Formal Methods for Control of Traffic Flow

Automated control synthesis from finite-state transition models

Samuel Coogan, Murat Arcak, and Calin Belta
POC: Sam Coogan (scoogan@ucla.edu)
September 13, 2016

Today's increasingly populous cities require intelligent transportation systems that make efficient use of existing transportation infrastructure. However, inefficient traffic management is pervasive [1], [2], costing US$160 billion in the United States in 2015, including 6.9 billion hours of additional travel time and 3.1 billion gallons of wasted fuel [3]. To mitigate these costs, the next generation of transportation systems will include connected vehicles, connected infrastructure, and increased automation. In addition, these advances must coexist with legacy technology into the foreseeable future. This complexity makes the goal of improved mobility and safety even more daunting.

To address this complexity, scalable and automated verification and synthesis techniques for transportation systems are required. Methods from formal verification and synthesis of control systems are highly promising for providing automated tools that guarantee safety and improve mobility. Formal methods were originally developed for specifying and verifying the correct behavior of software and hardware systems, as well as for synthesis of such systems. An important research task now is to ensure these approaches are scalable, adaptable, and reliable for transportation systems.

The first aim of this article is to review a broad technique from formal methods for synthesizing correct-by-design controllers for dynamical systems by first obtaining a finite-state abstraction and then applying a game-based algorithm for synthesizing a control strategy to satisfy a linear temporal logic specification. The primary goal is to give an accessible introduction to formal methods for control systems. The second objective is to review vehicular traffic-flow models and to characterize a general model amenable to formal control synthesis. The dynamics of traffic-flow networks are shown to exhibit considerable structure that enables efficient finite-state abstraction. In doing so, the importance of identifying and exploiting inherent structural properties is emphasized. The contents of this tutorial article are based on results presented in [4], [5], [6].

Formal Methods For Control Synthesis

Control techniques often focus on limited objectives of system behavior such as stabilizing a system around an equilibrium point or ensuring that the system does not enter an unsafe operating condition. In contrast, formal methods have been developed in the field of computer science to verify that software and hardware systems satisfy rich objectives expressed in temporal logic. Examples of properties easily expressed in temporal logic include fairness (whenever some condition occurs, another condition is guaranteed to eventually occur), repeated reachability (a certain condition occurs infinitely often), and sequentiality (a condition only occurs after another condition). For example, a safety requirement may be relaxed to a requirement that if the system enters an unsafe condition, it eventually exits the unsafe condition.

Researchers from the control theory and computer science communities are increasingly interested in combining control-theoretic tools for complex physical systems with formal methods for accommodating complex specifications. A major difference is that, typically, formal methods for software and hardware verification rely on finite-state models whereas control-theoretic approaches to system analysis usually consider continuous state spaces. A full review of the rapidly growing literature in this area is beyond the scope of this article, but some work that is particularly relevant is highlighted. Certain classes of systems allow finite *bisimulations* such that the dynamics are exactly represented by a finite-state transition model [7], [8], or are amenable to a related notation of approximate bisimilarity [9], [10], [11], [12], [13]. When a finite bisimulation is not possible, methods exist to approximate the behavior of the underlying system with finite abstractions; [14], [15], [16], [17], [18], [19], [20] are particularly relevant to the ideas presented in this article. Applications and special cases include robotic path planning [21], [22], [23], [24], switched continuous-time systems [25], piecewise-linear systems [26], [27], model predictive control formulations [28], [29], and control of Markov decision processes [30], [31], [32], [33].

In this article, the focus is on a particular approach for applying formals methods to control systems and defining finite-state abstractions for discrete-time dynamical systems that overapproximate the behavior of the underlying dynamics. The overapproximation is such that, by considering the possible evolution of the finite abstraction for given inputs, properties of the behavior of the original dynamical system are guaranteed subject to these inputs. Using the finite-state abstraction, an automatic controller-synthesis procedure is proposed to guarantee that the abstraction and the underlying system satisfy an objective given in temporal logic. The synthesis algorithm, illustrated schematically in Figure 1, relies on automata theory and fixed-point algorithms to compute a finite-memory control strategy.

Traffic-Flow Networks

Next, the formal control synthesis approach presented in this article is specialized to traffic-flow networks. First, a model that consists of links interconnected at junctions is defined. Vehicles flow from link to link through the junctions depending on physically and phenomenologically motivated flow properties. The state of the network at a given time is the number of vehicles occupying each link. This model captures the salient features of both networks of signalized intersections and freeway traffic networks.

The theory of monotone dynamical systems is then used for the finite abstraction of traffic-flow networks. Trajectories of monotone systems maintain a partial order on the state of the system, which restricts the possible transient and long-term behavior of such systems [34], [35], [36]. Congestion propagation precludes the dynamics of traffic-flow networks from being monotone; however, a generalization called *mixed monotonicity* [5], [37] is shown to be applicable. Mixed monotone systems are those that can be decomposed into increasing and decreasing components. Pairs of links in traffic-flow networks naturally exhibit mixed monotone dependencies whereby an increase in the state of one link causes an increase or a decrease in the flow to another link. Whether the flow increases or decreases depends on the topological relationship of the links under consideration.

Mixed monotonicity allows efficient computation of finite-state abstractions because one-step reachable sets may be approximated efficiently by evaluating a decomposition function at two extreme points. This approach is particularly attractive since the computational cost is limited to two function evaluations and does not increase with the dimension of the state space.

The article concludes by applying the techniques and results of the prior sections to a case study for which a control strategy is efficiently computed using the mixed monotone property of the traffic-flow dynamics. An alternative abstraction procedure that assumes piecewise affine dynamics is discussed.

A Dynamic Model for Traffic-Flow Networks

The traffic-flow model considered in this article takes a *macroscopic* view by considering aggregate conditions of the network such as occupancy of vehicles on each road segment and traffic-flow rate rather than considering movement of individual vehicles. This model was proposed in [4] and [38] and encompasses the cell-transmission model of freeway traffic flow [39], [40] and queue-forwarding models as in [41], which further account for the finite capacity of queues.

General Model

A traffic network consists of a set of *links* $\mathcal{L}$ interconnected at a set of *nodes* $\mathcal{V}$, as in Figure 2. In freeway networks, the nodes represent junctions where, for example, onramps enter, offramps exit, two freeways merge, a freeway diverges to two freeways, or a node serves to divide a longer link into two smaller links. In signalized networks, the nodes are signalized intersections. Let $\sigma : \mathcal{L} \to \mathcal{V}$ map each link to the node immediately downstream (the *head*) of link ℓ, and let $\tau : \mathcal{L} \to \mathcal{V} \cup \epsilon$ map each link to the node immediately upstream (the *tail*) of link ℓ; the symbol ϵ denotes that no upstream node is modeled in the network, thus links for which $\tau(\ell) = \epsilon$ direct exogenous flow onto the network.

Although this article presents a discrete-time model, the results easily extend to continuous time; see [6], [42]. The state of link $\ell \in \mathcal{L}$ at discrete time t is denoted by

$$x_\ell[t] \in [0, x_\ell^{\text{cap}}], \quad \text{for all } \ell \in \mathcal{L},$$

and represents the number of vehicles occupying link ℓ, where $x_\ell^{\text{cap}} \in \mathbb{R}_{\geq 0}$ is the maximum number of vehicles accommodated by link ℓ. For freeway networks, x_ℓ^{cap} is called the *jam density*. Note by adopting a fluid-like model of traffic flow $x_\ell[t]$ is not restricted to integer values. The domain is

$$\mathcal{X} \triangleq \prod_{\ell \in \mathcal{L}} [0, x_\ell^{\text{cap}}].$$

The main premise of the cell-transmission model and queue-forwarding models is that traffic flow from one link to another downstream link through a junction is restricted by the *demand* of vehicles to flow along the link as well as the *supply* of road capacity downstream. To this end, each link $\ell \in \mathcal{L}$ possesses an increasing *demand* function $D_\ell(\cdot)$ and a decreasing *supply* function $S_\ell(\cdot)$. The following are assumed for each $\ell \in \mathcal{L}$:

- The demand function $D_\ell : [0, x_\ell^{\text{cap}}] \to \mathbb{R}_{\geq 0}$ is strictly increasing and Lipschitz continuous with $D_\ell(0) = 0$.
- The supply function $S_\ell : [0, x_\ell^{\text{cap}}] \to \mathbb{R}_{\geq 0}$ is strictly decreasing and Lipschitz continuous with $S_\ell(x_\ell^{\text{cap}}) = 0$.

Prototypical demand and supply functions are shown in Figure 3. The demand function models the number of vehicles on a link that would flow through a junction in one time step if unimpeded by downstream congestion, while the supply function models the available capacity on a link to accept incoming flow. Thus, outgoing flow of a link does not exceed demand, and incoming flow does not exceed supply.

Junctions may be signalized so that the movement of vehicles through a junction v from an incoming link ℓ is allowed only if the link is *actuated* by the signal. Let

$$\mathcal{U} \subset 2^{\mathcal{L}}$$

be a collection of sets of links that may be simultaneously actuated where $2^{\mathcal{L}}$ denotes the set of all subsets of $\mathcal{L}$. An element $u \in \mathcal{U}$ is an *actuation*. Since signalized intersections are typically operated independently, the set $\mathcal{U}$ is often the Cartesian product of collections of subsets of incoming links for each intersection.

An important element of modeling transportation networks is to characterize the routing properties for junctions with multiple incoming and/or outgoing links that captures phenomenological properties of traffic flow. In particular, the routing policy must appropriately distribute the demand of links incoming to a junction among the outgoing links, and symmetrically, distribute supply of outgoing links among incoming links. For the former requirement, the *turn ratio* $\beta_{\ell k} \geq 0$ is introduced for each $\ell, k \in \mathcal{L}$ denoting the fraction of link ℓ's outgoing flow that routes to link k for links ℓ and k connected at a junction. Conservation of mass implies

$$\sum_{k \in \mathcal{L}} \beta_{\ell k} \leq 1, \qquad \text{for all } \ell \in \mathcal{L}, \tag{1}$$

where $\beta_{\ell k} \neq 0$ only if $\sigma(\ell) = \tau(k)$ and strict inequality in (1) implies that a nonzero fraction of the outgoing flow from link ℓ exits the network along, for example, unmodeled roads or driveways.

Symmetrically, the *supply ratio* $\alpha_{\ell k}$ is introduced for each $\ell, k \in \mathcal{L}$ denoting the fraction of link k's supply available to link ℓ. For all $u \in \mathcal{U}$,

$$\sum_{\{\ell \in u | \sigma(\ell) = \tau(k)\}} \alpha_{\ell k} = 1, \quad \text{for all } k \in \mathcal{L},$$

that is, the total supply of link k is divided among upstream, actuated links for each possible actuation $u \in \mathcal{U}$. For freeway networks, rather than assuming discrete actuation values so that a link is either actuated or not, it may be more appropriate to consider a controlled *metering* rate for links that represent onramps to the network. In this case, the controlled metering rate serves to threshold a link's demand at some upper limit; see [43] for a formalization of such an extension.

The outflow of vehicles from a link is defined as a function of the state $x \in \mathcal{X}$ and a chosen actuation $u \in \mathcal{U}$. The outflow of link ℓ is

$$f_\ell^{\text{out}}(x, u) = \begin{cases} \min\left\{ D_\ell(x_\ell), \min\limits_{k \text{ s.t. } \beta_{\ell k} \neq 0} \dfrac{\alpha_{\ell k}}{\beta_{\ell k}} S_k(x_k) \right\} & \text{if } \ell \in u, \\ 0 & \text{else,} \end{cases} \tag{2}$$

that is, the flow exiting link ℓ is as close to the demand of link ℓ as allowed by downstream supply. Conservation of mass completes the model so that

$$x_\ell[t+1] = F_\ell(x[t], u[t], d[t]) \triangleq x_\ell[t] - f_\ell^{\text{out}}(x[t], u[t]) + \sum_{k \in \mathcal{L}} \beta_{k\ell} f_k^{\text{out}}(x[t], u[t]) + d_\ell[t], \quad (3)$$

where $d_\ell[t]$ is an exogenous flow entering link ℓ. It is assumed that the exogenous flow is truncated, and therefore $d_\ell[t]$ is such that $x_\ell[t+1] \leq x_\ell^{\text{cap}}$ always. In general, it is further assumed that $d[t] \in \mathcal{D}$ for disturbance set $\mathcal{D} \subseteq (\mathbb{R}_{\geq 0})^{\mathcal{L}}$ for all time. Note that (2) minimizes over all downstream links so that lack of downstream supply on one link reduces flow to other links. This phenomenon of downstream traffic blocking flow to other downstream links at a diverging junction is referred to as the *first-in-first-out (FIFO)* property, [40], [44], and it is a feature of traffic flow that has been observed even on wide freeways with many lanes [45], [46].

Example 1. Consider the network shown in Figure 4(a) with $\mathcal{L} = \{1, 2, 3\}$ and $\beta_{12} = \beta_{13} = 0.5$, $\alpha_{12} = \alpha_{13} = 1$. Assume the intersection is not signalized so that the input set is $\mathcal{U} = \{u^{\text{all}}\}$, $u^{\text{all}} \triangleq \{1, 2, 3\}$, indicating flow along all links is allowed. Then

$$f_1^{\text{out}}(x, u^{\text{all}}) = \min\left\{ D_1(x_1), \frac{1}{0.5} S_2(x_2), \frac{1}{0.5} S_3(x_3) \right\},$$
$$f_\ell^{\text{out}}(x, u^{\text{all}}) = D_\ell(x_\ell), \quad \ell \in \{2, 3\}.$$

Special Case: Piecewise-Affine Model

Of particular importance is the case when the demand and supply functions are assumed to be piecewise linear. In particular,

$$D_\ell(x_\ell) = \min\{v_\ell x_\ell, q_\ell^{\text{max}}\}, \quad (4)$$

$$S_\ell(x_\ell) = w_\ell(x_\ell^{\text{cap}} - x_\ell), \quad (5)$$

for constants $v_\ell > 0$, $w_\ell > 0$, and $q_\ell^{\text{max}} > 0$. For freeway networks, v_ℓ and w_ℓ are the *free-flow speed* and *congested wave speed* [47]. For signalized networks, x_ℓ is interpreted as the queue length and $v_\ell = w_\ell = 1$. Then q_ℓ^{max} is the *saturation flow rate* [48], $D_\ell(x_\ell)$ is the minimum of the queue length x_ℓ and the saturation flow rate, and $S_\ell(x_\ell)$ is the unoccupied queue capacity of link ℓ.

When the demand and supply functions have the form (4) and (5), respectively, the dynamics are *piecewise affine*, that is, there exists a set of polytopes $\mathcal{P} = \{\mathcal{X}_q\}_{q \in \mathcal{Q}}$ for some index set $\mathcal{Q}$ such that $\cup_{q \in \mathcal{Q}} \mathcal{X}_q = \mathcal{X}$ and $\mathcal{X}_q \cap \mathcal{X}_{q'} = \emptyset$, for all $q, q' \in \mathcal{Q}$, and such that, for each $q \in \mathcal{Q}$,

$$F(x, u, d) = A_{q,u} x + b_{q,u} + d, \quad \text{for all } x \in \mathcal{X}_q \quad (6)$$

for some $A_{q,u} \in \mathbb{R}^{\mathcal{L}\times\mathcal{L}}$, $b_{q,u} \in \mathbb{R}^{\mathcal{L}}$. In other words, the traffic dynamics are affine within each polyhedral partition. The polytopes arise from the $\min\{\cdot\}$ functions in (2) and (4). Below, a finite-state abstraction is constructed using the tools in [26], [49] that exploit the piecewise affine nature of the dynamics.

Overapproximating Finite Abstractions

Next, a methodology for computing a finite-state abstraction that overapproximates, in a particular sense, the dynamics of a discrete-time dynamical system is described. The motivation for such an abstraction is two fold. First, for many physical systems, satisfactory performance is often defined in terms of a finite set of properties such as "no road segment becomes congested" for traffic networks, "temperature remains below a given threshold" for a chemical process, or "the power network can withstand one generator failure" for power networks. That is, performance is not based on a precise, continuous measurement of the state. Second, a finite-state abstraction is amenable to formal synthesis methods as described in the next section.

Consider a discrete-time dynamical system of the form

$$x[t+1] = F(x[t], u[t], d[t]), \tag{7}$$

for $u[t] \in \mathcal{U}$ with $\mathcal{U}$ a finite set, $x[t] \in \mathcal{X} \subseteq \mathbb{R}^n$, for all t, and $d[t] \in \mathcal{D} \subseteq \mathbb{R}^m$. Note that the traffic-network model proposed above satisfies these stipulations; however, the ideas presented here apply generally. This system is called the *real* system, in contrast to the finite-state *abstraction* developed subsequently.

Consider a partition of $\mathcal{X}$ with index set $\mathcal{Q}$, that is, the collection of nonempty sets $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ satisfies $\mathcal{X} = \cup_{q\in\mathcal{Q}}\mathcal{X}_q$ and $\mathcal{X}_q \cap \mathcal{X}_{q'} = \emptyset$, for all $q, q' \in \mathcal{Q}$. Let

$$\pi_{\mathcal{P}} : \mathcal{X} \to \mathcal{Q},$$
$$\pi_{\mathcal{P}}(x) = q, \text{ when } x \in \mathcal{X}_q,$$

be the projection map from $\mathcal{X}$ to $\mathcal{Q}$. For a trajectory $x[\cdot]$ of the dynamical system (7), let $\pi_{\mathcal{P}}(x[\cdot])$ denote the sequence $q[0]q[1]q[2]\cdots \in \mathcal{Q}^{\mathbb{Z}_{\geq 0}}$, where $\mathbb{Z}_{\geq 0}$ denotes the nonnegative integers. Note that, for some set W, $W^{\mathbb{Z}_{\geq 0}}$ denotes the set of infinite sequences of elements from W. In this article, this notation is exclusively used to represent a time sequence. Therefore, $w \in W^{\mathbb{Z}_{\geq 0}}$ is indexed with brackets and $w = w[0]w[1]w[2]\cdots$, where $w[t] \in W$, for all t. To emphasize the time dependence, the notation $w = w[\cdot]$ is also used.

Definition 1 (finite-state abstraction)**.** Given a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ of $\mathcal{X}$ for (7), $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$

with $\delta : \mathcal{Q} \times \mathcal{U} \to 2^{\mathcal{Q}}$ is a *finite-state abstraction* of (7) if

$$\text{for all } x \in \mathcal{X} \text{ and } d \in \mathcal{D}, \qquad x \in \mathcal{X}_q \text{ and } F(x,u,d) \in \mathcal{X}_{q'} \quad \text{implies} \quad q' \in \delta(q,u), \tag{8}$$

for any $q, q' \in \mathcal{Q}$, $u \in \mathcal{U}$. δ is called the *transition map*. An *execution* of the finite-state abstraction is a pair of sequences $q[\cdot]$, $u[\cdot]$ with each $q[t] \in \mathcal{Q}$ and each $u[t] \in \mathcal{U}$ for $t \geq 0$ for which $q[t+1] \in \delta(q[t], u[t])$, for all $t \geq 0$.

Figure 5 illustrates how a finite-state abstraction is obtained for a dynamical system. A finite-state abstraction of (7) captures the underlying dynamics at a level of granularity dependent on the partition $\mathcal{P}$. A finite-state abstraction is thus a transition system with a finite set of states $\mathcal{Q}$ and finite input set $\mathcal{U}$ inherited from the real system. Each input $u \in \mathcal{U}$ enables a set of transitions as determined by $\delta(q,u)$. From this set of transitions, the transition that is executed by the system is not controlled, and thus the abstraction is *nondeterministic*. The notion of *state* will be used to refer both to an element $q \in \mathcal{Q}$ in the finite-state abstraction and an element $x \in \mathcal{X}$ of the real system when it is clear that no confusion will arise.

Note the direction of implication in (8) allows the situation where $q' \in \delta(q,u)$ yet $F(x,u,d) \notin \mathcal{X}_{q'}$ for any $x \in \mathcal{X}_q$, $d \in \mathcal{D}$. When this holds, there exists a *spurious one-step transition* from q to q' under input u. Thus the transition function δ *overapproximates* the underlying dynamics and there may exist executions of the transition system that do not correspond with any trajectory of the original dynamical system. See "Spurious Transitions in Finite Abstractions" for details about how such spurious trajectories arise and techniques for mitigating their effects.

Even in the presence of spurious transitions, overapproximating finite-state abstractions are sufficient for control synthesis for linear temporal logic specifications, as discussed below. That is, a controller obtained algorithmically from the finite-state abstraction is applicable to the original real system with the same guarantees of performance. The drawback of excessive spurious transitions is that the synthesis algorithm may fail to find a feasible solution. In this article, spurious transitions are judiciously allowed for in order to obtain computational savings in the abstraction step without undermining the synthesis algorithm.

Specifying System Behavior

This article focuses on specifications for system behavior given in *linear temporal logic (LTL)*, an extension of propositional logic that allows for temporal modalities. The expressive power of LTL captures many objectives relevant for control of transportation networks, such as "link 1 eventually enters an uncongested state and remains in this condition, for all future

time." LTL formulae comprise a finite set of *observations*, denoted by $\mathcal{O}$, the standard Boolean connectives, and temporal modalities such as $\Box$ ("always") and $\diamondsuit$ ("eventually"), and a LTL formula is usually denoted by φ. See "Linear Temporal Logic Specifications of System Behavior" for an overview of the syntax and semantics of LTL and its use for specifying behaviors of finite systems. *Specification* and *objective* are used interchangeably to refer to the desired behavior of a finite system.

The synthesis approach for LTL specifications presented here relies on a finite-state abstraction of the underlying continuous system that overapproximates the system's dynamics as described in the previous section. The synthesis algorithm is then posed as a two-player game between a *controller* and the *environment* where the controller seeks control actions to ensure satisfaction of the behavior specification and the environment seeks to prevent satisfaction of the specification. First, a notion of LTL satisfaction for discrete-time dynamical systems is defined.

The dynamical system (7) with partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ is *labeled* if there exists a set of observations $\mathcal{O}$ and a labeling function $H : \mathcal{X} \to 2^{\mathcal{O}}$ that satisfies $x, y \in \mathcal{X}_q \implies H(x) = H(y)$, that is, elements in the same partition are labeled with the same observations. The corresponding finite-state abstraction $\mathcal{T}$ is then said to be *labeled* and the same notation is used to denote the well-defined labeling function $L : \mathcal{Q} \to 2^{\mathcal{O}}$ such that $H(q) = H(x)$, for all $x \in \mathcal{X}_q$.

For a trajectory $x[\cdot]$ of a labeled dynamical system (7), the sequence $H(x[0])H(x[1])H(x[2])\cdots \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$ is the *trace* of the trajectory. This sequence is abbreviated $H(x[\cdot])$. Similarly, for an execution $q[\cdot], u[\cdot]$ of a finite-state abstraction, the *trace* of the execution is the sequence $H(q[0])H(q[1])H(q[2])\cdots$, abbreviated as $H(q[\cdot])$.

Consider a labeled finite-state abstraction $\mathcal{T}$ of a labeled dynamical system (7) with partition $\mathcal{P}$. Equation (8) guarantees that, for a trajectory $x[\cdot]$ of the dynamical system (7) generated by the input sequence $u[\cdot]$, the pair $\pi_{\mathcal{P}}(x[\cdot])$, $u[\cdot]$ is an execution of $\mathcal{T}$.

LTL satisfaction for trajectories of dynamical systems and executions of finite-state abstractions is defined in the natural way: $x[\cdot]$ satisfies φ if its trace $H(x[\cdot])$ satisfies φ, and likewise for $q[\cdot]$ and $H(q[\cdot])$.

Formal Controller Synthesis From Abstractions

Consider the labeled dynamical system $x[t+1] = F(x[t], u[t], d[t])$ as in (7) with observations $\mathcal{O}$ and a partition $\mathcal{P}$ of the domain $\mathcal{X}$, along with a LTL objective φ. Informally, the objective is to find a feedback control strategy such that the resulting closed-loop trajectories satisfy φ. To make this formal, the objective is instead defined in terms of a labeled finite-state

abstraction $\mathcal{T}(\mathcal{Q},\mathcal{U},\delta)$ for the dynamical system; it will be shown below that synthesizing a controller from the finite-state abstraction is sufficient for obtaining a feedback controller for the real system.

let W^+ denote the set of nonempty, finite-length sequences of elements from W, that is, $w \in W^+$ takes the form $w = w[0]w[1]\cdots w[n]$, where $w[t] \in W$ for $t = 0,\ldots,n$ for some $n \geq 0$.

A feedback *control strategy* γ for a labeled finite-state abstraction $\mathcal{T}$ is a map

$$\gamma : (2^{\mathcal{O}})^+ \to \mathcal{U} \tag{9}$$

that prescribes a control input for each finite history $q[0]q[1]\cdots q[n]$.

Control synthesis objective. The objective is to find a control strategy γ of the form (9) and a set of initial conditions $\mathcal{Q}_0 \subseteq \mathcal{Q}$ for the finite-state abstraction $\mathcal{T} = (\mathcal{Q},\mathcal{U},\delta)$ such that φ holds for any execution $q[\cdot]$ satisfying $q[0] \in \mathcal{Q}_0$ and, for all $t \geq 0$, $q[t+1] \in \delta(q[t],u[t])$ with $u[t] = \gamma(q[0]q[1]\cdots q[t])$. ■

The initial condition is not considered to be fixed *a priori* because the employed synthesis algorithm identifies all acceptable initial conditions for the finite-state abstraction. If it is known that the abstraction will initiate in some subset of states, then the set of acceptable states as determined by the synthesis algorithm is compared to the specified set of initial conditions.

Definition 2 (Finite-memory control strategy)**.** The control strategy γ is said to be *finite-memory* if there exist:

- M, a finite set of *modes*,
- $m_0 \in M$, an initial mode,
- $\Delta : M \times \mathcal{Q} \to M$, a mode transition map,
- $g : M \times \mathcal{Q} \to \mathcal{U}$, a control selection map,

defining a transition system that describes the behavior of $u[t] = \gamma(q[0]\cdots q[t])$. Specifically, a finite-memory controller (abbreviated *controller*) is initialized so that $m[0] = m_0 \in M$ is the initial state of the controller. Then, inductively, $u[t] = g(m[t],q[t])$ and $m[t+1] = \Delta(m[t],q[t])$ for $t \geq 0$ where $q[t+1]$ is obtained via the abstraction $\mathcal{T} = (\mathcal{Q},\mathcal{U},\delta)$.

For a finite-memory control strategy, g selects an action based on the current state of the finite-state abstraction and mode of the controller, and Δ updates the finite mode (that is, memory) of the controller. Thus, $\gamma(q[0]q[1]\cdots q[t]) = g(m[t],q[t])$, where $m[t]$ is computed as

described in the above definition. As will be seen below, finite-memory controllers suffice for control synthesis from LTL specifications.

To synthesize a finite-memory controller for a LTL specification, consider a finite-state automaton that tracks progress towards the LTL specification using a finite set of modes. The automaton's transitions are labeled with the observations $\mathcal{O}$ so that an infinite trace of observations generates an infinite execution of the automaton. This article considers a particular class of automata, called *Rabin automata*, that *accept* infinite traces of observations if a certain set of modes are visited infinitely often and another set of modes are visited only finitely often.

Rabin automata serve two key purposes. First, there exist automated methods and off-the-shelf software for converting any LTL objective to a Rabin automaton that accepts all (and only) those traces that satisfy the LTL objective. Second, there exist algorithms for obtaining a control strategy for a finite-state abstraction from the Rabin automaton generated by the desired LTL specification. Moreover, the obtained control strategy is finite-memory, and the structure of the finite-memory controller is inherited from the structure of the Rabin automaton. For details on how a control strategy is synthesized from a Rabin automaton by playing a *Rabin game*, see "Controller Synthesis for LTL Specifications from Rabin Games."

A finite-memory controller of the form given in Definition 2 is applied to the original real system in the natural way. Specifically, let $u[t] = g(m[t], \pi_P(x[t]))$ at each time step t, and update the controller mode as $m[t+1] = \Delta(m[t], \pi_P(x[t]))$. The following Proposition implies that the overapproximating finite-state abstraction is sufficient for formal control synthesis.

Proposition 1. *Given the finite-memory control strategy γ and a set of initial states $\mathcal{Q}_0 \subseteq \mathcal{Q}$, if $q[\cdot]$ satisfies φ, for all executions of $\mathcal{T}$ for which $q[0] \in \mathcal{Q}_0$ and $u[t] = \gamma(q[0]q[1]\cdots q[t])$, then $x[\cdot]$ satisfies φ, for all trajectories $x[\cdot]$ of the real system for which $x[0] \in \cup_{q\in\mathcal{Q}_0}\mathcal{X}_q$ and $u[t] = \gamma(q[0]q[1]\cdots q[t])$ where $q[t] = \pi_P(x[t])$, for all t.*

The proof follows readily from the overapproximating nature of $\mathcal{T}$ as specified in (8). In particular, consider any trajectory $x[\cdot]$ of the real system induced by the input sequence $u[\cdot]$ for which $x[0] \in \cup_{q\in\mathcal{Q}_0}\mathcal{X}_q$ and $u[t] = \gamma(q[0]q[1]\cdots q[t]) = g(m[t], q[t])$, for all $t \geq 0$. The projected sequence $\pi_P(x[\cdot]) = q[\cdot] = q[0]q[1]q[2]\cdots$ is such that $q[0] \in \mathcal{Q}_0$ and $q[t+1] \in \delta(q[t], u[t])$, that is, $q[\cdot]$, $u[\cdot]$ is an execution of $\mathcal{T}$. By assumption, $q[\cdot]$ satisfies φ so that also $x[\cdot]$ also satisfies φ.

The importance of Proposition 1 is that a controller for the real system is obtainable by first constructing a finite-state abstraction and then computing a controller for the abstraction based on the automated synthesis approach of a Rabin game.

The resulting controller is *symbolic*, meaning that it only requires knowledge of $\pi_P(x[t])$, the currently occupied partition of the system. At each step t, the controller, which has internal state $m[t]$, receives the coarse measurement $q[t] = \pi_P(x[t])$ and applies the input $g(m[t], q[t])$. The controller's internal state is then updated by the finite mapping $\Delta(m[t], q[t])$.

Furthermore, the online memory and processing requirements are modest because the controller essentially consists of two lookup tables, each of dimension $|M| \times |\mathcal{Q}|$, corresponding to g and Δ. Even for very large $\mathcal{Q}$, the online computation time is low. The tradeoff is that the offline computation of g and Δ can be costly. Figure 6 illustrates how a finite-memory controller obtained from a Rabin automaton provides feedback control of a dynamical system from its finite-state abstraction.

For more general abstraction techniques that accommodate, for example, continuous input sets, a controller obtained from the abstraction may not be applicable to the original real system. Furthermore, if the relationship between the abstraction and the real system is not taken advantage of fully, a controller may be obtained that is not symbolic and, moreover, requires significantly more online computational resources. These intricacies have been the focus of recent research [50], [51].

Finally, Proposition 1 is based on the abstraction $\mathcal{T}$, which overapproximates the behavior of the real system. While this overapproximation is sufficient for ensuring correctness when a controller for the abstraction exists, attempts to synthesize a controller from the abstraction may fail due to spurious trajectories that are nonexistent in the real system. This conservatism is unavoidable for all but a few limited classes of dynamical systems that are amenable to *finite bisimulation* [13].

Finite Abstractions of Traffic-Flow Networks from Mixed Monotonicity

Thus far, it has been assumed that a finite-state abstraction is available so that a control strategy is synthesized for the real system from the overapproximating abstraction. This section addresses the difficulties of computing the finite-state abstraction.

Mixed Monotone Dynamical Systems

Again consider the dynamical system $x[t+1] = F(x[t], u[t], d[t])$ as in (7) and a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q \in \mathcal{Q}}$ that will be used to construct a finite-state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$. If it is possible to calculate an overapproximation of the one-step reachable set from $\mathcal{X}_q$ under input u,

$$R_{q,u} \supseteq \{F(x, u, d) \mid x \in \mathcal{X}_q, d \in \mathcal{D}\}, \tag{10}$$

then it is possible to construct a transition map δ satisfying (8) from

$$q' \in \delta(q,u) \text{ if and only if } \mathcal{X}_{q'} \cap R_{q,u} \neq \emptyset. \tag{11}$$

That is, an overapproximation of the one-step reachable set from each partition for each input is used to construct a finite-state abstraction.

Certain classes of dynamical systems exhibit structure that allows efficient reachable-set computation. The well-studied class of *monotone* systems possess a partial order over the state-space that is maintained along the trajectories of the system [34], [35], [52]. Ignoring disturbances, the system $x[t+1] = F(x[t])$ is *monotone* if

$$x_1 \leq x_2 \text{ implies } F(x_1) \leq F(x_2), \tag{12}$$

for all x_1, x_2. Throughout this article, inequalities are interpreted elementwise so that $\leq$ characterizes the partial order induced by the positive orthant, although the definition of monotonicity extends readily to general partial orders. The ordering on trajectories implied by (12) allows sets of trajectories to be bounded by considering appropriate extremal trajectories. For example, (12) implies that for any x such that $x_1 \leq x \leq x_2$, $F(x_1) \leq F(x) \leq F(x_2)$. Therefore, an overapproximation for the reachable set of the hyperrectangle with extreme points x_1 and x_2 is another hyperrectangle with extreme points $F(x_1)$ and $F(x_2)$.

In this article, the focus is on computing one-step reachable sets for a class of *mixed monotone* systems that generalize monotone systems. A dynamical system is mixed monotone if the dependence of the update map F on x and d can be decomposed into increasing and decreasing dependencies as made precise in the definition below. It is then shown that traffic-flow networks are mixed monotone, allowing efficient computation of finite-state abstractions for traffic networks.

Definition 3 (Mixed monotone system)**.** The system (7) is *mixed monotone* if there exists a function $f : \mathcal{X}^2 \times \mathcal{U} \times \mathcal{D}^2 \to \mathcal{X}$ such that the following conditions hold, for all $u \in \mathcal{U}$:

C1) for all $x \in \mathcal{X}$ and $d \in \mathcal{D}$: $F(x,u,d) = f((x,x),u,(d,d))$,

C2) for all $\underline{x}, \overline{x}, y \in \mathcal{X}$ and $\underline{d}, \overline{d}, e \in \mathcal{D}$, it holds that $\underline{x} \leq \overline{x}$ and $\underline{d} \leq \overline{d}$ implies $f((\underline{x},y),u,(\underline{d},e)) \leq f((\overline{x},y),u,(\overline{d},e))$,

C3) for all $x, \underline{y}, \overline{y} \in \mathcal{X}$ and $d, \underline{e}, \overline{e} \in \mathcal{D}$, it holds that $\underline{y} \leq \overline{y}$ and $\underline{e} \leq \overline{e}$ implies $f((x,\overline{y}),u,(d,\overline{e})) \leq f((x,\underline{y}),u,(d,\underline{e}))$.

Mixed monotonicity may be extended to general partial orders of $\mathcal{X}$ and $\mathcal{D}$ [5]. Conditions C2–C3 imply that $f((x,y),u,(d,e))$ is nondecreasing in x and d and nonincreasing in y and e.

A function f satisfying C1–C3 above is called a *decomposition function* for $F(x,u,d)$.

If f is differentiable, then C2 and C3 is equivalent to the following conditions:

C2b) $\frac{\partial f}{\partial x}((x,y),u,(d,e)) \geq 0$ and $\frac{\partial f}{\partial d}((x,y),u,(d,e)) \geq 0$, for all $x,y \in \mathcal{X}$ and $d,e \in \mathcal{D}$,
C3b) $\frac{\partial f}{\partial y}((x,y),u,(d,e)) \leq 0$ and $\frac{\partial f}{\partial e}((x,y),u,(d,e)) \leq 0$, for all $x,y \in \mathcal{X}$ and $d,e \in \mathcal{D}$.

If $f((x,y),u,(d,e)) = F(x,u,d)$ constitutes a decomposition function satisfying C1–C3 above, then the standard characterizations of monotone systems with disturbances is recovered; see [36]. Note that a notion of monotonicity with respect to the controlled input u is not required since $\mathcal{U}$ is a finite set and C1–C3 holds for each input $u \in \mathcal{U}$.

Finding a decomposition function to show mixed monotonicity is often not straightforward. Below,it is shown that a simple decomposition function exists when the Jacobian matrices $\partial F/\partial x$ and $\partial F/\partial d$ are *sign-constant*, that is, the sign of each entry of the Jacobian matrices does not change as x and d varies.

Proposition 2 ([5, Proposition 1])**.** *Consider system* (7) *and assume F is continuously differentiable, and further assume $\mathcal{X}$ and $\mathcal{D}$ are hyperrectangles, that is, there exists $x^1, x^2 \in \mathbb{R}^n$ such that $\mathcal{X} = \{x \mid x^1 \leq x \leq x^2\}$, and similarly for $\mathcal{D}$. If, for all $u \in \mathcal{U}$ and all $i \in \{1,\ldots,n\}$,*

$$\text{for all } j \in \{1,\ldots,n\}, \text{ there exists } \mu_{i,j} \in \{-1,1\} \text{ such that } \mu_{i,j}\frac{\partial F_i}{\partial x_j}(x,u,d) \geq 0, \text{ for all } x,d,$$

and

$$\text{for all } j \in \{1,\ldots,m\}, \text{ there exists } \nu_{i,j} \in \{-1,1\} \text{ such that } \nu_{i,j}\frac{\partial F_i}{\partial d_j}(x,u,d) \geq 0, \text{ for all } x,d,$$

then (1) *is mixed monotone.*

The construction of the decomposition function from the sufficient condition given in Proposition 2 follows naturally from the sign-constant structure of the Jacobian matrices. In particular, the ith element of the decomposition function f_i is defined to be the ith element of the update map F_i where y_j is exchanged for x_j if $\partial F_i/\partial x_j \leq 0$, for all $x \in \mathcal{X}, d \in \mathcal{D}$, and, similarly, e_j is exchanged for d_j if $\partial F_i/\partial d_j \leq 0$, for all $x \in \mathcal{X}, d \in \mathcal{D}$.

Proposition 2 is analogous to the well-known Kamke condition for monotone systems whereby (12) holds if and only if $\partial F_i/\partial x_j \geq 0$, for all i,j [35, Section 3.1], although the condition given in Proposition (2) is only a sufficient condition for mixed monotonicity. Finally, while Proposition 2 assumed F to be continuously differentiable, the results in fact hold if F is

continuous and *piecewise differentiable*, and thus nondifferentiable on a set of measure zero as is the case for traffic networks.

One Step Reachable Set of Mixed Monotone Systems

One of the most important properties of mixed monotone systems is that reachable sets are overapproximated by evaluating the decomposition function at only two points. In particular, given a hyperrectangle of initial conditions and a hyperrectangular disturbance set, the set of states reachable in the next step lies within a hyperrectangle defined by evaluating the decomposition function f at two extreme points, as illustrated in Figure 7 and made precise in Theorem 1.

Theorem 1. *Let* (7) *be a mixed monotone system with decomposition function* $f((x,y),u,(d,e))$ *and consider* $\underline{x},\overline{x} \in \mathcal{X}$ *and* $\underline{d},\overline{d} \in \mathcal{D}$ *with* $\underline{x} \leq \overline{x}$ *and* $\underline{d} \leq \overline{d}$*. Then, for all* $u \in \mathcal{U}$*,*

$$f((\underline{x},\overline{x}),u,(\underline{d},\overline{d})) \leq F(x,u,d) \leq f((\overline{x},\underline{x}),u,(\overline{d},\underline{d})),$$
$$\text{for all } x \in \{x \mid \underline{x} \leq x \leq \overline{x}\},\ \text{for all } d \in \{d \mid \underline{d} \leq d \leq \overline{d}\}.$$

Returning to Figure 5(a), mixed monotonicity allows efficient computation of a hyperrectangle that bounds the darkly shaded one-step reachable set from the indicated partition.

Example 1 (continued)**.** Consider again the network in Figure 4(a) with $\beta_{12} = \beta_{13} = 1/2$, $\alpha_{12} = \alpha_{13} = 1$, and let $D_\ell(x_\ell) = \min\{c_\ell, x_\ell\}$ where $(c_1,c_2,c_3) = (20,5,30)$, for all ℓ, $S_\ell(x_\ell) = 50 - x_\ell$, for all ℓ, and $\mathcal{D} = \{d \mid \underline{d} \leq d \leq \overline{d}\}$, where $\underline{d} = [0\ 5\ 0]^T$ and $\overline{d} = [0\ 8\ 5]^T$. Let $\mathcal{I}_q = \{x \mid \underline{x} \leq x \leq \overline{x}\}$, where $\underline{x} = [40\ 15\ 30]^T$ and $\overline{x} = [40\ 30\ 45]^T$. Then

$$f((\underline{x},\overline{x}),u,(\underline{d},\overline{d})) = [20\ 20\ 10]^T,$$
$$f((\overline{x},\underline{x}),u,(\overline{d},\underline{d})) = [30\ 43\ 25]^T.$$

Then, by Theorem 1,

$$\{F(x,u,d) \mid x \in \mathcal{I}_q\ d \in \mathcal{D}\} \subseteq R,$$

where

$$R \triangleq \{x' \mid f((\underline{x},\overline{x}),u,(\underline{d},\overline{d})) \leq x' \leq f((\overline{x},\underline{x}),u,(\overline{d},\underline{d}))\}.$$

Figure 4(b) plots $\mathcal{I}_q$, R, and the actual reachable set projected in the x_2 vs. x_3 plane.

The partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ is said to be a *hyperrectangular partition* if each $\mathcal{X}_q$ is a hyperrectangle, that is, for all $q \in \mathcal{Q}$ there exists $a_\ell^q \leq b_\ell^q$, for all $\ell \in \mathcal{L}$ such that

$$\mathcal{X}_q = \prod_{\ell\in\mathcal{L}} [a_\ell^q, b_\ell^q]$$

as in Figure 5. Let $a^q = \{a^q_\ell\}_{\ell\in\mathcal{L}}$ and $b^q = \{b^q_\ell\}_{\ell\in\mathcal{L}}$ and assume $\mathcal{D}$ has the form

$$\mathcal{D} = \{d \mid \underline{d} \leq d \leq \overline{d}\}$$

for some $\underline{d}, \overline{d} \in \mathbb{R}^m$; the results extend readily to the case where $\mathcal{D}$ is the union of hyperrectangles.

The restriction to finite abstractions induced by hyperrectangular partitions for mixed monotone systems is justified by the reachability result of Theorem 1. In particular, given a hyperrectangular partition of a mixed monotone system, let

$$R_{q,u} \triangleq \{x' \mid f((a^q, b^q), u, (\underline{d}, \overline{d})) \leq x' \leq f((b^q, a^q), u, (\overline{d}, \underline{d}))\}. \tag{13}$$

Then, by Theorem 1, $R_{q,u}$ satisfies (10).

Theorem 2 ([5, Theorem 2])**.** *Consider the mixed monotone system* (7) *with hyperrectangular partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$. Let $\delta : \mathcal{Q}\times\mathcal{U} \to 2^{\mathcal{Q}}$ be defined as in* (11) *with $R_{q,u}$ given by* (13)*. Then $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ is a finite-state abstraction of* (7)*.*

For hyperrectangular partitions, it is computationally straightforward to identify whether $R_{q,u} \cap \mathcal{X}_{q'} = \emptyset$ by performing two componentwise comparisons of vectors of length $|\mathcal{L}|$, namely, comparing $f((a^q, b^q), u, (\underline{d}, \overline{d}))$ to $b^{q'}$ (resp. $f((b^q, a^q), u, (\overline{d}, \underline{d}))$ to $a^{q'}$). Thus, it is simple to compute $\delta(q, u)$ for each q and u from (11). See [5] for details regarding the computational requirements of obtaining a finite-state abstraction from a hyperrectangular partition using Theorem 2.

Mixed Monotonicity in Traffic Networks

We return to the traffic network dynamics (3). Under a mild technical assumption that $\frac{\partial F_\ell}{\partial x_\ell}(x) \geq 0$, for all $x \in \mathcal{X}$ and all $\ell \in \mathcal{L}$, that is, the diagonal elements of the Jacobian are nonnegative (see [4], [5] for conditions on $\alpha_{\ell k}$, $\beta_{\ell k}$, $D_\ell(x_\ell)$, and $S_\ell(x_\ell)$ that guarantee this assumption), the following theorem establishes mixed monotonicity.

Theorem 3. *Assume the traffic network dynamics are such that $\frac{\partial F_\ell}{\partial x_\ell}(x) \geq 0$, for all $x \in \mathcal{X}$ and all $\ell \in \mathcal{L}$. Then the traffic network dynamics are mixed monotone.*

Theorem 3 is proved for the special case when $S_\ell(x)$ and $D_\ell(x)$ are piecewise linear in [4, Theorem 1], and a more general case in [5, Proposition 3]. The proofs of [4, Theorem 1] and [5, Proposition 3] demonstrate sign constancy of the entries of the Jacobians $\partial F/\partial x$ and $\partial F/\partial d$ and then apply Proposition 2.

The significant step in the proof is establishing that $\frac{\partial F_\ell}{\partial x_k}(x) \leq 0$, for all x if $\ell \neq k$ and

$\tau(\ell) = \tau(k)$. This can be observed in (3) by noting that the incoming flow to link ℓ depends on the outgoing flow of upstream links, which in turn may be limited by the supply of another downstream link $k \neq \ell$. The physical interpretation of this case is as follows. When the supply of downstream link k is less than upstream demand due to congestion, link k inhibits flow through the junction. Therefore, an increase in the number of vehicles on link k would worsen the congestion (decrease supply), and vehicles destined for link k would further block flow to other outgoing links (in particular, link ℓ), causing a reduction in the incoming flow to these links. That is, the derivative of incoming flow to a downstream link $\ell \neq k$ with respect to link k is nonzero and, in particular, is negative since S_k is a decreasing function.

Because $\frac{\partial F_\ell}{\partial x_k} \leq 0$ for ℓ, k at a diverging junction, the traffic dynamics are not monotone in general since, for monotone systems, each entry of the Jacobian matrix is nonnegative. Some of the recent literature in dynamical flow models propose alternative modeling choices for diverging junctions, for example, [53], [54], that ensures the resulting dynamics are monotone but do not exhibit this FIFO property.

Abstraction from Piecewise Linearity

Consider again the special case for which the supply and demand functions are piecewise linear, resulting in the piecewise-affine dynamics (6). It has already been shown that a finite-state abstraction can be constructed by exploiting the mixed monotonicity of the dynamics. However, as noted above, one-step reachable sets that are overapproximations introduce conservatism in the abstraction. This section proposes an alternative abstraction technique that relies on the piecewise-affine dynamics.

For piecewise-affine dynamical systems, it is possible compute the *exact* one-step reachable set from a polytope as the image of the polytope under an affine transformation, which is itself a polytope, as suggested in Figure 5. From this observation, a modification to the above finite abstraction is considered as developed in [26].

Recall the formulation for the piecewise-affine case for which a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ is identified such that each $\mathcal{X}_q$ is a polytope and the dynamics are affine in $\mathcal{X}_q$. To construct a finite-state abstraction, begin again with a partition of the domain $\mathcal{X}$. For convenience of notation, consider the same partition $\mathcal{P}$ induced by the dynamics, however, it is straightforward to consider a refinement $\mathcal{P}' = \{\mathcal{X}_q\}_{q\in\mathcal{Q}'}$ of $\mathcal{P}$ satisfying $\mathcal{X}_{q'} \subseteq \mathcal{X}_q$ for some $q \in \mathcal{Q}$, for all $q' \in \mathcal{Q}'$. Consider the same finite input set $\mathcal{U}$ as before and now assume $\mathcal{D}$ is an arbitrary polytope. The exact one-step reachable set $R^{\text{exact}}_{q,u}$ for each $q \in \mathcal{Q}$ and $u \in \mathcal{U}$ is then computed using polyhedral operations. Defining $\delta(q,u) \triangleq \{q' \mid \mathcal{X}_{q'} \cap R^{\text{exact}}_{q,u} \neq \emptyset\}$ completes he construction

of the finite-state abstraction. This approach is used to compute finite abstractions of freeway network models in [43].

There are several advantages to this approach; first, it is possible to consider arbitrary polyhedral partitions $\mathcal{P}$. Similarly, $\mathcal{D}$ is assumed to be a general polytope. In addition, the exact reachable set computation ensures that there are no spurious one-step transitions in the finite-state abstraction. However, there are several drawbacks. Most seriously, computing the one-step reachable set and determining the set of intersected partitions requires operations that scale exponentially with the dimension of the state space of the real system [55], [56], which is $|\mathcal{L}|$ for traffic networks. For low-dimensional systems, this computation is efficient as compared to more general reachable-set computation techniques, but quickly becomes intractable even for systems of modest size. In contrast, over-approximating the reachable set using mixed monotonicity always requires evaluating the decomposition function at only two points, regardless of the dimension of the state space. Additionally, while exact reachable sets eliminate one-step spurious trajectories, more general spurious trajectories remain, as discussed in "Spurious Transitions in Finite Abstractions". Furthermore, as mentioned above, it is trivial to modify the abstraction algorithm for mixed monotone systems to allow $\mathcal{D}$ to be a union of hyperrectangles that suitably approximates more general disturbance sets. Finally, this alternative abstraction approach requires piecewise-affine dynamics, while the mixed monotone approach applies to a broad class of nonlinear systems.

Case Study

As a case study, consider the network in Figure 8 with five links, $\mathcal{L} = \{1, 2, 3, 4, 5\}$, and three signalized intersections that are denoted by "L", "C", and "R" for the left, center, and right intersections as they appear in Figure 8. The signal in the center either actuates link 1 ("green" mode), or actuates links 4 and 5 simultaneously ("red" mode). The left (respectively, right) signal actuates link 2 (resp. link 3) in "green" mode, and actuates no links in "red" mode (for example, some unmodeled link(s) are actuated in this mode). It follows that $|\mathcal{U}| = 8$ to capture the 8 possible combinations of red/green for the three signals. Time is discretized so that one time step is 15 seconds.

Links 1, 4, and 5 direct exogenous traffic onto the network. To this end, assume the disturbance $d[t] = [d_1\ d_2\ d_3\ d_4\ d_5]^T[t]$ is such that, for all $t \geq 0$,

$$\begin{aligned} d[t] \in \mathcal{D} \triangleq &\{d \mid [0\ 0\ 0\ 0\ 0]^T \leq d \leq [15\ 0\ 0\ 0\ 0]^T\} \\ &\cup \{d \mid [0\ 0\ 0\ 0\ 0]^T \leq d \leq [0\ 0\ 0\ 15\ 15]^T\}, \end{aligned} \tag{14}$$

that is, at each time step, up to 15 vehicles arrive at the queue on link 1, or up to 15 vehicles

each arrives at the queues on links 4 and 5. Assume that traffic divides evenly from link 1 to links 2 and 3 so that $\beta_{12} = \beta_{13} = 0.5$, and further assume $\beta_{52} = \beta_{43} = 0.6$. Furthermore, $\alpha_{52} = \alpha_{43} = \alpha_{12} = \alpha_{15} = 1$. The queue capacity is 40 vehicles so that $x_\ell^{\text{cap}} = 40$, for all ℓ.

We take

$$D_\ell(x_\ell) = \min\{x_\ell, c_\ell\},$$
$$S_\ell(x_\ell) = x_\ell^{\text{cap}} - x_\ell,$$

where it is assumed that $c_\ell = 20$ is the saturation flow, for all ℓ. Thus

$$\begin{aligned}
x_1[t+1] &= x_1[t] - \mathbf{1}(1 \in u) \min\left\{D_1(x_1), \frac{1}{0.5}S_2(x_2), \frac{1}{0.5}S_3(x_3)\right\} + d_1[t],\\
x_2[t+1] &= x_2[t] - \mathbf{1}(2 \in u) D_2(x_2) + \mathbf{1}(1 \in u) \min\left\{0.5D_1(x_1), S_2(x_2), S_3(x_3)\right\}\\
&\quad + \mathbf{1}(5 \in u) \min\left\{0.6D_5(x_5), S_2(x_2)\right\},\\
x_4[t+1] &= x_4[t] - \mathbf{1}(4 \in u) \min\left\{D_4(x_4), \frac{1}{0.6}S_3(x_3)\right\} + d_4[t],
\end{aligned}$$

where

$$\mathbf{1}(\ell \in u) = \begin{cases} 1 & \text{if } \ell \in u,\\ 0 & \text{else,}\end{cases}$$

and the update equations for $x_3[t+1]$ and $x_5[t+1]$ are analogous to $x_2[t+1]$ and $x_4[t+1]$, respectively.

Akin to Example 1, the flow from links 1, 4, and 5 may be blocked by the queues on links 2 and 3. Therefore the dynamics are not monotone but are mixed monotone as in Theorem 3.

The objective is to find a traffic signal control strategy so that the closed-loop dynamics satisfy the LTL objective

$$\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4, \tag{15}$$

where

$$\varphi_1 = \square\diamondsuit(\text{left signal is “red”}), \tag{16}$$
$$\varphi_2 = \square\diamondsuit(\text{right signal is “red”}),$$
$$\varphi_3 = \diamondsuit\square\left(\bigwedge_{i\in\{1,4,5\}} (x_i \le 30)\right),$$
$$\varphi_4 = \square\big((x_2 > 30 \vee x_3 > 30) \implies \diamondsuit(x_2 \le 10 \wedge x_3 \le 10)\big). \tag{17}$$

We interpret (16)–(17) as follows: φ_1 (resp. φ_2) is "infinitely often, the left (resp. right) signal is red", φ_3 is "eventually, the queue on links 1, 4, and 5 have fewer than 30 vehicles and this remains true, for all future time," and φ_4 is "whenever the queue on link 2 or link 3 exceeds 30 vehicles, at some future time, both queues have less than 10 vehicles."

To synthesize a control strategy, a hyperrectangular partition of the state space is first obtained by introducing a gridding of $\mathcal{X} = \prod_{\ell \in \mathcal{L}} [0, x_\ell^{\text{cap}}] \subset \mathbb{R}^5$. Specifically, $[0, x_\ell^{\text{cap}}]$ is divided into the sets of intervals

$$\begin{aligned} \mathbb{Q}_\ell &= \{[0,15], (15,20], (20,25], (25,30], (30,35], (35,40]\} \quad \text{for } \ell \in \{1,4,5\}, \\ \mathbb{Q}_\ell &= \{[0,10], (10,20], (20,30], (30,40]\}, \qquad\qquad\qquad\quad \text{for } \ell \in \{2,3\}. \end{aligned} \tag{18}$$

Then take

$$\mathcal{Q} = \prod_{\ell \in \mathcal{L}} \mathbb{Q}_\ell$$

to index the induced hyperrectangular partition so that, for $q = (q^1, q^2, q^3, q^4, q^5) \in \mathcal{Q}$ with $q^\ell \in \mathbb{Q}_\ell$,

$$\mathcal{X}_q = q^1 \times q^2 \times q^3 \times q^4 \times q^5 \subseteq \mathbb{R}^5.$$

Note that the intervals are compatible with the specification φ so that a labeling function exists mapping each observation that appears in φ to a set of partitions. For example, each $q \in \mathcal{Q}$ for which $q^2 = (30, 40]$ or $q^3 = (30, 40]$ is labeled with the observation $(x_2 > 30 \vee x_3 > 30)$. The resulting transition system has $|\mathcal{Q}| = \prod_{\ell \in \mathcal{L}} |\mathbb{Q}_\ell| = 3456$ states. Building the transition system using the mixed monotone properties of the dynamics takes 35 seconds on a Macbook Pro with a 2.3 GHz Intel Core i7 processor and 8GB of RAM. The average number of transitions from a given partition under a particular input is 73.9. Notice that φ includes specifications on the input, specifically, φ_1 and φ_2 impose conditions on the left and right signals. To accommodate specifications over $\mathcal{U}$, the transition system must be augmented to include the last applied input as a state variable; the details are omitted but are straightforward and the procedure may be found in [4]. The final transition system that models the behavior of the real system then has $8 \times 3456 = 27\,648$ states.

The LTL specification is transformed into a Rabin automaton with 29 modes and one acceptance pair using the `ltl2dstar` tool [57]. Solving the resulting Rabin game takes 43 minutes and results in a control strategy such that the specification is satisfied from any initial condition. Figure 9 plots a resulting trace of the traffic network dynamics. To produce the traces, a random disturbance input is synthesized satisfying (14) for which larger disturbances were favored.

Figure 9(a) shows a resulting trace using the synthesized, correct-by-design control strategy. The figure shows that φ_1 and φ_2 are both satisfied since the left and right signals repeatedly switch to the "red" mode; the switching is done in such a way to ensure that φ_4 is satisfied. To accommodate φ_3, the signaling mode at the center intersection responds to the present conditions that depend on the particular realization of the disturbance input.

In Figure 9(b), a naïve control strategy is used that satisfies φ_1 and φ_2 by using a cyclic control strategy with period 4. This strategy may be considered reasonable since it spends limited time in the "red" mode at the left and right signals, and evenly divides the time between "green" and "red" modes at the center signal. However, this fixed strategy is unable to react to the realized disturbance and does not satisfy φ_3. Furthermore, even if this naïve strategy happens to satisfy φ_4, it is difficult to verify this with certainty. The same initial condition and disturbance input is used in both cases in Figure 9.

In principle, the alternative abstraction method that relies on the piecewise-affine dynamics could be used. However, even with the relatively modest state-space dimension of $|\mathcal{L}| = 5$, computing the finite-state abstraction would be cost prohibitive because it would require $|\mathcal{U}||\mathcal{Q}| = 27\,648$ one-step reachable-set computations and as many as $|\mathcal{U}||\mathcal{Q}|^2 \approx 9.6 \times 10^7$ polyhedral intersection operations to compute δ. In contrast, computing the finite-state abstraction using mixed monotonicity as above takes less than one minute, negligible compared to the Rabin game synthesis computation, and the mixed monotonicity technique has been applied to traffic networks with as many as ten links [4]. Ongoing research for further scalability is discussed in the next section.

Discussion

This article described a formal methods approach to control of traffic-flow networks. First, a dynamical model that captures important traffic-flow phenomena such as blocked flow due to congestion is considered. Several simplifying assumptions were made to arrive at this model; for example, a "single commodity" perspective is adopted whereby all vehicles are assumed to behave similarly. In reality, multiple populations of drivers exist. For instance, truck and freight traffic occupy more physical space and thus links can accommodate fewer vehicles of this type. Accommodating such additions increases model complexity. Moreover, the aggregate model does not capture the interactions of individual vehicles or guarantee, for example, safety from collision, in contrast to [58], [59]. Developing traffic-flow models that are simple enough for computation and analysis yet capture required physical considerations is an important research domain. Many existing approaches to traffic-flow control do not provide guarantees of performance and often

rely on heuristics [60]. When theoretical guarantees are available, it is usually for simplified traffic models. For example, [41], [61] provides an approach to traffic-signal control that achieves optimal throughput but assumes link capacity is infinite.

Next, the article reviewed a general approach to formal synthesis of finite-memory controllers for discrete-time dynamical systems that relies on a finite-state abstraction that *overapproximates* the underlying dynamics. Specifically, for each input, the abstraction enables at least the transitions that are possible in the real system. This approach ensures that a controller synthesized for the abstraction guarantees that the real system satisfies the same specifications.

The general paradigm of abstracting physical control systems to finite-state transition systems for formal synthesis and verification is an important and active area of research. Numerous extensions and alternative approaches have been developed that accommodate a broad range of cases including continuous-time dynamics, continuous inputs, overlapping/uncertain state and input quantization, and probabilistic systems. Each of these cases poses unique challenges, and care must be taken to ensure that a controller synthesized from the abstraction can be effectively and efficiently applied to the real system.

An overarching concern is scalability; many abstraction and formal synthesis techniques do not apply to systems with more than two or three state dimensions due to the need to compute reachable sets. This article has shown that structural properties of the dynamics, such as mixed monotonicity and piecewise linearity, ameliorate some of these issues. Mixed monotonicity is a particularly powerful structural property since the one-step reachable set is overapproximated by computing the decomposition function at only two points regardless of the state-space dimension. This approach has been applied to systems with up to ten state dimensions.

Nonetheless, reachable-set computation is only one of the difficulties in efficient finite-state abstraction; for example, the size of the state-space partition generally increases exponentially with the state-space dimension. An important future direction of research is computing relatively small partitions that are still sufficient for formal synthesis. One approach is to compute the partitions online so that only a relevant subset of the state space is partitioned. Another approach is to methodically adjust the granularity of the partition; for example, in traffic-flow networks, it is plausible that regions of the state space corresponding to few vehicles in the network do not need to be finely partitioned. This idea appears to a degree in the case study, where the first interval of $\mathbb{Q}_\ell$ in (18) is the relatively large interval $[0, 15]$. Other approaches include avoiding partitioning the state space altogether [62], [63], [64], [65]. Furthermore, as seen in the case study, solving the Rabin game takes much longer than computing the abstraction. While Rabin automata can accommodate any LTL expression, "Linear Temporal Logic Specifications of System Behavior"

observed that restricted classes of LTL enable more efficient synthesis algorithms. It is a future direction of research to explore classes of specifications that enable efficient synthesis for traffic networks and other physical control systems. For example, *directed* specifications appear to be particularly relevant for monotone systems (however, these techniques do not yet extend to mixed monotone systems) [66].

Another consideration for scalability is *compositionality*, in which a composite system is viewed as the interconnection of a collection of subsystems. A controller for the composite system is then obtained by synthesizing controllers for each subsystem. Compositional synthesis has emerged as an important method for software verification and synthesis [67], [68]. One successful approach is the *assume-guarantee* framework [69] whereby each subsystem *assumes* a certain behavior from neighboring systems and symmetrically *guarantees* a prescribed behavior to its neighbors. These assumptions and guarantees reduce the synthesis task to decoupled subproblems of manageable complexity and yields local controllers rather than a single, centralized controller. Such an approach is well-suited for traffic networks that may be naturally divided into neighborhoods or towns interconnected via a few roads. This approach is currently being explored [70].

This article considers LTL as the specification language. LTL allows consideration of specific time horizons using repeated application of the "next" operator, however, this approach often results in large Rabin automata. Other temporal logics, such as signal temporal logic (STL), allow direct inclusion of time horizons, for example, by specifying that a certain observation occurs within a specified time horizon [71]. Additionally, it is possible to consider probabilisitic specifications and to include optimality constraints when synthesizing a controller [31], [32], [72]. Probabilistic guarantees are particularly appropriate for domains such as transportation management where correct control is desirable but not absolutely critical. For example, the specification may be "with 95% probability, the traffic link remains uncongested."

A key thesis of this article is that to obtain tractable and scalable formal methods for physical systems, the underlying structure of the systems must be identified and exploited. It is shown that traffic networks are a particularly rich example of physical systems with extensive structure induced by topology, physics, and phenomenological properties. By articulating key structural properties inherent to traffic networks with system-theoretic notions, general theory and algorithms were developed that are more broadly applicable.

From a practical perspective, the confluence of three distinct trends makes the methodology proposed in this article attractive for implementation. First, as already discussed, the rapidly expanding field of formal methods in controls provides ample new theoretical and algorithmic

tools for approaching complex engineering challenges. Second, rapid expansion of dense urban areas has given rise to an unprecedented need for efficient traffic management with systematic guarantees of performance rather than ad hoc approaches. The third enabling trend is the ubiquity of inexpensive and distributed sensors within and around infrastructure systems. Indeed, much of the sensing and control infrastructure required to implement the feedback controllers suggested here already exists. For example, wireless sensors embedded in roads are able to estimate the length of queued vehicles at an intersection and this information is often shared with nearby intersections or aggregated in real time at a centralized location.

Despite these encouraging developments, implementation of new traffic-control technologies requires overcoming many obstacles. For example, existing legacy traffic-control systems often provide little flexibility for new control strategies without expensive hardware upgrades. Furthermore, political barriers are often challenging due to the fact that traffic management responsibilities may be shared among various agencies. For instance, a state or regional agency may be responsible for freeway ramp metering while a local agency is responsible for traffic signal control on the adjacent arterial roads, requiring coordinated efforts among agencies with potentially conflicting interests. Nonetheless, it is becoming clear that there exists a need for smarter, better-engineered cities that take advantage of increasingly connected, societal-scale systems such as transportation infrastructure. As more resources are made available to improve the efficiency, resilience, and sustainability of cities, formal techniques for analysis and control will play a vital role.

Acknowledgements

This research was supported in part by the NSF under grants CNS-1446145 and CNS-1446151.

References

[1] A. A. Kurzhanskiy and P. Varaiya, "Traffic management: An outlook," *Economics of Transportation*, vol. 4, no. 3, pp. 135–146, 2015.

[2] R. Dowling and S. Ashiabor, "Traffic signal analysis with varying demands and capacities, draft final report," Tech. Rep. NCHRP 03-97, Transportation Research Board, 2012.

[3] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "2015 annual urban mobility scorecard," 2015. http://mobility.tamu.edu/ums/report/.

[4] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications," *IEEE Transactions on Control of Network Systems*, vol. 3, pp. 162–172, June 2016.

[5] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pp. 58–67, 2015.

[6] S. Coogan and M. Arcak, "A compartmental model for traffic networks and its dynamical behavior," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2698–2703, 2015.

[7] E. Haghverdi, P. Tabuada, and G. J. Pappas, "Bisimulation relations for dynamical, control, and hybrid systems," *Theoretical Computer Science*, vol. 342, pp. 229–261, Sept. 2005.

[8] P. Tabuada and G. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[9] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.

[10] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[11] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 116–126, 2010.

[12] M. Zamani, P. Mohajerin Esfahani, R. Majumdar, A. Abate, and J. Lygeros, "Symbolic control of stochastic systems via approximately bisimilar finite abstractions," *IEEE Transactions on Automatic Control*, vol. 59, pp. 3135–3150, Dec 2014.

[13] P. Tabuada, *Verification and control of hybrid systems: A symbolic approach*. Springer, 2009.

[14] T. Moor and J. Raisch, "Supervisory control of hybrid systems within a behavioural framework," *Systems & control letters*, vol. 38, no. 3, pp. 157–166, 1999.

[15] T. Moor and J. Raisch, "Abstraction based supervisory controller synthesis for high order monotone continuous systems," in *Modelling, Analysis, and Design of Hybrid Systems*, pp. 247–265, Springer, 2002.

[16] M. Kloetzer and C. Belta, "Dealing with nondeterminism in symbolic control," in *Hybrid Systems: Computation and Control*, pp. 287–300, Springer, 2008.

[17] G. Reissig, "Computing abstractions of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2583–2598, 2011.

[18] J. Liu and N. Ozay, "Abstraction, discretization, and robustness in temporal logic control of dynamical systems," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pp. 293–302, ACM, 2014.

[19] A.-K. Schmuck and J. Raisch, "Asynchronous ℓ-complete approximations," *Systems & Control Letters*, vol. 73, pp. 67–75, 2014.

[20] A. Schmuck, P. Tabuada, and J. Raisch, "Comparing asynchronous ℓ-complete approximations and quotient based abstractions," *arXiv preprint, arXiv:1503.07139*, 2015.

[21] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, pp. 1370–1381, Dec 2009.

[22] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[23] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, pp. 48–61, Feb 2010.

[24] J. Fu, N. Atanasov, U. Topcu, and G. J. Pappas, "Optimal temporal logic planning in probabilistic semantic maps," *arXiv preprint, arXiv:1510.06469*, 2015.

[25] J. Liu, N. Ozay, U. Topcu, and R. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, pp. 1771–1785, July 2013.

[26] B. Yordanov, J. Tůmová, I. Černá, J. Barnat, and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1491–1504, 2012.

[27] B. Yordanov and C. Belta, "Formal analysis of discrete-time piecewise affine systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2834–2840, 2010.

[28] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Sontrol*, pp. 101–110, ACM, 2010.

[29] E. A. Gol, M. Lazar, and C. Belta, "Temporal logic model predictive control," *Automatica*, vol. 56, pp. 78–85, 2015.

[30] E. Wolff, U. Topcu, and R. Murray, "Robust control of uncertain Markov decision processes with temporal logic specifications," in *Proceedings of the 51st IEEE Conference on Decision and Control*, pp. 3372–3379, 2012.

[31] J. Fu and U. Topcu, "Probably approximately correct MDP learning and control with temporal logic constraints," in *Proceedings of Robotics: Science and Systems*, 2014.

[32] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of Markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, 2014.

[33] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *IEEE Conference on Decision and Control*, pp. 1091–1096, 2014.

[34] M. W. Hirsch, "Systems of differential equations that are competitive or cooperative II: Convergence almost everywhere," *SIAM Journal on Mathematical Analysis*, vol. 16, no. 3, pp. 423–439, 1985.

[35] H. L. Smith, *Monotone dynamical systems: An introduction to the theory of competitive and cooperative systems*. American Mathematical Society, 1995.

[36] D. Angeli and E. Sontag, "Monotone control systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 10, pp. 1684–1698, 2003.

[37] H. Smith, "Global stability for mixed monotone systems," *Journal of Difference Equations and Applications*, vol. 14, no. 10-11, pp. 1159–1164, 2008.

[38] S. Coogan, E. Aydin Gol, M. Arcak, and C. Belta, "Controlling a network of signalized intersections from temporal logical specifications," in *Proceedings of the 2015 American Control Conference*, pp. 3919–3924, 2015.

[39] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[40] C. F. Daganzo, "The cell transmission model, part II: Network traffic," *Transportation Research Part B: Methodological*, vol. 29, no. 2, pp. 79–93, 1995.

[41] P. Varaiya, "Max pressure control of a network of signalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 177–195, 2013.

[42] S. Coogan and M. Arcak, "Stability of traffic flow networks with a polytree topology," *Automatica*, vol. 66, pp. 246–253, April 2016.

[43] S. Coogan and M. Arcak, "Freeway traffic control from linear temporal logic specifications," in *Proceedings of the 5th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 36–47, 2014.

[44] A. A. Kurzhanskiy and P. Varaiya, "Active traffic management on road networks: A macroscopic approach," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4607–4626, 2010.

[45] J. C. Munoz and C. F. Daganzo, "The bottleneck mechanism of a freeway diverge,"

Transportation Research Part A: Policy and Practice, vol. 36, no. 6, pp. 483–505, 2002.

[46] M. J. Cassidy, S. B. Anani, and J. M. Haigwood, "Study of freeway traffic near an off-ramp," *Transportation Research Part A: Policy and Practice*, vol. 36, no. 6, pp. 563–572, 2002.

[47] G. Gomes, R. Horowitz, A. A. Kurzhanskiy, P. Varaiya, and J. Kwon, "Behavior of the cell transmission model and effectiveness of ramp metering," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 4, pp. 485–513, 2008.

[48] Transportation Research Board, "Highway capacity manual," 2000.

[49] J. Tůmová, B. Yordanov, C. Belta, I. Černá, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 4230–4235, Dec 2010.

[50] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *arXiv preprint, arXiv:1503.03715*, 2015.

[51] G. Reissig and M. Rungger, "Feedback refinement relations for symbolic controller synthesis," in *IEEE Conference on Decision and Control*, pp. 88–94, Dec 2014.

[52] M. Hirsch and H. Smith, "Monotone maps: a review," *Journal of Difference Equations and Applications*, vol. 11, no. 4-5, pp. 379–398, 2005.

[53] G. Como, E. Lovisari, and K. Savla, "Throughput optimality and overload behavior of dynamical flow networks under monotone distributed routing," *IEEE Transactions on Control of Network Systems*, vol. 2, pp. 57–67, March 2015.

[54] E. Lovisari, G. Como, and K. Savla, "Stability of monotone dynamical flow networks," in *Proceedings of the 53rd Conference on Decision and Control*, pp. 2384–2389, 2014.

[55] A. Kurzhanskiy and P. Varaiya, "Computation of reach sets for dynamical systems," in *The Control Systems Handbook*, ch. 29, CRC Press, second ed., 2010.

[56] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proceedings of the European Control Conference*, (Zürich, Switzerland), pp. 502–510, July 17–19 2013. http://control.ee.ethz.ch/∼mpt.

[57] J. Klein, "`ltl2dstar`-LTL to deterministic Streett and Rabin automata," 2005. http://www.ltl2dstar.de/.

[58] T. T. Johnson and S. Mitra, "Safe and stabilizing distributed multi-path cellular flows," *Theoretical Computer Science*, vol. 579, pp. 9–32, 2015.

[59] H. Roozbehani and R. D'Andrea, "Adaptive highways on a grid," in *Robotics Research*, pp. 661–680, Springer, 2011.

[60] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[61] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections,"

in *Advances in Dynamic Network Modeling in Complex Transportation Systems*, pp. 27–66, Springer, 2013.

[62] M. Zamani, A. Abate, and A. Girard, "Symbolic models for stochastic switched systems: A discretization and a discretization-free approach," *Automatica*, vol. 55, pp. 183–196, 2015.

[63] E. Le Corronc, A. Girard, and G. Goessler, "Mode sequences as symbolic states in abstractions of incrementally stable switched systems," in *Proceedings of the 52nd IEEE Conference on Decision and Control*, pp. 3225–3230, 2013.

[64] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *IEEE Conference on Decision and Control*, pp. 2117–2122, IEEE, 2008.

[65] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5319–5325, IEEE, 2014.

[66] E. S. Kim, M. Arcak, and S. Seshia, "Directed specifications and assumption mining for monotone dynamical systems," in *ACM Conference on Hybrid Systems: Computation and Control*, 2016.

[67] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model checking*. MIT press, 1999.

[68] S. Berezin, S. Campos, and E. M. Clarke, *Compositional reasoning in model checking*. Springer, 1998.

[69] O. Grumberg and D. E. Long, "Model checking and modular verification," *ACM Transactions on Programming Languages and Systems*, vol. 16, pp. 843–871, May 1994.

[70] E. S. Kim, M. Arcak, and S. A. Seshia, "Compositional controller synthesis for vehicular traffic networks," in *IEEE Conference on Decision and Control*, 2015.

[71] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, Springer, 2004.

[72] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimal control with weighted average costs and temporal logic specifications.," in *Robotics: Science and Systems*, 2012.

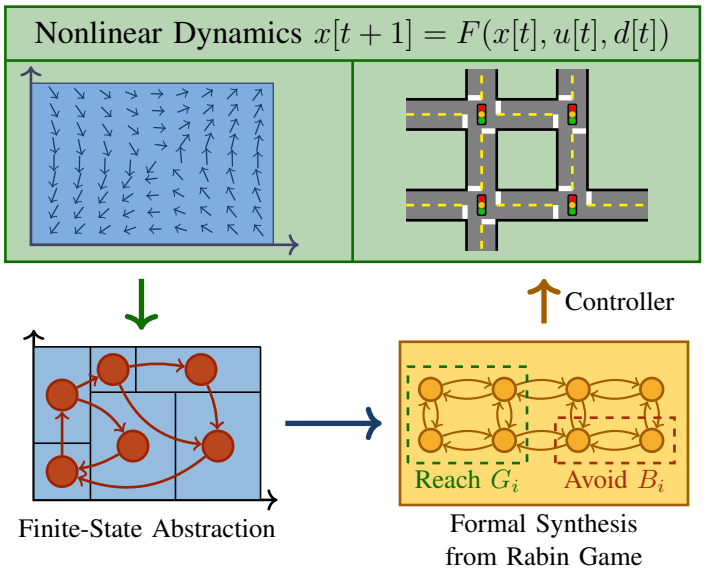


Figure 1. A schematic depiction of the traffic-control synthesis procedure presented in this article. The traffic-flow dynamics are modeled as a discrete-time dynamical system that is approximated with a finite-state abstraction obtained by partitioning the original (continuous) domain. Transitions in the abstraction are obtained from reachability computations and overapproximate the behavior of the system. A finite-memory controller is obtained by solving a Rabin game with a Rabin automaton generated by the specified objective given in linear temporal logic.

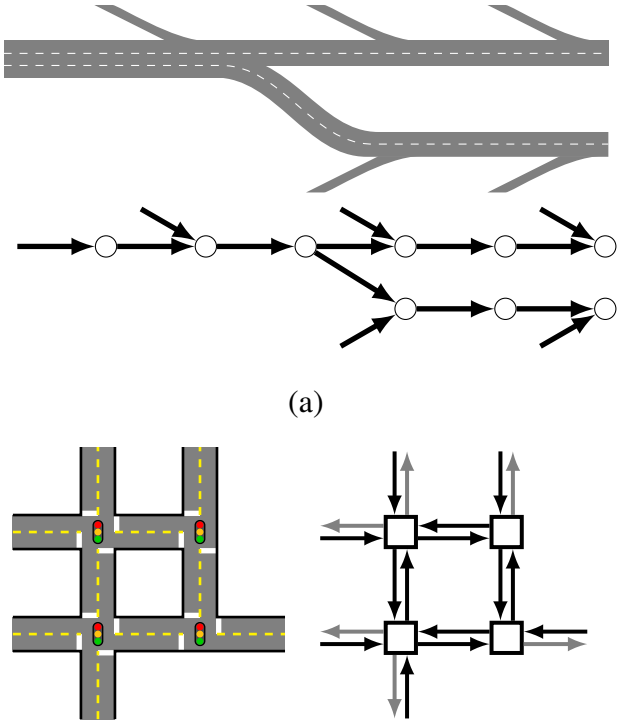


Figure 2. Traffic networks are modeled as interconnected links. (a) A standard freeway network consisting of one freeway with a diverge to a second freeway along with a schematic depiction of the resulting model where each link models a freeway segment. (b) A typical signalized network and its model. The greyed links are not explicitly modeled since they exit the network. At each time step, the signaling input actuates a subset of the incoming traffic. Each link is an incoming road; long links may be subdivided into multiple links, and roads with multiple lanes that are actuated independently may be subdivided into parallel links.

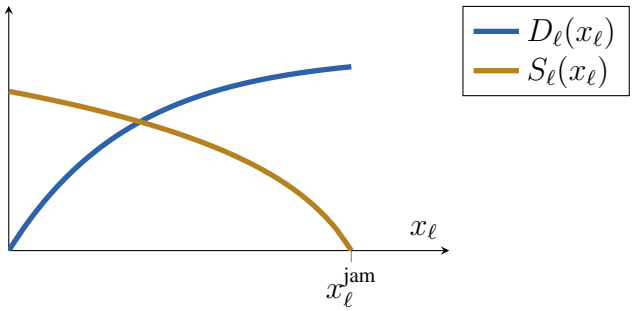
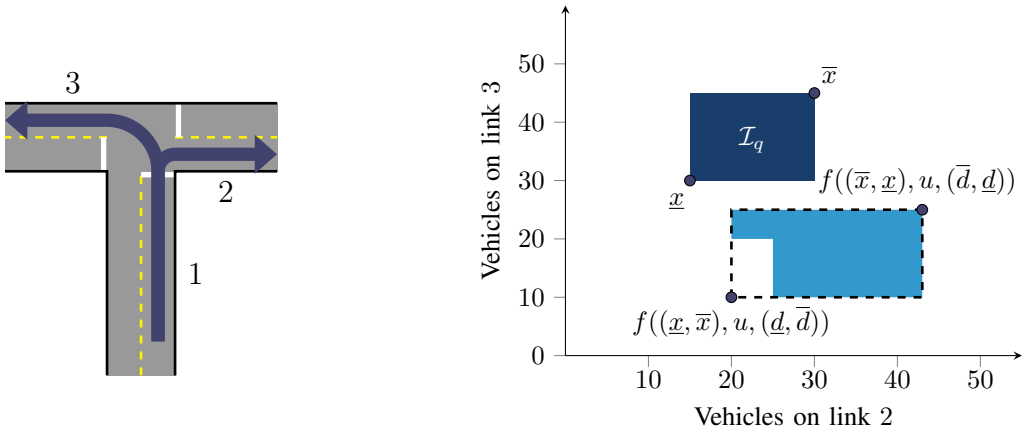
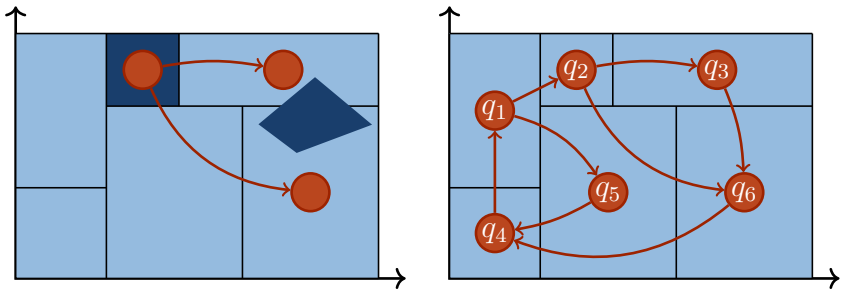


Figure 3. Plot of prototypical supply and demand functions S_ℓ and D_ℓ.



(a) (b)

Figure 4. Approximating the one-step reachable set of traffic states using mixed monotonicity. (a) A simple network with three links. (b) The one-step reachable set from the initial box $\mathcal{I}_q$ is bounded by evaluating the network dynamics of each link at two particular extreme points that depend on the topology of the network. The actual reachable set is shaded in light blue; the approximation R is outlined with a dashed line, and the results are plotted in the plane of Link 2 vs. Link 3.



(a) (b)

Figure 5. Schematic depiction of a finite-state abstraction. The lightly shaded region represents the domain $\mathcal{X}$. (a) The transition map captures all possible transitions from one partition of the state space under each possible input. Here, only one input is assumed for illustration. The darkly shaded region denotes one partition and its image under the state-update map F, that is, the corresponding one-step reachable set. (b) A finite-state abstraction represents the dynamics with a finite set of states and transitions between these states.

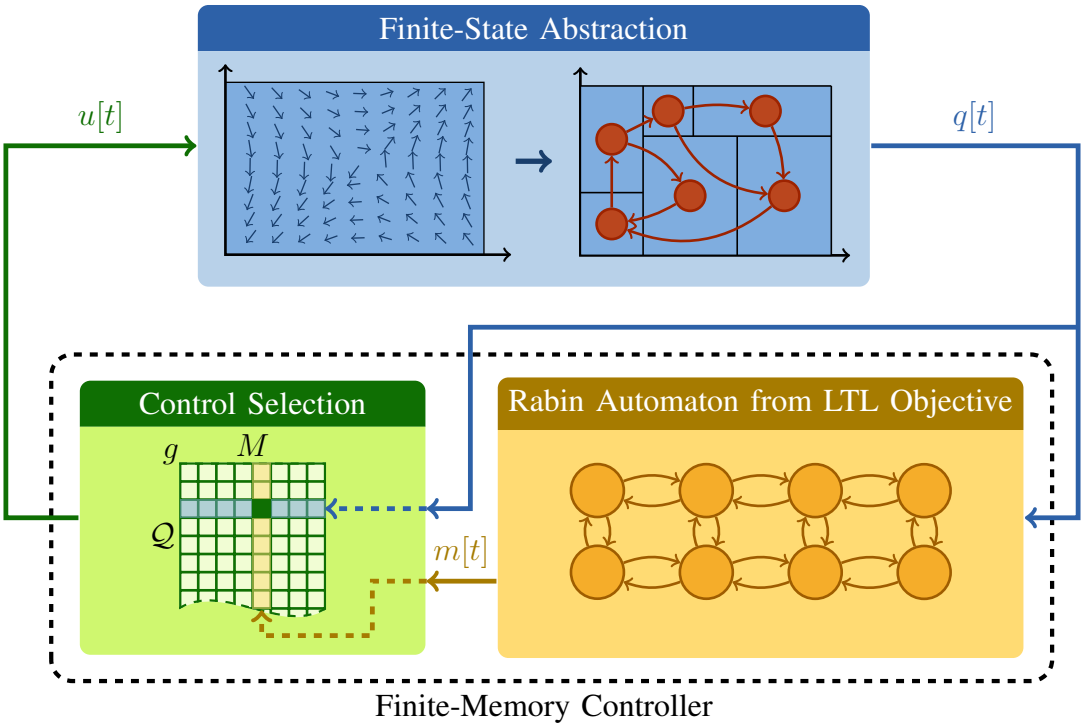
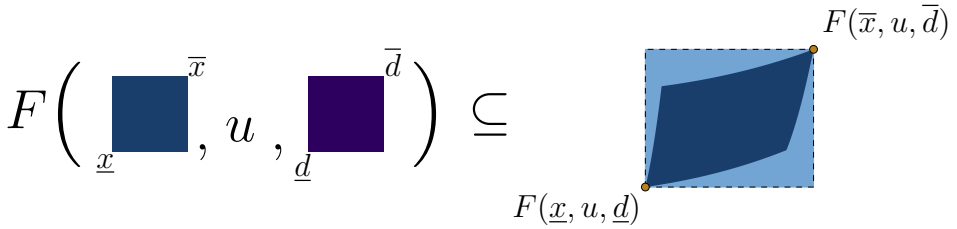
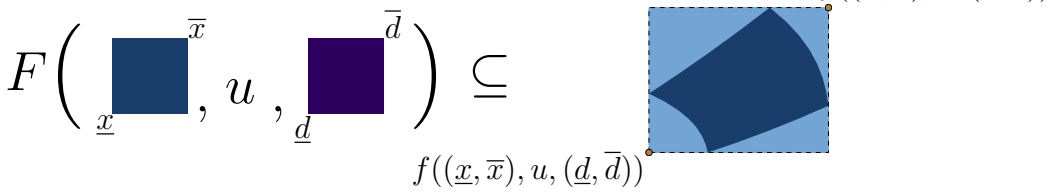


Figure 6. Control of a dynamical system from its finite-state abstraction using a finite-memory controller. An objective given in linear temporal logic (LTL) is converted to a finite Rabin automaton, the modes of which serve as the memory of the controller. The synthesis algorithm produces the control-selection map g, which acts as a finite look-up table mapping a pair $(q[t], m[t])$ to a control action $u[t]$, where $q[t]$ is the state of the system's finite-state abstraction and $m[t]$ is the mode of the Rabin automaton at time t. When this control action is applied to the system, the closed-loop execution is guaranteed to satisfy the LTL objective.



Monotone System

(a)



Mixed Monotone System

(b)

Figure 7. Mixed monotonicity enables efficient overapproximation of reachable sets. (a) For the special case of monotone systems, the hyperrectangle defined by evaluating the update map F at the extreme points $(\underline{x}, \underline{d})$ and $(\overline{x}, \overline{d})$ contains the set of reachable states from the hyperrectangle defined by $\underline{x}$ and $\overline{x}$ under a disturbance taken from the hyperrectangular set defined by $\underline{d}$ and $\overline{d}$. (b) An analogous result holds in the general mixed-monotone case when the decomposition function is evaluated at two extreme points.

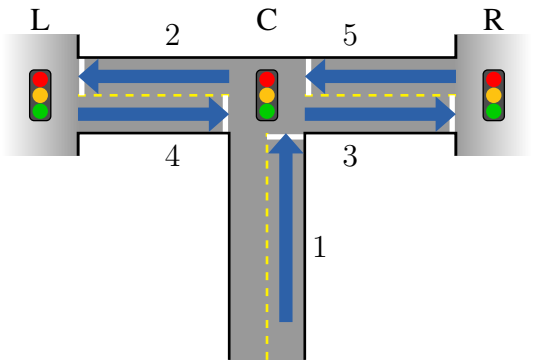
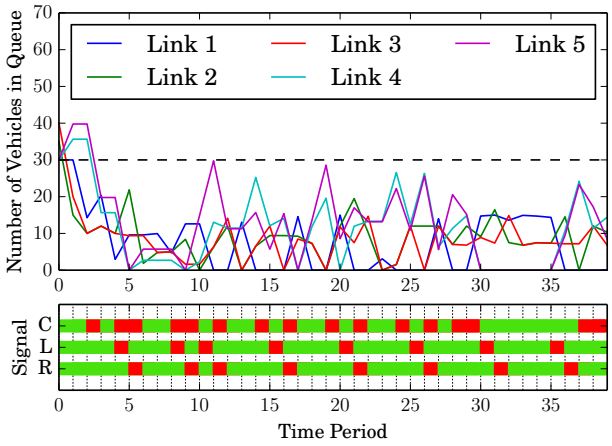
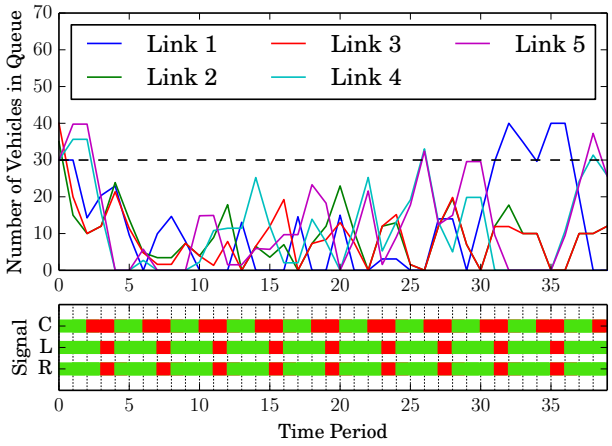


Figure 8. Example network with three signalized intersections and 5 links, denoted with arrows and numbered as shown. Links 1, 4, and 5 direct exogenous traffic onto the network, and at each time step, the exogenous arrivals are assumed to be within the disturbance set $\mathcal{D}$. If the queues on links 2 and 3 are long, they will block flow from links 1, 4, and 5. At each time step, the signal in the center actuates link 1 ("green" mode) or actuates links 4 and 5 simultaneously ("red" mode). The left (resp. right) signal actuates link 2 (resp. link 3) if in "green" mode and none of the modeled links if in "red" mode. A control strategy is synthesized that satisfies the linear temporal logic formula in (15).



(a)



(b)

Figure 9. Sample trajectories of the network in Figure 8. (a) The applied control at each time step is determined by the correct-by-design control strategy obtained using the formal abstraction and synthesis approach presented in this article. As suggested by the sample trajectory, the closed-loop behavior of the real system is guaranteed to satisfy the linear temporal logic (LTL) formula (15). (b) Sample trajectory when a fixed, cyclic control strategy is applied to the network. The initial condition and disturbance input are the same as in (a). The LTL formula (15) is not satisfied for this naïve controller as it cannot be guaranteed the state of links 1, 2, and 3 remain below 30 vehicles.

Sidebar 1
Spurious Transitions in Finite Abstractions

If the transition map δ is the smallest satisfying (8) where "smallest" is with respect to set inclusion, then no spurious one-step transitions exists and (8) holds with the biconditional "if and only if." In this case, $\mathcal{T}$ is called a *quotient-based abstraction* of the real system with respect to the partition $\mathcal{P}$. There are several advantages to stipulating this additional requirement on δ. First, this requirement implies that, given a partition, the corresponding finite-state abstraction is unique. Second, by avoiding spurious transitions, we reduce the conservatism inherent in formal synthesis from finite-state abstractions.

However, there are good reasons to accept spurious one-step transitions in the finite-state abstraction. In particular, computing the smallest transition map requires exact one-step reachability computations under the dynamics (7), which is often computationally difficult or impossible. Yet, for many classes of systems, there exist efficient algorithms for computing overapproximations of reachable sets that are not overly conservative.

Even in the absence of spurious one-step transitions in the map δ, there may still exist spurious executions of the finite-state abstraction that, after two or more steps, do not correspond to any trajectory of the real system.

Example S1. Consider a system for which $\mathcal{X} \subseteq \mathbb{R}^2$ is a rectangle as in Figure S1, $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ for $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$ partitions the domain into four polytopes as shown, and $\mathcal{U}$ and $\mathcal{D}$ are singleton sets. Equivalently, omit $\mathcal{U}$ and $\mathcal{D}$ so that (7) becomes $x[t+1] = F(x[t])$ and $\mathcal{T}$ is a finite abstraction with transition map $\delta : \mathcal{Q} \to 2^{\mathcal{Q}}$. Let $F(\mathcal{Y}) = \{F(x) \mid x \in \mathcal{Y}\}$ for $\mathcal{Y} \subseteq \mathcal{X}$. Suppose $F(\mathcal{X}_{q_1})$ is as in Figure S1 so that $\{q_2, q_3\} \subseteq \delta(q_1)$. Likewise, suppose $F(\mathcal{X}_{q_3})$ is as in the figure so that $\{q_1, q_4\} \subseteq \delta(q_3)$, and, furthermore, $F(F(\mathcal{X}_{q_1}) \cap \mathcal{X}_{q_3}) \subseteq \mathcal{X}_{q_1}$ as indicated by the shaded region of $F(\mathcal{X}_{q_3})$. Then, since $q_3 \in \delta(q_1)$ and $q_4 \in \delta(q_3)$, the sequence $q_1q_3q_4$ consists of valid transitions of the finite-state abstraction $\mathcal{T}$. However, there is no trajectory $x[\cdot]$ of the real system such that $x[n] \in \mathcal{X}_{q_1}$, $x[n+1] \in \mathcal{X}_{q_3}$, and $x[n+2] \in \mathcal{X}_{q_4}$ for some $n \geq 0$, that is, $q_1q_3q_4$ is a spurious sequence.

There are at least three approaches to limiting the existence of spurious executions. The first is to *refine* the partition $\mathcal{P}$. A refinement of a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ is a new partition $\mathcal{P}' = \{\mathcal{X}_q\}_{q\in\mathcal{Q}'}$ such that, for all $q' \in \mathcal{Q}'$, there exists $q \in \mathcal{Q}$ with $\mathcal{X}_{q'} \subseteq \mathcal{X}_q$. For instance, by partitioning $\mathcal{X}_{q_3}$ in the above example, it is possible to obtain an abstraction that does not exhibit the particular spurious sequence in this example. Standard iterative partition-refinement

algorithms exist for quotient-based abstractions [13].

The second approach, which traces its roots to behavioral systems theory [S1], is to compute an *ℓ-complete abstraction* given the fixed partition $\mathcal{P}$ that incorporates finite memory to track past behavior of the system. Here, ℓ refers to the length of the memory, and increasing ℓ reduces the conservatism of the abstraction (that is, removes spurious executions) [14], [19]. For example, in an ℓ-complete abstraction with $\ell = 2$, the currently and previously occupied partitions constitute an expanded state of a finite-state abstraction, and this expanded state is considered when constructing the transition map. In the above example, such an abstraction would not allow the spurious sequence $q_1q_3q_4$ because, for $x[0] \in \mathcal{X}_{q_1}$ and $x[1] \in \mathcal{X}_{q_3}$, it has been concluded that $x[2] \notin \mathcal{X}_{q_4}$ and thus there is not a transition from the expanded abstract state (q_1, q_3) to the expanded abstract state (q_3, q_4).

While ℓ-complete abstractions and quotient-based abstractions obtained from partition refinement are conceptually similiar, they are, in general, not the same. Depending on the real system, one or the other may produce a tighter abstraction. However, in some cases, the abstractions are equivalent; see [20] for a detailed comparison of the two approaches.

The third approach constructs *progress groups* for each finite state under each input that specifies a set of states in which the system cannot remain indefinitely, resulting in an augmented finite-transition system [S2], [S3]. It is possible to compute progress groups by certifying that a set of states is transient under the system dynamics using, for example, barrier functions and *sum-of-squares* programming [S2]. These progress groups translate to *justice* requirements on executions of the abstraction [S4].

The finite-state abstraction $\mathcal{T}$ is deterministic if $|\delta(q, u)| = 1$, for all $q \in \mathcal{Q}$ and $u \in \mathcal{U}$, and nondeterministic otherwise. Nondeterminism arises from three sources:

1) The disturbance d implies that $x[t + 1]$ is not uniquely determined by $x[t]$ and $u[t]$ alone;
2) As discussed above, δ may include spurious one-step transitions;
3) Even when reachable sets are computed exactly, the one-step reachable set from a partition may intersect multiple partitions, as in the example above.

Despite the nondeterminism and existence of spurious transitions, a controller obtained from an overapproximating finite-state abstraction guarantees at least the same level of performance for the real system [13], [50]. In particular, condition (8) implies that the real system is an *alternating simulation* [13] of the finite-state abstraction $\mathcal{T}$ and that there exists a feedback refinement [50] from the real system to the abstraction.

References

[S1] J. C. Willems, "Paradigms and puzzles in the theory of dynamical systems," *IEEE Transactions on Automatic Control*, vol. 36, no. 3, pp. 259–294, 1991.

[S2] N. Ozay, J. Liu, P. Prabhakar, and R. M. Murray, "Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems," in *American Control Conference (ACC)*, pp. 6237–6244, IEEE, 2013.

[S3] F. Sun, N. Ozay, E. M. Wolff, J. Liu, and R. M. Murray, "Efficient control synthesis for augmented finite transition systems with an application to switching protocols," in *American Control Conference (ACC)*, pp. 3273–3280, IEEE, 2014.

[S4] Y. Kesten and A. Pnueli, "Verification by augmented finitary abstraction," *Information and Computation*, vol. 163, no. 1, pp. 203–243, 2000.

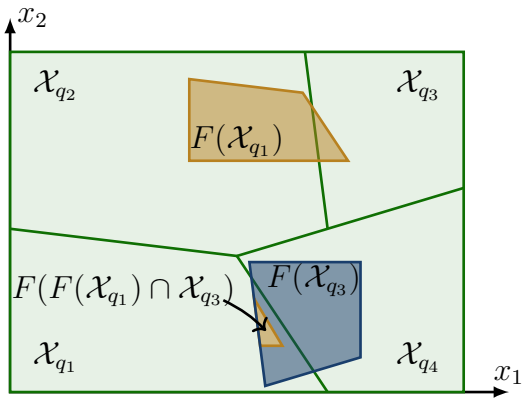


Figure S1. Spurious executions of finite-state abstractions. Since $F(\mathcal{X}_{q_1})$ intersects $\mathcal{X}_{q_2}$ and $\mathcal{X}_{q_3}$, $\{q_2, q_3\} \subseteq \delta(q_1)$, and, likewise, $\{q_1, q_4\} \subseteq \delta(q_3)$ so that the sequence $q_1q_3q_4$ consists of valid transitions of the finite-state abstraction. Yet no trajectory $x[\cdot]$ of the real system is such that there exists n with $x[n] \in \mathcal{X}_{q_1}$, $x[n+1] \in \mathcal{X}_{q_3}$, and $x[n+2] \in \mathcal{X}_{q_4}$ since $F(F(\mathcal{X}_{q_1}) \cap \mathcal{X}_{q_3}) \cap \mathcal{X}_{q_4} = \emptyset$, that is, the sequence $q_1q_3q_4$ is a spurious sequence.

Sidebar 2
Linear Temporal Logic Specifications of System Behavior

Linear temporal logic (LTL) is used to describe the temporal behavior of systems [67], [S6], [S7] and was first suggested for describing the operation of software in [S5]. LTL formulas are constructed from a set of observations, Boolean operators, and temporal operators, and an LTL formula expresses a property over an infinite trace of observations made during the infinite execution of a system. For example, a LTL formula may specify that some observation always holds (invariance), that some observation eventually holds (reachability), that some observation holds infinitely often (liveness), or that some particular observation always follows another observation (sequentiality). The power of LTL comes from its ability to concisely and intuitively express a wide range of relevant properties for system behavior. It is often straightforward to convert a plain English statement directly to a LTL formula as demonstrated below.

The standard notation for the Boolean operators is used, including $\top$ (true), $\neg$ (negation), $\wedge$ (conjunction), and the graphical notation for the temporal operators is used, including $\bigcirc$ ("next"), U ("until"), $\Diamond$ ("eventually"), and $\Box$ ("always"). For example, the LTL formula $\Diamond\Box o_1$ expresses the property that observation $o_1 \in \mathcal{O}$ eventually holds at some point in the future and continues to hold, for all future time. This interpretation, and the construction of valid LTL formulas, is made precise in the following.

Given a finite set of observations $\mathcal{O}$, LTL formulas are interpreted over infinite sequences of subsets of $\mathcal{O}$, that is, over $(2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$ where $2^{\mathcal{O}}$ denotes the set of all subsets of $\mathcal{O}$. For example, in a finite-state abstraction, a fixed set of observations holds for each partition of the continuous state space, and an execution of the dynamical system generates a sequence of sets of observations.

Consider such a sequence $\sigma = \sigma[0]\sigma[1]\sigma[2]\cdots$ with each $\sigma[t] \subseteq \mathcal{O}$. Given a LTL formula φ, the sequence σ either *satisfies* φ or it does not (as is typical, LTL formulas are denoted with the symbol φ). The set of all possible LTL formulas and the interpretation of their satisfaction is defined recursively as:

- For any $o \in \mathcal{O}$, o is a LTL formula and σ satisfies o if $o \in \sigma[0]$, that is, if observation o holds at the first time step.
- For any LTL formulas φ_1 and φ_2, $\varphi_1 \wedge \varphi_2$ is itself a LTL formula and σ satisfies $\varphi_1 \wedge \varphi_2$ if σ satisfies φ_1 and σ satisfies φ_2.
- For any LTL formula φ, $\neg\varphi$ is a LTL formula and σ satisfies $\neg\varphi$ if it is not the case that σ satisfies φ.
- For any LTL formula φ, $\bigcirc\varphi$ is a LTL formula and σ satisfies $\bigcirc\varphi$ if $\sigma[1\cdots] := \sigma[1]\sigma[2]\cdots$

satisfies φ.

- For any LTL formulas φ_1 and φ_2, $\varphi_1 \,\mathsf{U}\, \varphi_2$ is a LTL formula and σ satisfies $\varphi_1 \,\mathsf{U}\, \varphi_2$ if there exists $j \geq 0$ such that $\sigma[j \cdots] := \sigma[j]\sigma[j+1] \cdots$ satisfies φ_2 and, for all $i < j$, $\sigma[i \cdots] := \sigma[i]\sigma[i+1] \cdots$ satisfies φ_1.

The temporal operators $\Diamond$ ("eventually") and $\Box$ ("always") are defined using the "until" temporal operator U:

- For any LTL formula φ, $\Diamond\varphi := \top \,\mathsf{U}\, \varphi$ and thus σ satisfies $\Diamond\varphi$ if φ holds eventually at some time in the future, that is, there exists i such that $\sigma[i \cdots] = \sigma[i]\sigma[i+1] \cdots$ satisfies φ.
- For any LTL formula φ, $\Box\varphi := \neg\Diamond\neg\varphi$ and thus σ satisfies $\Box\varphi$ if $\sigma[i \cdots] := \sigma[i]\sigma[i+1] \cdots$ satisfies φ, for all $i \geq 0$.

Compactly, the syntax for LTL as described above is generated by the grammar

$$\varphi ::= \top \mid o \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2,$$

where $o \in \mathcal{O}$ is an observation and φ, φ_1 and φ_2 are LTL formulas.

It is possible to obtain a wide range of derived operators by combining the above rules. For example, two common temporal operators are $\Diamond\Box$ ("eventually always") and $\Box\Diamond$ ("always eventually"). Below are informal interpretations of the satisfaction of some frequently used LTL formulas.

- $\bigcirc\varphi$ is satisfied at the current step if φ is satisfied at the next step.
- $\varphi_1 \,\mathsf{U}\, \varphi_2$ is satisfied if φ_1 is satisfied "until" φ_2 becomes satisfied.
- $\Box\varphi$ is satisfied if φ is satisfied at each step (that is, φ is "always" satisfied).
- $\Box\neg\varphi$ is satisfied if $\neg\varphi$ is satisfied at each step (that is, φ is "never" satisfied).
- $\Diamond\varphi$ is satisfied if φ is satisfied at some future step (that is, φ is "eventually" satisfied).
- $\Diamond\Box\varphi$ is satisfied if φ becomes satisfied at some future step and remains satisfied for all following steps (*i.e.* φ is satisfied "eventually forever").
- $\Box\Diamond\varphi$ is satisfied if φ always becomes satisfied at some future step (*i.e.* φ is satisfied "infinitely often").

Example. Consider the sequences of subsets of observations generated by the transition diagram shown in Figure S2 for which $\mathcal{O} = \{o_1, o_2, o_3\}$ (in this example, an element of such a sequence is a singleton set containing exactly one observation from $\mathcal{O}$). A possible output word is $\sigma = \{o_1\}\{o_1\}\{o_2\}\{o_3\}(\{o_1\})^{\mathbb{Z}_{\geq 0}}$ where $(\cdot)^{\mathbb{Z}_{\geq 0}}$ indicates infinite repetition of the expression in the parentheses. The sequence σ satisfies LTL formulas $\varphi_1 = o_1$, $\varphi_2 = \Diamond\Box o_1$ and $\varphi_3 = o_1 \,\mathsf{U}\, o_2$.

A different output word is $\sigma' = (\{o_1\}\{o_1\}\{o_2\}\{o_3\})^{\mathbb{Z}_{\geq 0}}$, which satisfies formulas φ_1, φ_3 and $\varphi_4 = \square\diamondsuit o_3$. However, word σ does not satisfy formula φ_4 and σ' does not satisfy φ_2.

References

[S5] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science*, pp. 46–57, IEEE, 1977.

[S6] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[S7] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. LeeSeshia.org, 2011.

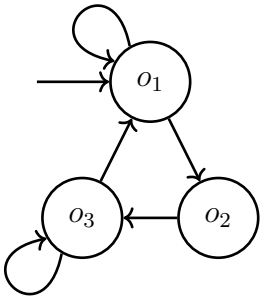


Figure S2. Example of linear temporal logic specifications for system behavior. The transition diagram above represents possible observations from $\mathcal{O} = \{o_1, o_2, o_3\}$ during the execution of a system. The system is initialized with observation o_1 and each edge indicates a possible next observation. One possible sequence of sets of observations for the system is $\sigma = \{o_1\}\{o_1\}\{o_2\}\{o_3\}(\{o_1\})^{\mathbb{Z}_{\geq 0}}$, and another possible sequence is $\sigma' = (\{o_1\}\{o_1\}\{o_2\}\{o_3\})^{\mathbb{Z}_{\geq 0}}$. The sequence σ satisfies $\diamondsuit\square o_1$ but does not satisfy $\square\diamondsuit o_3$. The opposite holds for sequence σ'.

Sidebar 3
Controller Synthesis for LTL Specifications from Rabin Games

Rabin automata connect LTL specifications to executions of finite state abstractions and provide a mechanism for obtaining finite memory controllers of finite state abstractions.

Definition S1 (Deterministic Rabin automaton)**.** A deterministic *Rabin* automaton consists of:

- A finite set of modes M, called *Rabin modes*,
- An initial mode $m_0 \in M$,
- A finite set of inputs $2^{\mathcal{O}}$ that is the set of all subsets of $\mathcal{O}$,
- A mode transition map $\Delta_R : M \times 2^{\mathcal{O}} \to M$, and
- An acceptance condition $F = \{(G_1, B_1), (G_2, B_2), \ldots, G_k, B_k)\}$ where $G_i, B_i \subseteq M$, for all $i \in \{1, \ldots, k\}$, for some $k \geq 1$.

Executions of a deterministic Rabin automaton are defined analogously to executions of a transition system. The input sequence $\sigma[\cdot] = \sigma[0]\sigma[1]\sigma[2]\cdots$ with $\sigma[t] \subseteq \mathcal{O}$, for all t, is *accepted* by the deterministic Rabin automaton if the unique induced execution $m[\cdot]$ satisfies the following *acceptance condition*: There exists a pair $(G_i, B_i) \in F$ for which

- $m[t] \in G_i$ for infinitely many $t \geq 0$, and
- $m[t] \in B_i$ for only finitely many $t \geq 0$ (equivalently, there exists t^* for which $m[t] \notin B_i$, for all $t \geq t^*$).

Each $(G_i, B_i) \in F$ is an *acceptance pair*.

The notational congruences between Definition 2 and Definition S1 are intentional, and the relevance of Rabin automata stems from the following result.

Proposition S1 ([S8], [S9], [S10])**.** *Given an LTL formula φ over the set of observations $\mathcal{O}$, let $\sigma \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$. There exists a deterministic Rabin automaton such that*

$$\sigma \text{ satisfies } \varphi \quad \text{if and only if} \quad \sigma \text{ is accepted by the Rabin automaton.}$$

Because all Rabin automata considered here are deterministic, the "deterministic" modifier is dropped.

Example S2. Assume $\mathcal{O} = \{a, b\}$ and consider the LTL formula $\varphi = \square(a \to \diamondsuit b)$, which is satisfied if, whenever a holds, b holds or will hold at some future time. Consider the Rabin

automaton with $M = \{m_0, m_1\}$, $F = (\{m_0\}, \emptyset)$, and, for $W \in 2^{\mathcal{O}}$, Δ_R is given by

$$\Delta_R(m_0, W) = m_1 \text{ if and only if } a \in W \text{ and } b \notin W, \tag{S1}$$

$$\Delta_R(m_1, W) = m_0 \text{ if and only if } b \in W. \tag{S2}$$

Then a trace $\sigma = \sigma[0]\sigma[1]\sigma[2]\cdots \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$ is accepted by the Rabin automaton if and only if σ satisfies φ; see Figure S3(a). Indeed, F implies that an execution $m[t]$ is accepted if and only if $m[t] = m_0$ for infinitely many $t \geq 0$. Reasoning about Δ_R as given in (S1)–(S2), this is the case if and only if whenever $a \in \sigma[t]$ there exists some time $\tau \geq t$ for which $b \in \sigma[t]$, that is, if and only if σ satisfies φ.

Moreover, there exist algorithms [S9] and readily available software [57] for constructing a Rabin automaton from a LTL formula.

From the above discussion, the interaction of a finite-state abstraction and a Rabin automaton generated from a desired LTL specification φ is as follows. From an initial condition $q[0]$, apply an input sequence $u[\cdot]$ to the finite-state abstraction to generate some nonunique execution $q[\cdot]$ and an associated trace $L(q[\cdot])$. Nonuniqueness of $q[\cdot]$ follows from the nondeterminism in the transition map. To check whether $q[\cdot]$ satisfies φ, apply $L(q[\cdot])$ as the input to the Rabin automaton, which produces a corresponding unique execution $m[\cdot]$ that is used to determine if $L(q[\cdot])$ is accepted according to the acceptance condition in Definition (S1).

The interaction of a finite-state abstraction and a Rabin automaton is envisioned as occurring in parallel; when the abstraction steps from $q[t]$ to $q[t+1]$, the set $L(q[t])$ of observations is passed to the Rabin automaton, which then steps from $m[t]$ to $m[t+1]$. Taking this view, it is possible to envision a controller as in Figure 6 that monitors both the evolution of the abstraction and the Rabin automaton and, at each time step, chooses a control action with the goal of satisfying the acceptance condition of the Rabin automaton so that $q[\cdot]$ satisfies φ.

Indeed, the synthesis is based on a *Rabin game* that is played as follows: a controller (also called the *protagonist* or *scheduler* [S6]), which has access to the current state $q[t]$ of the finite-state abstraction as well as $m[t]$ of the Rabin automaton, seeks a control input $u[t]$ at each time step so that, regardless of how an *adversary* chooses from among the set of possible next states dictated by the set $\delta(q[t], u[t])$, the resulting trajectory trace is accepted by the Rabin automaton.

There exist algorithms for solving Rabin games in time polynomial in $|\mathcal{Q}|$, $|M|$, and $|\delta| = \sum_{u \in \mathcal{U}, q \in \mathcal{Q}} |\delta(q, u)|$ and factorial in k, the number of acceptance pairs [S11], [S12], [S13]. For many LTL specifications of practical significance, k is usually small and often 1. Moreover,

solutions of the game are *memoryless*, meaning that the controller's decision is only a function of the current state q and mode m [S14]. That is, the result of these algorithms is a function $g : \mathcal{Q} \times M \to \mathcal{U}$ and a set $\mathcal{Q}_0$ such that, if the finite-state abstraction is initialized with $q[0] \in \mathcal{Q}_0$, then by choosing $u[t] = g(q[t], m[t])$ at each time instant, the controller is guaranteed to win the Rabin game no matter the choice of the adversary, and thus $q[\cdot]$ satisfies φ.

Example S2 (continued)**.** Consider the labeled transition system in Figure S3(b) where q_2 is labeled with observation a, q_3 is labeled with observation b, and q_1 has no observations, and again consider the LTL formula $\varphi = \square(a \to \diamondsuit b)$ with Rabin automaton in Figure S3(a). The set of available inputs is $\mathcal{U} = \{u_1, u_2\}$ and the induced transitions are indicated in the figure, thus defining the transition map δ with domain $\{q_1, q_2, q_3\} \times \{u_1, u_2\}$. Notice that some of the transitions are nondeterministic, that is, an input induces more than one transition from a particular state. For example, $\delta(q_2, u_1) = \{q_1, q_2\}$. The objective is to find a feedback control policy that selects an input at each time step so that infinite executions of the transition system in Figure S3(b) satisfy φ. It can be verified that any control mapping $g : \{q_1, q_2, q_3\} \times \{m_0, m_1\} \to \{u_1, u_2\}$ with $g(m_1, q_1) = g(m_1, q_2) = u_1$ ensures that φ is satisfied when the labeled transition system is initialized in any state, that is, $\mathcal{Q}_0 = \{q_1, q_2, q_3\}$.

To complete the picture, it is then straightforward to characterize the control strategy γ having the structure in Definition 2. In particular, the modes M and initial mode m_0 are inherited from the Rabin automaton, $\Delta(m, q) = \Delta_R(m, L(q))$, and g is obtained via the aforementioned algorithms. Thus, while the Rabin game solution is a memoryless map from $\mathcal{Q} \times M$ to $\mathcal{U}$, the controller itself has finite memory inherited from the modes of the Rabin automaton. In general, memory is required to accommodate the rich specifications allowed by LTL.

Finally, computational aspects and related approaches to formal synthesis are briefly noted. Implementation of a Rabin-game solver is provided in the `conPAS2` software package [26]. In the worst case, the number of Rabin modes is doubly exponential in the length of the LTL formula, that is, the number of operators in the formula [S9]. If (G_1, B_1) is the only acceptance pair and $B_1 = \emptyset$, as is the case in Figure S3(a), then the Rabin automaton is equivalent to a deterministic *Büchi* automaton, for which the synthesis algorithm is simpler. There are classes of LTL specifications, such as the *generalized reactivity (1)* class that is representable as a collection of deterministic Büchi automata, that enable even more efficient synthesis algorithms and software [21], [S15], [S16], [S17].

References

[S8] R. McNaughton, "Testing and generating infinite sequences by a finite automaton," *Information and control*, vol. 9, no. 5, pp. 521–530, 1966.

[S9] S. Safra, "On the complexity of ω-automata," in *29th Annual Symposium on Foundations of Computer Science*, pp. 319–327, 1988.

[S10] W. Thomas, "Automata on infinite objects," *Handbook of theoretical computer science*, vol. 2, 1990.

[S11] O. Kupferman and M. Y. Vardi, "Weak alternating automata and tree automata emptiness," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, (New York, NY, USA), pp. 224–233, ACM, 1998.

[S12] F. Horn, "Streett games on finite graphs," *Proc. 2nd Workshop Games in Design Verification (GDV)*, 2005.

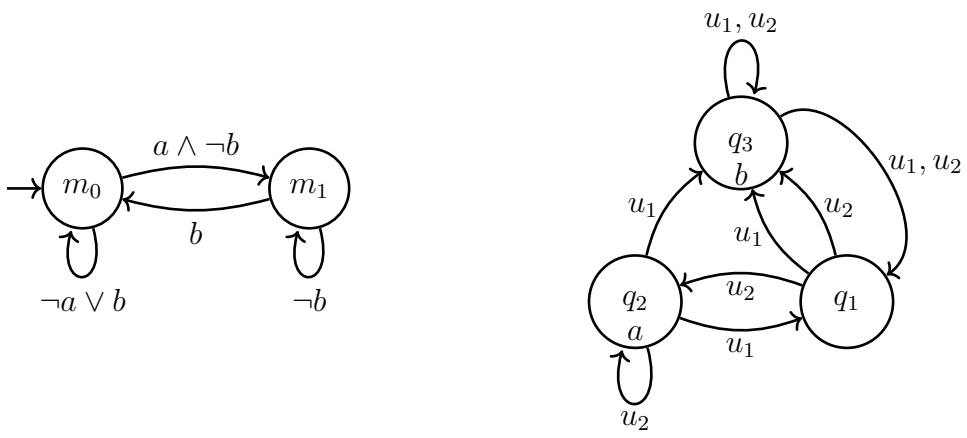
[S13] N. Piterman and A. Pnueli, "Faster solutions of Rabin and Streett games," in *21st Annual IEEE Symposium on Logic in Computer Science*, pp. 275–284, 2006.

[S14] E. A. Emerson, "Automata, tableaux, and temporal logics," in *Logics of Programs*, pp. 79–88, Springer, 1985.

[S15] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," in *Verification, Model Checking, and Abstract Interpretation*, pp. 364–380, Springer, 2006.

[S16] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, temporal logic and robot control," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1988–1993, IEEE, 2010.

[S17] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLIP: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pp. 313–314, ACM, 2011.



(a) (b)

Figure S3. A Rabin automaton generated for a LTL specification is used to generate a feedback control strategy for a labeled transition system. (a) Example of a Rabin automaton that corresponds to the LTL specification $\varphi = \square(a \rightarrow \diamondsuit b)$ for $\mathcal{O} = \{a, b\}$. It holds that $F = \{(m_0, \emptyset)\}$, thus a trace is accepted if and only if m_0 is visited infinitely often. The edge labeled b from m_1 to m_0 indicates that if $m[t] = m_1$ and the observation $\sigma[t] \in 2^{\mathcal{O}}$ is made at time t, then the Rabin automaton will transition to m_0 if and only if $b \in \sigma[t]$, and similarly for the other edges. (b) A labeled transition system for which the observation made at state q_2 is a and the observation made at q_3 is b, and two inputs $\mathcal{U} = \{u_1, u_2\}$ are available at each state. Any control selection map $g : \{q_1, q_2, q_3\} \times \{m_0, m_1\} \rightarrow \{u_1, u_2\}$ satisfying $g(m_1, q_1) = g(m_1, q_2) = u_1$ ensures that the transition system satisfies φ when the Rabin mode $m[t]$ is updated according to the Rabin automaton.

Author Information

Samuel Coogan is an assistant professor in the Electrical Engineering Department at the University of California, Los Angeles (UCLA). He received the B.S. degree in Electrical Engineering from Georgia Tech (2010), and the M.S. and Ph.D. degrees in Electrical Engineering from the University of California, Berkeley (2012 and 2015). In 2015, he was a postdoctoral research engineer at Sensys Networks, Inc., and in 2012 he was a research intern at NASA's Jet Propulsion Lab. He received the Leon O. Chua Award from UC Berkeley EECS in 2014 for achievement in nonlinear science, the best student paper award at the 2015 Hybrid Systems: Computation and Control conference, and the Eli Jury Award from UC Berkeley EECS in 2016 for achievement in systems and control. His research focuses on developing scalable tools for verification and design of networked systems with particular emphasis on applications to transportation systems.

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.

Calin Belta is a professor in the Department of Mechanical Engineering, Department of Electrical and Computer Engineering, and the Division of Systems Engineering at Boston University, where he is also affiliated with the Center for Information and Systems Engineering (CISE) and the Bioinformatics Program. His research focuses on dynamics and control theory, with particular emphasis on hybrid and cyber-physical systems, formal synthesis and verification, and applications in robotics and systems biology. He is a senior member of the IEEE and an associate editor for the SIAM Journal on Control and Optimization (SICON) and the IEEE Transactions on Automatic Control. He received the Air Force Office of Scientific Research Young Investigator Award and the National Science Foundation CAREER Award.