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)

Discrete-time linear control systems

Review

- Let us recall the notion of discrete-time linear control system.

Definition (Discrete-time linear control system)

A discrete-time control system is a quadruple $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ consisting of:

- the state space $\mathbb{R}^n$;
- the input space $\mathbb{R}^m$;
- the difference equation $\xi(k+1) = A\xi(k) + B\nu(k)$ where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $t \in \mathbb{N}_0$.

Discrete-time linear control systems

Review

- Let us recall the notion of discrete-time linear control system.

Definition (Discrete-time linear control system)

A discrete-time control system is a quadruple $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ consisting of:

- the state space $\mathbb{R}^n$;
- the input space $\mathbb{R}^m$;
- the difference equation $\xi(k+1) = A\xi(k) + B\nu(k)$ where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $t \in \mathbb{N}_0$.

- Can we represent discrete-time control systems as systems?

Exact finite-state abstractions for control

Control systems as systems

- Recall that a partition $\mathcal{P}$ of $\mathbb{R}^n$ is a collection of sets $\mathcal{P} = \{P_i\}_{i \in I}$ such that:
  - $\cup_{i \in I} P_i = \mathbb{R}^n$;
  - $P_i \cap P_j = \varnothing$ for $i \neq j$.

- Recall that a partition $\mathcal{P}$ of $\mathbb{R}^n$ is a collection of sets $\mathcal{P} = \{P_i\}_{i\in I}$ such that:
  - $\cup_{i\in I} P_i = \mathbb{R}^n$;
  - $P_i \cap P_j = \varnothing$ for $i \neq j$.

Definition

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system and let $\mathcal{P}$ be a partition of $\mathbb{R}^n$. The system associated with Σ and $\mathcal{P}$, denoted by $S_{\mathcal{P}}(\Sigma)$, consists of:

- $X = \mathbb{R}^n$;
- $U = \mathbb{R}^m$;
- $x \xrightarrow{u} x'$ if $x' = Ax + Bu$;
- $Y = \mathcal{P}$;
- H defined by $H(x) = P_i$ if $x \in P_i$.

- We start with:
  - a discrete-time linear control system Σ;
  - a finite partition $\mathcal{P}$ of the state space of Σ, e.g., $\mathcal{P}$ induced by the predicates appearing in a LTL formula.

Exact finite-state abstractions for control

Adapted sets

- We start with:
  - a discrete-time linear control system Σ;
  - a finite partition $\mathcal{P}$ of the state space of Σ, e.g., $\mathcal{P}$ induced by the predicates appearing in a LTL formula.
- and ask, when is $S_{\mathcal{P}}(\Sigma)$ bisimilar to a finite-state system?

Exact finite-state abstractions for control

Adapted sets

- We start with:
  - a discrete-time linear control system Σ;
  - a finite partition $\mathcal{P}$ of the state space of Σ, e.g., $\mathcal{P}$ induced by the predicates appearing in a LTL formula.
- and ask, when is $S_{\mathcal{P}}(\Sigma)$ bisimilar to a finite-state system?

Definition (Adapted sets)

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear system and consider a collection of m vectors $c_1, c_2, \ldots, c_m \in \mathbb{R}^n$ for which there exist numbers $\mu_1, \mu_2, \ldots, \mu_m \in \mathbb{N}$ satisfying:

1. $c_r^T b_r = 0, \; c_r^T A b_r = 0, \ldots, c_r^T A^{\mu_r - 2} b_r = 0, \; c_r^T A^{\mu_r - 1} b_r \neq 0, \, r = 1, \ldots, m;$
2. the vectors $c_1^T A^{\mu_1 - 1} B, c_2^T A^{\mu_2 - 1} B, \ldots, c_m^T A^{\mu_m - 1} B$ are linearly independent,

where $b_1, b_2, \ldots, b_m$ are the columns of B. The class of subsets of $\mathbb{R}^n$ adapted to Σ is formed by finite unions of sets defined by conjunctions of conditions of the form $f \sim 0$ with $f = \pm c_r^T A^l x \pm e$, $e \in \mathbb{R}$, $l \in \{0, 1, \ldots, \mu_r - 1\}$, and $\sim \in \{=, >\}$.

Exact finite-state abstractions for control

Adapted sets

- We start with:
  - a discrete-time linear control system Σ;
  - a finite partition $\mathcal{P}$ of the state space of Σ, e.g., $\mathcal{P}$ induced by the predicates appearing in a LTL formula.
- and ask, when is $S_{\mathcal{P}}(\Sigma)$ bisimilar to a finite-state system?

Definition (Adapted sets)

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear system and consider a collection of m vectors $c_1, c_2, \ldots, c_m \in \mathbb{R}^n$ for which there exist numbers $\mu_1, \mu_2, \ldots, \mu_m \in \mathbb{N}$ satisfying:

1. $c_r^T b_r = 0,\ c_r^T A b_r = 0, \ldots, c_r^T A^{\mu_r - 2} b_r = 0,\ c_r^T A^{\mu_r - 1} b_r \neq 0$, $r = 1, \ldots, m$;
2. the vectors $c_1^T A^{\mu_1 - 1} B, c_2^T A^{\mu_2 - 1} B, \ldots, c_m^T A^{\mu_m - 1} B$ are linearly independent,

where $b_1, b_2, \ldots, b_m$ are the columns of B. The class of subsets of $\mathbb{R}^n$ adapted to Σ is formed by finite unions of sets defined by conjunctions of conditions of the form $f \sim 0$ with $f = \pm c_r^T A^l x \pm e$, $e \in \mathbb{R}$, $l \in \{0, 1, \ldots, \mu_r - 1\}$, and $\sim \in \{=, >\}$.

Consider the controlled model for the national income inspired by Paul Samuelson's 1939 model:

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

where the national income is the sum $c+i+g$ of three kinds of expenditures: consumption (c), investment (i), and government expenditures (g).

Exact finite-state abstractions for control

Adapted sets

Consider the controlled model for the national income inspired by Paul Samuelson's 1939 model:

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

where the national income is the sum $c+i+g$ of three kinds of expenditures: consumption (c), investment (i), and government expenditures (g).

Taking $\alpha=\frac{1}{2}$ and $\beta=2$, the corresponding A and B matrices are given by:

$$A=\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad B=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Exact finite-state abstractions for control

Adapted sets

Consider the controlled model for the national income inspired by Paul Samuelson's 1939 model:

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

where the national income is the sum $c+i+g$ of three kinds of expenditures: consumption (c), investment (i), and government expenditures (g).

Taking $\alpha=\frac{1}{2}$ and $\beta=2$, the corresponding A and B matrices are given by:

$$A=\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad B=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

For this system $m=1$ and adapted sets are defined using a single vector c_1. The choice $c_1=\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ satisfies $c_1^T B=0$, $c_1^T AB\neq 0$ and thus $\mu_1=2$.

Exact finite-state abstractions for control

Adapted sets

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad c_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T.$$

Noting that $c_1^T x = c$ and $c_1^T A x = \frac{1}{2}(c + i + g)$, the 8 possible functions f for a fixed $e \in \mathbb{R}$ are given by:

$$\begin{aligned} f_1(c,i,g) &= c + e, & f_5(c,i,g) &= \frac{1}{2}(c+i+g) + e, \\ f_2(c,i,g) &= c - e, & f_6(c,i,g) &= \frac{1}{2}(c+i+g) - e, \\ f_3(c,i,g) &= -c + e, & f_7(c,i,g) &= -\frac{1}{2}(c+i+g) + e, \\ f_4(c,i,g) &= -c - e, & f_8(c,i,g) &= -\frac{1}{2}(c+i+g) - e. \end{aligned}$$

- What can we do with adapted sets?

- What can we do with adapted sets?

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- What can we do with adapted sets?

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Is this the kind of abstraction we need?

Exact finite-state abstractions for control

Existence of finite-state bisimilar abstractions

- What can we do with adapted sets?

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Is this the kind of abstraction we need?
- Since $S_{\mathcal{P}}(\Sigma)$ is deterministic, bisimulation and alternating bisimulation coincide.

- What can we do with adapted sets?

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Is this the kind of abstraction we need?
- Since $S_{\mathcal{P}}(\Sigma)$ is deterministic, bisimulation and alternating bisimulation coincide.
- How do we compute these abstractions?

Exact finite-state abstractions for control

Quotient systems

- Given a system S and a partition $\mathcal{P}$ on X, we can always construct a new system $S_{/\mathcal{P}}$ that simulates S.

- Given a system S and a partition $\mathcal{P}$ on X, we can always construct a new system $S_{/\mathcal{P}}$ that simulates S.

Definition (Quotient system)

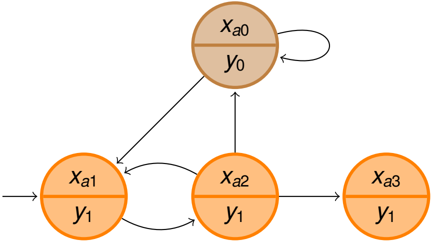
Let $S = (X, X_0, U, \longrightarrow, Y, H)$ be a system and let $\mathcal{P} = \{P_i\}_{i \in I}$ be a partition of X such that $x, x' \in P_i$ for some $i \in I$ implies $H(x) = H(x')$. The quotient of S by $\mathcal{P}$, denoted by $S_{/\mathcal{P}}$, is the system $(X_{/\mathcal{P}}, X_{/\mathcal{P}0}, U_{/\mathcal{P}}, \underset{/\mathcal{P}}{\longrightarrow}, Y_{/\mathcal{P}}, H_{/\mathcal{P}})$ consisting of:

- $X_{/\mathcal{P}} = \mathcal{P}$;
- $X_{/\mathcal{P}0} = \{P_i \in \mathcal{P} \mid P_i \cap X_0 \neq \varnothing\}$;
- $U_{/\mathcal{P}} = U$;
- $P_i \xrightarrow[/\mathcal{P}]{u} P_j$ if there exists $x \xrightarrow{u} x'$ in S with $x \in P_i$ and $x' \in P_j$;
- $Y_{/\mathcal{P}} = Y$;
- $H_{/\mathcal{P}}(P_i) = H(x)$ for any $x \in P_i$.

- Consider the following system and the partition:

$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$

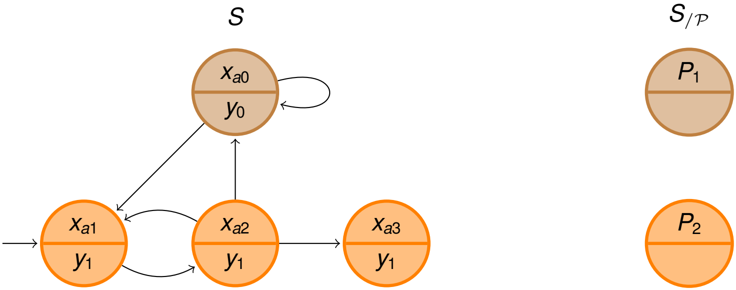
S



$S_{/\mathcal{P}}$

- Consider the following system and the partition:

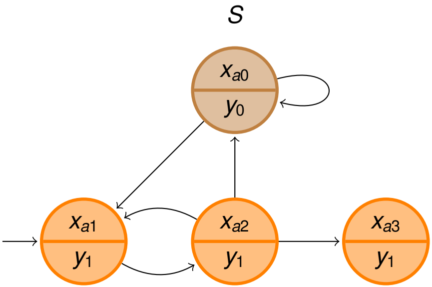
$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$

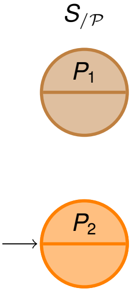


- States X;

- Consider the following system and the partition:

$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$

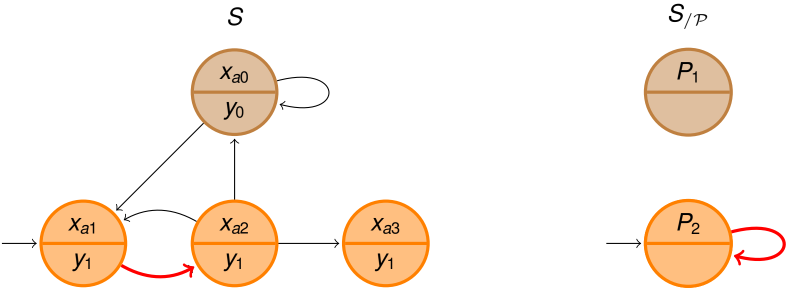




- States X; Initial states X_0;

- Consider the following system and the partition:

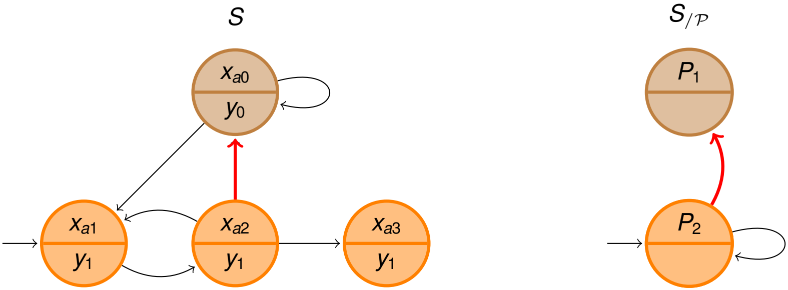
$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$



- States X; Initial states X_0;
- Transitions $\longrightarrow$;

- Consider the following system and the partition:

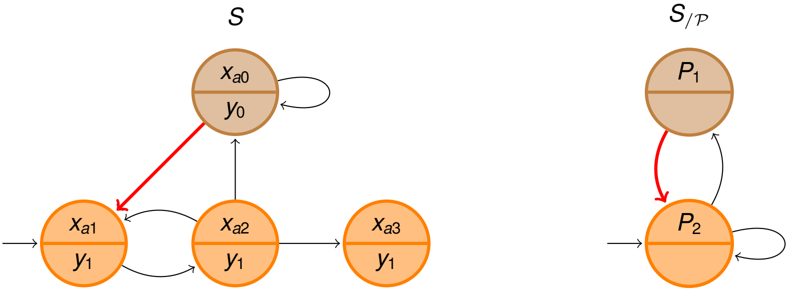
$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$



- States X; Initial states X_0;
- Transitions $\longrightarrow$;

- Consider the following system and the partition:

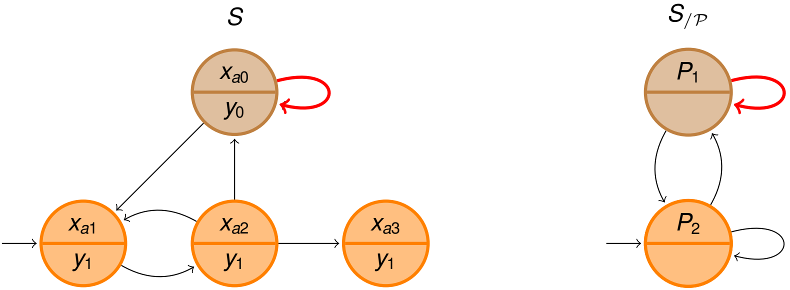
$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$



- States X; Initial states X_0;
- Transitions $\longrightarrow$;

- Consider the following system and the partition:

$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$



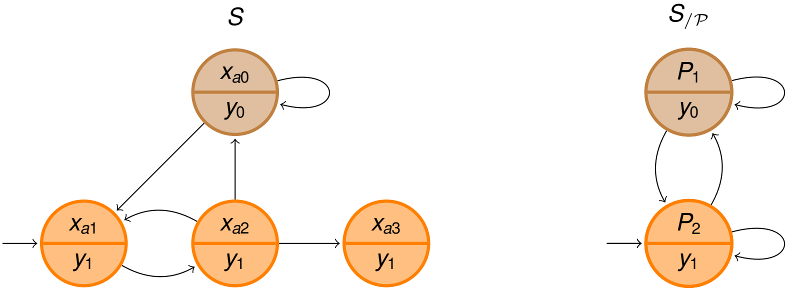
- States X; Initial states X_0;
- Transitions $\longrightarrow$;

Similarity Relationships

Quotient Systems

- Consider the following system and the partition:

$$\mathcal{P} = \{\{x_{a0}\}, \{x_{a1}, x_{a2}, x_{a3}\}\} = \{P_1, P_2\}.$$



- States X; Initial states X_0;
- Transitions $\longrightarrow$;
- Outputs $Y, H : X \to Y$.

- By construction, $S_{/\mathcal{P}}$ simulates S. The simulation relation is given by:

$$\left\{(x, P_i) \in X \times X_{/\mathcal{P}} \mid x \in P_i\right\}.$$

- By construction, $S_{/\mathcal{P}}$ simulates S. The simulation relation is given by:

$$\left\{(x, P_i) \in X \times X_{/\mathcal{P}} \mid x \in P_i\right\}.$$

- When can we strengthen this simulation to a bisimulation?

- By construction, $S_{/\mathcal{P}}$ simulates S. The simulation relation is given by:

$$\left\{(x, P_i) \in X \times X_{/\mathcal{P}} \mid x \in P_i\right\}.$$

- When can we strengthen this simulation to a bisimulation?

Theorem

Let $S = (X, X_0, U, \longrightarrow, Y, H)$ be a system and let $\mathcal{P} = \{P_i\}_{i \in I}$ be a partition of X such that $x, x' \in P_i$ for some $i \in I$ implies $H(x) = H(x')$. The relation:

$$R = \left\{(x, P_i) \in X \times X_{/\mathcal{P}} \mid x \in P_i\right\}$$

is a simulation relation from S to $S_{/\mathcal{P}}$. Moreover, R is a bisimulation relation between S and $S_{/\mathcal{P}}$ iff the following is a bisimulation relation between S and S:

$$\left\{(x, x') \in X \times X \mid x, x' \in P_i \text{ for some } P_i \in \mathcal{P}\right\}.$$

- When:

$$\{(x, x') \in X \times X \mid x, x' \in P_i \text{ for some } P_i \in \mathcal{P}\}$$

is not a bisimulation relation we can subdivide (refine) $\mathcal{P}$ until we reach a bisimulation.

- When:
  $$\{(x,x') \in X \times X \mid x,x' \in P_i \text{ for some } P_i \in \mathcal{P}\}$$
  is not a bisimulation relation we can subdivide (refine) $\mathcal{P}$ until we reach a bisimulation.
- We first define Pre (the dual of Post) for a given system S and a subset $P \subseteq X$:
  $$\mathrm{Pre}(P) = \left\{x \in X \mid \exists u \in U \ x \xrightarrow{u} x' \in P\right\}.$$

- When:

$$\{(x, x') \in X \times X \mid x, x' \in P_i \text{ for some } P_i \in \mathcal{P}\}$$

is not a bisimulation relation we can subdivide (refine) $\mathcal{P}$ until we reach a bisimulation.

- We first define Pre (the dual of Post) for a given system S and a subset $P \subseteq X$:

$$\text{Pre}(P) = \left\{x \in X \mid \exists u \in U \; x \xrightarrow{u} x' \in P\right\}.$$

and then use it in the following partition refinement algorithm:

$$\texttt{While } (\exists \texttt{P}_\texttt{i}, \texttt{P}_\texttt{j} \in \mathcal{P} \mid \varnothing \neq \texttt{P}_\texttt{j} \cap \texttt{Pre}(\texttt{P}_\texttt{i}) \neq \texttt{P}_\texttt{j})$$

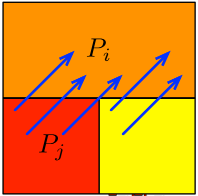
$$\{$$

$$\texttt{P}_\texttt{a} := \texttt{P}_\texttt{j} \cap \texttt{Pre}(\texttt{P}_\texttt{i});$$

$$\texttt{P}_\texttt{b} := \texttt{P}_\texttt{j} \backslash \texttt{Pre}(\texttt{P}_\texttt{i});$$

$$\mathcal{P} := (\mathcal{P} \backslash \{\texttt{P}_\texttt{j}\}) \cup \{\texttt{P}_\texttt{a}, \texttt{P}_\texttt{b}\}$$

$$\}$$



- When:

$$\{(x, x') \in X \times X \mid x, x' \in P_i \text{ for some } P_i \in \mathcal{P}\}$$

is not a bisimulation relation we can subdivide (refine) $\mathcal{P}$ until we reach a bisimulation.

- We first define Pre (the dual of Post) for a given system S and a subset $P \subseteq X$:

$$\text{Pre}(P) = \left\{x \in X \mid \exists u \in U \; x \xrightarrow{u} x' \in P\right\}.$$

and then use it in the following partition refinement algorithm:

$\mathtt{While}\ (\exists \mathtt{P_i}, \mathtt{P_j} \in \mathcal{P} \mid \varnothing \neq \mathtt{P_j} \cap \mathtt{Pre}(\mathtt{P_i}) \neq \mathtt{P_j})$

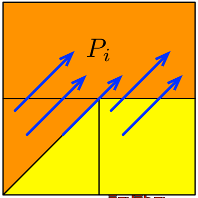
$\{$

$\mathtt{P_a} := \mathtt{P_j} \cap \mathtt{Pre}(\mathtt{P_i});$

$\mathtt{P_b} := \mathtt{P_j} \backslash \mathtt{Pre}(\mathtt{P_i});$

$\mathcal{P} := (\mathcal{P} \backslash \{\mathtt{P_j}\}) \cup \{\mathtt{P_a}, \mathtt{P_b}\}$

$\}$



Exact finite-state abstractions for control

Revisiting the economics example

- Consider again the controlled model for the national income and suppose that we are interested in reducing the internal consumption from 10 units to 2 units while keeping the national income above 20 units.

Exact finite-state abstractions for control

Revisiting the economics example

- Consider again the controlled model for the national income and suppose that we are interested in reducing the internal consumption from 10 units to 2 units while keeping the national income above 20 units.
- Which partition to use?

Exact finite-state abstractions for control

Revisiting the economics example

- Consider again the controlled model for the national income and suppose that we are interested in reducing the internal consumption from 10 units to 2 units while keeping the national income above 20 units.
- Which partition to use?

$$
\begin{aligned}
P_1 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g < 20\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge c \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge 2 < c < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge c \geq 10\}.
\end{aligned}
$$

Exact finite-state abstractions for control

Revisiting the economics example

- Consider again the controlled model for the national income and suppose that we are interested in reducing the internal consumption from 10 units to 2 units while keeping the national income above 20 units.
- Which partition to use?

$$
\begin{aligned}
P_1 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g < 20\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge c \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge 2 < c < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge c \geq 10\}.
\end{aligned}
$$

- Is this an adapted partition?

CyPhyLab UCLA

Exact finite-state abstractions for control

Revisiting the economics example

- Consider again the controlled model for the national income and suppose that we are interested in reducing the internal consumption from 10 units to 2 units while keeping the national income above 20 units.
- Which partition to use?

$$
\begin{aligned}
P_1 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g < 20\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge c \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge 2 < c < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c+i+g \geq 20 \wedge c \geq 10\}.
\end{aligned}
$$

- Is this an adapted partition? The equalities $c_1^T x = c$ and $c_1^T Ax = \frac{1}{2}(c+i+g)$ imply:

$$
\begin{aligned}
P_1 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T Ax < 10\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T Ax \geq 10 \wedge c_1^T x \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T Ax \geq 10 \wedge 2 < c_1^T x < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T Ax \geq 10 \wedge c_1^T x \geq 10\}.
\end{aligned}
$$

Exact finite-state abstractions for control

Revisiting the economics example

$$
\begin{aligned}
P_1 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x < 10\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge c_1^T x \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge 2 < c_1^T x < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge c_1^T x \geq 10\}.
\end{aligned}
$$

- In terms of these regions, our objective can be formulated as the existence of a control strategy driving all the points in region P_4 to region P_2 without entering region P_1.

Exact finite-state abstractions for control

Revisiting the economics example

$$
\begin{aligned}
P_1 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x < 10\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge c_1^T x \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge 2 < c_1^T x < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge c_1^T x \geq 10\}.
\end{aligned}
$$

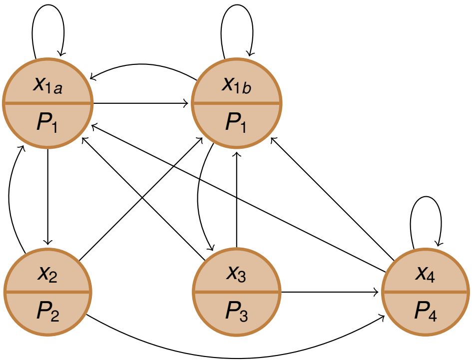
- In terms of these regions, our objective can be formulated as the existence of a control strategy driving all the points in region P_4 to region P_2 without entering region P_1.
- If we now compute the coarsest bisimulation relation between $S_{\mathcal{P}}(\Sigma)$ and $S_{\mathcal{P}}(\Sigma)$ we obtain:

$$
\begin{aligned}
P_{1a} &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \leq 2\} \\
P_{1b} &= \{(c,i,g) \in \mathbb{R}^3 \mid 2 < c_1^T A x < 10\} \\
P_2 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge c_1^T x \leq 2\} \\
P_3 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge 2 < c_1^T x < 10\} \\
P_4 &= \{(c,i,g) \in \mathbb{R}^3 \mid c_1^T A x \geq 10 \wedge c_1^T x \geq 10\}
\end{aligned}
$$

Exact finite-state abstractions for control

Revisiting the economics example

- Here is the resulting quotient system $S_{/\mathcal{P}}$.



- Does there exist a controller taking the output P_4 to the output P_2 without generating the output P_1?

Exact finite-state abstractions for control

Proving termination

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Why does partition refinement terminate for adapted sets?

Exact finite-state abstractions for control

Proving termination

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ *be a discrete-time linear control system. For any finite partition* $\mathcal{P}$ *of* $\mathbb{R}^n$ *adapted to* Σ *there exists a finite-state system bisimilar to* $S_{\mathcal{P}}(\Sigma)$*.*

- Why does partition refinement terminate for adapted sets?
- How different can two linear systems be?

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Why does partition refinement terminate for adapted sets?
- How different can two linear systems be?
- If we start with:

$$\xi(k+1) = A\xi(k) + B\nu(k) \tag{1}$$

and apply an invertible change of coordinates $\xi' = P\xi$ we obtain:

$$\xi'(k+1) = P\xi(k+1)$$

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Why does partition refinement terminate for adapted sets?
- How different can two linear systems be?
- If we start with:

$$\xi(k+1) = A\xi(k) + B\nu(k) \tag{1}$$

and apply an invertible change of coordinates $\xi' = P\xi$ we obtain:

$$\xi'(k+1) = P\xi(k+1) = PA\xi(k) + PB\nu(k)$$

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Why does partition refinement terminate for adapted sets?
- How different can two linear systems be?
- If we start with:

$$\xi(k+1) = A\xi(k) + B\nu(k) \tag{1}$$

and apply an invertible change of coordinates $\xi' = P\xi$ we obtain:

$$\xi'(k+1) = P\xi(k+1) = PA\xi(k) + PB\nu(k) = PAP^{-1}\xi'(k) + PB\nu(k).$$

Exact finite-state abstractions for control

Proving termination

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

- Why does partition refinement terminate for adapted sets?
- How different can two linear systems be?
- If we start with:

$$\xi(k+1) = A\xi(k) + B\nu(k) \tag{1}$$

and apply an invertible change of coordinates $\xi' = P\xi$ we obtain:

$$\xi'(k+1) = P\xi(k+1) = PA\xi(k) + PB\nu(k) = PAP^{-1}\xi'(k) + PB\nu(k).$$

If we further apply an invertible linear feedback $\nu = F\xi' + G\nu'$ we obtain:

$$\xi'(k+1) = (PAP^{-1} + PBF)\xi'(k) + PBG\nu'(k). \tag{2}$$

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

$$\begin{aligned} \xi(k+1) &= A\xi(k) + B\nu(k) && (3) \\ \xi'(k+1) &= (PAP^{-1} + PBF)\xi'(k) + PBG\nu'(k). && (4) \end{aligned}$$

- Systems (3) and (4) are equivalent.

Exact finite-state abstractions for control

Proving termination

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

$$\xi(k+1) = A\xi(k) + B\nu(k) \tag{3}$$

$$\xi'(k+1) = (PAP^{-1} + PBF)\xi'(k) + PBG\nu'(k). \tag{4}$$

- Systems (3) and (4) are equivalent.
- Can we choose P, F, and G to make system (4) simple?

CyPhyLab UCLA

Exact finite-state abstractions for control

Proving termination

Theorem

Let $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ be a discrete-time linear control system. For any finite partition $\mathcal{P}$ of $\mathbb{R}^n$ adapted to Σ there exists a finite-state system bisimilar to $S_{\mathcal{P}}(\Sigma)$.

$$\xi(k+1) = A\xi(k) + B\nu(k) \tag{3}$$

$$\xi'(k+1) = (PAP^{-1} + PBF)\xi'(k) + PBG\nu'(k). \tag{4}$$

- Systems (3) and (4) are equivalent.
- Can we choose P, F, and G to make system (4) simple?
- Yes, when system (3) is controllable.

Exact finite-state abstractions for control

Proving termination

Definition (Controllability)

A discrete-time linear control system $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ is controllable if for any two states $x, x' \in \mathbb{R}^n$ there exists $k \in \mathbb{N}$ and an input trajectory $\nu : \{0, 1, \ldots, k\} \to \mathbb{R}^m$ such that $\xi_{x,\nu} = x'$.

- Controllability can be checked by testing the equality:

$$\text{rank}\left[B|AB|A^2B|\ldots|A^{n-1}B\right] = n.$$

Exact finite-state abstractions for control

Proving termination

Definition (Controllability)

A discrete-time linear control system $\Sigma = (\mathbb{R}^n, \mathbb{R}^m, A, B)$ is controllable if for any two states $x, x' \in \mathbb{R}^n$ there exists $k \in \mathbb{N}$ and an input trajectory $\nu : \{0, 1, \ldots, k\} \to \mathbb{R}^m$ such that $\xi_{x,\nu} = x'$.

- Controllability can be checked by testing the equality:

$$\text{rank}\left[B|AB|A^2B|\ldots|A^{n-1}B\right] = n.$$

- Any controllable linear system can be transformed into the special Brunovsky canonical form by suitably choosing P, F, and G, e.g. ($n = 3$ and $m = 1$):

$$\begin{aligned} \xi_1(k+1) &= \xi_2(k) \\ \xi_2(k+1) &= \xi_3(k) \\ \xi_3(k+1) &= \nu(k). \end{aligned}$$

- Any controllable linear system can be transformed into the special Brunovsky canonical form, e.g. ($n = 3$ and $m = 1$):

$$\begin{aligned} \xi_1(k+1) &= \xi_2(k) \\ \xi_2(k+1) &= \xi_3(k) \\ \xi_3(k+1) &= \nu(k). \end{aligned}$$

- Consider now predicates that are unions of sets of the form $\alpha \le x_1 \le \beta$ (intervals).

- Any controllable linear system can be transformed into the special Brunovsky canonical form, e.g. ($n = 3$ and $m = 1$):

$$\begin{aligned} \xi_1(k+1) &= \xi_2(k) \\ \xi_2(k+1) &= \xi_3(k) \\ \xi_3(k+1) &= \nu(k). \end{aligned}$$

- Consider now predicates that are unions of sets of the form $\alpha \leq x_1 \leq \beta$ (intervals).
- $\text{Pre}(\alpha \leq x_1 \leq \beta) = \alpha \leq x_2 \leq \beta$,

Exact finite-state abstractions for control

Proving termination

- Any controllable linear system can be transformed into the special Brunovsky canonical form, e.g. ($n = 3$ and $m = 1$):

$$\begin{aligned} \xi_1(k+1) &= \xi_2(k) \\ \xi_2(k+1) &= \xi_3(k) \\ \xi_3(k+1) &= \nu(k). \end{aligned}$$

- Consider now predicates that are unions of sets of the form $\alpha \leq x_1 \leq \beta$ (intervals).
- $\mathrm{Pre}(\alpha \leq x_1 \leq \beta) = \alpha \leq x_2 \leq \beta$,
- $\mathrm{Pre}(\alpha \leq x_2 \leq \beta) = \alpha \leq x_3 \leq \beta$,

- Any controllable linear system can be transformed into the special Brunovsky canonical form, e.g. ($n = 3$ and $m = 1$):

$$
\begin{aligned}
\xi_1(k+1) &= \xi_2(k) \\
\xi_2(k+1) &= \xi_3(k) \\
\xi_3(k+1) &= \nu(k).
\end{aligned}
$$

- Consider now predicates that are unions of sets of the form $\alpha \leq x_1 \leq \beta$ (intervals).
- $\mathrm{Pre}(\alpha \leq x_1 \leq \beta) = \alpha \leq x_2 \leq \beta$,
- $\mathrm{Pre}(\alpha \leq x_2 \leq \beta) = \alpha \leq x_3 \leq \beta$,
- $\mathrm{Pre}(\alpha \leq x_3 \leq \beta) = \mathbb{R}^3$ since we can always choose an input u satisfying $\alpha \leq u \leq \beta$.

Exact finite-state abstractions for control

Proving termination

- Any controllable linear system can be transformed into the special Brunovsky canonical form, e.g. ($n = 3$ and $m = 1$):

$$\begin{aligned}\xi_1(k+1) &= \xi_2(k)\\ \xi_2(k+1) &= \xi_3(k)\\ \xi_3(k+1) &= \nu(k).\end{aligned}$$

- Consider now predicates that are unions of sets of the form $\alpha \leq x_1 \leq \beta$ (intervals).
- $\mathrm{Pre}(\alpha \leq x_1 \leq \beta) = \alpha \leq x_2 \leq \beta$,
- $\mathrm{Pre}(\alpha \leq x_2 \leq \beta) = \alpha \leq x_3 \leq \beta$,
- $\mathrm{Pre}(\alpha \leq x_3 \leq \beta) = \mathbb{R}^3$ since we can always choose an input u satisfying $\alpha \leq u \leq \beta$.
- Termination of the partition refinement algorithm is guaranteed for adapted sets. The cardinality of the resulting partition is bounded by $|\mathcal{P}|^{\max_r\{\mu_r\}} \leq |\mathcal{P}|^n$.

- Can we always find an adapted partition for any given specification?

Exact finite-state abstractions for control

Using adapted sets in practice

- Can we always find an adapted partition for any given specification?
- For any controllable discrete-time linear control system, the functions $f = \pm c_r^T A^l x \pm e$ defining the adapted sets can be chosen so that $(c_r^T A^l)^T$ span the state space.

- Can we always find an adapted partition for any given specification?
- For any controllable discrete-time linear control system, the functions $f = \pm c_r^T A^l x \pm e$ defining the adapted sets can be chosen so that $(c_r^T A^l)^T$ span the state space.
- In general, we have to approximate the relevant sets by adapted sets and this can be computationally demanding.

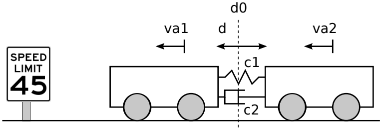
- Can we always find an adapted partition for any given specification?
- For any controllable discrete-time linear control system, the functions $f = \pm c_r^T A^l x \pm e$ defining the adapted sets can be chosen so that $(c_r^T A^l)^T$ span the state space.
- In general, we have to approximate the relevant sets by adapted sets and this can be computationally demanding.
- For safety specifications we can integrate the computation of finite-state abstractions with the synthesis of a controller (recent work with M. Rungger and M. Mazo Jr.).

Exact finite-state abstractions for control

A cruise control example

- Dynamics:

$$\dot{x} = \begin{bmatrix} 0 & -1 & 1 \\ \frac{k_s}{m} & -\frac{k_d}{m} & \frac{k_d}{m} \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$



with $x = (d, v_1, v_2) \in \mathbb{R}^3$ and $u \in \mathbb{R}$.

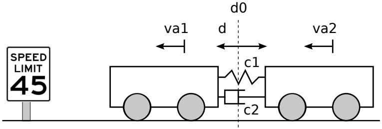
- d distance between the truck and the trailer
- v_1 velocity of the truck
- v_2 velocity of the trailer
- u acceleration

Exact finite-state abstractions for control

A cruise control example

- Dynamics:

$$\dot{x} = \begin{bmatrix} 0 & -1 & 1 \\ \frac{k_s}{m} & -\frac{k_d}{m} & \frac{k_d}{m} \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$



with $x = (d, v_1, v_2) \in \mathbb{R}^3$ and $u \in \mathbb{R}$.

  - d distance between the truck and the trailer
  - v_1 velocity of the truck
  - v_2 velocity of the trailer
  - u acceleration

- Specification:
  - compliance with speed limits v_a, v_b after at most $T \in \mathbb{N}$ time-steps
  - acceleration constraints $u \in [\underline{u}, \overline{u}]$
  - distance constraints $d \in [\underline{d}, \overline{d}]$

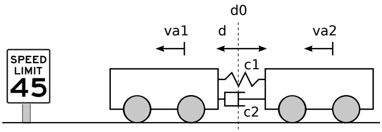
- Dynamics:

$$\dot{x} = \begin{bmatrix} 0 & -1 & 1 \\ \frac{k_s}{m} & -\frac{k_d}{m} & \frac{k_d}{m} \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

with $x = (d, v_1, v_2) \in \mathbb{R}^3$ and $u \in \mathbb{R}$.

  - d distance between the truck and the trailer
  - v_1 velocity of the truck
  - v_2 velocity of the trailer
  - u acceleration

- Specification:
  - compliance with speed limits v_a, v_b after at most $T \in \mathbb{N}$ time-steps
  - acceleration constraints $u \in [\underline{u}, \overline{u}]$
  - distance constraints $d \in [\underline{d}, \overline{d}]$



- Safe LTL formula

$$\square(U \wedge D \wedge \varphi_a \wedge \varphi_b)$$

with φ_a and φ_b given by

$$\begin{aligned} m_a &\implies \Diamond_{\leq T}(t_a \text{ W } m_b) \\ m_b &\implies \Diamond_{\leq T}(t_b \text{ W } m_a) \end{aligned}$$

  - m_i: v_i is active $i \in \{a, b\}$
  - t_i: $v_1 \leq v_i$

Exact finite-state abstractions for control

Reducing controller synthesis to safety games [Kupferman and Vardi, 2001]

- Construct the bad-prefix automaton $A_{\neg\varphi}$ from the safe LTL formula φ:

$$A_{\neg\varphi} = (Q, F, \delta, g, 2^{\mathcal{P}});$$

Exact finite-state abstractions for control

Reducing controller synthesis to safety games [Kupferman and Vardi, 2001]

- Construct the bad-prefix automaton $A_{\neg\varphi}$ from the safe LTL formula φ:

$$A_{\neg\varphi} = (Q, F, \delta, g, 2^{\mathcal{P}});$$

- Compose $A_{\neg\varphi}$ with the control system Σ:

$$S = A_{\neg\varphi} \| \Sigma;$$

Exact finite-state abstractions for control

Reducing controller synthesis to safety games [Kupferman and Vardi, 2001]

- Construct the bad-prefix automaton $A_{\neg\varphi}$ from the safe LTL formula φ:

$$A_{\neg\varphi} = (Q, F, \delta, g, 2^{\mathcal{P}});$$

- Compose $A_{\neg\varphi}$ with the control system Σ:

$$S = A_{\neg\varphi} \| \Sigma;$$

- Given the safe set $K = \bigcup_{q \in Q \setminus F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, compute its largest controlled invariant subset:

$$\mathcal{K}(K) = \left\{(q, x) \in K \mid \exists\, u \in \mathbb{R}^m,\ \mathrm{Post}_u(q, x) \subseteq K\right\}.$$

Exact finite-state abstractions for control

Reducing controller synthesis to safety games [Kupferman and Vardi, 2001]

- Construct the bad-prefix automaton $A_{\neg\varphi}$ from the safe LTL formula φ:

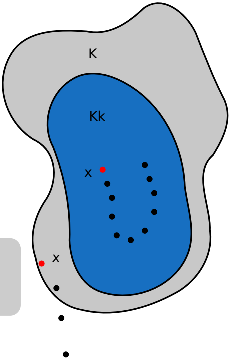
$$A_{\neg\varphi} = (Q, F, \delta, g, 2^{\mathcal{P}});$$

- Compose $A_{\neg\varphi}$ with the control system Σ:

$$S = A_{\neg\varphi} \| \Sigma;$$

- Given the safe set $K = \bigcup_{q \in Q \setminus F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, compute its largest controlled invariant subset:

$$\mathcal{K}(K) = \left\{(q, x) \in K \mid \exists\, u \in \mathbb{R}^m,\ \mathrm{Post}_u(q, x) \subseteq K\right\}.$$



Exact finite-state abstractions for control

Reducing controller synthesis to safety games [Kupferman and Vardi, 2001]

- Construct the bad-prefix automaton $A_{\neg\varphi}$ from the safe LTL formula φ:

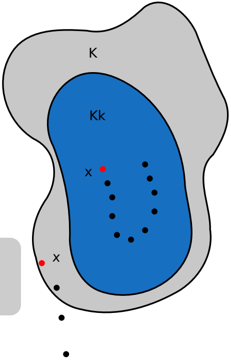
$$A_{\neg\varphi} = (Q, F, \delta, g, 2^{\mathcal{P}});$$

- Compose $A_{\neg\varphi}$ with the control system Σ:

$$S = A_{\neg\varphi} \| \Sigma;$$

- Given the safe set $K = \bigcup_{q \in Q \setminus F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, compute its largest controlled invariant subset:

$$\mathcal{K}(K) = \left\{ (q, x) \in K \mid \exists\, u \in \mathbb{R}^m, \mathrm{Post}_u(q, x) \subseteq K \right\}.$$



- We know that:

$(q, x) \in \mathcal{K}(K) \Leftrightarrow$ existence of a control strategy enforcing φ from x.

Exact finite-state abstractions for control

Computation of controlled invariant subsets

- Fixed point computation [Bertsekas, 1972]:

$$K_{j+1} = \mathrm{pre}(K_j) \cap K_j, \quad K_0 = K$$

- Fixed point computation [Bertsekas, 1972]:

$$K_{j+1} = \text{pre}(K_j) \cap K_j, \quad K_0 = K$$

- Safe set $K = \bigcup_{q \in Q \setminus F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, given by:

$$H_q = \bigcup_{i=1}^{p} H_{qi}, \quad \text{each } H_{qi} \text{ is a polytope}$$

$\Rightarrow$ each K_j is computable and the iteration is known to asymptotically converge:

$$\mathcal{K}(K) = \lim_{j \to \infty} K_j.$$

Exact finite-state abstractions for control

Computation of controlled invariant subsets: Known problems

- No termination guarantees;

- No termination guarantees;
- Set iterates K_j are not controlled invariant;

Exact finite-state abstractions for control

Computation of controlled invariant subsets: Known problems

- No termination guarantees;
- Set iterates K_j are not controlled invariant;
- Solutions for $K = \bigcup_{q \in Q \backslash F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, if H_q is convex and $|Q \backslash F| = 1$:
  - [De Santis et al., 2004]: iteration is initialized with a controlled invariant set $K_0 \subset K$;
  - [Blanchini and Miani, 2008]: modified iteration using contractive sets;
  - Several other methods based on approximations of K;

Exact finite-state abstractions for control

Computation of controlled invariant subsets: Known problems

- No termination guarantees;
- Set iterates K_j are not controlled invariant;
- Solutions for $K = \bigcup_{q \in Q \setminus F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, if H_q is convex and $|Q \setminus F| = 1$:
  - [De Santis et al., 2004]: iteration is initialized with a controlled invariant set $K_0 \subset K$;
  - [Blanchini and Miani, 2008]: modified iteration using contractive sets;
  - Several other methods based on approximations of K;
- For H_q given as union of polytopes, the iterative computation introduces combinatorial complexity!

CyPhyLab UCLA

Exact finite-state abstractions for control

Computation of controlled invariant subsets: Known problems

- No termination guarantees;
- Set iterates K_j are not controlled invariant;
- Solutions for $K = \bigcup_{q \in Q \setminus F} \{q\} \times H_q$, $H_q \subseteq \mathbb{R}^n$, if H_q is convex and $|Q \setminus F| = 1$:
  - [De Santis et al., 2004]: iteration is initialized with a controlled invariant set $K_0 \subset K$;
  - [Blanchini and Miani, 2008]: modified iteration using contractive sets;
  - Several other methods based on approximations of K;
- For H_q given as union of polytopes, the iterative computation introduces combinatorial complexity!

We can approximate K by sets adapted to the dynamics!
(Finite termination and symbolic implementation)

Exact finite-state abstractions for control

Completeness of approximation

We can under-approximate H_q by a finite union of adapted sets $\check{H}_q$.

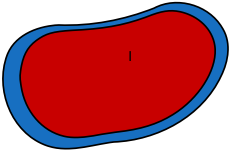
Exact finite-state abstractions for control

Completeness of approximation

We can under-approximate H_q by a finite union of adapted sets $\check{H}_q$.

We say that a set $I \subseteq \mathbb{R}^n$ is strictly inside a set $J \subseteq \mathbb{R}^n$ if there exists $\gamma > 0$ for which:

$$I + \mathcal{B}_\gamma(0) \subseteq J.$$



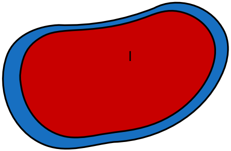
Exact finite-state abstractions for control

Completeness of approximation

We can under-approximate H_q by a finite union of adapted sets $\check{H}_q$.

We say that a set $I \subseteq \mathbb{R}^n$ is strictly inside a set $J \subseteq \mathbb{R}^n$ if there exists $\gamma > 0$ for which:

$$I + \mathcal{B}_\gamma(0) \subseteq J.$$



Theorem (Completeness [Rungger et al., 2013])

If there exists a controlled invariant set $I \subseteq \mathcal{K}(K)$ for which I_q is strictly inside $\mathcal{K}_q(K)$, then there exists an under-approximation $\check{K} = \bigcup_{q \in Q \setminus F} \{q\} \times \check{H}_q$ of K, with $\check{H}_q$ being adapted sets, such that $I \subseteq \mathcal{K}(\check{K})$.

Exact finite-state abstractions for control

Symbolic implementation

- Use binary decision diagrams (BDDs) to implement the iteration:

$$K_{j+1} = \mathrm{pre}(K_j) \cap K_j.$$

Exact finite-state abstractions for control

Symbolic implementation

- Use binary decision diagrams (BDDs) to implement the iteration:

$$K_{j+1} = \mathrm{pre}(K_j) \cap K_j.$$

  - Each set K_j is encoded by a BDD;

- Use binary decision diagrams (BDDs) to implement the iteration:

$$K_{j+1} = \mathrm{pre}(K_j) \cap K_j.$$

  - Each set K_j is encoded by a BDD;
  - The combination of the special Brunovsky normal form with adapted sets results in a simple expression for $\mathrm{pre}(B_i)$ with $B_i = [a_1^i, b_1^i] \times \ldots \times [a_n^i, b_n^i]$:

$$\mathrm{pre}(B_i) = \mathbb{R} \times [a_1^i, b_1^i] \times \ldots \times [a_{n-1}^i, b_{n-1}^i];$$

Exact finite-state abstractions for control

Symbolic implementation

- Use binary decision diagrams (BDDs) to implement the iteration:

$$K_{j+1} = \mathrm{pre}(K_j) \cap K_j.$$

  - Each set K_j is encoded by a BDD;
  - The combination of the special Brunovsky normal form with adapted sets results in a simple expression for $\mathrm{pre}(B_i)$ with $B_i = [a_1^i, b_1^i] \times \ldots \times [a_n^i, b_n^i]$:

$$\mathrm{pre}(B_i) = \mathbb{R} \times [a_1^i, b_1^i] \times \ldots \times [a_{n-1}^i, b_{n-1}^i];$$

  - Symbolical computation of $\mathrm{pre}(K_j)$ can be done by shifting and variable reordering.

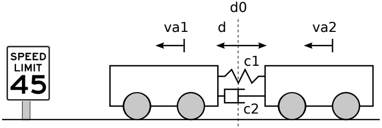
Exact finite-state abstractions for control

Revisiting the cruise control example

Problem description:

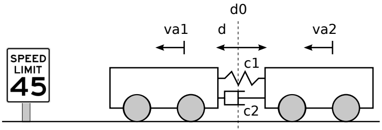
- Σ : 3 states, 1 input;
- Safe LTL formula:

$$\Box(D \wedge U \wedge \varphi_a \wedge \varphi_b \wedge \varphi_c)$$



Exact finite-state abstractions for control

Revisiting the cruise control example



Problem description:

- Σ : 3 states, 1 input;
- Safe LTL formula:

$$\Box(D \wedge U \wedge \varphi_a \wedge \varphi_b \wedge \varphi_c)$$

Parameters:

- $T \in \{2, 10\}$ number of time steps after which speed limit is enforced;
- $N \in \{10, \ldots, 13\}$ number of bits (2^N boxes) used in each dimension.

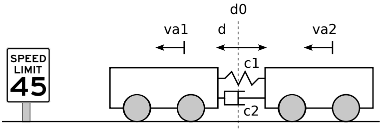
Exact finite-state abstractions for control

Revisiting the cruise control example

Problem description:

- Σ : 3 states, 1 input;
- Safe LTL formula:

$$\square(D \wedge U \wedge \varphi_a \wedge \varphi_b \wedge \varphi_c)$$



Parameters:

- $T \in \{2, 10\}$ number of time steps after which speed limit is enforced;
- $N \in \{10, \ldots, 13\}$ number of bits (2^N boxes) used in each dimension.

Error bound:

$$\hat{e} = \frac{\text{vol}\,\mathcal{K}(\hat{K}) - \text{vol}\,\mathcal{K}(\check{K})}{\text{vol}\,\mathcal{K}(\check{K})} \geq \frac{\text{vol}\,\mathcal{K}(K) - \text{vol}\,\mathcal{K}(\check{K})}{\text{vol}\,\mathcal{K}(K)}$$

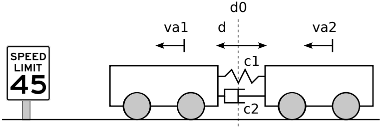
Exact finite-state abstractions for control

Revisiting the cruise control example

Problem description:

- Σ : 3 states, 1 input;
- Safe LTL formula:

$$\Box(D \wedge U \wedge \varphi_a \wedge \varphi_b \wedge \varphi_c)$$



Parameters:

- $T \in \{2, 10\}$ number of time steps after which speed limit is enforced;
- $N \in \{10, \ldots, 13\}$ number of bits (2^N boxes) used in each dimension.

Error bound:

$$\hat{e} = \frac{\mathrm{vol}\,\mathcal{K}(\hat{K}) - \mathrm{vol}\,\mathcal{K}(\check{K})}{\mathrm{vol}\,\mathcal{K}(\check{K})} \geq \frac{\mathrm{vol}\,\mathcal{K}(K) - \mathrm{vol}\,\mathcal{K}(\check{K})}{\mathrm{vol}\,\mathcal{K}(K)}$$

N\T	2		10	
	t_r	$\hat{e}$	t_r	$\hat{e}$
10	1m39s	2.31	2m40s	2.38
11	4m09s	1.01	4m31s	1.04
12	6m48s	0.58	7m52s	0.62
13	10m38s	0.43	16m01s	0.46

Exact finite-state abstractions for control

Revisiting the cruise control example: Comparison with the polyhedral approach

- Example 5.1 in [Pérez et al., 2011]:
  3 states + 2 inputs
- Workspace:
  $X = [0, 30]^3$ and $U = [0, 2]^2$
- Obstacles in the state space:
  - $O_1 = [-5, 15]^3$
  - $O_2 = [-5, 5]^3$
  - $O_3 = [-15, 10]^3$
- Obstacles in the input space:
  - $V_1 = [-3/2, 1/2]^2$
  - $V_2 = [-1/4, 1/4]^2$
  - $V_3 = [2/5, 1/5]^2$

- Specification with increasing complexity:

$$
\begin{aligned}
\varphi_0 &= \Box(X \times U) \\
\varphi_1 &= \Box((X \wedge \neg O_1) \times U) \\
\varphi_2 &= \Box(X \times (U \wedge \neg V_1)) \\
\varphi_3 &= \Box((X \wedge \neg O_1) \times (U \wedge \neg V_1)) \\
\varphi_4 &= \Box((X \wedge_{i=1}^{2} \neg O_i) \times (U \wedge_{i=1}^{2} \neg V_i)) \\
\varphi_5 &= \Box((X \wedge_{i=1}^{3} \neg O_i) \times (U \wedge_{i=1}^{2} \neg V_i)) \\
\varphi_6 &= \Box((X \wedge_{i=1}^{3} \neg O_i) \times (U \wedge_{i=1}^{3} \neg V_i))
\end{aligned}
$$

Exact finite-state abstractions for control

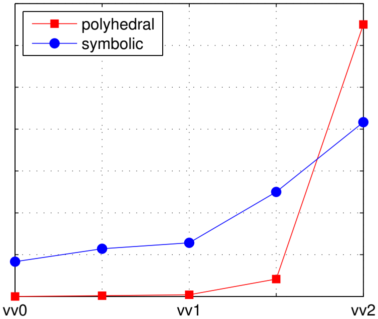
Revisiting the cruise control example: Comparison with the polyhedral approach

- Example 5.1 in [Pérez et al., 2011]:
  3 states + 2 inputs
- Workspace:
  $X = [0, 30]^3$ and $U = [0, 2]^2$
- Obstacles in the state space:
  - $O_1 = [-5, 15]^3$
  - $O_2 = [-5, 5]^3$
  - $O_3 = [-15, 10]^3$
- Obstacles in the input space:
  - $V_1 = [-3/2, 1/2]^2$
  - $V_2 = [-1/4, 1/4]^2$
  - $V_3 = [2/5, 1/5]^2$

- Specification with increasing complexity:

$$\begin{aligned}
\varphi_0 &= \Box(X \times U) \\
\varphi_1 &= \Box((X \wedge \neg O_1) \times U) \\
\varphi_2 &= \Box(X \times (U \wedge \neg V_1))
\end{aligned}$$

- Computation times:

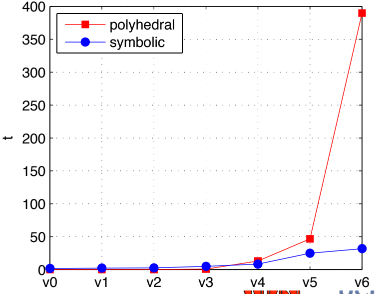


- Example 5.1 in [Pérez et al., 2011]:
  3 states + 2 inputs
- Workspace:
  $X = [0, 30]^3$ and $U = [0, 2]^2$
- Obstacles in the state space:
  - $O_1 = [-5, 15]^3$
  - $O_2 = [-5, 5]^3$
  - $O_3 = [-15, 10]^3$
- Obstacles in the input space:
  - $V_1 = [-3/2, 1/2]^2$
  - $V_2 = [-1/4, 1/4]^2$
  - $V_3 = [2/5, 1/5]^2$

- Specification with increasing complexity:

$$\begin{aligned}\varphi_0 &= \Box(X \times U)\\ \varphi_1 &= \Box((X \wedge \neg O_1) \times U)\\ \varphi_2 &= \Box(X \times (U \wedge \neg V_1))\end{aligned}$$

- Computation times:



Observations

- We can handle 5 or 6 continuous variables (state-of-the-art) but there is still room for improvement.

Observations

- We can handle 5 or 6 continuous variables (state-of-the-art) but there is still room for improvement.
- We recently discovered other classes of sets for which we can construct finite-state bisimulations. Still work in progress;

Observations

- We can handle 5 or 6 continuous variables (state-of-the-art) but there is still room for improvement.
- We recently discovered other classes of sets for which we can construct finite-state bisimulations. Still work in progress;
- Can we generalize these techniques to non-linear systems?

Observations

- We can handle 5 or 6 continuous variables (state-of-the-art) but there is still room for improvement.
- We recently discovered other classes of sets for which we can construct finite-state bisimulations. Still work in progress;
- Can we generalize these techniques to non-linear systems?
- No!

Observations

- We can handle 5 or 6 continuous variables (state-of-the-art) but there is still room for improvement.
- We recently discovered other classes of sets for which we can construct finite-state bisimulations. Still work in progress;
- Can we generalize these techniques to non-linear systems?
- No! But the approximate finite-state abstractions to be discussed in the next lecture do generalize to nonlinear systems.

References and more details

Bertsekas, D. (1972).
Infinite time reachability of state-space regions by using feedback control.
IEEE Transactions on Automatic Control, 17:604–613.

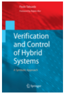
Blanchini, F. and Miani, S. (2008).
Set-Theoretic Methods in Control.
Systems & Control: Foundations & Applications. Birkhäuser.

De Santis, E., Di Benedetto, M. D., and Berardi, L. (2004).
Computation of maximal safe sets for switching systems.
IEEE Transactions on Automatic Control, 49:184–195.

Kupferman, O. and Vardi, M. Y. (2001).
Model checking of safety properties.
Formal Methods in System Design, 19:291–314.

Pérez, E., Ariño, C., Blasco, F. X., and Martínez, M. A. (2011).
Maximal closed loop admissible set for linear systems with non-convex polyhedral constraints.
Journal of Process Control, pages 529 – 537.

Rungger, M., Mazo Jr., M., and Tabuada, P. (2013).
Specification-guided controller synthesis for linear systems and safe linear-time temporal logic.
In *Proc. of Hybrid Systems: Computation and Control*.



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