in B_\epsilon(0)) \right] \]

“Continuity of the transition relation at the origin”
Asymptotic stability

“Lyapunov stability + Convergence”
Asymptotic stability

“Lyapunov stability + Convergence”
Asymptotic stability

“Lyapunov stability + Convergence”
Asymptotic stability

“Lyapunov stability + Convergence”

\[ \exists \delta > 0 \left[ (\sigma(0) \in B_\delta(0)) \Rightarrow \text{Converge}(\sigma, 0) \right] \]

\[ \text{Converge}(\sigma, 0) \equiv \forall \varepsilon > 0, \exists T \geq 0, \sigma(t) \in B_\varepsilon(0), \forall t \geq T \]
Linear dynamical systems

\[
\dot{x}(t) = ax(t)
\]

\[
x(t) = e^{at} x(0)
\]
Linear dynamical systems

\[ \dot{x}(t) = ax(t) \]

\[ x(t) = e^{at} x(0) \]

If \( a \) is negative, \( x(t) \) converges to 0
Linear dynamical systems

\[ \dot{x}(t) = ax(t) \]

\[ x(t) = e^{at}x(0) \]

If \( a \) is negative, \( x(t) \) converges to 0

If \( a \) is positive, \( x(t) \) diverges
Linear dynamical systems

\[ \dot{x}(t) = ax(t) \]

\[ x(t) = e^{at} x(0) \]

If \( a \) is negative, \( x(t) \) converges to 0
If \( a \) is positive, \( x(t) \) diverges
If \( a \) is 0, \( x(t) \) is always \( x(0) \)
Linear dynamical systems

\[ \dot{x}(t) = A x(t) \]

\[ x(t) = e^{At} x(0) \]
Linear dynamical systems

\[ \dot{x}(t) = Ax(t) \]
\[ x(t) = e^{At} x(0) \]

\[
\begin{pmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3
\end{pmatrix} =
\begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 3
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
\]
Linear dynamical systems

\[
\dot{x}(t) = Ax(t)
\]

\[
x(t) = e^{At}x(0)
\]

Eigen value: \( \lambda \) such that there exists \( x \neq 0 \) with \( Ax = \lambda x \)
Linear dynamical systems

\[ \dot{x}(t) = Ax(t) \]
\[ x(t) = e^{At} x(0) \]

Eigen value: \( \lambda \) such that there exists \( x \neq 0 \) with \( Ax = \lambda x \)

The linear system is **stable** if all the eigen values of \( A \) have negative real parts
Linear dynamical systems

\[ \dot{x}(t) = Ax(t) \]

\[ x(t) = e^{At}x(0) \]

\[
\begin{pmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 
\end{pmatrix} =
\begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 3
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
\]

Eigen value: \( \lambda \) such that there exists \( x \neq 0 \) with \( Ax = \lambda x \)

The linear system is **stable** if

all the eigen values of \( A \) have negative real parts

The linear system is **unstable** if

\( A \) has at least one eigen value with positive real parts
Non-linear systems
Non-linear systems

Lyapunov’s first method - Linearization
Non-linear systems

Lyapunov’s first method - Linearization

System: \[ \dot{x} = F(x) \]

Equilibrium: \[ F(0) = 0 \]
Non-linear systems

Lyapunov’s first method - Linearization

System \[ \dot{x} = F(x) \]
Equilibrium \[ F(0) = 0 \]
Linearization \[ \dot{x} = Ax \]
Non-linear systems

Lyapunov’s first method - Linearization

System \[ \dot{x} = F(x) \]

Equilibrium \[ F(0) = 0 \]

Linearization \[ \dot{x} = Ax \]

A is the Jacobian of \( F \) evaluated at 0: \( A[i, j] = \frac{\partial F}{\partial x_i \partial x_j}(0) \)
Non-linear systems

Lyapunov’s first method - Linearization

System \[ \dot{x} = F(x) \]

Equilibrium \[ F(0) = 0 \]

Linearization \[ \dot{x} = Ax \]

\( A \) is the Jacobian of \( F \) evaluated at 0: \[ A[i, j] = \frac{\partial F}{\partial x_i \partial x_j}(0) \]

\( f(x) = x^2 + x \)

\[ A = (2x + 1)(0) = 1 \]
Non-linear systems

Lyapunov’s first method - Linearization

System \( \dot{x} = F(x) \)

Equilibrium \( F(0) = 0 \)

Linearization \( \dot{x} = Ax \)

A is the Jacobian of \( F \) evaluated at 0: \( A[i, j] = \frac{\partial F}{\partial x_i \partial x_j}(0) \)

\( f(x) = x^2 + x \)

\( A = (2x + 1)(0) = 1 \)

\( f(x) = x^2 - x \)

\( A = (2x - 1)(0) = -1 \)
Non-linear systems

Lyapunov’s first method - Linearization

System \[ \dot{x} = F(x) \]

Equilibrium \[ F(0) = 0 \]

Linearization \[ \dot{x} = Ax \]

A is the Jacobian of \( F \) evaluated at 0: \[ A[i, j] = \frac{\partial F}{\partial x_i \partial x_j}(0) \]

\[ f(x) = x^2 + x \]

\[ A = (2x + 1)(0) = 1 \]

\[ f(x) = x^2 - x \]

\[ A = (2x - 1)(0) = -1 \]

If \( A \) has is hyperbolic (all eigen values have non-zero real part), then stability of the original system is equivalent to the stability of the linearization.
Linear Switched Systems
Linear Switched Systems
Linear Switched Systems
Linear Switched Systems

Eigen value analysis doesn’t extend to mixed discrete continuous systems
Lyapunov’s Second Method

System  \[ \dot{x} = F(x) \]
Equilibrium  \[ F(0) = 0 \]
Solution  \[ \varphi(x, t) \]
Lyapunov’s Second Method

\[ \dot{x} = F(x) \]

System

Equilibrium \[ F(0) = 0 \]

Solution \[ \varphi(x, t) \]

If there exists a Lyapunov function for the system,
Lyapunov’s Second Method

System \[ \dot{x} = F(x) \]
Equilibrium \[ F(0) = 0 \]
Solution \[ \varphi(x, t) \]

If there exists a Lyapunov function for the system, then it is Lyapunov stable.
Lyapunov function: Illustration
Lyapunov function: Illustration
Lyapunov function: Illustration

Continuously differentiable
Lyapunov function: Illustration

- Continuously differentiable
- Positive Definite
Lyapunov function: Illustration

- Continuously differentiable
- Positive Definite
- Value decreases along any trajectory
Lyapunov function
Lyapunov function

Continuously differentiable

$V : \mathbb{R}^n \rightarrow \mathbb{R}^+$
Lyapunov function

Continuously differentiable

\[ V : \mathbb{R}^n \rightarrow \mathbb{R}^+ \]

Positive Definite

\[ V(x) = 0 \text{ if and only if } x = 0 \]
\[ V(x) \geq 0 \text{ for all } x \]
Lyapunov function

Continuously differentiable

\[ V : \mathbb{R}^n \to \mathbb{R}^+ \]

Positive Definite

\[ V(x) = 0 \text{ if and only if } x = 0 \]
\[ V(x) \geq 0 \text{ for all } x \]

Value along a trajectory decreases

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) \text{ is negative definite} \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(X) = 0 \text{ if and only if } x = 0 \]
Lyapunov function: Example

System \[ \dot{x} = -x \]
Lyapunov function: Example

System

\[ \dot{x} = -x \]

Candidate Lyapunov Function

\[ V(x) = x^2 \]
Lyapunov function: Example

System \[ \dot{x} = -x \]

Candidate Lyapunov Function

\[ V(x) = x^2 \]

\[ V(x) \geq 0 \text{ for all } x \]

\[ V(x) = 0 \text{ if and only if } x = 0 \]
Lyapunov function: Example

**System**

\[ \dot{x} = -x \]

**Candidate Lyapunov Function**

\[ V(x) = x^2 \]

\[ V(x) \geq 0 \text{ for all } x \]

\[ V(x) = 0 \text{ if and only if } x = 0 \]

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) = 2x \cdot (-x) \]
Lyapunov function: Example

System \[ \dot{x} = -x \]

Candidate Lyapunov Function

\[ V(x) = x^2 \]

\[ V(x) \geq 0 \text{ for all } x \]
\[ V(x) = 0 \text{ if and only if } x = 0 \]

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) = 2x \cdot (-x) \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(x) = 0 \text{ if and only if } x = 0 \]
Lyapunov function: Example

System
\[ \dot{x} = -x \]

Candidate Lyapunov Function
\[ V(x) = x^2 \]

\[ V(x) \geq 0 \text{ for all } x \]
\[ V(x) = 0 \text{ if and only if } x = 0 \]

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) = 2x \cdot (-x) \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(x) = 0 \text{ if and only if } x = 0 \]

\[ \bullet \text{ A quadratic Lyapunov function exists for every stable linear system} \]
Lyapunov function: Example

System
\[ \dot{x} = -x \]

Candidate Lyapunov Function
\[ V(x) = x^2 \]

\[ V(x) \geq 0 \text{ for all } x \]

\[ V(x) = 0 \text{ if and only if } x = 0 \]

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) = 2x \cdot (-x) \]

\[ \dot{V}(x) \leq 0 \text{ for all } x \]

\[ \dot{V}(x) = 0 \text{ if and only if } x = 0 \]

• A quadratic Lyapunov function exists for every stable linear system
• It can be computed by solving a linear matrix inequality
Lyapunov function: Example

System \[ \dot{x} = -x \]

Candidate Lyapunov Function
\[ V(x) = x^2 \]

\[ V(x) \geq 0 \text{ for all } x \]
\[ V(x) = 0 \text{ if and only if } x = 0 \]

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) = 2x \cdot (-x) \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(x) = 0 \text{ if and only if } x = 0 \]

- A quadratic Lyapunov function exists for every stable linear system
- It can be computed by solving a linear matrix inequality
- Such complete results do not exist for non-linear systems or hybrid systems (even in the linear case)
Lyapunov functions for hybrid systems
Lyapunov functions for hybrid systems

- Common Lyapunov function - a function which is a Lyapunov function for each mode of the system
Lyapunov functions for hybrid systems

- Common Lyapunov function - a function which is a Lyapunov function for each mode of the system
- Multiple Lyapunov function - a function for each mode along with certain conditions that need to be satisfied at the switching
Lyapunov functions for hybrid systems

- Common Lyapunov function - a function which is a Lyapunov function for each mode of the system
- Multiple Lyapunov function - a function for each mode along with certain conditions that need to be satisfied at the switching
- Reference: Switching in Systems and Control - Daniel Liberzon
Lyapunov function: Computation

Continuously differentiable
\[ V : \mathbb{R}^n \rightarrow \mathbb{R}^+ \]

Positive Definite
\[ V(x) = 0 \text{ if and only if } x = 0 \]
\[ V(x) \geq 0 \text{ for all } x \]

Value along a trajectory decreases
\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) \text{ is negative definite} \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(X) = 0 \text{ if and only if } x = 0 \]
Lyapunov function: Computation

Continuously differentiable

\[ V : \mathbb{R}^n \rightarrow \mathbb{R}^+ \]

Positive Definite

\[ V(x) = 0 \text{ if and only if } x = 0 \]

\[ V(x) \geq 0 \text{ for all } x \]

Value along a trajectory decreases

\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) \text{ is negative definite} \]

\[ \dot{V}(x) \leq 0 \text{ for all } x \]

\[ \dot{V}(X) = 0 \text{ if and only if } x = 0 \]
Lyapunov function: Computation

Continuously differentiable
\[ V : \mathbb{R}^n \rightarrow \mathbb{R}^+ \]

Positive Definite
\[ V(x) = 0 \text{ if and only if } x = 0 \]
\[ V(x) \geq 0 \text{ for all } x \]

Value along a trajectory decreases
\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) \text{ is negative definite} \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(X) = 0 \text{ if and only if } x = 0 \]
Lyapunov function: Computation

Continuously differentiable
\[ V : \mathbb{R}^n \rightarrow \mathbb{R}^+ \]

Positive Definite
\[ V(x) = 0 \text{ if and only if } x = 0 \]
\[ V(x) \geq 0 \text{ for all } x \]

Value along a trajectory decreases
\[ \dot{V}(x) = \frac{\partial V}{\partial x} F(x) \text{ is negative definite} \]
\[ \dot{V}(x) \leq 0 \text{ for all } x \]
\[ \dot{V}(X) = 0 \text{ if and only if } x = 0 \]

Check if there exist coefficients for which the polynomial is a sum-of-squares (SOS)

Express again as a sum-of-squares constraint
Abstraction refinement for stability
Abstraction refinement for stability

• An abstraction refinement framework for a systematic proof search
Abstraction refinement for stability

- An abstraction refinement framework for a systematic proof search
- Notions of abstraction not well-studied
Abstraction refinement for stability

• An abstraction refinement framework for a systematic proof search
• Notions of abstraction not well-studied
• Do the discrete abstraction techniques for safety work? No!
Abstraction refinement for stability

- An abstraction refinement framework for a systematic proof search
- Notions of abstraction not well-studied
- Do the discrete abstraction techniques for safety work? No!
- Modified predicate abstraction
Piecewise Constant Derivative System
Piecewise Constant Derivative System

\begin{align*}
  p_1 & : (-1, 1) \\
  p_2 & : (-1, -1) \\
  p_3 & : (1, -1) \\
  p_4 & : (1, 1)
\end{align*}
Piecewise Constant Derivative System

Lyapunov stable, but not asymptotically stable
Piecewise Constant Derivative System

\[ (-1/2, -1) \quad (1, -1) \]

\[ (1/2, 1) \quad (-1, 1) \]
Piecewise Constant Derivative System

\[ (-1/2, -1) \rightarrow (1, -1) \rightarrow (1/2, 1) \rightarrow (-1, 1) \]
Piecewise Constant Derivative System

Lyapunov stable and asymptotically stable
Piecewise Constant Derivative System
Piecewise Constant Derivative System

\[
\begin{align*}
(1, -1) & \quad (2, 1) \\
(1, 1) & \quad (-2, -1) \\
\end{align*}
\]
Neither Lyapunov stable nor asymptotically stable
Stability Analysis: Graph Construction
Stability Analysis: Graph Construction
Stability Analysis: Graph Construction
Stability Analysis: Graph Construction

Capture information about distance to the origin along the executions
Stability Analysis: Weight computation

Need to capture information about distance to the origin along the executions
Stability Analysis: Weight computation

Need to capture information about distance to the origin along the executions
Stability Analysis: Weight computation

Need to capture information about distance to the origin along the executions
Stability Analysis: Weight computation

Need to capture information about distance to the origin along the executions

$$w(e) = \frac{|d_2|}{|d_1|}$$
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Weight computation

\[ w(e) = \frac{|d_2|}{|d_1|} \]
Stability Analysis: Weighted graph
Stability Analysis: Weighted graph
Stability Analysis: Weighted graph

\[ \sigma = \sigma(p_{i_1}p_{i_2})\sigma(p_{i_2}p_{i_3}) \cdots \sigma(p_{i_{n-1}}p_{i_n}) \]
Stability Analysis: Weighted graph

\[ \sigma = \sigma(p_{i1}p_{i2})\sigma(p_{i2}p_{i3})\cdots\sigma(p_{i_{n-1}}p_{i_n}) \]

\[ w(\sigma) = \frac{d(\sigma(T))}{d(\sigma(0))} \]
Stability Analysis: Weighted graph

\[ \sigma = \sigma(p_{i_1}p_{i_2})\sigma(p_{i_2}p_{i_3}) \cdots \sigma(p_{i_{n-1}}p_{i_n}) \]

\[ w(\sigma) = \frac{d(\sigma(T))}{d(\sigma(0))} = w(e_{i_1,i_2})w(e_{i_2,i_3}) \cdots w(e_{i_{n-1},i_n}) \]
Lyapunov stable, but not asymptotically stable
Stability Analysis: Example 1

Lyapunov stable, but not asymptotically stable
Stability Analysis: Example 2

Lyapunov stable and asymptotically stable
Stability Analysis: Example 2

Lyapunov stable and asymptotically stable
Neither Lyapunov stable nor asymptotically stable
Neither Lyapunov stable nor asymptotically stable
Theorem: (Lyapunov stability)

A Piecewise Constant Derivative System is Lyapunov stable if the weighted graph does not contain any cycles with the product of weights > 1.
Stability Analysis: PCD

Theorem: (Lyapunov stability)
A Piecewise Constant Derivative System is Lyapunov stable if the weighted graph does not contain any cycles with the product of weights > 1

Theorem: (Asymptotic stability)
A Piecewise Constant Derivative System is asymptotically stable if the weighted graph does not contain any cycles with the product of weights >= 1
Stability Analysis Tools
Stability Analysis Tools

- SOSTOOLS - Sum-of-squares programming
Stability Analysis Tools

- SOSTOOLS - Sum-of-squares programming
- LMI solvers - CVX
Stability Analysis Tools

- SOSTOOLS - Sum-of-squares programming
- LMI solvers - CVX
- Stability Analysis: Stabhyli
Summary

- Hybrid Systems verification is challenging
  - Undecidable for simple subclasses
  - Standard techniques from the purely discrete and continuous worlds do not extend in a straightforward manner
- Model-checking & Deductive verification
- Approximation techniques
  - Safety - Predicate abstraction & CEGAR, hybridization
  - Stability - Linearization, Predicate abstraction
Some research directions

• Scalability
  • Efficient data structures
  • Approximation methods
  • Compositional analysis

• Applications
  • Exploiting structures

• Bridging the gap between model and implementation