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. Yet challenges, particularly in scalable abstraction of physical systems, have impeded the applicability of such tools. This tutorial paper exploits structural properties in such networks to overcome some of these challenges and points towards new directions of research. The focus is on synthesis of finite memory controllers to satisfy control objectives expressed in linear temporal logic (LTL). The paper reviews recent literature and presents a compartmental traffic flow model that 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. This overapproximation is suitable for control synthesis posed as a two-player game in which the controller seeks to satisfy the LTL objective and an adversary seeks the opposite. The game formulation utilizes a Rabin automaton to encode the LTL objective. These results are demonstrated on a case study which leads to a discussion of open problems and avenues for future research.**

I. Introduction

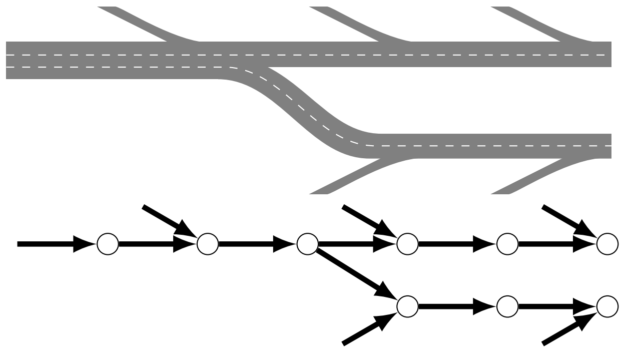
A. Inefficient Traffic Management Is Pervasive

Increased urban congestion requires efficient use of existing transportation infrastructure, yet inefficient traffic management is pervasive. Over ninety percent of the 300,000 signalized intersections in the United States are regulated by pre-timed controllers whose timing plans determine how well they operate. Surveys of practitioners suggest that only sixty percent of traffic signals are retimed at intervals less than five years [1], and the National Transportation Operations Coalition has given a grade of "C−" to signal timing practices and a grade of "F" to traffic monitoring and data collection in the United States. This inefficiency leads to congestion, the cost of which has increased five-fold in the past three decades to \$120 billion annually and include 5.5 billion hours of additional travel time and 2.9 billion gallons of wasted fuel [2].

Today's increasingly populous cities will require intelligent transportation systems that mitigate this inefficiency, and 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. These approaches should leverage recent advances in formal verification and synthesis of control systems to provide automated tools that guarantee safety and improve mobility. However, to ensure these approaches are scalable, reliable, and adaptable, future research must identify and exploit intrinsic structural properties found in transportation systems.

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


(a)

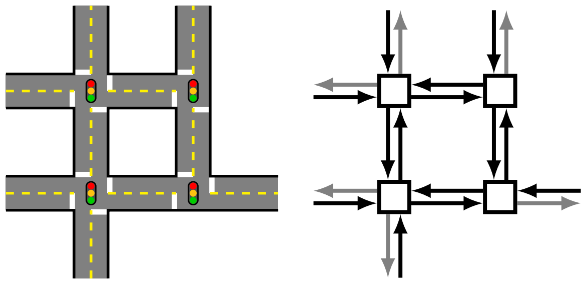

(b)

Fig. 1. 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.

B. Formal Methods For Control of Networked Systems

Control applications 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 of behavior such as "fairness" properties ensuring that, *e.g.*, every time an action is requested by a user it is eventually performed.

For at least the past decade, researchers from the control theory and computer science communities have sought to close the gap between control theoretic tools for controlling complex physical systems and formal methods for accommodating complex specifications of system behavior. A full review of this extensive and rapidly growing literature is beyond the scope of the present paper, but we highlight some particularly relevant work in this domain. Certain classes of systems allow finite *bisimulations* such that the dynamics may be exactly, in a particular sense, represented by a finite state abstraction [3], [4], or are amenable to a related notation of approximate bisimilarity [5], [6], [7], [8]. Other work focuses on model predictive control formulations [9], [10]. Applications and special cases include robotic path planning [11], [12], [13], switched continuous time system [14], and piecewise linear systems [15], [16]. Closely related to the notion of formal methods in dynamical systems is the idea of obtaining a finite state abstraction to approximate the dynamic behavior of the underlying physical system. To this end, in addition to the literature above, a body of literature focuses on computation of finite abstractions [17], [18], [19], [20], [21], [22] and are particularly relevant to the ideas presented in this paper.

C. Paper Outline

The aim of this tutorial paper is two-fold: first, to define and characterize a general model for vehicular traffic flow networks amenable to formal control synthesis; second, 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. We focus in particular on structure in the dynamical system that allows efficient finite state abstraction, and we specialize the synthesis results to traffic flow networks, which exhibit considerable structure.

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-B to guarantee that the behavior of the abstraction and the underlying dynamical system satisfies an objective given in LTL. This approach poses controller synthesis as a game between a controller and an adversary that resolves any nondeterminism present in the abstraction. The synthesis algorithm relies on automata theory and fixed point algorithms to compute a finite memory control strategy for the system.

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* [23], [24], a generalization of monotone dynamical systems [25], [26], [27]. 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 $t = 0, \dots, n$ 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 (iff) $x_i \leq y_i$ for $i = 1, \dots, n$, 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. A compartmental model 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 than considering movement of individual vehicles. This model is general and, in particular, encompasses the Cell Transmission Model of freeway traffic flow [28], [29] and queue forwarding models as in [30] where we further account for the finite capacity of queues. The model presented here is based on the model developed in [31], [32].

A. General Model

A traffic network consists of a set of *links* $\mathcal{L}$ interconnected at a set of *nodes* $\mathcal{V}$ as in Figure 1. In freeway networks, the nodes represent junctions where, *e.g.*, onramps enter, offramps exit, two freeways merge, a freeway diverges to two freeways, *etc.*, 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 $\tau(\ell) = \epsilon$ serve to direct exogenous flow onto the network.

We assume time is discrete, although the compartmental model easily extends to continuous time; see [33], [34]. The state of link $\ell \in \mathcal{L}$ at discrete time t is denoted by

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

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 key insight of queue forwarding models and the CTM 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 assuption for each $\ell \in \mathcal{L}$:

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

Prototypical demand and supply functions are shown in Figure 2. 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 in one time step. 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 with $\sigma(\ell) = v$ 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 (each subset is called a *phase* for the intersection), but this is not required in our formulation.

An important element of modeling transportation networks is determining a routing policy for junctions with multiple incoming and/or outgoing links that captures observed 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* $\beta_{\ell k}$ 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

$$\sum_{k \in \mathcal{L}} \beta_{\ell k} \leq 1 \tag{4}$$

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 via, *e.g.*, unmodeled roads or driveways.

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

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

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

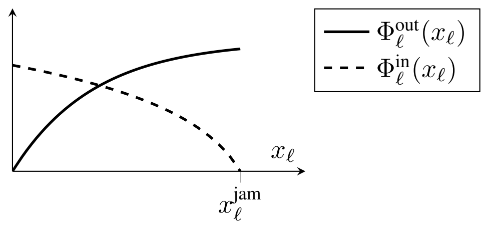


Fig. 2. Plot of prototypical supply and demand functions $\Phi_\ell^{\text{in}}(x_\ell)$ and $\Phi_\ell^{\text{out}}(x_\ell)$.

at some upper limit; see [35] 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

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

that is, the flow exiting link ℓ is the maximum such that neither demand of link ℓ nor downstream supply is violated. Conservation of mass completes the model:

$$\begin{aligned} x_\ell[t+1] &= F_\ell(x[t],u[t],d[t]) \qquad (7)\\ &\triangleq x_\ell[t] - f_\ell^{\text{out}}(x[t],u[t]) + \sum_{k\in\mathcal{L}} \beta_{k\ell} f_k^{\text{out}}(x[t],u[t]) + d_\ell[t] \qquad (8)\end{aligned}$$

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.

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,

$$\Phi_\ell^{\text{out}}(x_\ell) = \min\{v_\ell x_\ell, q_\ell^{\max}\} \tag{9}$$

$$\Phi_\ell^{\text{in}}(x_\ell) = w_\ell(x_\ell^{\text{cap}} - x_\ell) \tag{10}$$

for constants $v_\ell > 0$, $w_\ell > 0$, and $q_\ell^{\max} > 0$. When (9) and (10) are assumed for freeway networks, v_ℓ and w_ℓ are the *free flow speed* and *congested wave speed*, respectively [36]. For signalized networks, it is common to interpet x_ℓ as the queue length and take $v_\ell = w_\ell = 1$, and $q_\ell^{\max}$ is the *saturation flow rate* [37] so that Φ_ℓ^{out} is the minimum of the queue length x_ℓ and the saturation flow rate, and Φ_ℓ^{in} is the unoccupied queue capacity of link ℓ.

When the demand and supply functions have the form (9)–(10), 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

$$F(x,u,d) = A_{q,u}x + b_{q,u} + d \quad \forall x \in \mathcal{X}_q \tag{11}$$

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 in each polyhedral partition. The polytopes arise from the $\min\{\cdot\}$ functions in (6) and (9). We will see in Section V-D that we are able to construct a finite state abstraction using the tools in [15], [38] that exploit the piecewise affine nature of the dynamics, however the computational complexity significantly increases.

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 given 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 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 Section IV.

We consider a discrete-time dynamical system of the form

$$x[t+1] = F(x[t],u[t],d[t]) \tag{12}$$

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 in Section II 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 Abstraction

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

$$\pi_{\mathcal{P}} : \mathcal{X} \to \mathcal{Q} \tag{13}$$

$$\pi_{\mathcal{P}}(x) = q \text{ for which } x \in \mathcal{X}_q \tag{14}$$

be the well-defined projection map from $\mathcal{X}$ to $\mathcal{Q}$. For a trajectory $x[\cdot]$ of the dynamical system (12), we let $\pi_{\mathcal{P}}(x[\cdot])$ denote the sequence $q[0]q[1]q[2]\cdots$.

Definition 1 (Finite state abstraction)**.** Given a partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ of $\mathcal{X}$ for system (12), the tuple $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ with $\delta : \mathcal{Q}\times\mathcal{U} \to 2^{\mathcal{Q}}$ is a *finite state abstraction* of (12) if

$$\forall x \in \mathcal{X}, \forall d \in \mathcal{D} \qquad x \in \mathcal{X}_q \text{ and } F(x,u,d) \in \mathcal{X}_{q'} \text{ implies } \quad q' \in \delta(q,u) \tag{15}$$

for any $q, q' \in \mathcal{X}$, $u \in \mathcal{U}$. We call δ the *transition map*. ■

A finite state abstraction of the concrete dynamical system (12) 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.

Definition 2 (Execution)**.** An *execution* of the finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ 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 $t \geq 0$. ■

A few remarks are in order. First, it is important to note the direction of implication in the definition of (15). In particular, it may be that $q' \in \delta(q, u)$ 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.

Often, $\delta(q, u)$ is assumed to be the smallest transition map satisfying (15) where "smallest" is with respect to set inclusion. That is, it is assumed that $\delta(q, u)$ does not contain spurious one-step transitions and thus satisfies (15) when the implication is replaced with the biconditional "if and only if." For this case, $\mathcal{T}$ is then 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 (12), 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. We focus on this important aspect of the problem in Section V, and as we will see there, it is possible to efficiently compute a finite state abstraction from the overapproximating reachable sets, even though this approximation may contain spurious one-step transitions.

Even in the absence of spurious one-step transitions in the map δ, there still may exist spurious executions of the finite state abstraction that do not correspond to any trajectory of the concrete system. Therefore, a degree of conservatism is inevitably present when formal synthesis is applied to a finite abstraction.

Definition 3 (Spurious sequence and execution)**.** Given a finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ of a concrete system (12). 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 $t = n, \dots, m$. 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 1. Consider a system for which $\mathcal{X} \subseteq \mathbb{R}^2$ is a rectangle as in Figure 3, $\mathcal{P} = \{\mathcal{X}_q\}_{q \in \mathcal{Q}}$ 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 (12) 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 Figure 3 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 Example 1, we could obtain an abstraction that does not exhibit the particular spurious sequence in this example. For quotient based abstractions, standard iterative partition refinement algorithms exist [39].

The second approach, which traces its roots to behavioral systems theory [40], is to compute an *ℓ-complete abstraction* given the fixed partition $\mathcal{P}$ that incorporates finite memory to track past behavior of the system; the ℓ refers to the length of the memory, and increasing ℓ reduces the conservatism (*i.e.*, spurious executions) of the abstraction [17], [21]. For example, in an ℓ-complete abstraction with $\ell = 2$, the currently and previously occupied partition constitute an expanded state of a finite state abstraction, and this expanded state is considered when constructing the transition map. In Example 1, 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 [22] 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 distinct 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

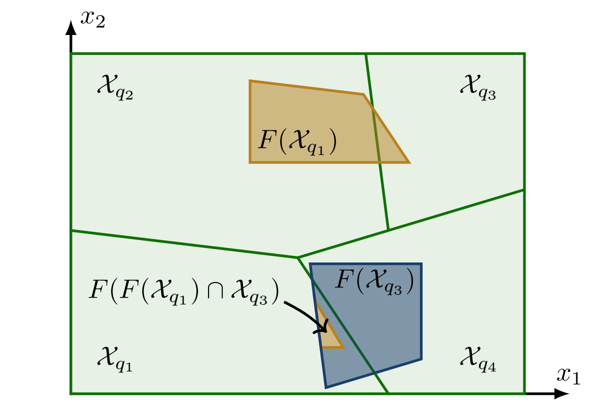


Fig. 3. Spurious executions of finite state abstractions. Since $F(\mathcal{X}_{q_1})$ intersects $\mathcal{X}_{q_2}$ and $\mathcal{X}_{q_3}$, we have that $\{q_2, q_3\} \subseteq \delta(q_1)$, and likewise $\{q_1, q_4\} \subseteq \delta(q_3)$ so that the sequence $q_1q_3q_4$ consists of valid transitions 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.

one-step reachable set from a partition may intersect multiple partitions, as in Example 1.

Despite the nondeterminism and existence of spurious transitions, an overapproximating finite state abstraction is suitable for control synthesis [39], [41]; as we will see in Section IV, a controller obtained from analysis of the finite state abstraction can be appropriately applied to the original concrete system with the same guaranteed performance. In particular, condition (15) implies that the concrete system is an alternating simulation [39] of the finite state abstraction $\mathcal{T}$ and that there exists a feedback refinement [41] from the concrete system to the abstraction. Alternating simulation relations and feedback refinement relations are both sufficient for control synthesis, although it has recently been observed that existence of a feedback refinement relation implies existence of *symbolic* controllers, meaning that the synthesized controller only relies on feedback of the abstract states $\mathcal{Q}$, while alternating simulations may require feedback of the full state [41]. That is, in our setting, a symbolic controller only requires knowledge of the currently occupied partition q at time t and not $x[t]$, as we see in the following section.

IV. Formal Controller Synthesis From Abstractions

We now present a versatile approach for automatically synthesizing a control strategy for a dynamical system so that the resulting closed-loop behavior satisfies a specification given in *linear temporal logic (LTL)*. LTL is an extension of propositional logic that allows for temporal modalities. The expressive power of LTL captures many objectives relevant for control of transportation networks and other physical systems.

The synthesis approach presented here relies on a finite state abstraction of the underlying continuous system that overapproximates the system's dynamics as detailed 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 adversarially resolve any nondeterminism 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) formulae are used to describe the temporal behavior of systems [42], [43], [44]. For example, we may specify desired reachability behavior as "the system eventually enters operating condition a and remains in this operating condition for all future time." To formalize the specification of such behavior, LTL formulae comprise: a finite set of *atomic propositions*, denoted by $\mathcal{O}$; the standard Boolean connectives such as $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), and $\implies$ (implication); and temporal modalities including $\bigcirc$ ("next"), U ("until"), $\square$ ("always"), and $\lozenge$ ("eventually").

A LTL formula is said to be *over* the set of atomic propositions $\mathcal{O}$ and is often denoted by φ. For example, suppose $\mathcal{O} = \{a, b\}$, that is, we consider LTL formula over the two atomic propositions a and b where, *e.g.*, proposition a is true if the system is in a particular operating mode, and likewise for b. Below are examples of LTL formula and their intuitive meanings:

- $\varphi = \square a$ "a is true now and for all future time"
- $\varphi = \lozenge a$ "eventually a becomes true"
- $\varphi = \lozenge\square a$ "eventually a becomes true and remains true for all future time"
- $\varphi = \square\lozenge a$ "infinitely often a is true"
- $\varphi = a \,\mathsf{U}\, b$ "b becomes true at some time in the future and until then a is true"
- $\varphi = \bigcirc\varphi_1$ "the LTL formula φ_1 holds at the next time step" where φ_1 may be, *e.g.*, any formula above.

We now formalize these intuitive meanings by defining the semantics of LTL formulae. LTL formulae are interpreted over infinite sequences of atomic propositions; as we will see in the sequel, we will connect atomic propositions to a concrete system and its finite state abstraction as defined in Section III by assuming that a fixed subset of atomic propositions holds for each $q \in \mathcal{Q}$, equivalently, for each $x \in \mathcal{X}_q$.

To this end, consider a sequence $\sigma = \sigma[0]\sigma[1]\sigma[2]\cdots$ where each $\sigma[t] \subseteq \mathcal{O}$. We use the notation $\sigma[j\cdots] \triangleq \sigma[j]\sigma[j+1]\cdots$ for $j \geq 0$ to denote the infinite sequence that is equivalent to σ starting at position j. We write $\sigma \models \varphi$ to mean that σ *satisfies* φ, where satisfaction is defined inductively as follows [44]:

- $\sigma \models a$ iff $a \in \sigma[0]$
- $\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$
- $\sigma \models \neg\varphi$ iff not $\sigma \models \varphi$
- $\sigma \models \bigcirc\varphi$ iff $\sigma[1\cdots] = \sigma[1]\sigma[2]\cdots \models \varphi$
- $\sigma \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists $j \geq 0$ such that $\sigma[j\cdots] \models \varphi_2$ and $\sigma[i\cdots] \models \varphi_1$ for all $0 \leq i < j$.

From this induction, we derive the expected meaning of $\Diamond\varphi \triangleq \text{true} \,\mathsf{U}\, \varphi$ and $\Box\varphi \triangleq \neg\Diamond\neg\varphi$:

- $\sigma \models \Diamond\varphi$ iff there exists $j \geq 0$ such that $\sigma[j \cdots] \models \varphi$.
- $\sigma \models \Box\varphi$ iff for all $j \geq 0$ $\sigma[j \cdots] \models \varphi$.

We similarly derive more complex modalities such as $\Box\Diamond$ (infinitely often) and $\Diamond\Box$ (eventually always).

We now return to finite state abstractions of dynamical systems and define a corresponding notion of LTL satisfaction.

Definition 4 (Labeled systems)**.** We say the dynamical system (12) 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 $x \in \mathcal{X}_q$. When clear from context, we use $L(\cdot)$ to denote either labeling function. ■

Definition 5 (Trace)**.** Consider a trajectory $x[\cdot]$ of a labeled dynamical system (12) 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 (12) with partition $\mathcal{P}$. Equation (15) guarantees that, for a trajectory $x[\cdot]$ of the dynamical system (12) generated by the input sequence $u[\cdot]$, the pair $\pi_{\mathcal{P}}(x[\cdot])$, $u[\cdot]$ is an execution of $\mathcal{T}$. It follows immediately that

$$L(x[\cdot]) = L(\pi_{\mathcal{P}}(x[\cdot])). \tag{16}$$

We now naturally define LTL satisfaction for trajectories of dynamical systems and executions of finite state abstractions.

Definition 6 (LTL satisfaction for trajectories and executions)**.** A trajectory $x[\cdot]$ of a dynamical system (12) with partition $\mathcal{P}$ satisfies φ if $L(x[\cdot]) \models \varphi$, for which we write $x[\cdot] \models \varphi$. An execution $q[\cdot]$ of a finite state abstraction satisfies an LTL specification φ if $L(q[\cdot]) \models \varphi$, for which we write $q[\cdot] \models \varphi$. ■

B. Controller Synthesis from Rabin Games

We consider the labeled dynamical system $x[t+1] = F(x[t], u[t], d[t])$ as in (12) with atomic propositions $\mathcal{O}$ 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 $x[\cdot] \models \varphi$. 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 below 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 7 (Control strategy)**.** A feedback *control strategy* γ for a labeled finite state abstraction $\mathcal{T}$ is a map

$$\gamma : (2^{\mathcal{O}})^{+} \to \mathcal{U} \tag{17}$$

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 and a set of initial conditions $\mathcal{Q}_0 \subseteq \mathcal{Q}$ for the finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$ with LTL objective φ so that, for any execution $q[\cdot]$, $u[\cdot]$ satisfying $q[0] \in \mathcal{Q}_0$ and

- $q[t+1] \in \delta(q[t], u[t])$ for all t,
- $u[t] = \gamma(q[0]q[1]\cdots q[t])$ for all t,

it is the case that $q[\cdot] \models \varphi$. ■

We see that in our control synthesis objective, 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, for the synthesis algorithm we employ, we are able to identify 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 8 (Finite memory control strategy)**.** The control strategy γ is said to be *finite memory* if there exists a tuple (M, m_0, Δ, g) with

- 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 γ 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 (*i.e.*, 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. For finite memory controllers, we make the identification $\gamma = (M, \Delta, g, I)$. As we will see below, finite memory controllers suffice for our purposes.

To make the connection between LTL formulae and finite memory controllers, we introduce *Rabin automata* that are capable of *accepting* infinite traces. Rabin automata serve two key purposes. First, there exists 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 exists an algorithm 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 9 (Deterministic Rabin automaton)**.** A deterministic *Rabin* automaton is a tuple $(M, m_0, \mathcal{O}, \Delta_R, F)$ where:

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

An execution of a deterministic Rabin automaton is defined analogously to Definition 2. The input sequence $\sigma[\cdot] = \sigma[0]\sigma[1]\sigma[2]\cdots$ with $\sigma[t] \in 2^{\mathcal{O}}$ 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 8 and Definition 9 are intentional, as we will see subsequently. As all Rabin automata in this paper are deterministic, we drop the "deterministic" modifier.

Our interest in Rabin automata stems from the following well-known result:

Proposition 1 ([45], [46], [47])**.** *Given an LTL formula φ over the set of atomic propositions $2^{\mathcal{O}}$, let $\sigma \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$. There exists a deterministic Rabin automaton such that*

$$\sigma \models \varphi \quad \textit{iff} \quad \sigma \textit{ is accepted by the Rabin automaton.} \tag{18}$$

Example 2. Assume $\mathcal{O} = \{a, b\}$ and consider the LTL formula $\varphi = \Box(a \to \Diamond b)$ which is satisfied if whenever a holds, it is the case that at some future time b will hold. Consider the Rabin automaton $(M, m_0, \mathcal{O}, \Delta_R, F)$ 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:

$$\Delta_R(m_0, W) = m_1 \text{ iff } a \in W, \tag{19}$$

$$\Delta_R(m_1, W) = m_0 \text{ iff } b \in W. \tag{20}$$

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 iff $\sigma \models \varphi$, see Figure 4. Indeed, F implies that an execution $m[t]$ is accepted iff $m[t] = m_0$ for infinitely many $t \geq 0$. Reasoning about Δ_R as given in (19)–(20), this is the case iff whenever $a \in \sigma[t]$ there exists some time $\tau > t$ for which $b \in \sigma[t]$, that is, iff $\sigma \models \varphi$. ■

Moreover, there exists algorithms [46] and readily available software [48] for constructing a Rabin automaton from a LTL formula. An important difficulty is that, for the worst case, the number of Rabin modes $|M|$ is doubly exponentially as large as the length of the LTL formula

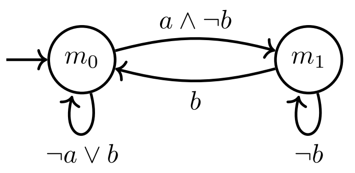


Fig. 4. Example of a Rabin automaton that corresponds to the LTL specification $\varphi = \Box(a \to \Diamond b)$ for $\mathcal{O} = \{a, b\}$. We have $F = \{(m_0, \emptyset)\}$, thus a trace is accepted iff m_0 is visited infinitely often. The edge labeled b from m_1 to m_0 indicates that if $m[t] = m_1$ and the input $\sigma[t] \in 2^{\mathcal{O}}$ is applied at time t, then the Rabin automaton will transition to m_0 iff $b \in \sigma[t]$, and similarly for the other edges. Reasoning about the automaton's operation, we see that $\sigma \in (2^{\mathcal{O}})^{\mathbb{Z}_{\geq 0}}$ is accepted (*i.e.*, visits m_0 infinitely often) iff $\sigma \models \varphi$.

where the length is in terms of the number of operators [46]. 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 $q[\cdot], u[\cdot]$ and an associated trace $L(q[\cdot])$. To check whether $q[\cdot] \models \varphi$, 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 (9).

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 atomic propositions 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 $q[\cdot] \models \varphi$.

Indeed, the result is a *Rabin game* that is played as follows: a controller (also called the *protagonist* or *scheduler* [44]), 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* resolves the nondeterminism of the transition map $\delta(q[t], u[t])$, 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 [49], [50], [51]. 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 [52]. That is, the result of these algorithms is a function $g : \mathcal{Q} \times M \to \mathcal{U}$ and a set $\mathcal{Q}_0$ such that, if the finite state abstraction is initialized with $q[0] \in \mathcal{Q}_0$, then by choosing $u[t] = g(q[t], m[t])$ at each time instant, the controller is guaranteed to win the Rabin game no matter the choice of the adversary, and thus

$q[\cdot] \models \varphi$.

To complete the picture, it is then straightforward to characterize γ having the structure as in Definition 8, that is, $\gamma = (M, m_0, \Delta, g)$ with modes M and initial mode m_0 inherited from the Rabin automaton, $\Delta(m, q) = \Delta_R(m, L(q))$, and g obtained via the aforementioned algorithms.

C. Refining Abstraction-Based Controllers

We conclude this section by describing how a control strategy computed from a finite state abstraction, as described in Section IV-B, is *refined* to establish a controller for the original concrete system. The setup is as before, where we consider a dynamical system as in (12) and a finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$. The construction of the refined controller is straightforward:

Definition 10 (Refined control strategy)**.** Given a partition $P = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$, a finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$, and a finite memory control strategy $\gamma = (M, m_0, \Delta, g)$ as in Definition 8. The *refined finite memory control strategy* is the tuple $\overline{\gamma} = (M, m_0, \overline{\Delta}, \overline{g})$ where

- $\overline{\Delta} : M \times \mathcal{X} \to M$ is given by $\overline{\Delta}(m, x) = \Delta(m, \pi_P(x))$
- $\overline{g} : M \times \mathcal{X} \to \mathcal{U}$ is given by $\overline{g}(m, x) = g(m, \pi_P(x))$.

■

As in the case for finite memory control of finite abstractions, we write $\overline{\gamma}(x[0]x[1]x[2]\cdots x[t])$ to mean $\overline{g}(m[t], x[t])$ where $m[t]$ is understood to have been generated according to $\overline{\Delta}$.

The following implies soundness of the synthesis approach:

Proposition 2. *Given the finite memory control strategy γ and a set of initial states $\mathcal{Q}_0 \subseteq \mathcal{Q}$. If $q[\cdot] \models \varphi$ for all executions of $\mathcal{T}$ satisfying $q[0] \in \mathcal{Q}_0$ and $u[t] = \gamma(q[0]q[1]\cdots q[t])$, then $x[\cdot] \models \varphi$ for all trajectories $x[\cdot]$ of the concrete system satisfying $x[0] \in \cup_{q\in\mathcal{Q}_0}\mathcal{X}_q$ and $u[t] = \overline{\gamma}(x[0]x[1]x[2]\cdots x[t])$.*

Proof: The proof follows readily from the overapproximating nature of $\mathcal{T}$ as specified in (15). 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 $u[t] = \overline{\gamma}(x[0]x[1]x[2]\cdots x[t])$ for all $t \geq 0$. It follows that the projected sequence $\pi_P(x[\cdot]) = q[\cdot] = q[0]q[1]q[2]\cdots$ is such that $q[0] \in \mathcal{Q}_0$, $q[t+1] \in \delta(q[t], u[t])$, that is, $q[\cdot]$, $u[\cdot]$ is an execution of $\mathcal{T}$ satisfying $q[0] \in \mathcal{Q}_0$ and $u[t] = \gamma(q[0]q[1]\cdots q[t])$. By hypothesis, we have that $q[\cdot] \models \varphi$, that is, $L(q[\cdot]) \models \varphi$. By (16), we have $x[\cdot] \models \varphi$. ■

The clear 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 Section IV-B. Fig. 5 illustrates the overall control procedure presented here. We now make a few remarks.

First, it is intriguing to observe that the resulting refined controller is *symbolic*, meaning that full state feedback is not required. Indeed, at any time instant, the controller 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 as input the coarse measurement $q[t] = \pi_P(x[t])$ consisting of the currently occupied partition and applies the input as dictated by the finite mapping $g(m[t], q[t])$. 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 relatively minimal 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, as described in Section IV-B, can be costly.

Additionally, it is important to note that for more general finite abstraction techniques that accomodate, *e.g.*, abstractions based on coverings of the domain rather than partitions or continuous input sets, the controller refinement step is not as straightforward and one must take care to preserve soundness. 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 [41], [53].

Finally, while Proposition 2 implies that our synthesis algorithm is sound, it is apparent that it is not complete in the sense that existence of a controller for the concrete system such that $x[\cdot] \models \varphi$ for all trajectories $x[\cdot]$ does not imply existence of a controller for a given, or even any, finite state abstraction. The reasons for this gap are apparent; perhaps the most noteworthy of these is that, as remarked above, $\mathcal{T}$ overapproximates the behavior of the concrete system. While this overapproximation is sufficient for ensuring soundness of our controller refinement procedure when a controller for the abstraction exists, it is entirely possible that our attempt to synthesize a controller from the abstraction fails 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* [39].

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.

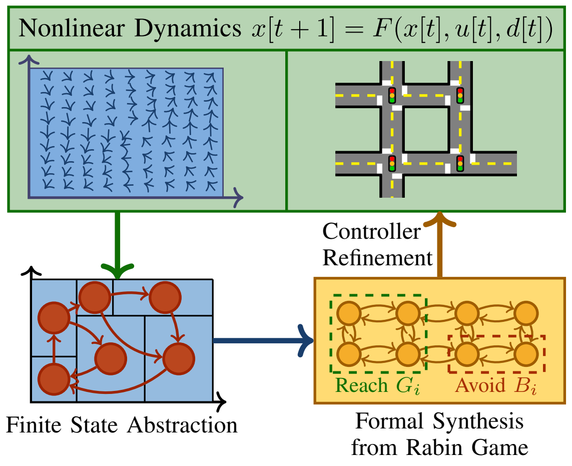

Nonlinear Dynamics $x[t+1] = F(x[t], u[t], d[t])$
Controller Refinement
Reach G_i
Avoid B_i
Finite State Abstraction
Formal Synthesis from Rabin Game

Fig. 5. 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. This controller is then refined for use with the original system.

A. Mixed Monotone Dynamical Systems

We again consider the dynamical system $x[t+1] = F(x[t], u[t], d[t])$ as in (12) and partition $\mathcal{P} = \{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ for which we wish to construct a finite state abstraction $\mathcal{T} = (\mathcal{Q}, \mathcal{U}, \delta)$. Consider computing $\delta(q, u)$ for some $q \in \mathcal{Q}$, $u \in \mathcal{U}$, and suppose we are able to calculate an overapproximation of the one-step reachable set from $\mathcal{X}_q$ under input u, denoted by $R_{q,u}$, that is

$$R_{q,u} \supseteq \{F(x, u, d) \mid x \in \mathcal{X}_q, d \in \mathcal{D}\}. \tag{21}$$

By defining

$$q' \in \delta(q, u) \iff \mathcal{X}_{q'} \cap R_{q,u} \neq \emptyset, \tag{22}$$

we obtain δ so that (15) holds. That is, we may use an overapproximation of the one-step reachable set from each partition for each input to construct a finite state abstraction.

Here, we focus on computing one-step reachable sets for the class of *mixed monotone* systems. Roughly, 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. We then show that traffic flow networks are mixed monotone, allowing us to efficiently compute finite state abstractions of traffic networks.

Definition 11 (Mixed monotone system)**.** The system (12) 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) $\forall x \in \mathcal{X},\ \forall d \in \mathcal{D}$: $F(x, u, d) = f((x, d), u, (x, d))$

C2) $\forall \underline{x}, \overline{x}, y \in \mathcal{X},\ \forall \underline{d}, \overline{d}, e \in \mathcal{D}$: $\underline{x} \leq \overline{x}$ 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) $\forall x, \underline{y}, \overline{y} \in \mathcal{X},\ \forall d, \underline{e}, \overline{e} \in \mathcal{D}$: $\underline{y} \leq \overline{y}$ and $\underline{e} \leq \overline{e}$ implies $f((x, \overline{y}), u, (d, \overline{e})) \leq f((x, \underline{y}), u, (d, \underline{e}))$.

■

Above, recall that $\leq$ is interpreted elementwise, that is, we consider the partial order induced by the positive orthant; mixed monotonicity may be extended to general partial orders of $\mathcal{X}$ and $\mathcal{D}$ [24]. We see that $f((x, y), u, (d, e))$ satisfying C2–C3 is nondecreasing in x and d and nonincreasing in y and e.

Definition 12 (Decomposition function)**.** A function f satisfying C1–C3 above is a *decomposition function* for $F(x, u, d)$. ■

If f is differentiable, then we may replace C2 and C3 with the following conditions:

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

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

Mixed monotone systems are a generalization of the well known class of monotone dynamical systems [25], [26], [54]. Indeed, if $f((x, y), u, (d, e)) = F(x, u, d)$ is a decomposition function, we recover standard characterizations of monotone systems with disturbances; see [27]. 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 $u \in \mathcal{U}$.

Mixed monotone systems are characterized by the existence of a decomposition function, and much like the search for Lyapunov functions for stability analysis, finding a decomposition function is often not straightforward. Below, we characterize a class of mixed monotone systems in terms of the Jacobian matrices $\partial F/\partial x$ and $\partial F/\partial d$ for which a decomposition function is naturally constructed.

Proposition 3 ([24, Proposition 1])**.** *Consider system* (12) *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 $u \in \Sigma$ and for all $i \in \{1, \dots, n\}$*

$$\forall j \in \{1, \dots, n\}$$
$$\exists \mu_{i,j} \in \{-1, 1\} : \mu_{i,j} \frac{\partial F_i}{\partial x_j}(x, u, d) \geq 0\ \forall x, d \tag{23}$$

and

$$\forall j \in \{1, \dots, m\}$$
$$\exists \nu_{i,j} \in \{-1, 1\} : \nu_{i,j} \frac{\partial F_i}{\partial d_j}(x, u, d) \geq 0\ \forall x, d \tag{24}$$

then (4) *is mixed monotone and a decomposition function is given by*

$$f((x, y), u, (d, e)) = \begin{bmatrix} f_1((x, y), u, (d, e)) \\ \vdots \\ f_n((x, y), u, (d, e)) \end{bmatrix}, \tag{25}$$

$$f_i((x, y), u, (d, e)) \triangleq F_i(z^i(x, y), u, w^i(d, e)) \tag{26}$$

where

$$z^i(x,y) = \begin{bmatrix} z_1^i(x,y) & \cdots & z_n^i(x,y) \end{bmatrix}^T, \tag{27}$$

$$z_j^i(x,y) \triangleq \begin{cases} x_j & \text{if } \mu_{i,j} = 1 \\ y_j & \text{if } \mu_{i,j} = -1 \end{cases} \quad \forall j \in \{1,\ldots,n\} \tag{28}$$

and

$$w^i(x,y) = \begin{bmatrix} w_1^i(x,y) & \cdots & w_m^i(x,y) \end{bmatrix}^T, \tag{29}$$

$$w_j^i(x,y) \triangleq \begin{cases} d_j & \text{if } \nu_{i,j} = 1 \\ e_j & \text{if } \nu_{i,j} = -1 \end{cases} \quad \forall j \in \{1,\ldots,m\}. \tag{30}$$

Condition (23)–(24) states that 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.

Additionally, the construction of the decomposition function (25)–(30) 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 $x \in \mathcal{X}, d \in \mathcal{D}$.

Note that Proposition 3 assumed F to be continuously differentiable, however 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. Mixed Monotonicity in Traffic Networks

We now return to the traffic network dynamics of Section II. We first note that the dynamics of (7)–(8) may be viewed as a discretization of an appropriate continuous time model (see [33], [34] for such a model), and under this interpretation, it is natural to assume that $\frac{\partial F_\ell}{\partial x_\ell}(x) \geq 0$ for all $x \in \mathcal{X}$ and all $\ell \in \mathcal{L}$. Indeed, if this were not the case, it is possible to perturb some state x to $\tilde{x}$ by adding a sufficiently small number of vehicles to link ℓ (that is, $\tilde{x}$ and x differ only in that $\tilde{x}_\ell > x_\ell$), for which $F_\ell(\tilde{x},u,d) < F_\ell(x,u,d)$, *i.e*, in the next time step, link ℓ then has *fewer* vehicles, an unreasonable phenomenon for small enough sampling time[1].

Conditions on $\alpha_{\ell k}$, $\beta_{\ell k}$, $\Phi_\ell^{\text{out}}(x_\ell)$ and $\Phi_\ell^{\text{in}}(x_\ell)$ for all ℓ, k ensuring that $\frac{\partial F_\ell}{\partial x_\ell}(x) \geq 0$ for all $x \in \mathcal{X}$ and all $\ell \in \mathcal{L}$ are given in [24], [31]. Such conditions amount to sufficiently small sampling time of the implicit continuous time model.

Theorem 1. *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 1 is proved for the special case when $\Phi_\ell^{\text{in}}(x)$ and $\Phi_\ell^{\text{out}}(x)$ are piecewise linear in [31, Theorem 1], and a more general case in [24, Proposition 3]. The proofs of [31, Theorem 1] and [24, 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.

[1]Equivalently, for a system $\dot{x} = G(x)$ for $G(x)$ bounded and with forward Euler approximation $x[t+1] = F(x[t]) = x[t] + \tau G(x[t])$, we may always choose sampling time τ small enough so that $\frac{\partial F_i}{\partial x_i}(x) \geq 0$ for all x and all i.

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 $\Phi_k^{\text{in}}(x_k)$ 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, [29], [55], and it is a feature of traffic flow that has been observed even on wide freeways with many lanes, [56], [57]. Some of the recent literature in dynamical flow models propose alternative modeling choices for diverging junctions, *e.g.*, [58], [59], which ensures that the resulting dynamics are monotone but do not exhibit this FIFO property.

We comment that, since $\partial F_\ell/\partial d_\ell \geq 0$ and $\partial F_\ell/\partial d_k = 0$ for all $\ell \neq k$, the independent variable e does not appear in the decomposition function $f((x,y),u,(d,e))$ for the traffic network dynamics. We nonetheless continue to develop the theory for the more general case.

C. One Step Reachable Set of Mixed Monotone Systems

For our purposes, one of the most important properties of mixed monotone systems is the following:

Theorem 2. *Let* (12) *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 $\underline{d} \leq \overline{d}$, for all $u \in \mathcal{U}$. Then*

$$f((\underline{x},\overline{x}),u,(\underline{d},\overline{d})) \leq F(x,u,d) \leq f((\overline{x},\underline{x}),u,(\overline{d},\underline{d}))$$
$$\forall x \in \{x \mid \underline{x} \leq x \leq \overline{x}\} \; \forall d \in \{d \mid \underline{d} \leq d \leq \overline{d}\}. \tag{31}$$

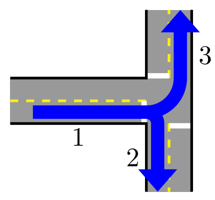
Example 3. Consider the network shown in Fig 6(a) with $\mathcal{L} = \{1,2,3\}$, $\Phi_\ell^{\text{out}}(x_\ell) = \max\{c_\ell, x_\ell\}$ where $(c_1,c_2,c_3) = (20,5,30)$, $\Phi_\ell^{\text{in}}(x_\ell) = 50 - x_\ell$ for all ℓ, $\beta_{12} = \beta_{13} = 1/2$, $\alpha_{12} = \alpha_{13} = 1$, and $\mathcal{D} = \{d \mid \underline{d} \leq d \leq \overline{d}\}$ where $\underline{d} = [0\ 5\ 0]^T$ and $\overline{d} = [0\ 8\ 5]^T$. We assume the intersection is not signalized and thus the input set is a singleton set $\mathcal{U} = \{u\}$, $u = \{1,2,3\}$ indicating flow along all links is allowed. 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

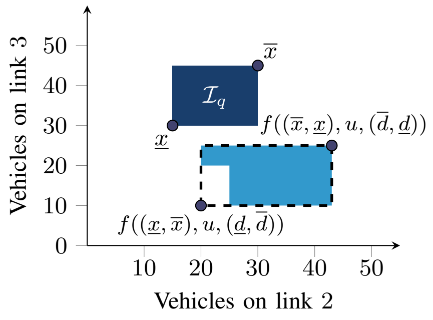
$$f((\underline{x},\overline{x}),u,(\underline{d},\overline{d})) = [20\ 20\ 10]^T \tag{32}$$

$$f((\overline{x},\underline{x}),u,(\overline{d},\underline{d})) = [30\ 43\ 25]^T. \tag{33}$$

Let

$$R \triangleq \{x' \mid f((\underline{x},\overline{x}),u,(\underline{d},\overline{d})) \leq x' \leq f((\overline{x},\underline{x}),u,(\overline{d},\underline{d}))\}. \tag{34}$$





(a) (b)

Fig. 6. 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.

Then, by Theorem 2,

$$\{F(x,u,d)\mid x\in\mathcal{I}_q\ d\in\mathcal{D}\}\subseteq R. \tag{35}$$

Fig. 6(b) plots $\mathcal{I}_q$, R, and the actual reachable set projected in the x_2 vs. x_3 plane.

■

Definition 13 (Interval partition)**.** The partition $\mathcal{P}=\{\mathcal{X}_q\}_{q\in\mathcal{Q}}$ is an *interval partition* if each $\mathcal{X}_q$ is a hyperrectangle, that is, for all $q\in\mathcal{Q}$ there exists intervals $\{\iota_\ell^q\}_{\ell\in\mathcal{L}}$ where each ι_ℓ^q is a nonempty interval of the form $\iota_\ell^q=[a_\ell^q,b_\ell^q]$ or $\iota_\ell^q=[a_\ell^q,b_\ell^q)$ or $\iota_\ell^q=(a_\ell^q,b_\ell^q]$ or $\iota_\ell^q=(a_\ell^q,b_\ell^q)$ for $a_\ell^q\le b_\ell^q$ for all $\ell\in\mathcal{L}$ such that

$$\mathcal{X}_q=\prod_{\ell\in\mathcal{L}}\iota_\ell^q. \tag{36}$$

In this case, let $a^q=\{a_\ell^q\}_{\ell\in\mathcal{L}}$ and $b^q=\{b_\ell\}_{\ell\in\mathcal{L}}$. ■

In the sequel, we assume $\mathcal{D}$ has the form

$$\mathcal{D}=\{d\mid \underline{d}\le d\le\overline{d}\} \tag{37}$$

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 2 justifies our interest in finite abstractions induced by interval partitions for mixed monotone systems. In particular, given an interval partition of a mixed monotone system, let

$$R_{q,u}\triangleq \{x'\mid f((a^q,b^q),u,(\underline{d},\overline{d}))\le x'\le f((b^q,a^q),u,(\underline{d},\overline{d})). \tag{38}$$

Then, by Theorem 2, $R_{q,u}$ satisfies (21). The following now follows readily:

Theorem 3 ([24, Theorem 2])**.** *Consider mixed monotone system* (12) *with interval 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* (22) *with $R_{q,u}$ given by* (38)*. Then $\mathcal{T}=(\mathcal{Q},\mathcal{U},\delta)$ is a finite state abstraction of* (12).

Observe that, for interval 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. $f((b^q,a^q),u,(\underline{d},\overline{d}))$ to $a^{q'}$). Thus, considering (22), it is straightforward to compute $\delta(q,u)$ for each q and u. See [24] for details regarding the computational requirements of obtaining a finite state abstraction from an interval partition using Theorem 3.

D. Abstraction from Piecewise Linearity

We return to the special case in Section II-B for which the supply and demand functions are piecewise linear, resulting in the piecewise affine dynamics (11). 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. From this observation, we propose a modification to the above finite abstraction as developed in [15].

We recall the formulation in Section II-B 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 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 $R_{q,u}^{\text{exact}}$ for each $q\in\mathcal{Q}$ and $u\in\mathcal{U}$ using polyhedral operations, and then determine the set $\delta(q,u)\triangleq\{q'\mid\mathcal{X}_{q'}\cap R_{q,u}^{\text{exact}}\neq\emptyset\}$, completing the construction of the finite state abstraction. This approach is used to compute finite abstractions of freeway network models in [35].

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 [60], [61], which is $|\mathcal{L}|$ for traffic networks. For low dimensional systems, this computation is efficient as compared to more general reachable set computation techniques, but quickly becomes intractable even for systems of modest size. In contrast, over-approximating the reachable set using mixed monotonicity always requires evaluating the decomposition function at only two points, regardless of the dimension of the state space. Additionally, while exact

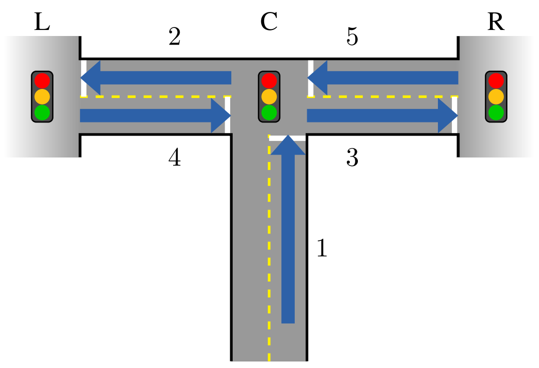


Fig. 7. Example network with three signalized intersections and 5 links, denoted with arrows and numbered as shown. Links 1, 4, and 5 direct exogenous traffic onto the network, and at each time step, the exogenous arrivals are assumed to be within the disturbance set $\mathcal{D}$. If the queues on links 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 φ_{case} as given in (43).

reachable sets eliminate one-step spurious trajectories, more general spurious trajectories remain such as those exhibited in Example 1. 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 Figure 7 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 Figure 7. 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 (*e.g.*, some unmodeled link(s) are actuated in this mode). It follows that $|\mathcal{U}| = 8$ to capture the 8 possible combinations of red/green for the three signals. Time is discretized so that one time step is 15 seconds.

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

$$d[t] \in \mathcal{D} \triangleq \{d \mid 0 \leq d \leq [15\ 0\ 0\ 0\ 0]^T\} \tag{39}$$
$$\cup \{d \mid 0 \leq d \leq [0\ 0\ 0\ 15\ 15]^T\}, \tag{40}$$

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 adopt the piecewise affine model suggested in Section II-B so that

$$\Phi_\ell^{\text{out}}(x_\ell) = \min\{x_\ell, c_\ell\} \tag{41}$$
$$\Phi_\ell^{\text{in}}(x_\ell) = x_\ell^{\text{cap}} - x_\ell \tag{42}$$

where we assume $c_\ell = 20$ is the saturation flow for all ℓ. We see that, akin to Example 3, flow from links 1, 4, and 5 may be blocked by the queues on links 2 and 3. Nonetheless, the choice of parameters ensures that $\partial F_\ell / \partial x_\ell \geq 0$ for all ℓ so that the dynamics are mixed monotone.

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

$$\varphi_{\text{case}} = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \tag{43}$$

where

$$\varphi_1 = \Box\Diamond(\text{left signal is "red"}) \tag{44}$$
$$\varphi_2 = \Box\Diamond(\text{right signal is "red"}) \tag{45}$$
$$\varphi_3 = \Diamond\Box\left(\bigwedge_{i\in\{1,4,5\}} (x_i \leq 30)\right) \tag{46}$$
$$\varphi_4 = \Box\big((x_2 > 30 \vee x_3 > 30) \implies (x_2 \leq 10 \wedge x_3 \leq 10)\big). \tag{47}$$

We interpret (44)–(47) 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 an interval 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:

$$\mathbb{Q}_\ell = \{[0,15],(15,20],(20,25],(25,30],(30,35],(35,40]\},\quad \ell \in \{1,4,5\} \tag{48}$$
$$\mathbb{Q}_\ell = \{[0,10],(10,20],(20,30],(30,40]\},\qquad \ell \in \{2,3\}. \tag{49}$$

Then we take

$$\mathcal{Q} = \prod_{\ell\in\mathcal{L}} \mathbb{Q}_\ell \tag{50}$$

to index the induced interval 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

$$\mathcal{X}_q = q^1 \times q^2 \times q^3 \times q^4 \times q^5 \subseteq \mathbb{R}^5. \tag{51}$$

The resulting transition system has $|\mathcal{Q}| = \prod_{\ell\in\mathcal{L}} |\mathbb{Q}_\ell| = 3456$ states. Building the transition system using the mixed monotone properties of the dynamics takes 35.2 seconds on a standard laptop. The average number of transitions from a given partition under a particular input is 73.9. Notice that φ_{case} includes specifications on the input, specifically, φ_1 and φ_2 impose conditions on the left and right signals. To accomodate 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 may be found in [31]. The final transition system that models the behavior of our concrete system then has $8 \times 3456 = 27\,648$ states.

The LTL specification φ_{case} is transformed into a Rabin automaton with 29 states and one acceptance pair using the `ltl2dstar` tool [48]. Solving the resulting Rabin game takes 42.8 minutes on a standard laptop and results in a control strategy such that the specification φ_{case} is satisfied from any initial condition. We plot a resulting trace of the traffic network dynamics in Figure 8. To produce the traces, a random disturbance input is synthesized satisfying (39) for which larger disturbances were favored.

Figure 8 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 accomodate φ_3, the signaling mode at the center intersection responds to the present conditions which depend on the particular realization of the disturbance input.

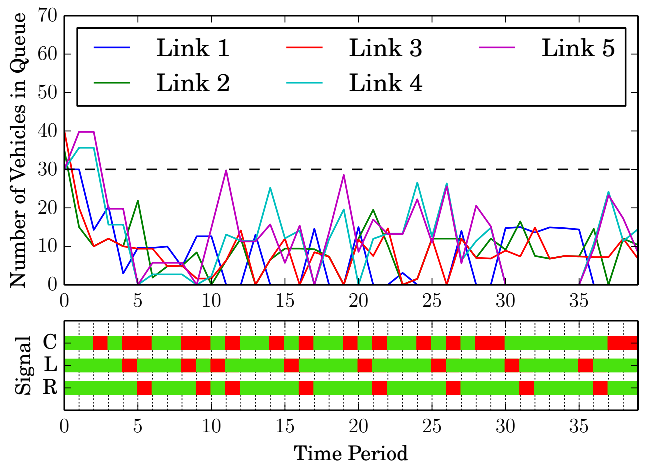
In Figure 8, a naïve control strategy is used that satisfies φ_1 and φ_2 by using a cyclic control strategy with period 4. This strategy may be considered reasonable since it spends limited time in the "red" mode at the left and right signals, and evenly divides the time between "green" and "red" modes at the center signal. However, this fixed strategy is unable to react to the realized disturbance and does not satisfy φ_4. Furthermore, even if this naïve strategy happens to satisfy φ_3, it is difficult to verify this with certainty. The same initial condition and disturbance input is used in both cases in Figure 8.

Finally, we note that, in principle, we could use the alternative abstraction method that relies on the piecewise affine dynamics as proposed in Section V-D. 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 $|\mathcal{U}||\mathcal{Q}|^2 \approx 9.6\times10^7$ polyhedral intersection operations to compute δ, rendering this method nearly intractable. 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.

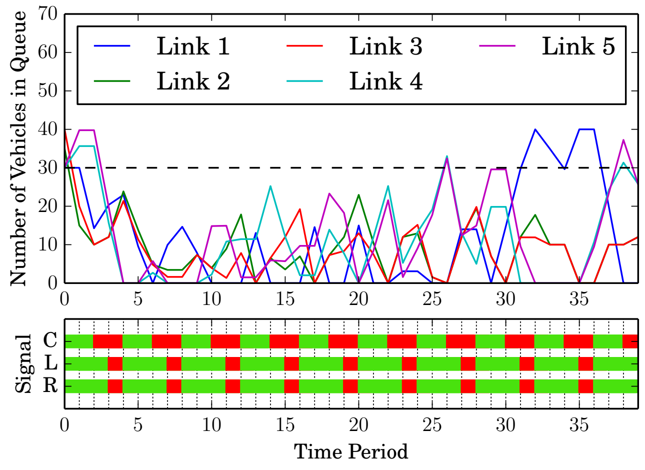
VII. Conclusions

In this tutorial paper, we have described a formal methods approach to control of traffic flow networks. First, we considered a compartmental model for transportation networks 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 accomodate fewer vehicles of this type. Accommodating such additions in the model 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 by constructing a finite state abstraction. This approach is general and 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



(a)



(b)

Fig. 8. (a) Sample trace of the controlled traffic network dynamics for the network in Figure 7. 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 paper. As suggested by the sample trace, the closed-loop behavior of the concrete system is guaranteed to satisfy φ_{case}. (b) Sample trace 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 φ_{case} is not satisfied for this naïve controller.

that a controller synthesized from the abstraction may be applied to the original concrete system.

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 accomodate a broad range of cases including continuous-time dynamics, continuous inputs, overlapping/uncertain state and input quantization, and probabilistic systems. Each of these cases poses unique challenges, and care must be taken to ensure that a controller synthesized from the abstraction can be effectively and efficiently applied to the original 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. 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.

Regardless, reachable set computations is only one of the difficulties in efficient finite state abstraction. For example, the size of the state space partition is generally required to increase 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 (48) is the relatively large interval $[0, 15]$. Other approaches include avoiding partitioning of the state space altogether [62], [63], an idea closely related to ℓ-complete approximations [17], [21], [22].

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 [43], [64]. One successful approach is the *assume-guarantee* framework [65] whereby each subsystem *assumes* a certain behavior from neighboring systems and symmetrically *guarantees* a certain level of performance. Each subsystem then synthesizes a controller under these assumptions and guarantees. Such an approach is well-suited for traffic networks as they may be naturally divided into neighborhoods or towns interconnected via a few roads. This approach is explored in [66].

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. This paper has shown that traffic networks are a particularly rich example of physical systems with extensive structure induced by topology, physics, and observed phenomenological properties. By relying on such systems for concreteness, we are able to develop general theory and algorithms that not only lead to more efficient transportation, but also enable powerful and scalable new tools for control of complex systems.

References

[1] 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.

[2] T. Lomax, D. Schrank, and B. Eisele, "2012 annual urban mobility report," tech. rep., Texas Transportation Institute, 2012.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] A. Schmuck, P. Tabuada, and J. Raisch, "Comparing asynchronous ℓ-complete approximations and quotient based abstractions," *CoRR*, vol. abs/1503.07139, 2015.

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

[24] 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.

[25] 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.

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

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

[28] 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.

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

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

[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, 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.

[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. arXiv:1409.6354.

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

[35] 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.

[36] 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.

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

[38] 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.

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

[40] 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.

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

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

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

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

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

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

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

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

[49] 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.

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

[51] 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.

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

[53] G. Reissig and M. Rungger, "Feedback refinement relations for symbolic controller synthesis," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 88–94, Dec 2014.

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

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

[61] 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.

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

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

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

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

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