Automatica 83 (2017) 10–19

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Symbolic control design for monotone systems with directed specifications☆

Eric S. Kim, Murat Arcak, Sanjit A. Seshia

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA

ARTICLE INFO

Article history:
Received 20 June 2016
Received in revised form 14 November 2016
Accepted 4 April 2017

Keywords:
Directed specifications
Abstraction
Monotone systems
Linear temporal logic
Controller synthesis

ABSTRACT

We study the control of monotone systems when the objective is to maintain trajectories in a directed set (that is, either upper or lower set) within a signal space. We define the notion of a directed alternating simulation relation and show how it can be used to tackle common bottlenecks in abstraction-based controller synthesis. First, we develop sparse abstractions to speed up the controller synthesis procedure by reducing the number of transitions. Next, we enable a compositional synthesis approach by employing directed assume–guarantee contracts between systems. In a vehicle traffic network example, we synthesize an intersection signal controller while dramatically reducing runtime and memory requirements compared to previous approaches.

1. Introduction

A variety of tools for symbolic controller synthesis have been developed over the last decade to enforce complex specifications such as those encoded in temporal logic. This paper's goal is to reduce the computational burden incurred with a growing system size by exploiting system structure and specifications. In particular we focus on monotone systems as investigated by Angeli and Sontag (2003) and Hirsch (1985) which preserve a partial order of states, and directed specifications which encourage either high or lower valued signal trajectories.

A common paradigm for symbolic controller synthesis entails first abstracting a dynamical system with a continuous state space to a discrete system with a finite state space. A synthesis engine takes the abstraction and solves a game where the system's controller seeks to enforce a specification and an adversarial environment seeks to induce a specification violation. The choice of game solver depends on the type of specification to be enforced. A winning control strategy for the abstract system (if it exists) is subsequently refined into a strategy for the continuous system; the notion of a system relation formalizes conditions for refinement from an abstract controller to a concrete one, as summarized by Tabuada (2009). As the system size increases, the abstraction procedure results in three symptoms that increase the synthesis engine's runtime and memory requirements:

(1) Reachability calculations within the abstraction procedure become more expensive or conservative.
(2) The number of discrete transitions grows exponentially.
(3) The number of discrete states grows exponentially.

Addressing the first challenge, Coogan and Arcak (2015); Moor and Raisch (2002) have used variants of monotonicity to efficiently upper and lower bound the reachable sets by simulating the dynamics from two points. However, the latter two computational challenges remain open. To tackle these issues, this paper utilizes directed specifications. We develop the notion of directed alternating simulation relations and present two immediate results. First, we introduce sparse abstractions, which prevent an exponential blowup in the number of transitions in the abstract system, and an associated controller refinement procedure. Second, we limit the exponential growth in discrete states by breaking apart the synthesis into a set of smaller problems. These smaller problems then use assume–guarantee reasoning between sub-systems; soundness of this procedure was demonstrated by Lomuscio, Strulo, Walker, and Wu (2010). We also show that assumptions and guarantees can

☆ This work was supported in part by National Science Foundation grant CNS-1446145, the National Science Foundation Graduate Research Fellowship Program, National Science Foundation Expeditions grant CCF-1139138 and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. The material in this paper was partially presented at the 54th IEEE Conference on Decision and Control, December 15–18, 2015, Osaka, Japan and at the 19th International Conference on Hybrid Systems: Computation and Control (HSCC), April 12–14, 2016, Vienna, Austria. This paper was recommended for publication in revised form by Associate Editor Bert Tanner under the direction of Editor Christos G. Cassandras.

E-mail addresses: eskim@eecs.berkeley.edu (E.S. Kim), arcak@eecs.berkeley.edu (M. Arcak), sseshia@eecs.berkeley.edu (S.A. Seshia).

http://dx.doi.org/10.1016/j.automatica.2017.04.060
0005-1098/

be designed with binary search. The computational speedups from our results are showcased in a vehicular traffic example using the controller synthesis tool in conPAS2.

Existing tools such as CoSyMA by Mouelhi, Girard, and Gössler (2013), SCOTS by Rungger and Zamani (2016), Pessoa by Mazo Jr., Davitian, and Tabuada (2010), and conPAS2 by Yordanov, Tumov, Černy, Barnat, and Belta (2012) each take a dynamical system model and perform the abstraction and refinement steps, but defer the synthesis procedure to outside synthesis engines. The first three tools use a fixed point algorithm over binary decision diagrams as a synthesis engine, while cosPAS2 solves the game over a graph; Baier and Katoen (2008) provide an overview of both these algorithms.

Instances of compositional synthesis exist in the literature. Rungger and Zamani (2015) construct an abstraction for an interconnected system from the abstractions of individual sub-systems via a small-gain theorem. Nilsson and Ozay (2016) and Sadraddini and Belta (2016) construct robust controlled invariant sets which act as assume–guarantee contracts between sub-systems. Meyer, Girard, and Witrant (2015) also synthesize controllers for monotone systems, but do not exploit directedness to reduce the abstraction size. Dallal and Tabuada (2015) enforce a discrete stability condition by constructing abstractions that respect the level curves of a Lyapunov function.

This paper expands upon our previous results in Kim, Arcak, and Seshia (2015, 2016). We developed a compositional synthesis framework for vehicle traffic networks via assume–guarantee contracts in Kim et al. (2015), along with preliminary results on identifying contract parameters when specifications were restricted to a combination of safety and reachability. In Kim et al. (2016) we introduced the notion of directed specifications for an assumption mining problem. Our results about directed alternating simulation relations in Section 4 and sparse abstractions in Section 5 are new. Section 6 on compositional synthesis subsumes the results of Kim et al. (2015) which were derived for a traffic flow model.

2. Preliminaries

2.1. Notation

For a set $\mathcal{P}$, let $|\mathcal{P}|$, $2^{\mathcal{P}}$, $\mathcal{P}^*$, and $\mathcal{P}^\omega$ respectively represent $\mathcal{P}$'s cardinality, powerset (set of all subsets), and sets of finite and infinite sequences of elements of $\mathcal{P}$. We let $\mapsto$ denote a functional map between a domain and a codomain, and $\Longrightarrow$ represent Boolean implication. The *image* $f(\mathcal{L})$ of a set $\mathcal{L} \subseteq \mathcal{P}$ under function $f : \mathcal{P} \mapsto \mathcal{R}$ is the set of points $\{f(x) : x \in \mathcal{L}\}$ and the *preimage* $f^{-1}(\mathcal{N})$ of a set $\mathcal{N} \subseteq \mathcal{R}$ is $\{x \in \mathcal{P} : f(x) \in \mathcal{N}\}$. Boolean true and false are denoted by $\top$ and $\bot$. The Boolean negation of a proposition a is $\neg a$, and we have logical operations $\wedge$ (and/conjunction) and $\vee$ (or/disjunction).

We consider both set and logic based viewpoints for specifications ϕ. When ϕ is a set we say that $x \in \phi$ ("x is a member of the specification set ϕ"). When ϕ is a logical formula, we say that $x \models \phi$ ("x satisfies the formula ϕ"). It is easy to move between these two viewpoints because $x \models \phi$ if and only if $x \in \phi$.

2.2. Transition systems and environments

We consider two types of transition systems. First, an *open system* $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, \delta, \mathcal{Y}, h)$ consists of a state space $\mathcal{X}$, an initial set $\mathcal{X}_0 \subseteq \mathcal{X}$, a finite set of control modes $\mathcal{U}$, a set of uncontrollable environment disturbances $\mathcal{D}$, a non-deterministic transition relation $\delta : \mathcal{X} \times \mathcal{U} \times \mathcal{D} \mapsto 2^{\mathcal{X}}$, an output space $\mathcal{Y}$, and a deterministic output map $h : \mathcal{X} \mapsto \mathcal{Y}$. Such a transition system is said to be *nonblocking* if $\delta(x, u, d)$ is non-empty for all $x \in \mathcal{X}$, $u \in \mathcal{U}$, and $d \in \mathcal{D}$, and *deterministic* if such an $\delta(x, u, d)$ is a singleton.

We consider assumptions on the environment behavior $\phi_a \subseteq \mathcal{D}$ that restrict disturbances to a subset of the disturbance space at all times. For example, $\mathcal{D}$ may be $\mathbb{R}_{\geq 0}$ while $\phi_a = \{x \in \mathbb{R}_{\geq 0} : x \leq 5\}$. In later sections, this assumption set will vary as part of a contract between systems. From an open system and environment assumption, we construct a *nonreactive system* $\Lambda(\Sigma, \phi_a) = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \Delta, \mathcal{Y}, h)$ where $\Delta : \mathcal{X} \times \mathcal{U} \mapsto 2^{\mathcal{X}}$ and $x' \in \Delta(x, u)$ if and only if there exists a $d \in \phi_a$ such that $x' \in \delta(x, u, d)$. In effect, any exogenous disturbance is now implicitly encoded as additional non-determinism in the transition relation Δ and $\delta(x, u, d) \subseteq \Delta(x, u)$ for all $d \in \phi_a$ and all x, u.

A *behavior* of $\Lambda(\Sigma, \phi_a)$ is any sequence $y = y_0 y_1 \ldots \in \mathcal{Y}^\omega$ such that there exist an $x_0 \in \mathcal{X}_0$ and a pair of sequences $u = u_0 u_1 \ldots \in \mathcal{U}^\omega$ and $x = x_0 x_1 \ldots \in \mathcal{X}^\omega$, generating y according to $x_{k+1} \in \Delta(x_k, u_k)$ and $y_k = h(x_k)$. A control policy $\mathbb{C} : \mathcal{X}^* \mapsto \mathcal{U}$ takes a finite state sequence $x = x_0 x_1 \ldots x_k$ and outputs a control mode u_k for all times k, and is paired with a set of initial states $\mathcal{X}_0^{\mathbb{C}} \subseteq \mathcal{X}_0$, forming a tuple $(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C})$. We let $\mathcal{B}_{\Lambda(\Sigma,\phi_a)}(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C}) \subseteq \mathcal{Y}^\omega$ be the set of all possible behaviors when the control strategy is placed in closed loop with $\Lambda(\Sigma, \phi_a)$, i.e., a controlled trajectory $y = y_0 y_1 \ldots \in \mathcal{B}_{\Lambda(\Sigma,\phi_a)}(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C})$ satisfies $x_0 \in \mathcal{X}_0^{\mathbb{C}}$, $x_{k+1} \in \Delta(x_k, u_k)$, $u_k = \mathbb{C}(x_0 x_1 \ldots x_k)$, and $y_k = h(x_k)$. An output specification $\phi_{\text{spec}} \subseteq \mathcal{Y}^\omega$ encodes a set of desirable behaviors.

Problem 1. Given a nonreactive system $\Lambda(\Sigma, \phi_a)$ and a specification $\phi_{\text{spec}} \subseteq \mathcal{Y}^\omega$, find a control strategy $(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C})$ such that $\mathcal{X}_0^{\mathbb{C}}$ is nonempty and all trajectories of the closed loop system satisfy $\mathcal{B}_{\Lambda(\Sigma,\phi_a)}(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C}) \subseteq \phi_{\text{spec}}$.

Solving Problem 1 can be viewed as a sequential game between a controller who wants to satisfy the specification despite all actions from an adversarial environment that picks disturbances. After the controller picks a control mode, the environment may choose amongst the set of possible transitions allowed under transition relation Δ. We do not develop a new controller synthesis engine, but in our examples use an existing solver in cosPAS2 and propose effective pre-processing techniques that enable us to solve larger problems.

3. Partial orders, directed sets, and monotone systems

3.1. Partial orders

A partially ordered set $\mathcal{P}$ has an associated binary relation $\preceq_{\mathcal{P}}$ where for all $p_1, p_2, p_3 \in \mathcal{P}$ the binary relation satisfies (1) $p_1 \preceq_{\mathcal{P}} p_1$, (2) if $p_1 \preceq_{\mathcal{P}} p_2$ and $p_2 \preceq_{\mathcal{P}} p_1$ then $p_1 = p_2$ and, (3) if $p_1 \preceq_{\mathcal{P}} p_2$ and $p_2 \preceq_{\mathcal{P}} p_3$ then $p_1 \preceq_{\mathcal{P}} p_3$. We define $\succeq_{\mathcal{P}}$ so that $p_1 \succeq_{\mathcal{P}} p_2$ holds if and only if $p_2 \preceq_{\mathcal{P}} p_1$. If neither $p_1 \preceq_{\mathcal{P}} p_2$ nor $p_1 \succeq_{\mathcal{P}} p_2$ holds, we say that p_1 and p_2 are *incomparable*. An interval $[a, b] \subseteq \mathcal{P}$ is the set $\{x \in \mathcal{P} : a \preceq_{\mathcal{P}} x \preceq_{\mathcal{P}} b\}$. Given a collection of partially ordered sets $\mathcal{P}_i$ and relations $\preceq_{\mathcal{P}_i}$ with index $i \in \mathcal{A}$, let $\mathcal{P} = \prod_{i \in \mathcal{A}} \mathcal{P}_i$, and $\pi_i(p) : \mathcal{P} \mapsto \mathcal{P}_i$ map $p \in \mathcal{P}$ to its ith component. For $p_1, p_2 \in \mathcal{P}$, the product ordering relation $p_1 \preceq_{\mathcal{P}} p_2$ holds if and only if $\pi_i(p_1) \preceq_{\mathcal{P}_i} \pi_i(p_2)$ for all $i \in \mathcal{A}$. Similarly, a partial ordering $p \preceq_{\mathcal{P}^\omega} q$ between a pair of infinite sequences $p = p_0 p_1 \ldots$ and $q = q_0 q_1 \ldots$ holds if and only if $p_k \preceq_{\mathcal{P}} q_k$ for all k. A function between partially ordered sets $f : \mathcal{P} \mapsto \mathcal{R}$ is a *monotone function* if $p_1 \preceq_{\mathcal{P}} p_2$ implies $f(p_1) \preceq_{\mathcal{R}} f(p_2)$ for all $p_1, p_2 \in \mathcal{P}$. The composition of monotone functions is also a monotone function.

3.2. Directed specifications

Definition 2. A subset $\mathcal{L}$ of the partially ordered set $\mathcal{P}$ is a lower set if for all pairs $x, y \in \mathcal{P}$

$$(y \in \mathcal{L} \ \wedge \ x \preceq_{\mathcal{P}} y) \implies x \in \mathcal{L}. \tag{1}$$

It is an upper set if $y \in \mathcal{L}$ and $x \succeq_{\mathcal{P}} y$ implies $x \in \mathcal{L}$. It is directed if it is either a lower or an upper set.