Verification of hybrid systems

George J. Pappas

Departments of ESE and CIS

University of Pennsylvania

pappasg@ee.upenn.edu

http://www.seas.upenn.edu/~pappasg

HYBRIDGE Summer School on

Hybrid Systems : A Formal Paradigm for

Safety Critical Embedded Systems

Patras, Greece

September 22-24, 2004

Thanks to

<u>School Organizers</u>

Marika di Benedetto

Kostas Kyriakopoulos

John Lygeros

and HYBRIDGE

Acknowledgments

Postdocs

- Mohamed Babaali
- Antoine Girard

Ph.D Students

- Ali Ahmazadeh
- George Fainekos
- Hadas Kress Gazit
- Hakan Yazarel
- Michael Zavlanos

Collaborators

Rajeev Alur, Datta Godbole, Tom Henzinger, Ali Jadbabaie, John Koo, Vijay Kumar, Gerardo Lafferierre, Insup Lee, John Lygeros, Shankar Sastry, Omid Shakernia, Claire Tomlin, Sergio Yovine

Support

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

Penn

Goals for this mini-course

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

Synthesis of hybrid controllers

- Emphasis on temporal logic synthesis

Warning : All questions and answers are biased and incomplete!

Penn

Some references

Bisimilar linear systems
George J. Pappas
Automatica. 39(12):2035-2047 December 2003.

Model checking LTL over controllable linear systems is decidable
Paulo Tabuada and George J. Pappas
Hybrid Systems : Computation and Control, Lecture Notes in Computer Science, Prague, Czech Republic, April 2003

Symbolic reachability computations for families of linear vector fields
G. Lafferriere, G. J. Pappas, and S. Yovine
Journal of Symbolic Computation, 32(3):231-253, September 2001.

Discrete abstractions of hybrid systems
R.Alur, T. Henzinger, G. Lafferriere, G. Pappas
Proceedings of the IEEE, 88(2):971-984, July 2000.

Hierarchically consistent control systems
George J. Pappas, Gerardo Lafferriere, and Shankar Sastry
IEEE Transactions on Automatic Control, 45(6):1144-1160, June 2000.

O-minimal hybrid systems
G. Lafferriere, G. J. Pappas, and S. Sastry
Mathematics of Control, Signals, and Systems, 13(1):1-21, March 2000.

Penn
University of Pennsylvania

Outline of lectures

Lecture 1 : Thursday, September 23

Examples of hybrid systems and modeling formalisms

Transitions systems, temporal logics, abstraction

Discrete abstractions of hybrid systems for verification

Lecture 2 : Friday, September 24

Applications in motion planning and visibility games

Penn
University of Pennsylvania

Why hybrid ?

Enabling technologies

Advances in sensor and actuator technology

GPS, control of quantum systems

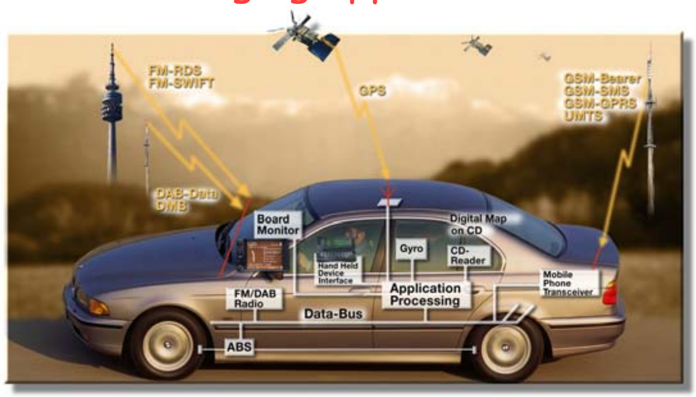
Invasion of powerful microprocessors in physical devices

Sophisticated software/hardware on board

Networking everywhere

Interconnects subsystems

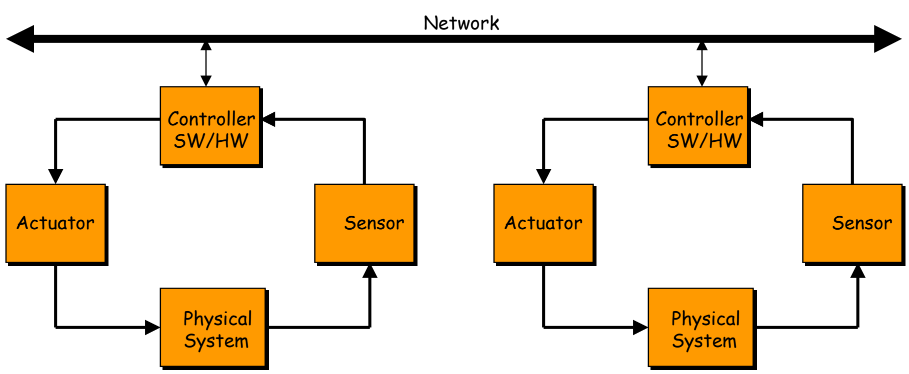
Emerging applications…



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

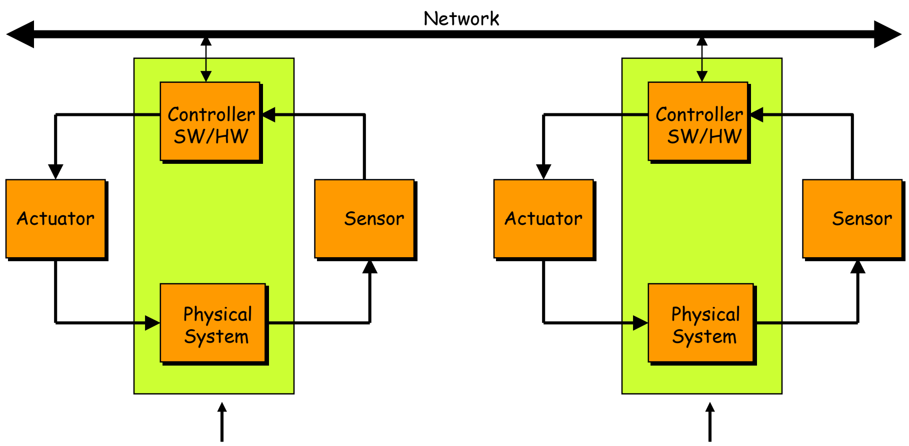
Penn

Networked embedded systems…



Penn

Networked embedded systems…

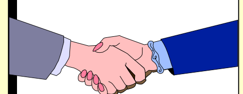


Physical system is continuous, software is discrete

Discrete and Continuous

Control Theory

- Continuous systems
- Stability, control
- Feedback, robustness



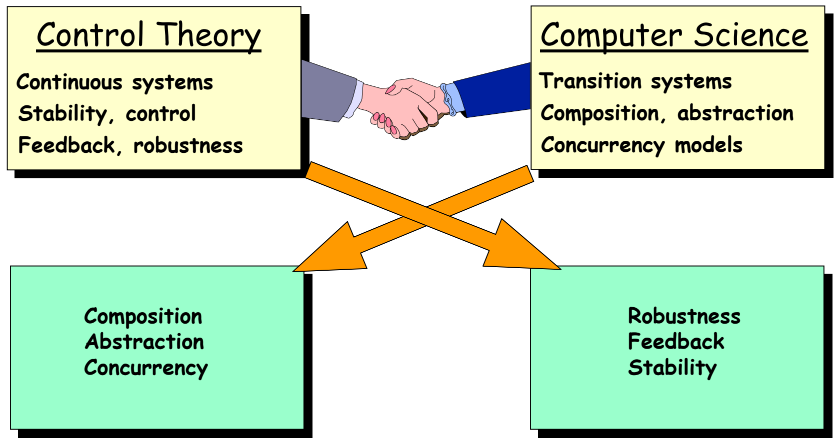
Computer Science

- Transition systems
- Composition, abstraction
- Concurrency models

Hybrid Systems

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

Exporting Science



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

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

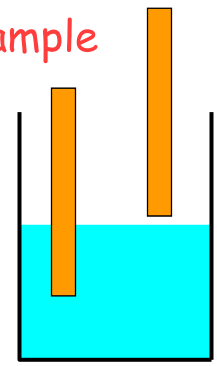
- Plant is continuous, but specification is discrete or hybrid...

Penn University of Pennsylvania

Logic based switching

Penn University of Pennsylvania

Nuclear reactor example



Without rods

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

With rod 1

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

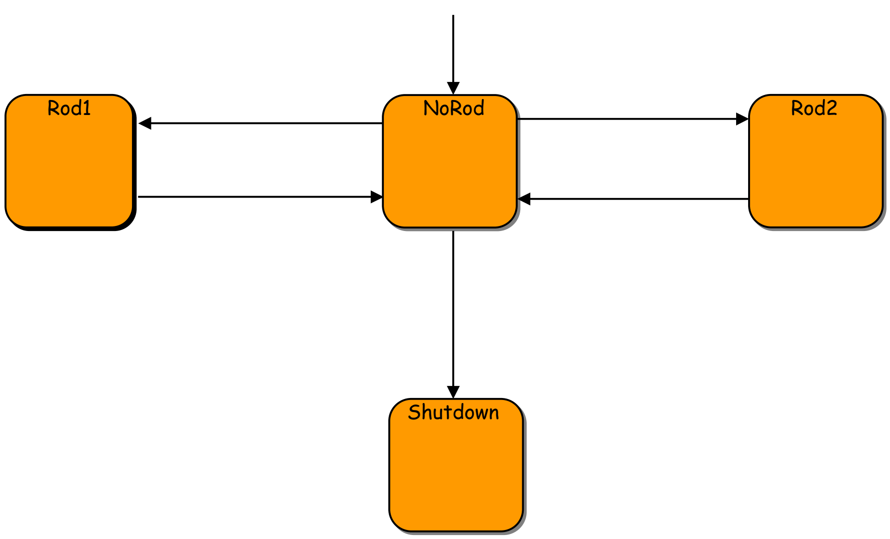
With rod 2

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

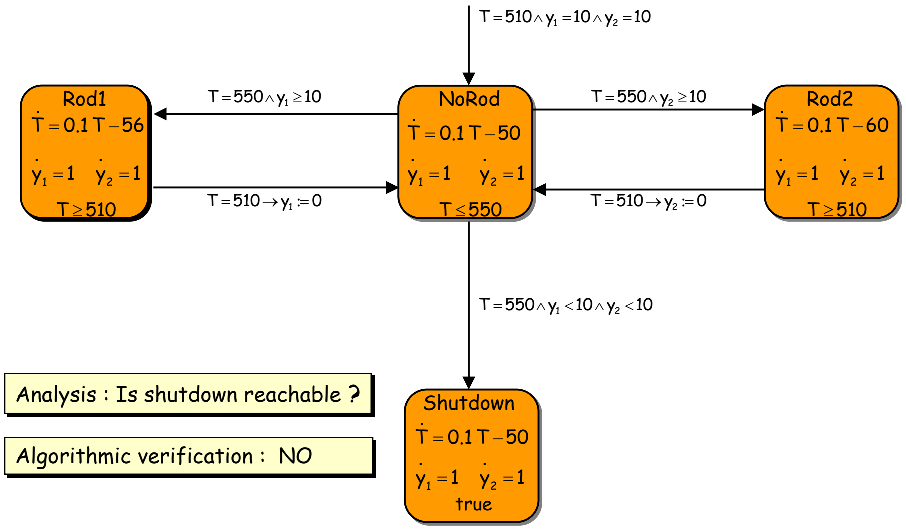
Rod 1 and 2 cannot be used simultaneously
Once a rod is removed, you cannot use it for 10 minutes

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



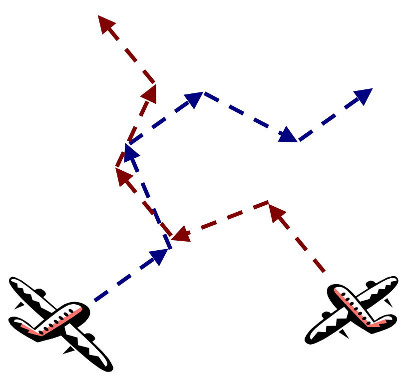
Hybrid model of nuclear reactor



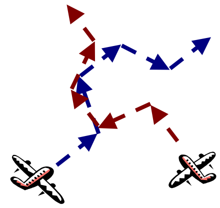
Analysis : Is shutdown reachable ?

Algorithmic verification : NO

Conflict Resolution in ATM*



Conflict Resolution Protocol

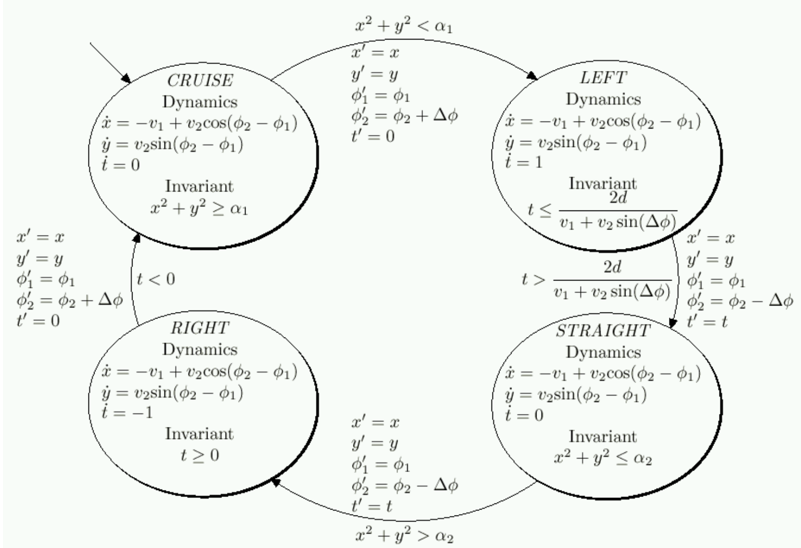


1. Cruise until a_1 miles away
2. Change heading by $\Delta\Phi$
3. Maintain heading until lateral distance d
4. Change to original heading
5. Change heading by - $\Delta\Phi$
6. Maintain heading until lateral distance - d
7. Change to original heading

Is this protocol safe ?

Penn

Conflict Resolution Maneuver



Penn

Computing Unsafe Sets

$$v_1 = 4; v_2 = 5; \lambda = 0$$

$$\texttt{unsafeCruise} = \texttt{Resolve}\left[\exists t > 0 \wedge (x - v_1 t + \lambda v_2 t)^2 + (y + \sqrt{1-\lambda^2}\, v_2 t)^2 \le 25\right]$$

$$= \left(y < -\tfrac{20}{\sqrt{41}} \wedge -\sqrt{41} - \tfrac{4y}{5} \le x \le \sqrt{41} - \tfrac{4y}{5}\right) \vee \left(y = -\tfrac{20}{\sqrt{41}} \wedge -\sqrt{41} - \tfrac{4y}{5} < x \le \sqrt{41} - \tfrac{4y}{5}\right) \vee$$
$$\left(y = \tfrac{20}{\sqrt{41}} \wedge -\sqrt{25-y^2} < x < \sqrt{41} - \tfrac{4y}{5}\right) \vee \left(\tfrac{20}{\sqrt{41}} \le y < 5 \wedge -\sqrt{25-y^2} < x < \sqrt{25-y^2}\right) \vee$$
$$\left(-\tfrac{20}{\sqrt{41}} < y < \tfrac{20}{\sqrt{41}} \wedge -\sqrt{25-y^2} < x \le \sqrt{41} - \tfrac{4y}{5}\right)$$

$$v_1 = 4; v_2 = 5; \lambda = \tfrac{3}{5}$$

$$\texttt{unsafeLeft} = \texttt{Resolve}\left[\exists t > 0 \wedge (x - v_1 t + \lambda v_2 t)^2 + (y + \sqrt{1-\lambda^2}\, v_2 t)^2 \le 25\right]$$

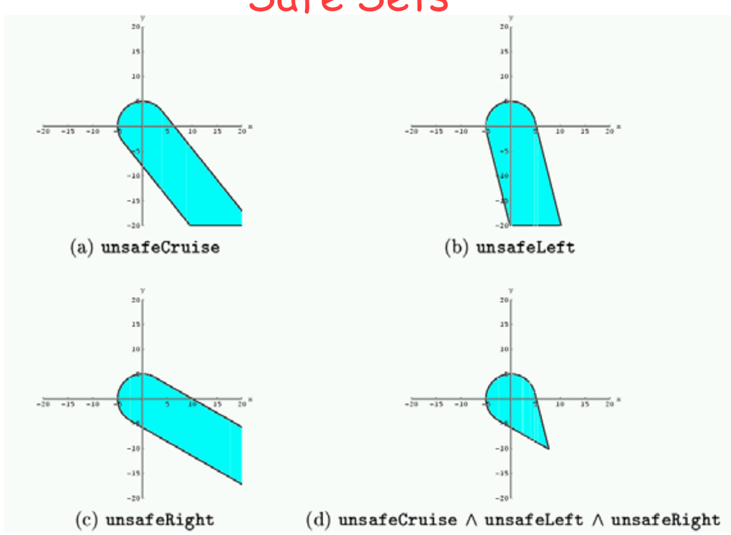
$$= \left(y < -\tfrac{5}{\sqrt{17}} \wedge -\tfrac{5\sqrt{17}}{4} - \tfrac{y}{4} \le x \le \tfrac{5\sqrt{17}}{4} - \tfrac{y}{4}\right) \vee \left(y = -\tfrac{5}{\sqrt{17}} \wedge -\tfrac{5\sqrt{17}}{4} - \tfrac{y}{4} < x \le \tfrac{5\sqrt{17}}{4} - \tfrac{y}{4}\right) \vee$$
$$\left(y = \tfrac{5}{\sqrt{17}} \wedge -\sqrt{25-y^2} < x < \tfrac{5\sqrt{17}}{4} - \tfrac{y}{4}\right) \vee \left(\tfrac{5}{\sqrt{17}} < y < 5 \wedge -\sqrt{25-y^2} < x < \sqrt{25-y^2}\right) \vee$$
$$\left(-\tfrac{5}{\sqrt{17}} < y < \tfrac{5}{\sqrt{17}} \wedge -\sqrt{25-y^2} < x \le \tfrac{5\sqrt{17}}{4} - \tfrac{y}{4}\right)$$

$$v_1 = 4; v_2 = 5; \lambda = -\tfrac{3}{5}$$

$$\texttt{unsafeRight} = \texttt{Resolve}\left[\exists t > 0 \wedge (x - v_1 t + \lambda v_2 t)^2 + (y + \sqrt{1-\lambda^2}\, v_2 t)^2 \le 25\right]$$

$$= \left(y < -7\sqrt{\tfrac{5}{13}} \wedge -\tfrac{5\sqrt{65}}{4} - \tfrac{7y}{4} \le x \le \tfrac{5\sqrt{65}}{4} - \tfrac{7y}{4}\right) \vee \left(y = -7\sqrt{\tfrac{5}{13}} \wedge -\tfrac{5\sqrt{65}}{4} - \tfrac{7y}{4} < x \le \tfrac{5\sqrt{65}}{4} - \tfrac{7y}{4}\right) \vee$$
$$\left(y = 7\sqrt{\tfrac{5}{13}} \wedge -\sqrt{25-y^2} < x < \tfrac{5\sqrt{65}}{4} - \tfrac{7y}{4}\right) \vee \left(7\sqrt{\tfrac{5}{13}} < y < 5 \wedge -\sqrt{25-y^2} < x < \sqrt{25-y^2}\right) \vee$$
$$\left(-7\sqrt{\tfrac{5}{13}} < y < 7\sqrt{\tfrac{5}{13}} \wedge -\sqrt{25-y^2} < x \le \tfrac{5\sqrt{65}}{4} - \tfrac{7y}{4}\right)$$

Safe Sets



(a) unsafeCruise

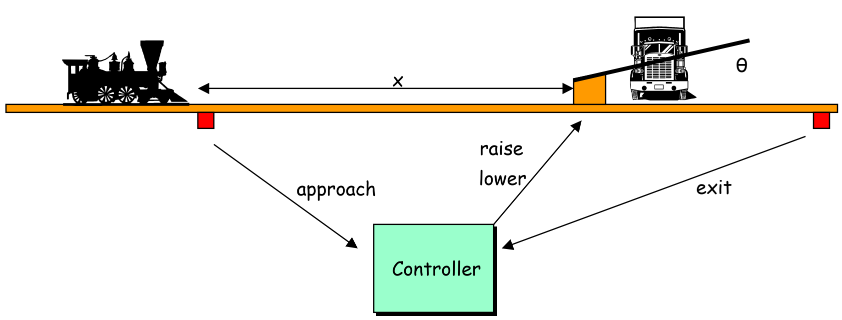
(b) unsafeLeft

(c) unsafeRight

(d) unsafeCruise ∧ unsafeLeft ∧ unsafeRight

Partial synchronization (Concurrency)

The train gate

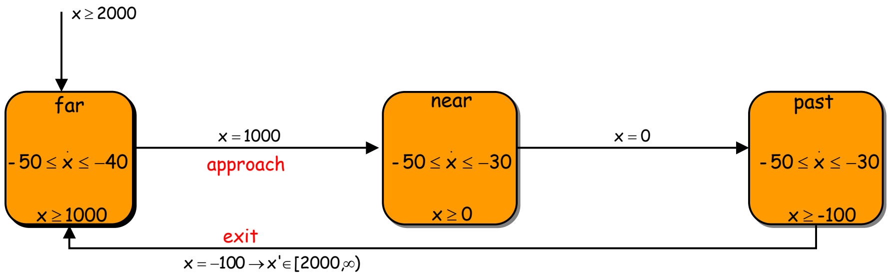


System = Train || Gate || Controller

Safety specification : If train is within 10 meters of the crossing, then gate should completely closed.

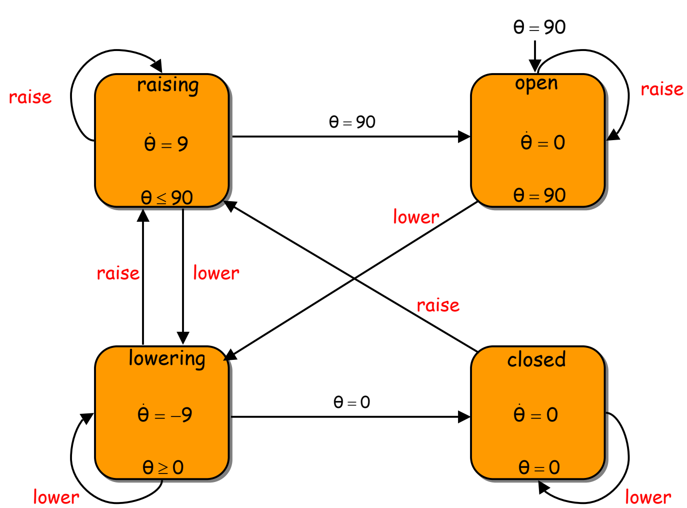
Liveness specification : Keep gate open as much as possible.

Train model



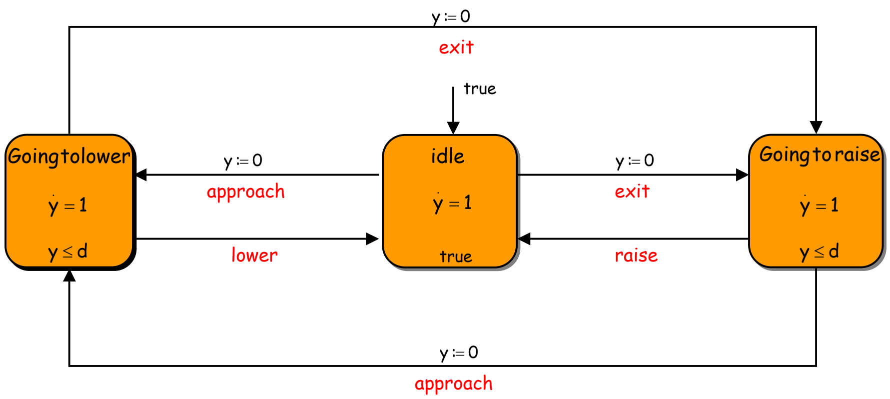
Penn

Gate model

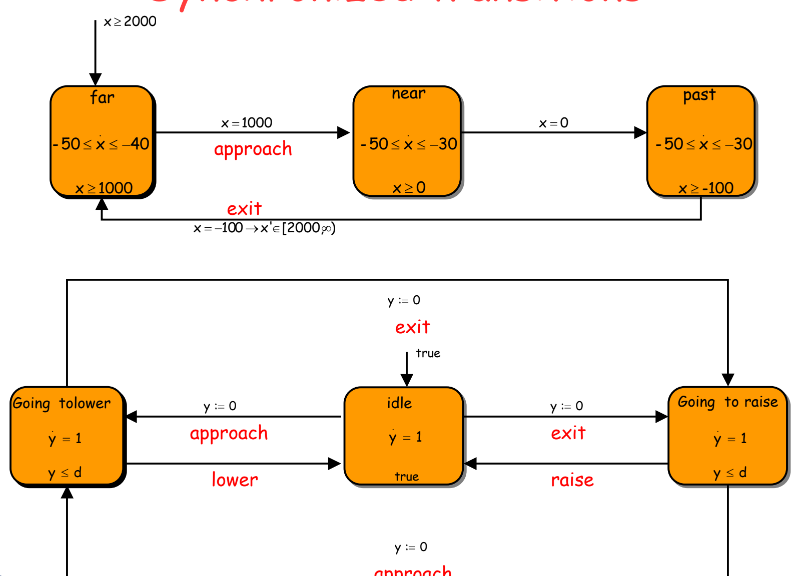


Penn

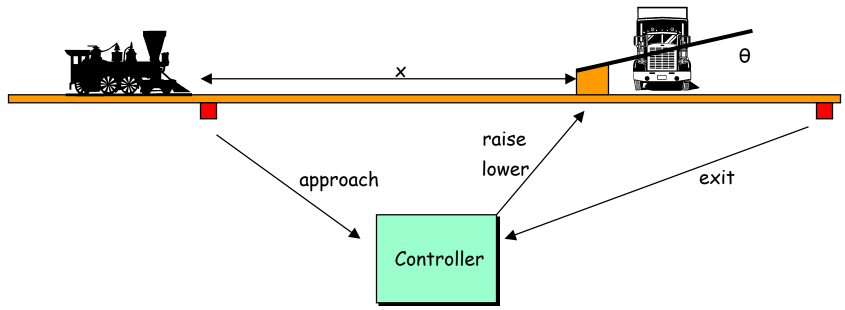
Controller model



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 $d \le \frac{49}{5}$

Penn

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

Penn

Outline of lectures

Lecture 1 : Thursday, September 23

- Examples of hybrid systems and modeling formalisms
- **Transitions systems, temporal logics, abstraction**
- Discrete abstractions of hybrid systems for verification

Lecture 2 : Friday, September 24

- Applications in motion planning and visibility games

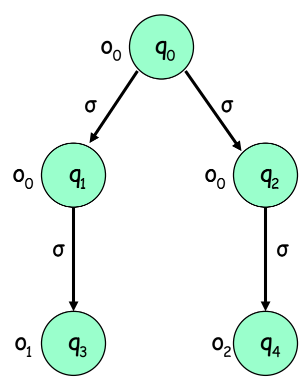
Transition Systems

A transition system

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

consists of

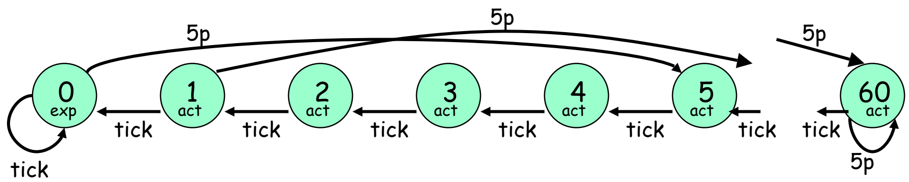
- A set of states Q
- A set of events Σ
- A set of observations O
- The transition relation $q_1 \xrightarrow{\sigma} q_2$
- The observation map $\langle q_1 \rangle = o_0$



Initial or final states may be incorporated
The sets Q, Σ, and O may be infinite
Language of T is all sequences of observations

A painful example

The parking meter



States Q ={0,1,2,…,60}

Events {tick,5p}

Observations {exp,act}

A possible string of observations (exp,act,act,act,act,act,exp,…)

Penn

A familiar example

$$T^{\Delta} = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle)$$

Δ

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k$$

Transition System T^{Δ}

State set $Q = X = R^n$

Label set $\Sigma = U = R^m$

Observation set $O = Y = R^p$

Linear Observation Map $\langle x \rangle = Cx$

Transition Relation $\rightarrow \subseteq X \times U \times X$

$$x_1 \xrightarrow{u} x_2 \Leftrightarrow x_2 = Ax_1 + Bu$$

Penn

Transition Systems

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

We define the following operators

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

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

Penn

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

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

Penn

Basic safety problems

Given transition system T and regions P, S determine

Forward Reachability

$$Post^*(P) \cap S \neq \emptyset$$

Backward Reachability

$$P \cap Pre^*(S) \neq \emptyset$$

Forward reachability algorithm

Forward Reachability Algorithm

initialize $R := P$
while **TRUE** **do**
  if $R \cap S \neq \emptyset$ **return UNSAFE ; end if;**
  if $Post(R) \subseteq R$ **return SAFE ; end if;**
  $R := R \cup Post(R)$
end while

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

Complexity : $O(n_I + m_R)$

(n_I: initial states; m_R: 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 T is infinite, then there is no guarantee of termination.

Penn
UNIVERSITY of PENNSYLVANIA

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

Penn
UNIVERSITY of PENNSYLVANIA

More complicated problems

More sophisticated properties can be expressed using

- Linear Temporal Logic (LTL)
- Computation Tree Logic (CTL)
- CTL*
- mu-calculus

The basic verification problem

Given transition system T, and temporal logic formula φ

Basic verification problem

$$T \models \varphi$$

Two main approaches

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

Another verification problem

Given transition system T, and specification system S

Another verification problem

$$L(T) \subseteq L(S)$$

Language inclusion problems

Penn

The basic synthesis problem

Given transition system T, and temporal logic formula φ

Basic synthesis problem

$$T \parallel C \models \varphi$$

Synthesis in computer science assumes disturbances

Deep relationship between synthesis and game theory

Penn

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

$\varphi_1 \vee \varphi_2$ $\qquad \neg\varphi_1$

Temporal operators

If φ_1 and φ_2 are formulas then

$\varphi_1 \; U \; \varphi_2$ $\qquad \bigcirc \varphi_1$

Linear temporal logic semantics

The LTL formulas are interpreted over infinite (omega) words

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

$$(w,i) \models p \quad \text{iff} \quad p_i = p$$

$$(w,i) \models \varphi_1 \vee \varphi_2 \quad \text{iff} \quad (w,i) \models \varphi_1 \;\text{ or }\; (w,i) \models \varphi_2$$

$$(w,i) \models \neg\varphi_1 \;\text{ iff }\; (w,i) \not\models \varphi_1$$

$$(w,i) \models \bigcirc\, \varphi_1 \quad \text{iff} \quad (w,i+1) \models \varphi_1$$

$$(w,i) \models \varphi_1\, U\, \varphi_2$$

$$\exists j \geq i\ (w,j) \models \varphi_2 \quad \text{and} \quad \forall\, i \leq k \leq j \quad (w,k) \models \varphi_2$$

$$w \models \phi \quad \text{iff} \quad (w,0) \models \varphi$$

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

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 transition system and LTL formula we have

LTL model checking

Determine if $T \models \varphi$ → System verified / Counterexample

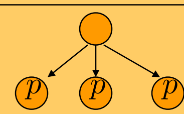
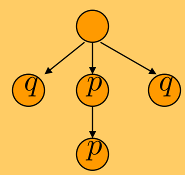
LTL model checking is decidable for finite T

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

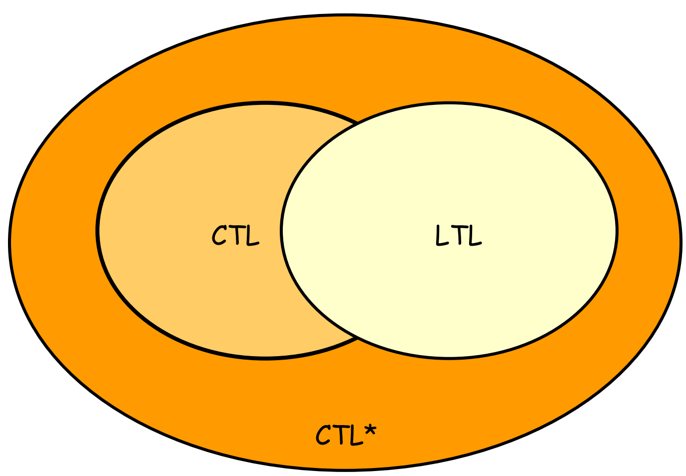
states (n), transitions (m), formula length (k)

Computation tree logic (informally)

Express specifications in computation trees (branching time)

Informally	Syntax	Semantics
Inevitably next p	$\forall \bigcirc p$	
Possibly always p	$\exists \Box p$	

Comparing logics



Dealing with complexity

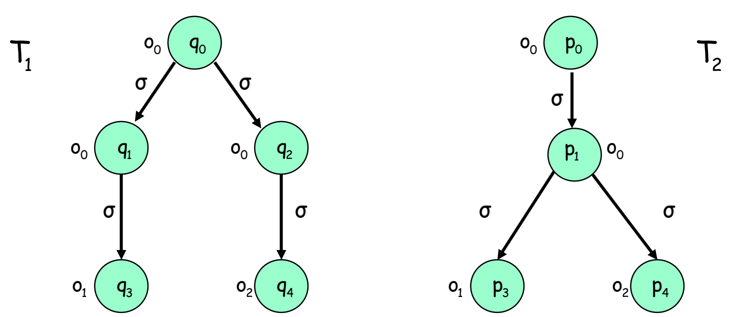
Bisimulation

Simulation

Language Inclusion

Language Equivalence

Consider two transition systems T_1 and T_2 over same Σ and O



Languanges are equivalent $L(T_1)=L(T_2)$

LTL equivalence

Consider two transition systems T_1 and T_2 and an LTL formula

Language equivalence

$$\text{If } L(T_1) = L(T_2) \text{ then } T_1 \models \varphi \Leftrightarrow T_2 \models \varphi$$

Language inclusion

$$\text{If } L(T_1) \subseteq L(T_2) \text{ then } T_2 \models \varphi \Rightarrow T_1 \models \varphi$$

Language equivalence and inclusion are difficult to check

Penn
University of Pennsylvania

Simulation Relations

Consider two transition systems

$$T_1 = (Q_1, \Sigma, \rightarrow_1, O, \langle\cdot\rangle_1)$$
$$T_2 = (Q_2, \Sigma, \rightarrow_2, O, \langle\cdot\rangle_2)$$

over the same set of labels and observations. A relation $S \subseteq Q_1 \times Q_2$ is called a simulation relation if it

1. Respects observations

   if $(q,p) \in S$ then $\langle q\rangle_1 = \langle p\rangle_2$

2. Respects transitions

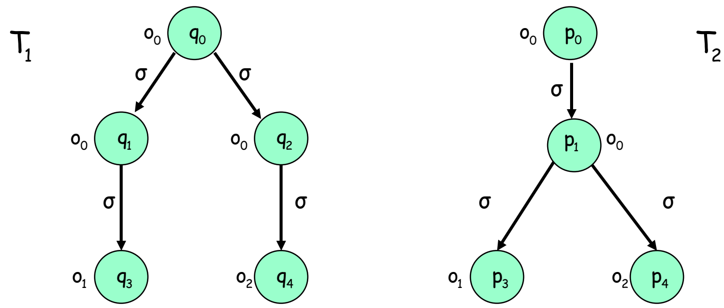
   if $(q,p) \in S$ and $q \xrightarrow{\sigma} q'$, then $p \xrightarrow{\sigma} p'$ for some $(q',p') \in S$

If a simulation relation exists, then $T_1 \leq T_2$

Penn
University of Pennsylvania

Game theoretic semantics

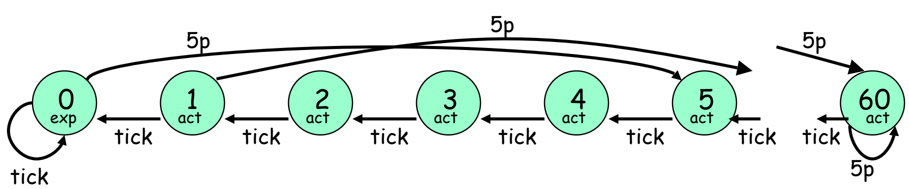
Simulation is a matching game between the systems



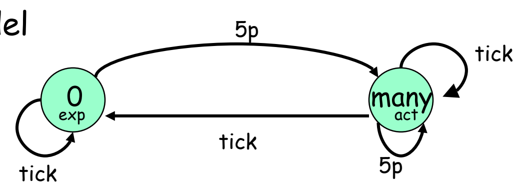
Check that $T_1 \leq T_2$ but it is not true that $T_2 \leq T_1$

The parking example

The parking meter



A coarser model



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

Simulation relations

Consider two transition systems T_1 and T_2

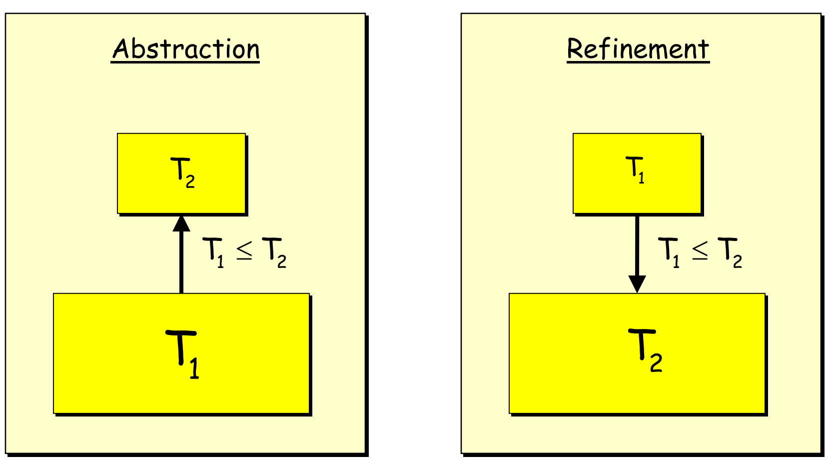
Simulation implies language inclusion

$$\text{If } T_1 \leq T_2 \text{ then } L(T_1) \subseteq L(T_2)$$

Complexity of $L(T_1) \subseteq L(T_2)$ $\quad O((n_1+m_1)2^{n_2})$

Complexity of $T_1 \leq T_2$ $\quad O((n_1+m_1)(n_2+m_2))$

Two important cases



Bisimulation

Consider two transition systems T_1 and T_2

Bisimulation

$$T_1 \equiv T_2 \;\text{ if }\; T_1 \leq T_2 \;\wedge\; T_2 \leq T_1$$

Bisimulation is a symmetric simulation

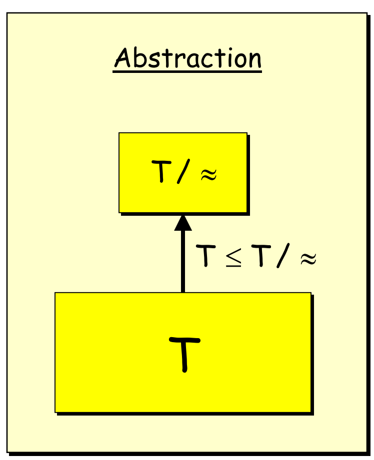
Strong notion of equivalence for transition systems

CTL* (and LTL) equivalence

$$\text{If } T_1 \equiv T_2 \text{ then } T_1 \models \varphi \Leftrightarrow T_2 \models \varphi$$

$$\text{If } T_1 \equiv T_2 \text{ then } L(T_1) = L(T_2)$$

Special quotients



When is the quotient language equivalent or bisimilar to T ?

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 $\langle p \rangle = o$

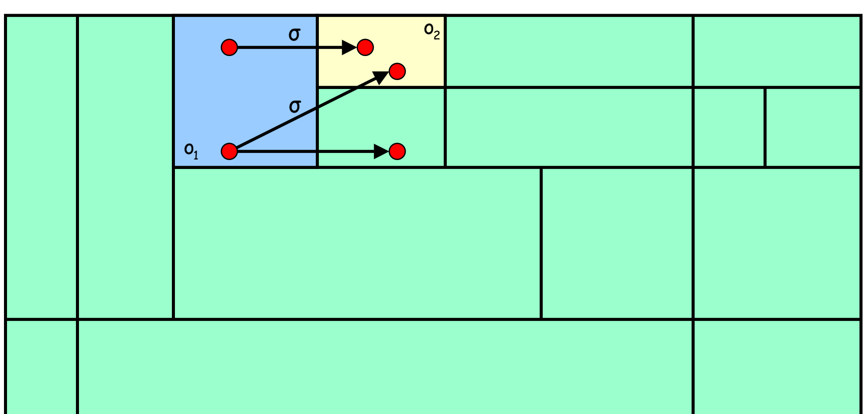
2. Transition Relation

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

Bisimulation Algorithm

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

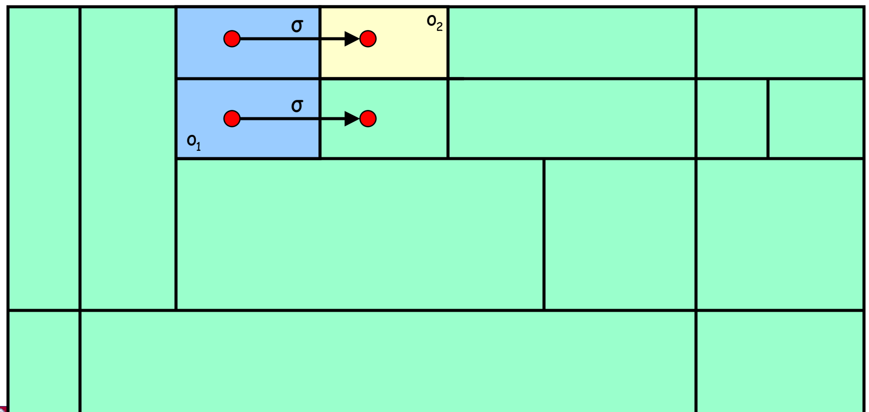
When does original system T simulate the quotient system $T/\approx$?



Bisimulation Algorithm

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

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



Penn
University of Pennsylvania

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

Penn
University of Pennsylvania

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

$O(m \cdot 2^n)$

Outline of lectures

Lecture 1 : Thursday, September 23

Examples of hybrid systems and modeling formalisms

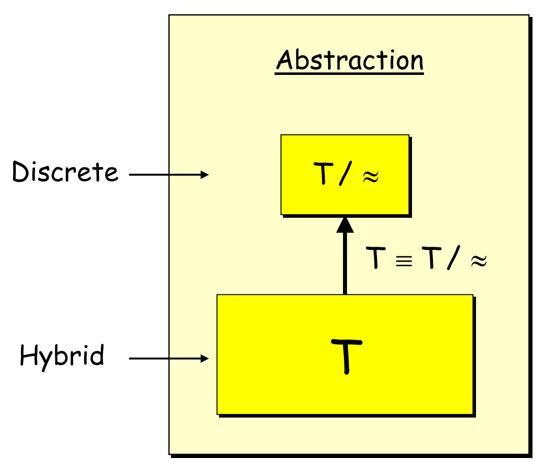
Transitions systems, temporal logics, abstraction

Discrete abstractions of hybrid systems for verification

Lecture 2 : Friday, September 24

Applications in motion planning and visibility games

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

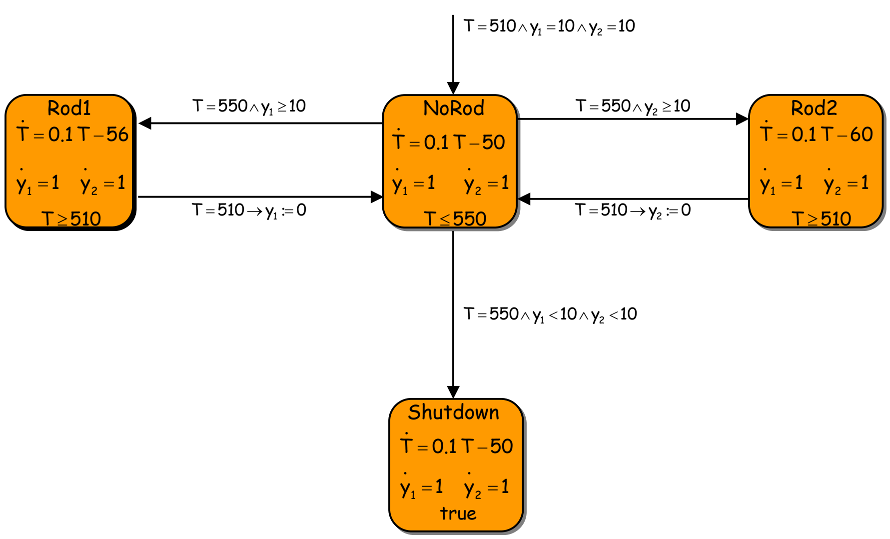
Define $E = \{(l, l') |\ \exists x \in Inv(l), x' \in Inv(l')\ ((l, x), (l', x')) \in R\}$

$$Init(l) = \{x \in Inv(l)\ |\ (l, x) \in X_0\}$$

$$Guard(e) = \{x \in Inv(l) |\ \exists x' \in Inv(l')\ ((l, x), (l', x')) \in R\}$$

$$Reset(e, x) = \{x' \in Inv(l') |\ ((l, x), (l', x')) \in R\}$$

An example



Transitions of Hybrid Systems

Hybrid systems can be embedded into transition systems

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

$Q = V \times \Re^n$

$Q_0 = X_0$

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

$\rightarrow \subseteq Q \times \Sigma \times Q$

Observation set and map depend on desired properties

Discrete transitions

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

Continuous (time-abstract) transitions

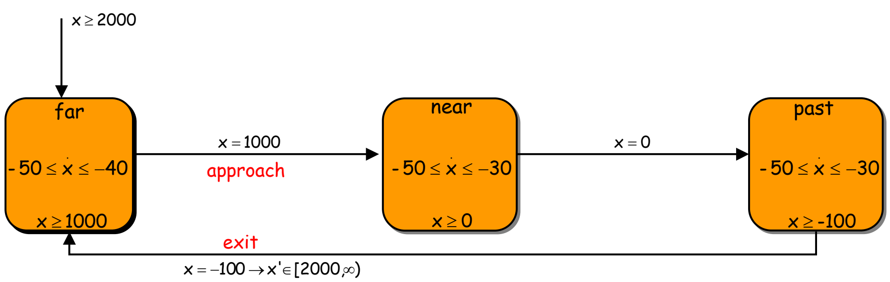
$$(l_1, x_1) \xrightarrow{\tau} (l_2, x_2) \quad \text{iff} \quad l_1 = l_2 \quad \text{and} \quad \exists \delta \geq 0 \quad x(\cdot) : [0, \delta] \rightarrow \Re^n$$

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

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

Rectangular hybrid automata

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

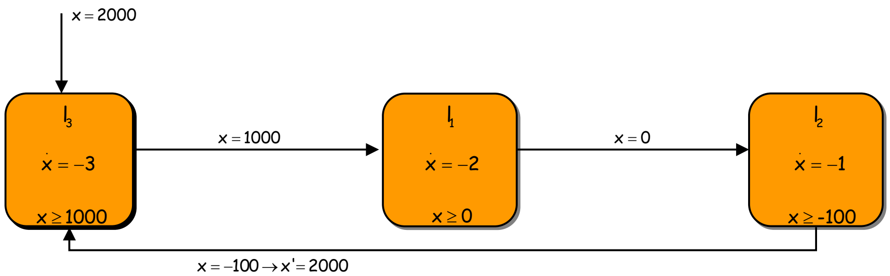


Rectangular hybrid automata are hybrid systems where

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

are rectangular sets

Multi-rate automata

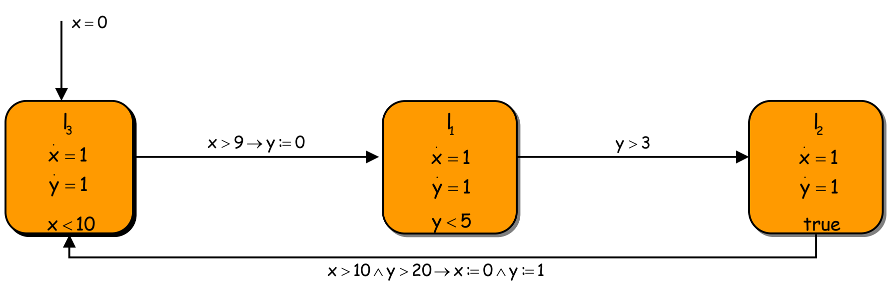


Multi-rate automata are rectangular hybrid automata where

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

are singleton sets

Timed automata



Timed automata are multi-rate automata where

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

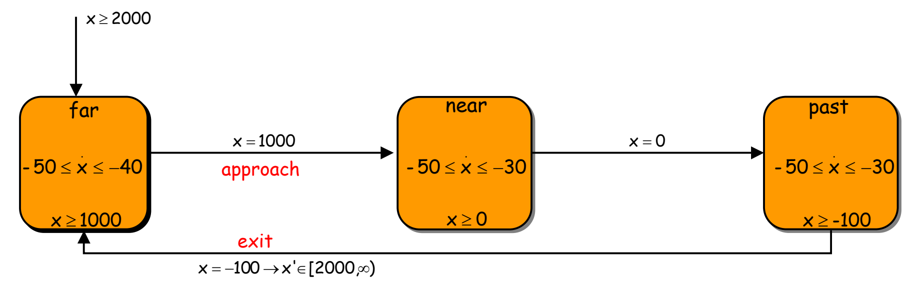
for all locations l and all variables.

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).

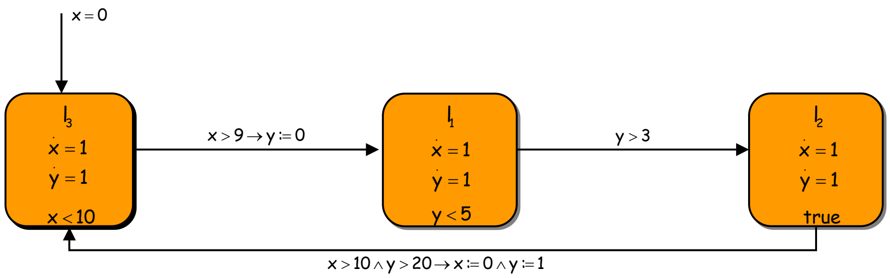
The reachability problem is undecidable for this class.

No algorithmic procedure exists.

Model checking temporal logic formulas is also undecidable

Initalization is necessary for decidability

Timed automata

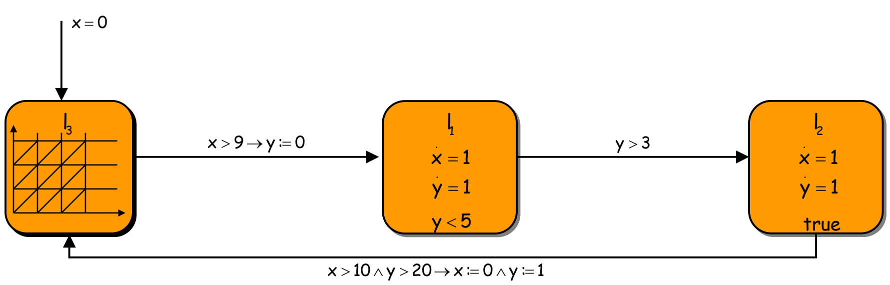


All timed automata admit a finite bisimulation

Hence CTL* model checking is decidable for timed automata

Penn

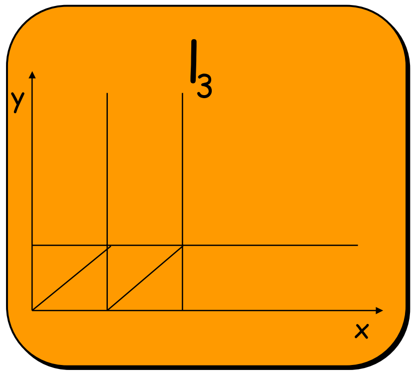
Timed automata



Approach : Discretize the clock dynamics using region equivalence

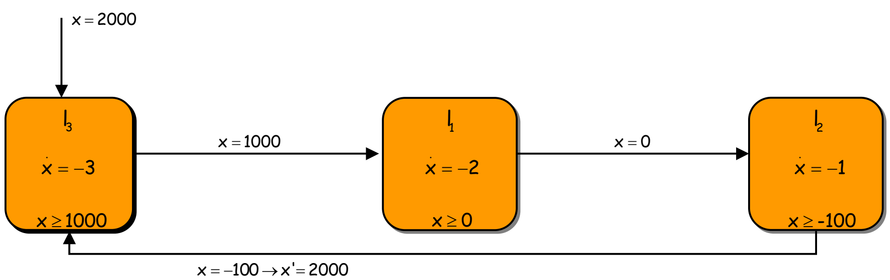
Penn

Region equivalence



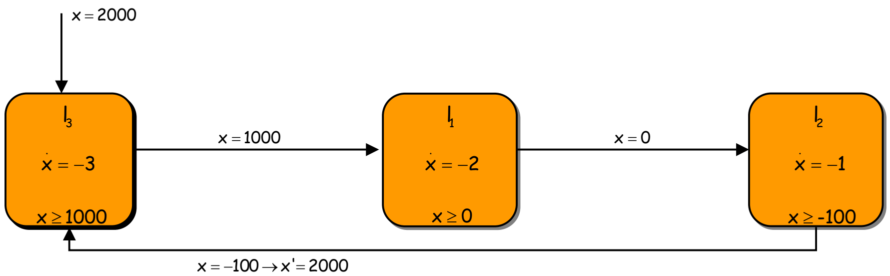
Equivalence classes : 6 corner points
14 open line segments
8 open regions

Multi-rate automata



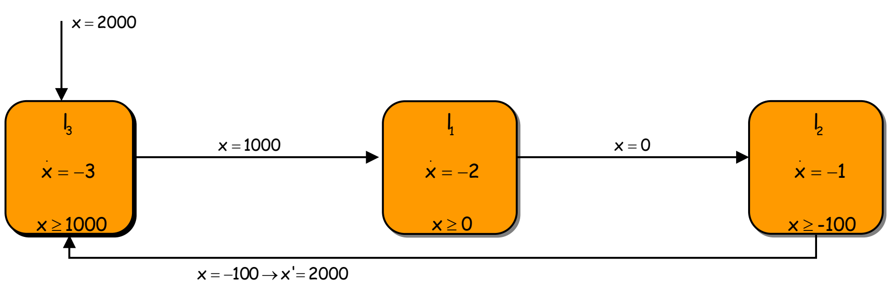
All initialized multi-rate automata admit a finite bisimulation

Rectangular automata



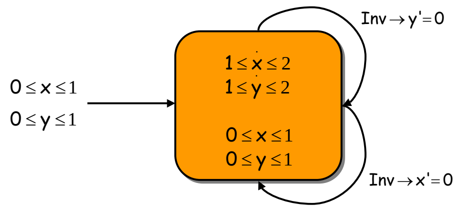
All initialized rectangular automata admit a finite bisimulation

Rectangular automata



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

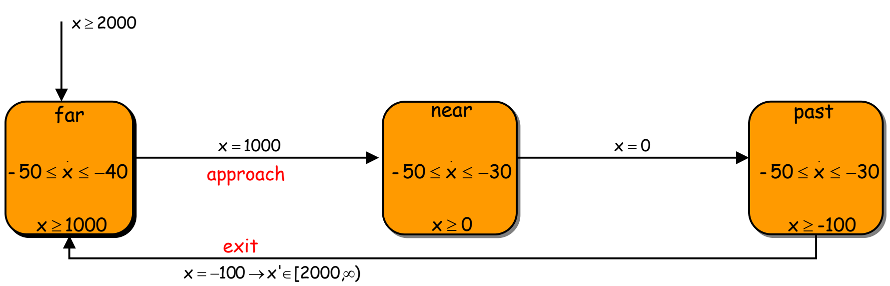
No finite bisimulation



Bisimulation algorithm never terminates

Penn

but…

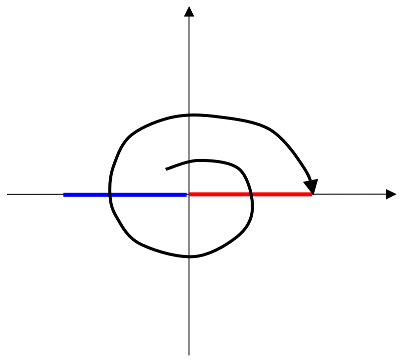


All initialized rectangular automata admit a finite language equivalence quotient which can be constructed effectively.

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

Reminder

Representation issues

- Symbolic representation for infinite sets
- Rectangular sets ? Semi-linear ? Semi-algebraic ?

Operations on sets

- Boolean (logical) operations
- Can we compute Pre and Post ?
- Is our representation closed under Pre and Post ?

Algorithmic termination (decidability)

- No guarantee for infinite transition systems
- We need "nice" alignment of sets and flows
- Globally finite properties

First-order logic

Every theory of the reals has an associated language

$$(\Re, <, +, -, 0, 1)$$

Universe: $\Re$; Relation: $<$; Functions: $+$, $-$; Constants: 0, 1

Variables : $x_1, x_2, x_3, \ldots$

TERMS : Variables, constants, or functions of them

$$x_1 - x_2 + 1, 1 + 1, -x_3$$

ATOMIC FORMULAS : Apply the relation and equality to the terms

$$x_1 + x_2 < -1, 2x_1 = 1, x_1 = x_3$$

(FIRST ORDER) FORMULAS : Atomic formulas are formulas
If φ_1, φ_2 are formulas, then $\varphi_1 \vee \varphi_2, \neg\varphi_1, \forall x.\varphi_1, \exists x.\varphi_1$

First-order logic

Useful languages

$(\Re, <, +, -, 0, 1)$	$\forall x \forall y (x + 2y \geq 0)$
$(\Re, <, +, -, \times, 0, 1)$	$\exists x . ax^2 + bx + c = 0$
$(\Re, <, +, -, \times, e^x, 0, 1)$	$\exists t . (t \geq 0) \wedge (y = e^t x)$

A theory of the reals is **decidable** if there is an algorithm which in a finite number of steps will decide whether a formula is true or not

A theory of the reals admits **quantifier elimination** if there is an algorithm which will eliminate all quantified variables.

$$\exists x . ax^2 + bx + c = 0 \equiv b^2 - 4ac \geq 0$$

First-order logic

Theory	Decidable ?	Quant. Elim. ?
$(\Re, <, +, -, 0, 1)$	YES	YES
$(\Re, <, +, -, \times, 0, 1)$	YES	YES
$(\Re, <, +, -, \times, e^x, 0, 1)$	?	NO

Tarski's result : Every formula in $(\Re, <, +, -, \times, 0, 1)$ can be decided

1. Eliminate quantified variables
2. Quantifier free formulas can be decided

Penn University of Pennsylvania

O-Minimal Theories

A definable set is $Y = \{(x_1, x_2, \ldots, x_n) \in \Re^n \mid \varphi(x_1, \ldots, x_n)\}$

A theory of the reals is called **o-minimal** if every definable subset of the reals is a finite union of points and intervals

Example: $Y = \{(x) \in \Re \mid p(x) \geq 0\}$ for polynomial p(x)

Recent o-minimal theories

$(\Re, <, +, -, 0, 1)$

$(\Re, <, +, -, \times, 0, 1)$

$(\Re, <, +, -, \times, e^x, 0, 1)$ → Related to Hilbert's 16th problem

$(\Re, <, +, -, \times, \hat{f}, 0, 1)$

$(\Re, <, +, -, \times, \hat{f}, e^x, 0, 1)$

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.

Corollaries

$(\Re, <, +, -, 0, 1)$

Consider continuous systems where

- Finite partition is polyhedral (semi-linear)
- Vector fields have linear flows (timed, multi-rate)

Then a finite bisimulation exists.

$(\Re, <, +, -, \times, 0, 1)$

Consider continuous systems where

- Finite partition is semialgebraic
- Vector fields have polynomial flows

Then a finite bisimulation exists.

Corollaries

$(\Re, <, +, -, \times, e^x, 0, 1)$

Consider continuous systems where

- Finite partition is semi-algebraic
- Vector fields are linear with real eigenvalues

Then a finite bisimulation exists.

$(\Re, <, +, -, \times, \hat{f}, 0, 1)$

Consider continuous systems where

- Finite partition is sub-analytic
- Vector fields are linear with purely imaginary eigenvalues

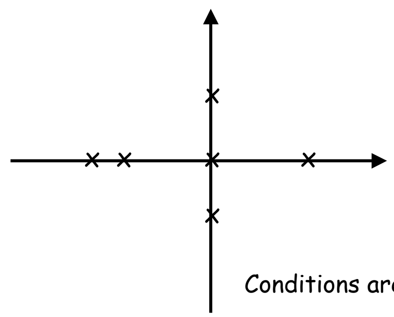
Then a finite bisimulation exists.

Corollaries

$(\Re,<,+,-,\times,\hat{f},e^x,0,1)$ Consider continuous systems where

- Finite partition is semi-algebraic
- Vector fields are linear with real or imginary eigenvalues

Then a finite bisimulation exists.



Conditions are sufficient but tight

Computability

Finite bisimulations exist, but can we compute them ?

Bisimulation Algorithm

initialize $Q/_\sim = \{p \sim q \ \text{ iff } \ <q>=<p>\}$
while $\exists P, P' \in Q/_\sim$ **such that** $\emptyset \neq \subsetneq P \cap Pre(P') \neq \subsetneq P'$

$P_1 := P \cap Pre(P')$

$P_2 := P \setminus Pre(P')$

$Q/_\sim := (Q/_\sim \setminus \{P\}) \cup \{P_1, P_2\}$

end while

Need to : Check emptiness
Perform boolean operations
Compute Pre (or Post)

Use $(\Re,<,+,-,\times,0,1)$

Computing reachable sets

Consider a linear system

$$\frac{dx}{dt} = Ax \quad A \in Q^{n \times n}$$ ← Rational entries

and a semi-algebraic set Y. If

$$Y = \{ y \in \Re^n \mid p(y) \}$$

Then

$$Pre(Y) = \{x \in \Re^n \mid \exists y \exists t. p(y) \wedge t \geq 0 \wedge x = e^{-tA} y\}$$

↑ Problem?

Penn University of Pennsylvania

Nilpotent Linear Systems

Nilpotent matrices: $\exists n \geq 0 \quad A^n = 0$

Then flow of linear system is polynomial

$$e^{-tA} = \sum_{k=0}^{n-1} (-1)^k \frac{t^k}{k!} A^k$$

Therefore Pre(Y) completely definable in $(\Re, <, +, -, \times, 0, 1)$

$$Pre(Y) = \{x \in \Re^n \mid \exists y \exists t. p(y) \wedge t \geq 0 \wedge x = \sum_{k=0}^{n-1} (-1)^k \frac{t^k}{k!} A^k y\}$$

Penn University of Pennsylvania

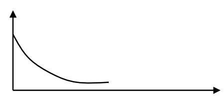
Diagonalizable, rational eigenvalues

Example system : $\dot{x} = 2x$

Compute all states that can reach the set Y = { y=5 }

$$Pre(Y) = \{x \in \Re \mid \exists y \exists t.\ y = 5 \wedge t \geq 0 \wedge x = e^{-2t} y\}$$

Let $s = e^{-t}$, then



$$Pre(Y) = \{x \in \Re \mid \exists y \exists t.\ y = 5 \wedge 1 \geq s \geq 0 \wedge x = s^2 y\}$$

$$Pre(Y) = \{x \in \Re \mid \ 0 < x \leq 5\}$$

Penn

Diagonalizable, rational eigenvalues

More generally $\dot{x} = Ax \Rightarrow x(t) = Te^{\Lambda t} T^{-1} x(0)$

Therefore $e^{-tA} = \left[\sum_{k=1}^{n} a_{ijk} e^{-\lambda_k t} \right]_{ij}$

1. Rescale rational eigenvalues to integer eingenvalues.
2. Eliminate negative integer eigenvalues
3. Perform the substitution $s = e^{-t}$

> Consider diagonalizable linear vector fields with real, rational eigenvalues, and let Y be a semi-algebraic set. Then Pre(Y) is also semi-algebraic (and computable)

Penn

Diagonalizable, imaginary eigenvalues

Procedure is similar if system is diagonalizable with purely imaginary, rational eigenvalues

Equivalence is obtained by $z_1 = \cos(t)$ $z_2 = \sin(t)$
Suffices to compute over a period

Consider diagonalizable linear vector fields with real, rational eigenvalues, and let Y be a semi-algebraic set. Then Pre(Y) is also semi-algebraic (and computable)

Composing all computability results together results in…

Penn
University of Pennsylvania

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
University of Pennsylvania

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.

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.

Outline of lectures

Lecture 1 : Thursday, September 23

Examples of hybrid systems and modeling formalisms

Transitions systems, temporal logics, abstraction

Discrete abstractions of hybrid systems for verification

Bisimulations of continuous systems (if time permits)

Lecture 2 : Friday, September 24

Applications in motion planning and visibility games

Controller synthesis

The main (controller) synthesis equation

$$A || X \cong B$$

or a more relaxed version…

$$A || X \leq B$$

Equations can be interpreted over various model types
Various semantics of composition and equivalence

Discrete semantics

The main (controller) synthesis equation

$$A || X \cong B$$

or a more relaxed version...

$$A || X \leq B$$

Models : Finite state automata
Composition :
Equivalence :
Order : $L(A||B) = L(A) \cap L(X)$

$A \cong B$ iff $L(A) = L(X)$

$A \leq B$ iff $L(A) \subseteq L(X)$

Continuous semantics

The main (controller) synthesis equation

$$A || X \cong B$$

or a more relaxed version...

$$A || X \leq B$$

Models : Control systems
Composition : Feedback composition
Equivalence : Asymptotic equivalence
Order : Not much...

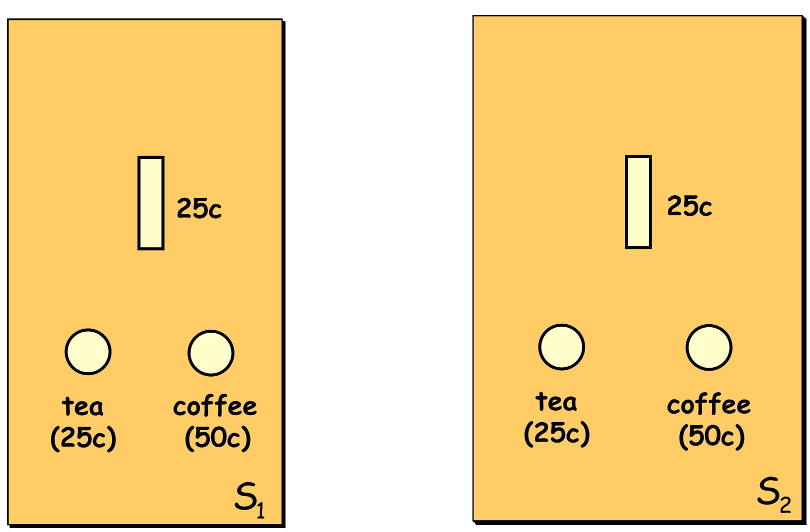
Notions of equivalence

Language equivalence for finite state systems has served us well as a notion of system equivalence for systems which are NOT interacting with other systems.

Asymptotic equivalence for control systems has served us well as a notion of system equivalence for systems which are NOT interacting with other systems.

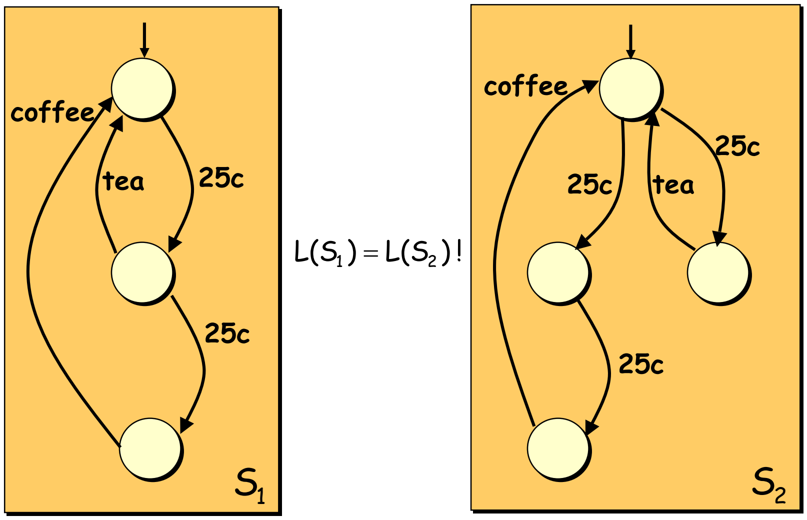
Challenge : Reactive notions of system equivalence

Two coffee machines*



*R. Milner, Communicating and mobile systems : the pi-calculus, Cambridge University Press, 1999

Two coffee machines



$L(S_1) = L(S_2)$!

Nondeterminism !

Penn

Simulation Relations

Consider two transition systems

$$S_1 = (Q_1, i_1, \Sigma, \rightarrow_1)$$
$$S_2 = (Q_2, i_2, \Sigma, \rightarrow_2)$$

over the same set of labels and observations. A relation $R \subseteq Q_1 \times Q_2$ is called a simulation relation if it

1. Respects initial states $(i_1, i_2) \in R$
2. Respects transitions

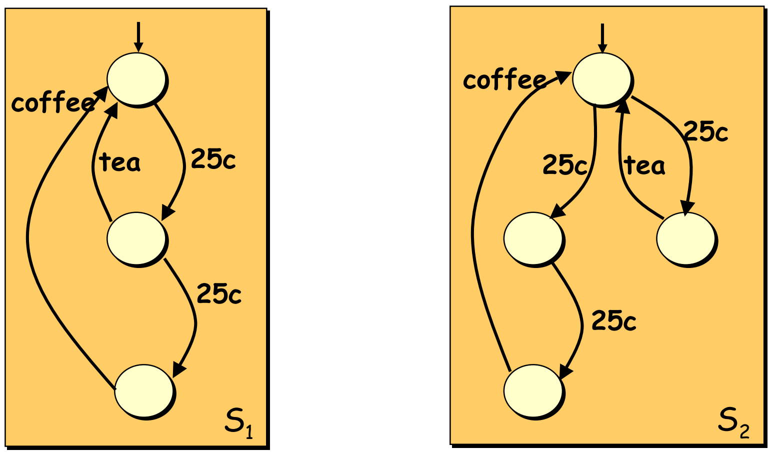
$$\begin{array}{ccc} q_1 & \xrightarrow{\sigma} & q_1' \\ R & & R \\ q_2 & \xrightarrow{\sigma} & q_2' \end{array}$$

If a simulation relation exists, then $S_1 \leq S_2$

Penn

Game theoretic semantics

Simulation is a matching game between the systems



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

Penn

Relationships

Simulation
$\forall$CTL

↓

Language Inclusion
Reachability

Bisimulation
CTL*

↓

Language Equivalence
LTL

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

Converse statements are true for **deterministic** systems

Penn

Bi-simulations of control systems*

$u_1(t)$ →

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

S_1 → $y_1(t)$

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

$u_2(t)$ →

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

S_2 → $y_2(t)$

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

*G.J. Pappas, G. Lafferriere, and S. Sastry, Hierarchically Consistent Control Systems, IEEE TAC, June 2000

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

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

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

Penn

Non-deterministic dynamics

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

S_1 → $y_1(t)$

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

S_2 → $y_2(t)$

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

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

$$C_1 x_1(t) = C_2 x_2(t)$$

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

Penn

H-related systems*

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

S_1 → $y_1(t)$

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

S_2 → $y_2(t)$

A linear relation (x,Hx) is a simulation relation iff for all $\forall d_1 \quad \exists d_2$

$$H(A_1 x_1 + E_1 d_1) = A_2 H x_1 + E_2 d_2$$
$$C_1 = C_2 H$$

S_2 simulates or is H-related to S_1

Penn

*G.J. Pappas, G. Lafferriere, and S. Sastry, Hierarchically Consistent Control Systems, IEEE TAC, June 2000

Deterministic systems

$$\dot{x}_1(t) = A_1 x_1(t)$$
$$y_1(t) = C_1 x_1(t)$$

S_1 → $y_1(t)$

$$\dot{x}_2(t) = A_2 x_2(t)$$
$$y_2(t) = C_2 x_2(t)$$

S_2 → $y_2(t)$

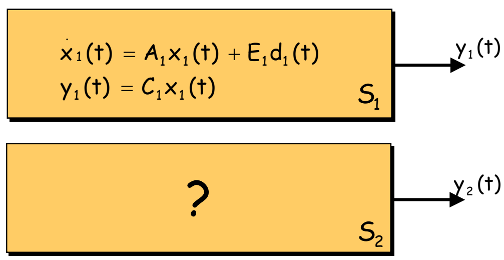
A linear relation (x,Hx) is a simulation relation iff

$$HA_1 = A_2 H$$
$$C_1 = C_2 H$$

Restrictive!

Penn

Advantage of non-determinism



Given surjective map $x_2 = Hx_1$ can we construct S_2 simulating S_1 ?

$$A_2 = HA_1H^+$$
$$E_2 = [HE_1 \quad HA_1Ker(H)]$$
$$C_2 = C_1H^+ \quad \text{if } Ker(H) \subseteq Ker(C_1)$$

*G.J. Pappas, G. Lafferriere, and S. Sastry, Hierarchically Consistent Control Systems, IEEE TAC, June 2000

Two remarks

Abstraction is **always** possible in the class of nondeterministic systems

The more you abstract, the more non-determinism you generate

Bi-simulation is finer

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = d_2(t)$$
$$y_1(t) = x_1(t)$$

S_1

$z_1 = x_1$ →

$$\dot{z}_1(t) = d_1(t)$$
$$y_1(t) = z_1(t)$$

S_2

$$L(S_1) = L(S_2)$$
$$S_1 \leq S_2$$
$$S_2 \not\leq S_1$$
$$S_1 \not\cong S_2$$

When is (x,Hx) a bisimulation relation ?

Penn

Bisimilar linear systems*

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

S_1 → $y_1(t)$

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

S_2 → $y_2(t)$

Let S_2 be H-related to S_1. Then the relation (x,Hx) is a bi-simulation relation if and only if

$$A_1 \mathrm{Ker}(H) \subseteq \mathrm{Ker}(H) + R(E_1)$$

Penn

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

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

Coarsest Bisimulation

Find map $x_2 = Hx_1$ which abstracts as much as possible.
Thus Ker(H) must be maximal but also must…

Preserve observations

$$Ker(H) \subseteq Ker(C_1)$$

Preserve transitions

$$A_1 Ker(H) \subseteq Ker(H) + R(E_1)$$

This lead to the well known algorithm…

Coarsest Bisimulation Algorithm

Maximal controlled invariant subspace computation

$$V_0 = Ker(C_1)$$

$$V_{k+1} = V_{k-1} \cap A_1^{-1}(V_{k-1} + R(E_1))$$

Then $V^* = V_n$ is the maximal desired subspace

Once V* is computed, then pick map $x_2 = Hx_1$ such that

Ker(H)=V*

and construct the H-related system.

Similar relationships

Simulation → Language Inclusion

Bisimulation → Language Equivalence

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

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

If $S_1 \leq S_2$ then $H(Reach(S_1, X)) \subseteq Reach(S_2, H(X))$

Extensions

Bi-simulations of 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. Grasee, Admissibility of trajectories in Phi-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, Submitted

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