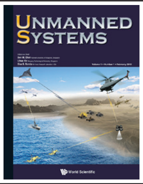
Unnmanned Systems, Vol. 2, No. 2 (2014) 201–216
© World Scientific Publishing Company
DOI: 10.1142/S2301385014300029

Unmanned Systems
http://www.worldscientific.com/worldscinet/us



Mission Accomplished: An Introduction to Formal Methods in Mobile Robot Motion Planning and Control

Hai Lin

Electrical Engineering Department, University of Notre Dame, Notre Dame, IN 46556, USA

A new trend in the robotic motion planning literature is to use formal methods, like model checking, reactive synthesis and supervisory control theory, to automatically design controllers that drive a mobile robot to accomplish some high level missions in a guaranteed manner. This is also known as the correct-by-construction method. The high level missions are usually specified as temporal logics, particularly as linear temporal logic formulas, due to their similarity to human natural languages. This paper provides a brief overview of the recent developments in this newly emerged research area. A number of fundamental topics, such as temporal logic, model checking, bisimulation quotient transition systems and reachability controller design are reviewed. Additionally, the key challenges and possible future directions in this area are briefly discussed with references given for further reading.

Keywords: Mobile robots; symbolic motion planning; correct-by-construction; hybrid control; cyber-physical systems.

1. Introduction

The current design process for robotic systems, especially in the high-level decision making and coordinations among multiple robots, is still of trial and error nature [44]. Engineers design a robotic system, and then test it in all conceivable scenarios that could be mimicked. Once a bug was found, the design is revised accordingly, and then the system is tested again. This design and test cycle repeats until no more errors were found and the engineer believes that the robots are reliable enough. However, even after an extensive period of testing, it is very possible that some bugs still remain undiscovered [47]. In addition, if a bug was found at a very late stage in the design process, it could be very costly to fix it as it may request an overhaul of the whole design. Furthermore, should the application scenarios change or new functions need to be added, we may have to repeat this time-consuming and error-prone design process. Besides these drawbacks, this trial and error design process is not suitable for some safety critical or emergency response applications, such as searching and removing mass destruction weapons [59] or cleaning-up in a meltdown nuclear power plant area [8], where we do not have a second chance and the consequences of errors are definitely unacceptable. Hence, it is of great needs to have formal design approaches to robotic system synthesis that can guarantee the performance and correctness of the designed robotic systems.

Motivated by these challenges, a recent trend in the robotic motion planning and control is to use formal methods, like model checking, reactive synthesis and supervisory control theory, to automatically generate controllers that, by construction, guarantee the robot to satisfy high-level missions [2, 5, 17, 20]. As an example of these high-level specifications, in a search and rescue mission, certain regions may need to be searched in a particular order and once a survivor was found, at least one robot will stay for assistance and stream back the location and situation to the base station for the human task force. It is nontrivial to convert these kinds of high-level missions into a sequence of "go from A to B while avoiding C" type of traditional motion planning primitives, but they can be readily captured by temporal logic formulas. Then, the design problem considered here can be generally stated as follows: Given a temporal logic specification, design low-level primitives, such as feedback controllers, coordination

Received 25 February 2014; Revised 29 March 2014; Accepted 1 April 2014; Published 21 April 2014. This paper was recommended for publication in its revised form by the editors-in-chief.
Email Address: *hlin1@nd.edu*

rules and communication protocols, for robots to accomplish the high-level mission. To distinguish from traditional motion planning, this is known as symbolic motion planning in the literature, see e.g., [2, 5, 11, 18, 19, 41, 46, 58, 77] and the references therein.

The basic design procedure for symbolic motion planning consists of the following steps. First, a finite abstracted model of the robotic system is obtained. Then, the design is carried out in the discrete domain and uses methods like model checking, reactive synthesis and/or supervisory control theory to generate feasible runs consisting of sequences of discrete (symbolic) states that satisfy the temporal logic specifications in concern. Finally, the generated sequence of discrete states or symbols are used by the continuous layer to construct continuous feedback control laws to drive the robot (or physically feasible trajectories for robots to follow). The critical step, also the most difficult part, of symbolic motion planning is how to obtain an abstraction of the robotic dynamics and environment. The abstracted finite model should be constructed in such a way that once one can find a run in the abstracted model satisfying the specification then there must exist corresponding continuous trajectories for the original robotic system satisfying the same specification. Most of research efforts in the literature have been devoted to answering the abstraction problem, using simulation-, bisimulation- and approximate bisimulation-based abstraction [1, 5, 73]. Since the temporal logic, abstracted model and discrete planning are all in the discrete domain while the robot dynamics follow continuous differential equations with physical continuous constraints, the controllers being designed must be of hybrid nature [54]. The salient feature of symbolic motion planning methods is that they can guarantee the satisfaction of temporal specifications provided that there exist low-level primitive feedback controllers to implement the actions requested by the hybrid controller.

This paper aims to provide a brief overview of the recent developments in this newly emerged multi-disciplinary research area. For such a purpose, the rest of paper is organized as follows. First, we formally state the symbolic motion planning problem in Sec. 2, with a brief explanation of the hierarchical control stack that is usually adopted in symbolic motion planning design. Then, Sec. 3 gives a quick review of some basic concepts like the labeled transition system, simulation and bisimulation relations and bisimilar quotient transition systems. The symbolic motion planning problem is investigated in Secs. 4 and 5. In particular, Sec. 4 considers a simple case where the robot dynamics can be modeled by a set of linear differential equations or approximated by a piecewise affine system. As an extension, the symbolic motion planning with a more general nonlinear dynamical model is discussed in Sec. 5 based on the ideas of approximate bisimulation and interface design. Finally, possible future directions, such as motion planning in uncertain environments, relaxing the perfect sensing and actuation assumptions, optimal symbolic control and multi-robot coordination, together with their key challenges and some promising approaches, are discussed in Sec. 6.

2. Symbolic Motion Planning Problem

We consider a robot that is modeled by the following continuous variable dynamical system Σ (dynamics model)

$$\Sigma : \begin{cases} \dot{x}(t) = f(x(t), u(t)), \\ y(t) = g(x(t)), \end{cases} \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state of the robot, $u(t) \in \mathbb{R}^r$ is the control input and $y(t) \in \mathbb{R}^p$ is the observed output of the robot such as its position, orientation and so on.

It is assumed that the initial condition x_0 is restricted to a certain region in the state space $X_0 \subseteq \mathbb{R}^n$. The goal is to design a controller such that the output $y(t)$ generated by the closed-loop system satisfies a given high level temporal specification. For example, one may request that the output $y(t)$ visits certain regions in the output (robot configuration) space $\mathbb{R}^p$ with a specific order while avoiding some forbidden regions in $\mathbb{R}^p$, e.g., obstacles.

To capture such requirements formally, we adopt the temporal logic over reals (RTL) [68] and define a set of atomic propositions as $\Pi = \{\pi_0, \pi_1, \ldots, \pi_m\}$, which is a finite set of symbols that label these regions of interest. Furthermore, the denotation $[\![\cdot]\!]$ of each symbol is defined as the region labeled by π, i.e., $[\![\pi_i]\!] \subseteq \mathbb{R}^p$ for any π_i in Π. The atom π_i is true if and only if $y(t)$ is in $[\![\pi_i]\!]$. The symbol π_0 is reserved for initial conditions, $y(0) \in [\![\pi_0]\!]$, and correspondingly the state starts from $x(0) \in X_0 \subseteq \mathbb{R}^n$. Here we are more interested in deciding whether the robot is in certain regions of interest than the exact position of the robot *per se*. For example, in the study of collision avoidance, we request the robot never visit the regions standing for the obstacles or two robots are never within the same region. It is also assumed that all these regions are given as polytopes, which is rather standard in the motion planning literature as the union of polytopes can be used to approximate the workspace of robots in an arbitrarily close manner.

With these notations, the syntax of the propositional RTL [68] can be formally introduced as below.

Definition 1. Let Π be a finite set of atomic propositions, i.e., $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, standing for the regions of interest. The set of all well-formed propositional RTL are recursively defined from predicates in Π according to the following rules

1. `true`, `false`, and π_i are RTL formulas for all $\pi_i \in \Pi$;

2. if φ_1 and φ_2 are RTL formulas, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ and $\neg\varphi_1$ are RTL formulas;
3. if φ_1 and φ_2 are RTL formulas, then $\varphi_1 \mathcal{U} \varphi_2$ and $\varphi_1 \mathcal{R} \varphi_2$ are RTL formulas

Formally, the semantics of RTL formulas are defined over continuous time signals. Given a function $y : \mathbb{R}^+ \to \mathbb{R}^r$, we define $y|_t$ to be the t time shift of y with definition $y|_t(s) = y(s + t)$ for all $s \in \mathbb{R}^+$.

Definition 2. Let $y(t)$ be a function $y : \mathbb{R}^+ \to \mathbb{R}^r$, and Π a finite set of atomic propositions, i.e., $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ with atomic mapping $[\![\cdot]\!]$. For $t, s \in \mathbb{R}^+$, the semantics of an RTL formula over Π can be defined as

1. $(y, [\![\cdot]\!]) \models \pi_i$ iff $y(0) \in [\![\pi_i]\!]$,
2. $(y, [\![\cdot]\!]) \models \neg p$ iff $y(0) \notin [\![\pi_i]\!]$,
3. $(y, [\![\cdot]\!]) \models \varphi_1 \wedge \varphi_2$ if $(y, [\![\cdot]\!]) \models \varphi_1$ and $(y, [\![\cdot]\!]) \models \varphi_2$,
4. $(y, [\![\cdot]\!]) \models \varphi_1 \vee \varphi_2$ if $(y, [\![\cdot]\!]) \models \varphi_1$ or $(y, [\![\cdot]\!]) \models \varphi_2$,
5. $(y, [\![\cdot]\!]) \models \varphi_1 \mathcal{U} \varphi_2$ if there exists $t \geq 0$ such that $(y|_t, [\![\cdot]\!]) \models \varphi_2$ and for all s with $0 \leq s \leq t$ we have $(y|_s, [\![\cdot]\!]) \models \varphi_1$,
6. $(y, [\![\cdot]\!]) \models \varphi_1 \mathcal{R} \varphi_2$ if for all $t \geq 0$ it is $(y|_t, [\![\cdot]\!]) \models \varphi_2$ or there exists some s such that $0 \leq s \leq t$ and $(y|_s, [\![\cdot]\!]) \models \varphi_1$.

Intuitively speaking, the formula $\varphi_1 \mathcal{U} \varphi_2$ expresses the property that over the trajectory $y(t)$, φ_1 is true until φ_2 becomes true. On the contrary, the release operator $\varphi_1 \mathcal{R} \varphi_2$ means that φ_2 should hold true and be released when φ_1 becomes true. Furthermore, we can derive several additional temporal operators such as

- $\Diamond\varphi = \texttt{true}\ \mathcal{U}\varphi$ means that the sub-formula φ *eventually* becomes true for a trajectory $y(t)$,
- $\Box\varphi = \neg\Diamond\neg\varphi$ indicates that φ *always* holds true for $y(t)$.

The following examples from [18] illustrate some typical control specifications that can be formulated as temporal logic formulas based on the atomic proposition set Π.

- Reachability while avoiding regions: The formula $\neg(\pi_1 \vee \pi_2) \sqcup \pi_3$ represents the requirement that the output finally reaches the region $[\![\pi_3]\!]$ while keeping away from the regions $[\![\pi_1]\!]$ and $[\![\pi_2]\!]$;
- Sequencing: The formula $\Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond\pi_3))$ represents the requirement that the output reaches the regions $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ in order;
- Coverage: The formula $\Diamond\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3$ represents the requirement that the output eventually reaches all the regions $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ without any particular order;
- Recurrence (Liveness): The formula $\Box(\Diamond\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3)$ represents the requirement that the output reaches these regions $[\![\pi_1]\!]$, $[\![\pi_2]\!]$ and $[\![\pi_3]\!]$ infinitely often.

More complicated specifications can be composed from the basic specifications using the logic operators. Now the temporal motion planning problem for robots can be stated as follows:

Problem 1. *Given the robot dynamics model Σ and an RTL formula ϕ, design control signals $u(t)$ such that the trajectories of the closed-loop system satisfy the formula ϕ.*

The difficulty lies in the fact that the specification is given as discrete logic formulas, while the plant to be controlled contains continuous-variable dynamics. To bridge the gap, a hierarchical control structure consisting of a discrete planning layer on top of a continuous implementation layer is usually adopted. The basic idea is to obtain an equivalent finite abstractions of the continuous dynamics of robots first, upon which a discrete synthesis- or searching-based algorithm is performed on the discrete planning layer using model checking, reactive synthesis or discrete event supervisory control approaches. The design is with respect to the temporal logic specification ϕ and its output is a sequence of regions (in symbols) to be visited by the robot in order. Next, this sequence of regions is converted to a hybrid controller (since the controller usually contains both discrete transitions and continuous flows) to drive the robot visiting these regions in order. Interested readers may refer to [54] for a recent survey on hybrid dynamical systems.

The effectiveness of the abstraction-based method depends on whether or not there exists an equivalent finite state discrete abstracted model for the original robotic continuous dynamic systems. The equivalence is in the sense that once one can find a run in the abstracted model satisfying the specification then there must exist corresponding continuous trajectories for the original robotic system satisfying the same specification. Therefore, significant research efforts have been devoted to developing such an equivalent abstract models, mainly using simulation-, bisimulation- and approximate bisimulation-based abstraction, see e.g., [1, 5, 73]. In the next section, we will give a brief review of the simulation and bisimulation relation on transition systems.

3. Simulation, Bisimulation and Abstractions

Some basic concepts and notations on simulation and bisimulation that are widely used in the symbolic motion planning literature will be reviewed here. Here, we follow the notations in [4]. Readers who are familiar with these notations may simply skip this section.

3.1. *Transition systems*

Transition systems are graph models that describe the evolution of the states under the action of transitions.

Definition 3. A *transition system* T is a four tuple $T = (S, S_0, U, \rightarrow)$ defined by

- A set of states S,
- A set of initial states $S_0 \subseteq S$,
- A set of actions U,
- A transition relation $\rightarrow \subseteq S \times U \times S$.

A transition system is called finite when the state set S and the action set U contain only a finite number of elements. Clearly, a finite automaton can be cast as a transition system with finite states. A transition system may have infinite number of states, and can be used to represent a large class of dynamical systems, such as continuous control systems and hybrid dynamical systems [54, 73]. The dynamical behavior of a transition system is conveniently described by the strings of its state evolution. Formally, we have the following definitions and notations.

Definition 4. A string $\alpha \in S^*$ ($\alpha \in S^\omega$) is a *run* of transition system $T = (S, S_0, U, \rightarrow)$ if

1. $\alpha(1) \in S_0$,
2. there exists a string $\beta \in U^*$ ($\beta \in U^\omega$) such that $(\alpha(i), \beta(i), \alpha(i+1)) \in \rightarrow$, for $i = 1, \ldots, |\alpha| - 1$ ($i \geq 1$ for $\beta \in U^\omega$).

Note that S^* stands for the set of all strings of finite length with symbols from S, while S^ω stands for the set of all strings of infinite length with symbols from S. The notation $|\alpha|$ stands for the length of the run α, which potentially contains infinite number of transitions, i.e., $|\alpha| \in \mathbb{N} \cup \{\omega\}$. Here $\mathbb{N}$ denotes the set of natural numbers and ω stands for infinity. For $i < |\alpha|$, the ith state of α, written as $\alpha(i)$, is the state s_i reached after i transitions. A complete execution is a run which is maximal, that is, which cannot be extended. It is either infinite, or it ends in a state s_n out of which no transition is defined. If the second case happens, we call it a deadlock.

Consider a transition system $T = (S, S_0, U, \rightarrow)$. For a particular state $s \in S$ and action $a \in U$, the set of successor states of s by action a are given by $\text{post}_a(s) = \{s' \in S | (s, a, s') \in \rightarrow\}$. The successor states of s in T for all possible actions is $\text{post}(s) = \cup_{a \in U} \text{post}_a(s)$.

Actions can be seen as inputs, and we can also introduce outputs for transition systems. Instead, we call them labels, which associate the states of a transition system with properties that hold true for the corresponding states. The properties of interest are denoted as symbols p_i, say p_1 = "the machine is busy," p_2 = "the machine is broken" and so on. The collection of such symbols (assumed to be finite) forms a set, denoted as $\mathcal{P} = \{p_1, p_2, \ldots, \}$ and called an atomic proposition set. A labeled transition system is a transition system with all its states being labeled with true or false for *atomic propositions* in $\mathcal{P}$.

Definition 5. A *labeled transition system* is a tuple (T, l), where $T = (S, S_0, U, \rightarrow)$ is a transition system and $l : S \rightarrow 2^{\mathcal{P}}$ (where $2^{\mathcal{P}}$ stands for the collection of all subsets of $\mathcal{P}$) is a label function that assigns each state s in T a subset of predicates $l(s) \subseteq \mathcal{P}$ satisfied by the state s.

Given a finite run α of the transition system T, we can define a *trace* generated from the labeled transition system (T, l) corresponding to the run α as a string $\gamma \in (2^{\mathcal{P}})^*$, where $\gamma(i) = l(\alpha(i))$. The collection of all finite traces that can be generated by the labeled transition system (T, l) is called the trace generated by (T, l), denoted as $\mathcal{T}(T, l)$. Note that the above definitions can be extended to the case where α is of infinite length. Then, γ is an infinite trace as defined above, i.e., $\gamma \in (2^{\mathcal{P}})^\omega$, and the collection of all such infinite length traces is called the ω-*trace* generated by (T, l), denoted as $\mathcal{T}_\omega(T, l)$.

3.2. *Simulation relation*

In particular, we focus on abstraction-based approaches by obtaining equivalent quotient transition systems that satisfy the same temporal logics. Here the quotient is taken with respect to simulation or bisimulation equivalences [61] defined for labeled transition systems as below.

Definition 6. Let $T_i = (S_i, S_i^0, \rightarrow)$ with $i = 1, 2$ be two transition systems,[a] and $l_i : S_i \rightarrow 2^{\mathcal{P}}$ labels their states, respectively. A relation $R \subseteq S_1 \times S_2$ is said to be a *simulation relationship* from labeled transition system (T_1, l_1) to (T_2, l_2) if the following hold:

1. For any pair $(s_1, s_2) \in R$, their labels are the same, i.e., $l_1(s_1) = l_2(s_2)$,
2. For any initial state $s_1 \in S_1^0$, there exists $s_2 \in S_2^0$ such that $(s_1, s_2) \in R$,
3. For any pair $(s_1, s_2) \in R$, if $s_1' \in \text{post}(s_1)$ in T_1 then there exists $s_2' \in S_2$ such that $s_2' \in \text{post}(s_2)$ in T_2 and $(s_1', s_2') \in R$.

Intuitively, a labeled transition system simulates another system if, for every run in the simulated system, there is a matching (w.r.t. labels) computation in the simulating system. If there exists a simulation relationship R from labeled transition system (T_1, l_1) to (T_2, l_2), we also say that (T_1, l_1) is simulated by (T_2, l_2), or (T_2, l_2) simulates (T_1, l_1), denoted as $(T_1, l_1) \prec_R (T_2, l_2)$, since for any trace in (T_1, l_1) one can find a corresponding equivalent trace in (T_2, l_2). Hence, if $(T_1, l_1) \prec_R (T_2, l_2)$, which also results $\mathcal{T}(T_1, l_1) \subseteq \mathcal{T}(T_2, l_2)$ and $\mathcal{T}_\omega(T_1, l_1) \subseteq \mathcal{T}_\omega(T_2, l_2)$.

[a] Since we adopt the label-based simulation relation and only concern the existence of actions, it is possible to omit the definition of action sets U_i and treat all actions equally.

Relation R is said to be bisimulation relation between (T_1, l_1) and (T_2, l_2) if R is a simulation relation from (T_1, l_1) to (T_2, l_2) and R^{-1} is a simulation relation from (T_2, l_2) to (T_1, l_1), i.e., $(T_2, l_2) \prec_{R^{-1}} (T_1, l_1)$. If such a bi-simulation relation R exists between (T_1, l_1) and (T_2, l_2), then we say that (T_1, l_1) is *bisimilar* to (T_2, l_2), denoted as $(T_1, l_1) \cong_{R \cup R^{-1}} (T_2, l_2)$. Usually, we only care about the existence of such an R, so we usually omit R and simply write $(T_1, l_1) \prec (T_2, l_2)$ or $(T_1, l_1) \cong (T_2, l_2)$. Since $(T_1, l_1) \cong (T_2, l_2)$ implies both $(T_1, l_1) \prec (T_2, l_2)$ and $(T_2, l_2) \prec (T_1, l_1)$, so two bisimilar labeled transition systems are trace equivalent, i.e., $(T_1, l_1) \cong (T_2, l_2)$ implies $\mathcal{T}(T_1, l_1) = \mathcal{T}(T_2, l_2)$ and $\mathcal{T}_\omega(T_1, l_1) = \mathcal{T}_\omega(T_2, l_2)$.

3.3. *Bisimulation quotient*

To reduce the computational complexity of model checking, we try to reduce the size of the state space of a transition system by clustering all bisimilar equivalent states. For such a purpose, we introduce self-bisimulation relation for a labeled transition system first.

Definition 7. Consider a labeled transition system (T, l) and a binary relationship $R \subseteq S \times S$. The relation R is called a *self-bisimulation relation* for (T, l) if the following hold:

- $\forall (s_1, s_2) \in R : l(s_1) = l(s_2)$.
- $\forall s_1' \in \text{post}(s_1), \exists s_2' \in \text{post}(s_2)$ with $(s_1', s_2') \in R$.
- $\forall s_2' \in \text{post}(s_2), \exists s_1' \in \text{post}(s_1)$ with $(s_1', s_2') \in R$.

States s_1 and s_2 are bisimilar denoted as $s_1 \sim s_2$ if there exists such a binary relation R, defined in the above definition, in (T, l). The bisimulation relation defined above forms an equivalence relation as it is reflexive, symmetric and transitive. Since R is an equivalence relation on S, it therefore induces a partition of the state set into a number of equivalent classes $S = \bigcup_{s \in S} [s]_R$, where $[s]_R$ is a collection if all bisimilar states to s, namely $[s]_R = \{s' \in S \mid (s, s') \in R\}$.

It can be easily shown that for any $s, s' \in S$

- $s \in [s]_R$,
- if $s' \in [s]_R$, then $[s]_R = [s']_R$,
- if $s' \notin [s]_R$, then $[s]_R \cap [s']_R = \emptyset$,
- $S = \cup_{s \in S} [s]_R$.

Hence, the bisimulation relation R does induce a partition of the state set S. For simplicity, we denote such a partition as $S/_R$, which stands for the quotient space, i.e., the set consisting of all equivalent classes. Based on the quotient space, we can define a quotient transition system as below.

Definition 8. Given a labeled transition system (T, l) and a bisimulation relation $R \subseteq S \times S$, the *quotient transition system* can be defined as $T/_R = (S/_R, S^0/_R, \to_R)$ where $S/_R = \{[s]_R\}_{s \in S}$, the initial states

$$S^0/_R = \{[s]_R \in S/_R : S^0 \cap [s]_R \neq \emptyset\},$$

and for $[s_1]_R, [s_2]_R \in S/_R$, $([s_1]_R, [s_2]_R) \in \to_R$ if and only if there exist $s_1 \in [s_1]_R$ and $s_2 \in [s_2]_R$ such that $(s_1, s_2) \in \to$. The new label map $l_R : S/_R \to 2^{\mathcal{P}}$ is defined as $l_R([s]_R) = l(s)$.

Note that since if $s' \in [s]_R$ then $l(s') = l(s)$ by definition, the new label map l_R is well defined. Also, it can be checked that there exists a relation on $S \times S/_R$ that satisfies the bisimulation relation definition. Particularly, we can choose $\mathcal{R} \subseteq S \times S/_R$ with $(s', [s]_R) \in \mathcal{R}$ if and only if $s' \in [s]_R$. Then, we can conclude that $(T, l) \cong_{\mathcal{R} \cup \mathcal{R}^{-1}} (T/_R, l_R)$, see e.g., [4, 12] for more detailed discussions. Hence, $(T, l) \cong (T/_R, l_R)$. The hope is that the quotient transition systems $T/_R$ has much fewer number of states compared to the original transition system T, so the model checking and synthesis problems can be significantly simplified.

4. Motion Planning Under Affine Dynamics

Now, we are ready to study the symbolic motion planning problem formulated in Sec. 2. We first consider the case when the robot dynamics model Σ, described in Eq. (1), can be approximated as an affine control system, i.e.,

$$\begin{cases} \dot{x}(t) = Ax(t) + a + Bu(t), \\ y(t) = Cx(t), \end{cases} \tag{2}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system, $u(t) \in \mathbb{R}^m$ is the control input and $y(t) \in \mathbb{R}^p$ is the observed output of the system, while $A \in \mathbb{R}^{n \times n}$, $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{p \times n}$ are state matrices. The results in this section are mainly based on [27, 40], where it is assumed that all regions of interest are bounded polyhedral sets.

4.1. *Affine functions on simplices*

A *polyhedral set* P is a subset of $\mathbb{R}^N$, described by a finite number of linear inequalities. Namely, there exist K nonzero vectors $n_1, \ldots, n_K \in \mathbb{R}^N$ and scalars $\alpha_1, \ldots, \alpha_K \in \mathbb{R}$, such that

$$P = \{x \in \mathbb{R}^N \mid \forall i = 1, \ldots, K : n_i^T x \leq \alpha_i\}. \tag{3}$$

Each linear inequality $n_i^T x \leq \alpha_i$ forms a half-space, so this is called the *half-space representation* of a polyhedral set P. Correspondingly, the hyperplane formed by $n_i^T x = \alpha_i$ is called a *supporting hyperplane* for P. A bounded polyhedral set is called a *polytope*. A polytope can alternatively be characterized as the convex hull of a finite number of points, $v_1, \dots, v_M \in \mathbb{R}^N$,

$$P = \left\{ x \in \mathbb{R}^N | x = \sum_{i=1}^{M} \lambda_i v_i, \sum_{i=1}^{M} \lambda_i = 1, \lambda_i \geq 0 \right\}, \qquad (4)$$

where $v_1, \dots, v_M$ are called vertices of the polytope P. Equation (4) is called the vertex representation of a polytope.

The half-space representation and the vertex representation for a polytope are equivalent and can be transformed from one to the other [80]. Which definition should be employed depends on what is more suitable for the problem at hand. The intersection of a polytope P with one of its supporting hyperplanes

$$F_i = \{x \in \mathbb{R}^N | n_i^T x = \alpha_i\} \cap P \qquad (5)$$

is called a facet of P, if the dimension of the intersection is equal to $N - 1$. The vector n_i is the normal vector of the facet F_i, $(i = 1, \dots, K)$, and, by convention, n_i is of unit length and always points out the polytope P. Also, the corresponding hyperplane $n_i^T x = \alpha_i$ is called a *bounding hyperplane* for P. Later on, we assume that each facet of P corresponds to exactly one n_i, and vice versa. Namely, each n_i corresponds to a bounding hyperplane and is linearly independent of other norm vectors.

The set of vertices of a polytope P is denoted by $\mathcal{V}(P)$. Given a vertex $v \in \mathcal{V}(P)$, we denote by $\mathcal{F}(v)$ the set of all facets containing v. It can be shown that for a polytope there are at least $N + 1$ vertices, i.e., $M \geq N + 1$. A polytope that has exactly $N + 1$ vertices is called a *simplex* in $\mathbb{R}^N$. Any full dimensional polytope can be triangularized [50]. In other words, for any full dimensional polytope P, there exists full dimensional simplex $S_1, \dots, S_L$ such that: (1) $P = \cup_{i=1,2,\dots,L} S_i$, (2) $S_i \cap S_j$ is either empty or a common facet of S_i and S_j, for all $i, j = 1, \dots, L$ with $i \neq j$. The set of vertices of simplex S_i is a subset of $\{v_1, v_2, \dots, v_M\}$ for all $i = 1, \dots, L$.

A reason for restricting polytopes is due to the fact that an affine function's value inside a polytope P will be uniquely determined by its values at the vertices of P. An affine function $f : \mathbb{R}^N \to \mathbb{R}^r$ is of the form

$$f(x) = Ax + b \qquad (6)$$

for constant matrices $A \in \mathbb{R}^{r \times N}$ and vector $b \in \mathbb{R}^r$. Due to convexity and linearity, it is easy to obtain:

Lemma 1. *Let $w \in \mathbb{R}^r$ and $d \in \mathbb{R}$. Then $w^T f(x) > d$ everywhere in a polytope P, if and only if the inequality holds at all the vertices of P, i.e., $w^T f(v_i) > d$ for $i = 1, \dots, N + 1$.*

It is easy to see that the result remains valid if $>$ is replaced by $\geq, =, <, \leq$. Also, the result remains valid if f is restricted to a facet F_i. The following result tells us that the function value of an affine function on polytope P is completely determined by the values of f on the vertices of P.

Lemma 2 ([27]). *Let P be a polytope in $\mathbb{R}^N$ and $f : \mathbb{R}^N \to \mathbb{R}^r$ an affine function. The function value of $x \in P$, $f(x)$ is completely determined by the values of f on the vertices of P, $f(v_i) = g_i$, $i = 1, \dots, M$, i.e., $x \in P \Rightarrow f(x) = \sum_{v \in \mathcal{V}(P)} \lambda_v f(v)$, $\sum_{v \in \mathcal{V}(P)} \lambda_v = 1$, $\lambda_v \geq 0$.*

Moreover, f is unique. Assume not, and there is another affine function $f' : \mathbb{R}^N \to \mathbb{R}^r$ satisfying $f'|_{\mathcal{V}(P)} = g$. Then, consider $f - f'$, which is affine and $(f - f')|_{\mathcal{V}(P)} = g - g = 0$. It then follows from the previous lemma that $f - f'$ is the function identically zero and thus $f = f'$. In particular, we can present an explicit reconstruction for the case of an affine function on a simplex.

Lemma 3 ([27]). *Let S_N be a simplex in $\mathbb{R}^N$ and $f : \mathbb{R}^N \to \mathbb{R}^r$ an affine function. The restriction of f to S_N is a convex combination of its values at the vertices, and is given by*

$$f(x) = GW^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad x \in P, \qquad (7)$$

where

$$G = [g_1 \quad \cdots \quad g_{N+1}],$$
$$W = \begin{bmatrix} v_1 & \cdots & v_{N+1} \\ 1 & \cdots & 1 \end{bmatrix}$$

are $r \times (N + 1)$ and $(N + 1) \times (N + 1)$ real matrices, respectively.

4.2. *Two basic control problems*

For the linear control system (2), it is assumed that all the regions are given as polytopes with $[\![\pi_i]\!] = \{y \in \mathbb{R}^p | n_i^T y \leq g_i\}$, where $n_i \in \mathbb{R}^p$ and $g_i \in \mathbb{R}$. Note that $y(t) \in [\![\pi_i]\!]$ holds true if and only if $n_i^T Cx(t) \leq g_i$, which corresponds to a polytope in the state space $P_i = \{x \in \mathbb{R}^n | n_i^T Cx \leq g_i\}$.

We restrict our control $u(t)$ to be in the form of an affine function of state, i.e., $u(t) = kx(t) + v$, where k and v are to be designed. The closed-loop dynamics will be

$$\dot{x}(t) = (A + Bk)x(t) + w, \qquad (8)$$

which has an affine function on its right-hand side with $w = Bv + a$. Our task is to design the controller u, i.e., k and v, in such a way that all trajectories of the closed-loop system starting from a polytope P_i can either all stay inside P_i or all transit to a neighboring polytope P_j after a finite period of time without intersecting with another region during this transition process. The closed-loop system becomes a piecewise affine system with polytopes as the partition of the state space, which represents a special class of a hybrid system.

For each polytope, the following two problems are considered:

- *Invariant Control Problem:* The invariant control problem for the linear control system (2) with respect to a polytope P seeks an affine feedback control law $u = kx + v$ such that all trajectories for the closed-loop system $\xi(t)$ starting from P will remain in P forever, i.e., $\xi(0) \in P \Rightarrow \xi(t) \in P, \forall t \in \mathbb{R}^+$.
- *Control to Facet Problem:* Consider the linear control system (2) on a polytope P, and let F be a facet of P. The control to facet problem is to determine whether there exists an affine feedback control law $u = kx + v$ such that all trajectories for the closed-loop system starting from P will leave P through F after a finite time τ. Namely, there exists a finite escape time $\tau > 0$ such that the following hold
  - $\xi(t) \in P$ for $0 \leq t < \tau$,
  - $\xi(\tau) \in F$,
  - $\exists \epsilon > 0$ such that $\xi(t) \notin P \cup F$ for $\tau < t < \tau + \epsilon$.

Since the value of an affine function in a polytope can be determined by its values at the vertices of the polytope, the existence of such an affine feedback can be determined by some linear inequalities evaluated at (a finite number of) vertices of the polytope P. The following proposition characterizes all affine vector fields for which the polytope is an invariant.

Theorem 1 ([27, 40]). *The invariant control problem for the linear control system* (2) *with respect to a polytope P is solvable provided that the following sets are non-empty*

$$U_P(v_i) = \bigcap_{F \in \mathcal{F}(v_i)} \{u \in \mathbb{R}^m |\ \eta_F^T(Av_i + a + Bu) < 0\}$$

for all $v_i \in \mathcal{V}(P)$, where η_F is the normal vector for the facet F.

It is a very intuitive condition, and basically requests the existence of a control signal to make the vector field point inside to the polytope P for all vertices. The conditions form a collection of linear inequalities can be easily checked. Moreover, when the sets are nonempty, a multi-affine control law $u = kx + v$ can be constructed, with the control value at vertex v_i being any element in $U_P(v_i)$. Similarly, the control to facet problem also admits a simple solution. This is illustrated in Fig. 1.

Theorem 2 ([27, 40]). *Let P be a polytope in $\mathbb{R}^n$ with a facet F and $\dot{x} = Ax + a + Bu$ be a linear control system. The control to facet problem admits a solution if the following sets are nonempty*

$$U_P(v_i) = \bigcap_{G \in \mathcal{F}(v_i)} \{u \in \mathbb{R}^m | \eta_G^T(Av_i + a + Bu) < 0\}$$

for all $v_i \in \mathcal{V}(P)$ such that $F \notin \mathcal{F}(v_i)$, and

$$U_P(v_i) = \bigcap_{G \in \mathcal{F}(v_i), G \neq F} \left\{ u \in \mathbb{R}^m \,\middle|\, \begin{array}{l} \eta_G^T(Av_i + a + Bu) < 0 \\ \eta_F^T(Av_i + a + Bu) > 0 \end{array} \right\}$$

for all $v_i \in \mathcal{V}(P)$ such that $F \in \mathcal{F}(v_i)$.

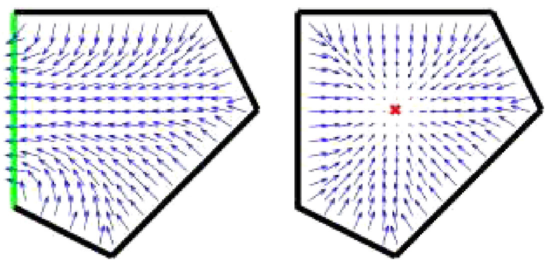


Fig. 1. Illustration of the (a) control to facet problem and (b) invariant control problem [18].

Intuitively speaking, the sufficient conditions above force the vector fields of the closed-loop system point inside to P for all vertices except the vertices of the facet F. It therefore guarantees that all trajectory starting from P will exit P through the facet F within finite time durations. Hence, such a control solves the "Control to Facet Problem." This is illustrated in Fig. 1.

Once the above two control problems are solved, we can deduce a finite labeled transition system that is bisimilar to the closed-loop continuous system (seen as an infinite state transition system with label mapping consistent with $[\![\cdot]\!]$). The construction of the finite abstraction is conceptually simple. Note that the specification regions $[\![\pi_i]\!]$ for the output $y(t)$ are assumed to be polytopes and mutually exclusive and only share facets if adjacent. Correspondingly, $[\![\pi_i]\!]$ will imply a collection of polyhedrons in R^n, denoted as P_i. All states in P_i are labeled with π_i for consistency. The collection of all such P_i forms the set of discrete states, while the initial state is the polytope containing x_0, i.e., $x_0 \in P_0$. There is a transition from P_i to P_j, i.e., $(P_i, P_j) \in \rightarrow_P$ (assume $i \neq j$), if they share a facet F and the control to the facet F is solvable for P_i. There is a self-loop transition for a polytope P_i, i.e., $(P_i, P_i) \in \rightarrow_P$, if the invariant control problem is feasible for the polytope P_i. Hence, we obtain a transition system $T_P = (\{P_i\}, \{P_0\}, \rightarrow_P)$, which has finite states. The symbol π_i is then used to label the discrete state

P_i. It is not difficult to see that the deduced labeled transition system is bisimilar to the continuous control system (2). Actually, the relation $R \subseteq \{P_i\} \times \mathbb{R}^n$, defined by $(P_i, x) \in R$ if $x \in P_i$, together with its reverse, forms a bisimulation relation between T_P and (2).

4.3. *Discrete motion planning*

After obtaining the finite abstraction model T_P, one can apply model checking methods [4,12] or supervisory control techniques [9,67] to design a sequence of region transitions such that the required RTL specifications are satisfied. Due to bisimulation relation between the linear control system and the abstracted model, the sequence of discrete region transitions can be mimicked by continuous trajectories $x(t)$, i.e., there exist continuous control signals to drive the output $y(t)$ so to satisfy the RTL specifications. Furthermore, continuous control signals can be designed based on Theorems 1 and 2. For example, if a self-loop transition for region π_i occurs, then the invariant control law for region P_i can be designed based on results in Theorem 1. On the other hand, if there is a transition from region π_i to π_j, the control law proposed in Theorem 2 can be adopted to achieve the region transition.

In particular, we give some details on the model checking-based methods in this subsection. The main idea is to reduce the model checking problem to an inclusion problem between automata [74]. In particular, Büchi automata are employed.

Definition 9 (Büchi Automaton). A Büchi automaton B is an automaton $(Q, Q^0, \Sigma, \delta, F)$, where $F \subseteq Q$ is a set of final states. A string $\alpha \in Q^\omega$ is a run of B if there exists $\beta \in \Sigma^\omega$ such that

1. $\alpha(1) \in Q^0$,
2. $\alpha(i+1) \in \delta(\alpha(i), \beta(i))$, for all $i \in \mathbb{N}$, and
3. there exists infinitely many $j \in \mathbb{N}$ such that $\alpha(j) \in F$.

The language recognized or accepted by B is the collection of all such β, called ω-language and denoted by $\mathcal{L}_\omega(B)$.

To do model checking based on automata theory, we first convert the labeled transition system (T, l) as a Büchi automaton B_T, such that the languages accepted by the Büchi automaton B_T, denoted as $\mathcal{L}_\omega(B_T)$, coincide with the set of ω-traces generated from (T, l), i.e., $\mathcal{L}_\omega(B_T) = \mathcal{T}_\omega(T, l)$. The conversion is very straightforward and can be described as follows.

Given a labeled transition system (T, l), with $T = (Q, Q_0, E, \rightarrow)$ and $l : Q \rightarrow 2^{\mathcal{P}}$, it can be transformed into a Büchi automaton $B_T = (Q \cup \{t\}, \{t\}, \Sigma, \delta, Q \cup \{t\})$, where

- $\Sigma = 2^{\mathcal{P}}$
- for any $q \in Q$, $q' \in \delta(q, \sigma)$ if and only if $\exists\, e \in E$ such that $(q, e, q') \in \rightarrow$ and $\sigma = l(q)$. In addition, $q \in \delta(t, \sigma)$ iff $q \in Q_0$ and $\sigma = l(q)$.

Next, the RTL specification φ (seen as a subclass of linear temporal logic (LTL) without the next operator) is translated into an equivalent Büchi automaton, denoted as B_φ. The equivalence is in the sense that $\mathcal{L}_\omega(B_\varphi)$ is exactly the set of paths satisfying the formula φ, that is $\alpha \models \varphi$ if and only if $\alpha \in \mathcal{L}_\omega(B_\varphi)$. The basic idea of the translation is to use the collections of all sub-formulas as the state of the Büchi automaton, and the state should contain exactly those sub-formulas that hold true for all runs starting from this state. This translation process is well investigated in the computer science literature and there exist free software tools to automate the process, see e.g., [23]. However, the obtained Büchi automaton could be very large in the sense that the size of its states could be of an exponential growth (actually double exponential) compared with the length of the formula.

After obtaining B_T and B_φ, the next step is to build a Büchi automaton B such that $\mathcal{L}_\omega(B) = \mathcal{L}_\omega(B_\varphi) \cap \mathcal{L}_\omega(B_T)$, and then check whether the Büchi automaton B has any accepted runs. To find the intersection of Büchi automata $B_1 = (Q_1, Q_1^0, \ \Sigma, \delta_1, F_1)$ and $B_2 = (Q_2, Q_2^0, \ \Sigma, \delta_2, F_2)$, we construct $B = (Q_1 \times Q_2 \times \{0, 1, 2\}, \ \ Q_1^0 \times Q_2^0 \times \{0\}, \Sigma, \delta, Q_1 \times Q_2 \times \{2\})$. We have $[q_1', q_2', j] \in \delta([q_1, q_2, i], \sigma)$ if $q_1' \in \delta_1(q_1, \sigma)$, $q_2' \in \delta_2(q_2, \sigma)$, and the third component is affected by teh accepting conditions of B_1 and B_2

$$j = \begin{cases} 1 & \text{if } i = 0 \quad \text{and} \quad q_1' \in F_1, \\ 2 & \text{if } i = 1 \quad \text{and} \quad q_2' \in F_2, \\ 0 & \text{if } i = 2, \\ i & \text{otherwise.} \end{cases}$$

The third component is responsible for guaranteeing that accepting states from both B_1 and B_2 appear infinitely often. The third component is initially 0. It changes from 0 to 1 when an accepting state of B_1 is seen. It then changes to 2 from 1 when the accepting state of B_2 is visited, and, in the next state, it returns back to 0. It can be seen that B accepts exactly infinitely many states from F_1 and infinite many states from F_1 occur. A simpler intersection is obtained when all of the states of one of the automata are accepting. Assume that all the state of B_1 is accepting while the accepting state set of B_2 is F_2. The intersection of B_1 and B_2 can be obtained as $B_3 = (Q_1 \times Q_2, Q_1^0 \times Q_2^0, \Sigma, \delta, Q_1 \times F_2)$. Moreover $[q_1', q_2'] \in \delta([q_1, q_2], \sigma)$ if and only if $q_1' \in \delta_1(q_1, \sigma)$ and $q_2' \in \delta_2(q_2, \sigma)$, see e.g., [4, 12].

Should the ω-language of the intersection Büchi automaton B be nonempty, its corresponding run will be accepted by both B_T and B_φ. It implies that the run can be implemented in the transition system T and satisfies the specification φ. So a solution to the motion planning problem can be found as an accepting run in the Büchi automaton B. For such a purpose, we first look for a strongly connected component (SCC) in the graphical representation of B. If at least one of the states in the SCC containing a marked state in B, and is reachable from the initial state of B (within a finite number of transitions), then we find an accepted run of B. Note that if the ω-language accepted by a Büchi automaton B is not empty, then one can always find such a SCC in its graphical representation. In addition, the identification of SCC and the construction of a feasible run in the Büchi automaton B can be efficiently realized. Actually, the Breadth First Search (BFS) [52] can be used over the graphical representation of B to find the all shortest paths from the initial state to the marked states of B. Then, from a reachable marked state q_a, a new BFS is performed to find the shortest path that leads back to q_a.

Besides model checking-based algorithms, the design in the discrete domain can also be based on the discrete event system supervisory control theory [66] or reactive synthesis approaches [63]. The supervisory control theory was developed by Ramadge and Wonham in the 1980's [66, 67], and has seen significant growth in 1990's [9]. The basic idea of discrete event supervisory control is to restrict the happening of some events in the plant (modeled as a finite automaton) such that the closed-loop system respects a given specification (usually given as a regular or ω-regular language). Not all events can be disabled, and the events that can be disabled are called controllable events. To derive the existence condition of such a supervisor to enforce the specification, the controllability and observability conditions of the specification languages are proposed. Furthermore, when the specifications fail to satisfy the existence conditions, algorithms to compute the sub-languages of the specification that is controllable and/or observable were proposed in the literature. Interested readers may refer to [9, 66] for more details and more advanced topics such as control under partial observations, modular and decentralized supervisory control. The application of supervisory control in the symbolic motion planning can be found in the work [38, 39], where conic partition of the fly-zone of unmanned helicopters was considered and finite automata models were obtained.

As a closely related field, the supervisory control of hybrid systems using abstracted models has been advocated in the literature since early 1990's, see e.g., [14, 43, 53, 65] and the references therein. Early work in the area of hybrid supervisory control, e.g., [14, 43, 65], mainly performed the supervisor synthesis with respect to regular language specifications based on language equivalent or approximating quotient systems. However, language equivalence does not guarantee branching logic specifications, so we adopt a stronger equivalence condition [4, 12], namely bisimulation, instead.

5. Nonlinear Control Dynamics

So far, we have considered special cases when the continuous dynamics are linear control systems. Now we turn to consider the case when a robot dynamic model is given as a general nonlinear control system

$$\Sigma : \begin{cases} \dot{x}(t) = f(x(t), u(t)), \\ y(t) = g(x(t)), \end{cases} \tag{9}$$

where $x(t) \in \mathbb{R}^n$ is the state of the system, $u(t) \in U \subseteq \mathbb{R}^r$ is the control input and $y(t) \in \mathbb{R}^p$ is the observed output of the system.

Once again, the goal is to design a controller such that the output $y(t)$ generated from the closed-loop system satisfies a given temporal logic specification ϕ. It is also assumed that ϕ is built from atomic propositions, $\Pi = \{\pi_0, \pi_1, \ldots, \pi_m\}$, where each $[\![\pi_i]\!]$ stands for a region in concern. Also, it is assumed that the region $[\![\pi_i]\!]$ is bounded and convex.

The development here follows the results in [18], and the basic idea is to introduce a simpler linear model in $\mathbb{R}^p$, for which we design a hybrid controller to achieve a modified version of the specification ϕ. Then, a controller is designed for the original nonlinear system to keep its trajectory always within a neighborhood of the trajectory from the designed linear systems provided that their initial conditions are close enough. Note that the modification on the specification ϕ is made in such a way that once the trajectories are close enough to a trajectory satisfying the modified version of ϕ, then the original specification ϕ holds true for all these nearby trajectories. For such a purpose, we under-approximate the region $[\![\pi_i]\!]$ using a polytope when it needs to be reached by $y(t)$, otherwise over-approximate it using another polytope. Based on these approximations, we introduce new atomic propositions, $\tilde{\pi}_i$, and construct a new version of the specification ϕ, called $\tilde{\phi}$. Then, based on the previous section results of controlling a linear control system over polytopes, we can design controllers and successful runs for proper initial conditions. Finally, we design controllers for the nonlinear system such that its output $y(t)$ tracks the trajectory of the closed-loop linear system with specified bounded errors. Then, the resulting output trajectory $y(t)$ is guaranteed to satisfy the initial user specification.

Next, we introduce briefly the tracking problem and modifications on the temporal logic specifications. The basic idea is first to approximate the nonlinear control system (9) as a linear control system:

$$\Sigma' : \dot{z}(t) = Az(t) + Bv(t), \quad z(t) \in \mathbb{R}^p,\ z_0 \in [\![\pi_0]\!],\ v \in V \tag{10}$$

through the design of an interface.

We would like the linear control system approximates the trajectories of the nonlinear control system (9) in the following sense.

Definition 10 ([24]). A relation $\mathcal{W} \subseteq \mathbb{R}^p \times \mathbb{R}^n$ is an approximate simulation relation of precision δ of Σ' by Σ if for all $(z_0, x_0) \in \mathcal{W}$,

1. $\| z_0 - g(x_0) \| \leq \delta$
2. For all state trajectories $z(t)$ of Σ' such that $z(0) = z_0$ there exists a state trajectory $x(t)$ of Σ such that $x(0) = x_0$ and satisfying $(z(t), x(t)) \in \mathcal{W}$ for all $t \geq 0$.

An interface associated with the approximation simulation relation $\mathcal{W}$ allows us to choose the control inputs for the nonlinear control system (9) so that the states in the linear control system (10) and the states of the nonlinear control system (9) remain in $\mathcal{W}$.

Definition 11 ([18]). A continuous function $u_{\mathcal{W}} : V \times \mathcal{W} \to U$ is an interface associated with the approximate simulation relation $\mathcal{W}$, if for all $(z_0, x_0) \in \mathcal{W}$, for all trajectories $z(t)$ of Σ' associated to input $v(t)$ and such that $z(0) = z_0$, the trajectory of Σ given by

$$\dot{x}(t) = f(x(t), u_{\mathcal{W}}(v(t), z(t), x(t))), \quad x(0) = x_0, \tag{11}$$

satisfies for all $t \geq 0$, $(z(t), x(t)) \in \mathcal{W}$.

It is clear from the definitions that interconnecting the linear control system (10) and the nonlinear control system (9) through the interface $u_{\mathcal{W}}$

$$\begin{cases} \dot{x}(t) = f(x(t), u_{\mathcal{W}}(v(t), z(t), x(t))), \quad x(0) = x_0, \\ y(t) = g(x(t)) \end{cases}$$

satisfies for all $t \geq 0$, $\|y(t) - z(t)\| \leq \delta$ provided $\|g(x_0) - z_0\| \leq \delta$.

The approximate simulation relation can be constructed by the level sets of a simulation function, which is a positive function bounding the distance between the observations and nonincreasing under parallel evolution of the systems.

Definition 12 ([24]). Let $\mathcal{V} : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^+$ be a continuous and piecewise differentiable function. Let $u_{\mathcal{V}} : V \times \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^p$ be a continuous function. The function $\mathcal{V}$ is a simulation function of Σ' by Σ, and $u_{\mathcal{V}}$ is an associated interface if for all $(z, x) \in \mathbb{R}^p \times \mathbb{R}^n$,

$$\mathcal{V}(z, x) \geq \|z - g(x)\|^2, \tag{12}$$

$$\sup_{v \in \mathcal{V}} \left(\frac{\partial \mathcal{V}(z,x)}{\partial z}(Ax + Bv) + \frac{\partial \mathcal{V}(z,x)}{\partial x} f(x, u_{\mathcal{V}}(v, z, x)) \right) \leq 0. \tag{13}$$

Then, the approximate simulation relation can be defined as level sets of the simulation function.

Theorem 3 ([18, 24]). *Let the relation $\mathcal{W} \subseteq \mathbb{R}^p \times \mathbb{R}^n$ be given by*

$$\mathcal{W} = \{(z, x) | \mathcal{V}(z, x) \leq \delta^2\}. \tag{14}$$

If for all $v \in V$, for all $(z, x) \in \mathcal{W}$, $u_{\mathcal{V}}(v, z, x) \in U$, then $\mathcal{W}$ is an approximate simulation relation of precision δ of Σ' by Σ and $u_{\mathcal{W}} : V \times \mathcal{W} \to U$ given by $u_{\mathcal{W}}(v, z, x) = u_{\mathcal{V}}(v, z, x)$ is an associated interface.

Usually it is not easy to find such a simulation function except in some special cases. Also, the arguments used by the simulation function is similar to Lyapunov functions, so it is usually very conservative.

In our setup, an RTL formula ϕ is provided as a controller specification for the original nonlinear control system, while we need to deduce a new RTL formula for the auxiliary linear system such that once a trajectory satisfies the modified specification all neighboring trajectories will satisfy the original specification ϕ. Hence, we introduce the notation of δ-contraction so as to capture the robustness of satisfaction for a formula.

Definition 13 ([18]). Given a radius $\delta \in \mathbb{R}^+ \cup \{+\infty\}$ and a point α in a normed space A, the δ-ball centered at α is defined as $B_\delta(\alpha) = \{\beta \in A | \ \| \alpha - \beta \| \leq \delta\}$. If $\Gamma \subseteq A$, then

$$C_\delta(\Gamma) = \{\alpha \in A | B_\delta(\alpha) \subseteq \Gamma\}$$

is the δ-contraction and $B_\delta(\Gamma) = \{\alpha \in A | B_\delta(\alpha) \cap \Gamma \neq \emptyset\}$ is the δ-expansion.

Now, we define a new set of atomic propositions

$$\tilde{\Pi} = \{\xi_\alpha \,|\, \alpha = \pi \text{ or } \neg\pi \text{ for } \pi \in \Pi\}.$$

Next, we describe how to translate an RTL ϕ on Π into a new RTL, denoted as **rob**(ϕ), on $\tilde{\Pi}$. First, we write ϕ into the Negation Normal Form (NNF). Second, replace the occurrence of atomic proposition π and $\neg\pi$ with ξ_π and $\xi_{\neg\pi}$, respectively. Third, we define a new atomic map $[\![\cdot]\!]_\delta$ as follows:

$$\forall\, \xi \in \tilde{\Pi},\ [\![\xi]\!]_\delta = \begin{cases} C_\delta([\![\pi]\!]^c) & \text{if } \xi = \xi_{\neg\pi}, \\ C_\delta([\![\pi]\!]) & \text{if } \xi = \xi_{\neg\pi}, \end{cases} \tag{15}$$

where $\delta \in \mathbb{R}^+$ is a given positive scalar, and $[\![\pi]\!]^c$ stands for the complement of the set $[\![\pi]\!]$.

Intuitively, it means that we expand the region that $y(t)$ must avoid and δ-contract the region that it needs to reach. This process is illustrated in Fig. 2. The following result tells

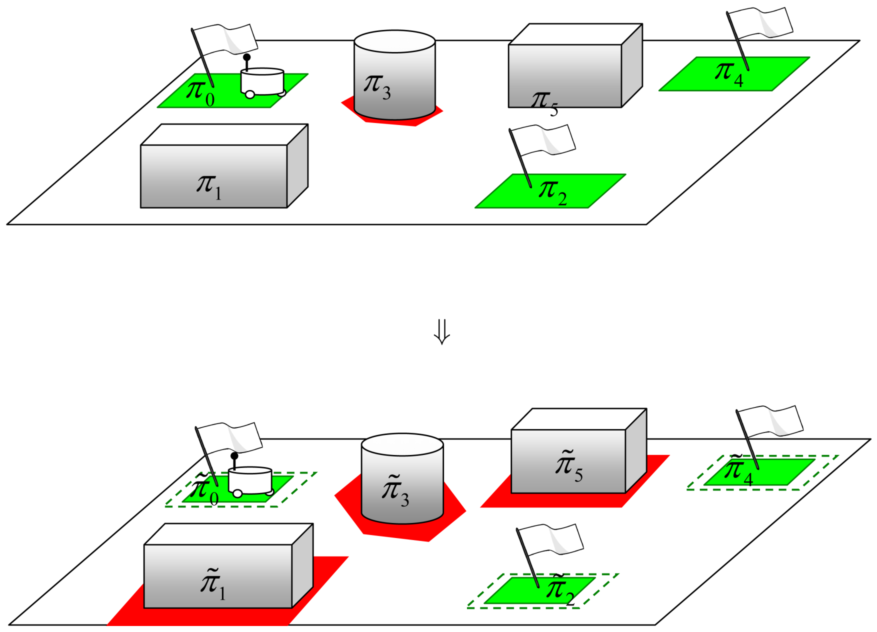


Fig. 2. An illustration of the RTL translation process. Three regions need to be visited, π_0, π_2 and π_4, are shrunken, while the regions need to be avoided, π_1 π_3 and π_5, are expanded.

us that if the trajectory satisfies the δ-robust specification, then any other trajectories that remain δ-close to the initial one will satisfy ϕ.

Theorem 4 ([18]). *Consider a formula $\phi \in \Phi_\Pi$, which is built on a set of atoms Π, a map $[\![\cdot]\!] : \Pi \to \mathcal{P}(\mathbb{R}^p)$, and a number $\delta \in \mathbb{R}^+$, then for all functions $y(t)$ and $z(t)$ from $\mathbb{R}^+$ to $\mathbb{R}^p$ such that for all $t \geq 0$, $\| z(t) - y(t) \| \leq \delta$, it holds that $(z, [\![\cdot]\!]_\delta) \models \mathbf{rob}(\phi) \Rightarrow (y, [\![\cdot]\!]) \models \phi$.*

Then, one can design a hybrid controller for the linear control system to satisfy $\mathbf{rob}(\phi)$ as introduced in Sec. 4, i.e., the closed-loop trajectory $z(t)$ satisfies $\mathbf{rob}(\phi)$. Once this is done, the remaining task is to design the interface so that the trajectory $y(t)$ always stays in the δ neighborhood of $z(t)$.

6. New Directions and Challenges

In this section, we give a brief discussion on some recent developments, possible trends and key challenges in the theoretical research for the symbolic motion planning problem.

6.1. *Uncertain environments*

In the developments so far, it is all assumed that the environment is static and we have full knowledge of the workspace in the form of an accurate map with known locations of obstacles. However, it is not always the case in real applications. In most real scenarios, we may only have partial knowledge of the environment, and the environment may be dynamic as the obstacles may move around and certain paths may become blocked. For instance, in an automated warehouse application, an aisle may be unexpectedly blocked by a box fallen down from the shelf so that the aisle become impassable for robots. Similarly, a door could be closed when the robot is trying to pass through. Once these happen, the robot needs to update its knowledge and replan accordingly.

To handle uncertain environments, sensor-based temporal-logic-motion planning was proposed in [45] where the robot is sensing the environment and performs synthesis with respect to a class of admissible environments. In particular, the specifications considered in [45] have the form of $\varphi_e \Rightarrow \varphi_s$, which is a fragment of LTL formulas and known as generalized reactivity(1) [GR(1)] formulas [62]. Here, the sub-formula φ_e stands for assumptions on the sensor readings about the environment and φ_s describes the desired behave of the robot. The specification is true if either φ_s holds true or φ_e is false. It means that the specification is satisfied if either the robot behave as expected or the environment is not admissible, namely φ_e is false. Hence, φ_e characterizes the admissible environments and is evaluated by the readings from the robot sensors. During the runtime, the robot keeps sensing the environment and synthesizes controllers if the environment is admissible. It is therefore called a reactive task as the robot reacts to possibly changing environments, and the performance is guaranteed provided that the environments are admissible.

Another advantage of restricting to GR(1) formulas is due to the existence of efficient algorithms based on temporal logic games to construct an equivalent Büchi automaton for a given GR(1) formula [62]. The construction algorithm is of $O(n^3)$ polynomial time complexity, where n is the size of the state space. Each state in this framework corresponds to an allowable truth assignment for the set of sensor and robot propositions (describing the robot sensor reading and actuation effects). Compared with general LTL, it represents a huge computational improvement as creating such an automaton for general LTL formulas is proven to be doubly exponential in the size of the formula, see e.g., [4]. However, the state space of a reactive task formula tends to be large as it requests many sensor and robot propositions to describe all possible admissible environments and the corresponding robot behaviors. So, formulating a reactive synthesis task itself could be difficult for human designers [45]. To facilitate the design process, a software tool called LTLMop was developed in [22], where structured English and LTL are used to write high level reactive task specifications.

A recent trend to handle environmental uncertainties relies on an iterative planning approach, see e.g., [3, 57, 60, 70]. The basic idea is to allow the robot to explore the uncertain environment using its sensors, and dynamically adjust/refine its abstraction of the workspace upon which a controller is re-synthesized once new obstacles or new regions were found. The iterative planning methods are more natural and suitable to deal with uncertain environments as they do not require a full characterization of the admissible environments.

6.2. *Imperfect sensing or actuation*

Besides uncertain environments, another type of uncertainties that commonly exists is due to the imperfect sensing or actuation from robots. The main concern here is whether the assumptions on the perfect sensing and actuations would cost us the guarantee of the performance. To address this concern, the authors in [31] relaxed the assumption of perfect sensing, and investigated the effect of sensor errors on the behavior of the robot. In particular, the sensor errors were characterized by a set of transition probabilities defining the probability distributions of the next sensor value given the next environment values and the current proposition values of the sensors and the robot, respectively. Under this framework, a discrete-time Markovian chain (DTMC) can be constructed to describe the behavior of the robot under imperfect sensing. Then, the probabilistic model-checking techniques [29] can be applied to the DTMC model to calculate the probability with which the controller satisfies a set of temporal logic specifications. The method proposed in [31] is therefore more of performance assessment, but the calculated probability may help to guide the redesign process.

On the other hand, the symbolic motion planning problem under imperfect actuations was considered in [49], where the outcome of the low-level motion controllers was treated in a probabilistic fashion. In particular, the authors of [49] considered the case when a robot moves in a partitioned environment by applying a given set of motion primitives that are predesigned (like we did in Sec. 4) and steer the robot between adjacent regions. However, due to imperfect actuations, the robot may end in other adjacent regions with some probabilities. It is assumed that the probabilities of these transitions are known and the robot can determine its current region precisely. Based on these assumptions, the motion of the robot was modeled as a Markov decision process (MDP) and the motion specification was formulated as probabilistic computation tree logic (PCTL). Hence, the symbolic motion planning problem under imperfect actuations was converted to a probabilistic model checking problem, for which there exist efficient algorithms to calculate the maximum probability or the minimum cost of satisfaction and a control strategy that achieves this probability or cost, see e.g., [4]. In addition, there exist free software tools implementing the probabilistic model checking algorithms, see e.g., the PRISM [48] tool.

6.3. *Optimality*

Usually, the solutions to a temporal logic satisfaction problem are not unique. On the other hand, typical robotic applications not only request a completion of the task but also need to do so in an efficient way. Efficiency could mean the shortest total travel length, minimum cost for robot deployments or maximum probability of success. Hence, a recent trend in the symbolic motion planning literature is to combine optimal control and optimization techniques with formal synthesis. Since the satisfactory runs for a temporal logic are usually of infinite length, it is not trivial to assign costs to a particular run. Accumulative, average, weight average, and expected average costs for transitions or between two consecutive satisfactions are all considered in the literature, see e.g., [71, 72, 75].

To distinguish different runs, the authors in [71] modeled the robot moving in a partitioned environment as a weighted transition systems, and then developed an off-line control strategy minimizing the maximum cost between two consecutive visits to a given set of states that must be consistently visited. The solution was obtained by a modified graph algorithm that searches for a run corresponding to the optimal robot path in the product automaton of the robot weighted transition system model and a Büchi automaton obtained from the given LTL specification. If the robot and its environment are modeled as a MDP, the optimality can be naturally considered as the maximum probability or the minimum cost of satisfaction, and ready results based on dynamic programming exist in the literature. In [16], the authors synthesized a control policy such that the MDP satisfies the given specification almost surely, if such a policy exists. The control strategies synthesis for MDP so to minimize the expected average cost between two consecutive satisfactions of a desired LTL property was considered in [72] by using results from the game theory [10].

Inspired from [76], a receding horizon control strategy was used in [15] to maximize the collected reward while satisfying the high level task specification. At each time step, a local collected rewards with time-varying rewards associated with states of the system is optimized over a finite horizon, and the immediate optimal control is applied. A software tool TuLip [78] has been developed to implement the receding horizon temporal logic synthesis. The adoption of the receding horizon control scheme is appealing as it helps to reduce the computational complexity [76, 77], as the state explosion problem is a well-documented challenge in the formal verification and synthesis literature. However, the receding horizon control scheme usually cannot guarantee the completeness or global optimality. In other words, the receding horizon-based synthesis may fail to find a satisfactory run even when such feasible solutions exist. How to guarantee the completeness and global optimality of the receding horizon-based controller design is still an ongoing research question.

As a generalization of the vehicle routing problem, the authors in [34] considered vehicle routing with metric temporal-logic specifications with the goal to minimize a cost function of the vehicle paths (such as total distance traveled). The basic idea is to convert the temporal specifications into a set of constraints suitable to a Mixed-Integer Linear Programming (MILP) formulation. Solving the resulting MILP provides an optimal plan that satisfies the given mission specification.

6.4. *Multi-robot systems*

Most of the results reviewed so far were developed for a single robot motion planning. Their extensions to the multi-robot case [30, 42] are nontrivial. One possible way is to compose all transition systems or MDP models for individual robots together to obtain a centralized model, upon which the product automaton with the specifications can be constructed and searching algorithms can be performed. After a specification satisfying run is synthesized centrally, it is then projected to each individual robots. This idea is pursued by the majority of existing work, see e.g., [11, 41, 58, 64, 79]. However, the centralized design suffers from the state explosion problem as the number of states in a composition system is usually huge. In addition, the centralized design lacks flexibility and is difficult to adjust if the task changes or the team needs to reconfigure (due to a failure in one of the team members). Another method for the multi-robot system is to treat other robots as part of the environment and design in a reactive fashion [45]. However, the reactive approaches cannot handle the cooperative tasks that need tight coordinations or joint decision making between robots. In addition, the reactive approaches need to specify all possible behaviors of other robots as part of admissible environments, which is nontrivial and could significantly increase the state space of the reactive task formula. Hence, both the centralized design and the reactive methods are difficult to scale-up for large number of robots.

In a recent work [36], a “divide-and-conquer” approach to the multi-robot coordination design was proposed. The basic idea is to first decompose the team (global) specification into subtasks for each individual robots, and a local supervisor is then synthesized separately for each agent to fulfill these subtasks. The key step is the task decomposition, and the decomposition is not arbitrary as it requests that the satisfaction of decomposed subtasks by all individual robots will imply the accomplishment of the team specification. In particular, [36] studied the cooperative tasking problems for two agents and then generalized to arbitrarily finite number of agents with a necessary and sufficient condition under which a deterministic task automaton is decomposable with respect to parallel composition and natural projections into local event sets in the sense that the task automaton is bisimilar to the parallel composition of its natural projections. The authors then extended their work to the case of fault-tolerant cooperative tasking in [35] and dealt with the robustness issues of the proposed top-down design approach with respect to event failures in the multi-agent systems, and necessary and sufficient conditions characterized by event passivity were proposed on failed events under which a decomposable global task can still be accomplished successfully. The top-down design approach was implemented to the coordination control of unmanned helicopters in [37, 39] using conic partitions of the fly zone.

7. Concluding Remarks

This paper gives an overview of the recent developments in the symbolic motion planning for robots to satisfy high level temporal logic specifications. This is a very ambitious goal as the filed is very dynamic with a wide spectrum of approaches and applications ranging from computer science, robotics and control theory. It is highly possible that we have missed important results in spite of our best efforts. If this has happened, we do apologize.

The reachability control on simplex for linear and affine dynamics follows the work in [27]. The concept of approximate bisimulation was proposed in [25]. Our treatment on the synthesis of hybrid controllers for nonlinear systems mainly follows the results in [26]. Although we only considered the symbolic control for continuous control systems, the extension of the idea to the cases of hybrid systems is not difficult provided that the regions in concern, such as invariant sets and guard sets, are all assumed to be polyhedrons. For example, the reachability and control problems for hybrid systems with piecewise affine dynamics defined on simplices were considered in [28] using the techniques discussed in Sec. 4. Piecewise affine systems are also called piecewise linear systems, and have been widely used in the study of circuits, see e.g., [51], since they can approximate nonlinear dynamics with arbitrary accuracy [55]. There also exist efficient computational techniques for the identification of piecewise affine models from input–output data, such as clustering-based methods [21], mixed-integer programming [69], and Bayesian methods [32].

Although we mainly emphasized the feedback controller-based approaches, there are other methods developed to design the low level controllers, such as the vector fields blending [56], composition of local potential functions [13] and sampling-based approaches [6, 7, 33]. These methods are intuitive to use and also can handle complex dynamics and environments.

Acknowledgment

The financial supports of NSF-CNS-1239222 and NSF-CAREER-1253488 for this work are greatly acknowledged.

References

[1] R. Alur, T. Henzinger, G. Lafferriere and G. J. Pappas, Discrete abstractions of hybrid systems, in *Proc. IEEE: Special Issue on Hybrid Systems*, Vol. 88, ed. P. J. Antsaklis (IEEE Press, 2000), pp. 971–984.

[2] M. Antoniotti and B. Mishra, Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers, in *1995 IEEE Int. Conf. Robotics and Automation, 1995. Proc.*, Vol. 2 (IEEE, 1995), pp. 1441–1446.

[3] A. I. Ayala, S. B. Andersson and C. Belta, Temporal logic motion planning in unknown environments, in *2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE, 2013), pp. 5279–5284.

[4] C. Baier and J. P. Katoen, *Principles of Model Checking* (MIT Press, Cambridge, 2008).

[5] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins and G. J. Pappas, Symbolic planning and control of robot motion, *IEEE Robot. Autom. Mag.* **14**(1) (2007) 61–70.

[6] A. Bhatia, L. E. Kavraki and M. Y. Vardi, Sampling-based motion planning with temporal goals, in *2010 IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE, 2010), pp. 2689–2696.

[7] A. Bhatia, M. R. Maly, E. E. Kavraki and M. Y. Vardi, Motion planning with complex goals, *IEEE Robot. Autom. Mag.* **18**(3) (2011) 55–64.

[8] R. Bogue, Robots in the nuclear industry: A review of technologies and applications, *Ind. Robot* **38**(2) (2011) 113–118.

[9] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd edn. (Springer-Verlag, 2008).

[10] K. Chatterjee and L. Doyen, Energy and mean-payoff parity markov decision processes, in *Mathematical Foundations of Computer Science 2011* (Springer, 2011), pp. 206–218.

[11] Y. Chen, X. Ding, A. Stefanescu and C. Belta, Formal approach to the deployment of distributed robotic teams, *IEEE Trans. Robot.* **28**(1) (2012) 158–171.

[12] E. M. Clarke, O. Grumberg and D. Peled, *Model Checking* (MIT press, 1999).

[13] D. C. Conner, A. A. Rizzi and H. Choset, Composition of local potential functions for global robot control and navigation, in *2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2003. (IROS 2003). Proc.* Vol. 4 (IEEE, 2003), pp. 3546–3551.

[14] J. E. R. Cury, B. H. Krogh and T. Niinomi, Synthesis of supervisory controllers for hybrid systems based on approximating automata, *IEEE Trans. Autom. Control* **43**(4) (1998) 564–568.

[15] X. C. Ding, M. Lazar and C. Belta, Receding horizon temporal logic control for finite deterministic systems, in *American Control Conf. (ACC), 2012* (IEEE, 2012), pp. 715–720.

[16] X. C. Ding, S. L. Smith, C. Belta and D. Rus, Mdp optimal control under temporal logic constraints, in *2011 50th IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC)* (IEEE, 2011), pp. 532–538.

[17] P. Doherty, F. Heintz and J. Kvarnström, High-level mission specification and planning for collaborative unmanned aircraft systems using delegation, *Unmanned Syst.* **1**(01) (2013) 75–119.

[18] G. E. Fainekos, A. Girard, H. Kress-Gazit and G. J. Pappas, Temporal logic motion planning for dynamic robots, *Automatica* **45**(2) (2009) 343–352.

[19] G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, Hybrid controllers for path planning: A temporal logic approach, in *44th IEEE Conf. Decision and Control, 2005 and 2005 European Control Conf. CDC-ECC'05.* (IEEE, 2005), pp. 4885–4890.

[20] G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, Temporal logic motion planning for mobile robots, in *Proc. 2005 IEEE Int. Conf. Robotics and Automation, 2005. ICRA 2005.* (IEEE, 2005), pp. 2020–2025.

[21] G. Ferrari-Trecate, M. Muselli, D. Liberati and M. Morari, A clustering technique for the identification of piecewise affine systems, *Automatica* **39**(2) (2003) 205–217.

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

[23] P. Gastin and D. Oddoux, Fast LTL to büchi automata translation, in *Computer Aided Verification* (Springer, 2001), pp. 53–65.

[24] A. Girard and G. J. Pappas, Approximation metrics for discrete and continuous systems (2005).

[25] A. Girard and G. J. Pappas, Approximation metrics for discrete and continuous systems, *IEEE Trans. Autom. Control* **52**(5) (2007) 782–798.

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

[27] L. Habets and J. H. van Schuppen, A control problem for affine dynamical systems on a full-dimensional polytope, *Automatica* **40**(1) (2004) 21–35.

[28] L. C. G. J. M. Habets, P. J. Collins and J. H. van Schuppen, Reachability and control synthesis for piecewise-affine hybrid systems on simplices, *IEEE Trans. Autom. Control* **51**(6) (2006) 938–948.

[29] E. M. Hahn, H. Hermanns and L. Zhang, Probabilistic reachability for parametric markov models, *Int. J. Softw. Tools Technol. Transf.* **13**(1) (2011) 3–19.

[30] J. Hu, J. Xu and L. Xie, Cooperative search and exploration in robotic networks, *Unmanned Syst.* **1**(1) (2013) 121–142.

[31] B. Johnson and H. Kress-Gazit, Probabilistic analysis of correctness of high-level robot behavior with sensor error (2011).

[32] A. L. Juloski, S. Weiland and W. P. M. H. Heemels, A bayesian approach to identification of hybrid systems, *IEEE Trans. Autom. Control* **50**(10) (2005) 1520–1533.

[33] S. Karaman and E. Frazzoli, Sampling-based motion planning with deterministic μ-calculus specifications, in *Proc. 48th IEEE Conf. Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conf. CDC/CCC 2009* (IEEE, 2009), pp. 2222–2229.

[34] S. Karaman and E. Frazzoli, Linear temporal logic vehicle routing with applications to multi-uav mission planning, *Int. J. Robust Nonlinear Control* **21**(12) (2011) 1372–1395.

[35] M. Karimadini and H. Lin, Fault-tolerant cooperative tasking for multi-agent systems, *Int. J. Control* **84**(12) (2011) 2092–2107.

[36] M. Karimadini and H. Lin, Guaranteed global performance through local coordinations, *Automatica* **47**(5) (2011) 890–898.

[37] A. Karimoddini, M. Karimadini and H. Lin, Decentralized hybrid formation control of unmanned aerial vehicles (2014), arXiv: 1403.0258.

[38] A. Karimoddini, H. Lin, B. M. Chen and T. H. Lee, Hybrid formation control of the unmanned aerial vehicles, *Mechatronics* **21**(5) (2011) 886–898.

[39] A. Karimoddini, H. Lin, B. M. Chen and T. H. Lee, Hybrid three-dimensional formation control for unmanned helicopters, *Automatica* **49**(2) (2013) 424–433.

[40] M. Kloetzer and C. Belta, A fully automated framework for control of linear systems from temporal logic specifications, *IEEE Trans. Autom. Control* **53**(1) (2008) 287–297.

[41] M. Kloetzer and C. Belta, Automatic deployment of distributed teams of robots from temporal logic motion specifications, *IEEE Trans. Robot.* **26**(1) (2010) 48–61.

[42] A. N. Kopeikin, S. S. Ponda, L. B. Johnson and J. P. How, Dynamic mission planning for communication control in multiple unmanned aircraft teams, *Unmanned Syst.* **1**(1) (2013) 41–58.

[43] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver and M. D. Lemmon, Supervisory control of hybrid systems, *Proc. IEEE* **88**(7) (2000) 1026–1049.

[44] H. Kress-Gazit, Robot challenges: Toward development of verification and synthesis techniques [from the guest editors], *IEEE Robot. Autom. Mag.* **18**(3) (2011) 22–23.

[45] H. Kress-Gazit, G. E. Fainekos and G. J. Pappas, Temporal-logic-based reactive mission and motion planning, *IEEE Trans. Robot.* **25**(6) (2009) 1370–1381.

[46] H. Kress-Gazit, G. E. Fainekos and G. J. Pappas, Temporal-logic-based reactive mission and motion planning, *IEEE Trans. Robot.* **25**(6) (2009) 1370–1381.

[47] H. Kress-Gazit, T. Wongpiromsarn and U. Topcu, Correct, reactive robot control from abstraction and temporal logic specifications, *IEEE Robot. Autom. Mag.* **18**(3) (2011) 65–74.

[48] M. Kwiatkowska, G. Norman and D. Parker, Prism: Probabilistic symbolic model checker, in *Computer Performance Evaluation: Modelling Techniques and Tools* (Springer, 2002), pp. 200–204.

[49] M. Lahijanian, S. B. Andersson and C. Belta, Temporal logic motion planning and control with probabilistic satisfaction guarantees, *IEEE Trans. Robot.* **28**(2) (2012) 396–409.

[50] C. W. Lee, Subdivisions and triangulations of polytopes, in *Handbook of Discrete and Computational Geometry* (CRC Press, Inc., 1997), pp. 271–290.

[51] D. M. W. Leenaerts and W. M. G. Van Bokhoven, *Piecewise Linear Modeling and Analysis* (Springer, 1998).

[52] C. E. Leiserson, R. L. Rivest, C. Stein and T. H. Cormen, *Introduction to Algorithms* (MIT press, 2001).

[53] M. D. Lemmon, K. He and I. Markovsky, Supervisory hybrid systems, *IEEE Control Syst.* **19**(4) (1999) 42–55.

[54] H. Lin and P. J. Antsaklis, Hybrid dynamical systems: An introduction to control and verification, *Found. Trends Syst. Control* **1**(1) (2014) 1–172.

[55] J. N. Lin and R. Unbehauen, Canonical piecewise-linear approximations, *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.* **39**(8) (1992) 697–699.

[56] S. R. Lindemann and S. M. LaValle, Smoothly blending vector fields for global robot navigation, in *44th IEEE Conf. Decision and Control, 2005 and 2005 European Control Conf. CDC-ECC'05* (IEEE, 2005), pp. 3553–3559.

[57] S. C. Livingston, R. M. Murray and J. W. Burdick, Backtracking temporal logic synthesis for uncertain environments, in *Robotics and Automation (ICRA) 2012 IEEE Int. Conf.* (IEEE, 2012), pp. 5163–5170.

[58] S. G. Loizou and K. J. Kyriakopoulos, Automatic synthesis of multi-agent motion tasks based on ltl specifications, in *43rd IEEE Conf. Decision and Control, 2004. CDC.* Vol. 1 (IEEE, 2004), pp. 153–158.

[59] D. M. Lyons, R. C. Arkin, P. Nirmal, S. Jiang, T. M. Liu and J. Deeb, Getting it right the first time: Robot mission guarantees in the presence of uncertainty, in *2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (IEEE, 2013), pp. 5292–5299.

[60] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit and M. Y. Vardi, Iterative temporal motion planning for hybrid systems in partially unknown environments, in *Proc. 16th Int. Conf. Hybrid Systems: Computation and Control* (ACM, 2013), pp. 353–362.

[61] R. Milner, *Communication and Concurrency* (Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989).

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

[63] A. Pnueli and R. Rosner, On the synthesis of a reactive module, in *Proc. 16th ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages* (ACM, 1989), pp. 179–190.

[64] M. M. Quottrup, T. Bak and R. I. Zamanabadi, Multi-robot planning: A timed automata approach, in *2004 IEEE Int. Conf. Robotics and Automation, 2004. Proc. ICRA'04.*, Vol. 5 (IEEE, 2004), pp. 4417–4422.

[65] J. Raisch and S. D. O'Young, Discrete approximation and supervisory control of continuous systems, *IEEE Trans. Autom. Control* **43**(4) (1998) 569–573.

[66] P. Ramadge and W. M. Wonham, The control of discrete event systems, *Proc. IEEE* **77**(1) (1989) 81–98.

[67] P. J. Ramadge and W. M. Wonham, Supervisory control of a class of discrete event processes, *SIAM J. Control Optim.* **25**(1) (1987) 206–230.

[68] M. Reynolds, Continuous temporal models, in *AI 2001: Advances in Artificial Intelligence* (Springer, 2001), pp. 414–425.

[69] J. Roll, A. Bemporad and L. Ljung, Identification of piecewise affine systems via mixed-integer programming, *Automatica* **40**(1) (2004) 37–50.

[70] S. Sarid, B. Xu and H. Kress-Gazit, Guaranteeing high-level behaviors while exploring partially known maps, in *Robotics* (2013), p. 377.

[71] S. L. Smith, J. Tùmová, C. Belta and D. Rus, Optimal path planning for surveillance with temporal-logic constraints, *Int. J. Robot. Res.* **30**(14) (2011) 1695–1708.

[72] M. Svorenova, I. Cerna and C. Belta, Optimal control of mdps with temporal logic constraints (2013), arXiv:1303.1942.

[73] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach* (Springer, New York, 2009).

[74] M. Y. Vardi, An automata-theoretic approach to linear temporal logic, in *Logics for Concurrency* (Springer, 1996), pp. 238–266.

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

[76] T. Wongpiromsarn, U. Topcu and R. M. Murray, Receding horizon control for temporal logic specifications, in *Proc. 13th ACM Int. Conf. Hybrid Systems: Computation and Control* (ACM, 2010), pp. 101–110.

[77] T. Wongpiromsarn, U. Topcu and R. M. Murray, Synthesis of control protocols for autonomous syst. *Unmanned Systems* **1**(1) (2013) 21–39.

[78] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu and R. M. Murray, Tulip: A software toolbox for receding horizon temporal logic planning, in *Proc. 14th Int. Conf. Hybrid Syst.: Computation and Control* (ACM, 2011), pp. 313–314.

[79] T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli and D. Rus, Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications, in *2013 IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE, 2013), pp. 5011–5018.

[80] G. M. Ziegler, *Lectures on Polytopes*, Vol. 152 (Springer, 1995).

Hai Lin obtained his B.S. degree at the University of Science and Technology Beijing and his M.S. degree from the Chinese Academy of Sciences in 1997 and 2000, respectively. In 2005, he received his Ph.D. degree from the University of Notre Dame. Dr. Lin is currently an Assistant Professor at the Department of Electrical Engineering, University of Notre Dame. Before returning to his alma mater, Hai has been working as an Assistant Professor in the National University of Singapore from 2006 to 2011. Dr. Lin's teaching and research interests are in the multidisciplinary study of the problems at the intersections of control, communication, computation and life sciences. His current research thrust is on cyber-physical systems, multi-robot cooperative tasking, systems biology and hybrid control.

Hai has been served in several committees and editorial board. He is the Program Chair for IEEE ICCA 2011, IEEE CIS 2011 and the Chair for IEEE Systems, Man and Cybernetics Singapore Chapter for 2009 and 2010. He is a senior member of IEEE and a recipient of 2013 NSF CAREER award.