www.unisi.it

1st HYCON PhD School on Hybrid Systems

www.ist-hycon.org

Verification of Hybrid Systems

George Pappas

University of Pennsylvania, USA

pappasg@central.cis.upenn.edu

Hybrid systems combine continuous dynamics (differential or difference equations) typical of physical plants and discrete dynamics (automata and logical conditions) typical of control logic. By combining disciplines of computer science and systems and control theory, research on hybrid systems provide a solid theory and computational tools for the analysis, simulation, verification, and control design of embedded systems", and are used in a large variety of applications (automotive systems, air traffic management, biological systems, process industries, and many others).

HYSCOM

IEEE CSS Technical Committee on Hybrid Systems

Information Society Technologies

Siena, July 19-22, 2005 - Rectorate of the University of Siena

Verification of hybrid systems

HYCON Summer School
on Hybrid Systems
Siena, Italy
July 19-22, 2005

George J. Pappas
Departments of ESE and CIS
University of Pennsylvania
pappasg@ee.upenn.edu
http://www.seas.upenn.edu/~pappasg

Thanks to

School Organizers

Alberto Bemporad

Maurice Heemels

and HYCON

Acknowledgments

Postdocs
- Antoine Girard
- Agung Julius

Ph.D Students
- Ali Ahmazadeh
- George Fainekos
- Hadas Kress Gazit
- Truong Nghiem
- Mahmut Serkar
- Hakan Yazarel
- Michael Zavlanos

Collaborators
Rajeev Alur, M. Babaali, Calin Belta, Volkan Isler, Ali Jadbabaie, John Koo, Vijay Kumar, Insup Lee, Stephen Prajna, Paulo Tabuada, Herbert Tanner.

Support
- NSF Career, PECASE
- NSF ITR (2)
- NSF EHS (3)
- ARO MURI (2)
- DARPA HURT
- Honeywell

Lecture goals

Why hybrid systems ?
- Emphasis on some engineering examples

Modeling of hybrid systems
- Emphasis on abstraction and refinement

Analysis of hybrid systems
- Emphasis on algorithmic verification

Approximations of discrete and continuous systems
- Emphasis on approximate (bi)-simulation

Warning : All questions and answers are biased and incomplete!

Outline of lectures

Lecture 1
- **Examples of hybrid systems and hybrid automata**
- A crash course in formal methods

Lecture 2
- Abstraction and refinement notions
- Discrete abstractions for hybrid systems verification

Lecture 3
- Approximation metrics for discrete/continuous systems
- Game theoretic interpretation of bisimulation

Why hybrid ?

Enabling technologies

Advances in sensor and actuator technology

GPS, control of quantum systems

Invasion of powerful microprocessors in physical devices

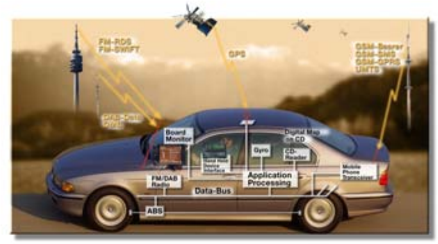
Sophisticated software/hardware on board

Networking everywhere

Interconnects subsystems

Penn

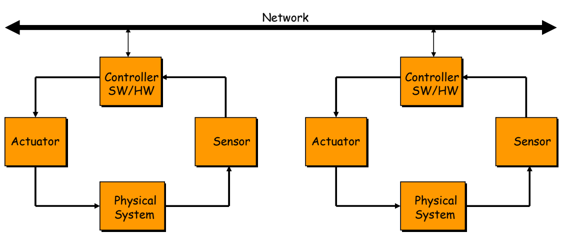
Emerging applications…



Latest BMW : 72 networked microprocessors
Boeing 777 : 1280 networked microprocessors

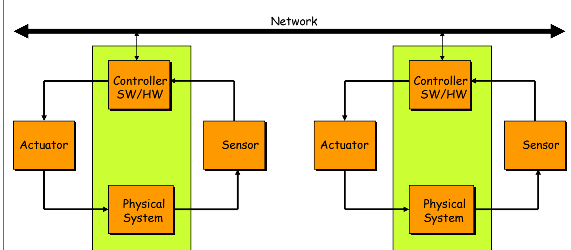
Penn

Networked embedded systems…



Penn

Networked embedded systems…



Physical system is continuous, software is discrete

Penn

Discrete and Continuous

Control Theory
Continuous systems
Stability, control
Feedback, robustness

Computer Science
Transition systems
Composition, abstraction
Concurrency models

Hybrid Systems
Software controlled systems
Multi-modal systems
Embedded real-time systems
Multi-agent systems

Penn

Exporting Science

Control Theory
Continuous systems
Stability, control
Feedback, robustness

Computer Science
Transition systems
Composition, abstraction
Concurrency models

Composition
Abstraction
Concurrency

Robustness
Feedback
Stability

Penn

Different views…

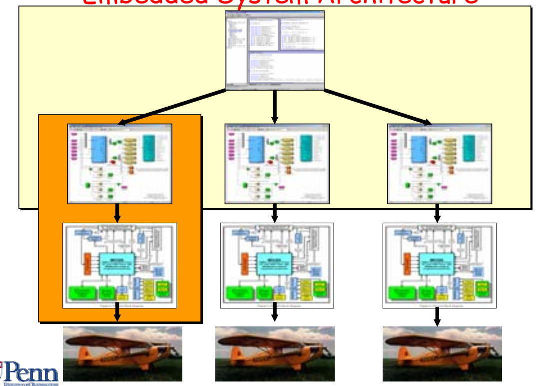
Computer science perspective

- View the physics from the eyes of the software
- Modeling result : Hybrid automaton

Control theory perspective

- View the software from the eyes of the physics
- Modeling result : Switched control systems

Embedded System Architecture



Hybrid behavior arises in

Hybrid dynamics

- Hybrid model is a simplification of a larger nonlinear model

Quantized control of continuous systems

- Input and observation sets are finite

Logic based switching

- Software is designed to supervise various dynamics/controllers

Partial synchronization of many continuous systems

- Resource allocation for competing multi-agent systems

Hybrid specifications of continuous systems

- Plant is continuous, but specification is discrete or hybrid…

Logic based switching

Nuclear reactor example

Without rods

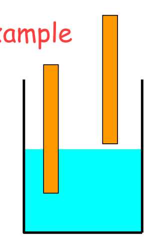
$$\dot{T} = 0.1T - 50$$

With rod 1

$$\dot{T} = 0.1T - 56$$

With rod 2

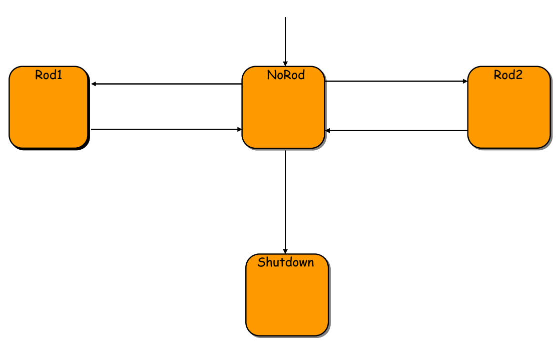
$$\dot{T} = 0.1T - 60$$



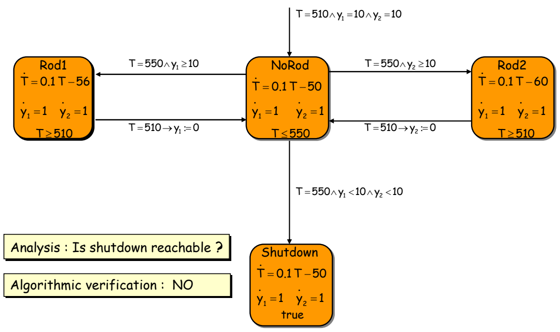
Rod 1 and 2 cannot be used simultaneously
Once a rod is removed, you cannot use it for 10 minutes

Specification : Keep temperature between 510 and 550 degrees.
If T=550 then either a rod is available or we shutdown the plant.

Software model of nuclear reactor



Hybrid model of nuclear reactor

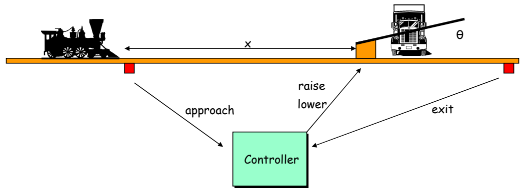


Analysis : Is shutdown reachable ?

Algorithmic verification : NO

Partial synchronization (Concurrency)

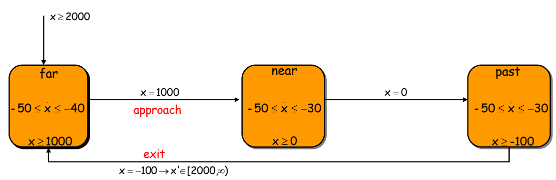
The train gate



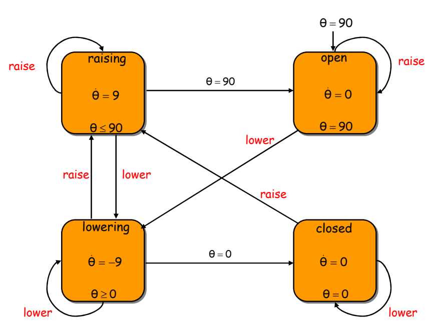
System = Train || Gate || Controller

Safety specification : If train is within 10 meters of the crossing, then gate should completely closed.
Liveness specification : Keep gate open as much as possible.

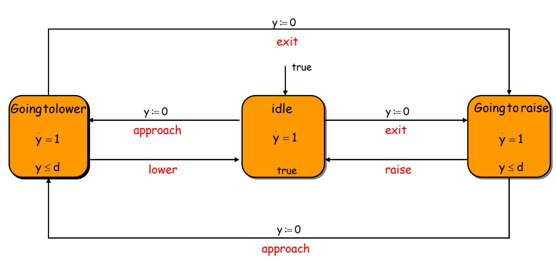
Train model



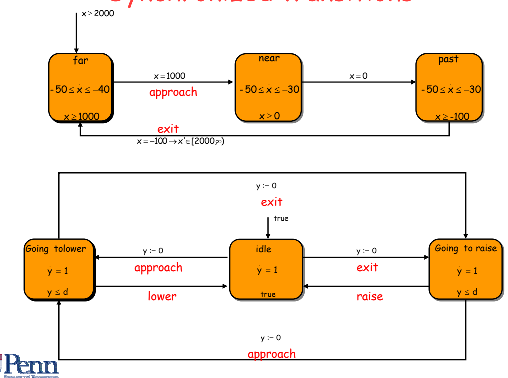
Gate model



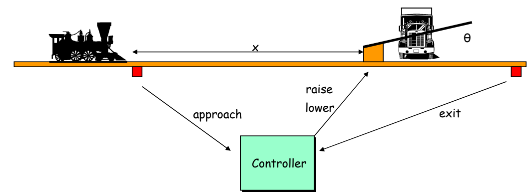
Controller model



Synchronized transitions



Verifying the controller



System = Train || Gate || Controller

Safety specification : Can we avoid the set $\theta > 0 \wedge (-10 \le x \le 10)$?

Parametric HyTech verification : YES if $d \le \frac{49}{5}$

Research Issues

Modeling Issues
- Well posedness, robustness, zenoness

Analysis
- Stability issues, qualitative theory, parametric analysis

Verification
- **Algorithmic methods that verify system performance**

Controller Synthesis
- Algorithmic methods that design hybrid controllers

Simulation
- Mixed signal simulation, event detection, modularity

Code generation
- From hybrid models to embedded code

Complexity
- **Compositionality and hierarchies**

Tools : HyTech, Checkmate, d/dt, HYSDEL, Stateflow, Charon

Outline of lectures

Lecture 1

Examples of hybrid systems and hybrid automata

A crash course in formal methods

Lecture 2

Abstraction and refinement notions

Discrete abstractions for hybrid systems verification

Lecture 3

Approximation metrics for discrete/continuous systems

Game theoretic interpretation of bisimulation

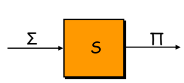
Transition Systems

A transition system

$$S = (Q, Q^0, \Sigma, \rightarrow, \Pi, \langle\cdot\rangle)$$

consists of

A set of states	Q
A subset of initial states	$Q^0 \subseteq Q$
A set of events	Σ
A set of observations	Π
The transition relation	$q \xrightarrow{\sigma} q'$
The observation map	$\langle q \rangle = \pi$



We assume systems to be non-blocking, possibly **nondeterministic**

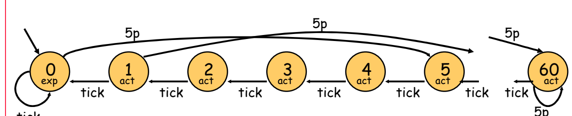
The sets Q, Σ, and Π may be infinite

Language $L(S)$ is all initialized sequences of observations

$\text{Reach}(S) = \{ \pi \in \Pi \mid \pi \text{ is reachable by } L(S) \}$

A discrete example

The parking meter



States Q ={0,1,2,...,60}

Events {tick,5p}

Observations {exp,act}

A possible string of observations $\text{exp} \xrightarrow{5p} \text{act} \xrightarrow{tick} \text{act} \xrightarrow{tick} \ldots$

A continuous example

Non-deterministic

$x' = F(x, d) \quad x(0) \in I$

$y = g(x) \quad d \in D$

$\longrightarrow$

$S = (Q, Q^0, \Sigma, \rightarrow, \Pi, \langle \cdot \rangle)$

State set $Q = X = R^n$

Label set $\Sigma = R_+$

Observation set $\Pi = Y = R^p$

Linear Observation Map $\langle x \rangle = g(x)$

Transition Relation $\rightarrow \subseteq X \times R_+ \times X$

$$x_1 \xrightarrow{t} x_2 \Leftrightarrow \begin{array}{l} \exists \; x(s), d(s) \text{ with } 0 \le s \le t \\ x(0) = x_1 \text{ and } x(t) = x_2 \text{ and} \\ x'(s) = F(x(s), d(s)) \end{array}$$

$$L(S) = \{ y_0 \xrightarrow{5} y_1 \xrightarrow{0.5} y_2 \xrightarrow{1} \ldots \}$$

Transition Systems

A region is a subset of states $P \subseteq Q$

We define the following operators

$$Pre_\sigma(P) = \{ q \in Q \mid \exists p \in P \quad q \xrightarrow{\sigma} p \}$$

$$Pre(P) = \{ q \in Q \mid \exists \sigma \in \Sigma \quad \exists p \in P \quad q \xrightarrow{\sigma} p \}$$

$$Post_\sigma(P) = \{ q \in Q \mid \exists p \in P \quad p \xrightarrow{\sigma} q \}$$

$$Post(P) = \{ q \in Q \mid \exists \sigma \in \Sigma \quad \exists p \in P \quad p \xrightarrow{\sigma} q \}$$

Transition Systems

We can recursively define

$$Pre_\sigma^1(P) = Pre_\sigma(P)$$

$$Pre_\sigma^n(P) = Pre_\sigma(Pre_\sigma^{n-1}(P))$$

Similarly for the other operators. Also

$$Pre^*(P) = \bigcup_{n \in N} Pre^n(P)$$

$$Post^*(P) = \bigcup_{n \in N} Post^n(P)$$

Safety and Invariance

Given transition system S, we consider two problems

Safety problem

Is $Reach(S) \cap \Pi_F$ empty?

Invariance problem

Is $Reach(S) \subseteq \Pi_F$?

Forward reachability algorithm

Forward Reachability Algorithm

```
initialize  R := P
while    TRUE      do
   if      R ∩ S ≠ ∅         return UNSAFE ; end if;
   if      Post(R) ⊆ R       return SAFE   ; end if;
   R := R ∪ Post(R)
end while
```

If S is finite, then algorithm terminates (decidability).

Complexity : $O(\underbrace{n_I}_{\text{initial states}} + \underbrace{m_R}_{\text{reachable transitions}})$

Backward reachability algorithm

Backward Reachability Algorithm

```
initialize  R := S
while    TRUE      do
   if      R ∩ P ≠ ∅         return UNSAFE ; end if;
   if      Pre(R) ⊆ R        return SAFE   ; end if;
   R := R ∪ Pre(R)
end while
```

If S is infinite, then there is no guarantee of termination.

Algorithmic issues

Representation issues
- Enumeration for finite sets
- Symbolic representation for infinite (or finite) sets

Operations on sets
- Boolean operations
- Pre and Post computations (closure?)

Algorithmic termination (decidability)
- Guaranteed for finite transition systems
- No guarantee for infinite transition systems

More complicated problems

More sophisticated properties can be expressed using
- Linear Temporal Logic (LTL)
- Computation Tree Logic (CTL)
- CTL*
- mu-calculus

Model checking

Given transition system S, and temporal logic formula φ

Basic verification problem

$$S \models \varphi$$

Two main approaches

- Model checking : Algorithmic, restrictive
- Deductive methods : Semi-automated, general

Linear temporal logic (informally)

Express temporal specifications along sequences

Informally	Syntax	Semantics
Eventually p	$\Diamond p$	$qqqqqqqqqqqp$
Always p	$\Box p$	$ppppppppppppp$
If p then next q	$p \Rightarrow \bigcirc q$	$qqqqqqqpq$
p until q	$p\, U\, q$	$pppppppppppppppq$

Linear temporal logic (formally)

Linear temporal logic syntax

The LTL formulas are defined inductively as follows

Atomic propositions
All observation symbols p are formulas

Boolean operators
If φ_1 and φ_2 are formulas then
$\varphi_1 \vee \varphi_2$ $\neg\varphi_1$

Temporal operators
If φ_1 and φ_2 are formulas then
$\varphi_1\, U\, \varphi_2$ $\bigcirc \varphi_1$

Linear temporal logic semantics

The LTL formulas are interpreted over infinite (omega) words

$$w = p_0\, p_1\, p_2\, p_3\, p_4 \ldots$$

$$(w,i) \models p \text{ iff } p_i = p$$
$$(w,i) \models \varphi_1 \vee \varphi_2 \text{ iff } (w,i) \models \varphi_1 \text{ or } (w,i) \models \varphi_2$$
$$(w,i) \models \neg\varphi_1 \text{ iff } (w,i) \not\models \varphi_1$$
$$(w,i) \models \bigcirc \varphi_1 \text{ iff } (w,i+1) \models \varphi_1$$
$$(w,i) \models \varphi_1\, U\, \varphi_2$$
$$\exists j \geq i\ (w,j) \models \varphi_2 \text{ and } \forall\, i \leq k < j\ (w,k) \models \varphi_1$$

$$w \models \phi \text{ iff } (w,0) \models \varphi$$
$$T \models \phi \text{ iff } \forall w \in L(T)\ \ w \models \varphi$$

Linear temporal logic

Syntactic boolean abbreviations

Conjunction	$\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$
Implication	$\varphi_1 \Rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$
Equivalence	$\varphi_1 \Leftrightarrow \varphi_2 = (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

Syntactic temporal abbreviations

Eventually	$\Diamond \varphi = \top \, U \, \varphi$
Always	$\Box \varphi = \neg \Diamond \neg \varphi$
In 3 steps	$\bigcirc_3 \varphi = \bigcirc\bigcirc\bigcirc \varphi$

LTL examples

Two processors want to access a critical section. Each processor can has three observable states

p1={inCS, outCS, reqCS}
p2={inCS, outCS, reqCS}

Mutual exclusion
Both processors are not in the critical section at the same time.

$$\Box \neg(p_1 = inCS \ \wedge \ p_2 = inCS)$$

Starvation freedom
If process 1 requests entry, then it eventually enters the critical section.

$$\Box \ p_1 = reqCS \Rightarrow \Diamond p_1 = inCS$$

LTL Model Checking

Given **finite** transition system and LTL formula we have

LTL model checking

Determine if $S \models \varphi$

System verified

Counterexample

Tools : SPIN (automata), SMV (BDD), SAT-based

Complexity : $O((n+m)(k+l)2^{O(k)})$

states transitions formula length

Outline of lectures

Lecture 1

- Examples of hybrid systems and hybrid automata
- A crash course in formal methods

Lecture 2

- **Abstraction and refinement notions**
- Discrete abstractions for hybrid systems verification

Lecture 3

- Approximation metrics for discrete/continuous systems
- Game theoretic interpretation of bisimulation

Dealing with model complexity

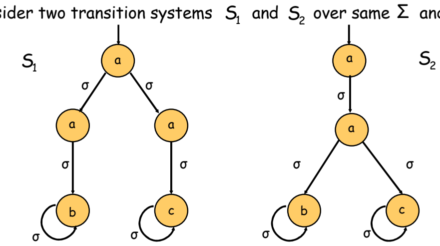
Bi-simulation

Simulation

Language Inclusion

Language Equivalence

Consider two transition systems S_1 and S_2 over same Σ and Π



Languanges are equivalent $L(S_1) = L(S_2) = \{ a \xrightarrow{\sigma} a \xrightarrow{\sigma} b \xrightarrow{\sigma} b \ldots,$

$a \xrightarrow{\sigma} a \xrightarrow{\sigma} c \xrightarrow{\sigma} c \ldots \}$

Safety equivalence

Language equivalence

If $L(S_1) = L(S_2)$ then $Reach(S_1) = Reach(S_2)$

Language inclusion

If $L(S_1) \subseteq L(S_2)$ then $Reach(S_1) \subseteq Reach(S_2)$

Language equivalence and inclusion are difficult to check

Penn

Simulation Relations

Consider two transition systems

$$S_1 = (Q_1, Q_1^0, \Sigma, \rightarrow_1, \Pi, \langle\cdot\rangle_1)$$
$$S_2 = (Q_2, Q_2^0, \Sigma, \rightarrow_2, \Pi, \langle\cdot\rangle_2)$$

$S_1 \leq S_2$

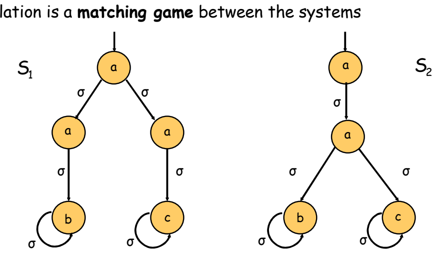
A relation $R \subseteq Q_1 \times Q_2$ is called a simulation relation if it

1. Respects initial states $\forall q_1 \in Q_1^0 \;\; \exists q_2 \in Q_2^0 \;\; (q_1, q_2) \in R$
2. Respects observations if $(q_1, q_2) \in R$ then $\langle q_1 \rangle_1 = \langle q_2 \rangle_2$
3. Respects transitions if $(q_1, q_2) \in R$ then $\begin{array}{ccc} q_1 & \xrightarrow{\sigma} & q_1' \\ R & & R \\ q_2 & \xrightarrow{\sigma} & q_2' \end{array}$

Penn

Simulation Games

Simulation is a **matching game** between the systems



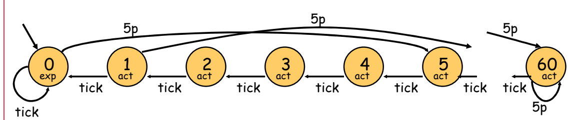
Note that $S_1 \leq S_2$ but it is not true that $S_2 \leq S_1$

The transition systems are **bisimilar** iff $S_1 \leq S_2$ and $S_2 \leq S_1$

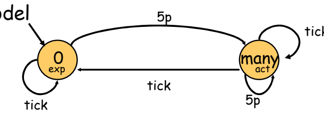
Penn

The parking example

The parking meter



A coarser model



$R = \{(0,0), (1, many), \ldots, (60, many)\}$

Penn

Simulation relations

Consider two transition systems S_1 and S_2

Simulation implies language inclusion

If $S_1 \leq S_2$ then $L(S_1) \subseteq L(S_2)$

Bi-simulation implies language equivalence

If $S_1 \cong S_2$ then $L(S_1) = L(S_2)$

Complexity of $L(S_1) \subseteq L(S_2)$ $\quad O((n_1 + m_1) 2^{n_2})$

Complexity of $S_1 \leq S_2$ $\quad O((n_1 + m_1)(n_2 + m_2))$

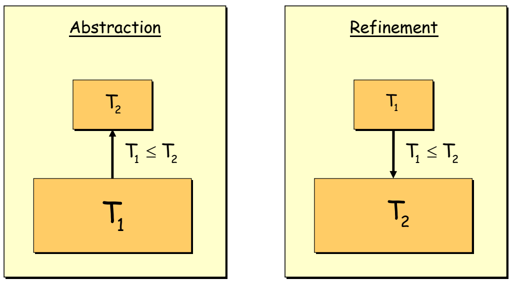
Penn

Exact Relationships

$$\begin{array}{ccc} S_1 \cong S_2 & \Rightarrow & S_1 \leq S_2 \\ \Downarrow & & \Downarrow \\ L(S_1) = L(S_2) & \Rightarrow & L(S_1) \subseteq L(S_2) \\ \Downarrow & & \Downarrow \\ Reach(S_1) = Reach(S_2) & \Rightarrow & Reach(S_1) \subseteq Reach(S_2) \end{array}$$

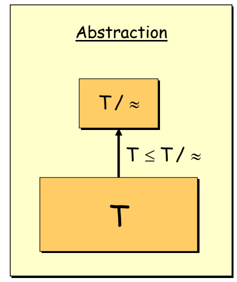
Penn

Two important cases



Penn

Special quotients



When is the quotient language equivalent or bisimilar to T ?

Penn

Quotient Transition Systems

Given a transition system

$$T = (Q, \Sigma, \rightarrow, O, \langle\cdot\rangle)$$

and an observation preserving partition $\approx \subseteq Q \times Q$, define

$$T/\approx = (Q/\approx, \Sigma, \rightarrow_{\approx}, O, \langle\cdot\rangle_{\approx})$$

naturally using

1. Observation Map

   $\langle P \rangle_{\approx} = o$ iff there exists $p \in P$ with $\langle p \rangle = o$

2. Transition Relation

   $P \xrightarrow{\sigma}_{\approx} P'$ iff there exists $p \in P, p' \in P'$ with $p \xrightarrow{\sigma} p'$

Penn

Outline of lectures

Lecture 1

Examples of hybrid systems and hybrid automata

A crash course in formal methods

Lecture 2

Abstraction and refinement notions

Discrete abstractions for hybrid systems verification

Lecture 3

Approximation metrics for discrete/continuous systems

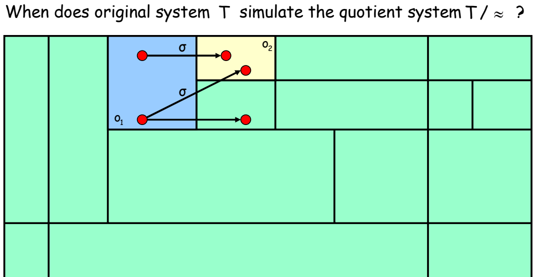
Game theoretic interpretation of bisimulation

Penn

Bisimulation Algorithm

Quotient system $T/\approx$ always simulates the original system T

When does original system T simulate the quotient system $T/\approx$?

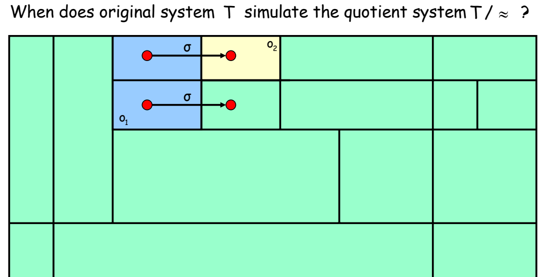


Penn

Bisimulation Algorithm

Quotient system $T/\approx$ always simulates the original system T

When does original system T simulate the quotient system $T/\approx$?



Penn

Bisimulation algorithm

Bisimulation Algorithm

`initialize` $Q/_{\sim} = \{p \sim q \text{ iff } <q> = <p>\}$
`while` $\exists P, P' \in Q/_{\sim}$ `such that` $\emptyset \neq \subseteq P \cap Pre(P') \neq \subseteq P$
$P_1 := P \cap Pre(P')$
$P_2 := P \setminus Pre(P')$
$Q/_{\sim} := (Q/_{\sim} \setminus \{P\}) \cup \{P_1, P_2\}$
`end while`

If T is finite, then algorithm computes coarsest quotient.
If T is infinite, there is no guarantee of termination

Relationships

Bisimulation
Strongest, more properties, easiest to check

Simulation
Weaker, less properties, easy to check

Language Inclusion
Weakest, less properties, difficult to check

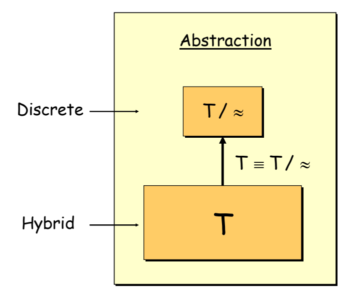
Complexity comparisons

Bisimulation
$O(m \cdot \log(n))$

Simulation
$O(m \cdot n)$

Language Equivalence
$O(m \cdot 2^n)$

Hybrid to discrete



Goal : Finite quotients of hybrid systems

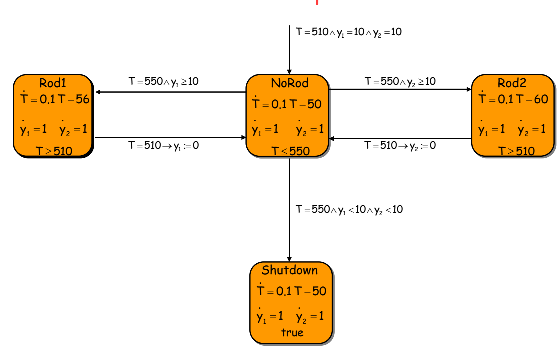
Hybrid System Model

A hybrid system $H = (V, \Re^n, X_0, F, Inv, R)$ consists of

- V is a finite set of states
- $\Re^n$ is the continuous state space
- $X = V \times \Re^n$ is the state space of the hybrid system
- $X_0 \subseteq X$ is the set of initial states
- $F(l,x) \subseteq \Re^n$ maps a diff. inclusion to each discrete state
- $Inv(l) \subseteq \Re^n$ maps invariant sets to each discrete state
- $R \subseteq X \times X$ is a relation capturing discontinuous changes

Define $E = \{(l,l') | \exists x \in Inv(l), x' \in Inv(l') \; ((l,x),(l',x')) \in R\}$
$Init(l) = \{x \in Inv(l) \mid (l,x) \in X_0\}$
$Guard(e) = \{x \in Inv(l) | \exists x' \in Inv(l') \; ((l,x),(l',x')) \in R\}$
$Reset(e,x) = \{x' \in Inv(l') | \; ((l,x),(l',x')) \in R\}$

An example



Transitions of Hybrid Systems

Hybrid systems can be embedded into transition systems

$$H = (V, \Re^n, X_0, F, Inv, R) \longrightarrow T_H = (Q, Q_0, \Sigma, \to, O, <\cdot>)$$

$Q = V \times \Re^n$

$Q_0 = X_0$

$\Sigma = E \cup \{\tau\}$

$\to \subseteq Q \times \Sigma \times Q$

Observation set and map depend on desired properties

Discrete transitions

$$(l_1, x_1) \xrightarrow{e} (l_2, x_2) \text{ iff } x_1 \in Guard(e), x_2 \in Reset(e, x_1)$$

Continuous (time-abstract) transitions

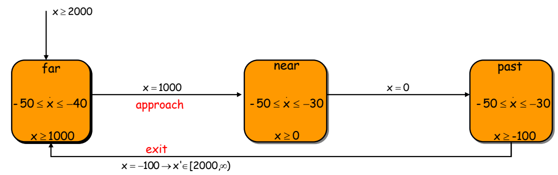
$$(l_1, x_1) \xrightarrow{\tau} (l_2, x_2) \text{ iff } l_1 = l_2 \text{ and } \exists \delta \geq 0 \quad x(\cdot) : [0, \delta] \to \Re^n$$

$$x(0) = x_1, x(\delta) = x_2, \text{ and } \forall t \in [0, \delta]$$

$$\dot{x} \in F(l_1, x(t)) \text{ and } x(t) \in Inv(l_1)$$

Rectangular hybrid automata

Rectangular sets : $\bigwedge_i x_i \sim c_i \quad \sim \in \{<, \leq, =, \geq, >\}, c_i \in Q$

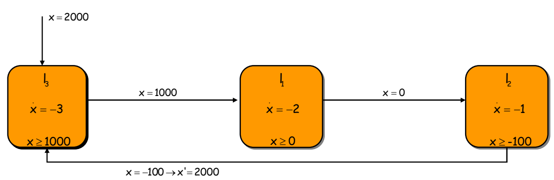


Rectangular hybrid automata are hybrid systems where

$$Init(l), Inv(l), F(l, x), Guard(e), Reset(e, x)_i$$

are rectangular sets

Multi-rate automata

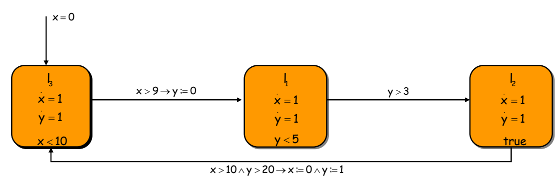


Multi-rate automata are rectangular hybrid automata where

$$Init(l), F(l, x), Reset(e, x)_i$$

are singleton sets

Timed automata



Timed automata are multi-rate automata where

$$F(l, x_i) = 1$$

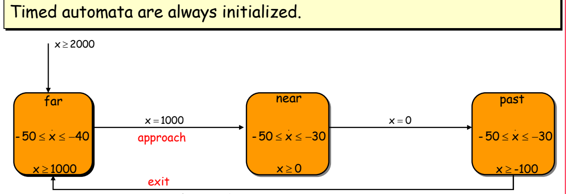
for all locations l and all variables.

Initialized automata

Rectangular hybrid automata are **initialized** if the following holds:

After a discrete transition, if the differential inclusion (equation) for a variable changes, then the variable must be reset to a fixed interval.

Timed automata are always initialized.



Bad news

Undecidability barriers

Consider the class of uninitialized multi-rate automata with n-1 clock variables, and one two slope variable (with two different rates).

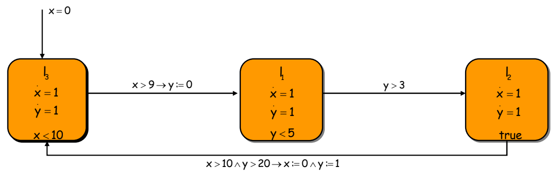
The reachability problem is undecidable for this class.

No algorithmic procedure exists.

Model checking temporal logic formulas is also undecidable

Initialization is necessary for decidability

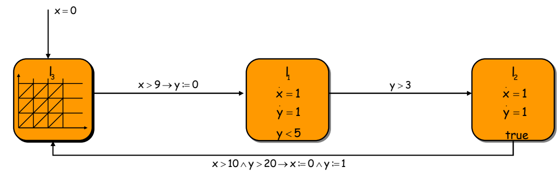
Timed automata



All timed automata admit a finite bisimulation

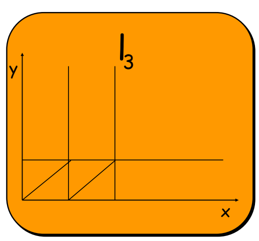
Hence CTL* model checking is decidable for timed automata

Timed automata



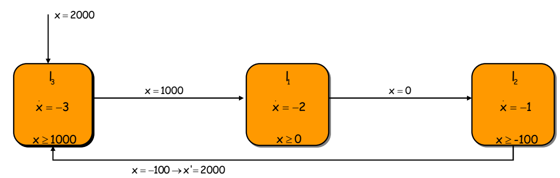
Approach : Discretize the clock dynamics using region equivalence

Region equivalence



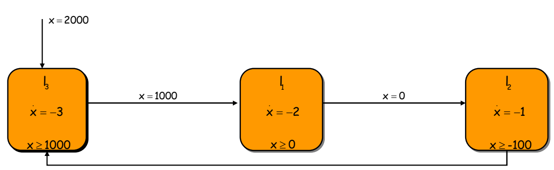
Equivalence classes : 6 corner points
14 open line segments
8 open regions

Multi-rate automata



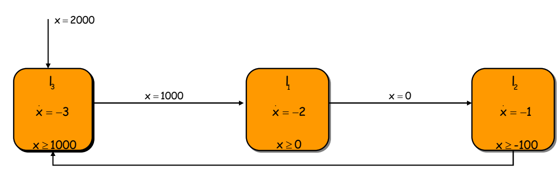
All initialized multi-rate automata admit a finite bisimulation

Rectangular automata



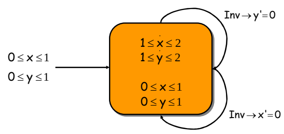
All initialized rectangular automata admit a finite bisimulation

Rectangular automata



~~All initialized rectangular automata admit a finite bisimulation~~

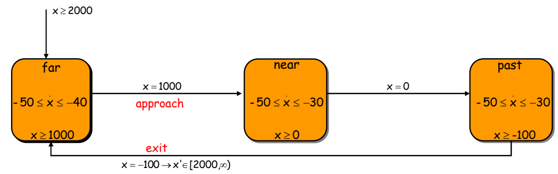
No finite bisimulation



Bisimulation algorithm never terminates

Penn

but…

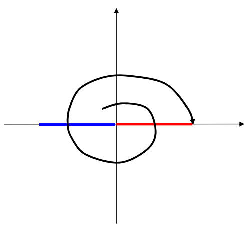


All initialized rectangular automata admit a finite language equivalence quotient which can be constructed effectively.

LTL model checking of rectangular automata is decidable.

Penn

More complicated dynamics?



Sets

$P_1 = \{(x,0) \mid 0 \le x \le 4\}$

$P_2 = \{(x,0) \mid -4 \le x < 0\}$

$P_3 = R^2 \setminus (P_1 \cup P_2)$

Dynamics

$\dot{x}_1 = 0.2x_1 + x_2$

$\dot{x}_2 = -x_1 + 0.2x_2$

Bisimulation algorithm never terminates !!

Penn

Basic problems

Finite bisimulations of continuous dynamical systems

Given a vector field F(x) and a finite partition of R^n

1. Does there exist a finite bisimulation ?
2. Can we compute it ?

Penn

Basic answers

Finite bisimulations of continuous dynamical systems

Consider a vector field X and a finite partition of R^n where

1. The flow of the vector field is definable in an o-minimal theory
2. The finite partition is definable in the same o-minimal theory

Then a finite bisimulation always exists.

Penn

Decidable problems for continuous systems

Consider linear vector fields of the form F(x)=Ax where

A is rational and nilpotent
A is rational, diagonalizable, with rational eigenvalues
A is rational, diagonalizable, with purely imaginary, rational eigenvalues

Then

1. The reachability problem between semi-algebraic sets is decidable.
2. Consider a finite semi-algebraic partition of the state space. Then a finite bisimulation always, exists and can be computed.
3. Consider a CTL* formula where atomic propositions denote semi-algebraic sets. Then CTL* model checking is decidable.

Penn

Decidable problems for hybrid systems

A hybrid system H is said to be o-minimal if

1. In each discrete state, all relevant sets and the flow of the vector field are definable in the same o-minimal theory.
2. After every discrete transition, state is reset to a constant set (forced initialization)

All o-minimal hybrid systems admit a finite bisimulation.

CTL* model checking is decidable for the class of o-minimal hybrid systems.

Penn

Decidable problems for hybrid systems

Consider a linear hybrid system H where

1. For each discrete state, all relevant sets are semi-algebraic
2. After every discrete transition, state is reset to a constant semi-algebraic set (forced initialization)
3. In each discrete location, the vector fields are of the form F(x)=Ax where
   - A is rational and nilpotent
   - A is rational, diagonalizable, with rational eigenvalues
   - A is rational, diagonalizable, with purely imaginary, rational eigenvalues

Then

CTL* model checking is decidable for this class of linear hybrid systems.

The reachability problem is decidable for such linear hybrid systems.

Penn

Safety verification of hybrid systems

Decidability boundary

- **Discrete abstraction of hybrid systems**, Alur, Henzinger, Lafferriere, Pappas
- **What's decidable about hybrid automata**, Henzinger, Kopke, Puri, Varaiya
- **Piecewise affine systems**, Sontag
- **Switched linear systems**, Blondel, Tsitsiklis

Symbolic rechability approaches

- **Linear hybrid automata**, Henzinger, Alur, Courcoubetis, Puri, Varaiya
- **Computer algebra**, Tiwari, Pappas, Manna, Mishra

Over-approximate rechability approaches

- **Level sets**, Tomlin, Mitchell, Bayen, Sastry
- **Flowpipes**, Krogh, Asarin, Maler, Pnueli
- **MILP**, Bemporad, Morari
- **Ellipsoids**, Kurzhanski, Varaiya
- **Zonotopes**, Girard
- **Predicate abstraction**, Alur, Clarke, Ivancic, Thang
- **Barrier certificates**, Prajna, Jadbabaie, Pappas, Roozbehani, Feron, Megretski

Tools : HyTech, Checkmate, d/dt, HYSDEL, Stateflow, Charon

Penn

Outline of lectures

Lecture 1

Examples of hybrid systems and hybrid automata

A crash course in formal methods

Lecture 2

Abstraction and refinement notions

Discrete abstractions for hybrid systems verification

Lecture 3

Approximation metrics for discrete/continuous systems

Game theoretic interpretation of bisimulation

Penn

Exact Relationships

$S_1 \cong S_2$ ⇒ $S_1 \leq S_2$

⇓ ⇓

$L(S_1) = L(S_2)$ ⇒ $L(S_1) \subseteq L(S_2)$

⇓ ⇓

$Reach(S_1) = Reach(S_2)$ ⇒ $Reach(S_1) \subseteq Reach(S_2)$

Penn

For deterministic systems

$S_1 \cong S_2$ ⇒ $S_1 \leq S_2$

⇕ ⇕

$L(S_1) = L(S_2)$ ⇒ $L(S_1) \subseteq L(S_2)$

⇓ ⇓

$Reach(S_1) = Reach(S_2)$ ⇒ $Reach(S_1) \subseteq Reach(S_2)$

Penn

Bi-simulations of control systems *

Input $u_1(t)$, output $y_1(t)$:

$$\dot{x}_1(t) = A_1 x_1(t) + B_1 u_1(t) + E_1 d_1(t)$$
$$y_1(t) = C_1 x_1(t) \qquad S_1$$

$$L(S_1) = \{(u_1(t), y_1(t)) \mid \exists x_1(t), d_1(t) \ \text{satisfying equations}\}$$

Input $u_2(t)$, output $y_2(t)$:

$$\dot{x}_2(t) = A_2 x_2(t) + B_2 u_2(t) + E_2 d_2(t)$$
$$y_2(t) = C_2 x_2(t) \qquad S_2$$

$$L(S_2) = \{(u_2(t), y_2(t)) \mid \exists x_2(t), d_2(t) \ \text{satisfying equations}\}$$

*G.J. Pappas, Bisimilar linear systems, Automatica, December 2003

*P. Tabuada and G.J. Pappas, Bisimilar control affine systems, Systems and Control Letters, May 2004.

*A. van der Schaft, Equivalence of dynamical system by bisimulation, IEEE TAC, December 2004

Non-deterministic dynamics

Output $y_1(t)$:

$$\dot{x}_1(t) = A_1 x_1(t) + E_1 d_1(t)$$
$$y_1(t) = C_1 x_1(t) \qquad S_1$$

Output $y_2(t)$:

$$\dot{x}_2(t) = A_2 x_2(t) + E_2 d_2(t)$$
$$y_2(t) = C_2 x_2(t) \qquad S_2$$

A relation R is a simulation relation if for all $\forall d_1(t) \ \exists d_2(t)$

$$x_1(0) \xrightarrow{d_1(t)} x_1(t)$$
$$R \qquad\qquad R \qquad\qquad C_1 x_1(t) = C_2 x_2(t)$$
$$x_2(0) \xrightarrow{d_2(t)} x_2(t)$$

R is a bi-simulation if converse is true as well

Exact bi-simulation

Nonlinear systems

G.J. Pappas and S.Simic, Consistent abstractions of affine control systems, IEEE TAC 2002.

P. Tabuada and G.J. Pappas, Abstractions of Hamiltonian systems, Automatica, 2003.

P. Tabuada and G.J. Pappas, Bisimilar control affine systems, Systems and control letters, 2003.

K. Grasse, Admissibility of trajectories in Φ-related systems, MCSS 2003

A. van der Schaft, Bisimulations of dynamical systems, Hybrid Systems : Computation and Control, 2004

Unifying discrete and continuous notions

E. Hagverdi, P.Tabuada, G.J. Pappas, Bisimulations of discrete, continuous, and hybrid systems, Theoretical Computer Science,2005

A.A.Julius, A.J. van der Schaft, A behavioral framework for compositionality, MTNS 2004

Extensions to hybrid systems

P. Tabuada, G.J. Pappas, P. Lima, Composing abstractions of hybrid systems, Discrete even dynamic systems, 2004

A. van der Schaft, Bisimulations of dynamical systems, Hybrid Systems : Computation and Control, 2004

G. Pola, A. van der Schaft, M. di Bennedeto, Equivalence of switching linear systems by bisimulation, IEEE CDC 2004

From exact to approximate

Exact relationships useful for binary answers

When dealing with the physical world, we use approximations
- Labeled Markov processes (Desharnais et. al., TCS 2004)
- Quantitative transition systems (de Alfaro et. al., ICALP 2004)
- Timed and hybrid systems

Approximate system relationships
- Enable larger system "compression"
- Quantify error/complexity tradeoffs
- Provide measures of robustness
- Potentially introduce different algorithms

Approximate Goal

Define **pseudo**-metrics on the set of transition systems:

$$d_L^{\rightarrow}(S_1, S_2) = 0 \quad \text{iff} \quad L(S_1) \subseteq L(S_2)$$
$$d_L(S_1, S_2) = 0 \quad \text{iff} \quad L(S_1) = L(S_2)$$
$$d_S^{\rightarrow}(S_1, S_2) = 0 \quad \text{iff} \quad S_1 \leq S_2$$
$$d_B(S_1, S_2) = 0 \quad \text{iff} \quad S_1 \cong S_2$$

Exact notions captured as zero sections of pseudo-metrics.

How can we define such metrics and how are they related ?

A. Girard and G.J. Pappas, Approximation metrics for discrete and continuous systems, 2005. Submitted.

Metrics

A **metric** d defined on a set E is a nonnegative function

$$d : E \times E \rightarrow R$$

Satisfying the usual properties

1. $d(e_1, e_2) = d(e_2, e_1)$
2. $d(e_1, e_2) = 0 \Leftrightarrow e_2 = e_1$
3. $d(e_1, e_3) \leq d(e_1, e_2) + d(e_2, e_3)$

Dropping property 1 results in a **directed** metric

Dropping $\Rightarrow$ in property 2 results in a **pseudo**-metric

Hausdorff distances

Given subsets A and B of E, the Hausdorff distance is

$$h^{\to}(A,B) = \sup_{a\in A} \inf_{b\in B} d(a,b)$$
$$h(A,B) = \max(h^{\to}(A,B), h^{\to}(B,A))$$

The classical result follows

$$h^{\to}(A,B) = 0 \Leftrightarrow cl(A) \subseteq cl(B)$$
$$h(A,B) = 0 \Leftrightarrow cl(A) = cl(B)$$

Metric Transition Systems

A transition system

$$S = (Q, Q^0, \Sigma, \to, \Pi, \langle\cdot\rangle)$$

is a called **metric** transition system if

The set of states is equipped with a metric $d_Q : Q \times Q \to R$
The set of events has the discrete metric
The set of observations is has a metric $d_\Pi : \Pi \times \Pi \to R$

Furthermore we assume that
1. Initial set is **compact**
2. Observation map is continuous
3. Post is continuous
4. Support(Post) is an open subset
5. Post(q) is **compact**

Reachability metrics

Since $Reach(S_1), Reach(S_2) \subseteq \Pi$ which is a metric space

$$d_R^{\to}(S_1,S_2) = h^{\to}(Reach(S_1), Reach(S_2))$$
$$d_R(S_1,S_2) = h(Reach(S_1), Reach(S_2))$$

The result follows

$$d_R^{\to}(S_1,S_2) = 0 \Leftrightarrow cl(Reach(S_1)) \subseteq cl(Reach(S_2))$$
$$d_R(S_1,S_2) = 0 \Leftrightarrow cl(Reach(S_1)) = cl(Reach(S_2))$$

Language metrics

Lifting the metric to sequences (in the infinity sense)

$$d_L^{\to}(S_1,S_2) = \sup_{r_1\in L(S_1)} \inf_{r_2\in L(S_2)} d_\Pi(r_1, r_2)$$
$$d_L(S_1,S_2) = \max\{ d_L^{\to}(S_1,S_2), d_L^{\to}(S_2,S_1) \}$$

The result follows

$$d_L^{\to}(S_1,S_2) = 0 \Leftrightarrow cl(L(S_1)) \subseteq cl(L(S_2))$$
$$d_L(S_1,S_2) = 0 \Leftrightarrow cl(L(S_1)) = cl(L(S_2))$$

Inequalities

$$d_R^{\to}(S_1,S_2) \le d_L^{\to}(S_1,S_2)$$
$$d_R(S_1,S_2) \le d_L(S_1,S_2)$$

$$Reach(S_1) \subseteq N(Reach(S_2), d_R^{\to}(S_1,S_2))$$
$$\subseteq N(Reach(S_2), d_L^{\to}(S_1,S_2))$$

Approximate Simulation Relations

Consider two transition systems and let $\delta \ge 0$ be given

$$S_1 = (Q_1, Q_1^0, \Sigma, \to_1, \Pi, \langle\cdot\rangle_1)$$
$$S_2 = (Q_2, Q_2^0, \Sigma, \to_2, \Pi, \langle\cdot\rangle_2)$$

Relation $R \subseteq Q_1 \times Q_2$ is a δ- simulation relation if it

1. Respects initial states $\forall q_1 \in Q_1^0 \quad \exists q_2 \in Q_2^0 \quad (q_1,q_2) \in R$
2. Respects observations if $(q_1,q_2) \in R$ then $d_\Pi(\langle q_1\rangle_1, \langle q_2\rangle_2) \le \delta$
3. Respects transitions if $(q_1,q_2) \in R$ then $\begin{array}{ccc} q_1 & \xrightarrow{\sigma} & q_1' \\ R & & R \\ q_2 & \xrightarrow{\sigma} & q_2' \end{array}$

Approximate simulation

For $\delta = 0$ we recover exact simulation relation.

S_2 approximately simulates S_1 (with precision $\delta \geq 0$),

$$S_1 \leq_\delta S_2$$

if there exists δ -simulation relation R.

For all $\delta, \delta' \geq 0$ we have

$$S_1 \leq_\delta S_1$$

$$S_1 \leq_\delta S_2 \text{ and } \delta' \geq \delta \text{ then } S_1 \leq_{\delta'} S_2$$

$$S_1 \leq_\delta S_2 \text{ and } S_2 \leq_{\delta'} S_3 \text{ then } S_1 \leq_{\delta+\delta'} S_3$$

Simulation metrics

The **simulation** metric is defined as the tightest precision with which S_2 simulates S_1

$$d_S^{\to}(S_1, S_2) = \inf_{\delta \geq 0} \{ S_1 \leq_\delta S_2 \}$$

For any transition system we have

$$\text{if } S_1 \leq S_2 \text{ then } d_S^{\to}(S_1, S_2) = 0$$

For **metric** transition systems we have

$$\text{if } d_S^{\to}(S_1, S_2) = 0 \text{ then } S_1 \leq S_2$$

Bi-simulation metrics

The **bi-simulation** metric is defined as the tightest precision with which S_2 bi-simulates S_1

$$d_B(S_1, S_2) = \inf_{\delta \geq 0} \{ S_1 \cong_\delta S_2 \}$$

For any transition system we have

$$\text{if } S_1 \cong S_2 \text{ then } d_B(S_1, S_2) = 0$$

For **metric** transition systems we have

$$\text{if } d_B(S_1, S_2) = 0 \text{ then } S_1 \cong S_2$$

Approximate relationships

$$\begin{tikzcd}
d_B(S_1,S_2) \arrow[r] \arrow[d] & d_S^{\to}(S_1,S_2) \arrow[d] \\
d_L(S_1,S_2) \arrow[r] \arrow[d] & d_L^{\to}(S_1,S_2) \arrow[d] \\
d_R(S_1,S_2) \arrow[r] & d_R^{\to}(S_1,S_2)
\end{tikzcd}$$

If metrics are zero then

$$\begin{tikzcd}
S_1 \cong S_2 \arrow[r] \arrow[d] & S_1 \leq S_2 \arrow[d] \\
cl(L(S_1)) = cl(L(S_2)) \arrow[r] \arrow[d] & cl(L(S_1)) \subseteq cl(L(S_2)) \arrow[d] \\
cl(Reach(S_1)) = cl(Reach(S_2)) \arrow[r] & cl(Reach(S_1)) \subseteq cl(Reach(S_2))
\end{tikzcd}$$

Simulation algorithm

Maximal (coarsest) simulation relation can be computed using the following algorithm

Given $\delta \geq 0$

$$R^0 = \{ (q_1, q_2) \mid d_\Pi(\langle q_1 \rangle, \langle q_2 \rangle) \leq \delta \}$$

$$R^{i+1} = \{ (q_1, q_2) \in R^i \mid \forall\, q_1 \xrightarrow{\sigma}_1 q_1' \ \exists\, q_2 \xrightarrow{\sigma}_2 q_2' \ (q_1', q_2') \in R^i \}$$

We obtain that

$$\bigcap_{i=0}^{i=+\infty} R^i = R^*$$

$$R^* \subseteq R^i$$

For $\delta = 0$, we obtain the usual simulation algorithm

Relations versus functions

Express relations as levels sets of functions

For any given $\delta \geq 0$

$$R^i = \{ (q_1,q_2) \mid f^i(q_1,q_2) \leq \delta \}$$

$$R^* = \{ (q_1,q_2) \mid f^*(q_1,q_2) \leq \delta \}$$

Simulation functions are obtained by a dual algorithm

$$f^0 = d_\Pi(\langle q_1\rangle,\langle q_2\rangle)$$

$$f^{i+1} = \max\{d_\Pi(\langle q_1\rangle,\langle q_2\rangle), \sup_{q_1 \to_1 q_1'} \inf_{q_2 \to_2 q_2'} f^i(q_1',q_2')\}$$

Simulation metric

The limit $f^* = \lim_{i\to+\infty} f^i$ exists and is the **minimal** solution of

$$f^* = \max\{d_\Pi(\langle q_1\rangle,\langle q_2\rangle), \sup_{q_1 \to_1 q_1'} \inf_{q_2 \to_2 q_2'} f^*(q_1',q_2')\}$$

Simulation functions define the simulation metric

$$d_S^{\to}(S_1,S_2) = \sup_{q_1 \in Q_1^0} \inf_{q_2 \in Q_2^0} f^*(q_1,q_2)$$

A similar story for bi-simulation metrics

Bi-simulation algorithm

Maximal (coarsest) bi-simulation relation can be computed using the following algorithm

Given $\delta \geq 0$

$$R^0 = \{ (q_1,q_2) \mid d_\Pi(\langle q_1\rangle,\langle q_2\rangle) \leq \delta \}$$

$$R^{i+1} = \{ (q_1,q_2) \in R^i \mid \forall q_1 \xrightarrow{\sigma}_1 q_1' \ \exists q_2 \xrightarrow{\sigma}_2 q_2' \ (q_1',q_2') \in R^i \}$$

$$\text{and } \forall q_2 \xrightarrow{\sigma}_2 q_2' \ \exists q_1 \xrightarrow{\sigma}_1 q_1' \ (q_1',q_2') \in R^i \}$$

We obtain that

$$\bigcap_{i=0}^{i=+\infty} R^i = R^*$$

$$R^* \subseteq R^i$$

For $\delta = 0$, we obtain the usual bi-simulation algorithm

Bi-simulation functions and metric

Bi-simulation functions are obtained by a dual algorithm

$$f^0 = d_\Pi(\langle q_1\rangle,\langle q_2\rangle)$$

$$f^{i+1} = \max\{d_\Pi(\langle q_1\rangle,\langle q_2\rangle), \sup_{q_1 \to_1 q_1'} \inf_{q_2 \to_2 q_2'} f^i(q_1',q_2'), \sup_{q_2 \to_2 q_2'} \inf_{q_1 \to_1 q_1'} f^i(q_1',q_2')\}$$

Using the limit $f^* = \lim_{i\to+\infty} f^i$ of the algorithm we can define

$$d_B(S_1,S_2) = \max\{ \sup_{q_1 \in Q_1^0} \inf_{q_2 \in Q_2^0} f^*(q_1,q_2), \sup_{q_2 \in Q_2^0} \inf_{q_1 \in Q_1^0} f^*(q_1,q_2) \}$$

Exact Computation

	Discrete*	Continuous
$d_L(S_1,S_2)$	PSPACE-complete	Impossible
$d_B(S_1,S_2)$	$O(n^4)$	One **dynamic** game One static game

*L. De Alfaro, M. Faella, and M Stoelinga, Metrics for quantitative transition systems, ICALP 2004

Bounding metrics

Relax the equality with inequality, and search for

$$f \geq \max\{d_\Pi(\langle q_1\rangle,\langle q_2\rangle), \sup_{q_1 \to_1 q_1'} \inf_{q_2 \to_2 q_2'} f(q_1',q_2')\}$$

Then $f \geq f^*$ and therefore

$$d_S^{\to}(S_1,S_2) \leq \sup_{q_1 \in Q_1^0} \inf_{q_2 \in Q_2^0} f(q_1,q_2)$$

A similar story for bi-simulation metrics

Lyapunov-like conditions

Inequalities can be expressed in Lyapunov-like form as

$$f(q_1,q_2) \geq d_\Pi(\langle q_1\rangle, \langle q_2\rangle)$$
$$f(q_1,q_2) \geq \sup_{q_1 \to_1 q_1'} \inf_{q_2 \to_2 q_2'} f(q_1', q_2')$$

Similarly, for bi-simulation

$$f(q_1,q_2) \geq d_\Pi(\langle q_1\rangle, \langle q_2\rangle)$$
$$f(q_1,q_2) \geq \sup_{q_1 \to_1 q_1'} \inf_{q_2 \to_2 q_2'} f(q_1', q_2')$$
$$f(q_1,q_2) \geq \sup_{q_2 \to_2 q_2'} \inf_{q_1 \to_1 q_1'} f(q_1', q_2')$$

Penn

Constrained linear systems

S_1: $\dot{x}_1(t) = A_1 x_1(t) + E_1 d_1(t),\ d_1(t) \in D_1$, $y_1(t) = C_1 x_1(t)$ → $y_1(t)$

S_2: $\dot{x}_2(t) = A_2 x_2(t) + E_2 d_2(t),\ d_2(t) \in D_2$, $y_2(t) = C_2 x_2(t)$ → $y_2(t)$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}, C = [C_1 \mid -C_2]$$
$$d = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}, E = \begin{bmatrix} E_1 & 0 \\ 0 & E_2 \end{bmatrix}$$

Restricting to quadratic functions $f(x_1,x_2) = \sqrt{x^T M x}$ we get

$$f^2(x_1,x_2) \geq x^T C^T C x$$
$$\sup_{d_1 \in D_1} \inf_{d_2 \in D_2} \nabla f\,(Ax + Ed) \leq 0$$
$$\sup_{d_2 \in D_2} \inf_{d_1 \in D_1} \nabla f\,(Ax + Ed) \leq 0$$

Penn

Constrained linear systems

S_1:
$\dot{x}_1 = -2x_1 + y_1 + z_1 + d_1$
$\dot{y}_1 = -x_1 + z_1 + d_1$
$\dot{z}_1 = x_1 - y_1 - 2z_1$
$y_1 = x_1(t)$, $d_1 \in [-1,1]$
→ $y_1(t)$

$I_1 = \{-1 \leq x_1(0) - y_1(0) - z_1(0) \leq 1$, $8 \leq y_1(0) \leq 9$, $-6 \leq z_1(0) \leq -4\}$

S_2:
$\dot{x}_2 = -x_2 + d_2$
$y_2 = x_2(t)$, $d_2 \in [-1,1]$
→ $y_2(t)$

$I_2 = \{2 \leq x_2(0) \leq 5\}$

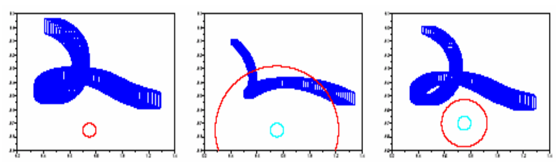
Function $f(x_1,y_1,z_1,x_2) = |x_1 - y_1 - z_1| + |y_1 + z_1 - x_2|$ satisfies conditions

$$d_S^{\rightarrow}(S_2,S_1) = 0$$
$$d_S^{\rightarrow}(S_1,S_2) \leq \sup_{I_1} \inf_{I_2} f = 1$$
$$d_B(S_2,S_1) \leq 1 \Rightarrow d_R(S_2,S_1) \leq 1 \Rightarrow \text{Reach}(S_1) \subseteq N(\text{Reach}(S_2),1)$$

Penn

Deterministic linear systems

Reduces to solving Lyapunov equations

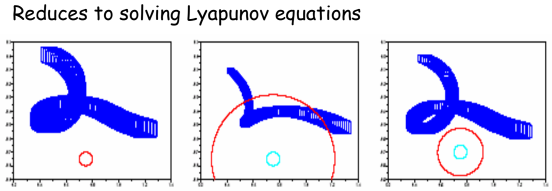


Reachable sets of the
1. 100 dimensional linear system,
2. 6 dimensional approximation,
3. 10 dimensional approximation.

Penn

Deterministic linear systems

Reduces to solving Lyapunov equations



The more robustly safe the system,
the more we can compress the model
the easier safety verification becomes

Penn

Constrained nonlinear systems

S_1: $\dot{x}_1 = f_1(x_1,d_1)$, $d_1(t) \in D_1$, $y_1 = g_1(x_1)$ → $y_1(t)$

S_2: $\dot{x}_2 = f_2(x_2,d_2)$, $d_2(t) \in D_2$, $y_2 = g_2(x_2)$ → $y_2(t)$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad d = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$
$$F(x,d) = \begin{bmatrix} f_1(x_1,d_1) \\ f_2(x_2,d_2) \end{bmatrix}$$
$$g(x) = g_1(x_1) - g_2(x_2)$$

We are looking for functions $f(x)$ satisfying

$$f^2(x) \geq \| g(x) \|_2^2$$
$$\sup_{d_1 \in D_1} \inf_{d_2 \in D_2} \nabla f \cdot F(x,d) \leq 0$$
$$\sup_{d_2 \in D_2} \inf_{d_1 \in D_1} \nabla f \cdot F(x,d) \leq 0$$

Penn

Nonlinear systems

S_1

$$\dot{x}_1 = -(1 + 0.1x_2^2)x_1$$
$$\dot{x}_2 = -\frac{1-0.1x_1^2}{2}x_2 + 2x_3$$
$$\dot{x}_3 = -2(1 - 0.1x_1)x_2 - \frac{1}{2}y_3$$

$y_1 = 0.1x_1 + x_2$
$y_2 = x_3$

$I_1 = [-2,2] \times \{0\} \times [4,6]$

S_2

$$\dot{z}_2 = -\frac{1}{2}z_2 + 2z_3$$
$$\dot{z}_3 = -2z_2 - \frac{1}{2}z_3$$

$y_1 = z_2$
$y_2 = z_3$

$I_2 = \{0\} \times [4,6]$

Using S.O.S., we obtained

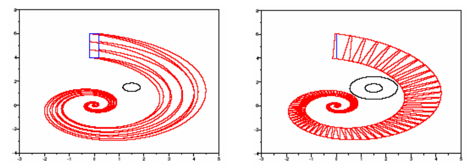
$$f = \sqrt{1.205(x_2 - z_2)^2 + 1.202(x_3 - z_3)^2 + 0.059x_1^2 + 0.007x_1^4}$$

$$d_B(S_1, S_2) \leq 0.590$$

Penn

Nonlinear systems

3D nonlinear system with 2D output.



Reachable sets of the three dimensional nonlinear system, and of a two dimensional linear approximation.

Penn

Main ideas

Metrics for discrete and continuous systems

Approximate language inclusion, bi-simulation

Fixed-point (game-theoretic) characterization

Lyapunov-like relaxations

Penn

Next steps

Metrics for hybrid systems

$$d(H_1, H_2)$$

Compositional approximations

$$S_1 \cong_\delta S_2 \overset{?}{\Rightarrow} S \,||\, S_1 \cong_\delta S \,||\, S_2$$

Robust, logical equivalence

$$S_1 \cong_\delta S_2 \overset{?}{\Rightarrow} S_1 \models \varphi \iff S_2 \models \varphi$$

Penn

Thanks again !

School Organizers

Alberto Bemporad

Maurice Heemels

and HYCON

Penn