Controller Synthesis using Symbolic Models

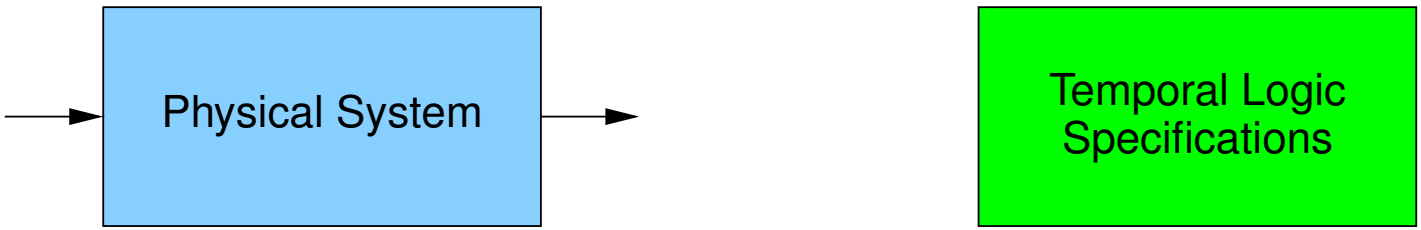
Antoine Girard

Laboratoire Jean Kuntzmann
Université Joseph Fourier, Grenoble, France

Correct-by-design embedded control software synthesis
Atlanta, USA, December 14 2010

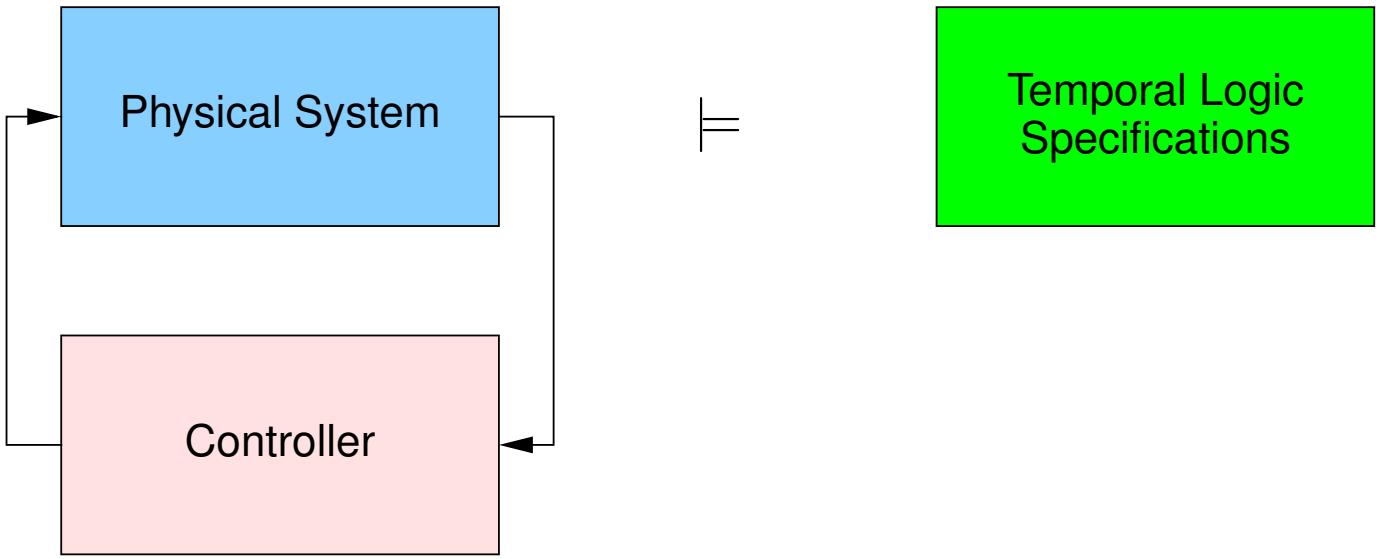
Motivation

Controller synthesis from high level (temporal logic) specifications:



Motivation

Controller synthesis from high level (temporal logic) specifications:



Temporal Logic Specifications

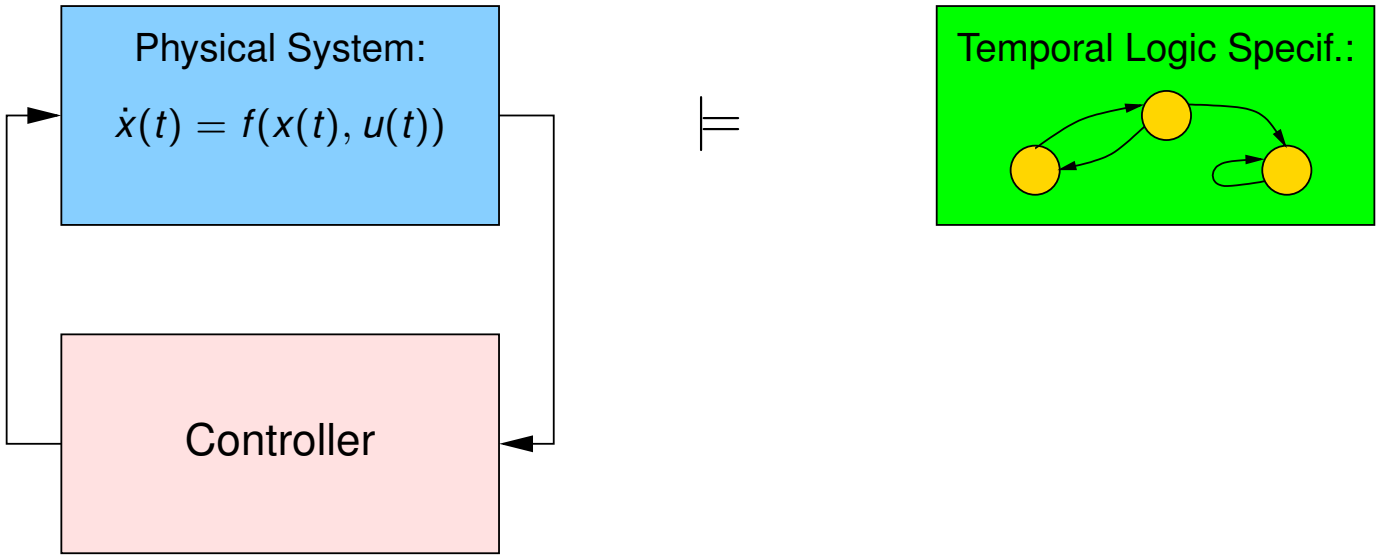
- Linear temporal logic (LTL): wide variety of properties.

Safety	$\Box S$	(*Always S*)
Reachability	$\Diamond T$	(*Eventually T*)
Stability	$\Diamond(\Box T)$	
Recurrence	$\Box(\Diamond T)$	
Sequencing	$\Diamond(T_1 \wedge \Diamond T_2)$	
Coverage	$\Diamond T_1 \wedge \Diamond T_2$	
Fault recovery	$\Box(F \implies \Diamond R)$	

- LTL formula admits an equivalent (Büchi) automaton.

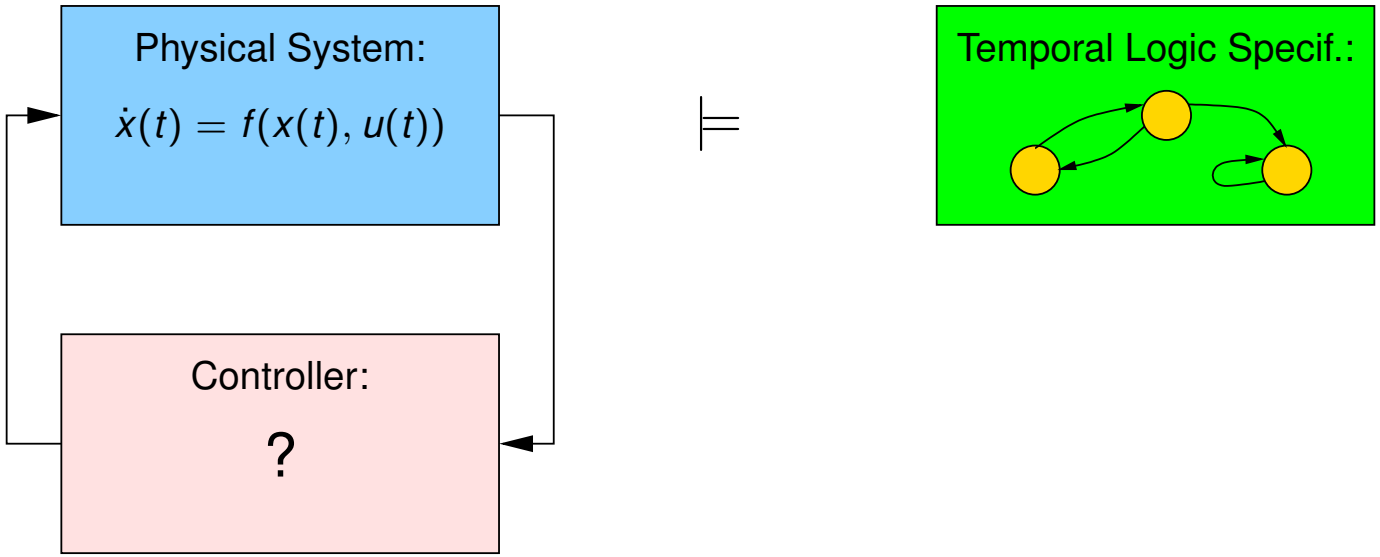
Motivation

Controller synthesis from high level (temporal logic) specifications:



Motivation

Controller synthesis from high level (temporal logic) specifications:



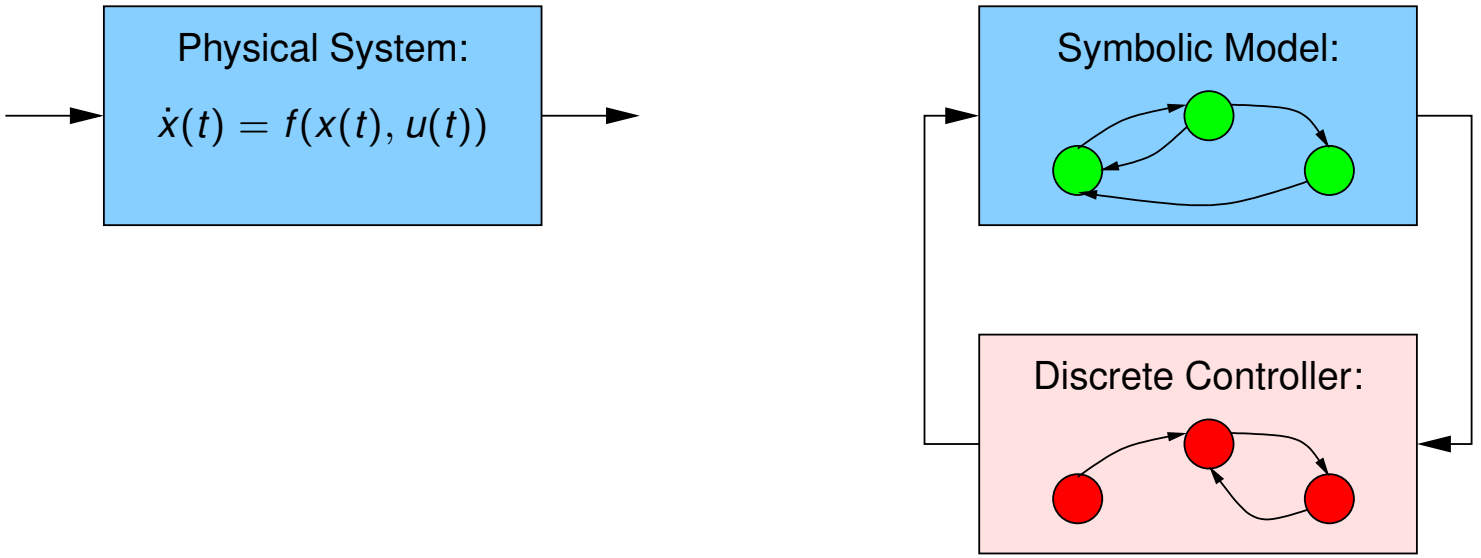
The problem is hard because the model and the specification are heterogeneous.

Symbolic Approach to Control Synthesis

Symbolic (*discrete*) model that is approximately equivalent to the (*continuous*) dynamics of the physical system:

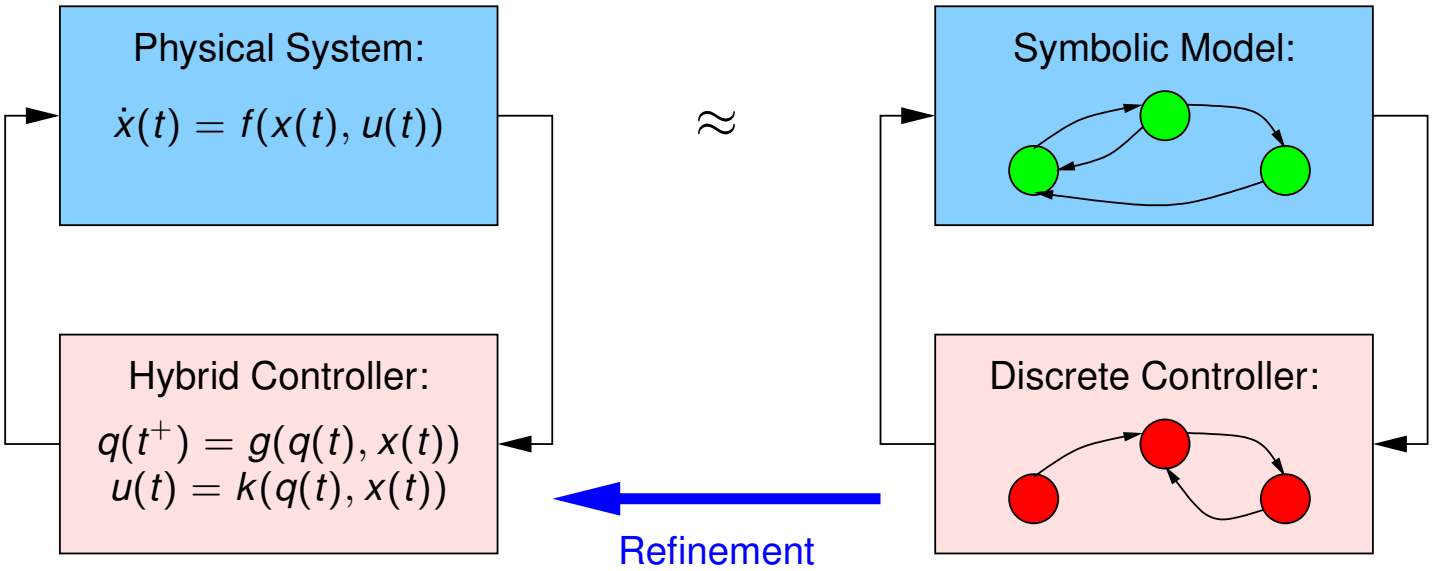
Symbolic Approach to Control Synthesis

Symbolic (*discrete*) model that is approximately equivalent to the (*continuous*) dynamics of the physical system:



Symbolic Approach to Control Synthesis

Symbolic (*discrete*) model that is approximately equivalent to the (*continuous*) dynamics of the physical system:



Symbolic Approach to Control Synthesis

- A three step approach to controller synthesis:
  1. Computation of a symbolic abstraction of the physical system.
  2. Discrete controller synthesis for the symbolic abstraction.
  3. Hybrid controller synthesis via discrete controller refinement.

Symbolic Approach to Control Synthesis

- A three step approach to controller synthesis:
  1. Computation of a symbolic abstraction of the physical system.
  2. Discrete controller synthesis for the symbolic abstraction.
  3. Hybrid controller synthesis via discrete controller refinement.
- This allows us to use discrete controller synthesis techniques:
  - Use supervisory control, algorithmic game theory...
  - Modular approaches for rich specifications.
  - Possibility of optimizing some performance criteria to choose among admissible controllers: dynamic programming, shortest path algorithms, branch and bound, α-β pruning...

Outline of the Talk

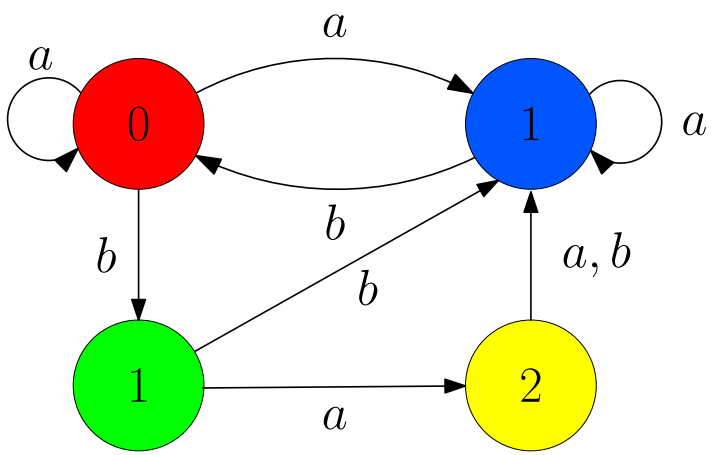
1. Approximation relationships for discrete and continuous systems
   - Approximate bisimulation.
   - Symbolic abstractions of switched systems.
2. Controller synthesis using approximately bisimilar abstractions
   - Generic technique for controller refinement.
   - Synthesis for safety specifications.
   - Synthesis for reachability specifications under time optimization.

Transition Systems

Definition

A *transition system* is a tuple $T = (X, U, \longrightarrow, Y, H)$ where

- X is a (discrete or continuous) set of states;
- U is a (discrete or continuous) set of inputs;
- $\longrightarrow \subseteq X \times U \times X$ is a transition relation;
- Y is a (discrete or continuous) set of outputs;
- $H : X \to Y$ is an ouput map.



$$
\begin{aligned}
X &= \{red, blue, green, yellow\} \\
U &= \{a, b\} \\
Y &= \{0, 1, 2\}
\end{aligned}
$$

Transition Systems

- A *trajectory* of the transition system T is a finite sequence of transitions:

$$s = x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} \dots \xrightarrow{u_{N-2}} x_{N-1} \xrightarrow{u_{N-1}} x_N.$$

- The associated *observed trajectory* is

$$o = y_0, y_1, \dots, y_{N-1}, y_N \text{ where } y_k = H(x_k),\ \forall k \in \{0, \dots, N\}.$$

- The transition system is said to be *deterministic* if for all $x \in X$, $u \in U$, $\mathrm{Post}_u(x)$ has zero or one element.

- The transition system is said to be *discrete* or *symbolic* if X and U are countable or finite. Otherwise, it is said to be *uncountable*.

Approximate Bisimulation

Let $T_i = (X_i, U, \xrightarrow[i]{}, Y, H_i)$, $i \in \{1, 2\}$, be transition systems with a common set of inputs U and outputs Y equipped with a metric d.

Definition

Let $\varepsilon \in \mathbb{R}^+$, $R \subseteq X_1 \times X_2$ is an *ε-approximate bisimulation relation* if for all $(x_1, x_2) \in R$:

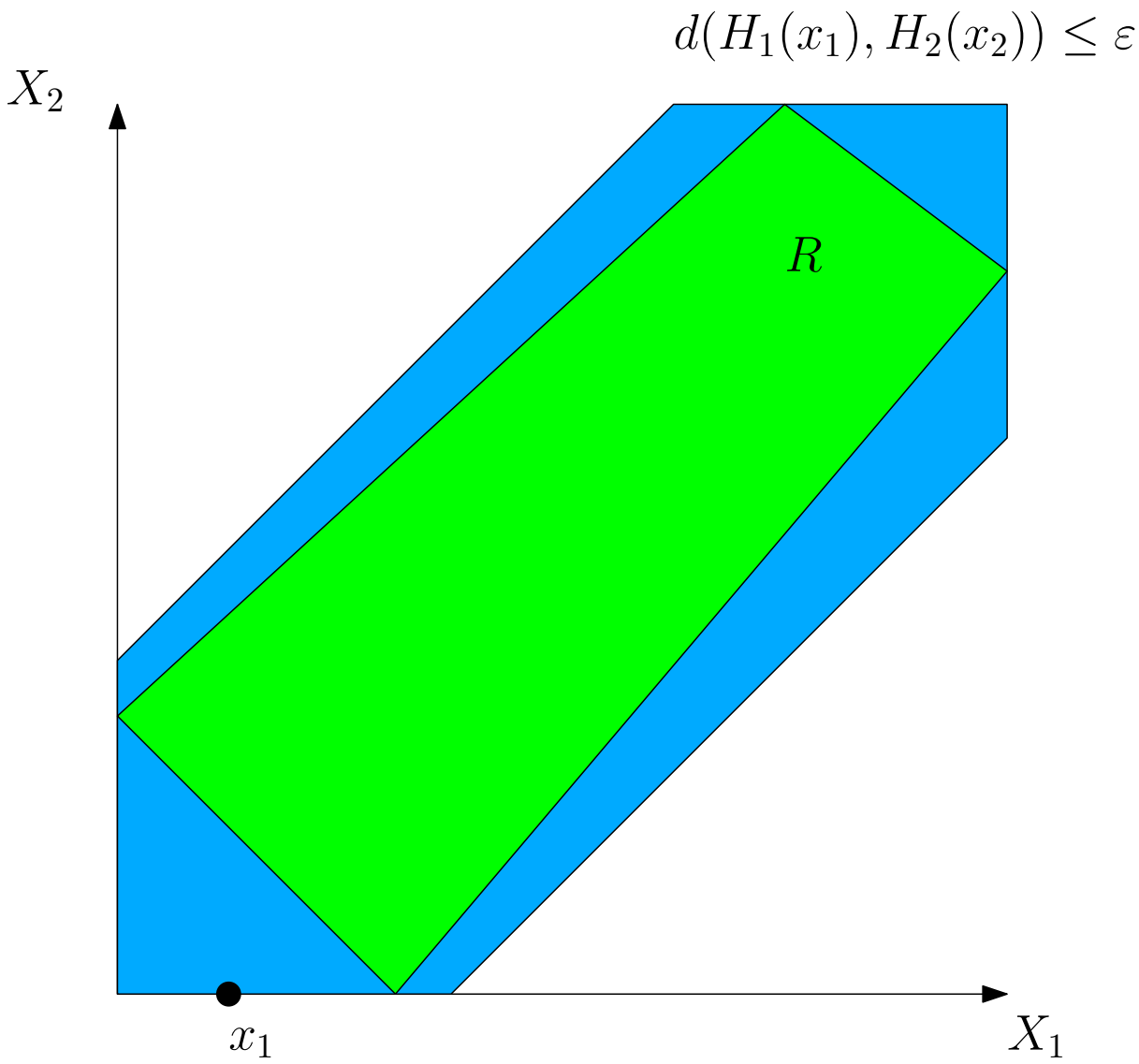
1. $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$;
2. $\forall x_1 \xrightarrow[1]{u} x_1'$, $\exists x_2 \xrightarrow[2]{u} x_2'$, such that $(x_1', x_2') \in R$;
3. $\forall x_2 \xrightarrow[2]{u} x_2'$, $\exists x_1 \xrightarrow[1]{u} x_1'$, such that $(x_1', x_2') \in R$.

Definition

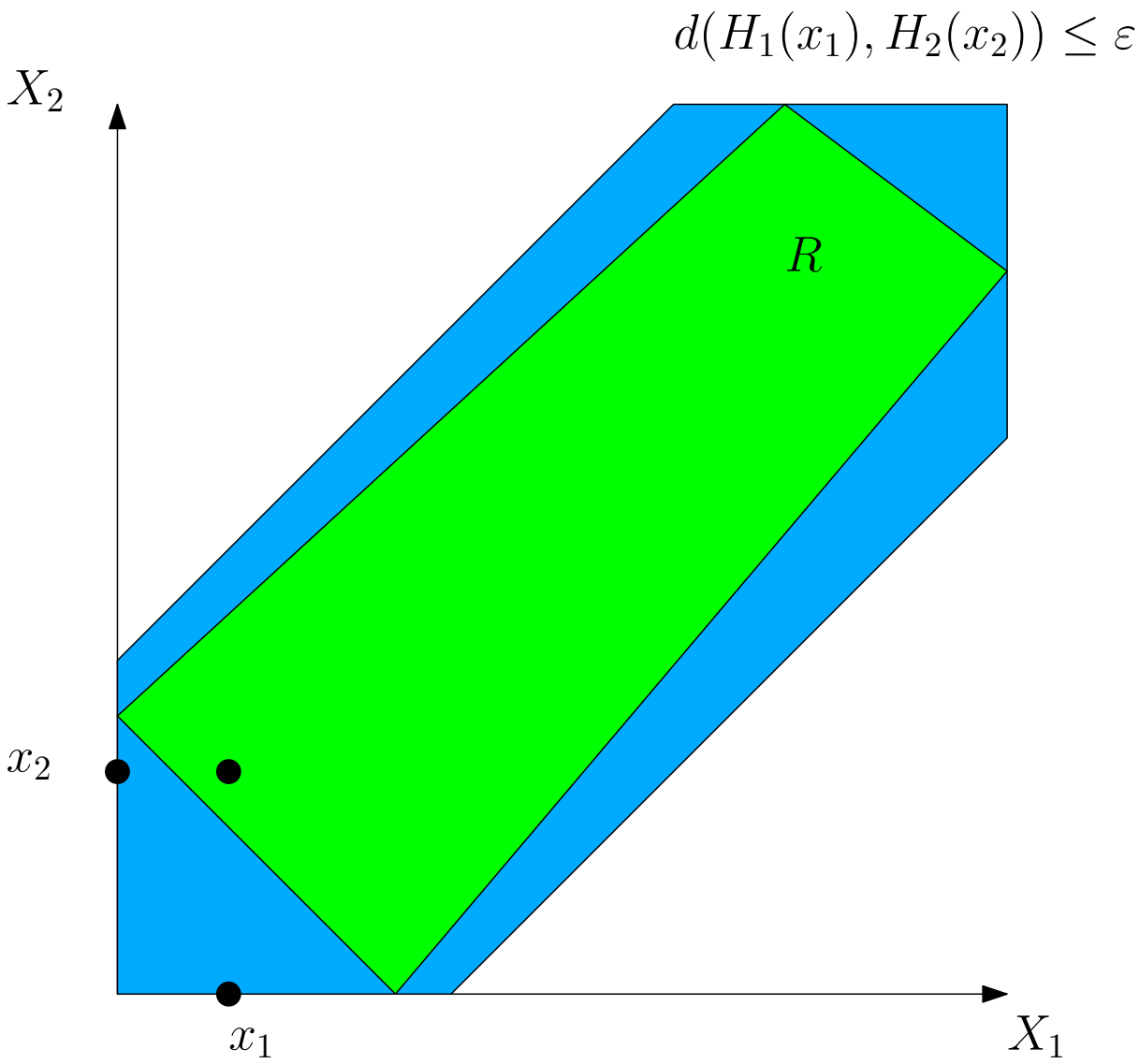
T_1 and T_2 are *ε-approximately bisimilar* ($T_1 \sim_\varepsilon T_2$) if :

1. For all $x_1 \in X_1$, there exists $x_2 \in X_2$, such that $(x_1, x_2) \in R$;
2. For all $x_2 \in X_2$, there exists $x_1 \in X_1$, such that $(x_1, x_2) \in R$.

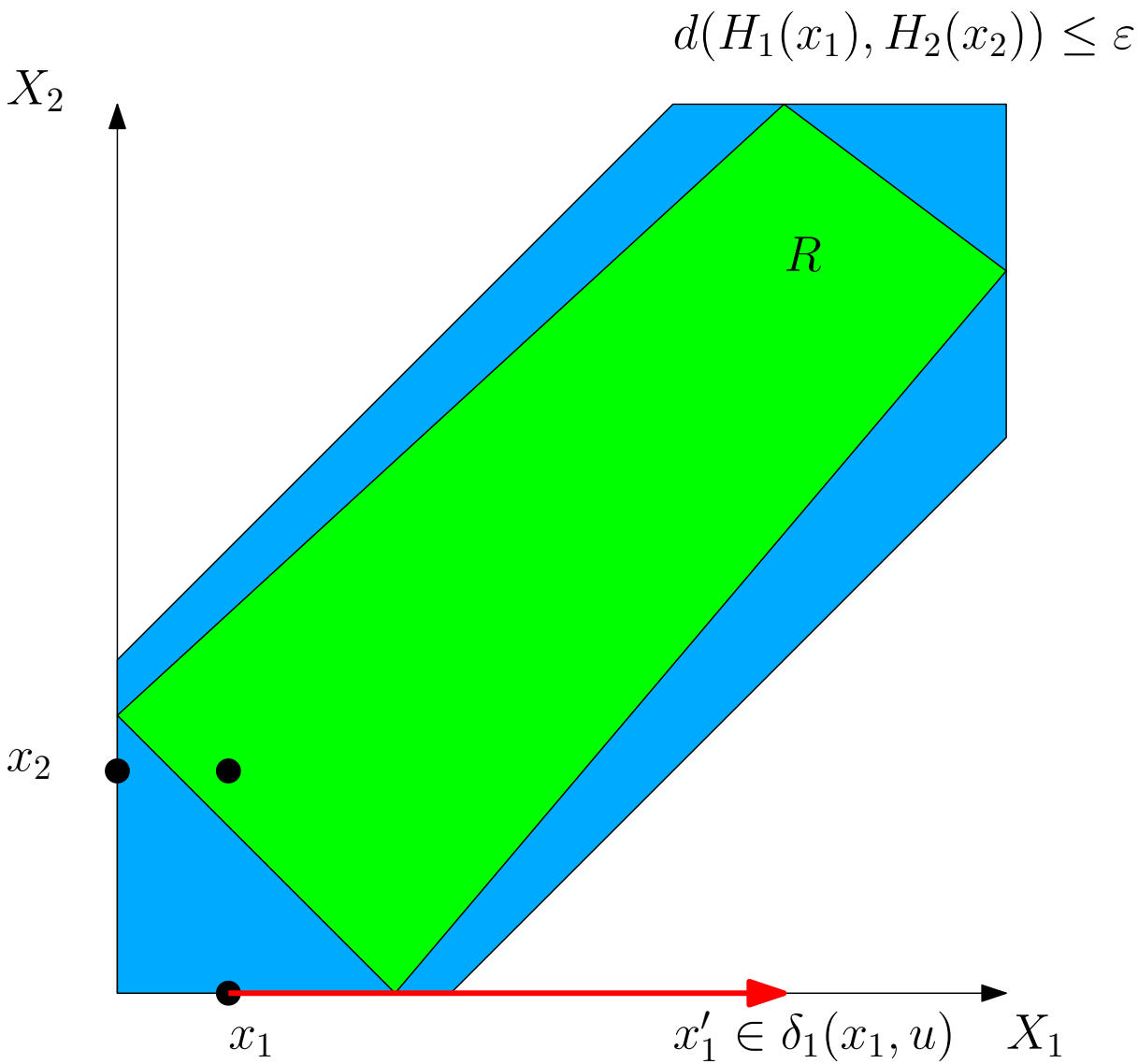
Approximate Bisimulation



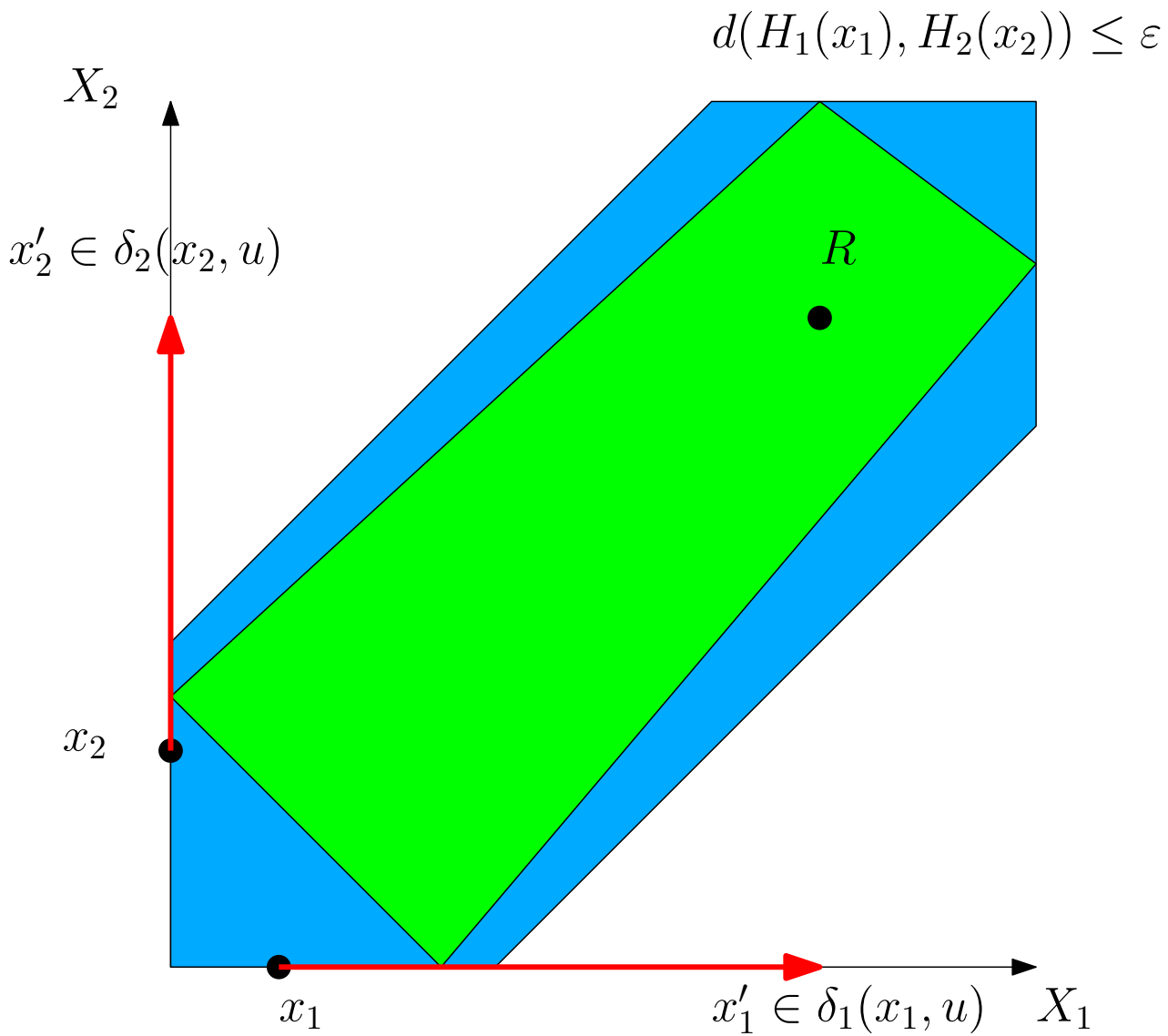
Approximate Bisimulation



Approximate Bisimulation



Approximate Bisimulation



Approximate Bisimulation

- The previous definition slightly differs from that presented in the previous lecture:
  $\Longrightarrow$ Matching transitions must have the same inputs.
- This assumption will be useful for discrete controller refinement.
- It can be relaxed if we consider deterministic systems, or for non-deterministic systems using the notion of alternating approximate bisimulation.

Approximate Bisimulation

Proposition

If $T_1 \sim_\varepsilon T_2$, *then for all trajectories of* T_1, $x_0^1 \xrightarrow[1]{u_0} \ldots \xrightarrow[1]{u_{N-1}} x_N^1$, *there exists a trajectory of* T_2, $x_0^2 \xrightarrow[2]{u_0} \ldots \xrightarrow[2]{u_{N-1}} x_N^2$, *such that*

$$\forall k \in \{0, \ldots, N\},\ (x_k^1, x_k^2) \in R.$$

The associated observed trajectories $y_0^1, \ldots, y_N^1$ *and* $y_0^2, \ldots, y_N^2$ *satisfy*

$$\forall k \in \{0, \ldots, N\},\ d(y_k^1, y_k^2) \leq \varepsilon.$$

For $\varepsilon = 0$, we recover the usual notion of bisimulation relation for studying "exact" equivalence of transition systems.

Outline of the Talk

1. Approximation relationships for discrete and continuous systems
   - Approximate bisimulation.
   - Symbolic abstractions of switched systems.
2. Controller synthesis using approximately bisimilar abstractions
   - Generic technique for controller refinement.
   - Synthesis for safety specifications.
   - Synthesis for reachability specifications under time optimization.

Switched Systems

Definition

A *switched system* is a tuple $\Sigma = (\mathbb{R}^n, P, \mathcal{F})$ where:

- $\mathbb{R}^n$ is the state space;
- $P = \{1, \ldots, m\}$ is the finite set of modes;
- $F = \{f_p : \mathbb{R}^n \to \mathbb{R}^n | \ p \in P\}$ is the collection of vector fields.

- For a switching signal $\mathbf{p} : \mathbb{R}^+ \to P$, initial state $x \in \mathbb{R}^n$, $\mathbf{x}(t, x, \mathbf{p})$ denotes the trajectory of Σ given by:

$$\dot{\mathbf{x}}(t) = f_{\mathbf{p}(t)}(\mathbf{x}(t)), \ \mathbf{x}(0) = x.$$

- For $p \in P$, $\mathbf{x}(t, x, p)$ denotes the trajectory of Σ associated to the constant switching signal $\mathbf{p}(t) = p$.

Switched Systems as Transition Systems

- Consider a switched system $\Sigma = (\mathbb{R}^n, P, \mathcal{F})$ and a time sampling parameter $\tau > 0$.
- Let $T_\tau(\Sigma)$ be the transition system where:
  - the set of states is $X = \mathbb{R}^n$;
  - the set of inputs is $U = P$;
  - the transition relation is given by

$$x \xrightarrow{p} x' \iff x' = \mathbf{x}(\tau, x, p);$$

  - the set of outputs is $Y = \mathbb{R}^n$;
  - the output map H is the identity map over $\mathbb{R}^n$.
- The transition system $T_\tau(\Sigma)$ is uncountable, can we compute a symbolic abstraction?

Computation of the Symbolic Abstraction

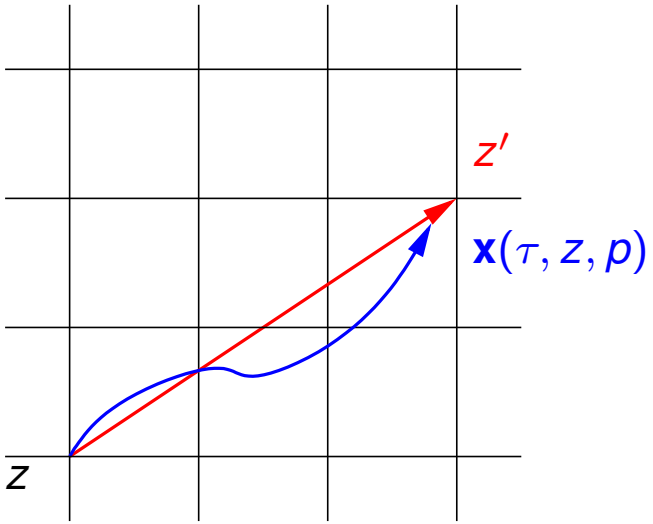
- We start by approximating the set of states $\mathbb{R}^n$ by:

$$[\mathbb{R}^n]_\eta = \left\{ z \in \mathbb{R}^n \;\middle|\; z_i = k_i \frac{2\eta}{\sqrt{n}},\ k_i \in \mathbb{Z},\ i = 1, ..., n \right\},$$

  where $\eta > 0$ is a state sampling parameter:

$$\forall x \in \mathbb{R}^n,\ \exists z \in [\mathbb{R}^n]_\eta,\ \|x - z\| \leq \eta.$$

- Approximation of the transition relation = "rounding":



Computation of the Symbolic Abstraction

- We define the transition system $T_{\tau,\eta}(\Sigma)$ where :
  - the set of states is $X = [\mathbb{R}^n]_\eta$;
  - the set of inputs is $U = P$;
  - the transition relation is given by

$$z \xrightarrow{p} z' \iff z' = \arg \min_{q \in [\mathbb{R}^n]_\eta} (\|\mathbf{x}(\tau, z, p) - q\|) ;$$

  - the set of outputs is $Y = \mathbb{R}^n$;
  - the output map is given by $H(z) = z \in \mathbb{R}^n$.
- The transition system $T_{\tau,\eta}(\Sigma)$ is discrete and deterministic.
- Are $T_\tau(\Sigma)$ and $T_{\tau,\eta}(\Sigma)$ approximately bisimilar ?

Computation of the Symbolic Abstraction

- We define the transition system $T_{\tau,\eta}(\Sigma)$ where :
  - the set of states is $X = [\mathbb{R}^n]_\eta$;
  - the set of inputs is $U = P$;
  - the transition relation is given by

$$z \xrightarrow{p} z' \iff z' = \arg \min_{q \in [\mathbb{R}^n]_\eta} \left(\|\mathbf{x}(\tau, z, p) - q\| \right);$$

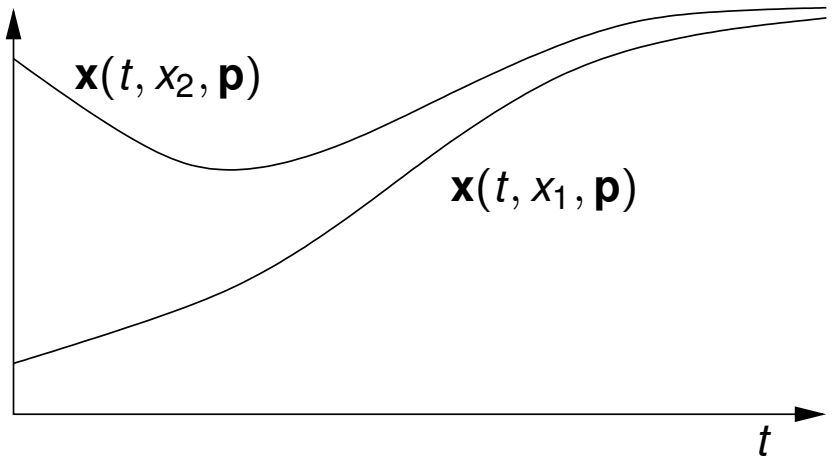
  - the set of outputs is $Y = \mathbb{R}^n$;
  - the output map is given by $H(z) = z \in \mathbb{R}^n$.
- The transition system $T_{\tau,\eta}(\Sigma)$ is discrete and deterministic.
- Are $T_\tau(\Sigma)$ and $T_{\tau,\eta}(\Sigma)$ approximately bisimilar ?
- Yes, if switched system Σ is *incrementally stable*.

Incremental Stability

Definition

The switched system Σ is *incrementally globally uniformly asymptotically stable* (δ-GUAS) if there exists a $\mathcal{KL}$ function β such that for all initial conditions $x_1, x_2 \in \mathbb{R}^n$, for all switching signals $\mathbf{p} : \mathbb{R}^+ \to P$, for all $t \in \mathbb{R}^+$:

$$\|\mathbf{x}(t, x_1, \mathbf{p}) - \mathbf{x}(t, x_2, \mathbf{p})\| \leq \beta(\|x_1 - x_2\|, t) \xrightarrow[t \to +\infty]{} 0.$$



δ-GAS Lyapunov Functions

Definition

$V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ is a *common δ-GUAS Lyapunov function* for Σ if there exist $\mathcal{K}_\infty$ functions $\underline{\alpha}$, $\overline{\alpha}$ and $\kappa \in \mathbb{R}^+$ such that for all $x_1, x_2 \in \mathbb{R}^n$:

$$\underline{\alpha}(\|x_1 - x_2\|) \leq V(x_1, x_2) \leq \overline{\alpha}(\|x_1 - x_2\|),$$

$$\forall p \in P, \ \frac{\partial V}{\partial x_1}(x_1, x_2) f_p(x_1) + \frac{\partial V}{\partial x_2}(x_1, x_2) f_p(x_2) \leq -\kappa V(x_1, x_2).$$

Theorem

If there exists a common δ-GUAS Lyapunov function, then Σ is δ-GUAS.

δ-GAS Lyapunov Functions

Definition

$V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ is a *common δ-GUAS Lyapunov function* for Σ if there exist $\mathcal{K}_\infty$ functions $\underline{\alpha}$, $\overline{\alpha}$ and $\kappa \in \mathbb{R}^+$ such that for all $x_1, x_2 \in \mathbb{R}^n$:

$$\underline{\alpha}(\|x_1 - x_2\|) \leq V(x_1, x_2) \leq \overline{\alpha}(\|x_1 - x_2\|),$$

$$\forall p \in P, \ \frac{\partial V}{\partial x_1}(x_1, x_2) f_p(x_1) + \frac{\partial V}{\partial x_2}(x_1, x_2) f_p(x_2) \leq -\kappa V(x_1, x_2).$$

Theorem

If there exists a common δ-GUAS Lyapunov function, then Σ is δ-GUAS.

Supplementary assumption (true if working on a compact of $\mathbb{R}^n$):
There exists a $\mathcal{K}_\infty$ function γ such that

$$\forall x_1, x_2, x_3 \in \mathbb{R}^n, \ |V(x_1, x_2) - V(x_1, x_3)| \leq \gamma(\|x_2 - x_3\|).$$

Approximation Theorem

Theorem

Let us assume that there exists $V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ *which is a common* δ*-GUAS Lyapunov function for* Σ*. Consider sampling parameters* $\tau, \eta \in \mathbb{R}^+$ *and a desired precision* $\varepsilon \in \mathbb{R}^+$*. If*

$$\eta \leq \min \left\{ \gamma^{-1} \left((1 - e^{-\kappa\tau}) \underline{\alpha}(\varepsilon) \right), \overline{\alpha}^{-1} \left(\underline{\alpha}(\varepsilon) \right) \right\}$$

then, the relation $R \subseteq \mathbb{R}^n \times [\mathbb{R}^n]_\eta$ *given by*

$$R = \{(x, z) \in \mathbb{R}^n \times [\mathbb{R}^n]_\eta \mid V(x, z) \leq \underline{\alpha}(\varepsilon)\}$$

is an ε*-approximate bisimulation relation and* $T_\tau(\Sigma) \sim_\varepsilon T_{\tau,\eta}(\Sigma)$*.*

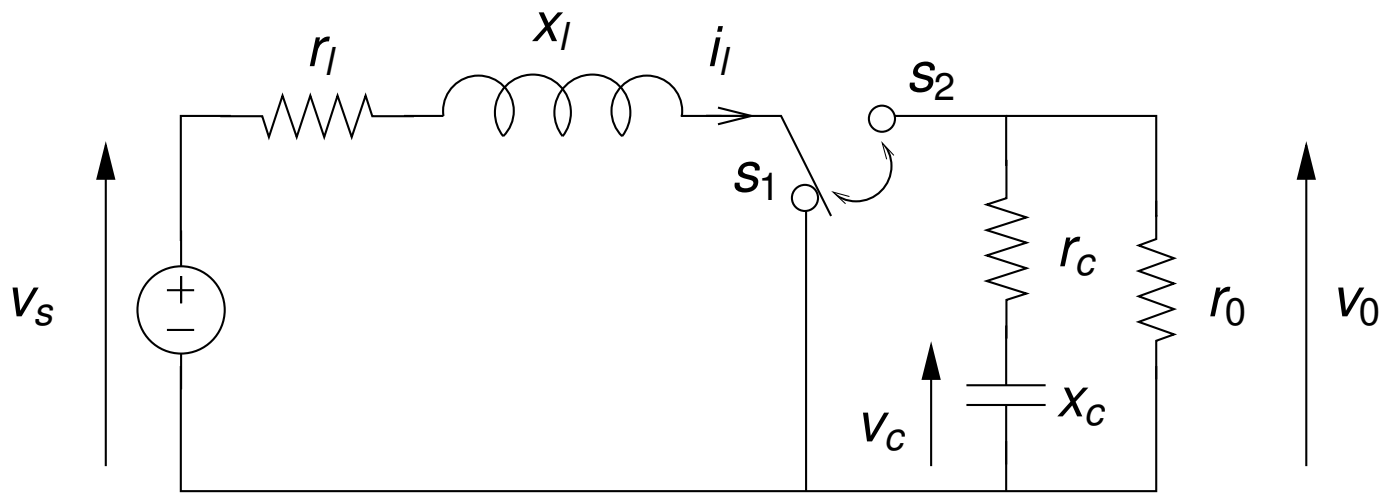
Main idea of the proof: show that accumulation of successive "rounding errors" is contained by incremental stability.

Comments on the Approximation Theorem

- Contrarily to more "traditional" partition-based abstractions, one concrete state is related to several (many) symbolic states.
- For a given time sampling parameter τ, any precision ε can be achieved by choosing appropriately the state sampling parameter η (the smaller τ or ε, the smaller η).
- If all vector fields are affine, one can search for a quadratic common δ-GUAS Lyapunov functions by solving a set of LMIs.
- For switched systems without a common δ-GUAS Lyapunov function, the result can be extended by using multiple δ-GUAS Lyapunov functions and by imposing a minimum dwell time.

Example: DC-DC Converter

- Power converter with switching control:

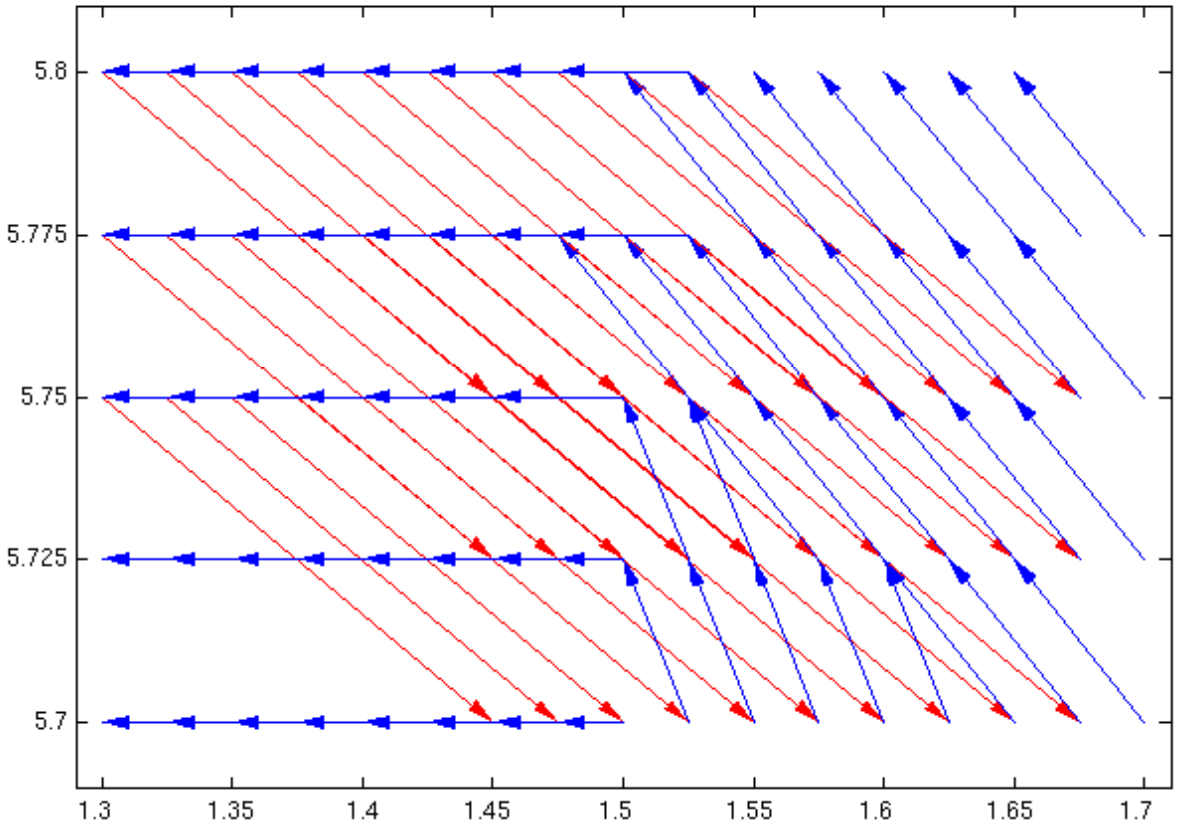


- State variable: $x(t) = [i_l(t), v_c(t)]^T$.
- System dynamics: $\dot{x}(t) = A_p x(t) + b,\ p \in \{1, 2\}$.
- Common δ-GUAS Lyapunov function of the form:

$$V(x, y) = \sqrt{(x - y)^T M (x - y)}.$$

Symbolic Abstraction of the DC-DC Converter

Symbolic abstraction: $\tau = 0.5,\ \eta = \frac{1}{40\sqrt{2}} \implies \varepsilon = 2.6.$



Outline of the Talk

1. Approximation relationships for discrete and continuous systems
   - Approximate bisimulation.
   - Symbolic abstractions of switched systems.
2. Controller synthesis using approximately bisimilar abstractions
   - Generic technique for controller refinement.
   - Synthesis for safety specifications.
   - Synthesis for reachability specifications under time optimization.

Controllers for Transition Systems

We will focus on static (memoryless) controllers:

Definition

Let $T = (X, U, \longrightarrow, Y, H)$, a *controller* for T is a map $\mathcal{S} : X \to 2^U$.
The *controlled system* is $T_{\mathcal{S}} = (X, U, \xrightarrow[\mathcal{S}]{}, Y, H)$ where the transition relation $\xrightarrow[\mathcal{S}]{}$ is given for all $x \in X$, $u \in U$, $x' \in X$ by

$$x \xrightarrow[\mathcal{S}]{u} x' \iff \left(u \in \mathcal{S}(x) \wedge x \xrightarrow{u} x' \right).$$

Controllers for Transition Systems

We will focus on static (memoryless) controllers:

Definition

Let $T = (X, U, \longrightarrow, Y, H)$, a *controller* for T is a map $\mathcal{S} : X \to 2^U$.
The *controlled system* is $T_{\mathcal{S}} = (X, U, \xrightarrow[\mathcal{S}]{}, Y, H)$ where the transition relation $\xrightarrow[\mathcal{S}]{}$ is given for all $x \in X$, $u \in U$, $x' \in X$ by

$$x \xrightarrow[\mathcal{S}]{u} x' \iff \left(u \in \mathcal{S}(x) \wedge x \xrightarrow{u} x' \right).$$

Controller refinement problem:

- Let T_1 be a transition system and T_2 an abstraction: $T_1 \sim_\varepsilon T_2$.
- Let $R \subseteq X_1 \times X_2$ denote the ε-approximate bisimulation relation between T_1 and T_2.
- Let $\mathcal{S}_2$ be a controller for the abstraction T_2.

Controllers for Transition Systems

We will focus on static (memoryless) controllers:

Definition

Let $T = (X, U, \longrightarrow, Y, H)$, a *controller* for T is a map $\mathcal{S} : X \to 2^U$.
The *controlled system* is $T_{\mathcal{S}} = (X, U, \xrightarrow[\mathcal{S}]{}, Y, H)$ where the transition relation $\xrightarrow[\mathcal{S}]{}$ is given for all $x \in X$, $u \in U$, $x' \in X$ by

$$x \xrightarrow[\mathcal{S}]{u} x' \iff \left(u \in \mathcal{S}(x) \wedge x \xrightarrow{u} x' \right).$$

Controller refinement problem:

- Let T_1 be a transition system and T_2 an abstraction: $T_1 \sim_\varepsilon T_2$.
- Let $R \subseteq X_1 \times X_2$ denote the ε-approximate bisimulation relation between T_1 and T_2.
- Let $\mathcal{S}_2$ be a controller for the abstraction T_2.
- How do we use $\mathcal{S}_2$ to control T_1 ?

Controller Refinement

We use the following control algorithm to control T_1:

- Initialization:
  1. Sense concrete state x_1.
  2. Select abstract state x_2, such that $(x_1, x_2) \in R$.
- Main loop:
  1. Select input $u \in \mathcal{S}_2(x_2)$.
  2. Let T_1 execute a transition $x_1 \xrightarrow[1]{u} x_1'$.
  3. Sense concrete state x_1'.
  4. Select abstract state $x_2 \xrightarrow[2]{u} x_2'$, such that $(x_1', x_2') \in R$.

Controller Refinement

We use the following control algorithm to control T_1:

- Initialization:
  1. Sense concrete state x_1.
  2. Select abstract state x_2, such that $(x_1, x_2) \in R$.
- Main loop:
  1. Select input $u \in \mathcal{S}_2(x_2)$.
  2. Let T_1 execute a transition $x_1 \xrightarrow[1]{u} x_1'$.
  3. Sense concrete state x_1'.
  4. Select abstract state $x_2 \xrightarrow[2]{u} x_2'$, such that $(x_1', x_2') \in R$.

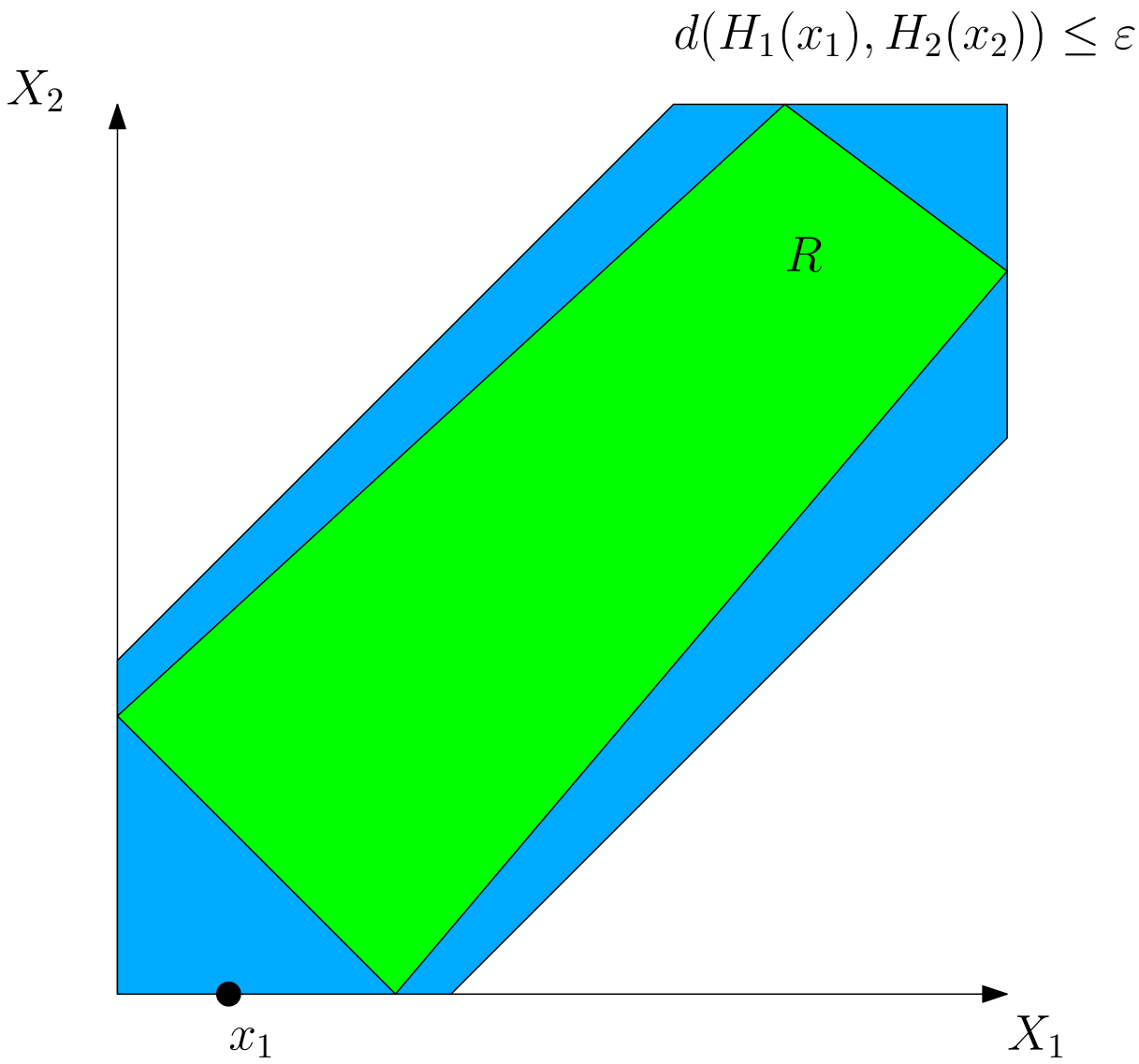
Theorem

Let $u_0, \ldots, u_{N-1}$, $x_0^1, \ldots, x_N^1$ and $x_0^2, \ldots, x_N^2$, be sequences of inputs, concrete states and abstract states generated by the control algorithm.

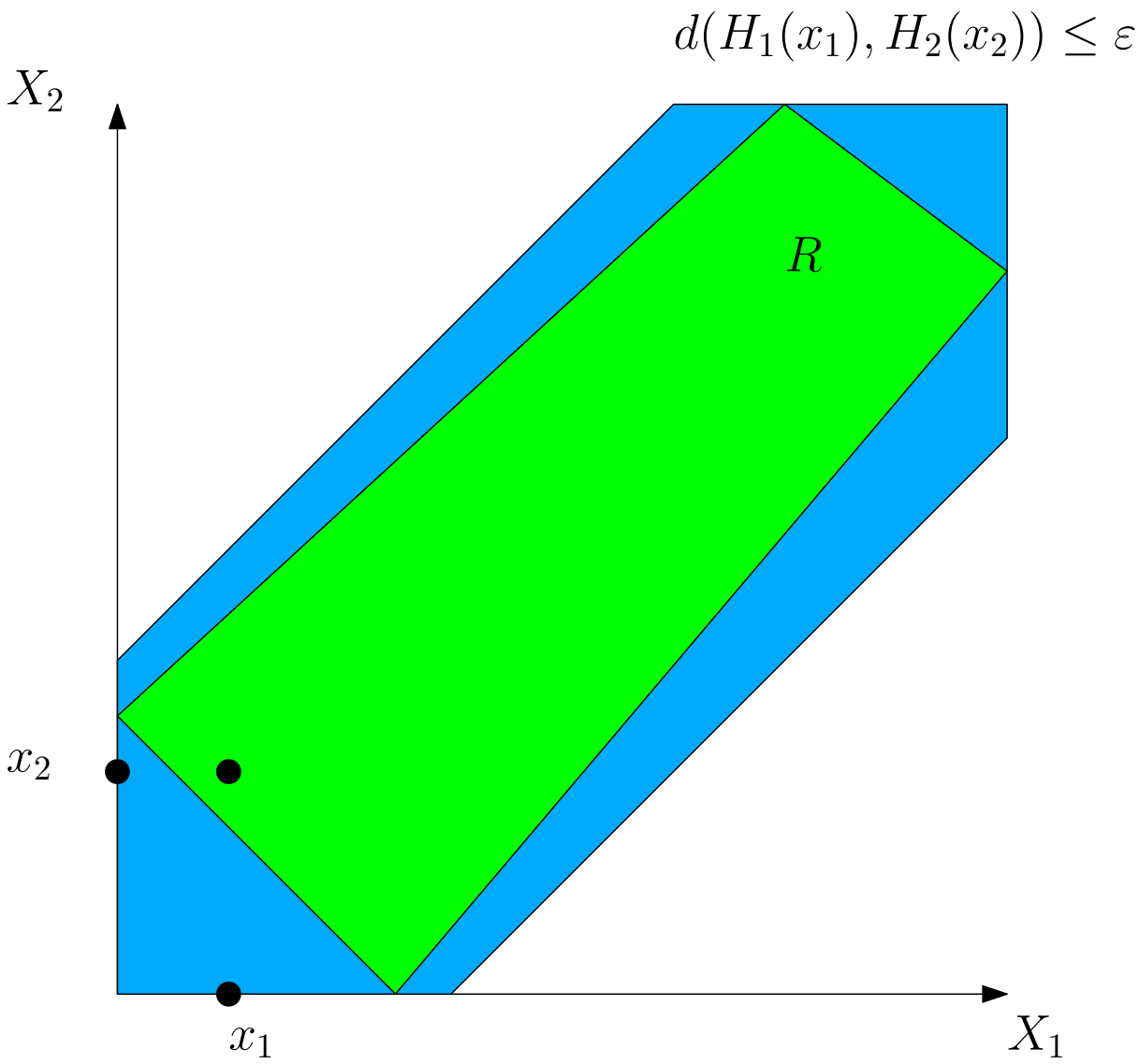
Then, $x_0^2 \xrightarrow[2]{u_0} \ldots x_{N-1}^2 \xrightarrow[2]{u_{N-1}} x_N^2$ is a trajectory of $T_{2,\mathcal{S}_2}$ and

$$\forall k \in \{0, \ldots, N\},\ d(H_1(x_k^1), H_2(x_k^2)) \leq \varepsilon.$$

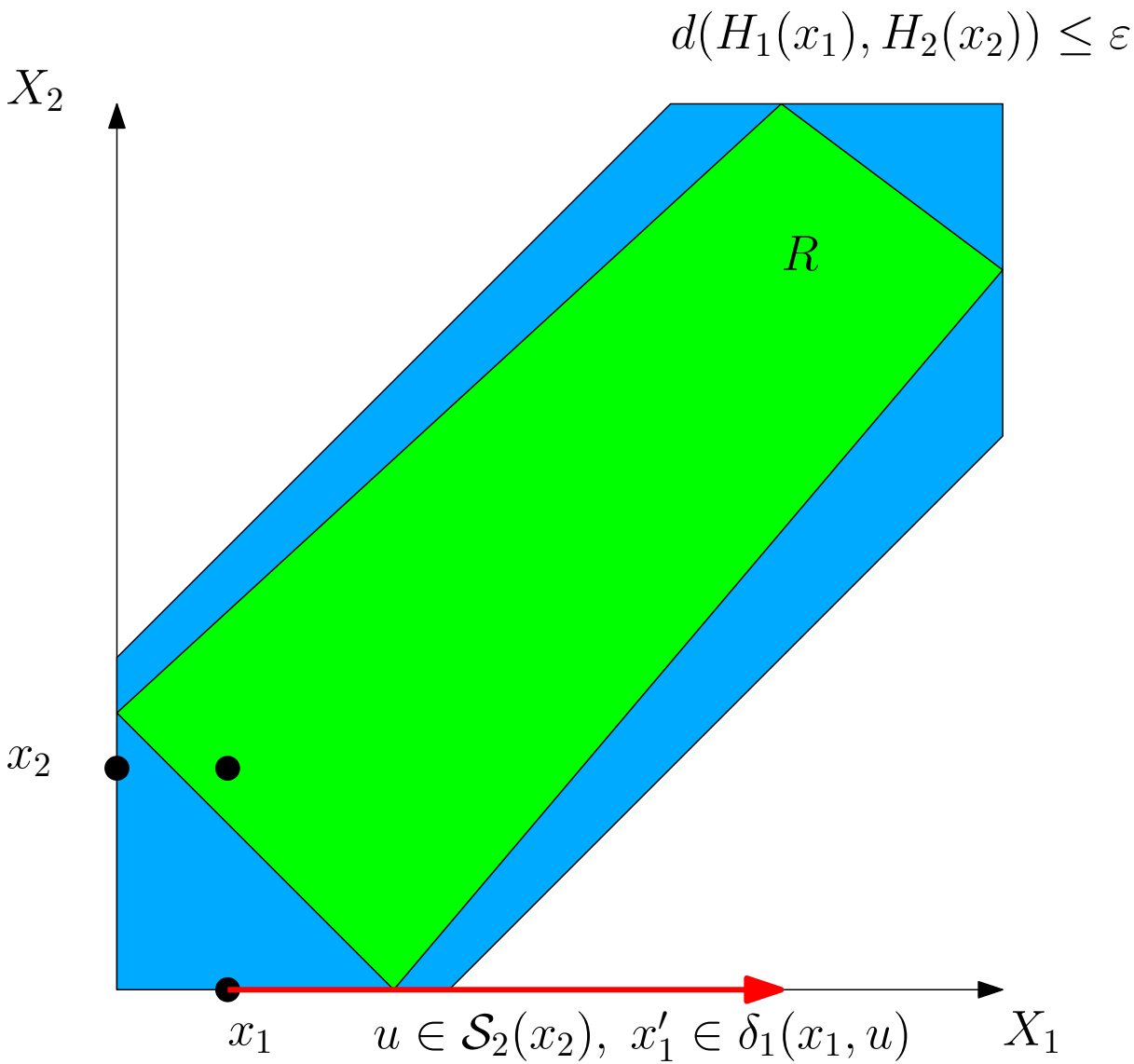
Controller Refinement



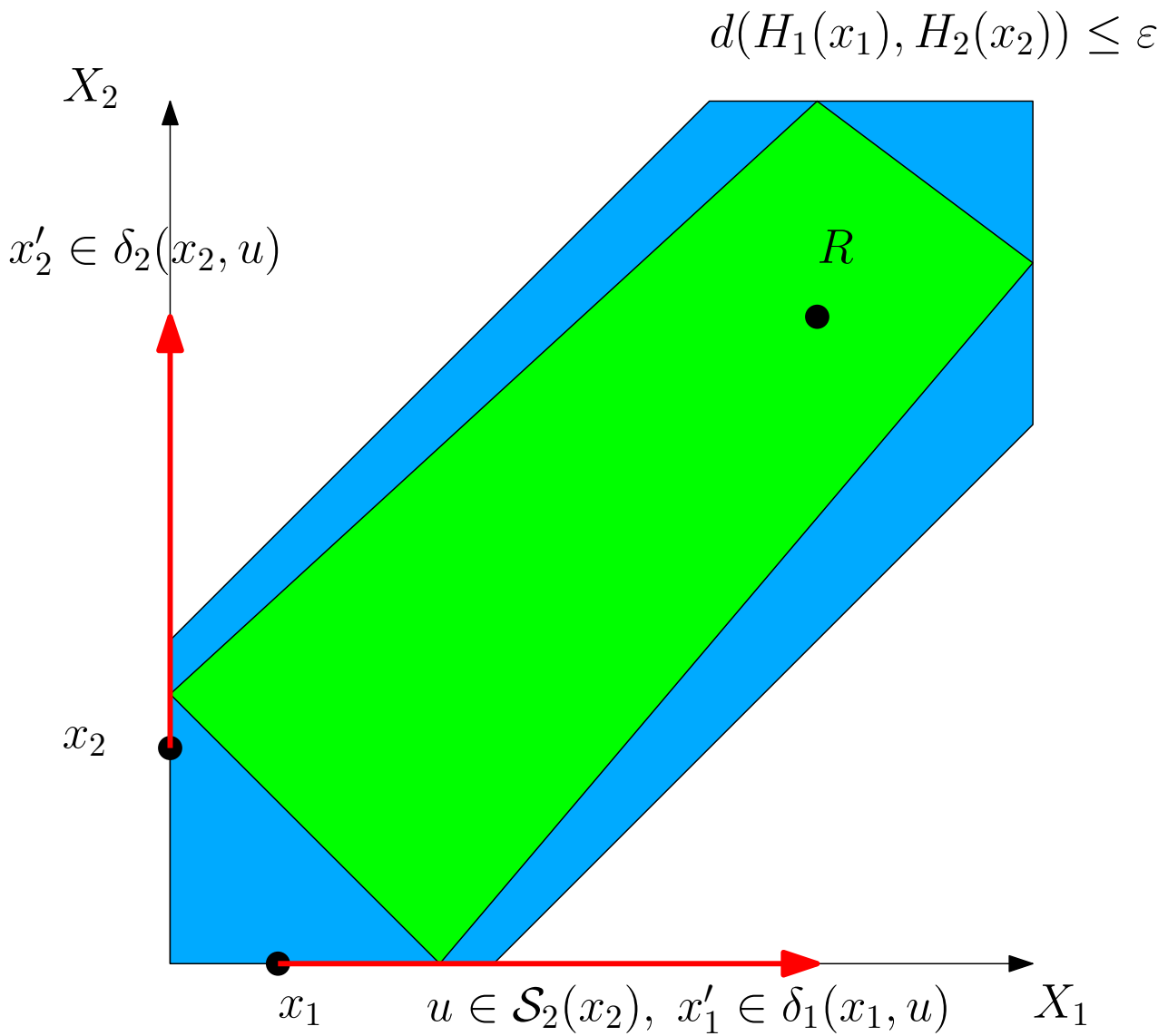
Controller Refinement



Controller Refinement



Controller Refinement



Comments on Controller Refinement

- The refinement procedure is generic:
  - It is independent from the specification of the control problem.
  - It ensures that observed trajectories of $T_{1,\mathcal{S}_1}$ are within distance ε from observed trajectories of $T_{2,\mathcal{S}_2}$.
- $\mathcal{S}_1$ is not a static controller:
  - The abstract state x_2 serves as a memory for the controller.
  - Implementation requires encoding the dynamics of T_2.
- For safety and reachability specifications:
  - It is known that static controllers are sufficient and optimal.
  - We develop specific refinement techniques to obtain a static controller $\mathcal{S}_1$.

Outline of the Talk

1. Approximation relationships for discrete and continuous systems
   - Approximate bisimulation.
   - Symbolic abstractions of switched systems.
2. Controller synthesis using approximately bisimilar abstractions
   - Generic technique for controller refinement.
   - Synthesis for safety specifications.
   - Synthesis for reachability specifications under time optimization.

Controllers for Safety Specifications

Let $T = (X, U, \longrightarrow, Y, H)$, let $Y_s \subseteq Y$ specify the safe states.

Definition

A controller $\mathcal{S}$ is *safe for specification Y_s* if, for all $x_0 \in X$ with $\mathcal{S}(x_0) \neq \emptyset$, for all trajectories of $T_{\mathcal{S}}$ starting from x_0, $x_0 \xrightarrow[\mathcal{S}]{u_0} \dots \xrightarrow[\mathcal{S}]{u_{N-1}} x_N$, the following conditions hold:

- $\forall k \in \{0, \dots, N\},\ H(x_k) \in Y_s$;
- $\mathcal{S}(x_N) \neq \emptyset$.

To compare the admissible safe controllers, we define the notion of permissivity:

Definition

Controller $\mathcal{S}_1$ is *more permissive* than controller $\mathcal{S}_2$ ($\mathcal{S}_2 \preceq \mathcal{S}_1$) if, for all $x \in X$, $\mathcal{S}_2(x) \subseteq \mathcal{S}_1(x)$.

Maximal Safe Controller

Definition

$\mathcal{S}^*$ is the *maximal safe controller* for specification Y_s if, $\mathcal{S}^*$ is safe and for all safe controllers $\mathcal{S}$, $\mathcal{S} \preceq \mathcal{S}^*$.

$\mathcal{S}^*$ can be determined by fixed point computation of F^*, the *largest controlled-invariant* of T, included in $H^{-1}(Y_s)$.

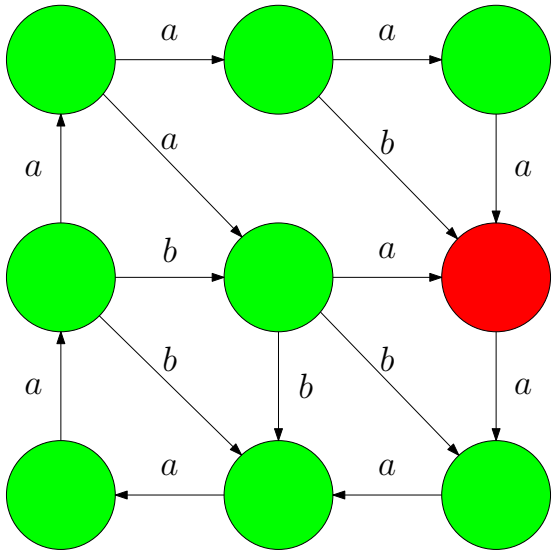
Maximal Safe Controller

Definition

$\mathcal{S}^*$ is the *maximal safe controller* for specification Y_s if, $\mathcal{S}^*$ is safe and for all safe controllers $\mathcal{S}$, $\mathcal{S} \preceq \mathcal{S}^*$.

$\mathcal{S}^*$ can be determined by fixed point computation of F^*, the *largest controlled-invariant* of T, included in $H^{-1}(Y_s)$.

A simple example:



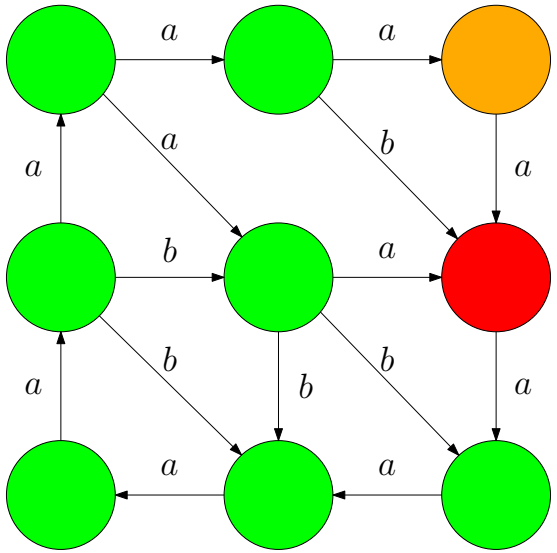
Maximal Safe Controller

Definition

$\mathcal{S}^*$ is the *maximal safe controller* for specification Y_s if, $\mathcal{S}^*$ is safe and for all safe controllers $\mathcal{S}$, $\mathcal{S} \preceq \mathcal{S}^*$.

$\mathcal{S}^*$ can be determined by fixed point computation of F^*, the *largest controlled-invariant* of T, included in $H^{-1}(Y_s)$.

A simple example:



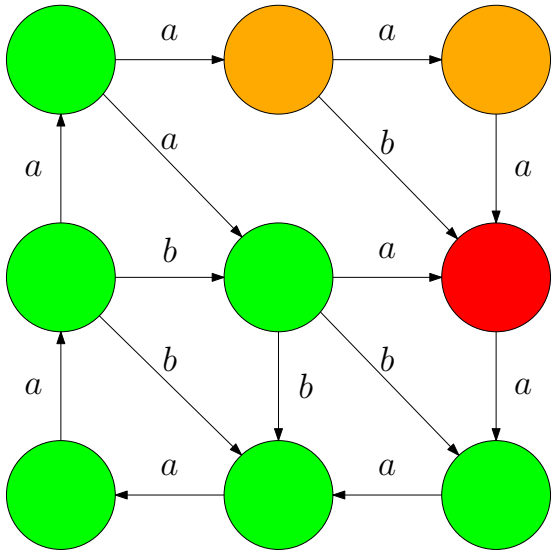
Maximal Safe Controller

Definition

$\mathcal{S}^*$ is the *maximal safe controller* for specification Y_s if, $\mathcal{S}^*$ is safe and for all safe controllers $\mathcal{S}$, $\mathcal{S} \preceq \mathcal{S}^*$.

$\mathcal{S}^*$ can be determined by fixed point computation of F^*, the *largest controlled-invariant* of T, included in $H^{-1}(Y_s)$.

A simple example:



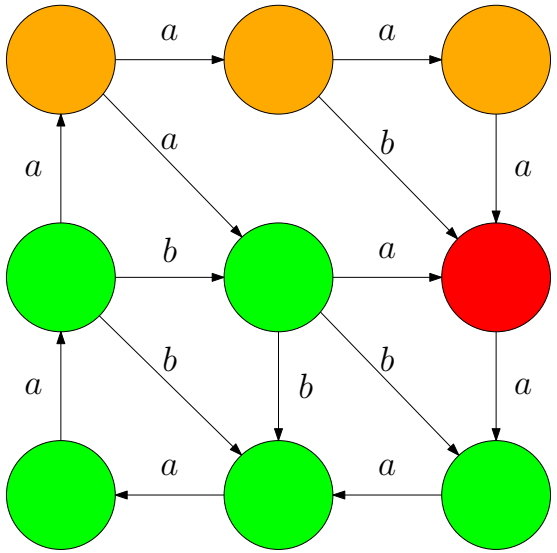
Maximal Safe Controller

Definition

$\mathcal{S}^*$ is the *maximal safe controller* for specification Y_s if, $\mathcal{S}^*$ is safe and for all safe controllers $\mathcal{S}$, $\mathcal{S} \preceq \mathcal{S}^*$.

$\mathcal{S}^*$ can be determined by fixed point computation of F^*, the *largest controlled-invariant* of T, included in $H^{-1}(Y_s)$.

A simple example:



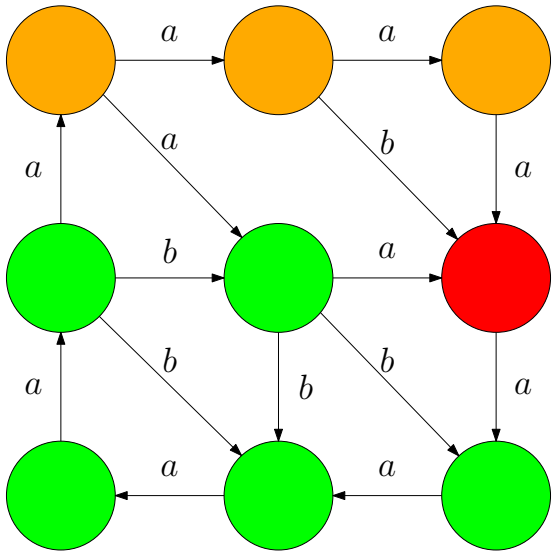
Computation of the Maximal Safe Controller

Theorem

Let $\mathcal{S}^* : X \to 2^U$ be the controller for T defined, for all $x \in X \setminus F^*$, by $\mathcal{S}^*(x) = \emptyset$, and for all $x \in F^*$ by

$$\mathcal{S}^*(x) = \left\{ u \in U | \ (\mathrm{Post}_u(x) \neq \emptyset) \wedge (\forall x' \in \mathrm{Post}_u(x), \ x' \in F^*) \right\}.$$

Then, $\mathcal{S}^*$ is the maximal safe controller for the specification Y_s.



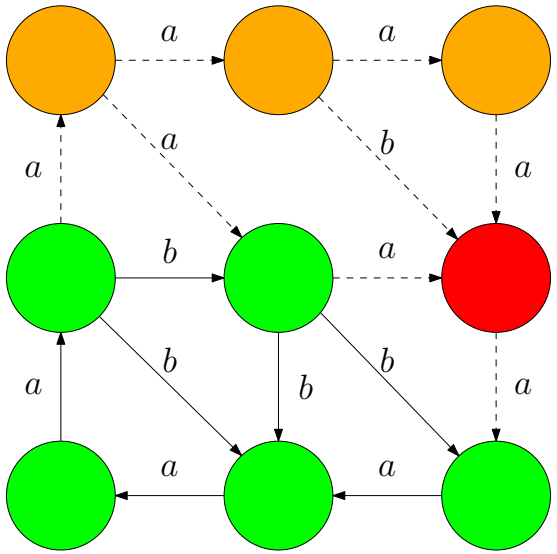
Computation of the Maximal Safe Controller

Theorem

Let $\mathcal{S}^* : X \to 2^U$ be the controller for T defined, for all $x \in X \setminus F^*$, by $\mathcal{S}^*(x) = \emptyset$, and for all $x \in F^*$ by

$$\mathcal{S}^*(x) = \left\{ u \in U | \ (\mathrm{Post}_u(x) \neq \emptyset) \wedge (\forall x' \in \mathrm{Post}_u(x), \ x' \in F^*) \right\}.$$

Then, $\mathcal{S}^*$ is the maximal safe controller for the specification Y_s.



Safe Controller Synthesis via Symbolic Abstractions

Maximal safe controllers are easy to compute for symbolic abstractions... We need a controller refinement procedure!

Safe Controller Synthesis via Symbolic Abstractions

Maximal safe controllers are easy to compute for symbolic abstractions... We need a controller refinement procedure!

Definition

Let $Y' \subseteq Y$ and $\varphi \geq 0$. The *φ-contraction* of Y' is the subset of Y is $C_\varphi(Y') = \{y \in Y' | \, \forall y' \in Y, d(y, y') \leq \varphi \implies y' \in Y'\}$.

Theorem

Let $T_1 \sim_\varepsilon T_2$, let $R \subseteq X_1 \times X_2$ denote the ε-approximate bisimulation relation between T_1 and T_2. Let $\mathcal{S}^*_{2,\varepsilon}$ be the maximal safe controller for T_2 for the specification $C_\varepsilon(Y_s)$. Let $\mathcal{S}_1$ be the controller for T_1 given by

$$\forall x_1 \in X_1, \; \mathcal{S}_1(x_1) = \bigcup_{x_2 \in R(x_1)} \mathcal{S}^*_{2,\varepsilon}(x_2)$$

where $x_2 \in R(x_1)$ iff $(x_1, x_2) \in R$. Then, $\mathcal{S}_1$ is safe for specification Y_s.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.
  - Then, $H_1(x_1) \in Y_s$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.
  - Then, $H_1(x_1) \in Y_s$.
- Let $u \in \mathcal{S}_1(x_1)$, $x_1 \xrightarrow[1]{u} x_1'$:

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.
  - Then, $H_1(x_1) \in Y_s$.
- Let $u \in \mathcal{S}_1(x_1)$, $x_1 \xrightarrow[1]{u} x'_1$:
  - There exists $x_2 \in R(x_1)$ such that $u \in \mathcal{S}^*_{2,\varepsilon}(x_2)$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.
  - Then, $H_1(x_1) \in Y_s$.
- Let $u \in \mathcal{S}_1(x_1)$, $x_1 \xrightarrow[1]{u} x'_1$:
  - There exists $x_2 \in R(x_1)$ such that $u \in \mathcal{S}^*_{2,\varepsilon}(x_2)$.
  - $x_2 \in R(x_1)$ gives that there exists $x_2 \xrightarrow[2]{u} x'_2$ such that $x'_2 \in R(x'_1)$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.
  - Then, $H_1(x_1) \in Y_s$.
- Let $u \in \mathcal{S}_1(x_1)$, $x_1 \xrightarrow[1]{u} x_1'$:
  - There exists $x_2 \in R(x_1)$ such that $u \in \mathcal{S}^*_{2,\varepsilon}(x_2)$.
  - $x_2 \in R(x_1)$ gives that there exists $x_2 \xrightarrow[2]{u} x_2'$ such that $x_2' \in R(x_1')$.
  - $u \in \mathcal{S}^*_{2,\varepsilon}(x_2)$ gives $\mathcal{S}^*_{2,\varepsilon}(x_2') \neq \emptyset$.

Sketch of the Proof

Induction step:

- Let $x_1 \in X_1$, such that $\mathcal{S}_1(x_1) \neq \emptyset$:
  - There exists $x_2 \in R(x_1)$ such that $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$.
  - $\mathcal{S}^*_{2,\varepsilon}(x_2) \neq \emptyset$ gives $H_2(x_2) \in C_\varepsilon(Y_s)$.
  - $x_2 \in R(x_1)$ gives $d(H_1(x_1), H_2(x_2)) \leq \varepsilon$.
  - Then, $H_1(x_1) \in Y_s$.
- Let $u \in \mathcal{S}_1(x_1)$, $x_1 \xrightarrow[1]{u} x'_1$:
  - There exists $x_2 \in R(x_1)$ such that $u \in \mathcal{S}^*_{2,\varepsilon}(x_2)$.
  - $x_2 \in R(x_1)$ gives that there exists $x_2 \xrightarrow[2]{u} x'_2$ such that $x'_2 \in R(x'_1)$.
  - $u \in \mathcal{S}^*_{2,\varepsilon}(x_2)$ gives $\mathcal{S}^*_{2,\varepsilon}(x'_2) \neq \emptyset$.
  - Then, $\mathcal{S}_1(x'_1) \neq \emptyset$.

Distance to the Maximal Safe Controller

Theorem

Let $\mathcal{S}_1$ be the safe controller for T_1 for specification Y_s defined in the previous theorem. Let $\mathcal{S}_1^*$ and $\mathcal{S}_{1,2\varepsilon}^*$ be the maximal safe controllers for T_1 for specifications Y_s and $C_{2\varepsilon}(Y_s)$, respectively. Then,

$$\mathcal{S}_{1,2\varepsilon}^* \preceq \mathcal{S}_1 \preceq \mathcal{S}_1^*.$$

Distance to the Maximal Safe Controller

Theorem

Let $\mathcal{S}_1$ be the safe controller for T_1 for specification Y_s defined in the previous theorem. Let $\mathcal{S}_1^*$ and $\mathcal{S}_{1,2\varepsilon}^*$ be the maximal safe controllers for T_1 for specifications Y_s and $C_{2\varepsilon}(Y_s)$, respectively. Then,

$$\mathcal{S}_{1,2\varepsilon}^* \preceq \mathcal{S}_1 \preceq \mathcal{S}_1^*.$$

Sketch of proof:

$\mathcal{S}_{1,2\varepsilon}^*(q_1)$

$$\mathcal{S}_1(q_1) = \bigcup_{q_2 \in R(q_1)} \mathcal{S}_{2,\varepsilon}^*(q_2) \quad \subseteq \quad \mathcal{S}_1^*(q_1)$$

$\uparrow$ ref

$\mathcal{S}_{2,\varepsilon}^*(q_2)$

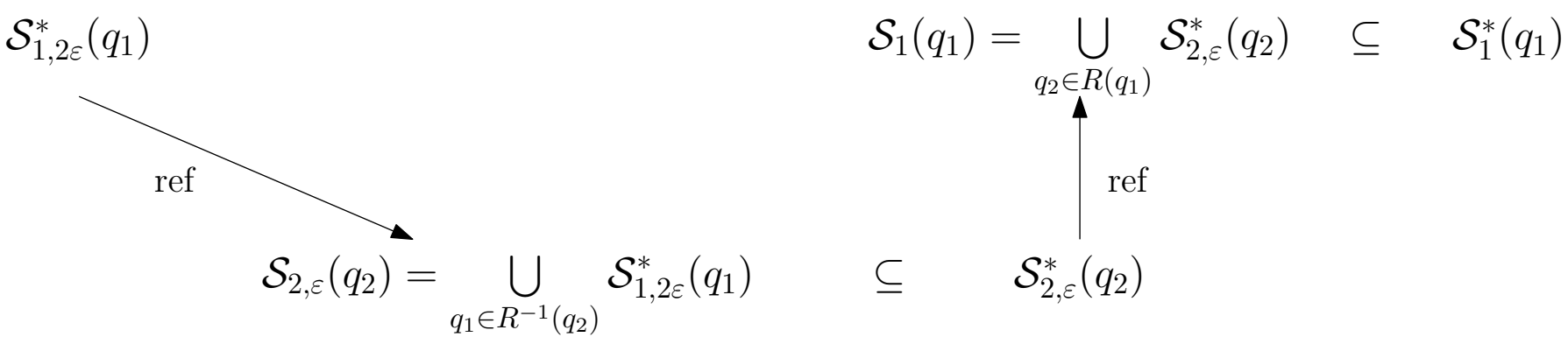
Distance to the Maximal Safe Controller

Theorem

Let $\mathcal{S}_1$ be the safe controller for T_1 for specification Y_s defined in the previous theorem. Let $\mathcal{S}_1^*$ and $\mathcal{S}_{1,2\varepsilon}^*$ be the maximal safe controllers for T_1 for specifications Y_s and $C_{2\varepsilon}(Y_s)$, respectively. Then,

$$\mathcal{S}_{1,2\varepsilon}^* \preceq \mathcal{S}_1 \preceq \mathcal{S}_1^*.$$

Sketch of proof:



Distance to the Maximal Safe Controller

Theorem

Let $\mathcal{S}_1$ be the safe controller for T_1 for specification Y_s defined in the previous theorem. Let $\mathcal{S}_1^*$ and $\mathcal{S}_{1,2\varepsilon}^*$ be the maximal safe controllers for T_1 for specifications Y_s and $C_{2\varepsilon}(Y_s)$, respectively. Then,

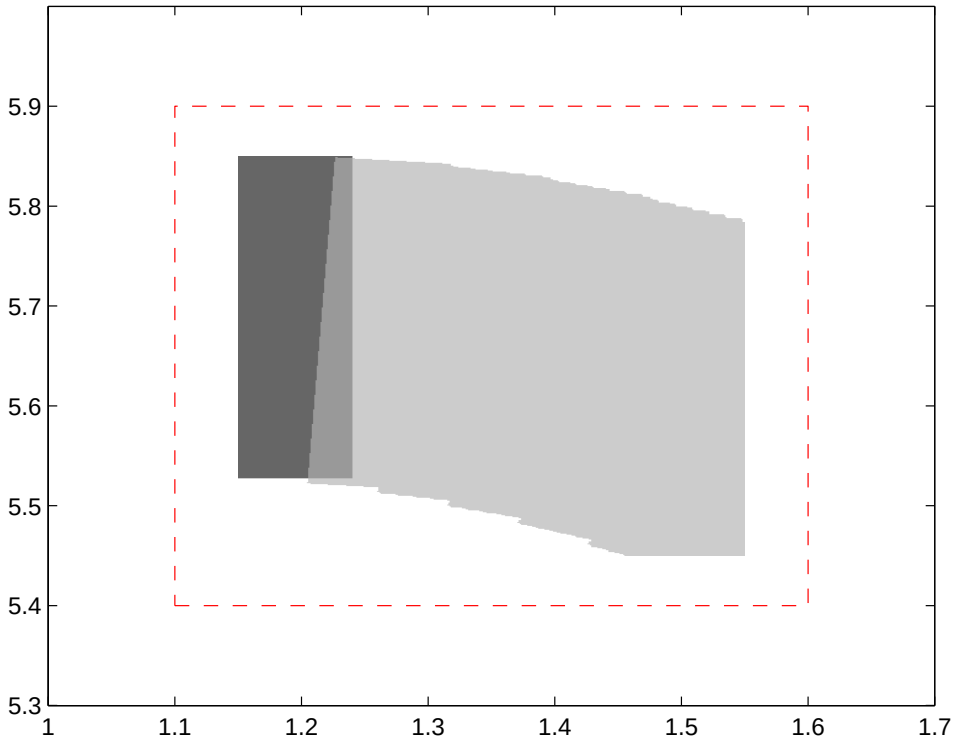
$$\mathcal{S}_{1,2\varepsilon}^* \preceq \mathcal{S}_1 \preceq \mathcal{S}_1^*.$$

Sketch of proof:

$$\begin{array}{ccccccccc}
\mathcal{S}_{1,2\varepsilon}^*(q_1) & \subseteq & \tilde{\mathcal{S}}_1(q_1) = \bigcup_{q_2 \in R(q_1)} \mathcal{S}_{2,\varepsilon}(q_2) & \subseteq & \mathcal{S}_1(q_1) = \bigcup_{q_2 \in R(q_1)} \mathcal{S}_{2,\varepsilon}^*(q_2) & \subseteq & \mathcal{S}_1^*(q_1) \\
 & \searrow^{\text{ref}} & \uparrow_{\text{ref}} & & \uparrow_{\text{ref}} & & \\
 & & \mathcal{S}_{2,\varepsilon}(q_2) = \bigcup_{q_1 \in R^{-1}(q_2)} \mathcal{S}_{1,2\varepsilon}^*(q_1) & \subseteq & \mathcal{S}_{2,\varepsilon}^*(q_2) & &
\end{array}$$

Safe Controller for the DC-DC Converter

- Abstraction parameters: $\tau = 1,\ \eta = 10^{-3} \implies \varepsilon = 0.05.$
- $Y_s = [1.1, 1.6] \times [5.4, 5.9] \implies C_\varepsilon(Y_s) = [1.15, 1.55] \times [5.45, 5.85].$
- The symbolic abstraction has 169744 states, the synthesis algorithm terminates in 2 iterations.

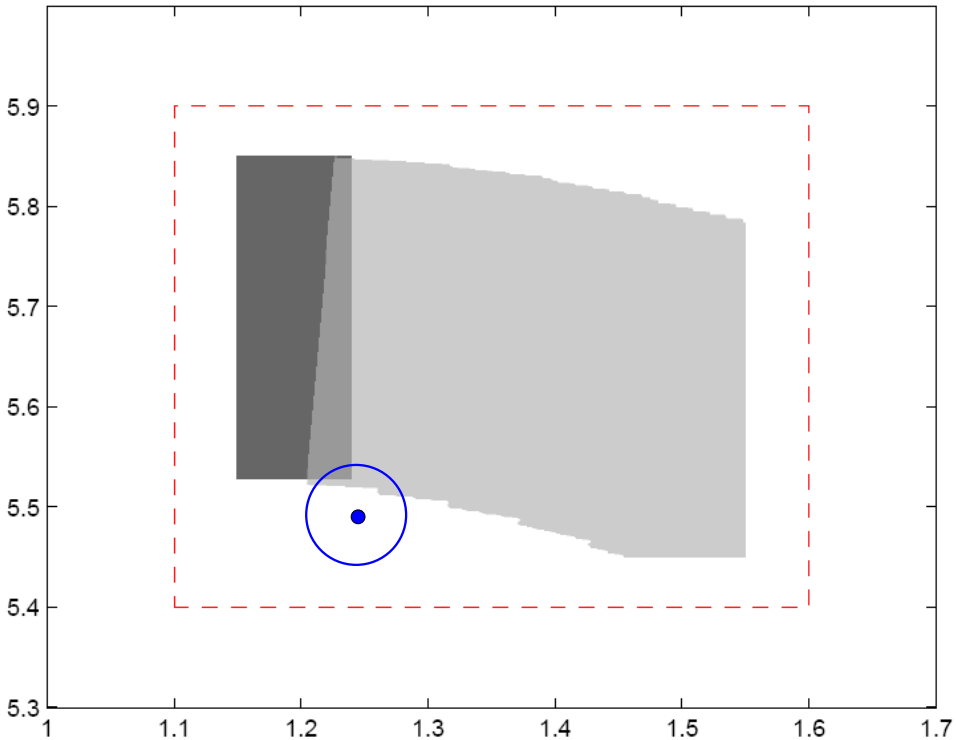


$\mathcal{S}^*_{2,\varepsilon}$

Safe Controller Refinement

Using the controller refinement equation:

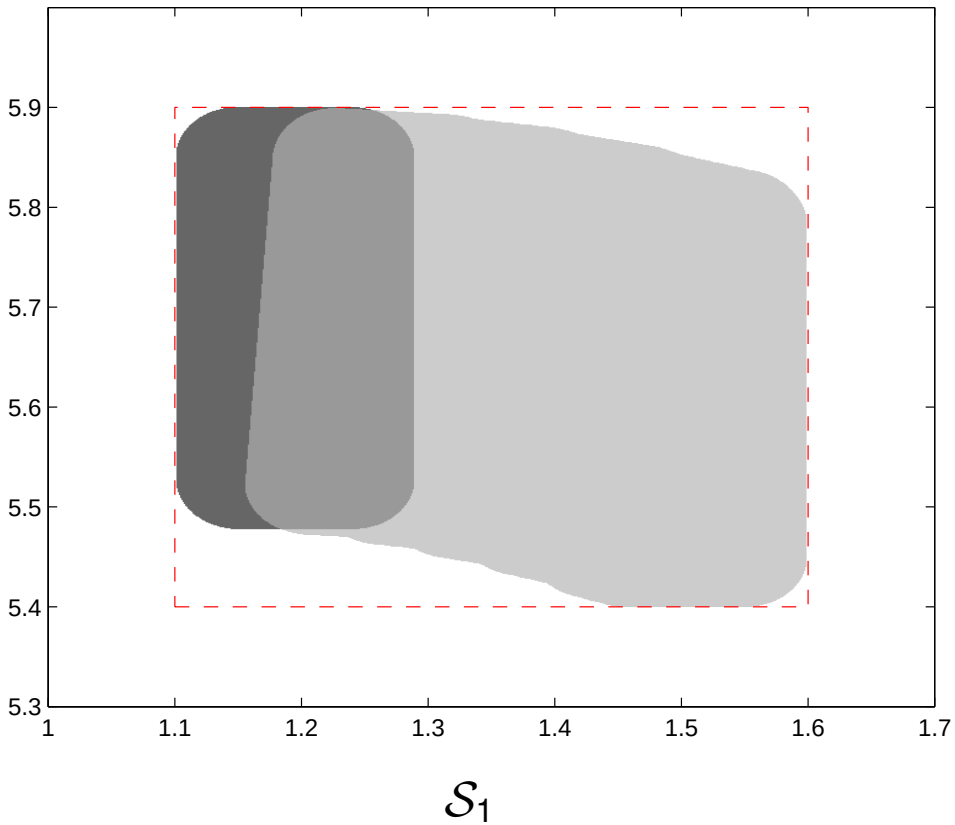
$$\forall x_1 \in X_1,\ \mathcal{S}_1(x_1) = \bigcup_{x_2 \in R(x_1)} \mathcal{S}^*_{2,\varepsilon}(x_2).$$



$\mathcal{S}^*_{2,\varepsilon}$

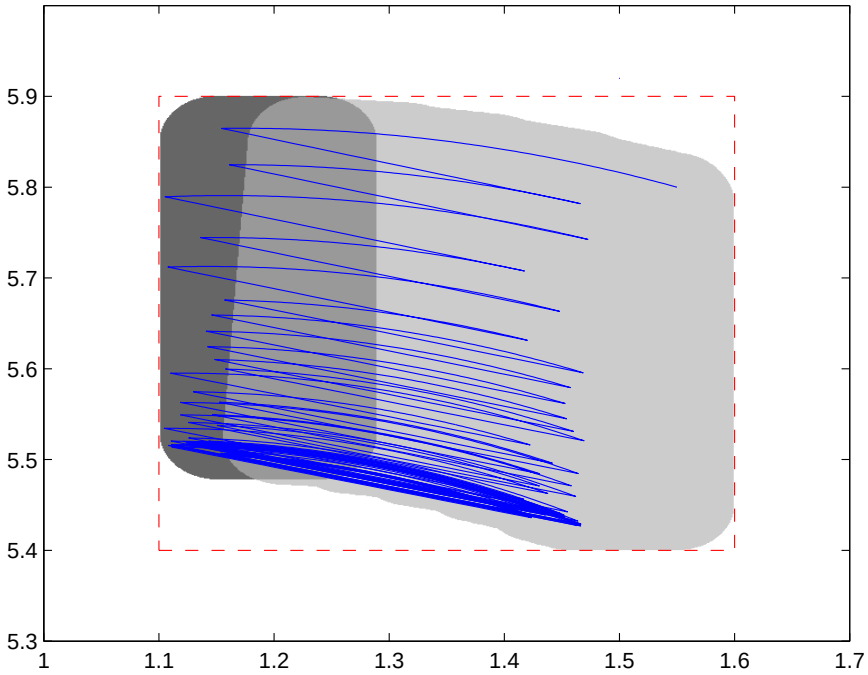
Switching Controller for the DC-DC Converter

- The synthesized controller is non-deterministic.
- Several implementations of the controller are possible.
- Possibility to ensure a posteriori secondary control objective.

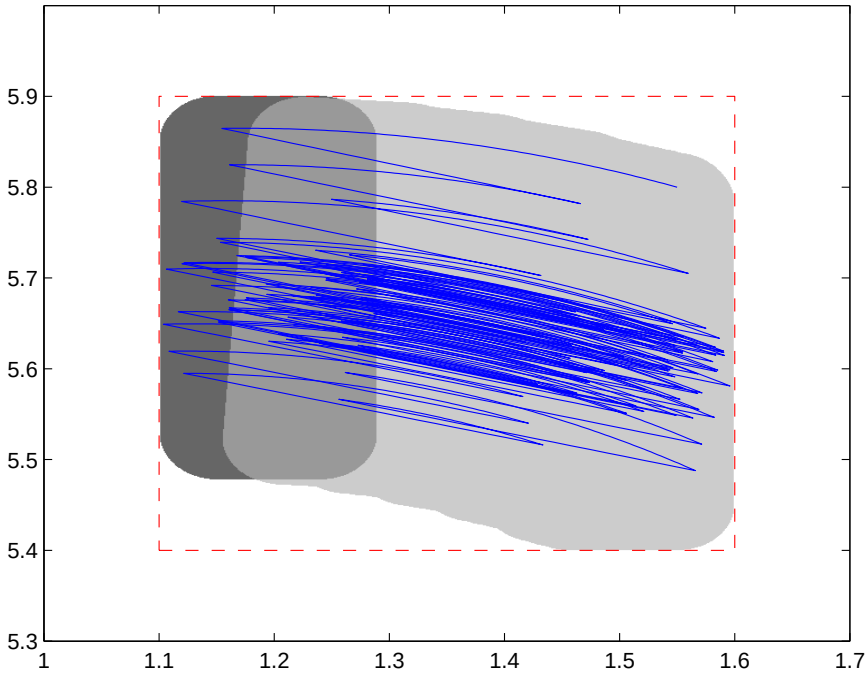


Switching Controller for the DC-DC Converter

Using two possible implementations:



Lazy control



Stochastic control

Both implementations satisfy the safety specification.

Outline of the Talk

1. Approximation relationships for discrete and continuous systems
   - Approximate bisimulation.
   - Symbolic abstractions of switched systems.
2. Controller synthesis using approximately bisimilar abstractions
   - Generic technique for controller refinement.
   - Synthesis for safety specifications.
   - Synthesis for reachability specifications under time optimization.

Controllers for Reachability Specifications

Let $T = (X, U, \longrightarrow, Y, H)$ and $\mathcal{S} : X \to 2^U$ be a controller for T.
Let $Y_s \subseteq Y$ specify the safe states $Y_t \subseteq Y_s$ specify the target states.

Definition

The *entry time of $T_{\mathcal{S}}$ from $x_0 \in X$ for specification (Y_s, Y_t)* is the smallest $N \in \mathbb{N}$ such that for all trajectories of $T_{\mathcal{S}}$ of length N and starting from x_0, $x_0 \xrightarrow{u_0} \dots \xrightarrow{u_{N-1}} x_N$, there exists $K \in \{0, \dots, N\}$ such that:

- $\forall k \in \{0, \dots, K\},\ H(x_k) \in Y_s$;
- $H(x_K) \in Y_t$.

The entry time is denoted by $J(T_{\mathcal{S}}, Y_s, Y_t, x_0)$.
If such N does not exist, then we define $J(T_{\mathcal{S}}, Y_s, Y_t, x_0) = +\infty$.

Time-Optimal Controller

The control objective is to minimize the entry time.

Definition

We say that a controller $\mathcal{S}^*$ is *time-optimal* for specification (Y_s, Y_t) if, for all controllers $\mathcal{S}$:

$$\forall x \in X,\ J(T_{\mathcal{S}^*}, Y_s, Y_t, x) \leq J(T_{\mathcal{S}}, Y_s, Y_t, x).$$

We define the *value function* of the time-optimal control problem as $J^*(T, Y_s, Y_t, x) = J(T_{\mathcal{S}^*}, Y_s, Y_t, x)$.

Time-Optimal Controller

The control objective is to minimize the entry time.

Definition

We say that a controller $\mathcal{S}^*$ is *time-optimal* for specification (Y_s, Y_t) if, for all controllers $\mathcal{S}$:

$$\forall x \in X,\ J(T_{\mathcal{S}^*}, Y_s, Y_t, x) \leq J(T_{\mathcal{S}}, Y_s, Y_t, x).$$

We define the *value function* of the time-optimal control problem as $J^*(T, Y_s, Y_t, x) = J(T_{\mathcal{S}^*}, Y_s, Y_t, x)$.

- There exists a time-optimal controller (may be not unique).
- It can be determined by dynamic programming and fixed point computation of the value function of the time-optimal control problem.

Computation of a Time-Optimal Controller

Theorem

Let $\mathcal{S}^* : X \to 2^U$ *be the controller for* T *defined, for all* $x \in X$ *by*

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ *is a time-optimal controller for the specification* (Y_s, Y_t).

Computation of a Time-Optimal Controller

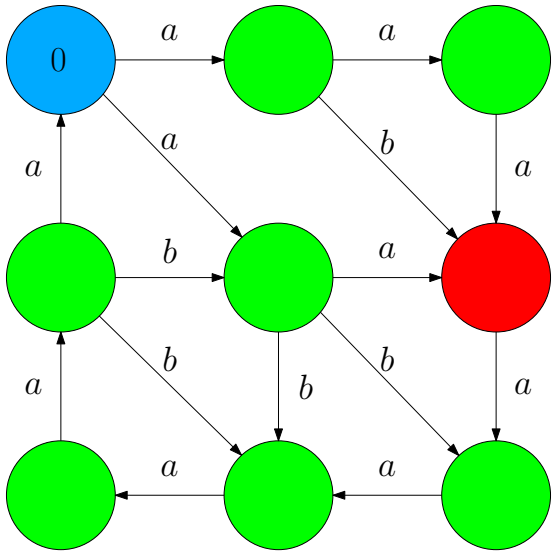
Theorem

Let $\mathcal{S}^* : X \to 2^U$ be the controller for T defined, for all $x \in X$ by

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ is a time-optimal controller for the specification (Y_s, Y_t).

A simple example:



Computation of a Time-Optimal Controller

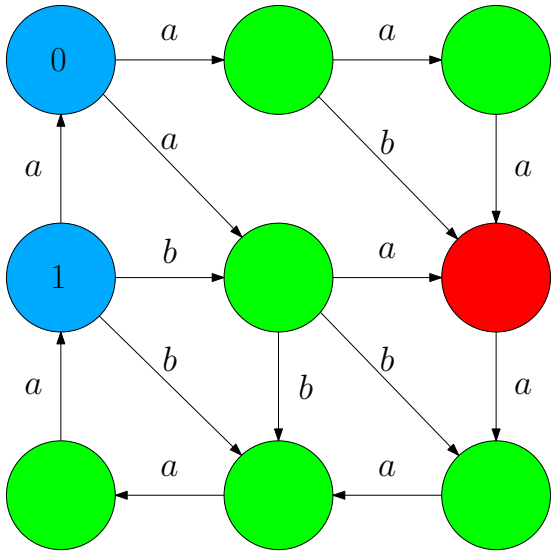
Theorem

Let $\mathcal{S}^* : X \to 2^U$ *be the controller for* T *defined, for all* $x \in X$ *by*

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ *is a time-optimal controller for the specification* (Y_s, Y_t).

A simple example:



Computation of a Time-Optimal Controller

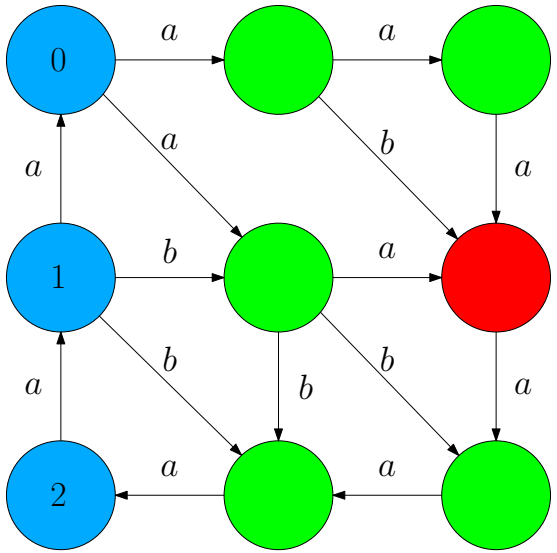
Theorem

Let $\mathcal{S}^* : X \to 2^U$ *be the controller for* T *defined, for all* $x \in X$ *by*

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ *is a time-optimal controller for the specification* (Y_s, Y_t).

A simple example:



Computation of a Time-Optimal Controller

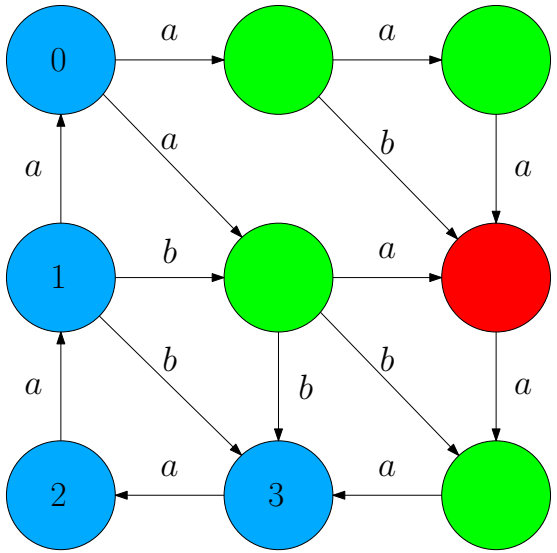
Theorem

Let $\mathcal{S}^* : X \to 2^U$ *be the controller for* T *defined, for all* $x \in X$ *by*

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ *is a time-optimal controller for the specification* (Y_s, Y_t).

A simple example:



Computation of a Time-Optimal Controller

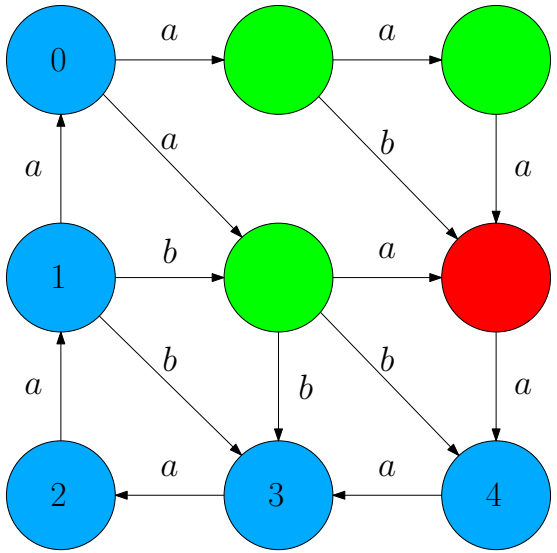
Theorem

Let $\mathcal{S}^* : X \to 2^U$ *be the controller for* T *defined, for all* $x \in X$ *by*

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ *is a time-optimal controller for the specification* (Y_s, Y_t).

A simple example:



Computation of a Time-Optimal Controller

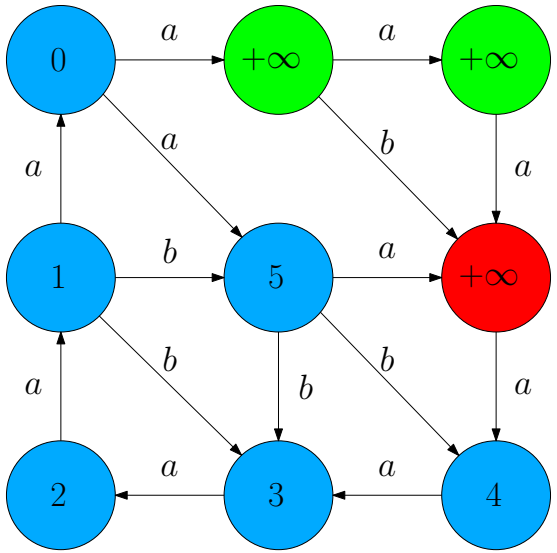
Theorem

Let $\mathcal{S}^* : X \to 2^U$ *be the controller for* T *defined, for all* $x \in X$ *by*

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ *is a time-optimal controller for the specification* (Y_s, Y_t).

A simple example:



Computation of a Time-Optimal Controller

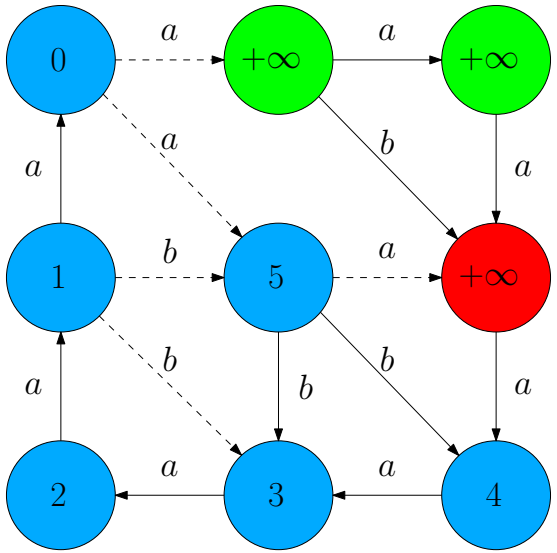
Theorem

Let $\mathcal{S}^* : X \to 2^U$ be the controller for T defined, for all $x \in X$ by

$$\mathcal{S}^*(x) = \arg\min_{u \in U} \left(\max_{x' \in \mathrm{Post}_u(x)} J^*(T, Y_s, Y_t, x') \right).$$

Then, $\mathcal{S}^*$ is a time-optimal controller for the specification (Y_s, Y_t).

A simple example:



Controller Synthesis via Symbolic Abstractions

Time-optimal controllers are easy to compute for symbolic abstractions... We need a controller refinement procedure!

Theorem

Let $T_1 \sim_\varepsilon T_2$, let $R \subseteq X_1 \times X_2$ denote the ε-approximate bisimulation relation between T_1 and T_2. Let $\mathcal{S}^*_{2,\varepsilon}$ be a time-optimal controller for T_2 for the specification $(C_\varepsilon(Y_s), C_\varepsilon(Y_t))$. Let $\mathcal{S}_1$ be the controller for T_1 given by

$$\forall x_1 \in X_1,\ \mathcal{S}_1(x_1) = \mathcal{S}^*_{2,\varepsilon}\left(\arg \min_{x_2 \in R(x_1)} J^*(T_2, C_\varepsilon(Y_s), C_\varepsilon(Y_t), x_2)\right).$$

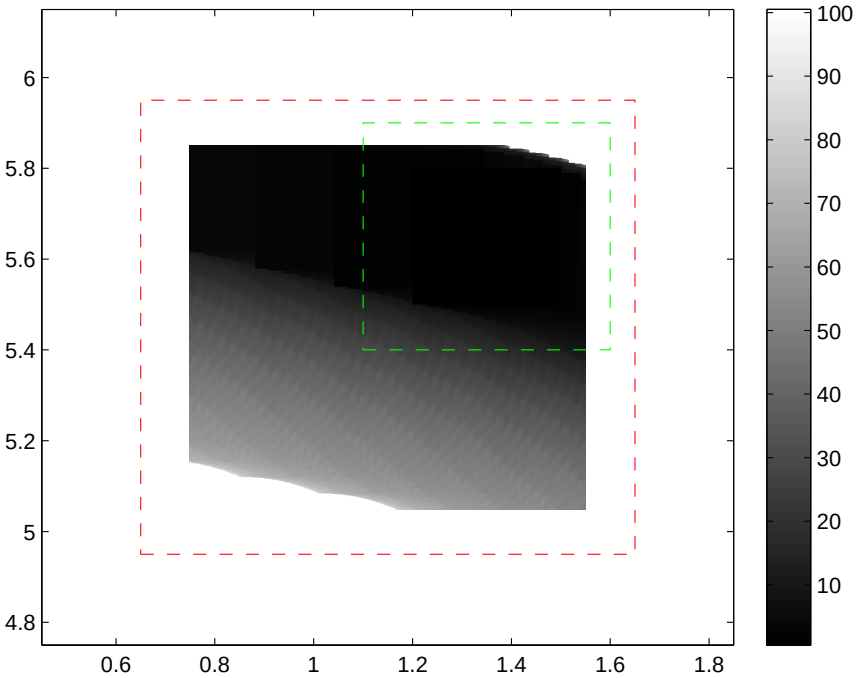
The entry time of $T_{\mathcal{S}_1}$ for specification (Y_s, Y_t) satisfies for all $x_1 \in X_1$:

$$J^*(T_1, Y_s, Y_t, x_1) \leq J(T_{1,\mathcal{S}_1}, Y_s, Y_t, x_1) \leq J^*(T_1, C_{2\varepsilon}(Y_s), C_{2\varepsilon}(Y_t), x_1).$$

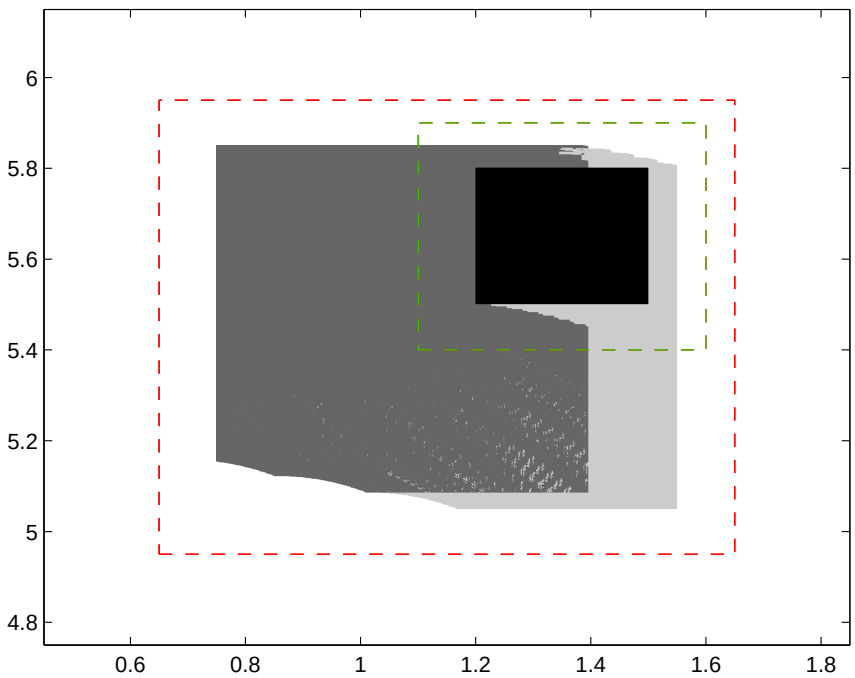
Proof is close to the case of safety controllers.

Reachability Controller for the DC-DC Converter

- Abstraction parameters: $\tau = 0.5,\ \eta = 10^{-3} \implies \varepsilon = 0.1.$
- $Y_s = [0.65, 1.65] \times [4.95, 5.95] \implies C_\varepsilon(Y_s) = [0.75, 1.55] \times [5.05, 5.85].$
- $Y_t = [1.1, 1.6] \times [5.4, 5.9] \implies C_\varepsilon(Y_t) = [1.1, 1.6] \times [5.4, 5.9].$
- The synthesis algorithm terminates in 94 iterations.



$J^*(T_2, C_\varepsilon(Y_s), C_\varepsilon(Y_t), x_2)$

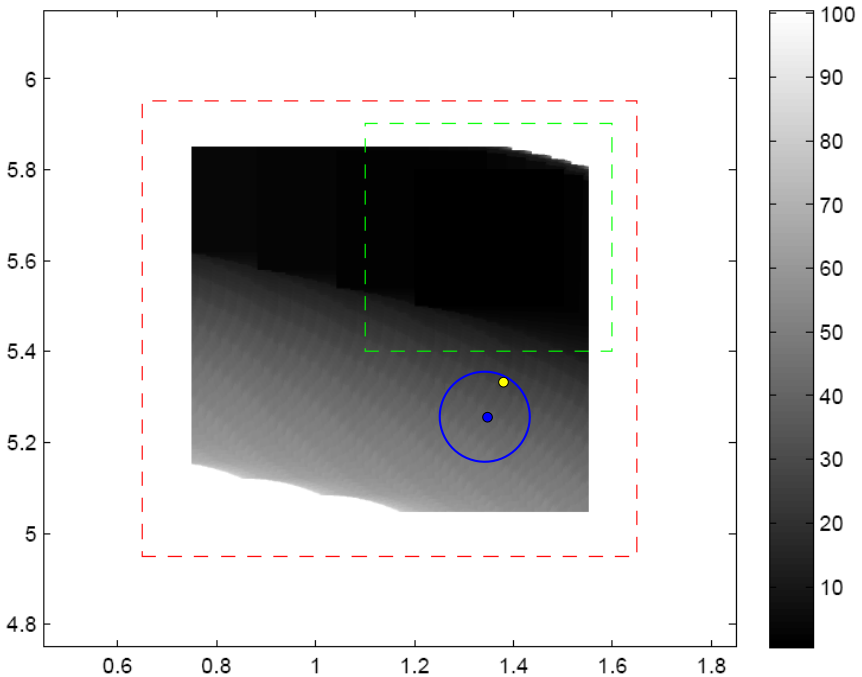


$S^*_{2,\varepsilon}$

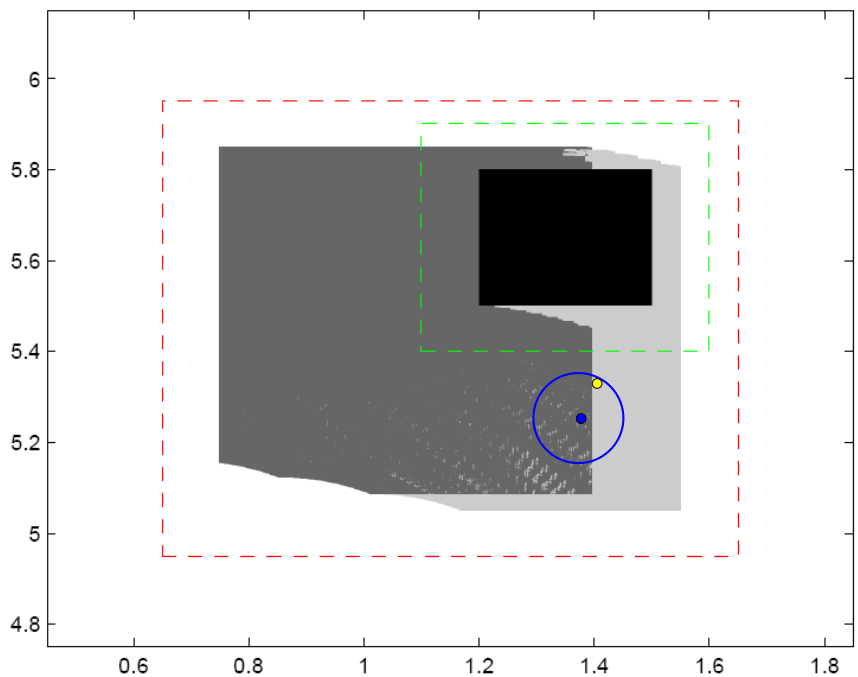
Reachability Controller Refinement

Using the controller refinement equation:

$$\mathcal{S}_1(x_1) = \mathcal{S}^*_{2,\varepsilon}\left(\arg \min_{x_2 \in R(x_1)} J^*(T_2, C_\varepsilon(Y_s), C_\varepsilon(Y_t), x_2)\right).$$



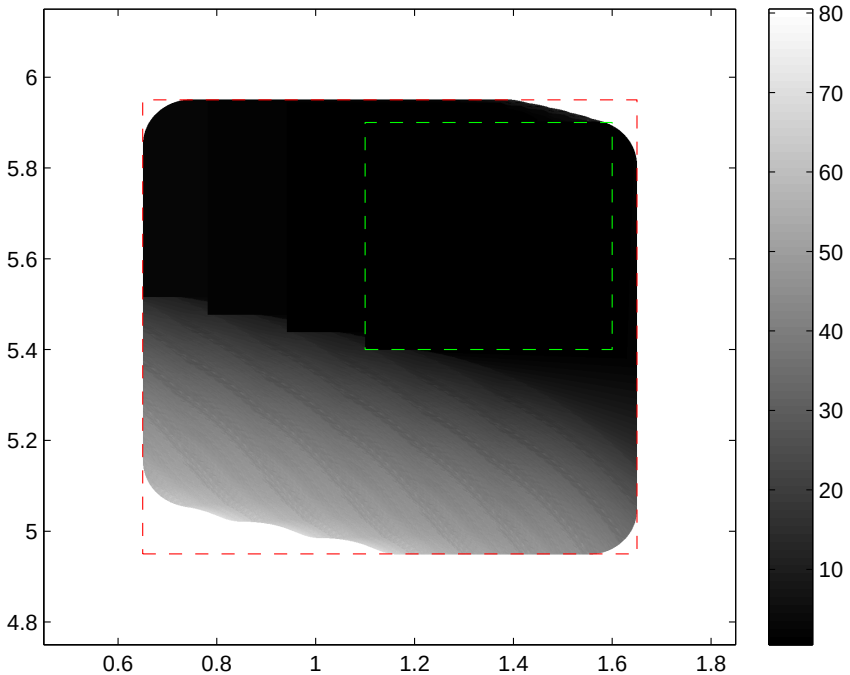
$J^*(T_2, C_\varepsilon(Y_s), C_\varepsilon(Y_t), x_2)$



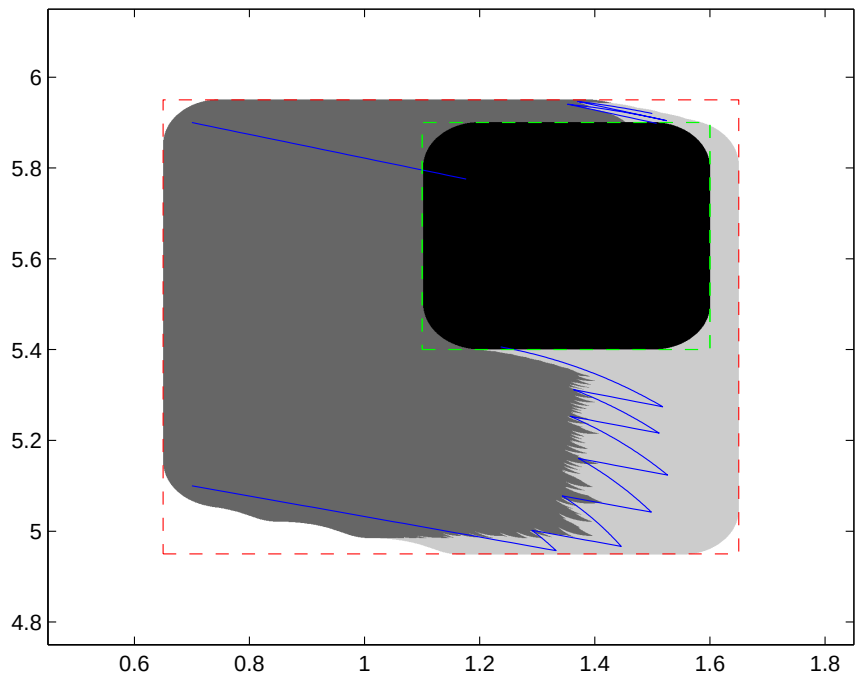
$\mathcal{S}^*_{2,\varepsilon}$

Switching Controller for the DC-DC Converter

- Controller $\mathcal{S}_1$ seems to be more "regular" than $\mathcal{S}^*_{2,\varepsilon}$.
- Trajectories of the controlled system meet the specification.



$J(T_{1,\mathcal{S}_1}, Y_s, Y_t, x_1)$



$\mathcal{S}_1$

Conclusions

- Approximately bisimilar symbolic abstractions:
  - A rigorous tool for controller synthesis:
    $\Longrightarrow$ Controllers are “correct by design”.
  - Allow us to leverage efficient algorithmic techniques from discrete systems to continuous and hybrid systems.
  - Computable for interesting classes of systems: switched systems, continuous control systems...
- Controller refinement procedures:
  - Generic controller refinement: independent of the specification, not a static controller.
  - Specific procedure for safety/reachability controllers: static controllers, estimates of distance to maximality/optimality provide guarantee of performance.