Data-Driven Abstraction of Monotone Systems

Anas Makdesi, Antoine Girard, Laurent Fribourg

Proceedings of Machine Learning Research vol 144:1–12, 2021

Data-Driven Abstraction of Monotone Systems *

Anas Makdesi ANAS.MAKDESI@L2S.CENTRALESUPELEC.FR

Antoine Girard ANTOINE.GIRARD@L2S.CENTRALESUPELEC.FR
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des Signaux et Systèmes
91190, Gif-sur-Yvette, France

Laurent Fribourg FRIBOURG@LSV.FR
Université Paris-Saclay, CNRS, ENS Paris-Saclay
Laboratoire Méthodes Formelles
91190, Gif-sur-Yvette, France

Abstract

In this paper, we introduce an approach for the data-driven abstraction of monotone dynamical systems. First, we introduce a set-valued simulating map, which over-approximates the dynamics of an unknown monotone system, using only a set of transitions generated by it. We establish the minimality of the introduced simulating map. Then, we show that the system, with this map as its transition relation, is equivalent (in the sense of alternating bisimulation) to a finite-state system. This equivalence enables the use of well-established symbolic control techniques to synthesize controllers. We show the effectiveness of the approach on a safety controller synthesis problem.

Keywords: Monotone transition systems, data-driven control, symbolic control, abstraction.

1. Introduction

Abstraction-based synthesis techniques have been gaining momentum in recent years (see e.g. Tabuada (2009), Belta et al. (2017) and the references therein). Using those techniques, we can build a finite-state approximation of a dynamical system, called "symbolic abstraction". Then, controllers for the finite-state approximation are synthesized using algorithmic approaches adopted from the fields of supervisory control (Ramadge and Wonham (1987)), or formal methods (Bloem et al. (2012)). The synthesized controllers can address broad types of specifications such as safety and reachability (Girard (2012)), behaviors described by linear temporal logic (Baier and Katoen (2008), Belta et al. (2017)) or finite automata (Pola and Di Benedetto (2019), Sinyakov and Girard (2019)). Those specifications go beyond the traditional properties, such as stability, to describe a desired behavior of the controlled system over time. The synthesized controllers can then be refined to controllers for the original system if some behavioral relationship relates the symbolic abstraction with the original system. In such cases, the refined controllers are "correct by construction". An example of such a relation is the alternating (bi)-simulation relation (Tabuada (2009)), or the feedback refinement relation (Reissig et al. (2016)).

Although symbolic control is essentially a model-driven technique, we present in this paper a data-driven approach to calculate the symbolic abstraction. Starting from a set of transitions

* This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 725144).

generated from an unknown monotone system, we find a simulating map over-approximating the dynamics of any system capable of producing these transitions. This simulating map is minimal in the sense that it is included in any other map doing the same work. We then calculate a symbolic abstraction of the system represented by this simulating map, and we prove the existence of alternating bisimulation relation between them.

Similar work to compute symbolic abstractions from data has been done in (Sadraddini and Belta (2018), Hashimoto et al. (2020)), but while they require bounds on the system continuity in a Lipschitz sense, our method does not have this requirement. We rather study the class of monotone systems and build our data-driven abstraction based only on the monotonicity property of the system.

Monotone systems can be found in many applications such as power networks (Zonetti et al. (2019)), traffic networks (Kim et al. (2017)), and biomedical systems (Angeli and Sontag (2003)). Meyer et al. (2015) applied symbolic control methods on a monotone temperature regulation system by only sampling the system dynamics on a given grid of states and inputs. Saoud et al. (2019) presented an efficient symbolic control synthesis technique for monotone systems, which was applied on a vehicle platooning problem.

The paper is organized as follows. In Section 2, some required preliminaries are provided. In Section 3, we describe an approach to compute a set-valued map that over-approximates the true system dynamics. In Section 4, we show that the dynamical system induced by this map is alternatingly bisimilar to a finite-state abstraction. Finally, In Section 5, we show the effectiveness of our approach by studying a cruise control problem.

2. Preliminaries

In this section, we define some notations and introduce the class of monotone dynamical systems under consideration in this paper.

2.1. Notation

$\mathbb{R}$, $\mathbb{R}_0^+$, and $\mathbb{N}$ denote the sets of real, non-negative real, and natural numbers, respectively. The empty set is denoted by $\emptyset$. Given a set X, we denote 2^X to the set of subsets of X. We define the partial order $\preceq$ on $\mathbb{R}^n$ as follows. Let $x = (x_1, \dots, x_n)$, $x' = (x'_1, \dots, x'_n)$ in $\mathbb{R}^n$, we say that:

$$x \preceq x' \iff (x_i \leq x'_i,\ \forall i = 1, \dots, n).$$

If $x \preceq x'$ then $x' \succeq x$. A relation $R \subseteq X \times Y$ is identified with the set-valued map $R : X \to 2^Y$ where $R(x) = \{y \in Y | (x, y) \in R\}$. The domain of R is $dom(R) = \{x \in X \mid R(x) \neq \emptyset\}$. The Cartesian product of the indexed sets $(X_i)_{i \in I}$ is denoted by $\prod_{i \in I} X_i$.

2.2. Monotone systems

First, let us consider a discrete-time dynamical system of the form:

$$x^+ \in F(x, u), x \in X, u \in U \tag{1}$$

where $F : X \times U \to 2^X$ is a nonempty set-valued map, $X \subseteq \mathbb{R}^n$, $U \subseteq \mathbb{R}^m$. We say that F is deterministic if for all $x \in X$, $u \in U$, $F(x, u)$ is a singleton (in this case we identify the set-valued

map F with the classical map $F : X \times U \to X$), otherwise it is said to be non-deterministic. We consider a partial order on the set of states $\mathbb{R}^n$ and the set of inputs $\mathbb{R}^m$.

Definition 1 *F is monotone if for all $x, x' \in X$, $u, u' \in U$ with $x \preceq x'$, $u \preceq u'$,*

$$\forall y \in F(x,u),\ \exists y' \in F(x',u'),\ y \preceq y',\ \textit{and}$$
$$\forall y' \in F(x',u'),\ \exists y \in F(x,u),\ y \preceq y'.$$

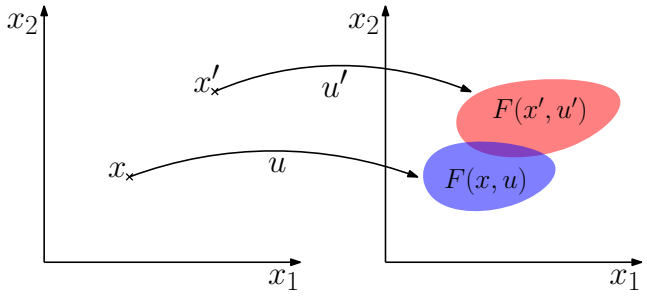


Figure 1: The set-valued image of two points, one is larger than the other and the map is monotone

Figure 1 shows a set-valued image of two points. Under the previous monotonicity definition, we do not require all the points in the image of x' to be larger than the points in the image of x. In the case where F is deterministic, Definition 1 becomes:

$$x, x' \in X, u, u' \in U, x \preceq x', u \preceq u' \implies F(x,u) \preceq F(x',u')$$

which resembles the ordinary definition of monotone functions.

3. Learning monotone systems from data

In this section, we present a data-driven approach to compute an optimal and guaranteed over-approximation of an unknown monotone map, using only a given set of transitions generated by it.

3.1. Formulation

We are given a set of transitions $\mathcal{D} = \{(x_k, u_k, x'_k) |\ k = 1, \dots, N\}$, this data has been generated by an unknown deterministic and monotone map $\tilde{F} : X \times U \to X$.

Definition 2 *A deterministic map $F : X \times U \to X$ is consistent with data $\mathcal{D}$ if:*

- *F is monotone;*
- *$F(x_k, u_k) = x'_k$, $k = 1, \dots, N$.*

Let us remark that the true unknown map $\tilde{F}$ is consistent with the data. However, there are generally an infinity of maps consistent with the data $\mathcal{D}$. We denote the set of consistent maps by $C(\mathcal{D})$. We aim at computing a set-valued map that captures all of them.

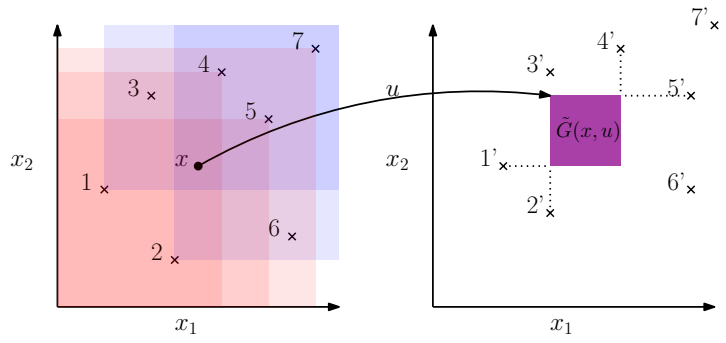


Figure 2: Data points (left) and their successors (right). On the left, the point x is smaller than the points $K^+ = \{4, 5, 7\}$ and bigger than the points $K^- = \{1, 2\}$. On the right, the set-valued map $\tilde{G}(x, u)$ is represented by the purple area.

Definition 3 *A set-valued map $G : X \times U \to 2^X$ simulates $\mathcal{D}$ if:*

- *It is monotone;*
- *For all $F \in C(\mathcal{D})$, for all $x \in X$, $u \in U$, $F(x, u) \in G(x, u)$.*

A map G, simulating $\mathcal{D}$, captures the true unknown map $\tilde{F}$: for all $x \in X$, $u \in U$, $\tilde{F}(x, u) \in G(x, u)$. There are generally an infinity of maps that simulates $\mathcal{D}$. We denote the set of simulating maps by $S(\mathcal{D})$.

Definition 4 *$\tilde{G} : X \times U \to 2^X$ is the minimal simulating map of $\mathcal{D}$ if:*

- *It simulates $\mathcal{D}$;*
- *For all $G \in S(\mathcal{D})$, for all $x \in X$, $u \in U$, $\tilde{G}(x, u) \subseteq G(x, u)$.*

The following proposition provides the expression of the minimal simulating map. Figure 2 illustrate the proposed map.

Proposition 5 *There exists a unique minimal simulating map of $\mathcal{D}$ given by*

$$\tilde{G}(x,u) = \left(\bigcap_{k \in K^-(x,u)} \{y \in X |\ x'_k \preceq y\} \right) \cap \left(\bigcap_{k \in K^+(x,u)} \{y \in X |\ y \preceq x'_k\} \right)$$

where

$$K^-(x,u) = \{k \mid x_k \preceq x, u_k \preceq u\},\ K^+(x,u) = \{k \mid x \preceq x_k, u \preceq u_k\}.$$

Proof Let us start with existence. Let $x, x^\star \in X$, $u, u^\star \in U$ with $x \preceq x^\star$, $u \preceq u^\star$. We define

$$\underline{y} = (\max_{k \in K^-(x,u)} x'_{k,1}, \ldots, \max_{k \in K^-(x,u)} x'_{k,n}),$$

where $x'_{k,i}$ is the i_{th} component of the k_{th} data point's successor. The point $\underline{y}$ represent the smallest point in the image $\tilde{G}(x, u)$ because it is the smallest point in the intersection $\bigcap_{k \in K^-(x,u)} \{y \in$

$X|\ x'_k \preceq y\}$. For example, in the case of Figure 2 the point $\underline{y}$ is $(x'_{2,1}, x'_{1,2})$. $\underline{y}$ belongs to $\tilde{G}(x,u)$ and because the partial order relation is transitive, we can say that $x_k \preceq x \preceq x^\star$ and $u_k \preceq u \preceq u^\star$ for all $k \in K^-(x,u)$, which mean that the set $\bigcap_{k\in K^-(x^\star,u^\star)}\{y \in X|\ x'_k \preceq y\}$ is included in the set $\bigcap_{k\in K^-(x,u)}\{y \in X|\ x'_k \preceq y\}$. Therefore, we now have:

$$\forall y^\star \in \tilde{G}(x^\star,u^\star), y^\star \succeq \underline{y} \in \tilde{G}(x,u). \tag{2}$$

Let us also define the point

$$\overline{y^\star} = (\min_{k\in K^+(x^\star,u^\star)} x'_{k,1}, \dots, \min_{k\in K^+(x^\star,u^\star)} x'_{k,n}).$$

$\overline{y^\star}$ is largest point in $\tilde{G}(x^\star,u^\star)$ and we know that $x \preceq x^\star \preceq x_k$ and $u \preceq u^\star \preceq u_k$ for all $k \in K^+(x^\star,u^\star)$. In the same way as above we can say:

$$\forall y \in \tilde{G}(x,u), y \preceq \overline{y^\star} \in \tilde{G}(x^\star,u^\star). \tag{3}$$

From (3) and (2) $\tilde{G}$ is monotone.

For all $F \in C(\mathcal{D})$, for all $x \in X$, $u \in U$, F is monotone which implies that $F(x,u) \succeq x'_k$ for all k such that $k \in K^-(x,u)$ and $F(x,u) \preceq x'_k$ for all k such that $k \in K^+(x,u)$. Therefore, $F(x,u) \subseteq \tilde{G}(x,u)$. Added to the fact that $\tilde{G}$ is monotone, we now have that $\tilde{G}$ simulates $\mathcal{D}$.

Then, we want to prove that for all $G \in S(\mathcal{D})$, for all $x \in X$, $u \in U$, $\tilde{G}(x,u) \subseteq G(x,u)$, so let us assume that it is wrong and that there exists $x^\star \in X$, $u^\star \in U$ such that there is $y \in \tilde{G}(x^\star,u^\star)$ but $y \notin G(x^\star,u^\star)$. Let us consider the deterministic and monotone map F satisfying $F(x_k,u_k) = x'_k$ for all $k = 1,\dots,N$ and $F(x^\star,u^\star) = y$. Then, $F \in C(\mathcal{D})$ but $F(x^\star,u^\star) \nsubseteq G(x^\star,u^\star)$, which means that G does not simulate $\mathcal{D}$, and we reach a contradiction.

We now prove uniqueness. If $\tilde{G}_0$ and $\tilde{G}_1$ both are minimal simulating map, then $\tilde{G}_0(x,u) \subseteq \tilde{G}_1(x,u)$ and $\tilde{G}_1(x,u) \subseteq \tilde{G}_0(x,u)$ which mean $\tilde{G}_1(x,u) = \tilde{G}_0(x,u)$ for all $x \in X$, $u \in U$. ■

It is worth noting that because $\tilde{G}(x,u)$ is the intersection of areas bounded from above and from below by the successors of the data points, adding new data points will make $\tilde{G}(x,u)$ more resembling to the true map $\tilde{F}$. This can be expressed in following property:

$$\mathcal{D} \subseteq \mathcal{D}' \Rightarrow \tilde{G}_{\mathcal{D}'} \subseteq \tilde{G}_{\mathcal{D}}$$

where $\tilde{G}_{\mathcal{D}}$ and $\tilde{G}_{\mathcal{D}'}$ denote the minimal simulating map of $\mathcal{D}$ and $\mathcal{D}'$ respectively.

3.2. Partitioning of the sets of states and inputs

Let us assume that sets of states and inputs are rectangular: $X = [a_1,b_1] \times \cdots \times [a_n,b_n]$ and $U = [c_1,d_1] \times \cdots \times [c_m,d_m]$. We can state the following result:

Proposition 6 *There exist finite rectangular partitions $(X_q)_{q\in Q}$, $(U_p)_{p\in P}$ of X and U, a finite collection $(Y_{q,p})_{q\in Q,p\in P}$ of rectangular subsets of X such that for all $x \in X_q$, $u \in U_p$, $\tilde{G}(x,u) = Y_{q,p}$.*

Proof First, let us introduce the following grid: on each axis of the state space, e.g., axis $i, i \in \{1,\dots,n\}$, we sort the corresponding components of all the data points (i.e., sort $x_{k,i}, k = 1,\dots,N$), meaning that on each axis, we have $x_{k_i^1,i} \leq \cdots \leq x_{k_i^N,i}$. The sorted values on the axes of the state

space define a grid. Inputs from the data set define a similar grid on the input space. Those grids will enclose hyperrectangle cells. Later, we will see that the value of the map $\tilde{G}$ will stay the same for all the points inside a hyperrectangle in the state space, under any input inside a particular hyperrectangle in the input state. To establish the finite rectangular partitions, we should also address the points of the grids (boundaries between the cells). That is what we are doing next. To describe the partition we will use two sets of symbols $Q = Q' \cup Q''$, where $Q' = \{1, \dots, N\}$ and $Q'' = \{1, \dots, N-1\}^n$. The first set of symbols Q' is dedicated to the data points, meaning:

$$\forall q \in Q', X_q = \{x_q\}.$$

A symbol $q \in Q''$ from the second set is a tuple $q = (q_1, \dots, q_n)$; It determines the hyperrectangle position on each axis (i.e., q_i is an index that refers to one of the values in the list of sorted data points' i^{th} components). q allows us to define (the lower and upper values of the hyperrectangle) $\underline{x_q} = (x_{k_1^{q_1},1}, \dots, x_{k_n^{q_n},n})$ and $\overline{x_q} = (x_{k_1^{q_1+1},1}, \dots, x_{k_n^{q_n+1},n})$. Now we have all the parts to define the partition:

$$\forall q \in Q'', X_q = \Big(\prod_{i=1}^{N} X_q^i\Big) \setminus X_{\mathcal{D}}$$

where $X_{\mathcal{D}} = \bigcup_{k=1}^{N} \{x_k\}$, and X_q^i is defined as follows:

$$X_q^i = \begin{cases} [x_{k_i^{q_i},i}, x_{k_i^{q_i+1},i}] & \text{if } x_{k_i^{q_i}} \preceq \underline{x_q} \text{ and } \overline{x_q} \preceq x_{k_i^{q_i+1}} \\ [x_{k_i^{q_i},i}, x_{k_i^{q_i+1},i}) & \text{if } x_{k_i^{q_i}} \preceq \underline{x_q} \text{ and } \overline{x_q} \not\preceq x_{k_i^{q_i+1}} \\ (x_{k_i^{q_i},i}, x_{k_i^{q_i+1},i}] & \text{if } x_{k_i^{q_i}} \not\preceq \underline{x_q} \text{ and } \overline{x_q} \preceq x_{k_i^{q_i+1}} \\ (x_{k_i^{q_i},i}, x_{k_i^{q_i+1},i}) & \text{if } x_{k_i^{q_i}} \preceq \underline{x_q} \text{ and } \overline{x_q} \preceq x_{k_i^{q_i+1}} \end{cases}$$

An example to illustrate the partition cand be found shortly after the proof. After we define U_p in the same way, we have under our definition for the partial order, for all x in X_q and for all u in U_p, $K^-(x,u)$ and $K^+(x,u)$ remain constant. Therefore, $\tilde{G}(x,u)$ will have the same value $\tilde{G}(x,u) = Y_{q,p}$ for all x in X_q and for all u in U_p. $Y_{q,p}$ will either be a single point $Y_{q,p} = \{x'_k\}$ in the case where $X_q = \{x_k\}$ and $U_p = \{u_k\}$ such that $(x_k, u_k, x'_k) \in \mathcal{D}$ or it will be define by the box $Y_{q,p} = [\underline{y}, \overline{y}]$ where $\underline{y} = (\max_{k \in K^-(x,u)} x'_{k,1}, \dots, \max_{k \in K^-(x,u)} x'_{k,n})$ and $\overline{y} = (\min_{k \in K^+(x,u)} x'_{k,1}, \dots, \min_{k \in K^+(x,u)} x'_{k,n})$ elsewhere. ■

Figure 3 offers an illustrative example of a finite rectangular partition (in the state space) based on four data points. We examine the cell which has the symbol $q \in Q''$ with $q = (2,1)$. We have $X_q = (x_{1,1}, x_{4,1}] \times [x_{2,2}, x_{3,2}]$. $x_{1,1}$ is excluded because $x_1 \not\preceq \underline{x_q}$, whereas $x_{4,1}, x_{2,2}$, and $x_{3,2}$ are included because $x_2 \preceq \underline{x_q}$, $\overline{x_q} \preceq x_3$, and $\overline{x_q} \preceq x_4$.

Remark 7 *In the following sections and in our numerical tests we will actually make the output $\tilde{G}(x,u) = Y_{q,p}$ for all the points x, u in the sets $X_q = [x_{k_1^{q_1},1}, x_{k_1^{q_1+1},1}) \times \cdots \times [x_{k_n^{q_n},n}, x_{k_n^{q_n+1},n})$ and $U_p = [u_{k_1^{p_1},1}, u_{k_1^{p_1+1},1}) \times \cdots \times [u_{k_m^{p_m},m}, u_{k_m^{p_m+1},m})$, $q \in Q = \{1, \dots, N-1\}^n$, $p \in P = \{1, \dots, N-1\}^m$. With this new definition of $\tilde{G}(x,u)$, $\tilde{G}$ is not strictly speaking the minimal simulating map, but this new simulating map is easier to deal with, and it differs from the minimal simulation map only on a set of measure 0.*

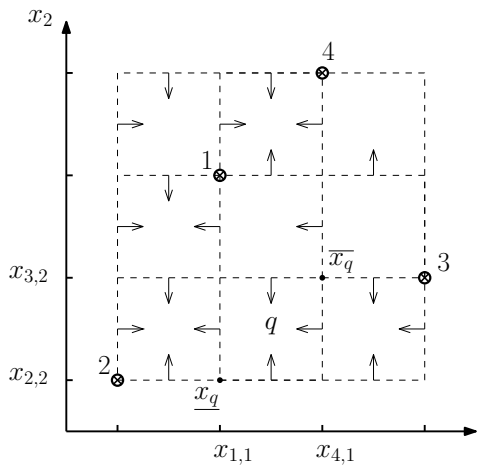


Figure 3: The partitioning of the state space based on the data points. Arrows indicate to which set, the boundaries between adjacent sets, belong.

In order to compute the sets $(X_q)_{q\in Q}$, $(U_p)_{p\in P}$ and $(Y_{q,p})_{q\in Q,p\in P}$ from the data set $\mathcal{D}$ we should follow those steps:

1. We sort the first component of our data points, then the second component, and so on till we reach the n_{th} component.
2. We do the same for input data till we reach the m_{th} component.
3. We partition the sets of states and inputs according to the sorted components.
4. We calculate $\tilde{G}(x,u)$ for every set.

4. Bisimilar finite abstraction

In this section, we show the equivalence (in the sense of alternating bisimulation) between the system defined by the calculated minimal simulating map and a finite state transition system.

Definition 8 *A transition system Σ is a tuple $\Sigma = (X, U, F, Y, H)$, where X is a set of states, U is a set of inputs, $F : X \times U \to 2^X$ is a transition relation, Y is a set of outputs, and $H : X \to Y$ is an output map.*

Σ is finite if X and U are finite. Now let us consider two transition systems $\Sigma_i = (X_i, U_i, F_i, Y_i, H_i)$, $i = 1, 2$. We assume that the systems are observed over the same set of outputs $Y = Y_1 = Y_2$.

Definition 9 *$R \subseteq X_1 \times X_2$ is an alternating bisimulation relation between Σ_1 and Σ_2 if for all $(x_1, x_2) \in R$,*

1. $H_1(x_1) = H_2(x_2)$
2. $\forall u_1 \in U_1$, $\exists u_2 \in U_2$, $\forall x_2' \in F_2(x_2, u_2)$, $\exists x_1' \in F_1(x_1, u_1)$ *such that* $(x_1', x_2') \in R$

3. $\forall u_2 \in U_2$, $\exists u_1 \in U_1$, $\forall x_1' \in F_1(x_1, u_1)$, $\exists x_2' \in F_2(x_2, u_2)$ *such that* $(x_1', x_2') \in R$

Σ_1 *and* Σ_2 *are alternatingly bisimilar if*

1. $\forall x_1 \in X_1$, $\exists x_2 \in X_2$, *such that* $(x_1, x_2) \in R$

2. $\forall x_2 \in X_2$, $\exists x_1 \in X_1$, *such that* $(x_1, x_2) \in R$

In the following, let us consider that $X_1 = X$, $U_1 = U$, $F_1 = \tilde{G}$ as defined in Remark 7. Let $X_2 = Q$, $U_2 = P$ as in Remark 7, and let us consider the transition map $F_2 : Q \times P \to 2^Q$ defined by

$$F_2(q, p) = \{q' \in Q |\ X_{q'} \cap Y_{q,p} \neq \emptyset\}.$$

We consider the output set $Y = Q$ and output maps given by $H_1(x) = q$ if $x \in X_q$ and $H_2(q) = q$. Let us remark that Σ_2 is a symbolic dynamical system (i.e. with finite set of states and inputs). In addition, let us assume that $X = \bigcup_{q \in Q} X_q$ and $U = \bigcup_{p \in P} U_p$.

Theorem 10 Σ_1 *and* Σ_2 *are alternatingly bisimilar and the alternating bisimulation relation is given by*

$$R = \{(x, q) \in X \times Q |\ x \in X_q\}.$$

Proof We will prove the statement for all $x_1 \in X_1$, there exists $x_2 \in X_2$, such that $(x_1, x_2) \in R$. The proof of the second statement is similar, and is therefore omitted. For all $x_1 \in X_1$, x_1 should belong to one of the cells in the partition $x_1 \in X_q$, so $(x_1, x_2) \in R$ if we take $x_2 = q$. Moreover, we will have:

1. From the definition of Σ_1 and Σ_2, we have $H_1(x_1) = H_2(x_2)$;

2. For all $u_1 \in U_1$, there exists $p \in P$ such that $u_1 \in U_p$. So let us take $u_2 = p$. For all $x_2' \in F_2(x_2, u_2)$, we are sure that there exists $x_1' \in F_1(x_1, u_1) = \tilde{G}(x_1, u_1) = Y_{q,p}$ such that $x_1' \in X_{x_2'}$ because we know, from the definition of F_2, that for any $x_2' \in F_2(x_2, u_2)$ we have $X_{x_2'} \cap Y_{q,p} \neq \emptyset$, which means $(x_1', x_2') \in R$;

3. For all $u_2 = p \in P$, there exists $u_1 \in U_p$, and for all $x_1' \in F_1(x_1, u_1)$, we know that $F_1(x_1, u_1) = Y_{q,p}$, and $Y_{q,p} \subseteq X$ which means that there exists x_2' such that $X_{x_2'} \cap Y_{q,p} \neq \emptyset$. In another way, there exists $x_2' \in F_2(x_2, u_2)$ such that $x_1' \in X_{x_2'}$. In the end we have $(x_1', x_2') \in R$.

■

As a consequence of the previous Theorem, it is optimal to use the abstraction provided by the finite-state transition system Σ_2 to synthesize a controller. This controller will work for all the systems consistent with the data; the finite-state transition system is alternatingly bisimilar to a system whose dynamics is simulating all of them. Not finding a controller using Σ_2 means that there is an uncontrollable system consistent with the data; any other map simulating the data will contain this uncontrollable system. Therefore, this is the best result we can find under the monotonicity condition.

5. Controller synthesis for safety specifications

To test the validity of the learnt abstraction, we want to use it to synthesize a controller for a safety problem, a commonly addressed problem in the field of symbolic control. Given a system $\Sigma = (X, U, F, Y, H)$ where $F = \tilde{G}$, and a safety specification $X^s \subseteq X$, we want to find a controller to keep the trajectories of this system inside the set X^s. This can be done by computing a safe controlled invariant defined as follows:

Definition 11 *A set $I \subseteq X$ is a safe controlled invariant for the system Σ and the safe set X^s if it satisfies:*

- $I \subseteq X^s$
- $\forall x \in I$ *and* $\exists u \in U, F(x, u) \subseteq I$

We can use the symbolic abstraction presented in the previous section and the iterative algorithm described in Tabuada (2009) to find the maximal safe controlled invariant.

5.1. Numerical implementation (cruise control problem)

Let us consider a model with two vehicles moving in one lane on an infinite straight road. The leader has a fixed velocity v_c while the follower is controllable. A discrete-time approximation of this model is given by equations:

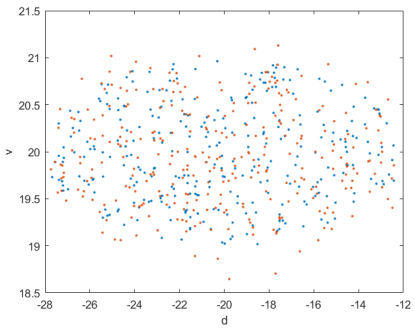
$$\begin{aligned} d_{k+1} &= d_k + (v_k - v_c)T_0 \\ v_{k+1} &= v_k + \alpha(u_k, v_k)T_0, \end{aligned} \tag{4}$$

Here u_k is the control input which is the torque applied to the wheels. d_k is the signed distance between the vehicles.

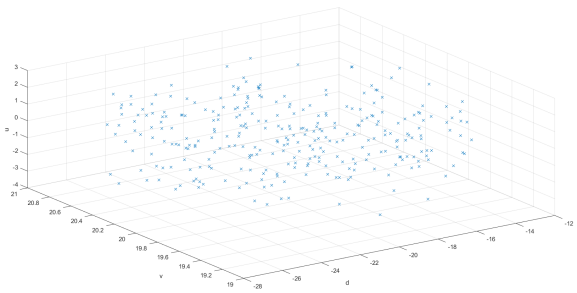
$$\alpha(u, v) = u - M^{-1}(f_0 + f_1 v + f_2 v^2);$$

The vector of parameters $f = (f_0, f_1, f_2) \in \mathbb{R}^3_+$ describes the road friction and vehicle aerodynamics whose numerical values are taken from Nilsson et al. (2015): $f_0 = 52\ N$, $f_1 = 1.2567\ N\ s/m$, $f_2 = 0.4342 N\ s^2/m^2$. For the rest parameters we chose:$v_c = 20\ m/s, M = 1370\ kg, T_0 = 0.1\ s$. This model is **only** used to generate the data. We wanted the follower car to stay in a distance between $10\ m$ and $30\ m$ from the leader, while maintaining a speed between $19\ m/s$ and $21\ m/s$. Let us remark that over these ranges of values the system (4) is monotone. We sampled randomly 300 points inside this safe set. We sampled also 300 random input values. We calculated the successors from the model given by (4). Data points, successors, and inputs are shown in Figure 4.

Calculating the abstraction took $923\ s$ to be finished (CPU: 2.9 GHz Intel Core i7, RAM: 8 Go 2133 MHz DDR34, Matlab R2020b). It took $341\ s$ to calculate the invariant set. The red area in Figure 5 represents the invariant set that we found. It should be noted that the size of the abstraction grows polynomially with the number of data points. In the example there are two states $n_x = 2$ and one input $n_u = 1$. The number of sets in the partitioning is $N^{n_x+n_u} = N^3$ where N is the number of data points. This makes using a big number of data points computationally difficult.



(*a*) Data points in blue and their successors in red.



(*b*) Inputs generated for data points.

Figure 4: Generated Transitions

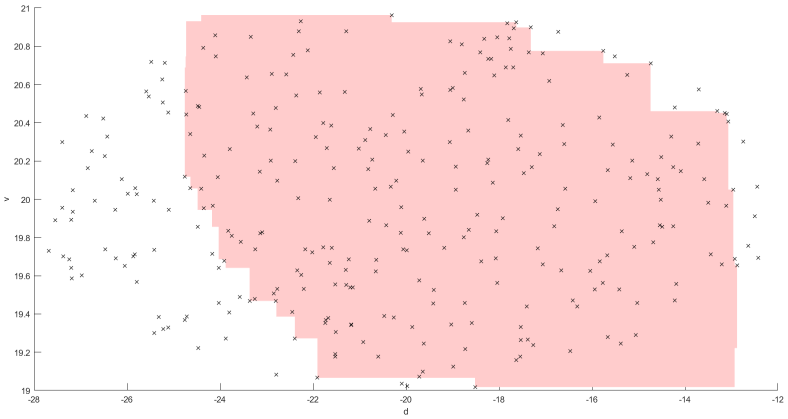


Figure 5: The calculated invariant set along with the sampled data points.

6. Conclusion

An algorithm for data-driven abstraction was introduced. To find the abstraction, we first calculated a map that minimally over-approximates the system dynamics. We showed that this abstraction contains all the information we can acquire from the data and the monotonicity condition. We demonstrated the effectiveness of our approach on a safety controller synthesis problem. The approach suffers from a scalability problem as the abstraction size grows polynomially with the number of data points.

As for the future, a way to deal with uncertainties and noises in the data points should be studied. Also, The scalability problem should be addressed maybe with the help of data aggregation techniques. The authors also want to investigate the use of data collected on-line to improve the calculated controller.

References

David Angeli and Eduardo D Sontag. Monotone control systems. *IEEE Transactions on automatic control*, 48(10):1684–1698, 2003.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press, 2008.

Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 89. Springer, 2017.

Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive (1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

Antoine Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.

Kazumune Hashimoto, Adnane Saoud, Masako Kishida, Toshimitsu Ushio, and Dimos Dimarogonas. Learning-based safe symbolic abstractions for nonlinear control systems. *arXiv preprint arXiv:2004.01879*, 2020.

Eric S Kim, Murat Arcak, and Sanjit A Seshia. Symbolic control design for monotone systems with directed specifications. *Automatica*, 83:10–19, 2017.

Pierre-Jean Meyer, Antoine Girard, and Emmanuel Witrant. Safety control with performance guarantees of cooperative systems using compositional abstractions. *IFAC-PapersOnLine*, 48(27): 317–322, 2015.

Petter Nilsson, Omar Hussien, Ayca Balkan, Yuxiao Chen, Aaron D Ames, Jessy W Grizzle, Necmiye Ozay, Huei Peng, and Paulo Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307, 2015.

Giordano Pola and Maria Domenica Di Benedetto. Control of cyber-physical-systems with logic specifications: a formal methods approach. *Annual Reviews in Control*, 47:178–192, 2019.

Peter J Ramadge and W Murray Wonham. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1):206–230, 1987.

Gunther Reissig, Alexander Weber, and Matthias Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2016.

Sadra Sadraddini and Calin Belta. Formal guarantees in data-driven model identification and control synthesis. In *International Conference on Hybrid Systems: Computation and Control*, pages 147–156, 2018.

Adnane Saoud, Elena Ivanova, and Antoine Girard. Efficient synthesis for monotone transition systems and directed safety specifications. In *IEEE 58th Conference on Decision and Control*, pages 6255–6260, 2019.

Vladimir Sinyakov and Antoine Girard. Formal controller synthesis from specifications given by discrete-time hybrid automata. *HAL preprint hal-02361404*, 2019.

Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

Daniele Zonetti, Adnane Saoud, Antoine Girard, and Laurent Fribourg. A symbolic approach to voltage stability and power sharing in time-varying dc microgrids. In *European Control Conference*, pages 903–909. IEEE, 2019.