

From formal verification to formal synthesis

Paulo Tabuada

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

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

- verification is used (too) late in the design process, typically after substantial resources have been invested and many design choices have been made;

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

- verification is used (too) late in the design process, typically after substantial resources have been invested and many design choices have been made;
- when a bug is found, the system (or sub-components) need to be redesigned and there is no guarantee that the redesign will be bug-free;

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

- verification is used (too) late in the design process, typically after substantial resources have been invested and many design choices have been made;
- when a bug is found, the system (or sub-components) need to be redesigned and there is no guarantee that the redesign will be bug-free;
- the iteration between verification and redesign is expensive.

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

- verification is used (too) late in the design process, typically after substantial resources have been invested and many design choices have been made;
- when a bug is found, the system (or sub-components) need to be redesigned and there is no guarantee that the redesign will be bug-free;
- the iteration between verification and redesign is expensive.

Can we use formal techniques earlier in the design process?

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

- verification is used (too) late in the design process, typically after substantial resources have been invested and many design choices have been made;
- when a bug is found, the system (or sub-components) need to be redesigned and there is no guarantee that the redesign will be bug-free;
- the iteration between verification and redesign is expensive.

Can we use formal techniques earlier in the design process?

Can we automate the design process?

Formal Verification

Formal verification is one of the few techniques providing rigorous guarantees of correctness and performance for complex engineered systems.

However,...

- verification is used (too) late in the design process, typically after substantial resources have been invested and many design choices have been made;
- when a bug is found, the system (or sub-components) need to be redesigned and there is no guarantee that the redesign will be bug-free;
- the iteration between verification and redesign is expensive.

Can we use formal techniques earlier in the design process?

Can we automate the design process?

Can we put more effort into the *design* so as to rely less on verification?

Formal Synthesis

Synthesis has always been the paradigm of choice in control theory:

- the design process for feedback control laws also furnishes a proof of its correctness and sometimes of its performance.

Formal Synthesis

Synthesis has always been the paradigm of choice in control theory:

- the design process for feedback control laws also furnishes a proof of its correctness and sometimes of its performance.

Although far from the paradigm of choice, synthesis is a old dream in computer science:

- In 1957 A. Church asked the following question: given a description of a desired input-output behavior, does there exist a switching circuit implementing such input-output behavior?

Formal Synthesis

Synthesis has always been the paradigm of choice in control theory:

- the design process for feedback control laws also furnishes a proof of its correctness and sometimes of its performance.

Although far from the paradigm of choice, synthesis is a old dream in computer science:

- In 1957 A. Church asked the following question: given a description of a desired input-output behavior, does there exist a switching circuit implementing such input-output behavior? The answer was later given by J. Büchi using automata on infinite words;

Formal Synthesis

Synthesis has always been the paradigm of choice in control theory:

- the design process for feedback control laws also furnishes a proof of its correctness and sometimes of its performance.

Although far from the paradigm of choice, synthesis is a old dream in computer science:

- In 1957 A. Church asked the following question: given a description of a desired input-output behavior, does there exist a switching circuit implementing such input-output behavior? The answer was later given by J. Büchi using automata on infinite words;
- In 1981, E. Clarke and E. Emerson considered the synthesis of synchronization skeletons from temporal logic;

Formal Synthesis

Synthesis has always been the paradigm of choice in control theory:

- the design process for feedback control laws also furnishes a proof of its correctness and sometimes of its performance.

Although far from the paradigm of choice, synthesis is a old dream in computer science:

- In 1957 A. Church asked the following question: given a description of a desired input-output behavior, does there exist a switching circuit implementing such input-output behavior? The answer was later given by J. Büchi using automata on infinite words;
- In 1981, E. Clarke and E. Emerson considered the synthesis of synchronization skeletons from temporal logic;
- In 1989, A. Pnueli and R. Rosner considered the synthesis of reactive modules;

Formal Synthesis

Synthesis has always been the paradigm of choice in control theory:

- the design process for feedback control laws also furnishes a proof of its correctness and sometimes of its performance.

Although far from the paradigm of choice, synthesis is a old dream in computer science:

- In 1957 A. Church asked the following question: given a description of a desired input-output behavior, does there exist a switching circuit implementing such input-output behavior? The answer was later given by J. Büchi using automata on infinite words;
- In 1981, E. Clarke and E. Emerson considered the synthesis of synchronization skeletons from temporal logic;
- In 1989, A. Pnueli and R. Rosner considered the synthesis of reactive modules;
- Many other publications since, especially in the last 5 years.

Formal Synthesis

Why isn't synthesis predominant in computer science?

Why isn't synthesis predominant in computer science?

- Solution complexity

Why isn't synthesis predominant in computer science?

- Solution complexity
 - The same advances that are making verification practical can also be used to make synthesis practical;

Why isn't synthesis predominant in computer science?

- Solution complexity
 - The same advances that are making verification practical can also be used to make synthesis practical;
 - Focus on structured software such as embedded controllers;

Why isn't synthesis predominant in computer science?

- Solution complexity
 - The same advances that are making verification practical can also be used to make synthesis practical;
 - Focus on structured software such as embedded controllers;
- Specification complexity
 - Focus on structured software such as embedded controllers;

Why isn't synthesis predominant in computer science?

- Solution complexity
 - The same advances that are making verification practical can also be used to make synthesis practical;
 - Focus on structured software such as embedded controllers;
- Specification complexity
 - Focus on structured software such as embedded controllers;
 - Need advances on compositional synthesis and partial synthesis.

Why isn't synthesis predominant in computer science?

- Solution complexity
 - The same advances that are making verification practical can also be used to make synthesis practical;
 - Focus on structured software such as embedded controllers;
- Specification complexity
 - Focus on structured software such as embedded controllers;
 - Need advances on compositional synthesis and partial synthesis.

The benefits are well worth the challenges that lie ahead!

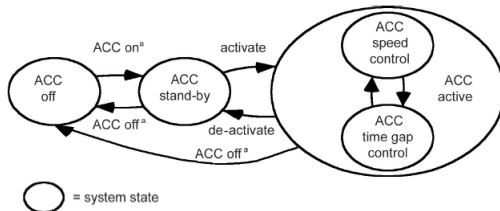
Synthesis of embedded control software

The hybrid nature (discrete+continuous) of many engineered systems is not inherent but man made.

Synthesis of embedded control software

The hybrid nature (discrete+continuous) of many engineered systems is not inherent but man made.

International Standard ISO 15622 (Adaptive Cruise Control)



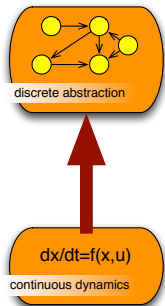
^a Manually and/or automatically after self test. Manual transition describes a switch to enable/disable ACC function. Automatic switch off can be forced by failure reaction.

Figure 3 — ACC states and transitions

Synthesis of embedded control software

Consider a three phase approach to the synthesis of embedded control software.

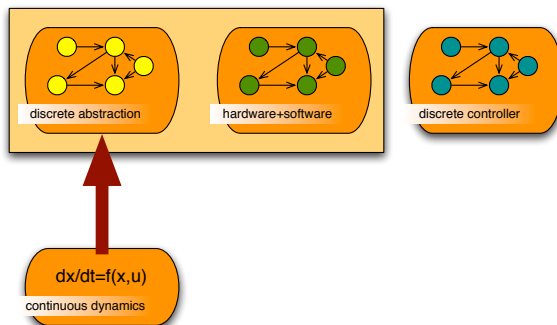
Abstraction



Synthesis of embedded control software

Consider a three phase approach to the synthesis of embedded control software.

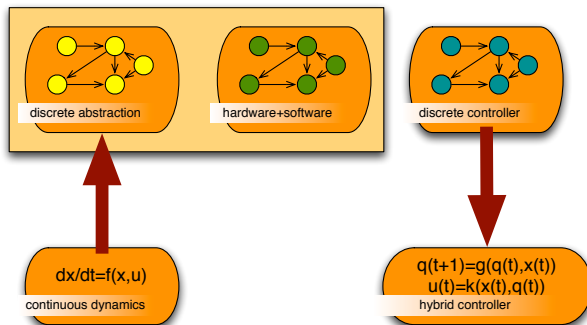
Discrete controller synthesis



Synthesis of embedded control software

Consider a three phase approach to the synthesis of embedded control software.

Controller refinement



Synthesis of embedded control software

- **Abstraction:** key step in this approach and focus of the remaining talk;
- **Discrete controller synthesis:** use existing algorithms from algorithmic game theory or discrete-event systems;
- **Controller refinement:** simple consequence of abstraction.

Abstractions for synthesis

Systems

Definition (System)

A system is a quintuple $S = (X, U, \longrightarrow, Y, H)$ consisting of:

- A set of states X ;
- A set of inputs U ;
- A transition relation $\longrightarrow \subseteq X \times U \times X$;
- An output set Y ;
- An output function $H : X \rightarrow Y$.

We will say that S is finite when X is finite.

Abstractions for synthesis

Control systems

Definition

A control system is a triple $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ where:

- \mathbb{R}^n is the state space;
- \mathcal{U} is “nice” subset of the set of all functions of time from intervals of the form $]a, b[\subseteq \mathbb{R}$ to \mathbb{R}^m with $a < 0$ and $b > 0$;
- $f : \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}^n$ is a “nice” continuous map.

Abstractions for synthesis

Control systems

Definition

A control system is a triple $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ where:

- \mathbb{R}^n is the state space;
- \mathcal{U} is “nice” subset of the set of all functions of time from intervals of the form $]a, b[\subseteq \mathbb{R}$ to \mathbb{R}^m with $a < 0$ and $b > 0$;
- $f : \mathbb{R}^n \times \mathcal{U} \rightarrow \mathbb{R}^n$ is a “nice” continuous map.

A “nice” curve $\xi :]a, b[\rightarrow \mathbb{R}^n$ is said to be a trajectory of Σ if there exists $v \in \mathcal{U}$ satisfying $\dot{\xi}(t) = f(\xi(t), v(t))$ for almost all $t \in]a, b[$.

Abstractions for synthesis

Control systems as systems

Given a control system $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ and a sampling time $\tau \in \mathbb{R}^+$, the system $S_\tau(\Sigma) := (X, U, \longrightarrow, Y, H)$ associated with Σ is given by:

- $X = \mathbb{R}^n$;
- U is the set of all the curves in \mathcal{U} of duration τ ;
- $x \xrightarrow{v} x'$ if $\xi_{xv}(\tau) = x'$;
- $Y = \mathbb{R}^n$;
- $H = 1_{\mathbb{R}^n}$.

The output set $Y = \mathbb{R}^n$ is equipped with the metric $\mathbf{d}(y, y') = \|y - y'\|$.

Abstractions for synthesis

Control systems as systems

Given a control system $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ and a sampling time $\tau \in \mathbb{R}^+$, the system $S_\tau(\Sigma) := (X, U, \longrightarrow, Y, H)$ associated with Σ is given by:

- $X = \mathbb{R}^n$;
- U is the set of all the curves in \mathcal{U} of duration τ ;
- $x \xrightarrow{v} x'$ if $\xi_{xv}(\tau) = x'$;
- $Y = \mathbb{R}^n$;
- $H = 1_{\mathbb{R}^n}$.

The output set $Y = \mathbb{R}^n$ is equipped with the metric $\mathbf{d}(y, y') = \|y - y'\|$.

Can we replace $S_\tau(\Sigma)$ with an equivalent and yet finite system?

Abstractions for synthesis

Approximate (bi)simulation

The usual notion of (bi)simulation requires exact matching of outputs.

Definition

Let $S_1 = (X_1, U_1, \xrightarrow{1}, Y, H_1)$ and $S_2 = (X_2, U_2, \xrightarrow{2}, Y, H_2)$ be systems with the same output space Y . A relation $R \subseteq X_1 \times X_2$ is said to be a simulation relation from S_1 to S_2 if $(x_1, x_2) \in R$ implies:

- 1 $H_1(x_1) = H_2(x_2)$;
- 2 $x_1 \xrightarrow{u_1} x'_1$ implies $x_2 \xrightarrow{u_2} x'_2$ with $(x_1, x_2) \in R$.

Abstractions for synthesis

Approximate (bi)simulation

The usual notion of (bi)simulation requires exact matching of outputs.

Definition

Let $S_1 = (X_1, U_1, \xrightarrow{1}, Y, H_1)$ and $S_2 = (X_2, U_2, \xrightarrow{2}, Y, H_2)$ be systems with the same output space Y . A relation $R \subseteq X_1 \times X_2$ is said to be a simulation relation from S_1 to S_2 if $(x_1, x_2) \in R$ implies:

- 1 $H_1(x_1) = H_2(x_2)$;
- 2 $x_1 \xrightarrow{u_1} x'_1$ implies $x_2 \xrightarrow{u_2} x'_2$ with $(x_1, x_2) \in R$.

Relation R is said to be a bisimulation relation between S_1 and S_2 if, in addition to 1. and 2., $(x_1, x_2) \in R$ also implies:

- 3 $x_2 \xrightarrow{u_2} x'_2$ implies $x_1 \xrightarrow{u_1} x'_1$ with $(x_1, x_2) \in R$.

Abstractions for synthesis

Approximate (bi)simulation

Relaxing the equality constraint $H_1(x_1) = H_2(x_2)$ leads to approximate (bi)simulation.

Definition (Girard and Pappas 2005, Tabuada 2005)

Let $S_1 = (X_1, U_1, \xrightarrow{1}, Y, H_1)$ and $S_2 = (X_2, U_2, \xrightarrow{2}, Y, H_2)$ be **metric** systems with the same output space Y and let $\varepsilon \in \mathbb{R}^+$ be a desired precision. A relation $R \subseteq X_1 \times X_2$ is said to be a ε -approximate simulation relation from S_1 to S_2 if $(x_1, x_2) \in R$ implies:

- 1 $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$;
- 2 $x_1 \xrightarrow{u_1} x'_1$ implies $x_2 \xrightarrow{u_2} x'_2$ with $(x_1, x_2) \in R$.

Abstractions for synthesis

Approximate (bi)simulation

Relaxing the equality constraint $H_1(x_1) = H_2(x_2)$ leads to approximate (bi)simulation.

Definition (Girard and Pappas 2005, Tabuada 2005)

Let $S_1 = (X_1, U_1, \xrightarrow{1}, Y, H_1)$ and $S_2 = (X_2, U_2, \xrightarrow{2}, Y, H_2)$ be **metric** systems with the same output space Y and let $\varepsilon \in \mathbb{R}^+$ be a desired precision. A relation $R \subseteq X_1 \times X_2$ is said to be a ε -approximate simulation relation from S_1 to S_2 if $(x_1, x_2) \in R$ implies:

1 $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$;

2 $x_1 \xrightarrow{u_1} x'_1$ implies $x_2 \xrightarrow{u_2} x'_2$ with $(x_1, x_2) \in R$.

Relation R is said to be a bisimulation relation between S_1 and S_2 if, in addition to 1. and 2., $(x_1, x_2) \in R$ also implies:

3 $x_2 \xrightarrow{u_2} x'_2$ implies $x_1 \xrightarrow{u_1} x'_1$ with $(x_1, x_2) \in R$.

Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

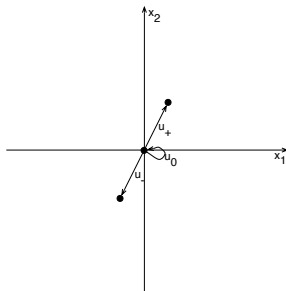
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

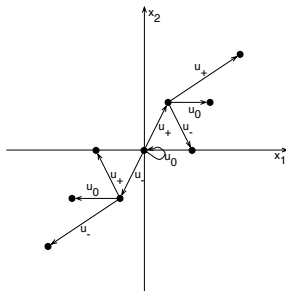
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

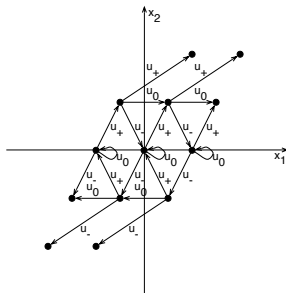
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

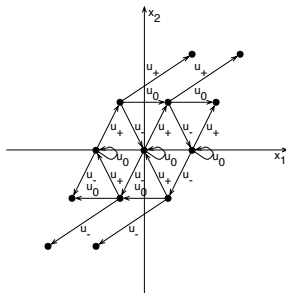
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Can we extrapolate from this finite system?

Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

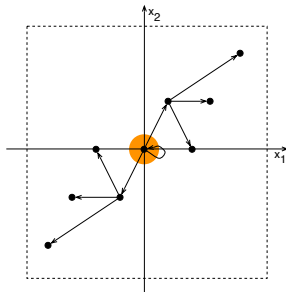
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Yes, provided that we know how to robustify it!

Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

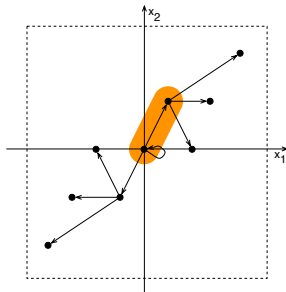
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Yes, provided that we know how to robustify it!

Abstractions for synthesis

A simple idea

$$\dot{x}_1 = x_2$$

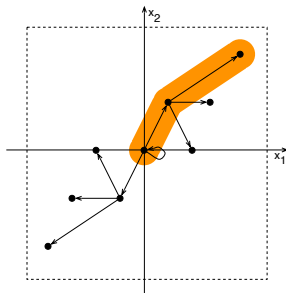
$$\dot{x}_2 = u$$

$$\mathcal{U} = \{u_-, u_0, u_+\}$$

$$u_-(t) = -1 \quad \forall t \in [0, 1]$$

$$u_0(t) = 0 \quad \forall t \in [0, 1]$$

$$u_+(t) = 1 \quad \forall t \in [0, 1]$$



Yes, provided that we know how to robustify it!

Abstractions for synthesis

Existence

Theorem

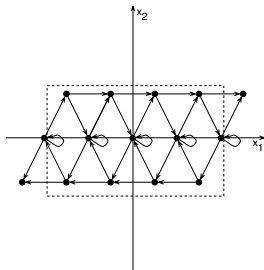
Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

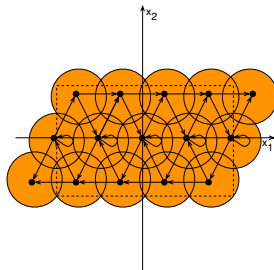


Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

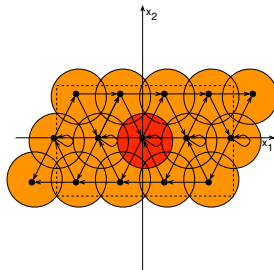


Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

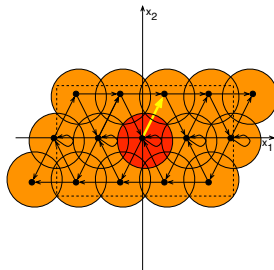


Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

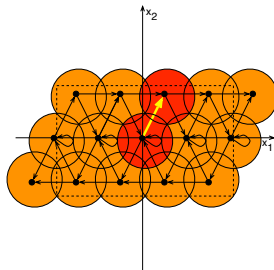


Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.



Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

There exists a controller enforcing a desired specification on $S_\tau(\Sigma)$ iff there exists a controller enforcing the same specification on the bisimilar finite system.

Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

There exists a controller enforcing a desired specification on $S_\tau(\Sigma)$ iff there exists a controller enforcing the same specification on the bisimilar finite system.

For linear systems, δ -ISS is equivalent to asymptotic stability.

Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally input-to-state stable (δ -ISS), then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately bisimilar to a finite system.

There exists a controller enforcing a desired specification on $S_\tau(\Sigma)$ iff there exists a controller enforcing the same specification on the bisimilar finite system.

For linear systems, δ -ISS is equivalent to asymptotic stability.

If this assumption is not satisfied, we may be able to synthesize a continuous controller enforcing it.

Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally forward complete, then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately alternatingly simulated by a finite system.

Abstractions for synthesis

Existence

Theorem

Let Σ be a digital control system and let $\varepsilon \in \mathbb{R}^+$ be any desired precision. If Σ is incrementally forward complete, then the restriction of $S_\tau(\Sigma)$ to any compact subset of X is ε -approximately alternatingly simulated by a finite system.

If there exists a controller enforcing a desired specification on the finite system, then there exists a controller enforcing the same specification on $S_\tau(\Sigma)$.

Does it really work?

Pessoa

We've built a tool around these ideas, called **Pessoa**, that will be publicly available in the next weeks.

A simple inverted pendulum example:

$$\Sigma : \begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \frac{g}{l} x_1 - \frac{\gamma}{m l^2} x_2 + \frac{1}{m l^2} u, \end{cases}$$

$X = [-1, 1] \times [-1, 1]$, $U = [-4, 4]$, $\varepsilon = 0.05$.

The objective is to reach and stay in the set $W = [-0.06, 0.06] \times [-0.3, 0.3]$ (eventually always W).

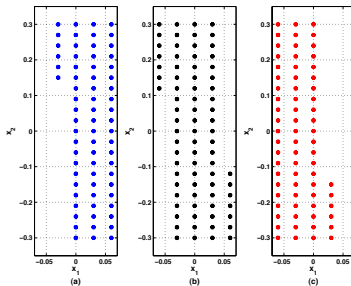
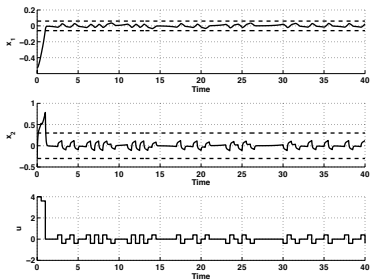
Some numbers:

Abstraction with 99981 transitions computed in 78 seconds. Less than one second to compute controller. Computations performed on a 2.4GHz Mac with 2GB of memory.

Does it really work?

Pessoa

Initial condition $(-0.5, 0.3)$.



Questions?

- Synthesis offers an alternative to existing design methodologies.
- Many challenges remain to be addressed in order to transition synthesis from academia into industry.

Questions?

- Synthesis offers an alternative to existing design methodologies.
- Many challenges remain to be addressed in order to transition synthesis from academia into industry.

The work described in this talk was the result of:

- the dedication of my current and former students and postdocs: **Anna Davitian, Manuel Mazo Jr.**, Giordano Pola, **Majid Zamani**;
- the inspiring discussions with my collaborators: **Antoine Girard, Rupak Majumdar, and George Pappas**;
- the financial support from the National Science Foundation.

For papers and more information:

<http://www.cyphylab.ee.ucla.edu/>

<http://www.ee.ucla.edu/~tabuada>