Synthesis for Cyber-Physical Systems

Paulo Tabuada

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

ExCAPE
Expeditions in Computer Augmented Program Engineering

CyPhyLab UCLA

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)

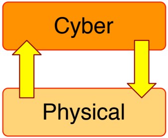
- Different people interpret the expression Cyber-Physical Systems (CPSs) differently;

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.

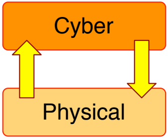


Introduction to Cyber-Physical Systems

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.

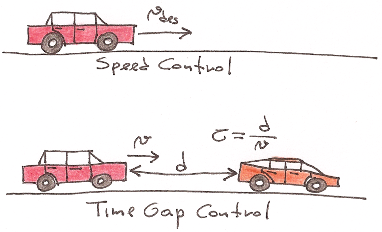
http://www.jpl.nasa.gov/video/

- The timing of the decisions made by the software is critical to the success of the mission.
- The “dynamics” of the software needs to analyzed in conjunction with the dynamics of the physical components.

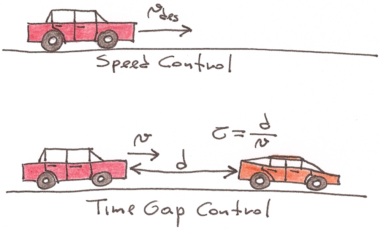
http://www.xerox.com/

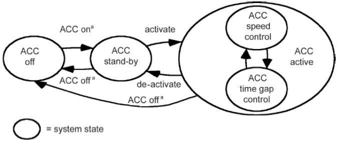
- A combination of time-driven and event-driven precise control decisions is required to avoid paper jams and maintain high throughput.

- Control theory provides analysis and design techniques for simple objectives.



- 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).





[a] 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.

Figure 3 — ACC states and transitions

- 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.

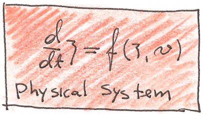
- Control theory provides analysis and design techniques for simple objectives.
- But current applications require sophisticated functionalities, e.g., adaptive cruise control (International Standard ISO 15622).
- Unfortunately, it is well known that switching between correct control software may lead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

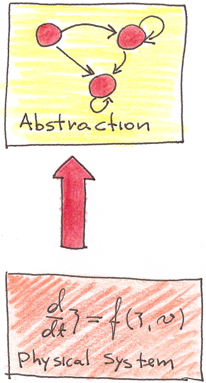
Synthesis for Cyber-Physical Systems

Key ingredients



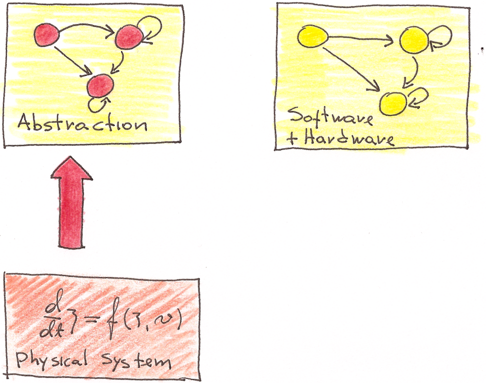
Synthesis for Cyber-Physical Systems

Key ingredients



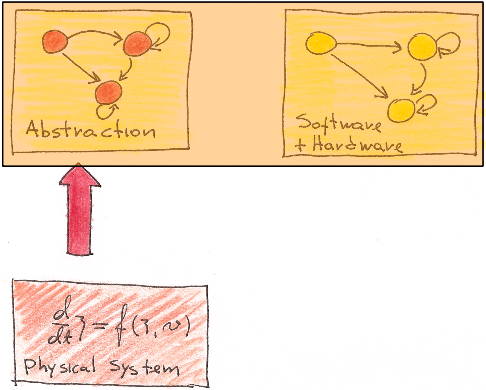
Synthesis for Cyber-Physical Systems

Key ingredients



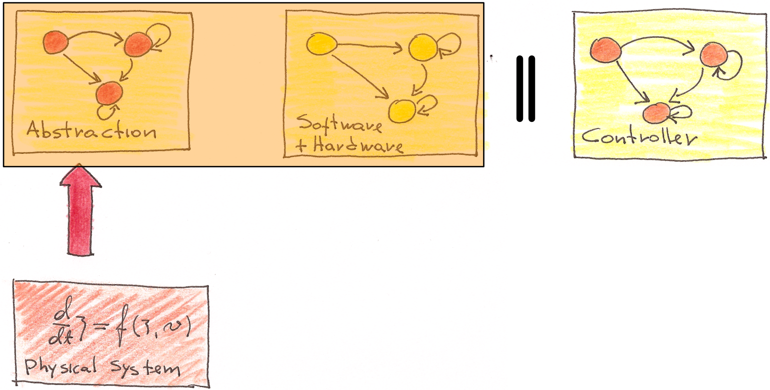
Synthesis for Cyber-Physical Systems

Key ingredients

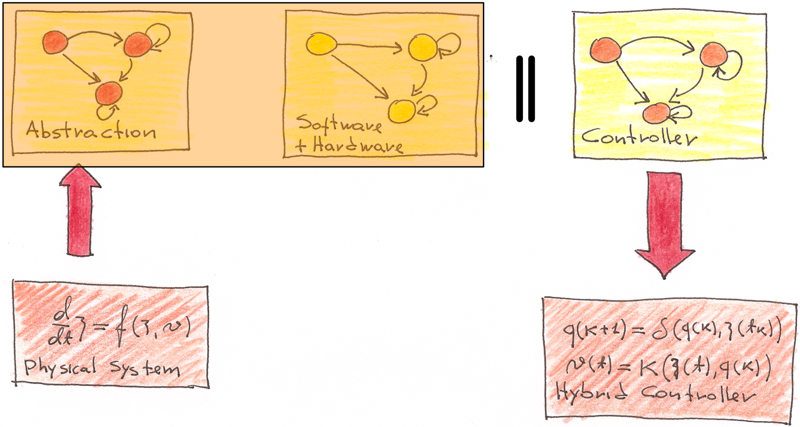


Synthesis for Cyber-Physical Systems

Key ingredients



Synthesis for Cyber-Physical Systems
Key ingredients



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:

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

- 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 $\xi : \mathbb{R}_0^+ \to \mathbb{R}^n$;
  - 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 $\xi : \mathbb{R}_0^+ \to \mathbb{R}^n$;
  - Inputs $u \in \mathbb{R}^m$ and input trajectories $\nu : \mathbb{R}_0^+ \to \mathbb{R}^m$;
- For each initial state $x \in \mathbb{R}^n$ and input trajectory $\nu : \mathbb{R}_0^+ \to \mathbb{R}^m$ there exists a unique state trajectory (solution, execution, run, trace, ...) $\xi_{x,\nu} : \mathbb{R}_0^+ \to \mathbb{R}^n$ 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:

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

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

- 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 $\tau \in \mathbb{R}^+$;

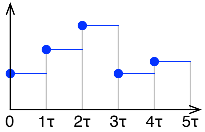
Models for the physical world

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:

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

- 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 $[k\tau, (k+1)\tau[$, $k \in \mathbb{N}_0$.



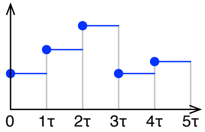
- Differential equations is the most common model for the physical world. In these lectures we will consider only linear differential equations:

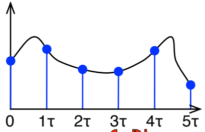
$$\frac{d}{dt}\xi = A\xi + B\nu. \tag{1}$$

- 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 $\xi'_{x,\nu'} : \mathbb{N}_0 \to \mathbb{R}^n$ of the difference equation:

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

satisfies $\xi'_{x,\nu'}(k) = \xi_{x,\nu'}(k\tau)$.





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

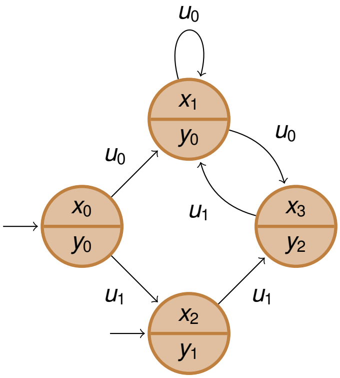
Definition (System)

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 $H : X \to Y$.

Models for the cyber and the physical world

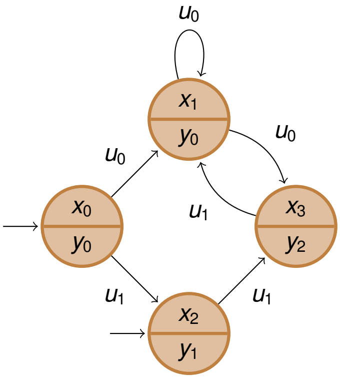
A useful graphical description



$$X = \{x_0, x_1, x_2, x_3\}$$

Models for the cyber and the physical world

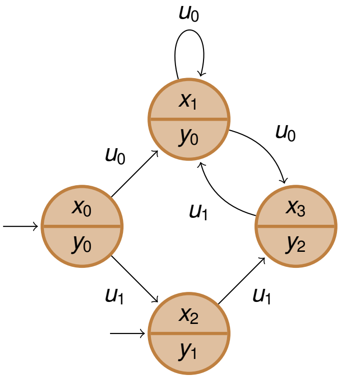
A useful graphical description



$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_2\}
\end{aligned}
$$

Models for the cyber and the physical world

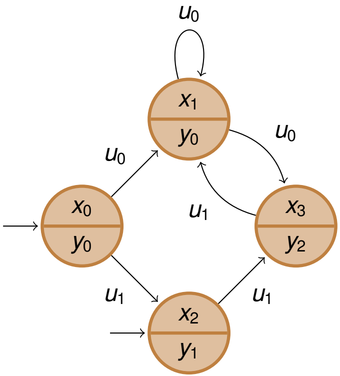
A useful graphical description



$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_2\} \\
U &= \{u_0, u_1\}
\end{aligned}
$$

Models for the cyber and the physical world

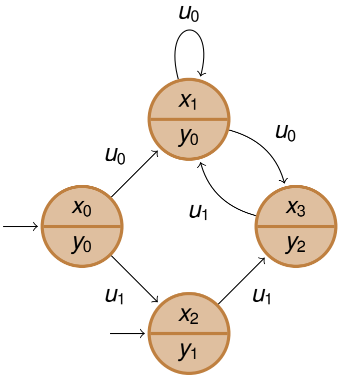
A useful graphical description



$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_2\} \\
U &= \{u_0, u_1\} \\
\longrightarrow &= \{(x_0, u_0, x_1), (x_0, u_1, x_2), (x_1, u_0, x_1), \\
&\quad (x_1, u_0, x_3), (x_2, u_1, x_3), (x_3, u_1, x_1)\}
\end{aligned}
$$

Models for the cyber and the physical world

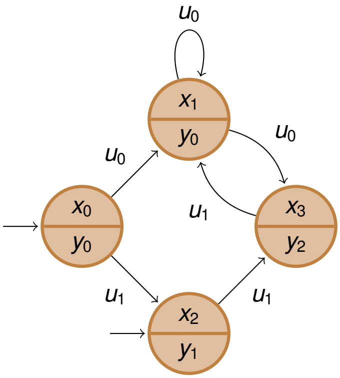
A useful graphical description



$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\} \\
X_0 &= \{x_0, x_2\} \\
U &= \{u_0, u_1\} \\
\longrightarrow &= \{(x_0, u_0, x_1), (x_0, u_1, x_2), (x_1, u_0, x_1), \\
&\quad (x_1, u_0, x_3), (x_2, u_1, x_3), (x_3, u_1, x_1)\} \\
Y &= \{y_0, y_1, y_2\}
\end{aligned}
$$

Models for the cyber and the physical world

A useful graphical description



$$
\begin{aligned}
X &= \{x_0, x_1, x_2, x_3\}\\
X_0 &= \{x_0, x_2\}\\
U &= \{u_0, u_1\}\\
\longrightarrow &= \{(x_0, u_0, x_1), (x_0, u_1, x_2), (x_1, u_0, x_1),\\
&\quad (x_1, u_0, x_3), (x_2, u_1, x_3), (x_3, u_1, x_1)\}\\
Y &= \{y_0, y_1, y_2\}\\
H(x_0) &= y_0, \quad H(x_1) = y_0, \quad H(x_2) = y_1\\
H(x_3) &= y_2.
\end{aligned}
$$

Models for the cyber and the physical world

A software example

- 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.

Models for the cyber and the physical world

A software example

- 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.

$$
\begin{aligned}
&\mathtt{x} := 0;\\
&\mathtt{n} := 0;\\
&\mathtt{While}(\textbf{true})\{\\
&\quad \mathtt{y} := \mathtt{read}(\mathtt{input});\\
&\quad \mathtt{x} := \mathtt{x}\frac{\mathtt{n}}{\mathtt{n}+1} + \mathtt{y}\frac{1}{\mathtt{n}+1};\\
&\quad \mathtt{n} := \mathtt{n}+1;\}
\end{aligned}
$$

- The variable *y* contains the latest received number and the variable *x* contains the average of the numbers that have been received so far.

Models for the cyber and the physical world

A software example

- 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.

$$\mathtt{x} := 0;$$

$$\mathtt{n} := 0;$$

$$\mathtt{While}(\textbf{true})\{$$

$$\quad \mathtt{y} := \mathtt{read}(\mathtt{input});$$

$$\quad \mathtt{x} := \mathtt{x}\frac{\mathtt{n}}{\mathtt{n}+1} + \mathtt{y}\frac{1}{\mathtt{n}+1};$$

$$\quad \mathtt{n} := \mathtt{n}+1;\}$$

- 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 $\{1, 2\}$.

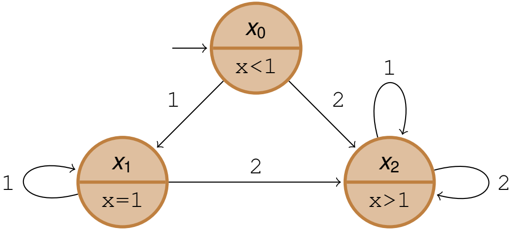
CyPhyLab UCLA

Models for the cyber and the physical world

A software example

- 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.

$$
\begin{array}{l}
\mathtt{x} := 0; \\
\mathtt{n} := 0; \\
\mathtt{While}(\textbf{true})\{ \\
\quad \mathtt{y} := \mathtt{read}(\mathtt{input}); \\
\quad \mathtt{x} := \mathtt{x}\dfrac{\mathtt{n}}{\mathtt{n}+1} + \mathtt{y}\dfrac{1}{\mathtt{n}+1}; \\
\quad \mathtt{n} := \mathtt{n}+1; \}
\end{array}
$$



- 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]:

$$\begin{aligned} c(k+1) &= \alpha\big(c(k)+i(k)+g(k)\big) \\ i(k+1) &= \beta\alpha\big(c(k)+i(k)+g(k)\big) - \beta c(k) \\ g(k+1) &= d(k). \end{aligned} \tag{2}$$

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.

[1] Interactions between the multiplier analysis and the principle of acceleration. The Review of Economic Statistics, 21(2):75-78, 1939.

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[1]:

$$\begin{aligned} c(k+1) &= \alpha\big(c(k)+i(k)+g(k)\big) \\ i(k+1) &= \beta\alpha\big(c(k)+i(k)+g(k)\big) - \beta c(k) \\ g(k+1) &= d(k). \end{aligned} \tag{2}$$

  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.

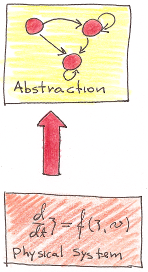
- We can describe this model by the following system:
  - $X = \mathbb{R}^3$, $X_0 = X$, $U = \mathbb{R}_0^+$;
  - $(c(k), i(k), g(k)) \xrightarrow{d(k)} (c(k+1), i(k+1), g(k+1))$ if equations (2) are satisfied;
  - $Y = \mathbb{R}$, $H = c + i + g$.

[1] Interactions between the multiplier analysis and the principle of acceleration. The Review of Economic Statistics, 21(2):75-78, 1939.

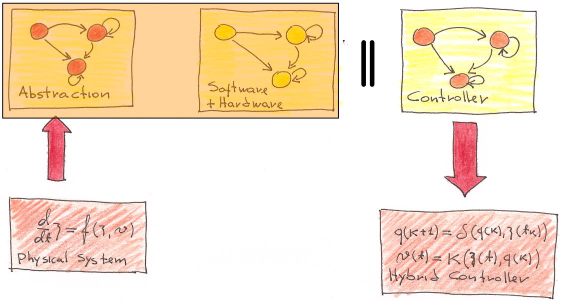
Relating systems and their properties

Simulation relations

- 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?

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

Definition (Simulation Relation)

Let $S_a = (X_a, X_{a0}, U_a, \xrightarrow[a]{}, Y_a, H_a)$ and $S_b = (X_b, X_{b0}, U_b, \xrightarrow[b]{}, Y_b, H_b)$ 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:

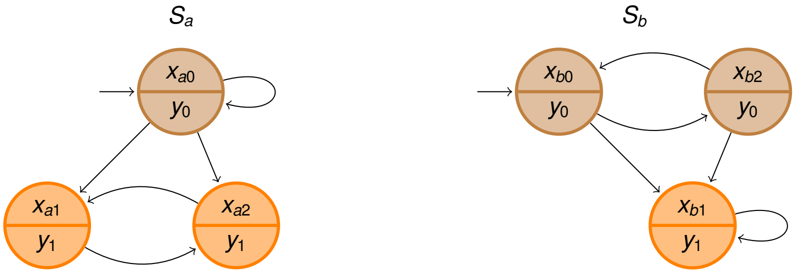
1. for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
3. for every $(x_a, x_b) \in R$ we have that:

   $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ in S_b 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:

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

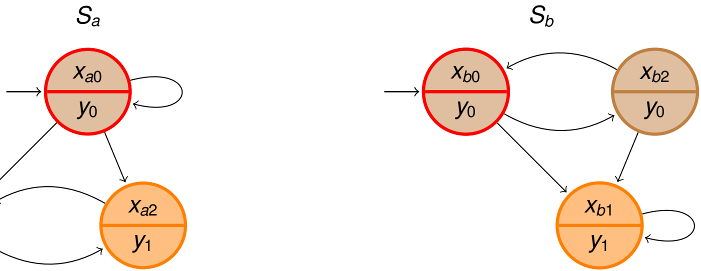


Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:

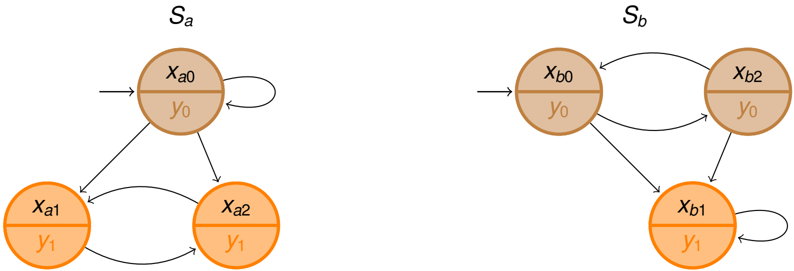
$$R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}.$$



1. for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;

- Consider the following two systems and the relation:

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



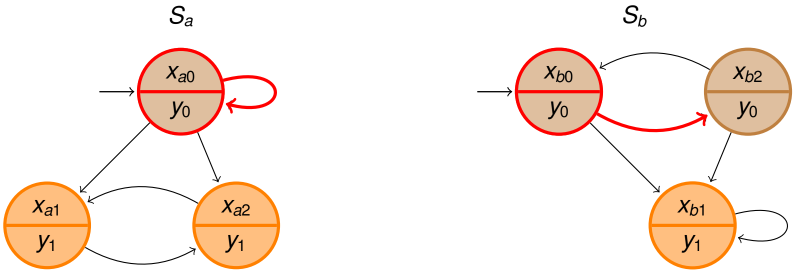
1. for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;

Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:

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



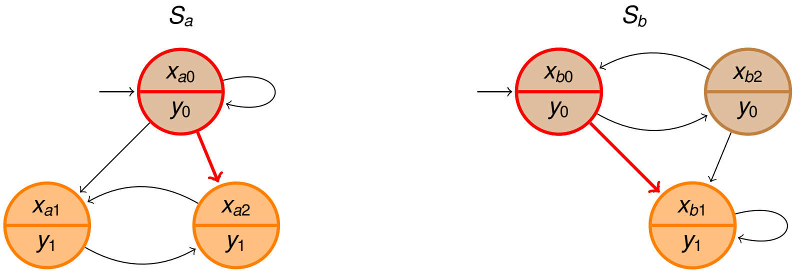
1. for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
3. $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ in S_b satisfying $(x'_a, x'_b) \in R$.

Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:

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



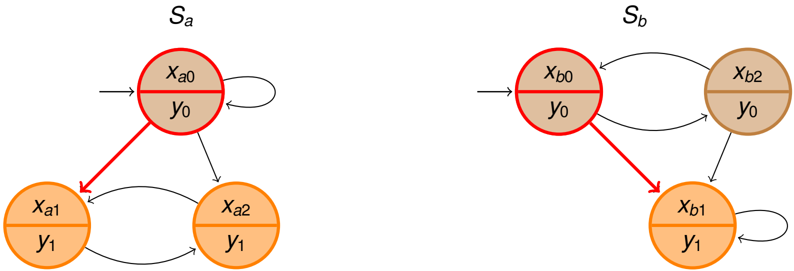
1. for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
3. $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ in S_b satisfying $(x'_a, x'_b) \in R$.

Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:

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



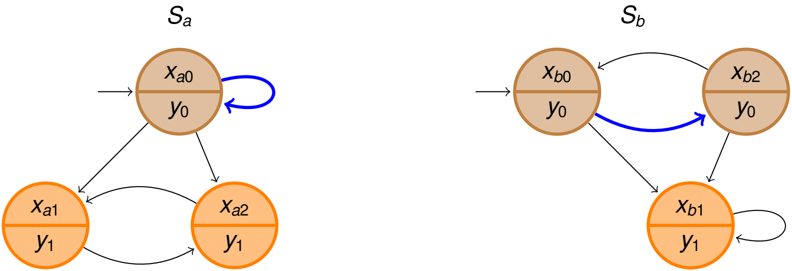
1. for every $x_{a0} \in X_{a0}$, there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $H_a(x_a) = H_b(x_b)$;
3. $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ in S_b satisfying $(x'_a, x'_b) \in R$.

Relating systems and their properties

Simulation relations: Example

- Consider the following two systems and the relation:

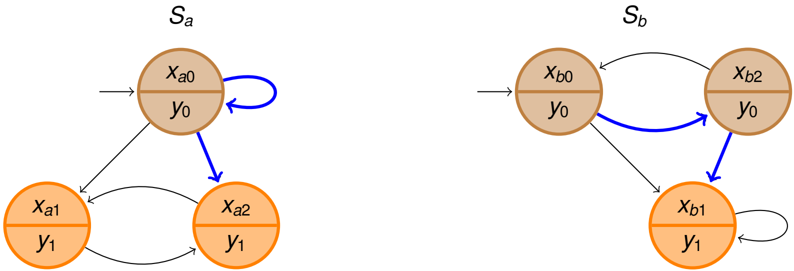
$$R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}.$$



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. $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ in S_b satisfying $(x'_a, x'_b) \in R$.

- Consider the following two systems and the relation:

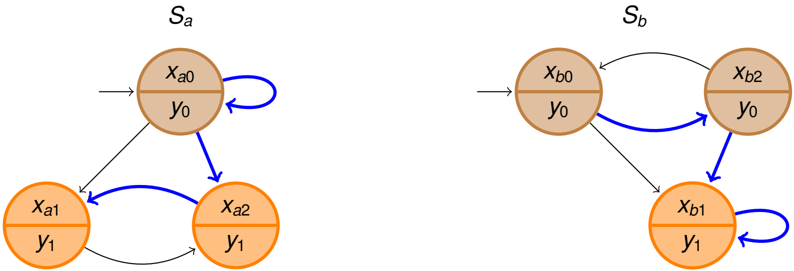
$$R = \{(x_{a0}, x_{b0}), (x_{a0}, x_{b2}), (x_{a1}, x_{b1}), (x_{a2}, x_{b1})\}.$$



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. $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ 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. $x_a \xrightarrow[a]{u_a} x'_a$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x'_b$ 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 φ we have:

$$S_a \preceq_{\mathcal{S}} S_b \implies (S_b \models \varphi \implies S_a \models \varphi)\,.$$

- 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 φ we have:

$$S_a \preceq_{\mathcal{S}} S_b \implies (S_b \models \varphi \implies S_a \models \varphi)\,.$$

- 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 φ we have:

$$S_a \preceq_{\mathcal{S}} S_b \implies (S_b \models \varphi \implies S_a \models \varphi) \,.$$

- 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.

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 $x \xrightarrow{u} x'$.

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.

- 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 $\mathrm{Post}_u(x)$ the set of all the states that can be reached from x under input u, formally:

$$\mathrm{Post}_u(x) = \{x' \in X \mid x \xrightarrow{u} x'\}.$$

- The enforcement of properties expressed in LTL can be transferred between systems related by alternating simulations.

Definition (Alternating simulation relation)

Let $S_a = (X_a, X_{a0}, U_a, \underset{a}{\longrightarrow}, Y_a)$ and $S_b = (X_b, X_{b0}, U_b, \underset{b}{\longrightarrow}, Y_b)$ 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 $x_b' \in \mathrm{Post}_{u_b}(x_b)$ there exists $x_a' \in \mathrm{Post}_{u_a}(x_a)$ satisfying $(x_a', x_b') \in R$.

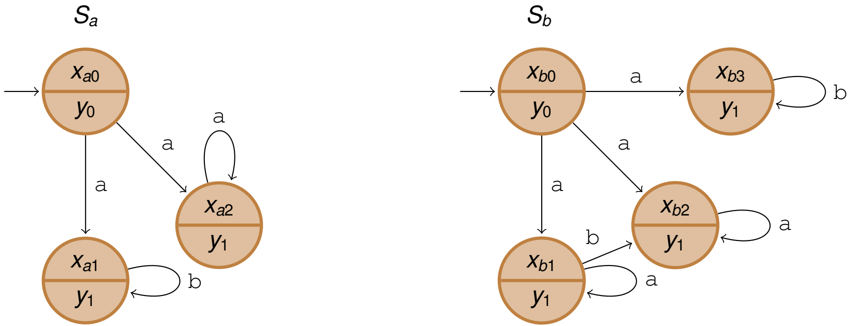
We say that S_a is alternatingly simulated by S_b or that S_b alternatingly simulates S_a, denoted by $S_a \preceq_{\mathcal{AS}} S_b$, if there exists an alternating simulation relation from S_a to S_b.

- How is simulation related to alternating simulation?

Relating systems and their properties

Alternating simulation relations

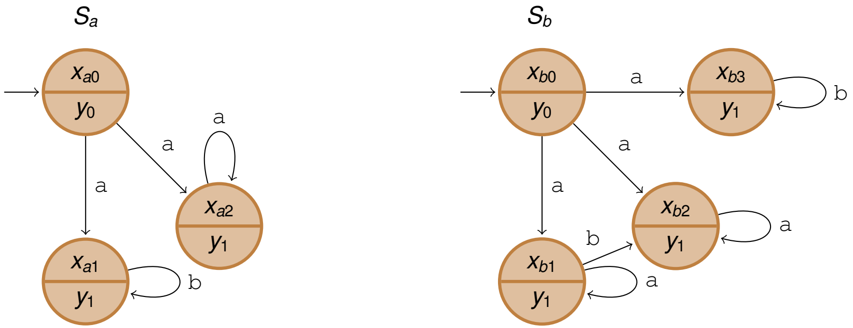
- How is simulation related to alternating simulation?



- 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 $x_{b3} \in \text{Post}_{\mathtt{a}}(x_{b0})$ in S_b no state in S_a is related to x_{b3}.

Relating systems and their properties
Alternating simulation relations

- 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 $x_{a0} \xrightarrow[a]{\mathtt{a}} x_{a2}$ in S_a cannot be matched by S_b.

- How is simulation related to alternating simulation?
- Alternating simulation degenerates into simulation in the very special case of deterministic systems.
- Determinism implies $|\mathrm{Post}_{u_b}(x_b)| \leq 1$ and $|\mathrm{Post}_{u_a}(x_a)| \leq 1$.

- How is simulation related to alternating simulation?
- Alternating simulation degenerates into simulation in the very special case of deterministic systems.
- Determinism implies $|\mathrm{Post}_{u_b}(x_b)| \leq 1$ and $|\mathrm{Post}_{u_a}(x_a)| \leq 1$. Hence:

$\forall u_a \in U_a \; \exists u_b \in U_b \; \forall x_b' \in \mathrm{Post}_{u_b}(x_b) \; \exists x_a' \in \mathrm{Post}_{u_a}(x_a)$ satisfying $(x_a', x_b') \in R$.

becomes:

$\forall u_a \in U_a \; \exists u_b \in U_b$ such that $(x_a', x_b') \in R$.

Relating systems and their properties

Alternating simulation relations

- How is simulation related to alternating simulation?
- Alternating simulation degenerates into simulation in the very special case of deterministic systems.
- Determinism implies $|\mathrm{Post}_{u_b}(x_b)| \leq 1$ and $|\mathrm{Post}_{u_a}(x_a)| \leq 1$. Hence:

  $\forall u_a \in U_a \; \exists u_b \in U_b \; \forall x_b' \in \mathrm{Post}_{u_b}(x_b) \; \exists x_a' \in \mathrm{Post}_{u_a}(x_a)$ satisfying $(x_a', x_b') \in R$.

  becomes:

  $\forall u_a \in U_a \; \exists u_b \in U_b$ such that $(x_a', x_b') \in R$.

  Equivalently:

  $x_a \xrightarrow[a]{u_a} x_a'$ in S_a implies the existence of $x_b \xrightarrow[b]{u_b} x_b'$ in S_b 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 $|\mathrm{Post}_{u_b}(x_b)| \leq 1$ and $|\mathrm{Post}_{u_a}(x_a)| \leq 1$. Hence:

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

  becomes:

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

  Equivalently:

  $$x_a \xrightarrow[a]{u_a} x_a' \text{ in } S_a \text{ implies the existence of } x_b \xrightarrow[b]{u_b} x_b' \text{ in } S_b \text{ satisfying } (x_a', x_b') \in R.$$

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

CyPhyLab UCLA

Proposition

Assume that $S_a \preceq_{\mathcal{AS}} S_b$. 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 $S_a \preceq_{\mathcal{AS}} S_b$. 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.
- How about completeness?

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 $S_a \cong_{\mathcal{S}} S_b$ ($S_a \cong_{\mathcal{AS}} S_b$), 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 $S_a \cong_{\mathcal{S}} S_b$ ($S_a \cong_{\mathcal{AS}} S_b$), 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.

- $S_a \preceq_{\mathcal{AS}} S_b$: every controller designed for S_a leads to a controller for S_b.

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 $S_a \cong_{\mathcal{S}} S_b$ ($S_a \cong_{\mathcal{AS}} S_b$), 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.

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

- $S_a \preceq_{\mathcal{AS}} S_b$: every controller designed for S_a leads to a controller for S_b.
- $S_b \preceq_{\mathcal{AS}} S_a$: 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?

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 $S_a \cong_{\mathcal{S}} S_b$ ($S_a \cong_{\mathcal{AS}} S_b$), 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.

- $S_a \preceq_{\mathcal{AS}} S_b$: every controller designed for S_a leads to a controller for S_b.
- $S_b \preceq_{\mathcal{AS}} S_a$: 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.

Definition ((alternating) Bisimulation)

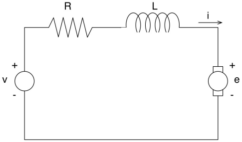
Let S_a and S_b be systems with $Y_a = Y_b$. We say that S_a is (alternatingly) bisimilar to S_b, denoted by $S_a \cong_{\mathcal{S}} S_b$ ($S_a \cong_{\mathcal{AS}} S_b$), if there exists a (alternating) simulation relation R from S_a to S_b such that R^{-1} is a (alternating) simulation relation from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

- $S_a \preceq_{\mathcal{AS}} S_b$: every controller designed for S_a leads to a controller for S_b.
- $S_b \preceq_{\mathcal{AS}} S_a$: 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.

Playtime with PESSOA

A preview

- Consider a DC motor:



$$\begin{aligned} \dot{x}_1 &= -\frac{B}{J}x_1 + \frac{k}{J}x_2 \\ \dot{x}_2 &= -\frac{k}{L}x_1 - \frac{R}{L}x_2 + \frac{1}{L}u \end{aligned}$$

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.

Playtime with PESSOA

A preview

The specification is:

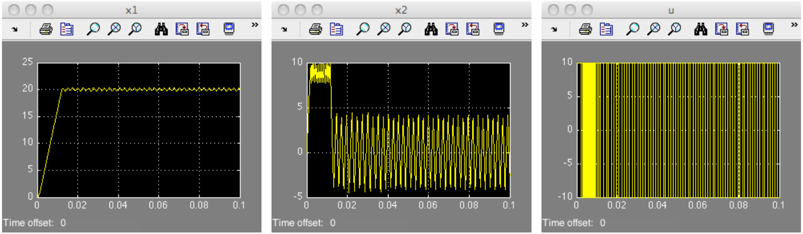
- reach and stay at an angular velocity of 20 rad/s.
- equivalently, $\Diamond\Box(x_1 = 20)$.

Playtime with PESSOA

A preview

The specification is:

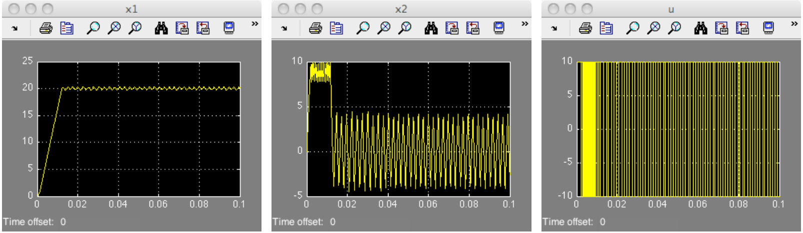
- reach and stay at an angular velocity of 20 rad/s.
- equivalently, $\Diamond\Box(x_1 = 20)$.



Playtime with PESSOA
A preview

The specification is:

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



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

Playtime with PESSOA

A preview

Change specification to:

- reach and stay at an angular velocity of 20 rad/s AND never exceed $\pm$3A before reaching 20 rad/s AND never exceed $\pm$0.7A after.

Change specification to:

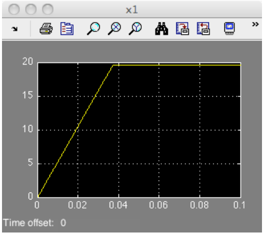
- reach and stay at an angular velocity of 20 rad/s AND never exceed ± 3A before reaching 20 rad/s AND never exceed ± 0.7A after.
- equivalently, $(-3 \leq x_2 \leq 3)\, \mathrm{U}\, \Box (x_1 = 20 \wedge -0.7 \leq x_2 \leq 0.7)$.

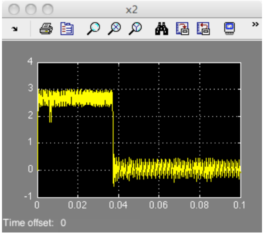
Playtime with PESSOA

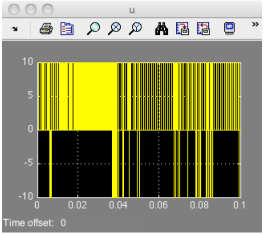
A preview

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, $(-3 \leq x_2 \leq 3)\,\mathrm{U}\,\Box(x_1 = 20 \wedge -0.7 \leq x_2 \leq 0.7)$.

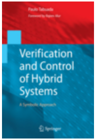






References and more details

- All the missing details and references can be found in:



Verification and Control of Hybrid Systems: A Symbolic Approach
Springer, 2009.