AAAI Spring Symposium on Embedded Reasoning (22-24 Mar 2010, Stanford)
http://www.cds.caltech.edu/~murray/papers/wtm10-aaai.html

Automatic Synthesis of Robust Embedded Control Software

Tichakorn Wongpiromsarn, **Ufuk Topcu** and **Richard M. Murray**
California Institute of Technology
Pasadena, California 91125

Abstract

We propose a methodology for automatic synthesis of embedded control software that accounts for exogenous disturbances. The resulting system is guaranteed, by construction, to satisfy a given specification expressed in linear temporal logic. The embedded control software consists of three components: a goal generator, a trajectory planner, and a continuous controller. We demonstrate the effectiveness of the proposed technique through an example of an autonomous vehicle navigating an urban environment. This example also illustrates that the system is not only robust with respect to exogenous disturbances but also capable of handling violation of the environment assumptions.

1. Introduction

Design and verification of modern engineered systems with a tight link between computational and physical elements have become increasingly difficult due to their complexity and the interleaving between the high-level logics and the low-level controllers. Recently, there have been growing interests in integrating the methodologies from computer science and control to allow automatic synthesis of embedded control software for such systems (Karaman and Frazzoli 2009; Kress-Gazit, Fainekos, and Pappas 2007; Conner et al. 2007; Kloetzer and Belta 2008; Wongpiromsarn, Topcu, and Murray 2009; 2010). The goal of the synthesis is to guarantee that, by construction, the system satisfies its specification regardless of the environment in which it operates (subject to certain assumptions on the environment that need to be stated in the specification). The specification is usually expressed in the language of temporal logic (Manna and Pnueli 1992; Huth and Ryan 2004; Emerson 1990). With its expressive power, a wider class of properties than safety and stability, typically studied in control, can be specified. As a consequence, the system will be capable of performing much more complex tasks than, e.g., converging to a desired operating point while always staying within a safe set. For example, in (Wongpiromsarn, Topcu, and Murray 2009) and (Wongpiromsarn, Topcu, and Murray 2010), we described how tasks such as reaching certain areas and visiting certain areas infinitely often and traffic rules such as avoiding obstacles, staying in the travel lane and respecting precedence order at intersections can be expressed in linear temporal logic.

A common approach to automatic synthesis of embedded control software that guarantees the correctness of the resulting system is to construct a finite transition system that serves as an abstract model of the physical system (which typically has infinitely many states) and synthesize a strategy, represented by a finite state automaton, satisfying the specification based on the abstract model. This leads to a hierarchical, two-layer design with a discrete planner computing a strategy based on the abstract model and a continuous controller computing a control signal based on the physical model to continuously implement the strategy. Simulations/bisimulations (Alur et al. 2000) provide the proof that the continuous execution preserves the desired properties.

The correctness of this hierarchical approach relies on the abstraction of systems evolving on a continuous domain into equivalent (in the simulation sense) finite state models. If the abstraction is done properly such that the continuous controller is capable of implementing any strategy computed by the trajectory planner, then it is guaranteed that the correctness of the strategy is preserved in the continuous execution.

Several abstraction methods have been proposed based on a fixed abstraction. For example, a continuous-time, time-invariant model was considered in (Kress-Gazit, Fainekos, and Pappas 2007), (Conner et al. 2007) and (Kloetzer and Belta 2008) for special cases of fully actuated ($\dot{s}(t) = u(t)$), kinematic ($\dot{s}(t) = A(s(t))u(t)$) and piecewise affine (PWA) dynamics, respectively, while a discrete-time, time-invariant model was considered in (Wongpiromsarn, Topcu, and Murray 2009) and (Tabuada and Pappas 2006) for special cases of PWA and controllable linear systems respectively. Reference (Girard and Pappas 2009) deals with more general dynamics by relaxing the bisimulation requirement and using the notions of approximate simulation and simulation functions (Girard, Julius, and Pappas 2008). More recently, a sampling-based method has been proposed for μ-calculus specifications (Karaman and Frazzoli 2009). However, these approaches do not take into account the presence of exogenous disturbances and the resulting system may fail to satisfy its specification if its evolution does not exactly match its model.

To increase the robustness of the system against the effects of direct, external disturbances and a mismatch between the actual system and its model, in this paper, we ex-

tend our work in (Wongpiromsarn, Topcu, and Murray 2009; 2010) to deal with a discrete-time linear time-invariant state space model with exogenous disturbances and provide an approach to automatically compute a finite state abstraction for such a model. We demonstrate the effectiveness of the proposed technique through an example of an autonomous vehicle navigating an urban environment. This example also illustrates that the system is not only robust with respect to exogenous disturbances but also capable of handling violation of the environment assumptions.

2. PRELIMINARIES

We use linear temporal logic (LTL) to describe the desired properties of the system. In this section, we first give formal definitions of terminology and notations used throughout the paper. Then, based on these definitions, we briefly describe LTL and some important classes of LTL formulas. The exposition in this section is parallel to (Wongpiromsarn, Topcu, and Murray 2010) and is provided for the completeness of the paper.

Definition 1. *A system consists of a set V of variables. The* domain *of V, denoted by $dom(V)$, is the set of valuations of V. A* state *of the system is an element $v \in dom(V)$.*

Definition 2. *A* finite transition system *is a tuple $\mathbb{T} := (\mathcal{V}, \mathcal{V}_0, \rightarrow)$ where $\mathcal{V}$ is a finite set of states, $\mathcal{V}_0 \subseteq \mathcal{V}$ is a set of inital states, and $\rightarrow \subseteq \mathcal{V} \times \mathcal{V}$ is a transition relation. Given states $\nu_i, \nu_j \in \mathcal{V}$, we write $\nu_i \rightarrow \nu_j$ if there is a transition from ν_i to ν_j.*

Definition 3. *An* atomic proposition *is a statement on system variables v that has a unique truth value (True or False) for a given value of v. Let $v \in dom(V)$ be a state of the system and p be an atomic proposition. We write $v \Vdash p$ if p is True at the state v. Otherwise, we write $v \nVdash p$.*

Definition 4. *An* execution *σ of a discrete-time system is an infinite sequence of the system states over a particular run, i.e., σ can be written as $\sigma = v_0 v_1 v_2 \ldots$ where for each $t \geq 0$, $v_t \in dom(V)$ is the state of the system at time t.*

Linear Temporal Logic

Linear temporal logic (Manna and Pnueli 1992; Huth and Ryan 2004; Emerson 1990) is a powerful specification language for unambiguously and concisely expressing a wide range of properties of systems. LTL is built up from a set of atomic propositions, the logic connectives ($\neg$, $\vee$, $\wedge$, $\Longrightarrow$), and the temporal modal operators ($\bigcirc$, $\Box$, $\Diamond$, $\mathcal{U}$ which are read as "next," "always," "eventually," and "until," respectively). An LTL formula is defined inductively as follows: (1) any atomic proposition p is an LTL formula; and (2) given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are also LTL formulas.

Other operators can be defined as follows: $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\varphi \Longrightarrow \psi = \neg\varphi \vee \psi$, $\Diamond\varphi = True\, \mathcal{U}\, \varphi$, and $\Box\varphi = \neg\Diamond\neg\varphi$. A *propositional* formula is one that does not include temporal operators. Given a set of LTL formulas $\varphi_1, \ldots, \varphi_n$, their *Boolean combination* is an LTL formula formed by joining $\varphi_1, \ldots, \varphi_n$ with logic connectives.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given an execution $\sigma = v_0 v_1 v_2 \ldots$ and an LTL formula φ, we say that *φ holds at position $i \geq 0$* of σ, written $v_i \models \sigma$, if and only if (iff) φ holds for the remainder of the execution σ starting at position i. The semantics of LTL is defined inductively as follows: (a) For an atomic proposition p, $v_i \models p$ iff $v_i \Vdash p$; (b) $v_i \models \neg\varphi$ iff $v_i \not\models \varphi$; (c) $v_i \models \varphi \vee \psi$ iff $v_i \models \varphi$ or $v_i \models \psi$; (d) $v_i \models \bigcirc\varphi$ iff $v_{i+1} \models \varphi$; and (e) $v_i \models \varphi \,\mathcal{U}\, \psi$ iff $\exists j \geq i, v_j \models \psi$ and $\forall k \in [i, j), v_k \models \varphi$. Based on this definition, $\bigcirc\varphi$ holds at position v_i iff φ holds at the next state v_{i+1}, $\Box\varphi$ holds at position i iff φ holds at every position in σ starting at position i, and $\Diamond\varphi$ holds at position i iff φ holds at some position $j \geq i$ in σ.

Definition 5. *An execution $\sigma = v_0 v_1 v_2 \ldots$* satisfies *$\varphi$, denoted by $\sigma \models \varphi$, if $v_0 \models \varphi$.*

Definition 6. *Let Σ be the set of all executions of a system. The system is said to be* correct *with respect to its specification φ, written $\Sigma \models \varphi$, if all its executions satisfy φ, that is, $(\Sigma \models \varphi) \iff \big(\forall\sigma, (\sigma \in \Sigma) \implies (\sigma \models \varphi)\big)$.*

Examples: Given propositional formulas p and q describing the states of interest, important and widely-used properties can be defined in terms of their corresponding LTL formulas as follows.

Safety (invariance): A safety formula is of the form $\Box p$, which simply asserts that the property p remains invariantly true throughout an execution. Typically, a safety property ensures that nothing bad happens. A classic example of safety property frequently used in the robot motion planning domain is obstacle avoidance.

Guarantee (reachability): A guarantee formula is of the form $\Diamond p$, which guarantees that the property p becomes true at least once in an execution, i.e., a state satisfying p is reachable. Reaching a goal state is an example of a guarantee property.

Obligation: An obligation formula is a disjunction of safety and guarantee formulas, $\Box p \vee \Diamond q$. It can be easily shown that any safety and progress property can be expressed using an obligation formula. (By letting $q \equiv False$, we obtain a safety formula and by letting $p \equiv False$, we obtain a guarantee formula.)

Progress (recurrence): A progress formula is of the form $\Box\Diamond p$, which essentially states that the property p holds infinitely often in an execution. As the name suggests, a progress property typically ensures that the system keeps making progress throughout the execution.

Response: A response formula is of the form $\Box(p \implies \Diamond q)$, which states that following any point in an execution where the property p is true, there exists a point where the property q is true. A response property can be used to describe how the system should react to changes in the operating conditions.

Stability (persistence): A stability formula is of the form $\Diamond\Box p$, which asserts that there is a point in an execution where the property p becomes invariantly true for the remainder of the execution. This definition corresponds to the definition of stability in the controls domain since it essentially ensures that eventually, the system converges to a de-

sired operating point and remains there for the remainder of the execution.

Remark 1. *Properties typically studied in the control and hybrid systems domains are safety (usually in the form of constraints on the system state) and stability (i.e., convergence to an equilibrium or a desired state). LTL thus offers extensions to properties that can be expressed. Not only can it express other classes of properties, but it also allows more general safety and stability properties than constraints on the system state or convergence to an equilibrium since p in $\Box p$ and $\Diamond\Box p$ can be any propositional formula.*

3. Problem Formulation

We are interested in designing embedded control software for a system that interacts with its (potentially dynamic and not a priori known) environment. This software needs to ensure that the system satisfies the desired property φ_s in the presence of exogenous disturbances for any initial condition and any environment in which it operates, provided that the initial condition and the environment satisfy certain assumptions, φ_{init} and φ_e, respectively.

We assume that the desired property φ_s and the assumptions φ_{init} and φ_e are expressed in LTL. Specifically, we define the system model $\mathbb{S}$, the desired property φ_s and the assumptions φ_{init} and φ_e as follows.

System Model: Consider a system model $\mathbb{S}$ with a set $V = S \cup E$ of variables where S and E are disjoint sets that represent the set of variables controlled by the system and the set of variables controlled by the environment respectively. The domain of V is given by $dom(V) = dom(S) \times dom(E)$ and a state of the system can be written as $v = (s, e)$ where $s \in dom(S) \subseteq \mathbb{R}^n$ and $e \in dom(E)$. Throughout the paper, we call s the *controlled* state and e the *environment* state.

Assume that the controlled state evolves according to the following discrete-time linear time-invariant state space model:

$$\left.\begin{array}{rcl} s[t+1] & = & As[t] + Bu[t] + Ed[t] \\ u[t] & \in & U \\ d[t] & \in & D \\ s[0] & \in & dom(S) \end{array}\right\} \forall t \in \mathbb{N} \quad (1)$$

where $U \subseteq \mathbb{R}^m$ is the set of admissible control inputs, $D \subseteq \mathbb{R}^p$ is the set of exogenous disturbances and $s[t]$, $u[t]$ and $d[t]$ are the controlled state, the control signal, and the exogenous disturbance, respectively, at time t.

Example 1. *Consider a robot motion planning problem where a robot needs to navigate an environment populated with (potentially dynamic) obstacles and explore certain areas of interest. S typically includes the state (e.g. position and velocity) of the robot while E typically includes the positions of obstacles (which are generally not known a priori and may change over time). The evolution of the controlled state (i.e., the state of the robot) is simply governed its equations of motion.*

Desired Properties and Assumptions: Let Π be a finite set of atomic propositions of variables from V. Each of the atomic propositions in Π essentially captures the states of interest. We assume that the desired property φ_s is an LTL specification built from Π and can be expressed as a conjunction of safety, guarantee, obligation, progress, response and stability properties as follows:

$$\begin{array}{rl} \varphi_s = & \bigwedge_{j\in J_1} \Box p^s_{1,j} \ \wedge \ \bigwedge_{j\in J_2} \Diamond p^s_{2,j} \ \wedge \\ & \bigwedge_{j\in J_3} (\Box p^s_{3,j} \ \vee \ \Diamond q^s_{3,j}) \ \wedge \ \bigwedge_{j\in J_4} \Box\Diamond p^s_{4,j} \ \wedge \\ & \bigwedge_{j\in J_5} \Box(p^s_{5,j} \implies \Diamond q^s_{5,j}) \ \wedge \ \bigwedge_{j\in J_6} \Diamond\Box p^s_{6,j}, \end{array} \quad (2)$$

where $J_1, \ldots, J_6$ are finite sets and for any i and j, $p^s_{i,j}$ and $q^s_{i,j}$ are propositional formulas of variables from V that are built from Π.

We further assume that the initial condition of the system satisfies a propositional formula φ_{init} built from Π and the environment satisfies an assumption φ_e where φ_e can be expressed as a conjunction of justice requirements and propositions that are assumed to be true throughout an execution as follows:

$$\varphi_e = \bigwedge_{i\in I_1} \Box p^e_{f,i} \ \wedge \ \bigwedge_{i\in I_2} \Box\Diamond p^e_{s,i}, \quad (3)$$

where $p^e_{f,i}$ and $p^e_{s,i}$ are propositional formulas built from Π and only contain variables from E (i.e., environment variables).

In summary, the specification φ of $\mathbb{S}$ is given by

$$\varphi = \big(\varphi_{init} \ \wedge \ \varphi_e\big) \implies \varphi_s. \quad (4)$$

Observe, from (4), that the desired property φ_s is guaranteed only when the assumptions on the initial condition and the environment are satisfied.

Remark 2. *We restrict φ_s and φ_e to be of the form (2) and (3), respectively, for the clarity of presentation. Our framework only requires that the specification (4) can be reduced to a subclass of GR[1] formula of the form:*

$$\begin{array}{l} \big(\psi_{init} \ \wedge \ \Box\psi^e_e \bigwedge_{i\in I_f} \Box\Diamond\psi^e_{f,i}\big) \\ \implies \big(\bigwedge_{i\in I_s} \Box\psi_{s,i} \ \wedge \ \bigwedge_{i\in I_g} \Box\Diamond\psi_{g,i}\big), \end{array} \quad (5)$$

where (a) ψ_{init} and $\psi_{g,i}$ are propositional formulas of variables from V, (b) ψ^e_e is a Boolean combination of propositional formulas of variables from E and expressions of the form $\bigcirc\psi^t_e$ where ψ^t_e is a propositional formula of variables from E that describes the assumptions on the transitions of environment states, (c) $\psi^e_{f,i}$ is a propositional formulas of variables from E, and (d) $\psi_{s,i}$ is a Boolean combination of propositional formulas of variables from V and expressions of the form $\bigcirc\psi^t_s$ where ψ^t_s is a propositional formula of variables from V that describes the constraints on the transitions of controlled states. Throughout the paper, we call the left hand side and the right hand side of (5) the "assumption" part and the "guarantee" part, respectively.

In (Wongpiromsarn, Topcu, and Murray 2010), we showed that the specification (4) can be reduced to the form of equation (5). Hence, for the rest of the paper, we will only consider a specification of the form (5).

4. Embedded Control Software

A common approach to automatic synthesis of embedded control software as described in Section 3. is to construct a finite transition system $\mathbb{D}$ that serves as an abstract model of the physical system $\mathbb{S}$ (which typically has infinitely many states) and synthesize a strategy, represented by a finite state automaton, satisfying the specification (5) based on the abstract model $\mathbb{D}$. This leads to a hierarchical, two-layer design with a discrete planner computing a strategy based on the abstract model $\mathbb{D}$ and a continuous controller computing a control signal based on the physical model $\mathbb{S}$ to continuously implement the strategy. Simulations/bisimulations provide the proof that the continuous execution preserves the desired properties.

One of the main challenges of this approach is the computational complexity in the synthesis of finite state automata (i.e., discrete planners). Although it has been shown that for a specification of the form (5), an automaton can be automatically computed in polynomial time (Piterman, Pnueli, and Sa'ar 2006), the applications of the synthesis tool are limited to small problems due to the state explosion issue. To address this problem, in (Wongpiromsarn, Topcu, and Murray 2010), we described an extension of traditional receding horizon control to incorporate linear temporal logic specifications of the form (5). Its implementation leads to the decomposition of the discrete planner into a goal generator and a trajectory planner and the resulting embedded control software consists of three components as depicted in Figure 1. The goal generator essentially reduces the synthesis problem to a sequence of smaller problems of short horizon while preserving the desired system-level temporal properties. Subsequently, in each iteration, the trajectory planner solves the corresponding short-horizon problem with the currently observed state as the initial state and generates a feasible trajectory to be implemented by the continuous controller. We showed that the sequence of trajectories generated by the trajectory planner satisfies the given specification (5), provided that all the assumptions on the environment stated in (5) hold throughout the execution. To handle failures that may occur due to direct, external disturbances, a mismatch between the actual system and its model and violation of the environment assumptions, we proposed a response mechanism that enables the system to respond to certain failures and continue to exhibit a correct behavior.

The correctness of this hierarchical approach relies on the abstraction of systems evolving on a continuous domain into equivalent (in the simulation sense) finite state models. If the abstraction is done properly such that the continuous controller is capable of implementing any strategy computed by the trajectory planner, then it is guaranteed that the correctness of the strategy is preserved in the continuous execution.

Several abstraction methods have been proposed for different cases of system dynamics (Kress-Gazit, Fainekos, and Pappas 2007; Conner et al. 2007; Kloetzer and Belta 2008; Wongpiromsarn, Topcu, and Murray 2009; Tabuada and Pappas 2006; Girard and Pappas 2009; Karaman and Frazzoli 2009). However, these approaches do not take into account the presence of exogenous disturbances and the resulting system may fail to satisfy its specification if its evolution does not exactly match its model.

To increase the robustness of the system with respect to exogenous disturbances, in the next section, we extend our work in (Wongpiromsarn, Topcu, and Murray 2009) to deal with a discrete-time linear time-invariant state space model with bounded disturbances as in (1) and provide an approach to automatically compute a finite state abstraction $\mathbb{D}$ of $\mathbb{S}$.

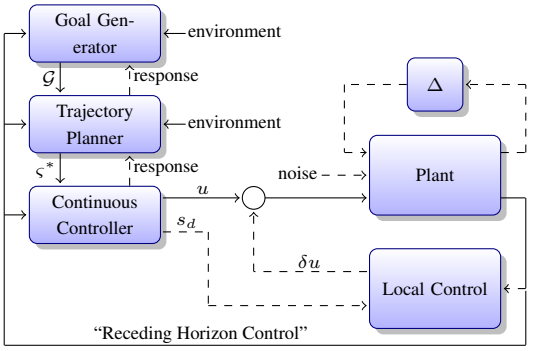

Figure 1: A system with the proposed embedded control software. Δ models uncertainties in the plant model. The local control is implemented to account for the effect of the noise and unmodeled dynamics captured by Δ.

5. Computing Finite State Abstraction

To construct a finite transition system $\mathbb{D}$ from the physical model $\mathbb{S}$, we first partition $dom(S)$ and $dom(E)$, as in (Kress-Gazit, Fainekos, and Pappas 2007; Kloetzer and Belta 2008), into a finite number of equivalence classes or cells $\mathcal{S}$ and $\mathcal{E}$, respectively, such that the partition is *proposition preserving* (Alur et al. 2000). Roughly speaking, this means that for any atomic proposition $\pi \in \Pi$ and any states v_1 and v_2 that belong to the same cell in the partition, if v_1 satisfies π, then v_2 also satisfies π. We denote the resulting discrete domain of the system by $\mathcal{V} = \mathcal{S} \times \mathcal{E}$. Throughout the paper, we call $v \in dom(V)$ a *continuous state* and $\nu \in \mathcal{V}$ a *discrete state* of the system. For a discrete state $\nu \in \mathcal{V}$, we say that ν satisfies an atomic proposition $\pi \in \Pi$, denoted by $\nu \Vdash_d \pi$, if and only if there exists a continuous state v contained in the cell labeled by ν such that v satisfies π. Given an infinite sequence of discrete states $\sigma_d = \nu_0\nu_1\nu_2 \ldots$ and an LTL formula φ built from Π, we say that φ holds at position $i \geq 0$ of σ_d, written $\nu_i \models_d \varphi$, if and only if φ holds for the remainder of σ_d starting at position i. With these definitions, the semantics of LTL for a sequence of discrete states can be derived from the general semantics of LTL (Manna and Pnueli 1992; Huth and Ryan 2004; Emerson 1990).

Next, we need to determine the transition relations $\rightarrow$ of $\mathbb{D}$. In Section 5.1, we use a variant of the notion of *reachability* that is sufficient to guarantee that if a discrete controlled state ς_j is reachable from ς_i, the transition from ς_i to ς_j can be continuously *implemented* or *simulated* by a continuous controller. See, for example, (Tanner and Pappas 2002) for the exact definition. A computational scheme that provides a sufficient condition for reachability between two

discrete controlled states and subsequently refines the state space partition is also presented in Sections 5.3 and 5.4.

5.1 Finite Time Reachability

Let $\mathcal{S} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_l\}$ be a set of discrete controlled states. We define a map $T_s : dom(S) \rightarrow \mathcal{S}$ that sends a continuous controlled state to a discrete controlled state of its equivalence class. That is, $T_s^{-1}(\varsigma_i) \subseteq dom(S)$ is the set of all the continuous controlled states contained in the cell labeled by ς_i and $\{T_s^{-1}(\varsigma_i), \ldots, T_s^{-1}(\varsigma_n)\}$ is the partition of $dom(S)$. We define the reachability relation, denoted by $\rightsquigarrow$, as follows: a discrete state ς_j is reachable from a discrete state ς_i, written $\varsigma_i \rightsquigarrow \varsigma_j$, only if starting from any point $s[0] \in T_s^{-1}(\varsigma_i)$, there exists a horizon length $N \in \mathbb{N}$ and a control law $u[t] \in U$ that takes the system (1) to a point $s[N] \in T_s^{-1}(\varsigma_j)$ satisfying the constraint $s[t] \in T_s^{-1}(\varsigma_i) \cup T_s^{-1}(\varsigma_j), \forall t \in \{0, \ldots, N\}$ for any exogenous disturbances $d[t] \in D$. Note that this is stronger than the usual definition of reachability (Vinter 1980; Prajna 2005) since we also impose the requirement that the trajectory always remain within the "safe" set. We write $\varsigma_i \not\rightsquigarrow \varsigma_j$ if ς_j is not reachable from ς_i.

In general, for two discrete states ς_i and ς_j, verifying the reachability relation $\varsigma_i \rightsquigarrow \varsigma_j$ is hard. Therefore, we consider the restricted case where the horizon length is fixed and given and U, D and $T_s^{-1}(\varsigma_i), i \in \{1, \ldots, l\}$ are polyhedral sets. Our approach relies on solving the following problem: Given discrete controlled states $\varsigma_i, \varsigma_j \in \mathcal{S}$, the set of admissible control inputs $U \subseteq \mathbb{R}^m$, the set of exogenous disturbances $D \subseteq \mathbb{R}^p$, the matrices A, B and E as in (1), a horizon length $N \geq 0$, find the set of initial states $\mathcal{S}_0 \subseteq \mathbb{R}^n$ such that for any $s[0] \in \mathcal{S}_0$, there exist $u[0], u[1], \ldots, u[N-1] \in \mathbb{R}^m$ such that

$$\begin{array}{l} s[t+1] = As[t] + Bu[t] + Ed[t] \\ s[t] \in T_s^{-1}(\varsigma_i) \cup T_s^{-1}(\varsigma_j), \;\; s[N] \in T_s^{-1}(\varsigma_j), \;\; u[t] \in U, \\ \forall t \in \{0, \ldots, N-1\}, d[0], \ldots, d[N-1] \in D \end{array} \tag{6}$$

5.2 Preliminaries on Polyhedral Convexity

We consider the case where U, D and $T_s^{-1}(\varsigma_i), i \in \{1, \ldots, l\}$ are polyhedral sets defined as follows.

Definition 7. *A subset P of $\mathbb{R}^n$ is said to be a* polyhedral set *if it is nonempty and has the form $P = \{p \mid Gp \leq h\}$ for some $G \in \mathbb{R}^{r \times n}$ and $h \in \mathbb{R}^r$.*

Definition 8. *Let P be a nonempty convex set. A point $p \in P$ is an* extreme point *of P if and only if it does not lie strictly between the endpoints of any line segment contained in the set, i.e.,*

$$p = \lambda p_1 + (1-\lambda)p_2, \; p_1, p_2 \in P, \; \lambda \in (0,1) \implies p = p_1 = p_2.$$

To compute the set $\mathcal{S}_0$ of initial states for which (6) is feasible, we apply the following results on polyhedral convexity. The proofs for the next three propositions can be found in (Bertsekas, Nedić, and Ozdaglar 2003).

Proposition 1. *Let P be a polyhedral subset of $\mathbb{R}^n$. If P has the form $P = \{p \in \mathbb{R}^n \mid g_j' p \leq h_j, j = 1, \ldots, r\}$ where $g_j \in \mathbb{R}^n$ and $h_j \in \mathbb{R}$, then a point $p \in P$ is an extreme point of P if and only if the set $G_p \triangleq \{g_j \mid g_j' p = h_j, j \in \{1, \ldots, r\}\}$ contains n linearly independent vectors.*

Proposition 2. *Let P be a nonempty convex subset of $\mathbb{R}^n$. If P is closed, then P has at least one extreme point if and only if it does not contain a line, i.e., a set of the form $\{p + \lambda h \mid \lambda \in \mathbb{R}\}$, where $h \in \mathbb{R}^n$ is nonzero and $p \in P$.*

Proposition 3 (Fundamental Theorem of Linear Programming)**.** *Let P be a polyhedral set that has at least one extreme point. A linear function that is bounded below over P attains a mininum at some extreme point of P.*

Using Proposition 1, we can derive the following proposition. The proof is omitted owing to limited space.

Proposition 4. *Let P be a polyhedral subset of $\mathbb{R}^n$ and let $\overline{P}$ be the set of all its extreme points. For any natural number N, $P^N \triangleq \underbrace{P \times \ldots \times P}_{N\text{times}}$ is a polyhedral subset of $\mathbb{R}^{nN}$ and $\overline{P}^N \triangleq \underbrace{\overline{P} \times \ldots \times \overline{P}}_{N\text{times}}$ is the set of all its extreme points.*

In addition, the following proposition can be proved using Proposition 2.

Proposition 5. *Let P be a polyhedral subset of $\mathbb{R}^n$. If P is closed and bounded, then P has at least one extreme point.*

Proof. Assume, to arrive at a contradiction, that P does not have an extreme point. Then, from Proposition 2, P contains a line $L = \{p + \lambda h \mid \lambda \in \mathbb{R}\}$ where $h \in \mathbb{R}^n$ is nonzero and $p \in P$. This contradicts the assumption that P is bounded. □

Finally, the next three propositions can be found in standard textbooks on topology, e.g., (Rudin 1976).

Proposition 6 (Heine-Borel Theorem)**.** *A subset of Euclidean space $\mathbb{R}^n$ is compact if and only if it is closed and bounded.*

Proposition 7 (Tychonoff's Theorem)**.** *The product of any collection of compact topological spaces is compact.*

Proposition 8 (Extreme Value Theorem)**.** *A continuous real-valued function on a nonempty compact space is bounded and attains its supremum.*

5.3 Verifying the Reachability Relation

Given the discrete controlled states $\varsigma_i, \varsigma_j \in \mathcal{S}$, to determine whether $\varsigma_i \rightsquigarrow \varsigma_j$, we essentially have to verify that $T_s^{-1}(\varsigma_i) \subseteq \mathcal{S}_0$ where $\mathcal{S}_0$ is the set of initial states for which (6) is feasible. In this section, we describe how $\mathcal{S}_0$ can be computed using an idea from constrained robust optimal control (Borrelli 2003).

Lemma 1. *Suppose U, D and $T_s^{-1}(\varsigma_i), i \in \{1, \ldots, l\}$ are polyhedral sets, i.e., there exist matrices L_1, L_2 and L_3 and vectors M_1, M_2 and M_3 such that $T_s^{-1}(\varsigma_i) = \{s \in \mathbb{R}^n \mid L_1 s \leq M_1\}$, $U = \{u \in \mathbb{R}^m \mid L_2 u \leq M_2\}$ and $T_s^{-1}(\varsigma_j) = \{s \in \mathbb{R}^n \mid L_3 s \leq M_3\}$. Then, (6) can be rewritten in the form*

$$L \begin{bmatrix} s[0] \\ \hat{u} \end{bmatrix} \leq M - G\hat{d} \tag{7}$$

where $\hat{u} \triangleq [u[0]', \ldots, u[N-1]']' \in \mathbb{R}^{mN}$, $\hat{d} \triangleq [d[0]', \ldots, d[N-1]']' \in D^N$ *and the matrices* $L \in \mathbb{R}^{r \times n+mN}$ *and* $G \in \mathbb{R}^{r \times pN}$ *and the vector* $M \in \mathbb{R}^r$ *can be obtained from* L_1, L_2, L_3, M_1, M_2, M_3, A, B *and* E.

Proof. Equation (7) can be obtained by substituting

$$s[t] = A^t s[0] + \sum_{k=0}^{t-1} \left(A^k B u[t-1-k] + A^k E d[t-1-k] \right)$$

and replacing $s[t] \in T_s^{-1}(\varsigma_i) \cup T_s^{-1}(\varsigma_j)$, $u[t] \in U$ and $s[N] \in T_s^{-1}(\varsigma_j)$ with $L_1 s[t] \leq M_1$, $L_2 u[t] \leq M_2$ and $L_3 s[N] \leq M_3$, respectively, in (6). □

Theorem 1. *Suppose* D *is a closed and bounded polyhedral subset of* $\mathbb{R}^p$ *and* $\overline{D}$ *is the set of all its extreme points. Let* $P \triangleq \{y \in \mathbb{R}^{n+mN} \mid Ly \leq M - G\hat{d}_i, \forall \hat{d}_i \in \overline{D}^N\}$ *and let* $\mathcal{S}_0$ *be the projection of* P *onto its first* n *coordinates, i.e.,*

$$\mathcal{S}_0 = \left\{ s \in \mathbb{R}^n \mid \exists \hat{u} \in \mathbb{R}^{mN} \text{ s.t. } L \begin{bmatrix} s \\ \hat{u} \end{bmatrix} \leq M - G\hat{d}_i, \forall \hat{d}_i \in \overline{D}^N \right\}. \tag{8}$$

Then, the problem in (6) is feasible for any $s[0] \in \mathcal{S}_0$.

Proof. Using Lemma 1, to show that the problem in (6) is feasible for any $s \in \mathcal{S}_0$ defined in (8), we will show that for any $s[0] \in \mathcal{S}_0$, there exists $\hat{u} \in \mathbb{R}^{mN}$ such that for all $\hat{d} \in D^N$, $L \begin{bmatrix} s[0] \\ \hat{u} \end{bmatrix} \leq M - G\hat{d}$.

From the Heine-Borel theorem, the Tychonoff's theorem and Proposition 4, we get that D^N is compact. For each $j \in \{1, \ldots, r\}$, let m_j be the j^{th} element of M and g_j' be the j^{th} row of G and define a linear function $f_j : D^N \to \mathbb{R}$ by $f_j(\hat{d}) = m_j - g_j'\hat{d}$. Since f_j is continuous and D^N is compact, from the extreme value theorem, f_j is bounded below over D^N. In addition, since D^N is compact, from the Heine-Borel theorem and Proposition 5, D^N has at least one extreme point. Using the fundamental theorem of linear programming, we can conclude that f_j attains a minimum at some extreme point of D^N.

Assume, for the sake of contradiction, that there exists $s[0] \in \mathcal{S}_0$ and $\hat{d} \in D^N$ such that for any $\hat{u} \in \mathbb{R}^{mN}$, $L \begin{bmatrix} s[0] \\ \hat{u} \end{bmatrix} > M - G\hat{d}$. Then, there exists $j \in \{1, \ldots, r\}$ such that $l_j' \begin{bmatrix} s[0] \\ \hat{u} \end{bmatrix} > f_j(\hat{d})$ where l_j' is the j^{th} row of L. But since f_j attains a minimum at some extreme point of D^N, there exists $\hat{d}_i \in \overline{D}^N$ such that $l_j' \begin{bmatrix} s[0] \\ \hat{u} \end{bmatrix} > f_j(\hat{d}_i)$. This contradicts the assumption that $s[0] \in \mathcal{S}_0$. □

Using Theorem 1, the problem of computing the set $\mathcal{S}_0$ of initial states for which (6) is feasible is reduced to computing a projection of the intersection of finite sets and can be automatically solved using, for example, the Multi-Parametric Toolbox (MPT) (Kvasnica, Grieder, and Baotić 2004).

5.4 State Space Discretization and Correctness of the System

In general, given the previous partition of $dom(S)$ and any $i, j \in \{1, \ldots, n\}$, the reachability relation between ς_i and ς_j may not be established through the set $\mathcal{S}_0$ of initial states for which (6) is feasible since $T_s^{-1}(\varsigma_i)$ is not necessarily covered by $\mathcal{S}_0$ (due to the constraints on u and a specific choice of the finite horizon N). Hence, we refine the partition based on the reachability relation defined earlier to increase the number of valid discrete state transitions of $\mathbb{D}$. The underlying idea is that starting with an arbitrary pair of ς_i and ς_j, we determine the set $\mathcal{S}_0$ of feasible initial states of (6). Then, we partition $T_s^{-1}(\varsigma_i)$ into $T_s^{-1}(\varsigma_i) \cap \mathcal{S}_0$, labeled by $\varsigma_{i,1}$, and $T_s^{-1}(\varsigma_i) \setminus \mathcal{S}_0$, labeled by $T_s^{-1}(\varsigma_{i,2})$, and obtain the following reachability relations: $\varsigma_{i,1} \rightsquigarrow \varsigma_j$ and $\varsigma_{i,2} \not\rightsquigarrow \varsigma_j$. This process is continued until some pre-specified termination criteria are met. More details on the discretization algorithm, including the pseudo-code, can be found in (Wongpiromsarn, Topcu, and Murray 2009).

We denote the resulting set of all the discrete controlled states corresponding to the resulting partition of $dom(S)$ after applying the discretization algorithm by $\mathcal{S}'$. From the definition of reachability, we can follow the argument in (Wongpiromsarn, Topcu, and Murray 2009) to show that for any strategy computed by the trajectory planner and exogenous disturbances $d[t] \in D$, there exists a sequence of control signal $u[t], t \in \{0, \ldots, N-1\}$ that enables the continuous evolution of the system to *simulate* the given strategy. Hence, from the correctness of the strategy proved in (Wongpiromsarn, Topcu, and Murray 2010), the resulting system is guaranteed, by construction, to be correct with respect to the specification (5).

Proposition 9. *Let* $\sigma_d = \nu_0 \nu_1 \ldots$ *be an infinite sequence of discrete states of* $\mathbb{D}$ *where for each natural number* k, $\nu_k \to \nu_{k+1}$, $\nu_k = (\varsigma_k, \epsilon_k)$, $\varsigma_k \in \mathcal{S}'$ *is the discrete controlled state and* $\epsilon_k \in \mathcal{E}$ *is the discrete environment state. If* $\sigma_d \models_d \varphi$, *then by applying a sequence of control laws, each corresponding to a solution of (6) with* $\varsigma_i = \varsigma_k$ *and* $\varsigma_j = \varsigma_{k+1}$, *the infinite sequence of continuous states* $\sigma = v_0 v_1 v_2 \ldots$ *satisfies* φ.

A solution $u[0], \ldots, u[N-1]$ of (6) can be computed by formulating a constrained optimal control problem, which can be automatically solved using a computational software package such as MPT (Kvasnica, Grieder, and Baotić 2004), YALMIP (Löfberg 2004) or NTG (Murray et al. 2003).

6. Example

We revisit the problem of autonomous vehicle navigating an urban environment studied in (Wongpiromsarn, Topcu, and Murray 2010) where the effect of disturbances was not taken into account. Here, we add exogenous disturbances d_x and d_y to the system model, i.e., we assume that the state (x, y) of the vehicle follows a fully actuated model $\dot{x}(t) = u_x(t) + d_x(t)$ and $\dot{y}(t) = u_y(t) + d_y(t)$ subject to the following constraints on the control effort and disturbance: $u_x(t), u_y(t) \in [-1, 1]$ and $d_x(t), d_y(t) \in [-0.1, 0.1], \forall t \geq 0$. The desired properties of the system include visiting certain areas infinitely often and obeying traffic rules (no col-

lision, staying in the travel lane, stopping at a stop line and proceeding through an intersection only when it is clear). To illustrate that the system is capable of handling certain violation of environment assumptions, we design the embedded control software based on the assumption that obstacles may not block a road. We refer the reader to (Wongpiromsarn, Topcu, and Murray 2010) for more details on this example, including assumptions on the environment and the initial state of the system.

In (Wongpiromsarn, Topcu, and Murray 2010), we showed that the system is capable of responding to certain failures caused by violation of environment assumptions. In this section, we show that if the presence of disturbances is incorporated in the embedded control software synthesis, then the resulting system will also be robust with respect to exogenous disturbances and always exhibit a correct behavior regardless of the disturbances, provided that the disturbances remain within the set $D = [-0.1, 0.1] \times [-0.1, 0.1]$.

We consider the road network shown in Figure 2 with 3 intersections, I_1, I_2 and I_3, and 6 roads, R_1, R_2 (joining I_1 and I_3), R_3, R_4 (joining I_2 and I_3), R_5 (joining I_1 and I_3) and R_6 (joining I_1 and I_2). Based on the specification of the system, we partition the roads and intersections into $N = 282$ cells as shown in Figure 2.

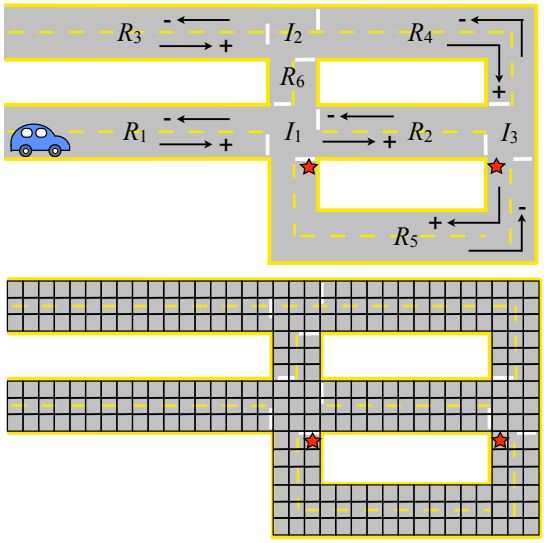

Figure 2: The road network and its partition for the autonomous vehicle example. The solid (black) lines define the states in the set $\mathcal{V}$ of the finite state model $\mathbb{D}$ used by the trajectory planner. The stars indicate the areas that need to be visited infinitely often.

We start by computing the corresponding discrete-time model of the vehicle and compute its finite state abstraction $\mathbb{D}$ as described in Section 5. It can be shown that with the horizon length $N = 12$, each of the states of $\mathbb{D}$ corresponds to a cell in the road network and for any two states ν_i, ν_j of $\mathbb{D}$, $\nu_i \rightsquigarrow \nu_j$ if ν_i and ν_j are adjacent cells (i.e., they share an edge in the road network of Figure 2).

Given this finite abstraction $\mathbb{D}$ of the vehicle, we follow the approach in (Wongpiromsarn, Topcu, and Murray 2010) to synthesize the trajectory planner and the goal generator. A simulation result is shown in Figure 3 (top) where the presence of exogenous disturbances is incorporated in the embedded control software synthesis. In the first loop, there is no obstacle blocking R_2 so the vehicle picks the shorter route. In the second loop, a road blockage is added. Once the vehicle detects it, the trajectory planner cannot find a strategy for the system to satisfy its specification since the assumption on the environment that a road may not be blocked is violated. The trajectory planner then informs the goal generator of the failure. Subsequently, the goal generator recomputes a path to the areas marked by star. As a result, the vehicle continues to exhibit a correct behavior by making a U-turn and completing the task using a different path.

The result with exactly the same setup is also shown in Figure 3 (bottom) where the presence of exogenous disturbances is not incorporated in the embedded control software synthesis. Once the vehicle overtakes the obstacles on R_1, the continuous controller computes the sequence of control inputs that is expected to bring the vehicle back to its travel lane as commanded by the trajectory planner. However, due to the disturbance, the vehicle remains in the opposite lane. As a consequence, the trajectory planner continues to tell the continuous controller to bring the vehicle back to its travel lane but even though the control inputs computed by the continuous controller are supposed to bring the vehicle back to its travel lane, the vehicle remains in the opposite lane due to the disturbance. In the meantime, the disturbance also causes the vehicle to drift slowly to the right. This cycle continues, leading to violation of the desired property that the vehicle has to stay in the travel lane unless there is an obstacle blocking the lane.

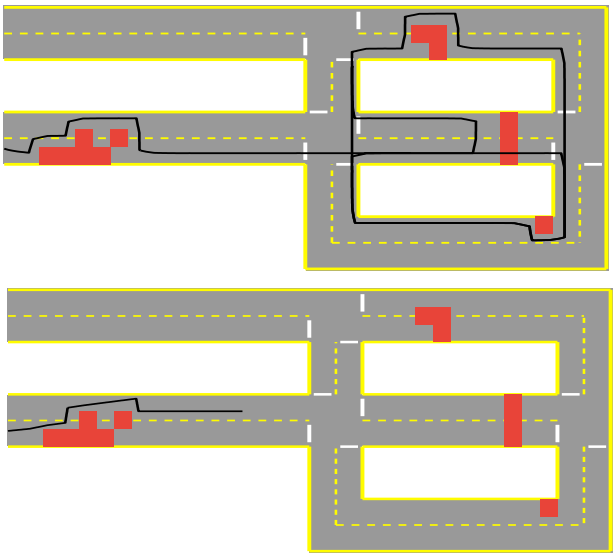

Figure 3: Simulation results with (top) the presence of disturbances incorporated, and (bottom) the presence of disturbances not incorporated in the embedded control software synthesis.

7. Conclusions

We proposed an approach to automatically compute a finite state abstraction of a discrete-time linear time-invariant system, taking into account the presence of exogenous disturbances. This allows us to automatically synthesize embedded control software for the system that is guaranteed, by construction, to satisfy its specification regardless of the environment in which it operates (subject to certain assumptions on the environment that need to be stated in the specification). The resulting system is provably robust with respect to bounded exogenous disturbances. In certain cases, the system is also capable of properly responding to failures that may occur due to violation of the environment assumptions.

Acknowledgments

This work is partially supported by AFOSR and the Boeing Corporation.

References

Alur, R.; Henzinger, T. A.; Lafferriere, G.; and Pappas, G. J. 2000. Discrete abstractions of hybrid systems. In *Proc. of the IEEE*, 971–984.

Bertsekas, D. P.; Nedić, A.; and Ozdaglar, A. E. 2003. *Convex Analysis and Optimization*. Athena Scientific.

Borrelli, F. 2003. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer.

Conner, D.; Kress-Gazit, H.; Choset, H.; Rizzi, A.; and Pappas, G. 2007. Valet parking without a valet. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 572–577.

Emerson, E. A. 1990. Temporal and modal logic. In *Handbook of theoretical computer science (vol. B): formal models and semantics*. Cambridge, MA, USA: MIT Press. 995–1072.

Girard, A., and Pappas, G. J. 2009. Brief paper: Hierarchical control system design using approximate simulation. *Automatica* 45(2):566–571.

Girard, A.; Julius, A. A.; and Pappas, G. J. 2008. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems* 18(2):163–179.

Huth, M., and Ryan, M. 2004. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2nd edition.

Karaman, S., and Frazzoli, E. 2009. Sampling-based motion planning with deterministic μ-calculus specifications. In *Proc. of IEEE Conference on Decision and Control*.

Kloetzer, M., and Belta, C. 2008. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transaction on Automatic Control* 53(1):287–297.

Kress-Gazit, H.; Fainekos, G.; and Pappas, G. 2007. Where's waldo? Sensor-based temporal logic motion planning. In *Proc. of IEEE International Conference on Robotics and Automation*, 3116–3121.

Kvasnica, M.; Grieder, P.; and Baotić, M. 2004. Multi-Parametric Toolbox (MPT). Software available at http://control.ee.ethz.ch/∼mpt/.

Löfberg, J. 2004. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*. Software available at http://control.ee.ethz.ch/∼joloef/yalmip.php.

Manna, Z., and Pnueli, A. 1992. *The temporal logic of reactive and concurrent systems*. Springer-Verlag.

Murray, R. M.; Hauser, J.; Jadbabaie, A.; Milam, M. B.; Petit, N.; Dunbar, W. B.; and Franz, R. 2003. Online control customization via optimization-based control. In *Software-Enabled Control: Information Technology for Dynamical Systems*, 149–174. Wiley-Interscience. Software available at http://www.cds.caltech.edu/∼murray/software/2002a_ntg.html.

Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2006. Synthesis of reactive(1) designs. In *Verification, Model Checking and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, 364 – 380. Springer-Verlag. Software available at http://jtlv.sourceforge.net/.

Prajna, S. 2005. *Optimization-based methods for nonlinear and hybrid systems verification*. Ph.D. Dissertation, California Institute of Technology.

Rudin, W. 1976. *Principles of mathematical analysis*. New York: McGraw-Hill Book Co., third edition.

Tabuada, P., and Pappas, G. J. 2006. Linear time logic control of linear systems. *IEEE Transaction on Automatic Control* 51(12):1862–1877.

Tanner, H., and Pappas, G. J. 2002. Simulation relations for discrete-time linear systems. In *Proc. of the IFAC World Congress on Automatic Control*, 1302–1307.

Vinter, R. 1980. A characterization of the reachable set for nonlinear control systems. *SIAM Journal on Control and Optimization* 18(6):599–610.

Wongpiromsarn, T.; Topcu, U.; and Murray, R. M. 2009. Receding horizon temporal logic planning for dynamical systems. In *Proc. of IEEE Conference on Decision and Control*.

Wongpiromsarn, T.; Topcu, U.; and Murray, R. M. 2010. Receding horizon control for temporal logic specifications. In *Proc. of the 13th International Conference on Hybrid Systems: Computation and Control*. submitted.