Finite State Abstraction and Formal Methods for Traffic Flow Networks

Samuel Coogan, Murat Arcak, and Calin Belta

***Abstract*— Formal methods from computer science have emerged as a powerful suite of tools that, under appropriate modifications, are applicable to a wide range of physical control systems. These methods promise automated algorithms for verification and synthesis of controllers to accomplish specifications and objectives that are not accommodated by traditional approaches. However, formal methods require a finite abstraction of the underlying physical process, and challenges in obtaining scalable abstraction techniques have impeded the applicability of these automated tools. This tutorial paper exploits structural properties in a class of networked systems motivated by traffic flow networks to overcome some of these challenges and points towards new directions of research.**

The first aim of this tutorial paper is to review a broad technique 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 second objective is to review vehicular traffic flow models and to characterize a general model amenable to formal control synthesis. This general model is shown to be mixed monotone, a generalization of monotone dynamical systems. Using properties of the mixed monotone dynamics, a finite state abstraction is efficiently computed by overapproximating the set of states that are one-step reachable under the traffic flow dynamics. These results are demonstrated on a case study which leads to a discussion of open problems and avenues for future research.

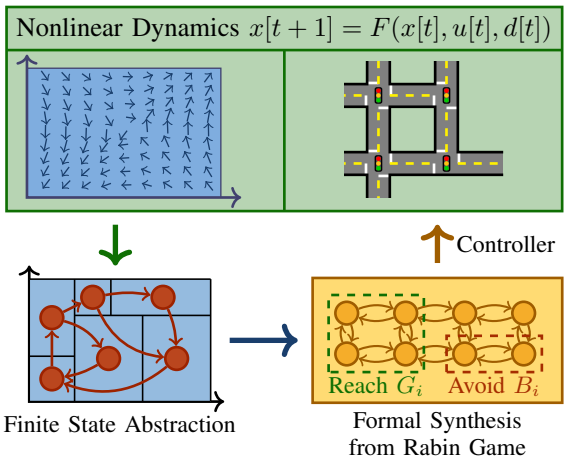


Fig. 1. A schematic depiction of the traffic control synthesis procedure presented in this paper. The traffic flow dynamics are modeled as a discrete-time dynamical system which is approximated with a finite state abstraction obtained by partitioning the original (continuous) domain. Transitions in the 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 LTL objective.

I. Introduction

A. Inefficient Traffic Management is Pervasive

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 $160 billion annually, 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 ever more daunting.

Addressing this complexity requires scalable and automated verification and synthesis techniques for transportation systems. 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.

This research was supported in part by the NSF under grants CNS-1446145 and CNS-1446151. S. Coogan is with the Electrical Engineering Department, UCLA, scoogan@ucla.edu. M. Arcak is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, arcak@eecs.berkeley.edu. C. Belta is with the Department of Mechanical Engineering, Boston University, cbelta@bu.edu.

B. Formal Methods For Control Synthesis

Control techniques often focus on relatively mundane 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).

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 formal methods rely on finite state models

whereas control theoretic approaches typically consider continuous state spaces. A full review of the rapidly growing literature in this area is beyond the scope of this paper, but we highlight some work that is particularly relevant. Certain classes of systems allow finite *bisimulations* such that the dynamics are exactly represented by a finite transition model [4], [5], or are amenable to a related notation of approximate bisimilarity [6], [7], [8], [9], [10]. When a finite bisimulation is not possible, methods exist to approximate the behavior of the underlying system with finite abstractions; [11], [12], [13], [14], [15], [16], [17] are particularly relevant to the ideas presented in this paper. Applications and special cases include robotic path planning [18], [19], [20], [21], switched continuous time systems [22], piecewise linear systems [23], [24], model predictive control formulations [25], [26], and control of Markov decision processes [27], [28], [29], [30].

C. Paper Outline

The first aim of this tutorial paper is to review a broad technique 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. Our primary goal is to give an accessible review of these results and, more generally, formal methods for control synthesis. The second objective is to review vehicular traffic flow models and to characterize a general model amenable to formal control synthesis. We show that the dynamics of traffic flow networks exhibit considerable structure and demonstrate that such structure enables efficient finite state abstraction. In doing so, we point to the importance of identifying and exploiting inherent structural properties. The contents of this tutorial paper are based on results presented in [31], [32], [33].

In Section II, we define a compartmental model for traffic flow networks. A compartmental model considers traffic flow networks to be a collection of links interconnected at junctions, and vehicles flow from link to link through the junctions depending on physically and phenomenologically motivated flow policies. 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.

In Section III, we define 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, we may guarantee properties of the behavior of the original dynamical system subject to these inputs.

Using the finite state abstraction, we propose an automatic controller synthesis procedure in Section IV to guarantee that the abstraction and the underlying system satisfy an objective given in temporal logic. The synthesis algorithm, illustrated schematically in Fig. 1, relies on automata theory and fixed point algorithms to compute a finite memory control strategy.

In Section V, we specialize the ideas of Sections II and III to traffic flow networks. We observe that traffic flow networks are *mixed monotone* [34], [32], a generalization of monotone dynamical systems [35], [36], [37]. Mixed monotone systems are decomposable 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. We then note that mixed monotonicity allows efficient computation of finite state abstractions because one-step reachable sets may be approximated efficiently by computing a certain decomposition function at two extreme points. This approach is particularly attractive since the computational cost of approximating the set of states that are one-step reachable from another set of states does not increase with the dimension of the state space.

In Section VI, we apply the techniques and results of the prior sections and consider a case study example for which a control strategy is efficiently computed using the mixed monotone property of the traffic flow dynamics. We also comment on the practical limitations of an alternative abstraction procedure that instead uses the piecewise affine properties of the underlying dynamics; this approach is developed in Sections II-B and V-D. In Section VII, we comment on extensions and areas for future work.

D. Notation

For set $Y \subset \mathbb{R}$, finite set $\mathcal{I}$, and an element $v \in Y^{\mathcal{I}}$, where $Y^{\mathcal{I}}$ denotes the set of functions from $\mathcal{I}$ to Y, we use subscript to index the elements of v, that is, $v_i = v(i) \in Y$ for $i \in \mathcal{I}$ so that $v = \{v_i\}_{i\in\mathcal{I}}$. We identify v with the obvious corresponding element of the Euclidean space $\mathbb{R}^{|\mathcal{I}|}$ where $|\mathcal{I}|$ denotes the cardinality of $\mathcal{I}$ and we assume some enumeration of $\mathcal{I}$. The powerset of $\mathcal{I}$ is denoted by $2^{\mathcal{I}}$.

Let $\mathbb{Z}_{\geq 0}$ denote the nonnegative integers so that $W^{\mathbb{Z}_{\geq 0}}$ denotes the set of infinite sequences of elements from some set W. As we use this construction exclusively to represent a sequence of time, we index $w \in W^{\mathbb{Z}_{\geq 0}}$ with brackets and write $w = w[0]w[1]w[2]\cdots$ where $w[t] \in W$ for all t. We sometimes write $w = w[\cdot]$ to emphasize the time dependence.

We further 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 $n \geq 0$.

We let $\mathbb{R}_{\geq 0} = \{x \mid x \geq 0\}$. For vectors $x, y \in \mathbb{R}^n$, we interpret $x \leq y$ elementwise, that is, $x \leq y$ if and only if $x_i \leq y_i$ for $i = 1, \ldots, n$, and similarly for $<, \geq, >$.

II. A Compartmental Model for Dynamic Traffic Flow Networks

In this section, we present a compartmental model for the dynamics of traffic flow networks. The model in this paper takes a *macroscopic* view of traffic flow by considering aggregate conditions of the network such as occupancy of vehicles on each road segment and traffic flow rate rather

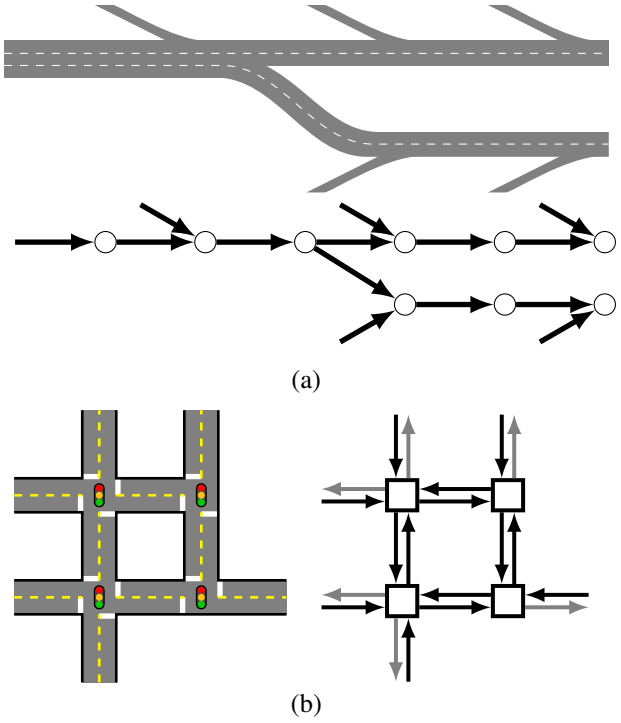


Fig. 2. Vehicular traffic networks are modeled as interconnected links in a compartmental model. (a) A standard freeway network consisting of one freeway with a diverge to a second freeway along with a schematic depiction of the resulting compartmental model where each link models a freeway segment. (b) A typical signalized network and its compartmental model. The greyed links are not explicitly modeled since they exit the network. At each time step, the signalling 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.

than considering movement of individual vehicles. This model was proposed in [31], [38] and encompasses the cell transmission model of freeway traffic flow [39], [40] and queue forwarding models as in [41] where we further account for the finite capacity of queues.

A. General Model

A traffic network consists of a set of *links* $\mathcal{L}$ interconnected at a set of *nodes* $\mathcal{V}$ as in Fig. 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 we present a discrete-time model, the results easily extend to continuous time; see [33], [42]. The state of link $\ell \in \mathcal{L}$ at discrete time t is denoted by

$$x_\ell[t] \in [0, x_\ell^{\text{cap}}] \quad \text{for all } \ell \in \mathcal{L} \tag{1}$$

and represents the number of vehicles occupying link ℓ where x_ℓ^{cap} is the maximum number of vehicles accommodated by link ℓ. For freeway networks, x_ℓ^{cap} is called the *jam density*. Note that we adopt a fluid-like model of traffic flow and do not restrict $x_\ell[t]$ to integer values. The domain is

$$\mathcal{X} \triangleq \prod_{\ell \in \mathcal{L}} [0, x_\ell^{\text{cap}}]. \tag{2}$$

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 $\Phi_\ell^{\text{out}}(\cdot)$ and a decreasing *supply* function $\Phi_\ell^{\text{in}}(\cdot)$. We make the following assumption for each $\ell \in \mathcal{L}$:

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

Prototypical demand and supply functions are shown in Fig. 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

$$\mathcal{U} \subset 2^{\mathcal{L}} \tag{3}$$

be a collection of sets of links that may be simultaneously actuated. We call $u \in \mathcal{U}$ 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, we introduce the *turn ratio* $\beta_{\ell k} > 0$ 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

$$\sum_{k \in \mathcal{L}} \beta_{\ell k} \leq 1 \quad \text{for all } \ell \in \mathcal{L} \tag{4}$$

where $\beta_{\ell k} \neq 0$ only if $\sigma(\ell) = \tau(k)$ and strict inequality in (4) implies that a nonzero fraction of the outgoing flow from link ℓ exits the network along, for example, unmodeled roads or driveways.

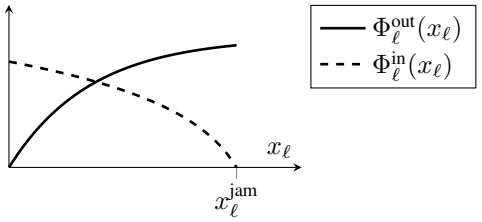


Fig. 3. Plot of prototypical supply and demand functions $\Phi_\ell^{\text{in}}(x_\ell)$ and $\Phi_\ell^{\text{out}}(x_\ell)$.

We symmetrically introduce the *supply ratio* $\alpha_{\ell k}$ for each $\ell, k \in \mathcal{L}$ denoting the fraction of link k's supply available to link ℓ. For all $u \in \mathcal{U}$, we have

$$\sum_{\{\ell \in u \mid \sigma(\ell)=\tau(k)\}} \alpha_{\ell k} = 1 \quad \text{for all } k \in \mathcal{L}, \tag{5}$$

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.

We are now able to define the outflow of vehicles from a link as a function of the state $x \in \mathcal{X}$ and a chosen actuation $u \in \mathcal{U}$. The outflow of link ℓ is

$$f_\ell^{\text{out}}(x,u) = \begin{cases} \min\left\{\Phi_\ell^{\text{out}}(x_\ell), \min\limits_{k \text{ s.t. } \beta_{\ell k}\neq 0} \dfrac{\alpha_{\ell k}}{\beta_{\ell k}} \Phi_k^{\text{in}}(x_k)\right\} & \\ \qquad\qquad \text{if } \ell \in u & \\ 0 \qquad\qquad \text{else,} & \end{cases} \tag{6}$$

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:

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

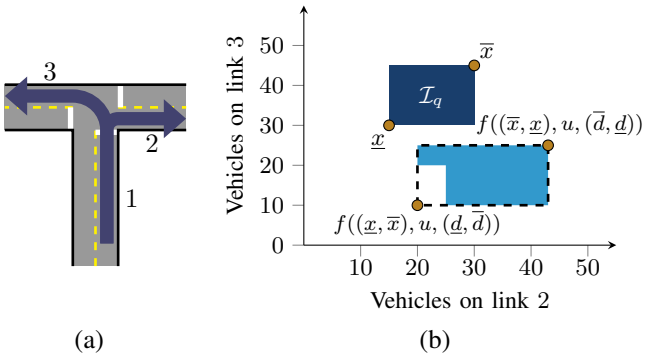
$$\triangleq x_\ell[t] - f_\ell^{\text{out}}(x[t],u[t]) + \sum_{k\in\mathcal{L}} \beta_{k\ell} f_k^{\text{out}}(x[t],u[t]) + d_\ell[t] \tag{8}$$

where $d_\ell[t]$ is an exogenous flow entering link ℓ. We assume that the exogenous flow is truncated and therefore $d_\ell[t]$ is such that $x_\ell[t+1] \le x_\ell^{\text{cap}}$ always. In general, we further assume $d[t] \in \mathcal{D}$ for disturbance set $\mathcal{D} \subseteq (\mathbb{R}_{\ge 0})^{\mathcal{L}}$ for all time.

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

$$f_1^{\text{out}}(x,u^{\text{all}}) = \min\left\{D_1(x_1), \frac{1}{0.5}S_2(x_2), \frac{1}{0.5}S_3(x_3)\right\} \tag{9}$$

$$f_\ell^{\text{out}}(x,u^{\text{all}}) = D_\ell(x_\ell), \quad \ell \in \{2,3\}. \tag{10}$$



(a) (b)

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

■

B. 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,

$$\Phi_\ell^{\text{out}}(x_\ell) = \min\{v_\ell x_\ell, c_\ell\} \tag{11}$$

$$\Phi_\ell^{\text{in}}(x_\ell) = w_\ell(x_\ell^{\text{cap}} - x_\ell) \tag{12}$$

for constants $v_\ell > 0$, $w_\ell > 0$, and $c_\ell > 0$. For freeway networks, v_ℓ and w_ℓ are the *free flow speed* and *congested wave speed* [44]. For signalized networks, we interpet x_ℓ as the queue length and take $v_\ell = w_\ell = 1$. Then c_ℓ is the *saturation flow rate* [45], $\Phi_\ell^{\text{out}}(x_\ell)$ is the minimum of the queue length x_ℓ and the saturation flow rate, and $\Phi_\ell^{\text{in}}(x_\ell)$ is the unoccupied queue capacity of link ℓ.

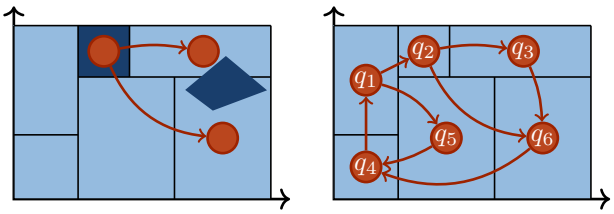
When the demand and supply functions have the form (11)–(12), the dynamics are *piecewise affine*, that is, there exists a set of polytopes $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ for some index set $\mathcal{Q}$ such that $\cup_{q\in\mathcal{Q}}\mathcal{X}_q = \mathcal{X}$ and $\mathcal{X}_q \cap \mathcal{X}_{q'} = \emptyset$ for all $q, q' \in \mathcal{Q}$, and such that for each $q \in \mathcal{Q}$, we have

$$F(x,u,d) = A_{q,u}x + b_{q,u} + d \quad \text{for all } x \in \mathcal{X}_q \tag{13}$$

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 (6) and (11). In the sequel, we construct a finite state abstraction using the tools in [23], [46] that exploit the piecewise affine nature of the dynamics.

III. Overapproximating Finite Abstractions

We now describe a methodology for computing a finite state abstraction that overapproximates, in a particular sense, the dynamics of a discrete-time dynamical system. 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



(a) (b)

Fig. 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, we assume only one input 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.

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.

We consider a discrete-time dynamical system of the form

$$x[t+1] = F(x[t], u[t], d[t]) \tag{14}$$

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. We call this system the *concrete* system, in contrast with the finite state *abstraction* developed subsequently.

A. Finite State Abstractions

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

$$\pi_{\mathcal{P}} : \mathcal{X} \to \mathcal{Q} \tag{15}$$

$$\pi_{\mathcal{P}}(x) = q \text{ when } x \in \mathcal{X}_q \tag{16}$$

be the projection map from $\mathcal{X}$ to $\mathcal{Q}$. For a trajectory $x[\cdot]$ of the dynamical system (14), we let $\pi_{\mathcal{P}}(x[\cdot])$ denote the sequence $q[0]q[1]q[2]\cdots \in \mathcal{Q}^{\mathbb{Z}_{\geq 0}}$.

Definition 1 (Finite state abstraction)**.** Given a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ of $\mathcal{X}$ for system (14), we say that $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ with $\delta : \mathcal{Q} \times \mathcal{U} \to 2^{\mathcal{Q}}$ is a *finite state abstraction* of (14) if

$$\text{for all } x \in \mathcal{X} \text{ and } d \in \mathcal{D} \quad x \in \mathcal{X}_q \text{ and } F(x,u,d) \in \mathcal{X}_{q'} \text{ implies } q' \in \delta(q,u) \tag{17}$$

for any $q, q' \in \mathcal{X}$, $u \in \mathcal{U}$. We call δ 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 $t \geq 0$. ■

Fig. 5 illustrates how a finite state abstraction is obtained for a dynamical system. A finite state abstraction of the concrete dynamical system (14) 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 concrete system. Each input $u \in \mathcal{U}$ enables a set of transitions as determined by $\delta(q, u)$. We will use the notion of *state* to refer both to an element $q \in \mathcal{Q}$ in the finite state abstraction and an element $x \in \mathcal{X}$ of the concrete system when it is clear that no confusion will arise.

Note the direction of implication in (17) 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, we say that 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.

B. Spurious Transitions in Finite Abstractions

If the transition map δ is the smallest satisfying (17) where "smallest" is with respect to set inclusion, then no spurious one-step transitions exists and (17) holds with the biconditional "if and only if." In this case, $\mathcal{T}$ is called a *quotient based abstraction* of the concrete 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 (14), 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 concrete system.

Definition 2 (Spurious sequence and execution)**.** Given a finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ of a concrete system (14). We say a sequence $q[n]q[n+1]\cdots q[m]$ with $n < m$ is *spurious* if there does not exist any trajectory $x[\cdot]$ of the concrete system for which $x[t] \in \mathcal{X}_{q[t]}$ for $t = n, \ldots, m$. An execution $q[\cdot], u[\cdot]$ of $\mathcal{T}$ is spurious if $q[\cdot]$ contains a spurious subsequence. ■

The idea of spurious executions is best illustrated with an example.

Example 2. Consider a system for which $\mathcal{X} \subseteq \mathbb{R}^2$ is a rectangle as in Fig. 6, $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ 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, we omit $\mathcal{U}$ and $\mathcal{D}$ so that (14) becomes $x[t+1] = F(x[t])$ and $\mathcal{T}$ is a finite abstraction with transition map $\delta : \mathcal{Q} \to 2^{\mathcal{Q}}$. We abbreviate

$F(\mathcal{Y}) = \{F(x) \mid x \in \mathcal{Y}\}$ for $\mathcal{Y} \subseteq \mathcal{X}$. Suppose $F(\mathcal{X}_{q_1})$ is as in Fig. 6 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, we have that $F(F(\mathcal{X}_{q_1}) \cap \mathcal{X}_{q_3}) \subseteq \mathcal{X}_{q_1}$ 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 concrete 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 two standard 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 example, by partitioning $\mathcal{X}_{q_3}$ in the above Example, we could obtain an abstraction that does not exhibit the particular spurious sequence in this example. Standard iterative partition refinement algorithms exist for quotient based abstractions [10].

The second approach, which traces its roots to behavioral systems theory [47], 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 (*i.e.*, removes spurious executions) [11], [16]. 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 $x[1] \in \mathcal{X}_{q_3}$, we have concluded that we must have $x[2] \notin \mathcal{X}_{q_4}$ 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 generally incomparable. Depending on the concrete system, one or the other may produce a tighter abstraction. In some cases, the abstractions are equivalent; see [17] for a detailed comparison of the two approaches.

The finite state abstraction $\mathcal{T}$ is deterministic if $|\delta(q, u)| = 1$ for all $q \in \mathcal{Q}$, $u \in \mathcal{U}$ 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 the same performance for the concrete system [10], [48]. In particular, condition (17) implies that the concrete system is an *alternating simulation* [10] of the finite state abstraction $\mathcal{T}$ and that there exists a feedback refinement [48] from the concrete system to the abstraction.

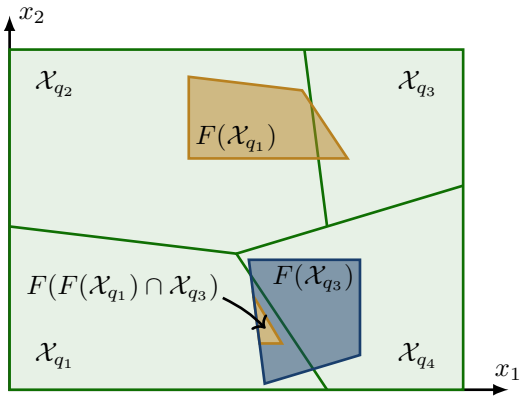


Fig. 6. Spurious executions of finite state abstractions. Since $F(\mathcal{X}_{q_1})$ intersects $\mathcal{X}_{q_2}$ and $\mathcal{X}_{q_3}$, we have that $\{q_2, q_3\} \subseteq \delta(q_1)$, and likewise $\{q_1, q_4\} \subseteq \delta(q_3)$ so that the sequence $q_1q_3q_4$ consists of valid transitions of the finite state abstraction. Yet no trajectory $x[\cdot]$ of the concrete 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.

IV. Specifying System Behavior

We focus 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 allows us to capture 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 $\Diamond$ ("eventually"), and a LTL formula is usually denoted by φ.

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 Section III. 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.

We first begin with a description of LTL and then show how a controller can be synthesized from a finite state abstraction and a LTL specification by solving a two-player game.

A. Linear Temporal Logic

Linear temporal logic (LTL) is used to describe the temporal behavior of systems [49], [50], [51] and was first suggested for describing the operation of software in [52]. LTL formulae are constructed from a set of observations, Boolean operators, and temporal operators, and a 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 we see below.

We use the standard notation for the Boolean operators including $\top$ (true), $\neg$ (negation), $\wedge$ (conjunction), and the graphical notation for the temporal operators including $\bigcirc$ ("next"), U ("until"), $\diamondsuit$ ("eventually"), and $\square$ ("always"). For example, the LTL formula $\diamondsuit\square o_1$ 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. We make this interpretation, and the construction of valid LTL formulae, precise in the following.

Given a finite set of observations $\mathcal{O}$, LTL formulae 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. The set of all possible LTL formulae and the interpretation of their satisfaction is defined recursively as follows:

- For any $o \in \mathcal{O}$, o is a LTL formula and σ satisfies o if $o \in \sigma[0]$, that is, if o holds at the first time step.
- For any LTL formulae φ_1 and φ_2, $\varphi_1 \wedge \varphi_2$ is 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 formulae 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 $i < j$, we have $\sigma[i\cdots] := \sigma[i]\sigma[i+1]\cdots$ satisfies φ_1.

Using the "until" temporal operator U, we define the temporal operators $\diamondsuit$ ("eventually") and $\square$ ("always"):

- For any LTL formula φ, $\diamondsuit\varphi := \top \mathsf{U} \varphi$ and thus σ satisfies $\diamondsuit\varphi$ if φ holds eventually at some future time, that is, there exists i such that $\sigma[i\cdots] = \sigma[i]\sigma[i+1]\cdots$ satisfies φ.
- For any LTL formula φ, $\square\varphi := \neg\diamondsuit\neg\varphi$ and thus σ satisfies $\square\varphi$ 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 following grammar:

$$\varphi ::= \top \mid o \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \tag{18}$$

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

By combining the above rules, we obtain a wide range of derived operators. For example, two common temporal operators are $\diamondsuit\square$ ("eventually always") and $\square\diamondsuit$ ("always eventually").

Below are informal interpretations of the satisfaction of some frequently used LTL formulae.

- $\bigcirc\varphi$ 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.
- $\square\varphi$ is satisfied if φ is satisfied at each step (*i.e.* φ is "always" satisfied).
- $\square\neg\varphi$ is satisfied if $\neg\varphi$ is satisfied at each step (*i.e.* φ is "never" satisfied).
- $\diamondsuit\varphi$ is satisfied if φ is satisfied at some future step (*i.e.* φ is "eventually" satisfied).
- $\diamondsuit\square\varphi$ is satisfied if φ becomes satisfied at some future step and remains satisfied for all following steps (*i.e.* φ is satisfied "eventually forever").
- $\square\diamondsuit\varphi$ is satisfied if φ always becomes satisfied at some future step (*i.e.* φ is satisfied "infinitely often").

Definition 3 (Labeled systems)**.** We say the dynamical system (14) with partition $\mathcal{P} = \{\mathcal{X}_q\}_{q \in \mathcal{Q}}$ is *labeled* if there exists a set of atomic propositions $\mathcal{O}$ and a labeling function $L : \mathcal{X} \to 2^{\mathcal{O}}$ that satisfies $x, y \in \mathcal{X}_q \implies L(x) = L(y)$, that is, elements in the same partition are labeled with the same atomic propositions. We then say that the corresponding finite state abstraction $\mathcal{T}$ is *labeled*, for which there exists a well-defined labeling function $L_{\mathcal{T}} : \mathcal{Q} \to 2^{\mathcal{O}}$ such that $L_{\mathcal{T}}(q) = L(x)$ for all $x \in \mathcal{X}_q$. Throughout this paper, we use $L(\cdot)$ to denote either labeling function. ■

Definition 4 (Trace)**.** Consider a trajectory $x[\cdot]$ of a labeled dynamical system (14) induced by the input sequence $u[\cdot]$. The *trace* of the trajectory is the sequence $L(x[0])L(x[1])L(x[2])\cdots \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$. We abbreviate this sequence as $L(x[\cdot])$. Similarly, for an execution $q[\cdot], u[\cdot]$ of a finite state abstraction, the *trace* of the execution is the sequence $L(q[0])L(q[1])L(q[2])\cdots$, abbreviated as $L(q[\cdot])$. ■

Consider a labeled finite state abstraction $\mathcal{T}$ of a labeled dynamical system (14) with partition $\mathcal{P}$. Equation (17) guarantees that, for a trajectory $x[\cdot]$ of the dynamical system (14) generated by the input sequence $u[\cdot]$, the pair $\pi_{\mathcal{P}}(x[\cdot]), u[\cdot]$ is an execution of $\mathcal{T}$.

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

B. Controller Synthesis from Rabin Games

We consider the labeled dynamical system $x[t+1] = F(x[t], u[t], d[t])$ as in (14) with observations $\mathcal{O}$ and a partition $\mathcal{P}$ of the domain $\mathcal{X}$, along with a LTL objective φ. Informally, our objective is to find a feedback control strategy such that the resulting closed loop trajectories satisfy φ. To make this formal, we will instead define our objective in terms of a labeled finite state abstraction $\mathcal{T}(\mathcal{Q}, \mathcal{U}, \delta)$ for the

dynamical system; we will see that synthesizing a controller from the finite state abstraction is sufficient for obtaining a feedback controller for the concrete system in a form to be made precise below.

Definition 5 (Control strategy)**.** A feedback *control strategy* γ for a labeled finite state abstraction $\mathcal{T}$ is a map

$$\gamma : (2^{\mathcal{O}})^+ \to \mathcal{U} \tag{19}$$

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

Control synthesis objective. Our objective is to find a control strategy γ of the form (19) 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 $u[t] = \gamma(q[0]q[1]\cdots q[t])$. ■

We do not consider the initial condition to be fixed *a priori* but instead consider a set of acceptable initial conditions to be a result of our synthesis procedure. This is because the synthesis algorithm we employ identifies all acceptable initial conditions for our finite state abstraction. If instead we know that the abstraction will initiate in some subset of states, we simply check to see if the set of acceptable states as determined by our algorithm contains the specified set of initial conditions.

Definition 6 (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 $u[t] = \gamma(q[0]\cdots q[t])$ in the following way: the controller transition system (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 we will see below, finite memory controllers suffice for our purposes.

To synthesize a finite memory controller for a LTL specification, we 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. We consider 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 which 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.

Definition 7 (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

$$F = \{(G_1,B_1),(G_2,B_2),\ldots,G_k,B_k)\} \tag{20}$$

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 $t \geq 0$
- $m[t] \in B_i$ 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 6 and Definition 7 are intentional. Our interest in Rabin automata stems from the following result:

Proposition 1 ([53], [54], [55])**.** *Given a LTL formula φ over the set of observations $\mathcal{O}$. There exists a deterministic Rabin automaton such that the following holds for all $\sigma \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$:*

$$\sigma \text{ satisfies } \varphi \iff \sigma \text{ is accepted by the Rabin automaton.} \tag{21}$$

As all Rabin automata considered here are deterministic, we drop the "deterministic" modifier.

Example 3. Assume $\mathcal{O} = \{a,b\}$ and consider the LTL formula $\varphi = \square(a \to \Diamond b)$ 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 the following rule:

$$\Delta_R(m_0,W) = m_1 \text{ if and only if } a \in W \text{ and } b \notin W, \tag{22}$$

$$\Delta_R(m_1,W) = m_0 \text{ if and only if } b \in W. \tag{23}$$

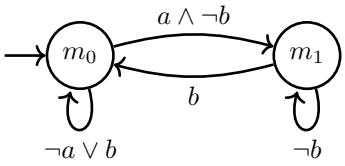


Fig. 7. Example of a Rabin automaton that corresponds to the LTL specification $\varphi = \square(a \rightarrow \lozenge b)$ for $\mathcal{O} = \{a, b\}$. We have $F = \{(m_0, \emptyset)\}$, thus a trace is accepted if and only if m_0 is visited infinitely often. The edge labeled b from m_1 to m_0 indicates 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.

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 Fig. 7(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 (22)–(23), 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 [54] and readily available software [56] for constructing a Rabin automaton from a LTL formula. An important difficulty is that, for the worst case, the number of Rabin modes $|M|$ grows doubly exponentially in the length of the LTL formula [54]. However, it has been observed that this theoretical worst case is rarely encountered in practice.

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 $q[0]$, we apply a sequence of inputs $u[\cdot]$ to the finite state abstraction to generate an execution $q[\cdot]$ and an associated trace $L(q[\cdot])$. To check whether $q[\cdot]$ satisfies φ, we apply $L(q[\cdot])$ as the input to the Rabin automaton, which produces a corresponding unique execution $m[\cdot]$ that we use to determine if $L(q[\cdot])$ is accepted according to the acceptance condition in Definition (7).

We may envision the interaction of a finite state abstraction and a Rabin automaton 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, we may then envision a controller 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* [50]), 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 a so-called *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 [57], [58], [59]. 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 [60]. That is, the result of these algorithms is a function $g : \mathcal{Q} \times M \rightarrow \mathcal{U}$ 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.

To complete the picture, it is then straightforward to characterize the finite memory control strategy γ having the structure in Definition 6. 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.

C. Employing Abstraction-Based Controllers

We apply a finite memory controller of the form given in Definition 6 to the original concrete system in the natural way: we let $u[t] = g(m[t], \pi_P(x[t]))$ at each time step t, and we update the controller mode as $m[t+1] = \Delta(m[t], \pi_P(x[t]))$. The following Proposition is straightforward and implies that the overapproximating finite state abstraction is sufficient for formal control synthesis.

Proposition 2. *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 concrete 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 we take $q[t] = \pi_P(x[t])$ for all t.*

The proof follows readily from the overapproximating nature of $\mathcal{T}$ as specified in (17). In particular, consider any trajectory $x[\cdot]$ of the concrete 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 2 is that we may obtain a controller for the concrete system 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, which implies a degree of robustness to measurement errors. 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 as the controller essentially consists of two lookup tables, each of dimension $|M| \times |\mathcal{Q}|$, corresponding to g and Δ. Even for very large $\mathcal{Q}$, we see that the online computation time is low. The tradeoff is that the offline computation of g and Δ can be costly.

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 concrete system. Furthermore, if the relationship between the abstraction and the concrete system is not taken advantage of fully, one may obtain a controller that is not symbolic and, moreover, requires significantly more online computational resources. These intricacies have been the focus of recent research [48], [61].

Finally, Proposition 2 is based on the abstraction $\mathcal{T}$ which overapproximates the behavior of the concrete system. While this overapproximation is sufficient for ensuring correctness when a controller for the abstraction exists, our attempt to synthesize a controller from the abstraction may fail due to spurious trajectories that are nonexistent in the concrete system. This conservatism is unavoidable for all but a few limited classes of dynamical systems that are amenable to *finite bisimulation* [10].

V. Finite Abstractions of Traffic Flow Networks from Mixed Monotonicity

In Section III, we defined finite state abstractions for discrete-time dynamical systems as a finite transition system that overapproximates the behavior of the underlying concrete dynamical system. In Section IV, we developed an algorithm to synthesize a control strategy for the concrete system from the overapproximating abstraction. The developments of these two sections were generally applicable to any discrete-time dynamical system, but we did not address the difficulties of computing the finite state abstraction; this is the focus of the present section.

A. Mixed Monotone Dynamical Systems

We again consider the dynamical system $x[t+1] = F(x[t], u[t], d[t])$ as in (14) and partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ for which we wish to construct a finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$. If we can calculate an overapproximation of the one-step reachable set from $\mathcal{X}_q$ under input u,

$$R_{q,u} \supseteq \{F(x,u,d) \mid x \in \mathcal{X}_q, d \in \mathcal{D}\}, \tag{24}$$

then we obtain a transition map δ satisfying (17) from

$$q' \in \delta(q,u) \iff \mathcal{X}_{q'} \cap R_{q,u} \neq \emptyset. \tag{25}$$

That is, we use an overapproximation of the one-step reachable set from each partition for each input 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 which is maintained along trajectories of the system [35], [36], [62]. Ignoring disturbances, the system $x[t+1] = F(x[t])$ is *monotone* if

$$x_1 \leq x_2 \implies F(x_1) \leq F(x_2) \tag{26}$$

for all x_1, x_2. Throughout this paper, 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 (26) allows us to bound sets of trajectories by considering appropriate extremal trajectories. For example, (26) implies that for any x such that $x_1 \leq x \leq x_2$, we have that $F(x_1) \leq F(x) \leq F(x_2)$. 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 paper, we focus on computing one-step reachable sets for a class of *mixed monotone* systems which 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 we make precise in the definition below. We then show that traffic flow networks are mixed monotone, allowing us to efficiently compute finite state abstractions of traffic networks.

Definition 8 (Mixed monotone system)**.** The system (14) 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) $\forall x \in \mathcal{X},\ \forall d \in \mathcal{D}$: $F(x,u,d) = f((x,x),u,(d,d))$

C2) $\forall \underline{x}, \overline{x}, y \in \mathcal{X}$ and $\underline{d}, \overline{d}, e \in \mathcal{D}$: $\underline{x} \leq \overline{x}$ 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) $\forall x, \underline{y}, \overline{y} \in \mathcal{X}$ and $d, \underline{e}, \overline{e} \in \mathcal{D}$: $\underline{y} \leq \overline{y}$ 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 $\mathcal{D}$ [32]. We see that $f((x,y),u,(d,e))$ satisfying C2–C3 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 we may replace C2–C3 with:

C2b) $\frac{\partial f}{\partial x}((x,y),u,(d,e)) \geq 0$ and $\frac{\partial f}{\partial d}((x,y),u,(d,e)) \geq 0$,

C3b) $\frac{\partial f}{\partial y}((x,y),u,(d,e)) \leq 0$ and $\frac{\partial f}{\partial e}((x,y),u,(d,e)) \leq 0$,

which must hold for all $x, y \in \mathcal{X}$, $d, e \in \mathcal{D}$.

If $f((x,y),u,(d,e)) = F(x,u,d)$ constitutes a decomposition function satisfying C1–C3 above, we recover standard characterizations of monotone systems with disturbances; see [37]. Note that we do not require a notion of monotonicity with respect to the controlled input u since we consider $\mathcal{U}$ to be a finite set, and we stipulate C1–C3 to hold for each input $u \in \mathcal{U}$.

Finding a decomposition function to show mixed monotonicity is often not straightforward. Below, we show 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.

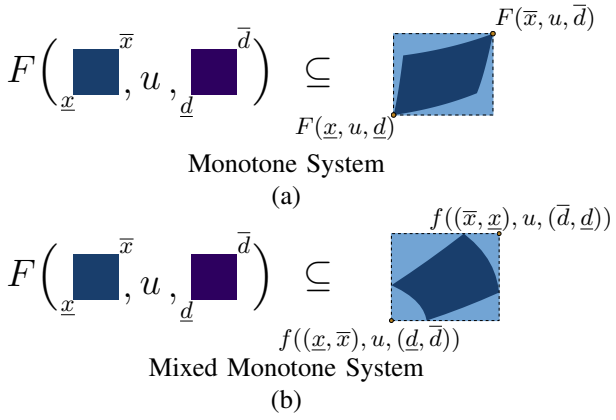


Fig. 8. Mixed monotonicity enables efficient overapproximation of reachable sets. (a) For the special case of monotone systems, the hyperrectangle defined by evaluating the update map F at the extreme points $(\underline{x}, \underline{d})$ and $(\overline{x}, \overline{d})$ contains the set of reachable 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.

Proposition 3 ([32, Proposition 1])**.** *Consider system* (14) *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 $i \in \{1, \dots, n\}$*

$$\begin{aligned} &\forall j \in \{1, \dots, n\} \\ &\exists \mu_{i,j} \in \{-1, 1\} : \mu_{i,j} \frac{\partial F_i}{\partial x_j}(x, u, d) \geq 0 \ \forall x, d \end{aligned} \tag{27}$$

and

$$\begin{aligned} &\forall j \in \{1, \dots, m\} \\ &\exists \nu_{i,j} \in \{-1, 1\} : \nu_{i,j} \frac{\partial F_i}{\partial d_j}(x, u, d) \geq 0 \ \forall x, d \end{aligned} \tag{28}$$

then (4) *is mixed monotone.*

The construction of the decomposition function from the sufficient condition given in Proposition 3 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 we exchange y_j for x_j if $\partial F_i / \partial x_j \leq 0$ for all $x \in \mathcal{X}, d \in \mathcal{D}$, and we similarly exchange e_j for d_j if $\partial F_i / \partial d_j \leq 0$ for all $x \in \mathcal{X}, d \in \mathcal{D}$.

Proposition 3 is analogous to the well-known Kamke condition for monotone systems whereby (26) holds if and only if $\partial F_i / \partial x_j \geq 0$ for all i, j [36, Section 3.1], although the condition given in Proposition (3) is only a sufficient condition for mixed monotonicity. Finally, while Proposition 3 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.

B. One Step Reachable Sets of Mixed Monotone Systems

One of the most important properties of mixed monotone systems is that we are able to overapproximate reachable sets 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 Fig. 8 and made precise in the following:

Theorem 1. *Let* (14) *be a mixed monotone system with decomposition function $f((x, y), u, (d, e))$. Given $\underline{x}, \overline{x} \in \mathcal{X}$ and $\underline{d}, \overline{d} \in \mathcal{D}$ with $\underline{x} \leq \overline{x}$ and $\underline{d} \leq \overline{d}$. Then for all $u \in \mathcal{U}$,*

$$\begin{aligned} f((\underline{x}, \overline{x}), u, (\underline{d}, \overline{d})) \leq F(x, u, d) \leq f((\overline{x}, \underline{x}), u, (\overline{d}, \underline{d})) \\ \forall x \in \{x \mid \underline{x} \leq x \leq \overline{x}\} \ \forall d \in \{d \mid \underline{d} \leq d \leq \overline{d}\}. \end{aligned} \tag{29}$$

Returning to Fig. 5(a), mixed monotonicity allows us to efficiently compute a hyperrectangle that bounds the darkly shaded one-step reachable set from the indicated partition.

Example 1 (continued)**.** Consider again the network in Fig. 4(a) with $\beta_{12} = \beta_{13} = 1/2$, $\alpha_{12} = \alpha_{13} = 1$, and let $\Phi_\ell^{\text{out}}(x_\ell) = \max\{c_\ell, x_\ell\}$ where $(c_1, c_2, c_3) = (20, 5, 30)$ for all ℓ, $\Phi_\ell^{\text{in}}(x_\ell) = 50 - x_\ell$ 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 $\overline{x} = [40\ 30\ 45]^T$. We have

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

$$f((\overline{x}, \underline{x}), u, (\overline{d}, \underline{d})) = [30\ 43\ 25]^T. \tag{31}$$

Then, by Theorem 1,

$$\{F(x, u, d) \mid x \in \mathcal{I}_q \ d \in \mathcal{D}\} \subseteq R, \tag{32}$$

where

$$R \triangleq \{x' \mid f((\underline{x}, \overline{x}), u, (\underline{d}, \overline{d})) \leq x' \leq f((\overline{x}, \underline{x}), u, (\overline{d}, \underline{d}))\}. \tag{33}$$

Fig. 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

$$\mathcal{X}_q = \prod_{\ell \in \mathcal{L}} [a_\ell^q, b_\ell^q] \tag{34}$$

as in Fig. 5. We let $a^q = \{a_\ell^q\}_{\ell \in \mathcal{L}}$ and $b^q = \{b_\ell^q\}_{\ell \in \mathcal{L}}$ and assume $\mathcal{D}$ has the form

$$\mathcal{D} = \{d \mid \underline{d} \leq d \leq \overline{d}\} \tag{35}$$

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 reachability result of Theorem 1 justifies our interest in finite abstractions induced by hyperrectangular partitions for mixed monotone systems. In particular, given a hyperrectangular partition of a mixed monotone system, let

$$\begin{aligned} &R_{q,u} \triangleq \\ &\{x' \mid f((a^q, b^q), u, (\underline{d}, \overline{d})) \leq x' \leq f((b^q, a^q), u, (\overline{d}, \underline{d}))\}. \end{aligned} \tag{36}$$

Then, by Theorem 1, $R_{q,u}$ satisfies (24). The following theorem now follows readily:

Theorem 2 ([32, Theorem 2])**.** *Consider the mixed monotone system* (14) *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* (25) *with* $R_{q,u}$ *given by* (36). *Then* $\mathcal{T} = (\mathcal{Q},\mathcal{U},\delta)$ *is a finite state abstraction of* (14)*.*

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 (25). See [32] for details regarding the computational requirements of obtaining a finite state abstraction from a hyperrectangular partition using Theorem 2.

C. Mixed Monotonicity in Traffic Networks

We return to the traffic network dynamics (7)–(8). 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 [31], [32] for conditions on $\alpha_{\ell k}$, $\beta_{\ell k}$, $\Phi^{\text{out}}_\ell(x_\ell)$ and $\Phi^{\text{in}}_\ell(x_\ell)$ 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 $\Phi^{\text{in}}_\ell(x)$ and $\Phi^{\text{out}}_\ell(x)$ are piecewise linear in [31, Theorem 1], and a more general case in [32, Proposition 3]. The proofs of [31, Theorem 1] and [32, Proposition 3] demonstrate sign constancy of the entries of the Jacobians $\partial F/\partial x$ and $\partial F/\partial d$ and then apply Proposition 3.

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 (8) 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 Φ^{in}_k is a decreasing function.

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], [63], and it is a feature of traffic flow that has been observed even on wide freeways with many lanes, [64], [65]. Because $\frac{\partial F_\ell}{\partial x_k}(x) \leq 0$ for ℓ, k at a diverging junction, we have that 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, [66], [67], which ensures that the resulting dynamics are monotone but do not exhibit the FIFO property.

D. Abstractions from Piecewise Linearity

We return to the special case for which the supply and demand functions are piecewise linear, resulting in the piecewise affine dynamics (13). We have already seen that we may construct a finite state abstraction by exploiting the mixed monotonicity of the dynamics. However, as noted above, by overapproximating one-step reachable sets, we introduce conservatism in the abstraction. We now propose an alternative abstraction technique that relies on the piecewise affine dynamics.

For piecewise affine dynamical systems, we may 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 Fig. 5. From this observation, we propose a modification to the above finite abstraction as developed in [23].

We recall the formulation for the piecewise affine case for which we identified a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ such that each $\mathcal{X}_q$ is a polytope and the dynamics are affine in $\mathcal{X}_q$. To construct a finite state abstraction, we again start with a partition of the domain $\mathcal{X}$. For convenience of notation, we consider the same partition $\mathcal{P}$ induced by the dynamics, however we may easily 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 $q' \in \mathcal{Q}'$. We consider the same finite input set $\mathcal{U}$ as before and now assume $\mathcal{D}$ is an arbitrary polytope. We then compute the exact one-step reachable set $R^{\text{exact}}_{q,u}$ for each $q \in \mathcal{Q}$ and $u \in \mathcal{U}$ using polyhedral operations, and then determine the set $\delta(q, u) \triangleq \{q' \mid \mathcal{X}_{q'} \cap R^{\text{exact}}_{q,u} \neq \emptyset\}$, completing the 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, we may consider arbitrary polyhedral partitions $\mathcal{P}$, as well as consider $\mathcal{D}$ 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 a number of 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 concrete system [68], [69], 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

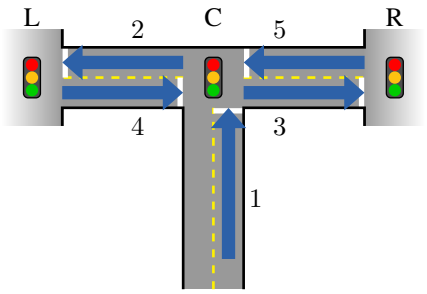


Fig. 9. Example network with three signalized intersections and 5 links, denoted with arrows and numbered as shown. Links 1, 4, and 5 direct exogenous traffic onto the network, and at each time step, the exogenous arrivals are assumed to be within the disturbance set $\mathcal{D}$. If the queues on links 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. We synthesize a control strategy that satisfies the LTL formula in (41).

exact reachable sets eliminate one-step spurious trajectories, more general spurious trajectories remain, as discussed in Section III-B. Furthermore, as mentioned above, we may easily modify the abstraction algorithm for mixed monotone systems to allow $\mathcal{D}$ to be a union of hyperrectangles and thus can suitably approximate more general disturbance sets. Finally, we reiterate that this alternative abstraction approach requires piecewise affine dynamics, while the mixed monotone approach applies to a much more general class of systems.

VI. Case Study

As a case study, we consider the network in Fig. 9 with five links, $\mathcal{L} = \{1, 2, 3, 4, 5\}$, and three signalized intersections which we denote by "L", "C", and "R" for the left, center, and right intersections as they appear in Fig. 9. The signal in the center either 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) 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 eight 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, we assume the disturbance $d[t] = [d_1\ d_2\ d_3\ d_4\ d_5]^T[t]$ is such that

$$d[t] \in \mathcal{D} \triangleq \{d \mid 0 \leq d \leq [15\ 0\ 0\ 0\ 0]^T\} \tag{37}$$
$$\cup\, \{d \mid 0 \leq d \leq [0\ 0\ 0\ 15\ 15]^T\}, \tag{38}$$

for all $t \geq 0$, 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. We 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

$$\Phi_\ell^{\text{out}}(x_\ell) = \min\{x_\ell, c_\ell\} \tag{39}$$
$$\Phi_\ell^{\text{in}}(x_\ell) = x_\ell^{\text{cap}} - x_\ell \tag{40}$$

where we assume $c_\ell = 20$ is the saturation flow for all ℓ. We see that, akin to Example 1, 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.

We wish to find a traffic signal control strategy so that the closed loop dynamics satisfy the following LTL objective:

$$\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \tag{41}$$

where

$$\varphi_1 = \square\lozenge(\text{left signal is "red"}) \tag{42}$$
$$\varphi_2 = \square\lozenge(\text{right signal is "red"}) \tag{43}$$
$$\varphi_3 = \lozenge\square\left(\bigwedge_{i\in\{1,4,5\}} (x_i \leq 30)\right) \tag{44}$$
$$\varphi_4 = \square\big((x_2 > 30 \vee x_3 > 30) \Longrightarrow \lozenge(x_2 \leq 10 \wedge x_3 \leq 10)\big). \tag{45}$$

We interpret (42)–(45) 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, we first obtain a hyperrectangular partition of the state space by introducing a gridding of $\mathcal{X} = \prod_{\ell\in\mathcal{L}}[0, x_\ell^{\text{cap}}] \subset \mathbb{R}^5$. Specifically, we divide $[0, x_\ell^{\text{cap}}]$ into the following sets of intervals:

$$\mathbb{Q}_\ell = \{[0,15], (15,20], (20,25], (25,30], (30,35], (35,40]\}, \quad \ell \in \{1,4,5\} \tag{46}$$
$$\mathbb{Q}_\ell = \{[0,10], (10,20], (20,30], (30,40]\}, \quad \ell \in \{2,3\}. \tag{47}$$

Then we take

$$\mathcal{Q} = \prod_{\ell\in\mathcal{L}} \mathbb{Q}_\ell \tag{48}$$

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

$$\mathcal{X}_q = q^1 \times q^2 \times q^3 \times q^4 \times q^5 \subseteq \mathbb{R}^5. \tag{49}$$

The resulting transition system has $|\mathcal{Q}| = \prod_{\ell\in\mathcal{L}} |\mathbb{Q}_\ell| = 3\,456$ states. Building the transition system using the mixed monotone properties of the dynamics takes 35.2 seconds on a standard laptop. 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 $\mathcal{U}$, we must augment our

transition system to include the last applied input as a state variable; the details are omitted but are straightforward and the procedure may be found in [31]. The final transition system that models the behavior of our concrete system then has $8 \times 3\,456 = 27\,648$ states.

The LTL specification is transformed into a Rabin automaton with 29 modes and one acceptance pair using the `ltl2dstar` tool [56]. Solving the resulting Rabin game takes 42.8 minutes on a standard laptop and results in a control strategy such that the specification is satisfied from any initial condition. We plot a resulting trace of the traffic network dynamics in Fig. 10. To produce the traces, a random disturbance input is synthesized satisfying (37) for which larger disturbances were favored.

Fig. 10(a) shows a resulting trace using the synthesized, correct-by-design control strategy. We see 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 which depend on the particular realization of the disturbance input.

In Fig. 10(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 Fig. 10.

In principle, we could use the alternative abstraction method that relies on the piecewise affine dynamics. However, even with the relatively modest state space dimension of $|\mathcal{L}| = 5$, computing the finite state abstraction would be cost prohibitive as 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 [31]. Ongoing research for further scalability is discussed in the next section.

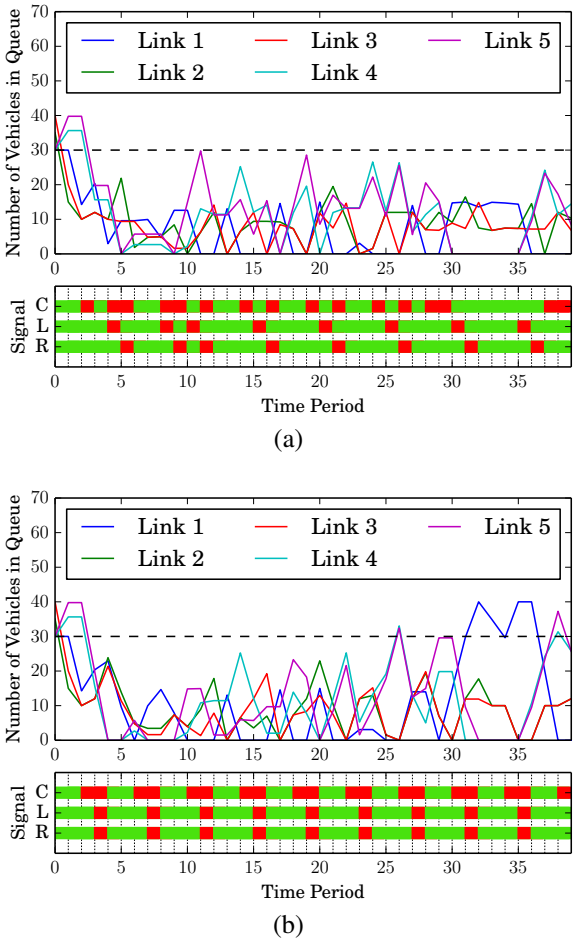


Fig. 10. Sample trajectories of the network in Figure 9. (a) The applied control at each time step is determined by the correct-by-design control strategy obtained using the formal abstraction and synthesis approach presented in this article. As suggested by the sample trajectory, the closed-loop behavior of the concrete system is guaranteed to satisfy the LTL formula (41). (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 the previous case. It is apparent that the LTL formula (41) is not satisfied for this naïve controller.

VII. Conclusions

In this tutorial paper, we have described a formal methods approach to control of traffic flow networks. First, we considered a dynamical model that captures important traffic flow phenomena such as blocked flow due to congestion. Several simplifying assumptions were made to arrive at this model; for example, we adopt a "single commodity" perspective 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. Developing traffic flow models that are simple enough for computation and analysis yet capture required physical considerations is an important area of future research.

Next, we 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 concrete system. This approach ensures that a controller synthesized for the abstraction guarantees that the concrete 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 includ-

ing 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 concrete 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. In this tutorial paper, we have 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, and 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 (46) is the relatively large interval $[0, 15]$. Other approaches include avoiding partitioning the state space altogether [70], [71], an idea closely related to ℓ-complete approximations [11], [16], [17].

Another important 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 [49], [72]. One successful approach is the *assume-guarantee* framework [73] 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. 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 [74].

A key thesis of this tutorial paper is that in order to obtain tractable and scalable formal methods for physical systems, the underlying structure of the systems must be identified and exploited. We have 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, we were able to develop general theory and algorithms that are more broadly applicable.

REFERENCES

[1] A. A. Kurzhanskiy and P. Varaiya, "Traffic management: An outlook," *Economics of Transportation*, 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," tech. rep., Texas Transportation Institute and Inrix, Inc., 2015.

[4] E. Haghverdi, P. Tabuada, and G. J. Pappas, "Bisimulation relations for dynamical, control, and hybrid systems," *Theor. Comput. Sci.*, vol. 342, pp. 229–261, Sept. 2005.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] J. Fu and U. Topcu, "Probably approximately correct MDP learning and control with temporal logic constraints," *arXiv preprint arXiv:1404.7073*, 2014.

[29] 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.

[30] 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.

[31] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications," *IEEE Transactions on Control of Network Systems*, 2015. Accepted for publication, arXiv:1408.1437.

[32] 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.

[33] S. Coogan and M. Arcak, "A compartmental model for traffic networks and its dynamical behavior," *IEEE Transactions on Automatic Control*, pp. 2698–2703, 2015.

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

[35] 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.

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

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

[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, 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] 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.

[45] Transportation Research Board, "Highway capacity manual," *Washington, DC*, 2000.

[46] 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.

[47] 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.

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

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

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

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

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

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

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

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

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

[57] 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.

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

[59] 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.

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

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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

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

[69] 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.

[70] 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.

[71] 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.

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

[73] O. Grumberg and D. E. Long, "Model checking and modular verification," *ACM Trans. Program. Lang. Syst.*, vol. 16, pp. 843–871, May 1994.

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