Title
Symbolic control design for monotone systems with directed specifications

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

Publication Date
2017-09-01

DOI
10.1016/j.automatica.2017.04.060

Peer reviewed

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
Available online 30 May 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 $\leq_{\mathcal{P}}$ where for all $p_1, p_2, p_3 \in \mathcal{P}$ the binary relation satisfies (1) $p_1 \leq_{\mathcal{P}} p_1$, (2) if $p_1 \leq_{\mathcal{P}} p_2$ and $p_2 \leq_{\mathcal{P}} p_1$ then $p_1 = p_2$ and, (3) if $p_1 \leq_{\mathcal{P}} p_2$ and $p_2 \leq_{\mathcal{P}} p_3$ then $p_1 \leq_{\mathcal{P}} p_3$. We define $\geq_{\mathcal{P}}$ so that $p_1 \geq_{\mathcal{P}} p_2$ holds if and only if $p_2 \leq_{\mathcal{P}} p_1$. If neither $p_1 \leq_{\mathcal{P}} p_2$ nor $p_1 \geq_{\mathcal{P}} p_2$ holds, we say that p_1 and p_2 are *incomparable*. An interval $[a, b] \subseteq \mathcal{P}$ is the set $\{x \in \mathcal{P} : a \leq_{\mathcal{P}} x \leq_{\mathcal{P}} b\}$. Given a collection of partially ordered sets $\mathcal{P}_i$ and relations $\leq_{\mathcal{P}_i}$ 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 $p_1 \leq_{\mathcal{P}} p_2$ holds if and only if $\pi_i(p_1) \leq_{\mathcal{P}_i} \pi_i(p_2)$ for all $i \in \mathcal{A}$. Similarly, a partial ordering $p \leq_{\mathcal{P}^\omega} q$ between a pair of infinite sequences $p = p_0 p_1 \ldots$ and $q = q_0 q_1 \ldots$ holds if and only if $p_k \leq_{\mathcal{P}} q_k$ for all k. A function between partially ordered sets $f : \mathcal{P} \mapsto \mathcal{R}$ is a *monotone function* if $p_1 \leq_{\mathcal{P}} p_2$ implies $f(p_1) \leq_{\mathcal{R}} f(p_2)$ 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 $x, y \in \mathcal{P}$

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

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

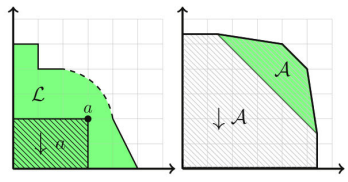

Fig. 1. A lower set in $\mathcal{L} \subset \mathcal{R}^2_{\geq 0}$ with the standard ordering. The principal lower set $\downarrow a$ with associated point a is a subset of $\mathcal{L}$. The patterned set $\downarrow \mathcal{A}$ includes all points for which there exists a point in $\mathcal{A}$ that is greater.

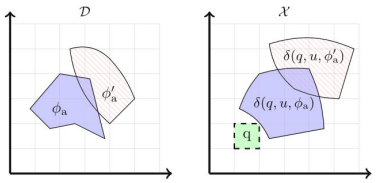

Fig. 2. Illustration of Lemma 5. The patterned assumption set ϕ'_a dominates the solid assumption set ϕ_a in that $\downarrow \phi_a \subseteq \downarrow \phi'_a$. The reachable sets they induce also preserve this ordering so the images $\downarrow \delta(q, u, \phi_a) \subseteq \downarrow \delta(q, u, \phi'_a)$.

A lower set $\mathcal{L}$ is illustrated in Fig. 1 (left). When $a \in \mathcal{P}$, let $\downarrow a = \{x \in \mathcal{P} : x \leq_{\mathcal{P}} a\} \subseteq \mathcal{P}$ and $\uparrow a = \{x \in \mathcal{P} : a \leq_{\mathcal{P}} x\}$. When $\mathcal{A} \subseteq \mathcal{P}$ then its *lower closure* is $\downarrow \mathcal{A} = \bigcup_{a \in \mathcal{A}} \downarrow a$, while its *upper closure* is $\uparrow \mathcal{A} = \bigcup_{a \in \mathcal{A}} \uparrow a$. By construction, $\downarrow \mathcal{A}$ is a lower set of $\mathcal{P}$ and $\mathcal{A} \subseteq \downarrow \mathcal{A}$; see Fig. 1 (right). Moreover, the preimage $f^{-1}(\mathcal{A})$ of a lower set $\mathcal{A}$ with a monotone function is also lower, so $f^{-1}(\mathcal{A}) = \downarrow f^{-1}(\mathcal{A})$. Interpreting the specification ϕ as a subset of a signal space leads to the following definition.

Definition 3. A directed specification ϕ is a directed subset of $\mathcal{Y}^\omega$ with the partial order relation $\leq_{\mathcal{Y}^\omega}$.

Directed specifications can include complex temporal dependencies that incorporate requirements enforcing safety, reachability, and liveness for directed (possibly non-convex) sets. Common examples of lower specifications in the control theory literature are norm-bounded non-negative signals. Linear temporal logic (LTL) is another specification language that has been studied in the hybrid systems community, and a fragment that ensures the directed property was identified in Kim et al. (2016). A more detailed introduction of LTL and the directed fragment appears in Section 7.

3.3. Monotone systems

Definition 4. An open deterministic, discrete time transition system $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, \delta, \mathcal{Y}, h)$ with transitions $\delta : \mathcal{X} \times \mathcal{U} \times \mathcal{D} \mapsto \mathcal{X}$ is monotone if for all $u \in \mathcal{U}$

$$x_1 \leq_{\mathcal{X}} x_2 \ \wedge \ d_1 \leq_{\mathcal{D}} d_2 \Longrightarrow \delta(x_1, u, d_1) \leq_{\mathcal{X}} \delta(x_2, u, d_2)$$

and the output function $h : \mathcal{X} \mapsto \mathcal{Y}$ is monotone.

The following lemma shows that monotone dynamics preserve the assumption set relation $\downarrow \phi_a \subseteq \downarrow \phi'_a$, as illustrated in Fig. 2.

Lemma 5. *If $\downarrow \phi_a \subseteq \downarrow \phi'_a$ then for all $q \subseteq \mathcal{X}$, $u \in \mathcal{U}$*

$$\downarrow \delta(q, u, \phi_a) \subseteq \downarrow \delta(q, u, \phi'_a). \tag{2}$$

An analogous property also holds where $\uparrow \phi_a \subseteq \uparrow \phi'_a$ implies $\uparrow \delta(q, u, \phi_a) \subseteq \uparrow \delta(q, u, \phi'_a)$.

4. Directed alternating simulation relations

This section defines directed alternating simulation relations which are used to refine a controller from an abstraction T' to a concrete system T when specifications are directed.

Definition 6. Let two nonreactive systems $T' = (\mathcal{X}', \mathcal{X}'_0, \mathcal{U}', \Delta', \mathcal{Y}, h')$ and $T = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \Delta, \mathcal{Y}, h)$ have a common partially ordered output space $\mathcal{Y}$. T upper (resp. lower) alternatingly simulates T', denoted as $T' \preceq_{UAS} T$ (resp. $T' \preceq_{LAS} T$), if there exists a relation $R \subseteq \mathcal{X} \times \mathcal{X}'$ such that

(1) For all $x'_0 \in \mathcal{X}'_0$, there exists an $x_0 \in \mathcal{X}_0$ such that $(x_0, x'_0) \in R$.
(2) For all $(x, x') \in R$, $h(x) \leq_{\mathcal{Y}} h'(x')$ (resp. $h'(x') \leq_{\mathcal{Y}} h(x)$).
(3) For all x_1, x'_1 such that $(x_1, x'_1) \in R$ and $u' \in \mathcal{U}'$, there exists a $u \in \mathcal{U}$ such that for all x_2 satisfying $(x_1, u, x_2) \in \Delta$ there exists an x'_2 satisfying $(x'_1, u', x'_2) \in \Delta'$ and $(x_2, x'_2) \in R$.

The only difference between this relation from traditional alternating simulation relations, as detailed by Alur, Henzinger, Kupferman, and Vardi (1998) and Tabuada (2009), is contained in the second condition. Traditional alternating simulation relations impose output equivalence, while the directed version relaxes that condition to an ordering relation.

Definition 7. Two nonreactive transition systems T, T' are mutually upper alternatingly similar, denoted with $T \tilde{\preceq}_{UAS} T'$, if both $T' \preceq_{UAS} T$ and $T \preceq_{UAS} T'$ (different relations are permitted for each direction). Likewise, they are mutually lower alternatingly similar, denoted with $T \tilde{\preceq}_{LAS} T'$, if both $T' \preceq_{LAS} T$ and $T \preceq_{LAS} T'$.

Lower specifications can be paired with upper alternating simulation relations to refine a controller satisfying the specification from one system to the other.

Theorem 8. *If $T' \preceq_{UAS} T$ (resp. $T' \preceq_{LAS} T$) and there exists a control strategy $(\mathcal{X}_0^{\mathbb{C}'}, \mathbb{C}')$ such that the closed loop behaviors $\mathcal{B}_{T'}(\mathcal{X}_0^{\mathbb{C}'}, \mathbb{C}')$ of system T' all satisfy lower (resp. upper) specification $\phi \subseteq \mathcal{Y}^\omega$, then there exists a control strategy $(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C})$ such that $\mathcal{B}_T(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C}) \subseteq \phi$.*

Proof. We prove the case when $T' \preceq_{UAS} T$ and ϕ is a lower specification. Let $\mathcal{X}_0^{\mathbb{C}} := h^{-1}(\downarrow h'(\mathcal{X}_0^{\mathbb{C}'}))$. This set is nonempty via the first and second conditions of Definition 6, and for all $x_0 \in \mathcal{X}_0^{\mathbb{C}}$ there exists an $x'_0 \in \mathcal{X}_0^{\mathbb{C}'}$ such that $(x_0, x'_0) \in R$. By iterating the third condition n times, for all sequences $u' = u'_0 u'_1 \dots u'_{n-1}$ there exists a sequence $u = u_0 u_1 \dots u_{n-1}$ such that for all possible sequences $x = x_0 x_1 \dots x_n$ that are consistent with the dynamics of T' there exists an $x' = x'_0 x'_1 \dots x'_n$ where $(x_i, x'_i) \in R$ and $h(x_i) \leq_{\mathcal{Y}} h'(x'_i)$ for all $i \in [0, \dots, n]$. Given $\mathbb{C}'(x'_0 x'_1 \dots x'_n) = u'_n$ let $\mathbb{C} : \mathcal{X}^* \mapsto \mathcal{U}$ be defined such that $\mathbb{C}(x_0 x_1 \dots x_n) = u_n$ as defined immediately above. It follows from ϕ's lower property that if all trajectories of T' under control strategy $(\mathcal{X}_0^{\mathbb{C}'}, \mathbb{C}')$ satisfy ϕ then all possible trajectories of T under $(\mathcal{X}_0^{\mathbb{C}}, \mathbb{C})$ also satisfy ϕ.

Consider an open system $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, \delta, \mathcal{Y}, h)$ and two distinct disturbance environments ϕ_a, ϕ'_a inducing nonreactive versions $\Lambda(\Sigma, \phi_a)$ and $\Lambda(\Sigma, \phi'_a)$. Proposition 9 shows that for monotone systems Σ, a simple set containment relation between ϕ_a and ϕ'_a implies a directed alternating simulation relation between $\Lambda(\Sigma, \phi_a)$ and $\Lambda(\Sigma, \phi'_a)$.

Proposition 9. *Let deterministic monotone system Σ and environment assumptions ϕ_a, ϕ'_a have associated nonreactive systems $\Lambda(\Sigma, \phi_a)$ and $\Lambda(\Sigma, \phi'_a)$. If*

- $\downarrow \phi_a \subseteq \downarrow \phi'_a$ *then* $\Lambda(\Sigma, \phi'_a) \preceq_{UAS} \Lambda(\Sigma, \phi_a)$.
- $\uparrow \phi_a \subseteq \uparrow \phi'_a$ *then* $\Lambda(\Sigma, \phi'_a) \preceq_{LAS} \Lambda(\Sigma, \phi_a)$.
- $\downarrow \phi_a = \downarrow \phi'_a$ *then* $\Lambda(\Sigma, \phi_a) \tilde{\preceq}_{UAS} \Lambda(\Sigma, \phi'_a)$.
- $\uparrow \phi_a = \uparrow \phi'_a$ *then* $\Lambda(\Sigma, \phi_a) \tilde{\preceq}_{LAS} \Lambda(\Sigma, \phi'_a)$.

Proof. For simplicity, we only prove the upper cases where the (mutual) upper alternating simulation relation is $R = \{(x, x') : x \preceq_{\mathcal{X}} x'\} \subseteq \mathcal{X} \times \mathcal{X}$. By definition, the sets $\mathcal{X}_0 = \mathcal{X}'_0$ so the first condition in Definition 6 is satisfied. The second condition is satisfied from the monotonicity of the output map. Finally, by setting $u = u'$, $\downarrow \phi_a \subseteq \downarrow \phi'_a$ implies that $\downarrow \delta(q, u, \phi_a) \subseteq \downarrow \delta(q, u', \phi'_a)$ via Lemma 5 and for all $x_2 \in \delta(q, u, \phi_a)$ there exists an $x'_2 \in \delta(q, u', \phi'_a)$ such that $x_2 \preceq_{\mathcal{X}} x'_2$. When $\downarrow \phi_a = \downarrow \phi'_a$, the mutual upper simulation claim follows because $\downarrow \phi_a \subseteq \downarrow \phi'_a$ and $\downarrow \phi'_a \subseteq \downarrow \phi_a$ both hold.

Theorem 8's controller refinement result is central in subsequent sections. It is used to develop sparse abstractions of monotone systems and to develop contracts between interconnected systems in Sections 5 and 6 respectively. We leverage Proposition 9 to systematically search for contract parameters.

5. Sparse abstractions of monotone systems

Monotone dynamics have been used for efficient reachability computations via bounding boxes and constructing finite abstractions by Coogan and Arcak (2015); Meyer et al. (2015), and Moor and Raisch (2002). These abstractions are then provided to a control synthesis solver. When the specification has a direction, the number of transitions in this abstraction can be dramatically reduced without introducing any conservatism to the synthesis algorithm. For clarity, the rest of this section assumes that the specification ϕ is lower. A few modifications are required when ϕ is upper, and these are highlighted in Remark 16.

5.1. State space box partitions

Let $\mathcal{X}$ be a closed and bounded subset of $\mathbb{R}^d$. The ordering $\leq$ is the standard coordinate-wise ordering and for $x, y \in \mathbb{R}^d$ $x < y$ means $x_i < y_i$ for all $i \in \{1, \ldots, d\}$. Half closed-open intervals are $[x, y] = \{z : x < z \leq y\}$ and $[x, y) = \{z : x \leq z < y\}$.

Let $\mathcal{Q}$ be a set of symbolic states with partial ordering $\leq_{\mathcal{Q}}$, and $\{I_q\}_{q \in \mathcal{Q}}$ be a collection of intervals that forms a partition of $\mathcal{X}$.

Definition 10. Symbolic set $\mathcal{Q}$ and ordering $\leq_{\mathcal{Q}}$ constitute a finite interval partition of $\mathcal{X}$ if:

- For all $q \in \mathcal{Q}$, there exist unique x_1^q, x_2^q such that
$$I_q = (x_1^q, x_2^q] \text{ or } [x_1^q, x_2^q]. \tag{3}$$
- $\bigcup_{q \in \mathcal{Q}} I_q = \mathcal{X}$ and $I_{q_1} \cap I_{q_2} = \emptyset$ if $q_1 \neq q_2$.
- For all $q_1, q_2 \in \mathcal{Q}$, $q_1 \leq_{\mathcal{Q}} q_2$ if and only if
$$x_2^{q_1} \leq_{\mathcal{X}} x_2^{q_2}. \tag{4}$$

Moreover, the discrete output map $h' : \mathcal{Q} \mapsto \mathcal{Y}$ is

$$h'(q) = h(x_2^q). \tag{5}$$

5.2. Box abstractions of monotone systems

We provide a brief overview of an existing monotonicity based abstraction procedure for systems with box partitions employed by Coogan and Arcak (2015). Let $\mathcal{U}$ be a finite set of control inputs and consider an assumption set $\phi_a = \cup_{m=1}^M [d_1^m, d_2^m]$ consisting of boxes. Recall that a monotone system's transition function δ is deterministic, but an exogenous disturbance acts as the source of non-determinism. For all $u \in \mathcal{U}$, the set of possible reachable states from an interval can efficiently be over-approximated by an interval generated by evaluating δ at $2M$ points in $\mathcal{X} \times \mathcal{D}$:

Lemma 11. *For all $u \in \mathcal{U}$, $x_1 \leq_{\mathcal{X}} x_2$, and $d_1 \leq_{\mathcal{D}} d_2$:*

$$\delta([x_1, x_2], u, [d_1, d_2]) \subseteq [\delta(x_1, u, d_1), \delta(x_2, u, d_2)]. \tag{6}$$

This reachability over-approximation can be used to efficiently construct a discrete box abstraction of the original system.

Definition 12. Given a deterministic monotone system $\Sigma = (\mathcal{X}, \mathcal{X}_0, \mathcal{U}, \mathcal{D}, \delta, \mathcal{Y}, h)$ subjected to assumption set $\phi_a = \cup_{m=1}^M [d_1^m, d_2^m]$, its box abstraction $\mathcal{BA}(\Sigma, \phi_a) = (\mathcal{Q}, \mathcal{Q}_0, \mathcal{U}, \Delta', \mathcal{Y}, h')$ is a tuple with state space $\mathcal{Q}$, initial states $\mathcal{Q}_0$ such that $q \in \mathcal{Q}_0$ if and only if

$$\exists x \in \mathcal{X}_0 \text{ where } x \in q, \tag{7}$$

input set $\mathcal{U}$, transition relation $\Delta' \subseteq \mathcal{Q} \times \mathcal{U} \times \mathcal{Q}$ such that for $I_q = (x_1, x_2]$, $I_{q'} = (x'_1, x'_2]$, $(q, u, q') \in \Delta'$ if and only if there exists an $m \in \{1, \ldots, M\}$ such that

$$[\delta(x_1, u, d_1^m), \delta(x_2, u, d_2^m)] \cap (x'_1, x'_2] \neq \emptyset, \tag{8}$$

output space $\mathcal{Y}$, and output map h' given by, Eq. (5).

By construction this abstraction is non-blocking. One can check condition (8) by evaluating if $x'_1 <_{\mathcal{X}} \delta(x_i, u, d_i^m) \leq_{\mathcal{X}} x'_2$ for both $i \in \{1, 2\}$. Box abstractions satisfy $\mathcal{BA}(\Sigma, \phi_a) \preceq_{UAS} \Lambda(\Sigma, \phi_a)$; the proof is omitted because it is similar to Proposition 14's proof below.

5.3. Sparse abstractions of monotone systems

The procedure above may result in abstractions with a substantial number of transitions. More transitions result in runtime and memory overhead for controller synthesis algorithms that explicitly store the discrete transition system in memory, e.g., as an adjacency matrix. Reducing the number of transitions also reduces the chance that the abstraction will exhibit spurious trajectories, which are trajectories that appear in an abstraction but are not possible under the continuous dynamics. With a sparse abstraction, we only take worst case transitions for a fixed disturbance d and significantly reduce the number of transitions in the abstraction.

Definition 13 (*Sparse Abstraction*)**.** Given a monotone system Σ with respect to $\phi_a = \cup_{m=1}^M [d_1^m, d_2^m]$, a sparse abstraction $\mathcal{SA}(\Sigma, \phi_a) = (\mathcal{Q}, \mathcal{Q}_0, \mathcal{U}, \Delta', \mathcal{Y}, h')$ is identical to the box abstraction $\mathcal{BA}(\Sigma, \phi_a)$ except it has a different transition relation. For $q = (x_1, x_2]$, $q' = (x'_1, x'_2]$, the new relation $\Delta' \subseteq \mathcal{Q} \times \mathcal{U} \times \mathcal{Q}$ contains transitions $(q, u, q') \in \Delta'$ if and only if there exists an $m \in \{1, \ldots, M\}$ such that

$$\delta(x_2, u, d_2^m) \in (x'_1, x'_2]. \tag{9}$$

The number of transitions out of each state and control action pair (q, u) is at most M and independent of $\mathcal{X}$'s dimension. Fig. 3 (right) illustrates this abstraction procedure for $M = 1$. We next show that $\mathcal{SA}(\Sigma, \phi_a) \preceq_{UAS} \Lambda(\Sigma, \phi_a)$.

Proposition 14. *If $\mathcal{SA}(\Sigma, \phi_a)$ is constructed via Definition 13 with the finite interval partition given in Definition 10, then $\mathcal{SA}(\Sigma, \phi_a) \preceq_{UAS} \Lambda(\Sigma, \phi_a)$ with the system relation*

$$R = \{(x, q) : x \leq_{\mathcal{X}} x_2^q\} \subseteq \mathcal{X} \times \mathcal{Q}. \tag{10}$$

Proof. Consider the three conditions on relation R in Definition 6. Since $\mathcal{Q}$ is a partition of $\mathcal{X}$, each $x_0 \in \mathcal{X}_0$ can be matched with a $q_0 \in \mathcal{Q}$ via (7) and a corresponding $x_2^{q_0}$ via Eq. (3). The second

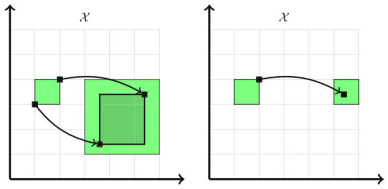

Fig. 3. Construction of a box abstraction (left) versus construction of a sparse abstraction (right) given a lower specification. For a fixed d, the sparse abstraction only evaluates one point and does not use a box to over-approximate the reachable set; it instead only considers the worst case transition, which is the upper right if the specification is lower.

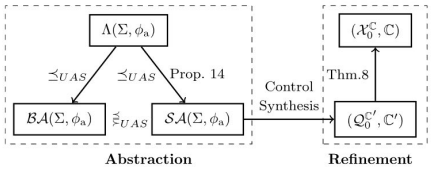

Fig. 4. The abstraction and controller refinement workflow. From a nonreactive system $\Lambda(\Sigma, \phi_s)$ we generate a box or sparse abstraction. After controller synthesis for $\mathcal{SA}(\Sigma, \phi_a)$, the result of Theorem 8 is used to refine an abstract controller $(\mathcal{Q}_0^C, \mathbb{C})$ to a concrete one $(\mathcal{X}_0^C, \mathbb{C})$.

condition holds from monotonicity of $h : \mathcal{X} \mapsto \mathcal{Y}$, which implies $h(x) \preceq_{\mathcal{Y}} h(x_2^q)$. Finally, for the last condition note that for all $d \in \phi_a$ there exists a d_2^m such that $d \leq d_2^m$. With such a d_2^m , $\delta(x, d) \leq \delta(x_2^q, d_2^m)$ and $\delta(x, d) \in \downarrow \delta(x_2^q, d_2^m)$ by monotonicity of δ.

The box and sparse abstractions are also mutually upper alternatingly similar because the reachable sets are upper bounded by the same upper right corner.

Corollary 15. $\mathcal{SA}(\Sigma, \phi_a) \tilde{\preceq}_{UAS} \mathcal{BA}(\Sigma, \phi_a)$.

5.4. Summarizing directed controller refinement

The ultimate conclusion of Proposition 14 and Corollary 15 can be summarized in the following relationship:

$$\mathcal{SA}(\Sigma, \phi_a) \tilde{\preceq}_{UAS} \mathcal{BA}(\Sigma, \phi_a) \preceq_{UAS} \Lambda(\Sigma, \phi_a). \tag{11}$$

Proposition 14 and Theorem 8 jointly imply that a control strategy for $\mathcal{SA}(\Sigma, \phi_a)$ enforcing a lower specification ϕ can be refined to a control strategy for the concrete system $\Lambda(\Sigma, \phi_a)$. Fig. 4 summarizes the system abstraction and controller refinement procedure. We require ϕ to be compatible with the partitions $\mathcal{Q}$ in that $\phi \subseteq \mathcal{Y}_{\mathcal{Q}}^{\omega} \subseteq \mathcal{Y}^{\omega}$ where $\mathcal{Y}_{\mathcal{Q}} = \{y \in \mathcal{Y} : y = h'(q) \text{ for some } q \in \mathcal{Q}\}$. This is a common assumption in abstraction based synthesis. Otherwise some care is required when translating ϕ from a continuous domain $\mathcal{X}$ to a discrete partition $\mathcal{Q}$.

Remark 16. *If ϕ is an upper specification, then Eqs. (3), (4), (5), (8), (9), and (10) are respectively replaced with*

$$l_q = [x_1^q, x_2^q) \text{ or } [x_1^q, x_2^q] \tag{12}$$
$$x_1^{q_1} \preceq_{\mathcal{X}} x_1^{q_2} \tag{13}$$
$$h'(q) = h(x_1^q) \tag{14}$$
$$[\delta(x_1, u, d_1^m), \delta(x_2, u, d_2^m)] \cap [x_1', x_2') \neq \emptyset \tag{15}$$
$$\delta(x_1, d_1^m) \in [x_1', x_2') \tag{16}$$
$$R = \{(x, q) \in \mathcal{X} \times \mathcal{Q} : x_1^q \preceq_{\mathcal{X}} x\}. \tag{17}$$

Proposition 14 and Corollary 15 now refer to (mutual) lower alternating simulations. It is possible for ϕ to be both upper and lower, but when $\mathcal{X}$ is $\mathbb{R}^d$ this is only true when $\phi = \mathcal{X}^{\omega}$ or the empty set.

6. Compositional synthesis with directed contracts

This section shows that directed specifications ensure that contracts between monotone systems naturally have a directed property and are amenable to binary search heuristics.

6.1. Interconnections

Consider a collection of N open sub-systems $\Sigma^n = (\mathcal{X}^n, \mathcal{X}_0^n, \mathcal{U}^n, \mathcal{D}^n, \delta^n, \mathcal{Y}^n, h^n)$ for $n = 1, \ldots, N$ and an interconnection function $\mathcal{I} : \prod_{n=1}^N \mathcal{Y}^n \times \mathcal{D}^{\text{exog}} \mapsto \prod_{n=1}^N \mathcal{D}^n$ where $\mathcal{D}^{\text{exog}}$ is an uncontrolled disturbance set that is exogenous to all systems. The interconnection function $\mathcal{I}$ assumes that Σ^n's disturbance is a function of the outputs, i.e. $\mathcal{I}^n : \prod_{i \neq n} \mathcal{Y}^i \times \mathcal{D}^{\text{exog}} \mapsto \mathcal{D}^n$. Often, the disturbance set $\mathcal{D}^n$ has smaller dimensionality than $\prod_{i \neq n} \mathcal{Y}^i \times \mathcal{D}^{\text{exog}}$ and does not depend on all system outputs. The following assumption on the separability of the specification is required for compositional synthesis.

Assumption 17. The global specification ϕ_{spec} has the form $\phi_{\text{spec}} = \bigwedge_{n=1}^N \phi_{\text{spec}}^n$ where ϕ_{spec}^n all have the same direction.

We now state the compositional controller synthesis problem.

Problem 18. Let there be a set of N transition systems $\Sigma^n = (\mathcal{X}^n, \mathcal{X}_0^n, \mathcal{U}^n, \mathcal{D}^n, \delta^n, \mathcal{Y}^n, h^n)$ with corresponding specifications ϕ_{spec}^n. Assume that for all time k, $d_k^n \in \mathcal{D}^n$ satisfies the interconnection function $\mathcal{I} : \prod_{n=1}^N \mathcal{Y}^n \times \mathcal{D}^{\text{exog}} \mapsto \prod_{n=1}^N \mathcal{D}^n$. Find a satisfying control strategy $((\mathcal{X}_0^{C'})^n, \mathbb{C}^n)$ for each Σ^n such that the interconnected system satisfies $\phi_{\text{spec}} = \bigwedge_{n=1}^N \phi_{\text{spec}}^n$.

6.2. Assume–guarantee contracts

One system's environment consists of the actions of adjacent systems plus an exogenous disturbance. When this environment consists of adjacent sub-systems, we incorporate promises between networks in the synthesis objective. In our framework, an *assume–guarantee contract* consists of a safety *guarantee* by imposing a bound on output variables, and *assumptions* in the form of bounds on input variables. If for each sub-system Σ^n a control strategy $(\mathcal{X}_0^{C^n}, \mathbb{C}^n)$ can be synthesized that guarantees satisfaction of the sub-system's specification (1) under the assumption that all other systems fulfill their promises and (2) such that the subsystem's promises to every other sub-system is satisfied, then the interconnected system with initial states $\prod_{n=1}^N \mathcal{X}_0^{C^n}$ will satisfy the specification ϕ_{spec}.

6.3. Directed contracts

We assume that the interconnection $\mathcal{I}$ is a monotone function. All systems Σ^n must have lower specifications or upper specifications. This assumption is satisfied when the original system has a directed objective before decomposition. Consider a single system Σ^n with a lower specification ϕ_{spec}^n. Its guarantees to other systems are set as upper bounds on its outputs.

6.3.1. Encoding guarantees

We restrict system Σ^n's behavior by imposing an additional requirement on its specification. We impose that Σ^n has an associ-

ated finite collection of output values $\nu^n = \{\nu_j^n : j = 1, \ldots, |\nu^n|\} \subseteq \mathcal{Y}^n$. The guarantee set is defined as

$$\phi_g^n(\nu^n) \coloneqq \prod_{i=0}^{\infty} \{y^n : y^n \in\downarrow \nu^n\} \subseteq (\mathcal{Y}^n)^\omega. \tag{18}$$

The infinite Cartesian product is interpreted as an invariant over all time. Effectively, the guarantee restricts output behaviors to be upper bounded by a set of points ν^n for all time and enforces the state to remain within the region $\{x^n : h^n(x^n) \in\downarrow \nu^n\}$. The new specification ϕ^n is simply the conjunction of the original specification with the guarantee $\phi^n \coloneqq \phi^n_{\text{spec}} \wedge \phi^n_g(\nu^n)$.

6.3.2. Encoding assumptions

Given a set of guarantees from other systems, we translate those guarantees to possible environment behaviors that Σ^n can observe. Consider an upper bound of the exogenous disturbance set through the relation $\mathcal{D}^{\text{exog}} \subseteq \bigcup_{j=1}^{M} \downarrow d_j^{\text{exog}}$ for some collection of d_j^{exog}'s. Let the collection $\eta^n = \{\eta_j^n : j = 1, \ldots, |\eta^n|\}$ of upper bounds on the disturbance be generated by applying the interconnection function to the sets of d_j^{exog}, guarantee values $\nu_j^i \in \nu^i$ for all $i \neq n$. That is,

$$\eta^n = \left\{\mathcal{I}^n(\nu_j^1, \ldots, \nu_j^{n-1}, \nu_j^{n+1}, \ldots, \nu_j^N, d_j^{\text{exog}}) : d_j^{\text{exog}}, i \neq n, \nu_j^i \in \nu^i\right\}. \tag{19}$$

Because the guarantee from other systems does not change over time, it suffices to represent the assumption set as a subset of $\mathcal{D}^n$ instead of $(\mathcal{D}^n)^\omega$. The assumption set as defined in Section 2.2 is

$$\phi_a^n(\eta^n) \coloneqq \{d^n : d^n \in\downarrow \eta^n\} \subseteq \mathcal{D}^n. \tag{20}$$

6.4. Identifying contract parameters via binary search

Given any synthesis procedure to satisfy a directed specification, we systematically prune the parameter search space based on the success or failure of this procedure by using the results in Theorem 8 and Proposition 9. As the guarantees ν are varied so that synthesis is harder for one system, it then becomes easier for adjacent systems to synthesize control strategies.

Proposition 19. *Let Σ^n be a monotone system. If sets ν^n, $\hat{\nu}^n$ satisfying $\downarrow \nu^n \subseteq\downarrow \hat{\nu}^n$ generate guarantees $\phi_g^n(\nu^n)$ and $\phi_g^n(\hat{\nu}^n)$ and there exists a control strategy $(\mathcal{X}_0^{\mathcal{C}^n}, \mathbb{C}^n)$ such that $\Lambda^n(\Sigma^n, \eta^n)$ satisfies guarantee $\phi_g^n(\nu^n)$ then it will also satisfy guarantee $\phi_g^n(\hat{\nu}^n)$.*

Proof. Follows from a simple set containment argument because specification $\phi_g^n(\hat{\nu}^n)$ is implied by $\phi_g^n(\nu^n)$.

Proposition 20 follows from Proposition 9 and Theorem 8.

Proposition 20. *Let Σ^n be a monotone system. If sets η^n, $\hat{\eta}^n$ satisfying $\downarrow \eta^n \subseteq\downarrow \hat{\eta}^n$ generate assumptions $\phi_a^n(\eta^n)$, $\phi_a^n(\hat{\eta}^n)$ and there exists a control strategy $(\mathcal{X}_0^{\hat{\mathcal{C}}^n}, \hat{\mathbb{C}}^n)$ such that $\Lambda^n(\Sigma^n, \phi_a^n(\hat{\eta}^n))$ satisfies a lower specification ϕ, then there exists a control strategy $(\mathcal{X}_0^{\mathcal{C}^n}, \mathbb{C}^n)$ such that $\Lambda^n(\Sigma^n, \phi_a^n(\eta^n))$ satisfies ϕ.*

Fig. 5 depicts a space of contract parameters, with different systems preferring opposite corners of the parameter space. The search algorithm's goal is to find the intersection of the two regions. After sampling a set of parameters and observing whether an abstract controller exists, we prune orthants (patterned regions) of the contract parameter space. From Proposition 19 and 20, sampling within a previously pruned space yields no progress and does not help guide the algorithm to the intersection. The search heuristic from Kim et al. (2016) is used to identify new contract parameters. When ϕ is an upper specification inequalities (18) and (20) should be reversed and the downarrows $\downarrow$ in Propositions 19 and 20 should be replaced by uparrows $\uparrow$.

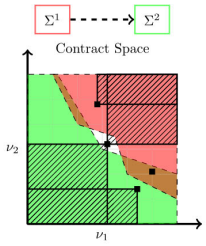

Fig. 5. Example of binary search over Σ^1's two dimensional guarantee space $\nu^1 = (\nu_1^1, \nu_2^1)$. With lower specifications, system Σ^1 prefers bigger ν because the guarantee is easier to satisfy, while Σ^2 prefers the lower region because its environment is weaker. To find the intersection region, we sample and prune the space by discarding an appropriate orthant (patterned regions) of the search space.

7. Example: Synthesizing a signaling controller

We consider the problem of synthesizing a signal controller for a traffic network, where the linear temporal logic (LTL) objective aims to mitigate congestion. Following a brief introduction to LTL, we introduce a directed subset and give an overview of LTL synthesis.

7.1. Directed linear temporal logic

Temporal logics are logical formalisms for expressing specifications as sets of admissible signals (Pnueli, 1977). We restrict our interest to signals over the discrete output space $\mathcal{Y}_{\mathbb{Q}}$. An atomic proposition μ is a subset of $\mathcal{Y}_{\mathbb{Q}}$. Linear temporal logic specification is constructed with the grammar

$$\phi \coloneqq \top | \mu | \neg\phi | \phi^1 \wedge \phi^2 | \bigcirc \phi | \phi^1 \mathbf{U} \phi^2 \tag{21}$$

where $\neg$ is Boolean negation, and $\wedge$ is a Boolean AND. Specification $\bigcirc\phi$ is true if and only if ϕ is true at the next time step. Specification $\phi^1 \mathbf{U} \phi^2$ is true if there exists a future time k such that ϕ^2 is true at time k and ϕ^1 is true until k. One can derive additional temporal operators $\diamond\phi = \top \mathbf{U} \phi$ for "ϕ is eventually true" and $\square\phi = \neg(\diamond\neg\phi)$ for "ϕ is always true". The guarantee (18) can be rewritten in LTL as $\square(y^n \in\downarrow \nu^n)$. In Kim et al. (2016), we previously identified a fragment of linear temporal logic that guarantees that a specification is directed:

Proposition 21. *Let all atomic propositions $\mu_l, \mu_u \subseteq \mathcal{P}$ be lower and upper sets respectively. A temporal logic specification $\phi_d \subseteq \mathcal{P}^\omega$ constructed with the following grammar is directed*

$$\phi_d \coloneqq \phi_l | \phi_u$$
$$\phi_u \coloneqq \top | \mu_u | \neg\phi_l | \mu_u^1 \wedge \mu_u^2 | \bigcirc \phi_u | \phi_u^1 \mathbf{U} \phi_u^2$$
$$\phi_l \coloneqq \top | \mu_l | \neg\phi_u | \mu_l^1 \wedge \mu_l^2 | \bigcirc \phi_l | \phi_l^1 \mathbf{U} \phi_l^2.$$

Proof. We provide a sketch of the full proof in Kim et al. (2016). A lower specification ϕ_l can be constructed by operations that preserve the lower property. A specification consisting of only a lower atomic proposition is lower. The set of lower sets of $\mathcal{P}$ is closed under union and intersection, and the complement of a lower specification is upper. A time delay of a lower specification ϕ_l with the $\bigcirc$ operator preserves the lower property. The $\mathbf{U}$ operator is interpreted as an appropriate combination of union and intersections in $\mathcal{P}^\omega$. Identical arguments hold for upper specifications ϕ_u.

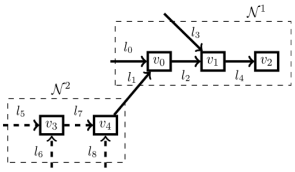

Fig. 6. Two sub-networks $\mathcal{N}^1$, $\mathcal{N}^2$. Solid arrows are links in $\mathcal{N}^1$ while dashed arrows are links in $\mathcal{N}^2$.

An automata theoretic method to synthesize a controller involves translating the LTL specification ϕ to a Rabin automaton that monitors specification satisfaction. A product automaton between the abstract dynamics and Rabin automaton is constructed. The non-determinism in the product automaton's transition relation is viewed as an adversarial environment. A game between the controller and environment is solved to yield the controller $\mathbb{C}$ and the initial states $x_0^{\mathbb{C}}$. More detailed descriptions of this control synthesis procedure are provided by Baier and Katoen (2008) and Yordanov et al. (2012).

7.2. Network dynamics

Consider the road network depicted in Fig. 6 that consists of two smaller networks and with parameters given in Fig. 7. The network dynamics are derived from Daganzo's (1994) cell transmission model (CTM). A network with no diverging junctions (nodes with multiple outgoing segments) was previously proven to exhibit monotone dynamics by Coogan and Arcak (2015) under some mild technical conditions which will be addressed. The network state x consists of vehicle occupancies on each link, where x_i represents the occupancy on link l_i. Each link has an associated maximum capacity x_i^{cap}. Both networks are controlled by a set of signalized intersections. Each vertex v_j has an associated finite set of traffic intersection configurations, where each configuration determines which incoming links are actuated. For instance, v_4 in $\mathcal{N}^2$ may actuate either l_7 or l_8 but not both. A choice of signal configuration for each intersection yields the entire network's control mode.

The flow from one link to another is determined by the notions of supply and demand. A link's demand is the number of vehicles it would like to send to downstream links and its supply is the number of vehicles it may accept from upstream. Link l_j's monotonically increasing demand function $\mathcal{I}_j^{\text{dem}}(x_j, u)$ under control mode u is

$$\mathcal{I}_j^{\text{dem}}(x_j, u) = \begin{Bmatrix} \min(x_j, m_j) & \text{if } l_j \text{ actuated under } u \\ 0 & \text{otherwise} \end{Bmatrix}$$

where, m_j represents a maximum free flow rate. Likewise, link j's supply function $\mathcal{I}_j^{\text{sup}}(x_j)$ is monotonically decreasing with respect to occupancy x_j:

$$\mathcal{I}_j^{\text{sup}}(x_j) = x_j^{\text{cap}} - x_j.$$

The proportion of demand allocated from upstream link j to downstream link i is given by β_{ji}. Supply allocated from downstream link i to upstream link j is given by α_{ji}^u and may depend on the signal mode u. Demand and supply allocations must satisfy the following:

$$\sum_i \beta_{ji} \leq 1 \text{ and } \sum_j \alpha_{ji}^u = 1. \tag{22}$$

Strict inequality for demand allocations β implies that some vehicles are routed to an unmodeled link, and $\beta_{ji} \neq 0$ only if l_j is upstream from link l_i. If link l_j is actuated, the flow exiting link l_j is a function of its demand and downstream link l_i's supply:

$$f_j(\mathcal{I}_j^{\text{dem}}, \mathcal{I}_i^{\text{sup}}) = \min\left(\mathcal{I}_j^{\text{dem}}, \frac{\alpha_{ji}^u}{\beta_{ji}} \mathcal{I}_i^{\text{sup}}\right). \tag{23}$$

For link l_4 which has no downstream link, we assume infinite supply so the right term in the minimization is never active. The next state x_j^+ of link l_j follows a conservation of mass equation

$$x_j^+ = \min\left\{ x_j^{\text{cap}}, x_j - f_j(\mathcal{I}_j^{\text{dem}}, \mathcal{I}_i^{\text{sup}}) + d_j^{\text{exog}} + \sum_{k=0}^{8} \beta_{kj} f_k(\mathcal{I}_k^{\text{dem}}, \mathcal{I}_j^{\text{sup}}) \right\} \tag{24}$$

where l_i is downstream from l_j. The exogenous demand d_i^{exog} may only be nonzero for input links $i \in \{0, 3, 5, 6, 8\}$ and is zero otherwise. The first term in the minimization is used to enforce that the link's current occupancy does not exceed the link's capacity. It is only necessary because exogenous disturbances are present; otherwise, a downstream link's supply prevents vehicle overflow by construction. The interconnection function $\mathcal{I}$ from Section 6 encodes the network topology and is depicted in Fig. 8.

Sub-network $\mathcal{N}^1$ parameters:

$(x_0^{\text{cap}}, \ldots, x_4^{\text{cap}}) = (60, 40, 80, 24, 60)$
$(c_0, \ldots, c_4) = (20, 15, 20, 12, 20)$
$\mathcal{D}^{\text{exog}} := \{d : 0 \leq d \leq [10, 0, 0, 3, 0]\}$
$\beta_{02} = \beta_{24} = 0.8,\ \beta_{12} = \beta_{34} = 1.0,$
$u_{v_0} \in \{\{l_0\}, \{l_0, l_1\}\}, \quad u_{v_1} \in \{\{l_2\}, \{l_2, l_3\}\}$
If l_1 actuated, then $\alpha_{02} = 0.7, \alpha_{12} = 0.3$
If l_3 actuated, then $\alpha_{24} = 0.7, \alpha_{34} = 0.3$
Otherwise $\alpha_{02} = \alpha_{24} = 1$

Sub-network $\mathcal{N}^2$ parameters:

$(x_5^{\text{cap}}, \ldots, x_8^{\text{cap}}) := (30, 18, 30, 30)$
$(c_5, \ldots, c_8) := (12, 6, 14, 14)$
$\mathcal{D}^{\text{exog}} := \{d : 0 \leq d \leq [4, 3, 0, 3]\}$
$u_{v_3} \in \{\{l_5\}, \{l_6\}\}, \quad u_{v_4} \in \{\{l_7\}, \{l_8\}\}$
$\beta_{57} = \beta_{67} = 1$
Link l_7's supply is allocated to the link actuated by v_3.

Inter-network flow parameters:

$\beta_{71} = \beta_{81} = .8$
Link l_1's supply is allocated to link actuated by v_4.

Fig. 7. Network dynamics parameters.

7.3. Monotonicity of network dynamics

The freeway dynamics are monotone with respect to a standard coordinate-wise ordering on states and a disturbance ordering given by higher demand and lower supply under a mild technical condition.

Assumption 22. For all pairs of links l_i, l_j and control modes u such that $\beta_{ji}, \alpha_{ji}^u > 0$:

$$m_i \leq x_i^{\text{cap}} - \frac{\beta_{ji}}{\alpha_{ji}^u} m_j. \tag{25}$$

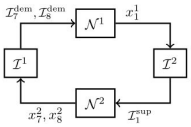

Fig. 8. Input and output variables of the interconnected network depicted in Fig. 6. The roles of the interconnection functions $\mathcal{I}^1, \mathcal{I}^2$ are identical to Section 6. The occupancies on links l_7, l_8 induce demands $\mathcal{I}_7^{\text{dem}}, \mathcal{I}_8^{\text{dem}}$ via $\mathcal{I}^1$. Likewise, the occupancy on link l_1 is transformed to a supply input for $\mathcal{N}^2$ via $\mathcal{I}^2$. Networks $\mathcal{N}^1$, $\mathcal{N}^2$ are both monotone systems when the states adopt a standard ordering and the input ordering is given by higher demand and lower supply.

Physically, Assumption 22 ensures that a link cannot be emptied while simultaneously restricting flow from upstream. Because the saturation flows m_l shrink with a smaller sampling period, the assumption is satisfied for a sufficiently fast sampling rate. Once the larger network is broken apart into smaller networks $\mathcal{N}^1, \mathcal{N}^2$ the two interact with supply and demand as input and output variables. The output functions $h^1(x^1), h^2(x^2)$ are the identity functions and states x^1, x^2 are augmented with superscripts for clarity. Note that the flow exiting link l_j in (23) is monotonically increasing with respect to $\mathcal{I}_j^{\text{dem}}$ and downstream supply $\mathcal{I}_i^{\text{sup}}$. Network $\mathcal{N}^1$ has link l_1 which experiences demand from network $\mathcal{N}^2$. The occupancy at the next time step increases with respect to $\mathcal{I}_7^{\text{dem}}, \mathcal{I}_8^{\text{dem}}$ for all u:

$$(x_1^1)^+ = x_1^1 - f_1(\mathcal{I}_1^{\text{dem}}, \mathcal{I}_2^{\text{sup}}) + \sum_{j=7,8} \beta_{j1} f_j(\mathcal{I}_j^{\text{dem}}, \mathcal{I}_1^{\text{sup}}).$$

For network $\mathcal{N}^2$, links l_7, l_8 are subject to supply arising from congestion in link l_1. As supply $\mathcal{I}_1^{\text{sup}}$ decreases, the occupancy of links l_7, l_8 grows. Thus,

$$(x_j^2)^+ = x_j^2 - f_j(\mathcal{I}_j^{\text{dem}}, \mathcal{I}_1^{\text{sup}}) + d_j^{\text{exog}} + \sum_{k=5,6} \beta_{kj} f_k(\mathcal{I}_k^{\text{dem}}, \mathcal{I}_j^{\text{sup}})$$

for $j \in \{7, 8\}$. Eq. (24) shows that increasing the exogenous disturbance d_j^{exog} to any input link l_j will increase its occupancy.

7.4. Compositional synthesis

Network $\mathcal{N}^1$'s objective is to ensure the on-ramps l_1, l_3 do not persistently overflow:

$$\phi_{\text{spec}}^1 := \bigwedge\nolimits_{i=1,3} \square \diamond (x_i \leq 20). \tag{26}$$

Similarly, network $\mathcal{N}^2$'s objective is to prevent saturation of links l_5, l_6, l_8:

$$\phi_{\text{spec}}^2 := \bigwedge\nolimits_{i=5,8} \square \diamond (x_i \leq 15) \wedge \square \diamond (x_6 \leq 12). \tag{27}$$

Both specifications are lower and satisfy the grammar in Proposition 21. Using Propositions 19 and 20 and binary search, the contract parameters demand guarantee $\nu^2 = 12.5$ and supply guarantee $\nu^1 = 31$ were found to yield admissible control strategies for both networks. These parameters translate to guarantees on the system states

$$\phi_{\text{g}}^1 = \square(x_1^1 \leq 31) \tag{28}$$

$$\phi_{\text{g}}^2 = \square(x_7^2 \leq 12.5 \ \wedge \ x_8^2 \leq 12.5). \tag{29}$$

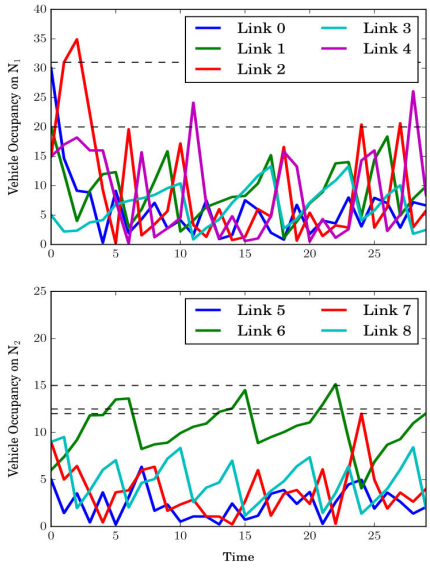

Fig. 9. Simulation graph of two networks satisfying a specification. Horizontal lines signify traffic levels that play a significant role in the specification or guarantee.

Because $\mathcal{I}_7^{\text{dem}}$ and $\mathcal{I}_8^{\text{dem}}$ depend on the control input in this example, (29) is a condition that ensures $\mathcal{I}_7^{\text{dem}}, \mathcal{I}_8^{\text{dem}} \leq 12.5$ under any of $\mathcal{N}^2$'s signal configurations. These guarantees are propagated via the interconnection functions $\mathcal{I}_7^{\text{dem}}, \mathcal{I}_8^{\text{dem}}, \mathcal{I}_1^{\text{sup}}$ and Eq. (19) to assumption sets with parameters $\eta^1 = 12.5$ and $\eta^2 = 9$:

$$\phi_{\text{a}}^2 = \square(\mathcal{I}_1^{\text{sup}} \geq 9) \tag{30}$$

$$\phi_{\text{a}}^1 = \square\Big((\mathcal{I}_7^{\text{dem}} \leq 12.5 \ \wedge \ \mathcal{I}_8^{\text{dem}} \leq 0) \vee (\mathcal{I}_7^{\text{dem}} \leq 0 \ \wedge \ \mathcal{I}_8^{\text{dem}} \leq 12.5)\Big). \tag{31}$$

Under the assumptions $\phi_{\text{a}}^1, \phi_{\text{a}}^2$ the controller synthesis engine finds control strategies that enforce both $\phi^1 = \phi_{\text{spec}}^1 \wedge \phi_{\text{g}}^1$ and $\phi^2 = \phi_{\text{spec}}^2 \wedge \phi_{\text{g}}^2$. For brevity, we focus only on the runtime results of $\mathcal{N}^1$ as the results from $\mathcal{N}^2$ are similar. The state space partition has 27 440 symbolic states and the Rabin automaton has 9 states, so the product automaton has 246 960 states. The python programming language is used to simulate dynamics and construct the abstraction while the Rabin game solver in the MATLAB-based toolbox `conPAS2` is used as a synthesis engine. Runtime results are summarized in Table 1 for the identified contract parameters and were obtained with an early 2013 Macbook Pro with 8 GB of RAM and a 2.4 GHz Intel Core i7 processor. The runtime and memory results are summarized in the first two columns of Table 1. The sparse abstraction exhibits a reduction in the number of transitions by a factor of 121.2. The memory required to explicitly store the abstraction was reduced by a comparable factor of 135. The time to construct the product automaton and synthesize the controller is reduced by a factor of 2. Both solvers recover the same set of initial states for which a control strategy exists because the sparse and

Table 1
Synthesis runtimes and memory requirements.

	Box $\mathcal{N}^1$	Sparse $\mathcal{N}^1$	Sparse $\mathcal{N}^3$
# Abs. State $\lvert\mathcal{Q}\rvert$	27 440	27 440	1536 640
# Abs. Trans. $\lvert\Delta\rvert$	1.33×10^7	1.09×10^5	1.22×10^7
Abs. Mem. (MB)	118.76	.88	98.3
# Product Trans.	9.6×10^8	7.6×10^6	2.34×10^9
Abs. time (s)	79	45	6678
Product time (s)	153	65	9723
Synthesis (s)	110	63	14 665
Total time (s)	342	173	24 388

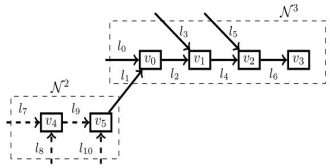

Fig. 10. Two sub-networks $\mathcal{N}^3, \mathcal{N}^2$. Solid arrows are links in $\mathcal{N}^3$ while dashed arrows are links in $\mathcal{N}^2$.

Larger Sub-network $\mathcal{N}^3$ parameters:

$(x_0^{\text{cap}}, \ldots, x_6^{\text{cap}}) = \quad (60, 40, 80, 24, 60, 24, 60)$

$(c_0, \ldots, c_6) = \quad (20, 15, 20, 12, 20, 12, 20)$

$\mathcal{D}^{\text{exog}} := \quad \{d : 0 \leq d \leq [10, 0, 0, 3, 0, 3, 0]\}$

$\beta_{02} = \beta_{24} = \beta_{46} = 0.8,\ \beta_{12} = \beta_{34} = \beta_{56} = 1.0,$

$u_0 \in \{\{l_0\}, \{l_0, l_1\}\},\ u_1 \in \{\{l_2\}, \{l_2, l_3\}\}$

$u_2 \in \{\{l_4\}, \{l_4, l_5\}\}$

If l_1 actuated, then $\alpha_{02} = 0.7, \alpha_{12} = 0.3$

If l_3 actuated, then $\alpha_{24} = 0.7, \alpha_{34} = 0.3$

If l_5 actuated, then $\alpha_{46} = 0.7, \alpha_{56} = 0.3$

Otherwise $\alpha_{02} = \alpha_{24} = \alpha_{46} = 1.0$

Fig. 11. Parameters for $\mathcal{N}^3$.

box abstractions are mutually upper alternatingly similar. System trajectories once the two networks are interconnected are shown in Fig. 9; the controllers are synthesized with the sparse abstraction and refined with Theorem 8.

7.5. Large network benchmark

We showcase the scalability of our approach by synthesizing a controller for a 7 dimensional system. Fig. 10 shows $\mathcal{N}^3$, which is almost identical to $\mathcal{N}^1$ but has two additional links and one additional intersection. Compared to $\mathcal{N}^1$, $\mathcal{N}^3$ has twice as many control modes and the discrete partition has 56 times more states. Moreover, the specification includes a clause about the additional link l_5, so $\phi^3_{\text{spec}} := \bigwedge_{i=1,3,5} \square \diamond (x_i \leq 20)$ and the Rabin automaton has 12 states instead of 9 previously. The parameters for $\mathcal{N}^3$ are mostly taken from the dynamics of $\mathcal{N}^1$, and can be found in Fig. 11. Otherwise, this problem is identical to the compositional synthesis problem between networks $\mathcal{N}^1$ and $\mathcal{N}^2$ and we use the same contract parameters. Simulated trajectories for the interconnected networks are presented in Fig. 12.

Table 1 presents synthesis benchmarks. We were unable to construct $\mathcal{N}^3$'s box abstraction due to memory constraints, yet $\mathcal{N}^3$'s sparse abstraction has a smaller memory footprint than $\mathcal{N}^1$'s

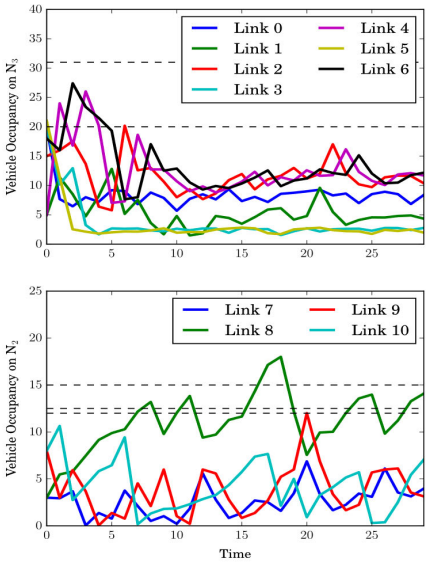

Fig. 12. Simulation of two networks with a larger network.

box abstraction. The total runtime grew by roughly two orders of magnitude, which is proportional to the increase in number of transitions in the product automaton. We expect a solver implemented in C/C++ to exhibit much better runtimes than the MATLAB-based conPAS2. conPAS2 is also written to accommodate a varying number of transitions from each discrete state. A more memory efficient data structure could exploit the constant number of transitions in a sparse abstraction.

8. Conclusion

This paper considered monotone systems and directed specifications, and captured the underlying structure of the controller synthesis problem by defining directed alternating simulations relations. These relations allowed us to develop sparse abstractions and directed contracts between monotone systems. Numerical results showcase a tangible speedup by systematically addressing algorithmic and memory bottlenecks in the abstraction and synthesis procedures. The use of directed alternating simulations relations for synthesis with directed specifications raises the intriguing possibility of crafting customized simulation relations for other classes of specifications and behavioral relations. Future work will explore time varying contracts.

References

Alur, R., Henzinger, T. A., Kupferman, O., & Vardi, M. Y. (1998). Alternating refinement relations. In *Proceedings of the 8th international conference on concurrence theory*.

Angeli, D., & Sontag, E. (2003). Monotone control systems. *IEEE Transactions of Automatic Control*, *48*(10), 1684–1698.

Baier, C., & Katoen, J.-P. (2008). *Principles of model checking*. (pp. I–XVII, 1–975). MIT Press.

Coogan, S., & Arcak, M. (2015). Efficient finite abstraction of mixed monotone systems. In *Proceedings of the 18th ACM international conference on hybrid systems: Computation and control*.

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research*, *28*, 269–287.

Dallal, E., & Tabuada, P. (2015). On compositional symbolic controller synthesis inspired by small-gain theorems. In *2015 54th IEEE conference on decision and control* (pp. 6133–6138).

Hirsch, M. W. (1985). Systems of differential equations that are competitive or cooperative II: Convergence almost everywhere. *SIAM Journal on Mathematical Analysis*, *16*, 423–439.

Kim, E. S., Arcak, M., & Seshia, S. A. (2015). Compositional controller synthesis for vehicular traffic networks. In *54th IEEE conference on decision and control*.

Kim, E. S., Arcak, M., & Seshia, S. A. (2016). Directed specifiations and assumption mining for monotone dynamical systems. In *19th conference on hybrid systems: Computation and control*.

Lomuscio, A., Strulo, B., Walker, N., & Wu, P. (2010). Assume-guarantee reasoning with local specifications. In *International conference on formal engineering methods* (pp. 204–219). Springer.

Mazo Jr., M., Davitian, A., & Tabuada, P. (2010). Pessoa: A tool for embedded controller synthesis. In *International conference on computer aided verification* (pp. 566–569). Springer.

Meyer, P.-J., Girard, A., & Witrant, E. (2015). Safety control with performance guarantees of cooperative systems using compositional abstractions. *IFAC-PapersOnLine*, *48*(27), 317–322.

Moor, T., & Raisch, J. (2002). Abstraction based supervisory controller synthesis for high order monotone continuous systems. In *Modelling, analysis, and design of hybrid systems* (pp. 247–265). Springer.

Mouelhi, S., Girard, A., & Gössler, G. (2013). CoSyMA: a tool for controller synthesis using multi-scale abstractions. In *Proceedings of the 16th international conference on hybrid systems: Computation and control* (pp. 83–88). ACM.

Nilsson, P., & Ozay, N. (2016). Synthesis of separable controlled invariant sets for modular local control design. In *American control conference, 2016* (pp. 5656–5663). IEEE.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th annual symposium on foundations of computer science* (pp. 46–57).

Rungger, M., & Zamani, M. (2015). Compositional construction of approximate abstractions. In *Proceedings of the 18th international conference on hybrid systems: Computation and control* (pp. 68–77). New York, NY, USA: ACM.

Rungger, M., & Zamani, M. (2016). SCOTS: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th international conference on hybrid systems: Computation and control* (pp. 99–104). ACM.

Sadraddini, S., & Belta, C. (2016). Safety control of monotone systems with bounded uncertainties. In *Decision and control, 2016 IEEE 55th conference on* (pp. 4874–4879). IEEE.

Tabuada, P. (2009). *Verification and control of hybrid systems: A symbolic approach*. Springer.

Yordanov, B., Tumov, J.,Čern, I., Barnat, J., & Belta, C. (2012). Temporal logic control of discrete-time piecewise affine systems. *IEEE Transactions on Automatic Control*, *57*(6), 1491–1504.

Eric S. Kim received a B.S. degree in Electrical Engineering and a B.S. degree in Physics from the University of Maryland, College Park in 2013. He is currently a graduate student researcher in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley. He is a recipient of the NSF Graduate Research Fellowship. His research interests include robustness, coordination, consensus problems for multi-agent and networked systems with an emphasis on applications to vehicular traffic flow networks.

Murat Arcak is a professor at U.C. Berkeley in the Electrical Engineering and Computer Sciences Department. He received the B.S. degree in Electrical Engineering from the Bogazici University, Istanbul, Turkey (1996) and the M.S. and Ph.D. degrees from the University of California, Santa Barbara (1997 and 2000). His research is in dynamical systems and control theory with applications to synthetic biology, multi-agent systems, and transportation. Prior to joining Berkeley in 2008, he was a faculty member at the Rensselaer Polytechnic Institute. He received a CAREER Award from the National Science Foundation in 2003, the Donald P. Eckman Award from the American Automatic Control Council in 2006, the Control and Systems Theory Prize from the Society for Industrial and Applied Mathematics (SIAM) in 2007, and the Antonio Ruberti Young Researcher Prize from the IEEE Control Systems Society in 2014. He is a member of SIAM and a fellow of IEEE.

Sanjit A. Seshia is a Professor in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley. He received an M.S. and Ph.D. in Computer Science from Carnegie Mellon University, and a B.Tech. in Computer Science and Engineering from the Indian Institute of Technology, Bombay. His research interests are in dependable computing and computational logic, with a current focus on applying automated formal methods to problems in cyber-physical systems, computer security, electronic design automation, and synthetic biology. His Ph.D. thesis work on the UCLID verifier and decision procedure helped pioneer the area of satisfiability modulo theories (SMT) and SMT-based verification. He is co-author of a widely-used textbook on embedded, cyber-physical systems and has led the development of technologies for cyber-physical systems education based on formal methods. His awards and honors include a Presidential Early Career Award for Scientists and Engineers (PECASE), an Alfred P. Sloan Research Fellowship, the Frederick Emmons Terman Award for contributions to electrical engineering and computer science education, and the School of Computer Science Distinguished Dissertation Award at Carnegie Mellon University.