Synthesis for Cyber-Physical Systems

Paulo Tabuada
Cyber-Physical Systems Laboratory
Department of Electrical Engineering
University of California at Los Angeles
Organization

- **Lecture 1**: Introduction to Cyber-Physical Systems, models, and relationships
- **Lecture 2**: Synthesis using exact finite-state abstractions
- **Lecture 3**: Synthesis using approximate finite-state abstractions
- **Lecture 4**: Playtime with Pessoa (Matthias Rungger)
What are Cyber-Physical Systems?

- Different people interpret the expression Cyber-Physical Systems (CPSs) differently;
Different people interpret the expression **Cyber-Physical Systems (CPSs)** differently;

In my lectures, CPSs satisfy the following 2 properties:

1. The cyber components receive information from the physical world, process it, and feed it back so as to influence the physical components;
2. The interaction between the cyber and physical components is so tight that these components cannot be studied in isolation.
Introduction to Cyber-Physical Systems

What are Cyber-Physical Systems?

- Different people interpret the expression **Cyber-Physical Systems (CPSs)** differently;
- In my lectures, CPSs satisfy the following 2 properties:

  1. The cyber components receive information from the physical world, process it, and feed it back so as to influence the physical components;
  2. The interaction between the cyber and physical components is so tight that these components cannot be studied in isolation.

- **Cyber** components have traditionally been studied in **Computer Science** while **physical** components have traditionally been studied in **Control Theory**.
- In these lectures we will use results and techniques from both these areas.
The timing of the decisions made by the software is critical to the success of the mission.

The “dynamics” of the software needs to be analyzed in conjunction with the dynamics of the physical components.

http://www.jpl.nasa.gov/video/
A combination of time-driven and event-driven precise control decisions is required to avoid paper jams and maintain high throughput.
Control theory provides analysis and design techniques for simple objectives.
Introduction to Cyber-Physical Systems

Some challenges

- Control theory provides analysis and design techniques for simple objectives.
- But current applications require sophisticated functionalities, e.g., adaptive cruise control (International Standard ISO 15622).

Figure 3 — ACC states and transitions

 ACC on
 ACC off
 ACC stand-by
 ACC off*
 ACC off *
 ACC active
 ACC speed control
 ACC time gap control

\[ c = \frac{d}{v} \]

* Manually and/or automatically after self test. Manual transition describes a switch to enable/disable ACC function. Automatic switch off can be forced by failure reaction.

\( = \text{system state} \)
Control theory provides analysis and design techniques for simple objectives.

But current applications require sophisticated functionalities, e.g., adaptive cruise control (International Standard ISO 15622).

Unfortunately, it is well known that switching between correct control software may lead to incorrect behavior.
Some challenges

- Control theory provides analysis and design techniques for simple objectives.
- But current applications require sophisticated functionalities, e.g., adaptive cruise control (International Standard ISO 15622).
- Unfortunately, it is well known that switching between correct control software may lead to incorrect behavior.
Control theory provides analysis and design techniques for simple objectives.

But current applications require sophisticated functionalities, e.g., adaptive cruise control (International Standard ISO 15622).

Unfortunately, it is well known that switching between correct control software may lead to incorrect behavior.

How to synthesize code enforcing high-level specifications on CPSs?
Our approach will be based on finite-state abstractions of the physical world so as to leverage reactive synthesis techniques (Prof. Vardi’s tutorial).
Synthesis for Cyber-Physical Systems

Key ingredients

\[ \frac{d}{dt} x = f(t, u) \]

Physical System
Synthesis for Cyber-Physical Systems

Key ingredients

\[ \frac{d\hat{x}}{dt} = f(t, x) \]

Physical System

Abstraction
Synthesis for Cyber-Physical Systems

Key ingredients

Abstraction

Software + Hardware

\[ \frac{d}{dt} \gamma = f(t, \omega) \]

Physical System
Synthesis for Cyber-Physical Systems

Key ingredients

- Abstraction
- Software + Hardware

Physical System: \[ \frac{d}{dt} x = f(t, x) \]
Synthesis for Cyber-Physical Systems

Key ingredients

- Abstraction
- Software + Hardware
- Controller

\[
\frac{d x}{dt} = f(t, u)
\]

Physical System
Synthesis for Cyber-Physical Systems

Key ingredients

\[ \frac{d}{dt} x = f(t, u) \]
Physical System

\[ q(k+1) = S(q(k), \delta(k)) \]
\[ \delta^*(k) = K(q^*(k), q(k)) \]
Hybrid Controller
Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:

\[
\frac{d}{dt} \xi = A\xi + B\nu. \tag{1}
\]
Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:

\[
\frac{d}{dt} \xi = A\xi + B\nu. \quad (1)
\]

There are 2 concepts that we need in order to make sense of (1):

- States \( x \in \mathbb{R}^n \) and state trajectories \( \xi : \mathbb{R}_0^+ \to \mathbb{R}^n \);
- Inputs \( u \in \mathbb{R}^m \) and input trajectories \( \nu : \mathbb{R}_0^+ \to \mathbb{R}^m \);
Models for the physical world
Differential and difference equations

- Differential equations is the most common model for the physical world. In these lectures we will consider only **linear** differential equations:

\[
\frac{d}{dt} \xi = A\xi + B\nu. \tag{1}
\]

- There are 2 concepts that we need in order to make sense of (1):
  - States \( x \in \mathbb{R}^n \) and state trajectories \( \xi : \mathbb{R}_0^+ \rightarrow \mathbb{R}^n \);
  - Inputs \( u \in \mathbb{R}^m \) and input trajectories \( \nu : \mathbb{R}_0^+ \rightarrow \mathbb{R}^m \);

- For each initial state \( x \in \mathbb{R}^n \) and input trajectory \( \nu : \mathbb{R}_0^+ \rightarrow \mathbb{R}^m \) there exists a unique state trajectory (solution, execution, run, trace, ...) \( \xi_{x,\nu} : \mathbb{R}_0^+ \rightarrow \mathbb{R}^n \) satisfying:
  - \( \xi_{x,\nu}(0) = x \);
  - the time derivative of \( \xi_{x,\nu} \) at time \( t \in \mathbb{R}_0^+ \) is equal to \( A\xi_{x,\nu}(t) + B\nu(t) \).
Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:

\[
\frac{d}{dt} \xi = A\xi + B\nu.
\]  

We can bring (1) closer to the models used in computer science by moving from continuous time to discrete time.
Models for the physical world
Differential and difference equations

- Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:

\[
\frac{d}{dt} \xi = A\xi + B\nu. \tag{1}
\]

- We can bring (1) closer to the models used in computer science by moving from continuous time to discrete time.
  1. Choose a sampling period \( \tau \in \mathbb{R}^+ \);
Models for the physical world
Differential and difference equations

- Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:
  \[
  \frac{d}{dt} \xi = A\xi + B\nu.
  \]  
  (1)

- We can bring (1) closer to the models used in computer science by moving from continuous time to discrete time.
  1. Choose a sampling period \( \tau \in \mathbb{R}^+ \);

  2. Keep the input constant during the intervals \([k\tau, (k+1)\tau] \), \( k \in \mathbb{N}_0 \).
Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:

\[
\frac{d}{dt} \xi = A\xi + B\nu. \tag{1}
\]

We can bring (1) closer to the models used in computer science by moving from continuous time to discrete time.

1. Choose a sampling period \( \tau \in \mathbb{R}^+ \);

2. Keep the input constant during the intervals \([k\tau, (k+1)\tau[\), \(k \in \mathbb{N}_0\).

3. Compute new matrices \( A' = e^{A\tau} \) and \( B' = \int_0^\tau e^{A(t-s)}B\,ds \) so that the solution \( \xi'_{x,\nu'} : \mathbb{N}_0 \rightarrow \mathbb{R}^n \) of the difference equation:

\[
\xi'(k + 1) = A'\xi'(k) + B'\nu'(k)
\]

satisfies \( \xi'_{x,\nu'}(k) = \xi_{x,\nu'}(k\tau) \).
Models for the cyber and the physical world

Systems

- Difference equations as well as finite-state automata can be modeled as systems:

**Definition (System)**

A system \( S \) is a sextuple \((X, X_0, U, \longrightarrow, Y, H)\) consisting of:

- a set of states \( X \);
- a set of initial states \( X_0 \subseteq X \);
- a set of inputs \( U \);
- a transition relation \( \longrightarrow \subseteq X \times U \times X \);
- a set of outputs \( Y \);
- an output map \( H : X \rightarrow Y \).
Models for the cyber and the physical world
A useful graphical description

\[ X = \{ x_0, x_1, x_2, x_3 \} \]
Models for the cyber and the physical world

A useful graphical description

\[ X = \{ x_0, x_1, x_2, x_3 \} \]

\[ X_0 = \{ x_0, x_2 \} \]
Models for the cyber and the physical world
A useful graphical description

\[ X = \{ x_0, x_1, x_2, x_3 \} \]
\[ X_0 = \{ x_0, x_2 \} \]
\[ U = \{ u_0, u_1 \} \]
Models for the cyber and the physical world
A useful graphical description

\[ X = \{ x_0, x_1, x_2, x_3 \} \]
\[ X_0 = \{ x_0, x_2 \} \]
\[ U = \{ u_0, u_1 \} \]
\[ \mathbb{Y} = \{ y_0, y_1, y_2 \} \]
\[ H(x_0) = y_0, \quad H(x_1) = y_0, \quad H(x_2) = y_1, \quad H(x_3) = y_2. \]
Models for the cyber and the physical world
A useful graphical description

\[
X = \{x_0, x_1, x_2, x_3\}
\]
\[
X_0 = \{x_0, x_2\}
\]
\[
U = \{u_0, u_1\}
\]
\[
Y = \{y_0, y_1, y_2\}
\]

\[
H(x_0) = y_0, \quad H(x_1) = y_0, \quad H(x_2) = y_1, \quad H(x_3) = y_2.
\]
Models for the cyber and the physical world
A useful graphical description

\[ X = \{ x_0, x_1, x_2, x_3 \} \]
\[ X_0 = \{ x_0, x_2 \} \]
\[ U = \{ u_0, u_1 \} \]
\[ Y = \{ y_0, y_1, y_2 \} \]
\[ H(x_0) = y_0, \quad H(x_1) = y_0, \quad H(x_2) = y_1 \]
\[ H(x_3) = y_2. \]
Suppose that we want to compute the average of a stream of numbers but we do not know a priori the length of the stream.
Suppose that we want to compute the average of a stream of numbers but we do not know a priori the length of the stream.

\[
x := 0;
n := 0;
\text{While}(true)\{
y := \text{read(input)};
x := x \frac{n}{n+1} + y \frac{1}{n+1};
n := n + 1;
\}
\]

The variable \( y \) contains the latest received number and the variable \( x \) contains the average of the numbers that have been received so far.
Suppose that we want to compute the average of a stream of numbers but we do not know a priori the length of the stream.

\[
x := 0;
\]
\[
n := 0;
\]
\[
\text{While(true)}\{
\]
\[
y := \text{read(input)};
\]
\[
x := x \frac{n}{n+1} + y \frac{1}{n+1};
\]
\[
n := n + 1;
\}
\]

The variable \( y \) contains the latest received number and the variable \( x \) contains the average of the numbers that have been received so far.

Assume now that we are interested in knowing if \( x \) is smaller, equal, or greater than 1 when \( y \) is restricted to assume values in the set \{1, 2\}. 
Suppose that we want to compute the average of a stream of numbers but we do not know a priori the length of the stream.

\[ x := 0; \]
\[ n := 0; \]
\[ \textbf{While}(\text{true})\{ \]
\[ \quad y := \text{read}(\text{input}); \]
\[ \quad x := x \frac{n}{n+1} + y \frac{1}{n+1}; \]
\[ \quad n := n + 1; \} \]

- The variable \( y \) contains the latest received number and the variable \( x \) contains the average of the numbers that have been received so far.
- Assume now that we are interested in knowing if \( x \) is smaller, equal, or greater than 1 when \( y \) is restricted to assume values in the set \( \{1, 2\} \).
Consider the controlled model for the national income inspired by Paul Samuelson’s 1939 model\(^1\):

\[
\begin{align*}
    c(k+1) &= \alpha (c(k) + i(k) + g(k)) \\
    i(k+1) &= \beta \alpha (c(k) + i(k) + g(k)) - \beta c(k) \\
    g(k+1) &= d(k).
\end{align*}
\]

where the national income is the sum \(c + i + g\) of three kinds of expenditures: consumption (c), investment (i), and government expenditures (g). The parameters \(\alpha, \beta \in \mathbb{R}\) are identified from data.

---

Models for the cyber and the physical world

An economy example

Consider the controlled model for the national income inspired by Paul Samuelson’s 1939 model\textsuperscript{1}:

\[
\begin{align*}
    c(k+1) &= \alpha(c(k) + i(k) + g(k)) \\
    i(k+1) &= \beta\alpha(c(k) + i(k) + g(k)) - \beta c(k) \\
    g(k+1) &= d(k).
\end{align*}
\]

where the national income is the sum $c + i + g$ of three kinds of expenditures: consumption ($c$), investment ($i$), and government expenditures ($g$). The parameters $\alpha, \beta \in \mathbb{R}$ are identified from data.

We can describe this model by the following system:

- $X = \mathbb{R}^3, X_0 = X, U = \mathbb{R}_0^+$;
- $(c(k), i(k), g(k)) \xrightarrow{d(k)} (c(k+1), i(k+1), g(k+1))$ if equations (2) are satisfied;
- $Y = \mathbb{R}, H = c + i + g$.

\textsuperscript{1} Interactions between the multiplier analysis and the principle of acceleration. The Review of Economic Statistics, 21(2):75-78, 1939.
We now have a class of models (systems) that can describe both physical systems as well as its finite-state abstractions.
We now have a class of models (systems) that can describe both physical systems as well as its finite-state abstractions.

But what is the relation between the properties enforced by the controller and the hybrid controller?
Relating systems and their properties

Simulation relations

- Properties expressed in LTL can be transferred between systems related by simulation relations.

**Definition (Simulation Relation)**

Let \( S_a = (X_a, X_{a0}, U_a, a \rightarrow, Y_a, H_a) \) and \( S_b = (X_b, X_{b0}, U_b, b \rightarrow, Y_b, H_b) \) be systems with \( Y_a = Y_b \). A relation \( R \subseteq X_a \times X_b \) is a simulation relation from \( S_a \) to \( S_b \) if the following three conditions are satisfied:

1. for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. for every \( (x_a, x_b) \in R \) we have \( H_a(x_a) = H_b(x_b) \);
3. for every \( (x_a, x_b) \in R \) we have that:
   \[ x_a \xrightarrow{u_a} x'_a \text{ in } S_a \implies \text{the existence of } x_b \xrightarrow{u_b} x'_b \text{ in } S_b \text{ satisfying } (x'_a, x'_b) \in R. \]

We say that \( S_a \) is simulated by \( S_b \) or that \( S_b \) simulates \( S_a \), denoted by \( S_a \preceq_S S_b \), if there exists a simulation relation from \( S_a \) to \( S_b \).
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]

1. for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \((x_{a0}, x_{b0}) \in R\);
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]

1. for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. for every \( (x_a, x_b) \in R \) we have \( H_a(x_a) = H_b(x_b) \);
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]

1. for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. for every \( (x_{a}, x_{b}) \in R \) we have \( H_{a}(x_{a}) = H_{b}(x_{b}) \);
3. \( x_{a} \xrightarrow{u_{a}} x'_{a} \) in \( S_{a} \) implies the existence of \( x_{b} \xrightarrow{u_{b}} x'_{b} \) in \( S_{b} \) satisfying \( (x'_{a}, x'_{b}) \in R \).
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]

1. For every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. For every \( (x_{a}, x_{b}) \in R \) we have \( H_a(x_{a}) = H_b(x_{b}) \);
3. \( x_{a} \xrightarrow{u_{a}} x'_{a} \) in \( S_a \) implies the existence of \( x_{b} \xrightarrow{u_{b}} x'_{b} \) in \( S_b \) satisfying \( (x'_{a}, x'_{b}) \in R \).
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]

1. For every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. For every \( (x_a, x_b) \in R \) we have \( H_a(x_a) = H_b(x_b) \);
3. \( x_a \xrightarrow{u_a} x'_a \) in \( S_a \) implies the existence of \( x_b \xrightarrow{u_b} x'_b \) in \( S_b \) satisfying \( (x'_a, x'_b) \in R \).
Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:

\[
R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}.
\]

1. For every \(x_{a0} \in X_{a0}\), there exists \(x_{b0} \in X_{b0}\) with \((x_{a0}, x_{b0}) \in R\);
2. For every \((x_a, x_b) \in R\) we have \(H_a(x_a) = H_b(x_b)\);
3. \(x_a \xrightarrow{u_a} x'_a\) in \(S_a\) implies the existence of \(x_b \xrightarrow{u_b} x'_b\) in \(S_b\) satisfying \((x'_a, x'_b) \in R\).
Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:
  \[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\} \].

1. for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. for every \( (x_a, x_b) \in R \) we have \( H_a(x_a) = H_b(x_b) \);
3. \( x_a \xrightarrow{u_a}{u_a} x'_a \) in \( S_a \) implies the existence of \( x_b \xrightarrow{u_b}{u_b} x'_b \) in \( S_b \) satisfying \( (x'_a, x'_b) \in R \).
Consider the following two systems and the relation:

\[ R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}. \]

1. for every \( x_{a0} \in X_{a0} \), there exists \( x_{b0} \in X_{b0} \) with \( (x_{a0}, x_{b0}) \in R \);
2. for every \( (x_a, x_b) \in R \) we have \( H_a(x_a) = H_b(x_b) \);
3. \( x_a \xrightarrow{u_a}{\ a} x'_a \) in \( S_a \) implies the existence of \( x_b \xrightarrow{u_b}{\ b} x'_b \) in \( S_b \) satisfying \( (x'_a, x'_b) \in R \).
Existence of a simulation relation from $S_a$ to $S_b$ can be used to transfer LTL properties from $S_b$ to $S_a$.

**Proposition**

*For any LTL formula $\varphi$ we have:*

$$S_a \preceq_S S_b \implies (S_b \models \varphi \implies S_a \models \varphi).$$
Existence of a simulation relation from $S_a$ to $S_b$ can be used to transfer LTL properties from $S_b$ to $S_a$.

**Proposition**

For any LTL formula $\varphi$ we have:

$$S_a \preceq_S S_b \implies (S_b \models \varphi \implies S_a \models \varphi).$$

But we really want to transfer properties enforced by controllers. In other words, we want to transfer, not properties, but the ability to enforce properties.
Existence of a simulation relation from $S_a$ to $S_b$ can be used to transfer LTL properties from $S_b$ to $S_a$.

**Proposition**

*For any LTL formula $\varphi$ we have:*

$$S_a \preceq_S S_b \implies (S_b \models \varphi \implies S_a \models \varphi).$$

But we really want to transfer properties enforced by controllers. In other words, we want to transfer, not properties, but the ability to enforce properties.

Enforcing properties, i.e., controlling, requires a battle against nondeterminism.
A system is **deterministic** if given a state \( x \in X \) and an input \( u \in U \) there exists at most one state \( x' \in X \) for which \( x \xrightarrow{u} x' \).
A system is **deterministic** if given a state $x \in X$ and an input $u \in U$ there exists at most one state $x' \in X$ for which $x \xrightarrow{u} x'$.

In general, we have to work with nondeterministic systems since models are not accurate:

- The differential equation models for physical systems are always an approximate description of reality.
- Sensors and actuators are never perfect and are subject to noise.
- Models for software only describe a partial view of what happens inside a computer.
Relating systems and their properties

Nondeterminism

- A system is **deterministic** if given a state $x \in X$ and an input $u \in U$ there exists at most one state $x' \in X$ for which $x \xrightarrow{u} x'$.

- In general, we have to work with nondeterministic systems since models are not accurate:
  - The differential equation models for physical systems are always an approximate description of reality.
  - Sensors and actuators are never perfect and are subject to noise.
  - Models for software only describe a partial view of what happens inside a computer.

- Given a state $x \in X$ and an input $u \in U$ we denote by $\text{Post}_u(x)$ the set of all the states that can be reached from $x$ under input $u$, formally:
  
  $$\text{Post}_u(x) = \{ x' \in X \mid x \xrightarrow{u} x' \}.$$
The enforcement of properties expressed in LTL can be transferred between systems related by alternating simulations.

**Definition (Alternating simulation relation)**

Let $S_a = (X_a, X_{a0}, U_a, \rightarrow_a, Y_a)$ and $S_b = (X_b, X_{b0}, U_b, \rightarrow_b, Y_b)$ be systems with $Y_a = Y_b$. A relation $R \subseteq X_a \times X_b$ is an alternating simulation relation from $S_a$ to $S_b$ if the following three conditions are satisfied:

1. for every $x_{a0} \in X_{a0}$ there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
3. for every $(x_a, x_b) \in R$ and for every $u_a \in U_a$ there exists $u_b \in U_b$ such that for every $x'_b \in \text{Post}_{u_b}(x_b)$ there exists $x'_a \in \text{Post}_{u_a}(x_a)$ satisfying $(x'_a, x'_b) \in R$.

We say that $S_a$ is alternatingly simulated by $S_b$ or that $S_b$ alternatingly simulates $S_a$, denoted by $S_a \preceq_{AS} S_b$, if there exists an alternating simulation relation from $S_a$ to $S_b$. 
How is simulation related to alternating simulation?
Relating systems and their properties

Alternating simulation relations

- How is simulation related to alternating simulation?

The relation $R = \{(x_{a0}, x_{b0}), (x_{a1}, x_{b1}), (x_{a2}, x_{b2})\}$ is a simulation relation from $S_a$ to $S_b$ but not an alternating simulation. Although $x_{b3} \in \text{Post}_a(x_{b0})$ in $S_b$ no state in $S_a$ is related to $x_{b3}$.
How is simulation related to alternating simulation?

Conversely, the relation $R' = \{ (x_{a0}, x_{b0}), (x_{a1}, x_{b1}), (x_{a1}, x_{b2}), (x_{a1}, x_{b3}) \}$ is an alternating simulation relation from $S_a$ to $S_b$ but not a simulation relation from $S_a$ to $S_b$. The transition $x_{a0} \xrightarrow{a} x_{a2}$ in $S_a$ cannot be matched by $S_b$. 

Paulo Tabuada (CyPhyLab - UCLA) Synthesis for Cyber-Physical Systems ExCAPE Summer School’13
How is simulation related to alternating simulation?

Alternating simulation degenerates into simulation in the very special case of deterministic systems.

Determinism implies $|\text{Post}_{u_b}(x_b)| \leq 1$ and $|\text{Post}_{u_a}(x_a)| \leq 1$. 

Can we use alternating simulations to relate properties enforced by controllers?
How is simulation related to alternating simulation?

Alternating simulation degenerates into simulation in the very special case of deterministic systems.

Determinism implies $|\text{Post}_{ub}(x_b)| \leq 1$ and $|\text{Post}_{ua}(x_a)| \leq 1$. Hence:

$$\forall u_a \in U_a \ \exists u_b \in U_b \ \forall x'_b \in \text{Post}_{ub}(x_b) \ \exists x'_a \in \text{Post}_{ua}(x_a) \text{ satisfying } (x'_a, x'_b) \in R.$$ 

becomes:

$$\forall u_a \in U_a \ \exists u_b \in U_b \text{ such that } (x'_a, x'_b) \in R.$$
How is simulation related to alternating simulation?

Alternating simulation degenerates into simulation in the very special case of deterministic systems.

Determinism implies $|\text{Post}_{u_b}(x_b)| \leq 1$ and $|\text{Post}_{u_a}(x_a)| \leq 1$. Hence:

$$\forall u_a \in U_a \exists u_b \in U_b \forall x'_b \in \text{Post}_{u_b}(x_b) \exists x'_a \in \text{Post}_{u_a}(x_a) \text{ satisfying } (x'_a, x'_b) \in R.$$ 

becomes:

$$\forall u_a \in U_a \exists u_b \in U_b \text{ such that } (x'_a, x'_b) \in R.$$ 

Equivalently:

$$x_a \xrightarrow{u_a}{_a} x'_a \text{ in } S_a \text{ implies the existence of } x_b \xrightarrow{u_b}{_b} x'_b \text{ in } S_b \text{ satisfying } (x'_a, x'_b) \in R.$$
How is simulation related to alternating simulation?

Alternating simulation degenerates into simulation in the very special case of deterministic systems.

Determinism implies $\left| \text{Post}_{ub}(x_b) \right| \leq 1$ and $\left| \text{Post}_{ua}(x_a) \right| \leq 1$. Hence:

$$\forall u_a \in U_a \ \exists u_b \in U_b \ \forall x'_b \in \text{Post}_{ub}(x_b) \ \exists x'_a \in \text{Post}_{ua}(x_a) \text{ satisfying } (x'_a, x'_b) \in R.$$ 

becomes:

$$\forall u_a \in U_a \ \exists u_b \in U_b \text{ such that } (x'_a, x'_b) \in R.$$ 

Equivalently:

$x_a \xrightarrow{ua}_a x'_a$ in $S_a$ implies the existence of $x_b \xrightarrow{ub}_b x'_b$ in $S_b$ satisfying $(x'_a, x'_b) \in R$. 

Can we use alternating simulations to relate properties enforced by controllers?
Proposition

Assume that $S_a \preceq_{AS} S_b$. If there exists a controller enforcing a LTL formula on $S_a$ then there exists a controller enforcing the same formula on $S_b$.

- This result ensures correctness of the approach. Any controller synthesized for the abstraction will lead to a controller for the original model.
Proposition

Assume that $S_a \preceq_{A_S} S_b$. If there exists a controller enforcing a LTL formula on $S_a$ then there exists a controller enforcing the same formula on $S_b$.

- This result ensures correctness of the approach. Any controller synthesized for the abstraction will lead to a controller for the original model.
- How about completeness?
Relating systems and their properties
Alternating simulation relations

Definition ((alternating) Bisimulation)
Let \( S_a \) and \( S_b \) be systems with \( Y_a = Y_b \). We say that \( S_a \) is (alternatingly) bisimilar to \( S_b \), denoted by \( S_a \sim AS S_b \), if there exists a (alternating) simulation relation \( R \) from \( S_a \) to \( S_b \) such that \( R^{-1} \) is a (alternating) simulation relation from \( S_b \) to \( S_a \).
### Definition ((alternating) Bisimulation)

Let $S_a$ and $S_b$ be systems with $Y_a = Y_b$. We say that $S_a$ is **(alternatingly) bisimilar** to $S_b$, denoted by $S_a \equiv_S S_b$ ($S_a \equiv_{AS} S_b$), if there exists a (alternating) simulation relation $R$ from $S_a$ to $S_b$ such that $R^{-1}$ is a (alternating) simulation relation from $S_b$ to $S_a$.

- $S_a \preceq_{AS} S_b$: every controller designed for $S_a$ leads to a controller for $S_b$. 
Relating systems and their properties

Alternating simulation relations

Definition ((alternating) Bisimulation)

Let $S_a$ and $S_b$ be systems with $Y_a = Y_b$. We say that $S_a$ is (alternatingly) bisimilar to $S_b$, denoted by $S_a \simeq_S S_b$ ($S_a \simeq_{AS} S_b$), if there exists a (alternating) simulation relation $R$ from $S_a$ to $S_b$ such that $R^{-1}$ is a (alternating) simulation relation from $S_b$ to $S_a$.

- $S_a \preceq_{AS} S_b$: every controller designed for $S_a$ leads to a controller for $S_b$.
- $S_b \preceq_{AS} S_a$: if there is a controller for $S_b$ then it can be synthesized as a controller for $S_a$.
Relating systems and their properties

Alternating simulation relations

Definition (alternating) Bisimulation

Let $S_a$ and $S_b$ be systems with $Y_a = Y_b$. We say that $S_a$ is (alternatingly) bisimilar to $S_b$, denoted by $S_a \cong_S S_b$ ($S_a \cong_{AS} S_b$), if there exists a (alternating) simulation relation $R$ from $S_a$ to $S_b$ such that $R^{-1}$ is a (alternating) simulation relation from $S_b$ to $S_a$.

- $S_a \preceq_{AS} S_b$: every controller designed for $S_a$ leads to a controller for $S_b$.
- $S_b \preceq_{AS} S_a$: if there is a controller for $S_b$ then it can be synthesized as a controller for $S_a$.

Given a linear differential equation, can we construct a finite-state abstraction related by an (alternating) (bi)simulation?
Relating systems and their properties

Alternating simulation relations

**Definition ((alternating) Bisimulation)**

Let $S_a$ and $S_b$ be systems with $Y_a = Y_b$. We say that $S_a$ is (alternatingly) bisimilar to $S_b$, denoted by $S_a \simeq_S S_b$ ($S_a \simeq_{AS} S_b$), if there exists a (alternating) simulation relation $R$ from $S_a$ to $S_b$ such that $R^{-1}$ is a (alternating) simulation relation from $S_b$ to $S_a$.

- $S_a \preceq_{AS} S_b$: every controller designed for $S_a$ leads to a controller for $S_b$.
- $S_b \preceq_{AS} S_a$: if there is a controller for $S_b$ then it can be synthesized as a controller for $S_a$.
- Given a linear differential equation, can we construct a finite-state abstraction related by an (alternating) (bi)simulation?
- In the second lecture we will solve this problem by placing restrictions on the predicates defining the output set $Y$ and the map $H$. 

In the third lecture we will drop these restrictions by relaxing the notion of (alternating) bisimulation to approximate (alternating) bisimulation.
Relating systems and their properties

Alternating simulation relations

Definition ((alternating) Bisimulation)

Let \( S_a \) and \( S_b \) be systems with \( Y_a = Y_b \). We say that \( S_a \) is \((alternatingly) bisimilar\) to \( S_b \), denoted by \( S_a \cong_S S_b \) \((S_a \cong_{AS} S_b)\), if there exists a \((alternating) simulation relation\) \( R \) from \( S_a \) to \( S_b \) such that \( R^{-1} \) is a \((alternating) simulation relation\) from \( S_b \) to \( S_a \).

- \( S_a \preceq_{AS} S_b \): every controller designed for \( S_a \) leads to a controller for \( S_b \).
- \( S_b \preceq_{AS} S_a \): if there is a controller for \( S_b \) then it can be synthesized as a controller for \( S_a \).

Given a linear differential equation, can we construct a finite-state abstraction related by an \((alternating) (bi)simulation\)?

- In the second lecture we will solve this problem by placing restrictions on the predicates defining the output set \( Y \) and the map \( H \).
- In the third lecture we will drop these restrictions by relaxing the notion of \((alternating) bisimulation\) to approximate \((alternating) bisimulation\).
Consider a DC motor:

\[
\begin{align*}
\dot{x}_1 &= -\frac{B}{J} x_1 + \frac{k}{J} x_2 \\
\dot{x}_2 &= -\frac{k}{L} x_1 - \frac{R}{L} x_2 + \frac{1}{L} u
\end{align*}
\]

where \( x_1 \) represents angular velocity and \( x_2 \) represents current.

The input voltage \( u \) is controlled by an H-bridge and thus ranges in the set \( \{-10, 0, 10\} \).
Playtime with PESSOA
A preview

The specification is:

- reach and stay at an angular velocity of 20 rad/s.
The specification is:

- reach and stay at an angular velocity of 20 rad/s.
- equivalently, $\Diamond \square (x_1 = 20)$. 
Playtime with PESSOA
A preview

The specification is:
- reach and stay at an angular velocity of 20 rad/s.
- equivalently, $\Diamond \Box (x_1 = 20)$. 
The specification is:

- reach and stay at an angular velocity of 20 rad/s.
- equivalently, $\diamondsuit \square (x_1 = 20)$.

- Maximal current too high!
- Large current ripple when velocity is close to 20 rad/s.
Change specification to:

- reach and stay at an angular velocity of 20 rad/s AND never exceed ±3A before reaching 20 rad/s AND never exceed ±0.7A after.
Change specification to:

- reach and stay at an angular velocity of 20 rad/s AND never exceed ±3A before reaching 20 rad/s AND never exceed ±0.7A after.
- equivalently, \((-3 \leq x_2 \leq 3) \cup \Box (x_1 = 20 \land -0.7 \leq x_2 \leq 0.7)\).
Change specification to:

- reach and stay at an angular velocity of 20 rad/s **AND** never exceed ±3A before reaching 20 rad/s **AND** never exceed ±0.7A after.
- equivalently, $(-3 \leq x_2 \leq 3) \U(x_1 = 20 \land -0.7 \leq x_2 \leq 0.7)$. 

All the missing details and references can be found in:

**Verification and Control of Hybrid Systems: A Symbolic Approach**