Transition systems, temporal logic, refinement notions

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

DISC Summer School on
Modeling and Control of Hybrid Systems
Veldhoven, The Netherlands
June 23-26, 2003
http://lcewww.et.tudelft.nl/~disc_hs/

Outline of this mini-course

Lecture 1 : Monday, June 23
Examples of hybrid systems, modeling formalisms
Lecture 2 : Monday, June 23
Transitions systems, temporal logic, refinement notions
Lecture 3 : Tuesday, June 24
Discrete abstractions of hybrid systems for verification
Lecture 4 : Tuesday, June 24
Discrete abstractions of continuous systems for control
Lecture 5 : Thursday, June 26
Bisimilar control systems

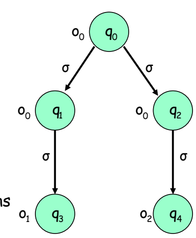
Transition Systems

A transition system

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

consists of

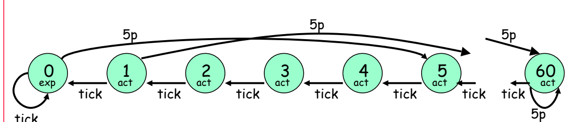
- A set of states Q
- A set of events Σ
- A set of observations O
- The transition relation $q_1 \xrightarrow{\sigma} q_2$
- The observation map $\langle q_1 \rangle = o_0$



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,…)

A familiar example

$T^\Delta = (Q, \Sigma, \to, O, \langle\cdot\rangle)$

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 $\to \subseteq X \times U \times X$

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

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

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

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

Similarly for the other operators. Also

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

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

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

Backward reachability algorithm

Backward Reachability Algorithm

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

If T 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

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

Penn

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	$p \Rightarrow \bigcirc q$	$qqqqqqqqpq$
p until q	$p\, U\, q$	$pppppppppppppppq$

Penn

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
$\varphi_1\, U\, \varphi_2$ $\bigcirc \varphi_1$

Penn

Linear temporal logic semantics

The LTL formulas are interpreted over infinite (omega) words

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

$(w,i) \models p$ iff $p_i = p$

$(w,i) \models \varphi_1 \vee \varphi_2$ iff $(w,i) \models \varphi_1$ or $(w,i) \models \varphi_2$

$(w,i) \models \neg\varphi_1$ iff $(w,i) \not\models \varphi_1$

$(w,i) \models \bigcirc \varphi_1$ iff $(w,i+1) \models \varphi_1$

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

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

$w \models \phi$ iff $(w,0) \models \varphi$

$T \models \phi$ 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 transition system and LTL formula we have

LTL model checking

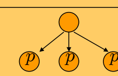
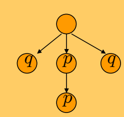
Determine if $T \models \varphi$ → System verified / Counterexample

LTL model checking is decidable for finite T

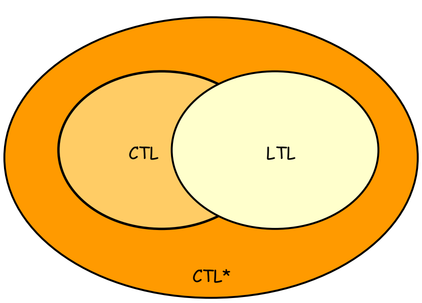
Complexity : $O((\underbrace{n}_{\text{states}} + \underbrace{m}_{\text{transitions}})(k+l)2^{O(\underbrace{k}_{\text{formula length}})})$

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

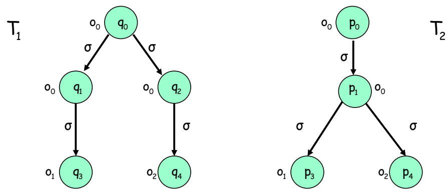
Bisimulation

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

Simulation Relations

Consider two transition systems

$$T_1 = (Q_1, \Sigma, \to_1, O, \langle\cdot\rangle_1)$$
$$T_2 = (Q_2, \Sigma, \to_2, O, \langle\cdot\rangle_2)$$

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 $\langle q\rangle_1 = \langle p\rangle_2$

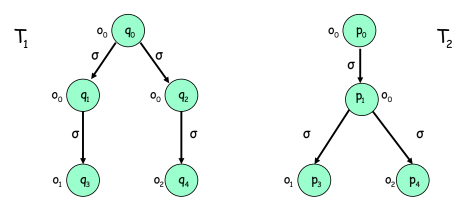
2. Respects transitions

   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 $T_1 \leq T_2$

Game theoretic semantics

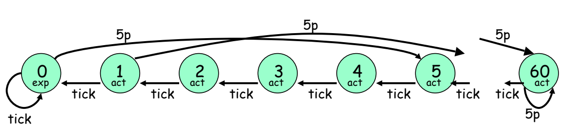
Simulation is a matching game between the systems



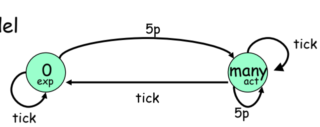
Check that $T_1 \leq T_2$ but it is not true that $T_2 \leq T_1$

The parking example

The parking meter



A coarser model



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

Simulation relations

Consider two transition systems T_1 and T_2

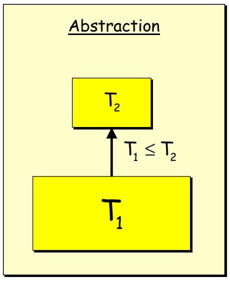
Simulation implies language inclusion

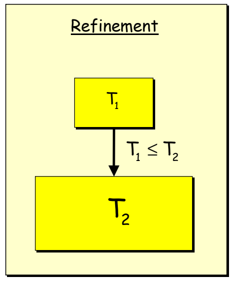
$$\text{If } T_1 \leq T_2 \text{ then } L(T_1) \subseteq L(T_2)$$

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





Penn

Bisimulation

Consider two transition systems T_1 and T_2

Bisimulation

$$T_1 \equiv T_2 \ \text{ if } \ T_1 \le T_2 \ \wedge \ T_2 \le T_1$$

Bisimulation is a symmetric simulation

Strong notion of equivalence for transition systems

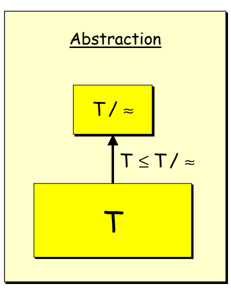
CTL* (and LTL) equivalence

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

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

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

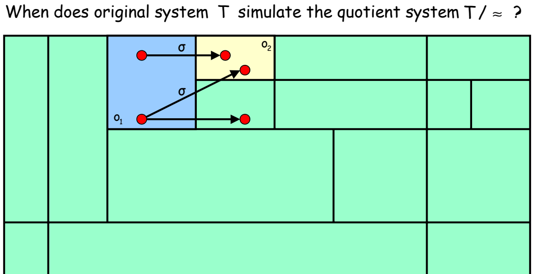
   $P \xrightarrow{\sigma}_{\approx} P'$ iff there exists $p \in P, p' \in P'$ with $p \xrightarrow{\sigma} p'$

Penn

Bisimulation Algorithm

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

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

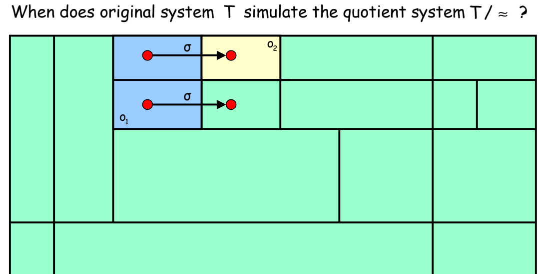


Penn

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

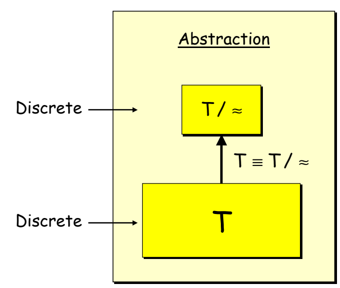
↓

Simulation
$O(m \cdot n)$

↓

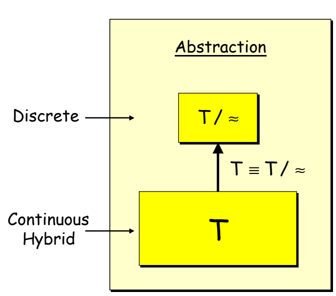
Language Equivalence
$O(m \cdot 2^n)$

Discrete to discrete



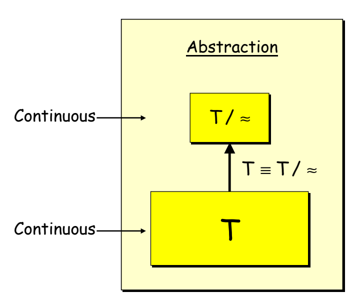
Goal : Complexity reduction, theoretical guarantees

Continuous to discrete (Lectures 3 & 4)



Goal : Algorithmic feasibility, decidability, property dependent quantization

Continuous to continuous (Lecture 5)



Goal : Property dependent reduction, hierarchical control, search for a unified systems theory