Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Guaranteed memory reduction in synthesis of correct-by-design invariance controllers ⋆

Elisei Macoveiciuc * **Gunther Reissig** *

* *Bundeswehr University Munich, Dept. Aerospace Eng., Inst. of Control Eng., D-85577 Neubiberg (Munich), Germany, elisei.macoveiciuc@unibw.de, http://www.reiszig.de/gunther/*

Abstract: Formal methods for analysis of dynamical systems through construction of finite symbolic abstractions have attracted significant interest as they allow solving complex control problems in a fully automated fashion. Nevertheless, their practical application is currently limited by the fact that they require enormous memory resources. We present a novel algorithm for solution of invariance problems within abstraction-based framework, which guarantees large storage reduction and fully applies to general non-linear plants. We also show that, in practice, the algorithm is faster compared to other methods.

Keywords: correct-by-design, symbolic synthesis, nonlinear control systems, state-space methods, formal methods.

1. INTRODUCTION

Rapid development and deployment of increasingly sophisticated electromechanical systems have stimulated the demand to solve complicated control problems on constrained domains with highly non-linear, partially unknown dynamics, possibly in the presence of obstacles with non-trivial geometry. In addition, modern critical infrastructure requires strong guarantees on the controller behavior.

This has motivated research of formal methods of continuous system analysis through construction of finite symbolic models - abstractions (Reissig et al. (2017); Reissig and Rungger (2019)). Discrete abstractions can then be algorithmically processed to solve the given control task. Moreover, if abstractions preserve certain relation to the original system, the obtained controllers are *correct-by-design*, i.e., solution of the continuous control problem is guaranteed (Reissig et al. (2017)).

Despite the powerful theoretic capabilities, practical application of vast majority of abstraction- based algorithms has been limited due to the fact that construction of symbolic models involves discretization of continuous state space and thus suffers from large time and memory complexity. This attracted extensive effort to increase applicability of the method.

Methods applying to special classes of continuous systems have been shown to cope with problems in higher dimensions: see Zamani et al. (2015); Girard and Gössler (2019) for abstractions without discretization of the state space for stochastic and incrementally stable switched systems respectively, Reißig (2010); Kim et al. (2017, 2018b); Pola et al. (2014); Nilsson and Ozay (2016); Mallik et al. (2019); Hussien et al. (2017); Lavaei et al. (2017); Boskos and Dimarogonas (2015); Dallal and Tabuada (2015) for abstraction of suitably decomposable continuous dynamics, Yordanov et al. (2013), Mouelhi et al. (2013) for algorithmic abstraction refinement for piece-wise affine and incrementally stable switched systems respectively.

A notable abstraction and synthesis decomposition algorithm for general continuous systems is presented by Meyer et al. (2018). The method is able to drastically reduce time and memory consumption at the cost of more conservative obtained controllers. Kim and Arcak (2019) also provide a general framework for modular construction of abstractions from components with similar drawback of additional non-determinism. Another system-independent method by Weber et al. (2017) optimizes state-space discretization parameters. This method applies to every abstract specification and can be combined with our proposed algorithm.

One more promising approach of reducing computational effort that allows weaker assumptions on continuous dynamics is to merge abstraction construction and controller synthesis into one step and to attempt to process only those parts of the abstraction that are needed for solution of the control task, i.e., abstractions are computed *on-the-fly*. Rungger et al. (2013); Saoud et al. (2019) abstract only control task-relevant parts of discrete-time linear and monotone systems respectively. For reach-avoid problems, methods by Rungger and Stursberg (2012); Hsu et al. (2018a); Macoveiciuc and Reissig (2019) are applicable to general classes of systems, with the latter work guaranteeing storage reduction. Algorithms independent of system dynamics for invariance problems also exist. De Alfaro and Roy (2007) provide a method to refine initial coarse abstraction until safety or reachability property over initial states is proven. Li and Liu (2018) refine grids over the state space during invariance synthesis using interval analysis. Hsu et al. (2018b) pre-compute a number of

⋆ This work has been supported by the German Research Foundation (DFG) under grant no. RE 1249/4-1, as well as by AdaCore (www.adacore.com). The first author is a Munich Aerospace scholarship recipient

coarser abstractions and attempt to minimize the synthesis effort spent at (partially computed) finer layers. While demonstrating significant improvement on several examples, these methods may perform worse memory-wise as a result of stored redundant abstractions, in cases where successful synthesis heavily depends on computations at finest layers, e.g. when system dynamics is heavily disturbed or obstacle environment is complicated. Hussien and Tabuada (2018) construct abstract transitions during synthesis until a (safety or reachability) control problem is solved. The mentioned approaches are heuristic in nature and no guarantees on computational reduction can be provided.

To summarize, guaranteed computational relief for invariance problems without controller conservatism has been obtained only for special classes of systems. In addition, while significant speed-up via parallelization is achievable for any system (Kim et al. (2018a)), state-of-the-art lacks general methods for *guaranteed* storage reduction. Thus memory remains major bottleneck of the abstraction-based approach.

We present a novel on-the-fly synthesis algorithm for solution of invariance problems that does not require storage of any part of symbolic model. The obtained controllers are provably at least as permissible as the ones produced by standard methods. Up to our knowledge, this is the first work for invariance problems to require no special properties of continuous dynamics and to guarantee large storage reduction without any additional controller conservatism. We demonstrate on an example that, although we focused on memory, the method outperforms other algorithms in time as well.

2. PRELIMINARIES

The relative complement of the set A in the set B is denoted by $B \setminus A$. $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$ and $\mathbb{Z}_+$ denote the sets of real numbers, non-negative real numbers, integers and non-negative integers, respectively, and $\mathbb{N} = \mathbb{Z}_+ \setminus \{0\}$. $\mathrm{card}(A)$ denotes cardinality of the set A. We adopt the convention that $\pm\infty + x = \pm\infty$ for any $x \in \mathbb{R}$. $[a, b]$, $]a, b[$, $[a, b[$, and $]a, b]$ denote closed, open and half-open, respectively, intervals with end points a and b, e.g. $[0, \infty[\; = \mathbb{R}_+$. $[a; b]$, $]a; b[$, $[a; b[$, and $]a; b]$ stand for discrete intervals, e.g. $[a; b] = [a, b] \cap \mathbb{Z}$, $[1; 4[\; = \{1, 2, 3\}$, and $[0; 0[\; = \emptyset$. $\max M$, $\min M$, $\sup M$ and $\inf M$ denote the maximum, the minimum, the supremum and the infimum, respectively, of the nonempty subset $M \subseteq [-\infty, \infty]$.

$f\colon A \rightrightarrows B$ denotes a *set-valued map* from the set A into the set B, whereas $f\colon A \to B$ denotes an ordinary map. The set of maps $A \to B$ is denoted B^A. If f is set-valued, then f is *strict* and *single-valued* if $f(a) \neq \emptyset$ and $f(a)$ is a singleton, respectively, for every a.

We identify set-valued maps $f\colon A \rightrightarrows B$ with binary relations on $A \times B$, i.e., $(a, b) \in f$ iff $b \in f(a)$. Moreover, if f is single-valued, it is identified with an ordinary map $f\colon A \to B$. The restriction of f to a subset $M \subseteq A$ is denoted $f|_M$. The inverse mapping $f^{-1}\colon B \rightrightarrows A$ is defined by $f^{-1}(b) = \{a \in A \,|\, b \in f(a)\}$, and the image of a subset $C \subseteq A$ under f is denoted $f(C)$, $f(C) = \bigcup_{a \in C} f(a)$. The set of minimum points of f in some subset $Q \subseteq X$ is denoted $\mathrm{argmin}\,\{f(x) \,|\, x \in Q\}$.

Let R be an order on a set S. Then R is called a total order if it is *reflexive*, *transitive*, *antisymmetric*, and either $(a, b) \in R$ or $(b, a) \in R$ for any two elements $a \neq b$ in S.

Let A be a finite set endowed with a total order denoted by $\leq$. We slightly abuse notation by letting $\perp(A)$ denote the (uniquely defined) *singleton set containing the minimum element in* A, i.e., $\perp(A) = \{a' \in A \,|\, \forall_{a \in A} a' \leq a\}$ if $A \neq \emptyset$, and let $\perp(\emptyset) = \emptyset$.

$[\![a, b]\!] \subseteq \mathbb{R}^n$ denotes a closed *hyper-interval*, i.e:

$$[\![a, b]\!] = \mathbb{R}^n \cap ([a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]), \qquad (1)$$

where $a, b \in (\mathbb{R} \cup \{\pm\infty\})^n, a < b, a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n)$.

A family of sets $(X_i)_{i \in [1;n]}, \forall_{i \in [1;n]} X_i \subseteq Y$ forms a cover of Y if $\bigcup_{i=1}^{n} X_i = Y$.

3. PROBLEM STATEMENT

3.1 Control systems

We consider continuous-state discrete-time systems of the form

$$x(t+1) \in F(x(t), u(t)), \qquad (2)$$

where $x(t) \in X$ and $u(t) \in U$ represents the *state* and the *input signal*, respectively. The set-valued *transition function* $F\colon X \times U \rightrightarrows X$ represents dynamics of the control system. In this paper we assume that F is defined implicitly through a solution to an initial value problem for differential inclusions. We start by providing definitions of controller and of closed-loop behavior associated with it, which we adapt from (Reissig and Rungger (2019)) for the case of invariance problems.

Definition 1. A system is a triple

$$(X, U, F) \qquad (3)$$

where X and U are nonempty state and input alphabets and $F\colon X \times U \rightrightarrows X$ is a strict transition map. A pair $(u, x) \in U^{\mathbb{Z}_+} \times X^{\mathbb{Z}_+}$ is a solution of the system (3) if (2) holds for all $t \in \mathbb{Z}_+$.

A controller C for the system (3) (denoted by $C \in \mathcal{F}(X, U)$) is a quintuple

$$(Z, Z_0, \widetilde{X}, \widetilde{U}, H) \qquad (4)$$

where Z, Z_0, $\widetilde{X}$, $\widetilde{U}$ are non-empty state, input and output controller alphabets, $Z_0 \subseteq Z$, $X \subseteq \widetilde{X}$, $\widetilde{U} \subseteq U$, and $H\colon Z \times \widetilde{X} \rightrightarrows Z \times \widetilde{U}$ is strict. A triple $(u, z, x) \in \widetilde{U}^{\mathbb{Z}_+} \times Z^{\mathbb{Z}_+} \times \widetilde{X}^{\mathbb{Z}_+}$ is a solution of the controller (4) if $z(0) \in Z_0$ and

$$(z(t+1), u(t)) \in H(z(t), x(t)), \qquad (5)$$

holds for all $t \in \mathbb{Z}_+$.

Definition 2. Let S denote the system (3) and suppose that $C \in \mathcal{F}(X, U)$, where C is of the form (4).

The behavior $\mathcal{B}(C \times S) \subseteq (U \times X)^{\mathbb{Z}_+}$ of the closed loop composed of C and S is defined by the requirement that $(u, x) \in \mathcal{B}(C \times S)$ iff there exists a signal $z\colon \mathbb{Z}_+ \to Z$ such that (u, z, x) is a solution of C and (u, x) is a solution of S. In addition, the behavior initialized at $p \in X$ is defined as $\mathcal{B}_p(C \times S) = \{(u, x) \in \mathcal{B}(C \times S) \,|\, x(0) = p\}$.

3.2 Invariance problems

Definition 3. Let S be a system of the form (3). An invariance control problem is a tuple

$$(X, U, F, g) \tag{6}$$

where $g\colon X \times X \times U \to \mathbb{R}_+ \cup \{\infty\}$. The problem (6) is called qualitative if g maps into the set $\{0, \infty\}$, and otherwise (6) is quantitative.

To solve an invariance problem of the form (6) means to find controllers that minimize, in a worst-case sense, the total cost $J\colon (U \times X)^{\mathbb{Z}_+} \to [0, \infty]$ defined by

$$J(u, x) = \sum_{t=0}^{\infty} g(x(t), x(t+1), u(t)), \tag{7a}$$

where $(u, x) \in \mathcal{B}_p(C \times S)$, $C \in \mathcal{F}(X, U)$. More formally, the *closed-loop performance* $L\colon X \to [0, \infty]$ of (6) associated with the controller $C \in \mathcal{F}(X, U)$ is given by

$$L(p) = \sup_{(u,x) \in \mathcal{B}_p(C \times S)} J(u, x), \tag{8}$$

and the *achievable performance* $V\colon X \to [0, \infty]$ *associated with (6)* is defined by

$$V(p) = \inf_{C \in \mathcal{F}(X,U)} \sup_{(u,x) \in \mathcal{B}_p(C \times S)} J(u, x). \tag{9}$$

If $L = V$, then the controller $C \in \mathcal{F}(X, U)$ is said to *solve* the invariance problem (6).

Let (X, U, F) be a system of the form (3). Suppose $D \subseteq X$, $M \subseteq X$ is a target and an obstacle set, respectively. Let and

$$g(p, q, u) = \begin{cases} \infty, & \text{if } p \in M \cup (X \setminus D) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Then, the qualitative invariance problem (6) corresponds to the requirement that the system trajectories should remain in D while avoiding obstacles for infinitely large period of time. Note, in this case, $V^{-1}(0)$ represents maximal controlled invariant subset of D.

3.3 Abstractions

We start this section with definition (Reissig and Rungger (2019)) of relation which is used to link abstraction to the original (continuous) system.

Definition 4. Let $\Pi_0 = (X_0, U_0, F_0, g_0)$ and Π be of the form (6). The relation $R\colon X_0 \rightrightarrows X$ is a valuated feedback refinement relation from Π_0 to Π, denoted $\Pi_0 \preccurlyeq_R \Pi$, if R is strict and the following conditions hold for all $(p_0, p), (q_0, q) \in R$ and all $u \in U$:

(1) $U \subseteq U_0$;
(2) $g_0(p_0, q_0, u) \leq g(p, q, u)$;
(3) $R(F_0(p_0, u)) \subseteq F(p, u)$.

We now briefly describe a method to construct abstractions according to Reissig et al. (2017).

Let $\Pi_0 = (X_0, U_0, F_0, g_0)$ be a control problem with (X_0, U_0, F_0) continuous-state sampled system of the form (2), where $X_0 = [\![a, b]\!] \subseteq \mathbb{R}^n$, $U_0 = [\![u, v]\!] \subseteq \mathbb{R}^m$.

Let X be a cover of X_0, $U \subset U_0, \mathrm{card}(U) < \infty$. X can be specified through discretization parameters $(d_1, d_2, ..., d_n)$, where d_i is the number of grid points in state space dimension $i, i = 1, n$. Grid points are then taken as centers of hyper-intervals (1) that form X.

Let $x \in X, x = [\![a, b]\!]$ and $F_0([\![a, b]\!], u) \subseteq \bar{F}_0([\![a, b]\!], u)$. Define

$$F(x, u) = \{y \in X \,|\, y = [\![c, d]\!] \wedge [\![c, d]\!] \cap \bar{F}_0([\![a, b]\!], u) \neq \emptyset\}$$

The usage of over-approximation $\bar{F}_0([\![a, b]\!], u)$ of the set $F_0([\![a, b]\!], u)$ in the above definition is motivated by the fact that in contrast to over-approximations, exact reachable sets are not possible to compute for general non-linear systems. Note, F also possesses implicit representation via functionality to compute intersecting sets with reachable set over-approximations. This fact is exploited in construction of on-the-fly synthesis algorithms.

Then $\Pi = (X, U, F, g)$ is an abstraction preserving valued feedback refinement relation to Π_0, provided that function $g\colon X \times X \times U \to \mathbb{R}_+ \cup \{\infty\}$ is defined to meet requirement (2) of definition 4.

4. ON-THE-FLY SYNTHESIS METHOD WITH MEMORY REDUCTION GUARANTEES

Algorithm 1

Input: Control problem $\Pi = (X, U, F, g)$, W_0,c_0
Input: Operator P
Input: Function **ProcessTransitions**
Require: X, U finite
1: $W := P(W_0)$ // W: value function
2: $X \supseteq Q :\supseteq \{x \in X | W(x) \neq W_0(x)\}$ // Q: queue
3: $E := \emptyset$ // E: set of settled states
4: $c := c_0$ // c: controller
5: **while** $Q \neq \emptyset$ **do**
6: $\quad \emptyset \neq Y :\subseteq \mathrm{argmin}\,\{W(x) \,|\, x \in Q\}$
7: $\quad Q := Q \setminus Y$
8: $\quad E := E \cup Y$
9: $\quad (Q, c, W) :=$ **ProcessTransitions**(Π, E, Y, Q, c, W)
Output: c, W

We use Algorithm 1 to solve invariance problems on abstractions of continuous-state systems. $c\colon X \rightrightarrows U$ and $W\colon X \to \mathbb{R}_+ \cup \{\infty\}$. Commands of the form $X \supseteq Q :\supseteq M$ on line 2 require that a set Q satisfying $M \subseteq Q \subseteq X$ is chosen, and similarly for the command on line 6. The Algorithm 9 iteratively expands E, and finds all (x, u) such that $F(x, u)$ and E satisfy certain relation. Thus, standard implementation of line 9 requires the inverse of map F to be available. While F is assumed to be given implicitly, F^{-1}, in general, can only be accessed from F stored appropriately in memory. Representation of full F in memory before synthesis is necessary for all standard abstraction-based algorithms, and suffers from extreme memory costs.

We further provide an on-the-fly variant of algorithm 1 by constructing suitable *ProcessTransition* function. In contrast to existing approaches, the proposed method does not require storage of any part of F in memory. Note that algorithm 1 has the same structure as Dijkstra algorithm for reach-avoid problems (Reissig and Rungger (2019)). However, functionality of line 9 algorithmically differs for invariance case. We start by providing conditions under which algorithm 1 solves a finite invariance problem.

***Loop Invariant** 1.* Let Π be of the form (6) and V be the achievable performance for Π.

$$E \cap Q = \emptyset \wedge W^{-1}(]0, \infty]) \subseteq E \cup Q \subseteq X \tag{11a}$$
$$V \geq W \geq P_Q(W) \tag{11b}$$

where

$$P_Q(W)(p) = \min\{\infty, W_Q(p)\} \tag{12}$$
$$W_Q(p) = \inf_{u \in U} \sup_{q \in F(p,u) \setminus Q} g(p, q, u) + W(q), \tag{13}$$

$\sup \emptyset = \infty$. Let L_C be the closed-loop performance of:

$$C = (Z, Z, X, U, H) \tag{14}$$

where Z is any singleton set and $H \colon Z \times X \rightrightarrows Z \times U$ is any map satisfying

$$\emptyset \neq H(Z, p) \subseteq \begin{cases} Z \times U, & \text{if } c(p) = \emptyset \\ Z \times c(p), & \text{otherwise} \end{cases} \tag{15}$$

for all $p \in X$ and c - output of algorithm 1.

Proposition 5. Let $\Pi = (X, U, F, g)$ be a control problem with finite X and U and g satisfying (10). Assume P defined as follows:

$$P(W)(p) = \min\left\{\infty, \inf_{u \in U} \sup_{q \in F(p,u)} g(p, q, u) + W(q)\right\} \tag{16}$$

and let $\forall_{x \in X} c_0(x) = U$ Then the following statements hold for Algorithm 1 ($\Pi, 0, c_0, P$):

(1) Loop Invariant 1 holds for Π upon execution of lines 3-4.

(2) Assume that each call to the function *ProcessTransitions* in Algorithm 1 terminates. Assume (11a) holds upon every execution of line 9 then $Y \neq \emptyset$, $E \cap Y = \emptyset$ after every execution of line 6. Moreover, set E is strictly enlarged after every execution of line 5 and Algorithm 1 terminates returning c and W.
Assume exists sequence $(\Pi_i)_{i \in [1;N]}$ $\Pi_i = (X, U, F_i, g)$, F_i - strict, $\forall_{i,j \in [1;N], i<j} F_i \subseteq F_j \subseteq F$, such that (11b) holds for Π_i upon execution of line 9 on iteration i of the while- loop. Then $\tilde{V} = W$ upon termination, where $\tilde{V}$ is the achievable performance (9) associated with $\tilde{\Pi} = \Pi_N$.

(3) Relation

$$\forall_{x \in W^{-1}(0)} F(x, c(x)) \subseteq (W^{-1}(0) \cup Q) \wedge c(x) \neq \emptyset \tag{17}$$

holds upon execution of lines 3-4. Assume, in addition that (17) holds for F_i upon execution of line 9 on iteration i of the while- loop. Then C defined by (14) and (15) is a static controller for $(X, U, \tilde{F})$ solving $\tilde{\Pi}$.

4.1 On-the-fly method

Before turning to actual implementation of the function to process transitions we introduce auxiliary definitions.

Definition 6. Let $\Pi_0 = (X_0, U_0, F_0, g_0), \Pi$ be of the form (6) such that $\Pi_0 \preccurlyeq_R \Pi$. A map $F^- \colon X \times U \rightrightarrows X$ is called *backward transition map* associated with Π if the following holds :

$$\forall_{(x,u,y) \in X \times U \times X} \forall_{x \notin F^-(y,u)} R^{-1}(y) \cap F_0(R^{-1}(x), u) = \emptyset \tag{18}$$

In practice, F^- is constructed (analogously to F described in Section 3.3) finding intersecting abstract states with over-approximations of reachable sets of the continuous system, but at negative sampling time. Thus, for any $(x, u) \in X \times U$ a set of functions is available to compute $F^-(x, u)$ on demand. Also note that due to the use of over-approximations of sets in construction of F and F^- we have $F^- \neq F^{-1}$. Adopting the use of F^- in our synthesis algorithm allows us to avoid storage of any part of F while keeping the time consumption reasonable.

Let $\Pi = (X, U, F, g), \Pi_0 \preccurlyeq_R \Pi$. We now define an auxiliary control problem associated with Π

$$\tilde{\Pi} = (X, U, \tilde{F}, g) \tag{19}$$

with $\forall_{(x,u) \in X \times U} \tilde{F}(x, u) \subseteq F(x, u) \wedge \forall_{y \in F(x,u) \setminus \tilde{F}(x,u)} x \notin F^-(y, u)$ and F^- satisfying (18).

The following is evident.

Lemma 7. Let Π_0, Π be such that $\Pi_0 \preccurlyeq_R \Pi$ and $\tilde{\Pi}$ be as in (19). Then $\Pi_0 \preccurlyeq_R \tilde{\Pi}$

We continue with implementation of the function **ProcessTransitions**.

Function 2 ProcessTransitions

Input: $\Pi = (X, U, F, g)$, E, Y, Q, c, W
Require: F^- satisfying (18)
1: **for all** $y \in Y$ **do**
2:     **for all** $(x, u) : x \in F^-(y, u) \wedge x \notin E \cup Q \wedge (c(x) = U \vee c(x) = \{u\})$ **do**
3:         **if** $c(x) = \{u\} \wedge y \in F(x, u)$ **then**
4:             $c(x) := \bot\{v \in U | u < v \wedge F(x, v) \cap E = \emptyset\}$
5:         **else if** $c(x) = U \wedge \operatorname{card} U > 1$ **then**
6:             $c(x) := \bot\{v \in U | F(x, v) \cap E = \emptyset\}$
7:         **if** $c(x) = \emptyset$ **then**
8:             $Q := Q \cup \{x\}$
9:             $W(x) = \infty$
Output: Q,c,W

Theorem 8. Assume hypothesis of Proposition 5. Let $\Pi = (X, U, F, g)$ be the input to the algorithm 1 and function *ProcessTransitions* be implemented as function 2. Assume, in addition, Π_0 given such that $\Pi_0 \preccurlyeq_R \Pi$.

Then each call to func. 2 terminates and there exists sequence $(\Pi_i)_{i \in [1;N]}$ $\Pi_i = (X, U, F_i, g)$, of the form (19),$\Pi_N = \tilde{\Pi}$ F_i - strict, $\forall_{i,j \in [1;N], i<j} F_i \subseteq F_j \subseteq F$, such that Loop invariant 1 and (17) hold for Π_i upon execution of line 9 on iteration i of the while- loop.

Remark 1. Note that computations of function 2 are structured according to order placed on finite set U (lines 4, 6). Existence of order relations on *continuous* state and input sets is also exploited by Saoud et al. (2019). However their work is applicable only to monotone systems. In contrast we require only abstract input set to be ordered and a suitable (total) order relation can always be defined since $\operatorname{card} U < \infty$. This places no assumption on continuous dynamics.

We can now provide correctness result of the proposed method defined by algorithm 1 together with function 2.

Theorem 9. Assume hypotheses of Proposition 5 and Theorem 8. Let $\Pi = (X, U, F, g)$ be the input to the algorithm 1 and Π_0 be such that $\Pi_0 \preccurlyeq_R \Pi$.

Then there exists $\tilde{\Pi}, \Pi_0 \preccurlyeq_R \tilde{\Pi}$ such that Algorithm 1 solves $\tilde{\Pi}$.

Remark 2. The fact that Π_0 and $\tilde{\Pi}$ are related through R guarantees that the obtained controllers are correct-by-design (Reissig et al. (2017)).

4.2 Time and memory consumption

Let $n = \text{card}(X), m = \sum_{p\in X}\sum_{u\in U}\text{card}(F(p,u)), m^- = \sum_{p\in X}\sum_{u\in U}\text{card}(F^-(p,u))$. The storage cost of full abstraction is evidently $O(m)$.

Theorem 10. Assume hypotheses of Proposition 5 and Theorem 8. Let $\Pi = (X, U, F, g)$ be the input to the algorithm 1. Additionally assume that

$$\forall_{x\in X}\exists_{[\![a,b]\!]\subseteq\mathbb{R}^n} x = [\![a,b]\!] \tag{20}$$

$$\forall_{(x,u)\in X\times U}\exists_{[\![a,b]\!]\subseteq\mathbb{R}^n} F(x,u) = \{x \in X \mid x \cap [\![a,b]\!] \neq \emptyset\} \tag{21}$$

Then there exists implementation of function 2 such that Algorithm 1 together with function 2

(1) requires $O(n)$ memory
(2) requires $O(m + m^-)$ time.

4.3 Controller conservatism

We end this section by discussing solution of the original (continuous) problem Π_0 and conservatism of obtained controllers.

Theorem 11. Assume hypothesis of theorem 9. Let $\Pi = (X, U, F, g)$ be the input to the algorithm 1, $\Pi_0, \tilde{\Pi}$ be as in theorem 8 and $W = \tilde{V}$ be the value function in the output of algorithm 1, where $\tilde{V}$ is the achievable performance for $\tilde{\Pi}$. Let V be the achievable performance for Π. Then

$$V_0(x_0) \leq \tilde{V}(x) \leq V(x) \text{ for every } (x_0, x) \in R$$

Remark 3. Most symbolic synthesis algorithms available in the literature solve Π which is an abstraction of a (continuous- state) system constructed using reachable sets at positive sampling time only. The above result implies that the maximal controlled invariant set obtained by solving $\tilde{\Pi}$ is at least as large as the invariant set obtained by solving Π.

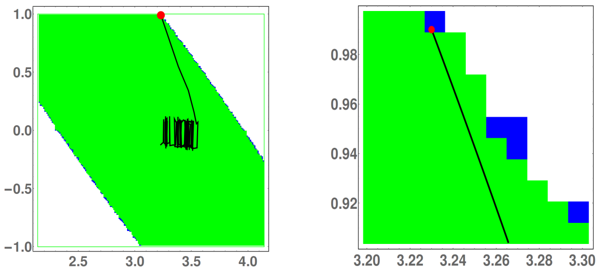
5. NUMERICAL EXAMPLE

We compare the proposed on-the-fly algorithm 1 against its standard variant, SCOTS (Rungger and Zamani (2016)) implementation of the fixed point iteration, and on-the-fly algorithm by Hsu et al. (2018b) implemented in the MASCOT tool. Standard variant of algorithm 1 and SCOTS pre-compute full abstraction before synthesis.

Our current implementation of both standard and on-the-fly versions of algorithm 1 stores abstraction and controllers in sparse matrices, SCOTS is able to use sparse matrices or Binary Decision Diagrams (BDDs) for abstraction storage and MASCOT uses BDDs only.

5.1 Control problem

Consider disturbed pendulum model (Reißig (2010))



(a) (b) - scaled (a)

Fig. 1. (a) - maximal controlled invariant sets by on-the-fly (green and blue) and standard (green only) variants of algorithm 1, simulated closed loop trajectory (black) from initial state (red) controlled only by on-the-fly algorithm.

$$\dot{x} \in \begin{pmatrix} x_2 \\ -\sin(x_1) - \cos(x_1)u + w \end{pmatrix} \tag{22}$$

where disturbance signal $w(t) \in [-0.5, 0.5]$ includes unknown friction dynamics. We find a controller that restricts system trajectories to an invariant subset of $[\pi - 1, \pi + 1] \times [-1, 1]$ with $u \in [-1.9, 1.9]$. To construct abstraction, discretization parameters (sec. 3.3) for state and input sets are $(256, 256)$ and (19) respectively. The sampling time is 0.2 seconds. MASCOT uses 3 abstraction layers with sampling times $0.8, 0.4, 0.2$. (see Hsu et al. (2018b)).

5.2 Discussion of results

Time and memory comparisons are presented in Table 1. On this example full abstraction in the BDD form is more than 3 times smaller compared to sparse matrix (lines 3-5 of Table 1). This comes at the cost of time consumption, increased by at least 6 times. Note, however, that the BDD memory savings cannot be guaranteed.

Table 1.

Algorithm or Tool	Memory (MB)	Time (s)
On-the-fly algorithm 1	20	4
Standard algorithm 1	500	14
SCOTS (BDDs)	140	76
SCOTS (sparse matrices)	460	5
MASCOT	293	32

The on-the-fly algorithm 1 required at least 7 times less memory compared to all other methods and was also faster. Figure 1 depicts maximal invariant sets obtained by standard and on-the-fly variants of algorithm 1. In accordance with remark 3 the maximal invariant set produced by the on-the-fly algorithm is larger than the one produced by the standard method.

REFERENCES

Boskos, D. and Dimarogonas, D.V. (2015). Decentralized abstractions for feedback interconnected multi-agent systems. In *Proc. IEEE Conf. Decision and Control (CDC), Osaka, Japan, 15-18 December 2015*, 282–287.

Dallal, E. and Tabuada, P. (2015). On compositional symbolic controller synthesis inspired by small-gain theorems. In *Proc. IEEE Conf. Decision and Control (CDC), Osaka, Japan, 15-18 December 2015*, 6133–6138.

De Alfaro, L. and Roy, P. (2007). Solving games via three-valued abstraction refinement. In *International Conference on Concurrency Theory*, 74–89. Springer.

Girard, A. and Gössler, G. (2019). Safety synthesis for incrementally stable switched systems using discretization-free multi-resolution abstractions. *Acta Inform.*

Hsu, K., Majumdar, R., Mallik, K., and Schmuck, A.K. (2018a). Lazy abstraction-based control for reachability. `https://arxiv.org/abs/1804.02722v1`.

Hsu, K., Majumdar, R., Mallik, K., and Schmuck, A.K. (2018b). Lazy abstraction-based control for safety specifications. In *Proc. 57th IEEE Conf. Decision and Control (CDC), Miami, FL, USA, 17-19 December 2018*, 4902–4907.

Hussien, O., Ames, A., and Tabuada, P. (2017). Abstracting partially feedback linearizable systems compositionally. *IEEE Control Systems Letters*, 1(2), 227–232.

Hussien, O. and Tabuada, P. (2018). Lazy controller synthesis using three-valued abstractions for safety and reachability specifications. In *Proc. 57th IEEE Conf. Decision and Control (CDC), Miami, FL, USA, 17-19 December 2018*, 3567–3572.

Kim, E., Arcak, M., Khaled, M., and Zamani, M. (2018a). Major computational breakthroughs in the synthesis of symbolic controllers via decomposed algorithms. In *Proc. 21st Intl. Conf. Hybrid Systems: Computation and Control (HSCC), Porto, Portugal, April 11-13, 2018*, 285–286. ACM.

Kim, E.S. and Arcak, M. (2019). Abstractions for symbolic controller synthesis are composable. *arXiv preprint arXiv:1807.09973*.

Kim, E.S., Arcak, M., and Seshia, S.A. (2017). Symbolic control design for monotone systems with directed specifications. *Automatica J. IFAC*, 83, 10–19.

Kim, E.S., Arcak, M., and Zamani, M. (2018b). Constructing control system abstractions from modular components. In *Proc. 21st Intl. Conf. Hybrid Systems: Computation and Control (HSCC), Porto, Portugal, April 11-13, 2018*, 137–146. ACM.

Lavaei, A., Soudjani, S.E.Z., Majumdar, R., and Zamani, M. (2017). Compositional abstractions of interconnected discrete-time stochastic control systems. In *Proc. 56th IEEE Conf. Decision and Control (CDC), Melbourne, Australia, 12-15 December 2017*, 3551–3556.

Li, Y. and Liu, J. (2018). Invariance control synthesis for switched nonlinear systems: an interval analysis approach. *IEEE Trans. Automat. Control*, 63(7), 2206–2211.

Macoveiciuc, E. and Reissig, G. (2019). Memory efficient symbolic solution of quantitative reach-avoid problems. In *Proc. American Control Conference (ACC), Philadelphia, U.S.A., 10-12 July 2019*, 1671–1677. doi:10.23919/ACC.2019.8814850.

Mallik, K., Schmuck, A., and Majumdar, R. (2019). Compositional synthesis of finite state abstractions. *IEEE Transactions on Automatic Control*, 64(6), 2629–2636.

Meyer, P.J., Girard, A., and Witrant, E. (2018). Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Trans. Automat. Control*, 63(6), 1835–1841.

Mouelhi, S., Girard, A., and Gössler, G. (2013). CoSyMA: A tool for controller synthesis using multi-scale abstractions. In *Proc. 16th Intl. Conf. Hybrid Systems: Computation and Control (HSCC), Philadelphia, PA, U.S.A., April 8-11, 2013*, 83–88. ACM, New York, NY, USA.

Nilsson, P. and Ozay, N. (2016). Control synthesis for large collections of systems with mode-counting constraints. In *Proc. 19th Intl. Conf. Hybrid Systems: Computation and Control (HSCC), Vienna, Austria, April 12-14, 2016*, 205–214.

Pola, G., Pepe, P., and Di Benedetto, M.D. (2014). Symbolic models for networks of discrete-time nonlinear control systems. In *American Control Conference (ACC), 2014*, 1787–1792. IEEE.

Reißig, G. (2010). Abstraction based solution of complex attainability problems for decomposable continuous plants. In *Proc. 49th IEEE Conf. Decision and Control (CDC), Atlanta, GA, U.S.A., 15-17 December 2010*, 5911–5917. doi:10.1109/CDC.2010.5718125.

Reissig, G. and Rungger, M. (2019). Symbolic optimal control. *IEEE Trans. Automat. Control*, 64(6), 2224–2239. doi:10.1109/TAC.2018.2863178.

Reissig, G., Weber, A., and Rungger, M. (2017). Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Trans. Automat. Control*, 62(4), 1781–1796. doi:10.1109/TAC.2016.2593947. .

Rungger, M., Mazo, M., and Tabuada, P. (2013). Specification-guided controller synthesis for linear systems and safe linear-time temporal logic. In *Proc. 16th Intl. Conf. Hybrid Systems: Computation and Control (HSCC), Philadelphia, PA, U.S.A., April 8-11, 2013*, 333–342. ACM.

Rungger, M. and Stursberg, O. (2012). On-the-fly model abstraction for controller synthesis. In *Proc. American Control Conference (ACC), Montréal, Canada, 27-29 June 2012*, 2645–2650.

Rungger, M. and Zamani, M. (2016). SCOTS: A tool for the synthesis of symbolic controllers. In *Proc. 19th Intl. Conf. Hybrid Systems: Computation and Control (HSCC), Vienna, Austria, April 12-14, 2016*, 99–104.

Saoud, A., Ivanova, E., and Girard, A. (2019). Efficient synthesis for monotone transition systems and directed safety specifications. In *Proc. 58th IEEE Conf. Decision and Control (CDC), Nice, France, 11-13 December 2019*, 6255–6260.

Weber, A. and Reißig, G. (2013). Local characterization of strongly convex sets. *J. Math. Anal. Appl.*, 400(2), 743–750. doi:10.1016/j.jmaa.2012.10.071.

Weber, A. and Reissig, G. (2014). Classical and strong convexity of sublevel sets and application to attainable sets of nonlinear systems. *SIAM J. Control Optim.*, 52(5), 2857–2876. doi:10.1137/130945983.

Weber, A., Rungger, M., and Reissig, G. (2017). Optimized state space grids for abstractions. *IEEE Trans. Automat. Control*, 62(11), 5816–5821. doi:10.1109/TAC.2016.2642794.

Yordanov, B., Tůmová, J., Černá, I., Barnat, J., and Belta, C. (2013). Formal analysis of piecewise affine systems through formula-guided refinement. *Automatica J. IFAC*, 49(1), 261–266.

Zamani, M., Abate, A., and Girard, A. (2015). Symbolic models for stochastic switched systems: a discretization and a discretization-free approach. *Automatica J. IFAC*, 55, 183–196.