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

School Organizers
Marika di Benedetto
Kostas Kyriakopoulos
John Lygeros

and HYBRIDGE
Acknowledgments

Collaborators
Rajeev Alur, Datta Godbole, Tom Henzinger, Ali Jadbabaie, John Koo, Vijay Kumar, Gerardo Lafferriere, 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

Postdocs
Mohamed Babaali
Antoine Girard

Ph.D Students
Ali Ahmazadeh
George Fainekos
Hadas Kress Gazit
Hakan Yazarel
Michael Zavlanos

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!
Some references

Bisimilar linear systems
George J. Pappas

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

Discrete abstractions of hybrid systems
R.Alur, T. Henzinger, G. Lafferriere, G. Pappas

Hierarchically consistent control systems
George J. Pappas, Gerardo Lafferriere, and Shankar Sastry

O-minimal hybrid systems
G. Lafferriere, G. J. Pappas, and S. Sastry

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
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

Networked embedded systems...
Physical system is continuous, software is discrete

Networked embedded systems...

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

**Control Theory**
- Continuous systems
- Stability, control
- Feedback, robustness

**Computer Science**
- Transition systems
- Composition, abstraction
- Concurrency models

**Composition**
- Abstraction
- Concurrency

**Robustness**
- Feedback
- Stability

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...

Logic based switching
Nuclear reactor example

Without rods

With rod 1

\[ \dot{T} = 0.1(T - 50) \]

With rod 2

\[ \dot{T} = 0.1(T - 56) \]

\[ \dot{T} = 0.1(T - 60) \]

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

Rod1 \[ \xrightarrow{\text{NoRod}} \] NoRod \[ \xrightarrow{\text{Rod2}} \] Rod2

\[ \xrightarrow{\text{Shutdown}} \]

Penn
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?

Conflict Resolution Maneuver

- **Cruise**
  - Dynamics:
    - $\dot{x} = x$
    - $\dot{y} = y$
    - $\dot{\phi}_1 = \phi_1$
    - $\dot{\phi}_2 = \phi_2 + \Delta \phi$
  - Invariant:
    - $x^2 + y^2 \geq a_1$
    - $t = 0$

- **Left**
  - Dynamics:
    - $\dot{x} = x$
    - $\dot{y} = y$
    - $\dot{\phi}_1 = \phi_1$
    - $\dot{\phi}_2 = \phi_2 + \Delta \phi$
  - Invariant:
    - $x^2 + y^2 \leq a_1$
    - $t \leq \frac{1}{v_1 + v_2 \sin(\Delta \phi)}$
    - $\Delta t = \frac{2d}{v_1 + v_2 \sin(\Delta \phi)}$
    - $\Delta t < t$

- **Right**
  - Dynamics:
    - $\dot{x} = x$
    - $\dot{y} = y$
    - $\dot{\phi}_1 = \phi_1$
    - $\dot{\phi}_2 = \phi_2 - \Delta \phi$
  - Invariant:
    - $t \geq 0$
    - $x^2 + y^2 > a_2$

- **Straight**
  - Dynamics:
    - $\dot{x} = x$
    - $\dot{y} = y$
    - $\dot{\phi}_1 = \phi_1$
    - $\dot{\phi}_2 = \phi_2 - \Delta \phi$
  - Invariant:
    - $x^2 + y^2 \leq a_2$
    - $t = 0$
Computing Unsafe Sets

unsafeCruise  \[ v_1 = 4; v_2 = 5; \lambda = 0 \]
\[ \text{Resolve } \exists t > 0 \land (x - v_1 t + \lambda v_2 t)^2 + (y + \sqrt{1 - \lambda^2} v_2 t)^2 \leq 25 \]
\[ = \left( y < \frac{-20}{\sqrt{11}} \land -\sqrt{11} - \frac{9}{8} \leq x \leq \sqrt{11} - \frac{9}{8} \right) \lor \left( y = \frac{-20}{\sqrt{11}} \land -\sqrt{11} - \frac{9}{8} < x \leq \sqrt{11} - \frac{9}{8} \right) \lor \left( y = \frac{-20}{\sqrt{11}} \land -\sqrt{25 - y^2} < x < \sqrt{25 - y^2} \right) \lor \left( \frac{-20}{\sqrt{11}} < y < \frac{20}{\sqrt{11}} \land -\sqrt{25 - y^2} < x \leq \sqrt{25 - y^2} \right) \]

unsafeLeft  \[ v_1 = 4; v_2 = 5; \lambda = \frac{2}{3} \]
\[ \text{Resolve } \exists t > 0 \land (x - v_1 t + \lambda v_2 t)^2 + (y + \sqrt{1 - \lambda^2} v_2 t)^2 \leq 25 \]
\[ = \left( y < -\frac{4}{\sqrt{11}} \land -\frac{44 + 15\sqrt{11}}{22} - \frac{9}{8} \leq x \leq \frac{44 + 15\sqrt{11}}{22} - \frac{9}{8} \right) \lor \left( y = -\frac{4}{\sqrt{11}} \land -\frac{44 + 15\sqrt{11}}{22} - \frac{9}{8} < x \leq \frac{44 + 15\sqrt{11}}{22} - \frac{9}{8} \right) \lor \left( y = -\frac{4}{\sqrt{11}} \land -\sqrt{25 - y^2} < x < \sqrt{25 - y^2} \right) \lor \left( \frac{-4}{\sqrt{11}} < y < \frac{4}{\sqrt{11}} \land -\sqrt{25 - y^2} < x \leq \sqrt{25 - y^2} \right) \]

unsafeRight  \[ v_1 = 4; v_2 = 5; \lambda = -\frac{2}{3} \]
\[ \text{Resolve } \exists t > 0 \land (x - v_1 t + \lambda v_2 t)^2 + (y + \sqrt{1 - \lambda^2} v_2 t)^2 \leq 25 \]
\[ = \left( y < -7\sqrt{\frac{5}{11}} \land -\frac{50 + 15\sqrt{11}}{22} - \frac{9}{8} \leq x \leq \frac{50 + 15\sqrt{11}}{22} - \frac{9}{8} \right) \lor \left( y = -7\sqrt{\frac{5}{11}} \land -\frac{50 + 15\sqrt{11}}{22} - \frac{9}{8} < x \leq \frac{50 + 15\sqrt{11}}{22} - \frac{9}{8} \right) \lor \left( y = -7\sqrt{\frac{5}{11}} \land -\sqrt{25 - y^2} < x < \sqrt{25 - y^2} \right) \lor \left( -7\sqrt{\frac{5}{11}} < y < 7\sqrt{\frac{5}{11}} \land -\sqrt{25 - y^2} < x \leq \sqrt{25 - y^2} \right) \]

Safe Sets

(a) unsafeCruise (b) unsafeLeft
(c) unsafeRight (d) unsafeCruise \land unsafeLeft \land 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

- $x > 2000$
- $-50 \leq x \leq -40$
- $x \geq 1000$

near

- $-50 \leq x \leq -30$
- $x \geq 0$

past

- $-50 \leq x \leq -30$
- $x > -100$

Gate model

- $\theta = 9$
- $\theta \leq 90$

raising

- $\theta = 0$
- $\theta = 0$

open

- $\theta = 0$
- $\theta = 0$

closed

- $\theta = -9$
- $\theta \geq 0$

lowering

- $\theta = 90$

raise

- $\theta = 90$

raise

- $\theta = 90$

raise

- $\theta = 90$

lower
Controller model

Synchronized transitions
Verifying the controller

System = Train || Gate || Controller

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

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

Research Issues

Modeling Issues
- Well posedness, robustness, zenoess

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: 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 \( \Sigma \)
- A set of observations \( O \)
- The transition relation \( q_i \rightarrow q_j \)
- The observation map \( \langle q_i \rangle = o_0 \)

Initial or final states may be incorporated
The sets \( Q, \Sigma, \) and \( O \) may be infinite
Language of \( T \) is all sequences of observations
A painful example

The parking meter

States $Q = \{0, 1, 2, \ldots, 60\}$

Events $\{\text{tick}, 5\text{p}\}$

Observations $\{\text{exp}, \text{act}\}$

A possible string of observations $\langle \text{exp, act, act, act, act, act, exp, ...} \rangle$

A familiar example

Transition System $T^\Delta$

State set $Q = X = \mathbb{R}^n$

Label set $\Sigma = U = \mathbb{R}^m$

Observation set $O = Y = \mathbb{R}^p$

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

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

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

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

$y_k = Cx_k$
Transition Systems

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

We define the following operators

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

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

We can recursively define

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

Similarly for the other operators. Also

- $\text{Pre}^*(P) = \bigcup_{n \in \mathbb{N}} \text{Pre}^n(P)$
- $\text{Post}^*(P) = \bigcup_{n \in \mathbb{N}} \text{Post}^n(P)$
Basic safety problems

Given transition system $T$ and regions $P, S$ determine

**Forward Reachability**

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

**Backward Reachability**

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

---

Forward reachability algorithm

**Forward Reachability Algorithm**

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

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

Complexity : $O(n_T + m_R)$
Backward reachability algorithm

### Backward Reachability Algorithm

```plaintext
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.

---

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*
- μ-calculus

The basic verification problem

Given transition system T, and temporal logic formula \( \varphi \)

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

---

The basic synthesis problem

Given transition system T, and temporal logic formula \( \varphi \)

Basic synthesis problem

\[ T \parallel C \models \varphi \]

Synthesis in computer science assumes disturbances

Deep relationship between synthesis and game theory
Linear temporal logic (informally)

Express temporal specifications along sequences

<table>
<thead>
<tr>
<th>Informally</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventually p</td>
<td>$\Diamond p$</td>
<td>$pppppppppppppp$</td>
</tr>
<tr>
<td>Always p</td>
<td>$\Box p$</td>
<td>$pppppppppppppp$</td>
</tr>
<tr>
<td>If p then next q</td>
<td>$p \Rightarrow \Diamond q$</td>
<td>$ppppppppppppppq$</td>
</tr>
<tr>
<td>p until q</td>
<td>$p U q$</td>
<td>$ppppppppppppppq$</td>
</tr>
</tbody>
</table>

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 $\varphi_1$ and $\varphi_2$ are formulas then
$\varphi_1 \lor \varphi_2$ and $\neg \varphi_1$

Temporal operators
If $\varphi_1$ and $\varphi_2$ are formulas then
$\varphi_1 U \varphi_2$ and $\Diamond \varphi_1$
Linear temporal logic semantics

The LTL formulas are interpreted over infinite (omega) words

\[ w = p_0 p_1 p_2 p_3 p_4 \ldots \]

\[(w, i) \models p \text{ iff } p_i = p\]
\[(w, i) \models \varphi_1 \lor \varphi_2 \text{ iff } (w, i) \models \varphi_1 \text{ or } (w, i) \models \varphi_2\]
\[(w, i) \models \neg \varphi_1 \text{ iff } (w, i) \not\models \varphi_1\]
\[(w, i) \models \varphi_1 \text{ next } \varphi_2 \text{ iff } (w, i + 1) \models \varphi_1\]
\[(w, i) \models \varphi_1 \text{ until } \varphi_2 \text{ iff } \exists j \geq i (w, j) \models \varphi_2 \text{ and } \forall i \leq k \leq j (w, k) \models \varphi_2\]

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

Linear temporal logic

Syntactic boolean abbreviations

**Conjunction**
\[ \varphi_1 \land \varphi_2 = \neg (\neg \varphi_1 \lor \neg \varphi_2) \]

**Implication**
\[ \varphi_1 \Rightarrow \varphi_2 = \neg \varphi_1 \lor \varphi_2 \]

**Equivalence**
\[ \varphi_1 \Leftrightarrow \varphi_2 = (\varphi_1 \Rightarrow \varphi_2) \land (\varphi_2 \Rightarrow \varphi_1) \]

Syntactic temporal abbreviations

**Eventually**
\[ \Diamond \varphi = T \ U \ \varphi \]

**Always**
\[ \Box \varphi = \neg \Diamond \neg \varphi \]

**In 3 steps**
\[ O_3 \varphi = O \ O \ O \ \varphi \]
**LTL examples**

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

\[ p_1 = \text{inCS, outCS, reqCS} \]
\[ p_2 = \text{inCS, outCS, reqCS} \]

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

\[ \Box \neg (p_1 = \text{inCS} \land p_2 = \text{inCS}) \]

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

\[ p_1 = \text{reqCS} \Rightarrow \Diamond p_1 = \text{inCS} \]

---

**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 | transitions | formula length
Computation tree logic (informally)

Express specifications in computation trees (branching time)

<table>
<thead>
<tr>
<th>Informally</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inevitably next $p$</td>
<td>$\forall \bigcirc p$</td>
<td>![Semantics]</td>
</tr>
<tr>
<td>Possibly always $p$</td>
<td>$\exists \square p$</td>
<td>![Semantics]</td>
</tr>
</tbody>
</table>

Comparing logics

- CTL
- LTL
- CTL*
Dealing with complexity

- Bisimulation
- Simulation
- Language Inclusion

Language Equivalence

Consider two transition systems $T_1$ and $T_2$ over same $\Sigma$ and $O$

$T_1$

- $q_0$ to $q_0$
- $q_0$ to $q_2$
- $q_1$ to $q_1$
- $q_1$ to $q_3$

$T_2$

- $p_0$ to $p_0$
- $p_0$ to $p_2$
- $p_1$ to $p_1$
- $p_1$ to $p_3$

Languages are equivalent $L(T_1) = L(T_2)$
**LTL equivalence**

Consider two transition systems $T_1$ and $T_2$ and an LTL formula

<table>
<thead>
<tr>
<th>Language equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $L(T_1) = L(T_2)$ then $T_1 \models \varphi \Leftrightarrow T_2 \models \varphi$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language inclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>If $L(T_1) \subseteq L(T_2)$ then $T_2 \models \varphi \Rightarrow T_1 \models \varphi$</td>
</tr>
</tbody>
</table>

Language equivalence and inclusion are difficult to check.

---

**Simulation Relations**

Consider two transition systems

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

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 $\overset{q}{\rightarrow} q'$, then $\overset{p}{\rightarrow} p'$ 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**

If $T_1 \leq T_2$ then $L(T_1) \subseteq L(T_2)$

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

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

Two important cases

**Abstraction**

$T_2$

$T_1 \leq T_2$

$T_1$

**Refinement**

$T_1$

$T_1 \leq T_2$

$T_2$
Bisimulation

Consider two transition systems \( T_1 \) and \( T_2 \)

**Bisimulation**

\[ T_1 \equiv T_2 \text{ if } T_1 \leq T_2 \land T_2 \leq T_1 \]

Bisimulation is a symmetric simulation

Strong notion of equivalence for transition systems

**CTL* (and LTL) equivalence**

If \( T_1 \equiv T_2 \) then \( T_1 \models \varphi \iff T_2 \models \varphi \)

If \( T_1 \equiv T_2 \) then \( L(T_1) = L(T_2) \)

Special quotients

Abstraction

\[ T/\approx \]

\[ T \leq T/\approx \]

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

\[ T/ \approx = (Q/ \approx, \Sigma, \rightarrow, O, \langle \cdot \rangle_\approx) \]

naturally using

1. **Observation Map**
   
   \( \langle P \rangle_\approx = o \) iff there exists \( p \in P \) with \( \langle p \rangle = o \)

2. **Transition Relation**

   \( P \rightarrow_\approx P' \) iff there exists \( p \in P, p' \in P' \) with \( p \rightarrow p' \)

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 $T / \approx$?

```
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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
$O(m \cdot 2^n)$
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 \( H = (V, \mathbb{R}^n, X_0, F, Inv, R) \) consists of

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

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

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

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

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

An example

- Rod1
  \[ T = 0.1, T = 56 \]
  \[ \dot{y}_1 = 1, \dot{y}_2 = 1 \]
  \[ T = 510 \]
- NoRod
  \[ T = 0.1, T = 50 \]
  \[ \dot{y}_1 = 1, \dot{y}_2 = 1 \]
  \[ T = 550 \]
- Rod2
  \[ T = 0.1, T = 60 \]
  \[ \dot{y}_1 = 1, \dot{y}_2 = 1 \]
  \[ T = 510 \]
- Shutdown
  \[ T = 0.1, T = 50 \]
  \[ \dot{y}_1 = 1, \dot{y}_2 = 1 \]
  \[ true \]
## Transitions of Hybrid Systems

Hybrid systems can be embedded into transition systems

\[ H = (V, \mathbb{R}^n, X_0, F, \text{Inv}, R) \quad \rightarrow \quad T_H = (Q, Q_0, \Sigma, \rightarrow, O, \langle \cdot, \cdot \rangle) \]

- \( Q = V \times \mathbb{R}^n \)
- \( Q_0 = X_0 \)
- \( \Sigma = E \cup \{ \tau \} \)
- \( \rightarrow \subseteq Q \times \Sigma \times Q \)

### Observation set and map depend on desired properties

**Discrete transitions**

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

**Continuous (time-abstract) transitions**

\( (l_1, x_1) \triangleright (l_2, x_2) \) iff \( l_1 = l_2 \) and \( \exists \delta \geq 0 : [0, \delta] \rightarrow \mathbb{R}^n x(\cdot) : [0, \delta] \rightarrow \mathbb{R}^n \)

\[ x(0) = x_1, x(\delta) = x_2, \text{ and } \forall t \in [0, \delta] x(t) \in \text{Inv}(l_1) \]

### Rectangular hybrid automata

**Rectangular sets**:

\[ \bigwedge_i x_i \sim c_i \sim \in \{<, \leq, =, \geq, >\}, c_i \in Q \]

- far: \( -50 \leq x \leq -40 \)
- near: \( -50 \leq x \leq -30 \)
- past: \( -50 \leq x \leq -30 \)

**Observation set and map** depend on desired properties

**Rectangular hybrid automata are hybrid systems where**

\[ \text{Init}(l), \text{Inv}(l), F(l, x), \text{Guard}(e), \text{Reset}(e, x)_{\sim} \]

are rectangular sets
Multi-rate automata are rectangular hybrid automata where $\text{Init}(l), F(l, x), \text{Reset}(e, x)$ are singleton sets.

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.

\[
\begin{align*}
\text{far} & : -50 \leq x \leq -40 \\
& \quad \text{x} \geq 1000 \\
\text{near} & : -50 \leq x \leq -30 \\
& \quad \text{x} \geq 0 \\
\text{past} & : -50 \leq x \leq -30 \\
& \quad \text{x} \geq -100
\end{align*}
\]

**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.
All timed automata admit a finite bisimulation

Hence CTL* model checking is decidable for 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

but...

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

LTL model checking of rectangular automata is decidable.
More complicated dynamics?

Bisimulation algorithm never terminates!!

Sets

\[ P_1 = \{ (x,0) | 0 \leq x \leq 4 \} \]
\[ P_2 = \{ (x,0) | -4 \leq x < 0 \} \]
\[ P_3 = \mathbb{R}^2 \setminus (P_1 \cup P_2) \]

Dynamics

\[ \dot{x}_1 = 0.2x_1 + x_2 \]
\[ \dot{x}_2 = -x_1 + 0.2x_2 \]

Basic problems

Finite bisimulations of continuous dynamical systems

Given a vector field \( F(x) \) and a finite partition of \( \mathbb{R}^n \)

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

\((\mathbb{R}, <, +, - , 0, 1)\)

\begin{array}{c}
\text{Universe} \\
\text{Relation} \\
\text{Functions} \\
\text{Constants}
\end{array}

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 \(\varphi_1, \varphi_2\) are formulas, then
\(\varphi_1 \lor \varphi_2, \neg \varphi_1, \forall x. \varphi_1, \exists x. \varphi_1\)
First-order logic

Useful languages

\[(\mathbb{R}, <, +, -, 0, 1) \quad \forall x \forall y(x + 2y \geq 0)\]

\[(\mathbb{R}, <, +, -, \times, 0, 1) \quad \exists x. ax^2 + bx + c = 0\]

\[(\mathbb{R}, <, +, -, \times, e^x, 0, 1) \quad \exists t. (t \geq 0) \land (y = e^t x)\]

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

<table>
<thead>
<tr>
<th>Theory</th>
<th>Decidable ?</th>
<th>Quant. Elim. ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\mathbb{R}, &lt;, +, -, 0, 1))</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>((\mathbb{R}, &lt;, +, -, \times, 0, 1))</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>((\mathbb{R}, &lt;, +, -, \times, e^x, 0, 1))</td>
<td>?</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Tarski's result**: Every formula in \((\mathbb{R}, <, +, -, \times, 0, 1)\) can be decided

1. Eliminate quantified variables
2. Quantifier free formulas can be decided
**O-Minimal Theories**

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

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: \( Y = \{(x) \in \mathbb{R} \mid p(x) \geq 0\} \) for polynomial \( p(x) \).

Recent o-minimal theories:

- \((\mathbb{R}, <, +, -, 0, 1)\)
- \((\mathbb{R}, <, +, -, \times, 0, 1)\)
- \((\mathbb{R}, <, +, -, \times, e^x, 0, 1)\)
- \((\mathbb{R}, <, +, -, \times, e^x, 0, 1)\)

Related to Hilbert's 16th problem.

---

**Basic answers**

**Finite bisimulations of continuous dynamical systems**

Consider a vector field \( X \) and a finite partition of \( \mathbb{R}^n \) 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

Consider continuous systems where
- Finite partition is polyhedral (semi-linear)
- Vector fields have linear flows (timed, multi-rate)
Then a finite bisimulation exists.

Consider continuous systems where
- Finite partition is semialgebraic
- Vector fields have polynomial flows
Then a finite bisimulation exists.

Consider continuous systems where
- Finite partition is semi-algebraic
- Vector fields are linear with real eigenvalues
Then a finite bisimulation exists.

Consider continuous systems where
- Finite partition is sub-analytic
- Vector fields are linear with purely imaginary eigenvalues
Then a finite bisimulation exists.
Corollaries

Consider continuous systems where

- Finite partition is semi-algebraic
- Vector fields are linear with real or imaginary 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 \iff <q> = <p>\}
while \exists P, P' \in Q/\sim \ such that \emptyset \not\subseteq P \cap Pre(P') \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
```

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

Use \((\mathbb{R}, <, +, -, \times, f, e, 0, 1)\)
Computing reachable sets

Consider a linear system

\[ \frac{dx}{dt} = Ax \quad A \in \mathbb{Q}^{n \times n} \]

and a semi-algebraic set \( Y \). If

\[ Y = \{ y \in \mathbb{R}^n | p(y) \} \]

Then

\[ \text{Pre}(Y) = \{ x \in \mathbb{R}^n | \exists y \exists t. p(y) \land t \geq 0 \land x = e^{-tA}y \} \]

Problem?

Nilpotent Linear Systems

Nilpotent matrices: \( \exists n \geq 0 \ A^n = 0 \)

Then flow of linear system is polynomial

\[ e^{-tA} = \sum_{k=0}^{n-1} (-1)^{k^n} A^k \]

Therefore \( \text{Pre}(Y) \) completely definable in \( (\mathbb{R},<,+,\times,0,1) \)

\[ \text{Pre}(Y) = \{ x \in \mathbb{R}^n | \exists y \exists t. p(y) \land t \geq 0 \land x = \sum_{k=0}^{n-1} (-1)^{k^n} A^k y \} \]
Diagonalizable, rational eigenvalues

Example system: \( \dot{x} = 2x \)

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

\[
Pre(Y) = \{ x \in \mathbb{R} \mid \exists y \exists t. y = 5 \land t \geq 0 \land x = e^{-2t}y \}
\]

Let \( s = e^{-t} \), then

\[
Pre(Y) = \{ x \in \mathbb{R} \mid \exists y \exists t. y = 5 \land 1 \geq s \geq 0 \land x = s^2y \}
\]

\[
Pre(Y) = \{ x \in \mathbb{R} \mid 0 < x \leq 5 \}
\]

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

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

1. Rescale rational eigenvalues to integer eigenvalues.
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)
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 \( \text{Pre}(Y) \) is also semi-algebraic (and computable)

Composing all computability results together results in...

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 \( \text{CTL}^* \) formula where atomic propositions denote semi-algebraic sets. Then \( \text{CTL}^* \) model checking is decidable.
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
\[ A \parallel X \equiv B \]
or a more relaxed version...
\[ A \parallel X \leq B \]
Equations can be interpreted over various model types
Various semantics of composition and equivalence
Discrete semantics

The main (controller) synthesis equation

\[ A \parallel X \simeq B \]

or a more relaxed version...

\[ A \parallel X \leq B \]

Models : Finite state automata
Composition : 
Equivalence :
Order : 

\[ L(A \parallel B) = L(A) \cap L(X) \]
\[ A \equiv B \iff L(A) = L(X) \]
\[ A \leq B \iff L(A) \subseteq L(X) \]

Continuous semantics

The main (controller) synthesis equation

\[ A \parallel X \simeq B \]

or a more relaxed version...

\[ A \parallel X \leq B \]

Models : Control systems
Composition : Feedback composition
Equivalence : Asymptotic equivalence
Order : Not much...
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(S₁) = L(S₂)

Nondeterminism!

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 \( q_i \rightarrow q_i \)

\[ R \subseteq R \]

\[ q_2 \rightarrow q_2 \]

If a simulation relation exists, then \( S_1 \leq S_2 \)
Game theoretic semantics

Simulation is a matching game between the systems

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

Relationships

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Bisimulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall \text{CTL}$</td>
<td>$\text{CTL}^*$</td>
</tr>
<tr>
<td><strong>Language Inclusion</strong></td>
<td><strong>Language Equivalence</strong></td>
</tr>
</tbody>
</table>
| Reachability | 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
**Bi-simulations of control systems**

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

$y_1(t) = C_1 x_1(t)$

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

$u_1(t)$

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

$y_2(t) = C_2 x_2(t)$

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

$u_2(t)$

---

**Non-deterministic dynamics**

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

$y_1(t) = C_1 x_1(t)$

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

$u_1(t)$

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

$y_2(t) = C_2 x_2(t)$

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

$u_2(t)$

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

$\begin{align*}
  x_1(0) &\rightarrow x_1(t) \\
  R &\rightarrow C_1 x_1(t) = C_2 x_2(t)
\end{align*}$

$R$ is a bi-simulation if converse is true as well.
A linear relation $(x, Hx)$ is a simulation relation iff for all $\forall d_1 \exists d_2$

\[
H(A_1x_1 + E_1d_1) = A_2Hx_1 + E_2d_2
\]

$C_1 = C_2H$

$S_2$ simulates or is $H$-related to $S_1$

---

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

\[
HA_1 = A_2H
\]

$C_1 = C_2H$

Restrictive!
Advantage of non-determinism

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

\[
\begin{align*}
\dot{x}_1(t) &= A_1 x_1(t) + E_1 d_1(t) \\
y_1(t) &= C_1 x_1(t) \\
\end{align*}
\]

\[
\begin{align*}
A_2 &= HA_1 H^+ \\
E_2 &= [HE_1 \quad HA_1 \text{Ker}(H)] \\
C_2 &= C_1 H^+ \quad \text{if Ker}(H) \subseteq \text{Ker}(C_1)
\end{align*}
\]

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

\[ x_1(t) = x_2(t) \]
\[ x_2(t) = d_2(t) \]
\[ y_1(t) = x_1(t) \]

\[ z_1(t) = x_1 \]
\[ \dot{z}_1(t) = d_1(t) \]
\[ y_1(t) = z_1(t) \]

\[ L(S_1) = L(S_2) \]
\[ S_1 \leq S_2 \]
\[ S_2 \not\leq S_1 \]
\[ S_1 \not\leq S_2 \]

When is \((x, Hx)\) a bisimulation relation?

Bisimilar linear systems*

\[
\begin{align*}
  x_1(t) &= A_1 x_1(t) + E_1 d_1(t) \\
  y_1(t) &= C_1 x_1(t) \\
  x_2(t) &= A_2 x_2(t) + E_2 d_2(t) \\
  y_2(t) &= C_2 x_2(t)
\end{align*}
\]

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

\[ A_1 \text{Ker}(H) \subseteq \text{Ker}(H) + R(E_1) \]

*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 \( x_2 = H x_1 \) which abstracts as much as possible. Thus \( \text{Ker}(H) \) must be maximal but also must...

Preserve observations
\[
\text{Ker}(H) \supseteq \text{Ker}(C_1)
\]

Preserve transitions
\[
A \cdot \text{Ker}(H) \subseteq \text{Ker}(H) + R(E_1)
\]

This lead to the well known algorithm...

---

**Coarsest Bisimulation Algorithm**

*Maximal controlled invariant subspace computation*

\[
V_0 = \text{Ker}(C_1)
\]

\[
V_{k+1} = V_{k-1} \cap A_{k-1}^{-1}(V_{k-1} + R(E_1))
\]

Then \( V^* = V_n \) is the maximal desired subspace

Once \( V^* \) is computed, then pick map \( x_2 = H x_1 \) such that \( \text{Ker}(H) = V^* \)

and construct the \( H \)-related system.
Similar relationships

- **Simulation**
- **Bisimulation**
- **Language Inclusion**
- **Language Equivalence**

If $S_1 \leq S_2$ then $L(S_1) \subseteq L(S_2)$
If $S_1 \equiv S_2$ then $L(S_1) = L(S_2)$
If $S_1 \leq S_2$ then $H(\text{Reach}(S_1, X)) \subseteq \text{Reach}(S_2, H(X))$

Extensions

**Bi-simulations of nonlinear systems**

**Unifying discrete and continuous notions**

**Extensions to hybrid systems**