To appear, 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
http://www.cds.caltech.edu/~murray/papers/hwm14-iros.html

A Compositional Approach to Stochastic Optimal Control with Co-safe Temporal Logic Specifications

Matanya B. Horowitz, Eric M. Wolff, Richard M. Murray

***Abstract*— We introduce an algorithm for the optimal control of stochastic nonlinear systems subject to temporal logic constraints on their behavior. We compute directly on the state space of the system, avoiding the expensive pre-computation of a discrete abstraction. An automaton that corresponds to the temporal logic specification guides the computation of a control policy that maximizes the probability that the system satisfies the specification. This reduces controller synthesis to solving a sequence of stochastic constrained reachability problems. Each individual reachability problem is solved via the Hamilton-Jacobi-Bellman (HJB) partial differential equation of stochastic optimal control theory. To increase the efficiency of our approach, we exploit a class of systems where the HJB equation is linear due to structural assumptions on the noise. The linearity of the partial differential equation allows us to pre-compute control policy primitives and then compose them, at essentially zero cost, to conservatively satisfy a complex temporal logic specification.**

I. Introduction

We present a method for synthesizing control policies for continuous-time stochastic nonlinear systems with co-safe temporal logic task specifications. We are motivated by safety-critical robotics applications involving autonomous ground and air vehicles executing complex tasks. In such applications, it is desirable to automatically synthesize a control policy that provably implements specified system behavior, despite nonlinearities and disturbances.

Linear temporal logic (LTL) is a task specification language that has been widely used for specifying properties of hybrid systems, robotics, and software. We use syntactically co-safe LTL, an expressive finite-time fragment of LTL, to specify a wide range of properties relevant to autonomous systems. These properties include safety, response to the environment, and goal visitation. Such properties generalize classical motion planning [21].

Common approaches for control policy synthesis for stochastic systems with LTL specifications first abstract the dynamical system as a finite Markov decision process (MDP) [10], [19]. Each state in this MDP corresponds to a subset of the system state space, and transition probabilities between states in the MDP encode possible system behaviors. Given such an MDP and an LTL specification, control policies can be automatically constructed using an automata-based approach [9], [19]. This approach extends work in the formal verification community [8], [18] to hybrid systems via the use of finite abstractions. The main drawback of this approach is that it is computationally expensive to create a finite abstraction [10], [19].

Matanya B. Horowitz, Eric M. Wolff, and Richard M. Murray are with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA. The corresponding author is mhorowit@caltech.edu

We avoid the expensive computation of an MDP abstraction, and instead directly compute on the state space of the system using techniques from stochastic optimal control. We use an automaton representing the temporal logic specification to guide the computation of a control policy that maximizes the probability that the system satisfies the specification. We treat this automaton as an MDP, where states (modes) encode progress towards task completion and each action corresponds to a control policy for the continuous system. We use dynamic programming to maximize the probability that the system satisfies the specification from its initial state. A feedback control policy is returned which selects the current action based on the system's continuous state and its mode.

By avoiding the computation of a discrete abstraction of the system, our approach lets one take advantage of recent advances in computing constrained reachability relations for nonlinear stochastic systems. Recently it has been shown that the Hamilton-Jacobi-Bellman (HJB) equation, a nonlinear partial differential equation (PDE) central to stochastic optimal control, may be transformed to a linear PDE given several mild assumptions [16], [30]. Linear PDEs can be solved with a number of computational tools, some of which scale relatively well with dimension, such as those based on the Feynman-Kac lemma [23], semidefinite programming [14], and sparse tensor discretization [15], [25].

Additionally, the solutions to linear PDEs obey the principle of superposition, a characteristic previously exploited in [29]. We build upon this previous work, leveraging superposition to quickly solve problems defined with temporal specifications. Indeed, the computation of solutions by superposition is appealing in situations where many control problems must be solved over a common domain, as in many temporal logic planning problems. For such problems, the specification creates a large number of constrained reachability subproblems, where we leverage the property of superposition to efficiently compose the subproblems.

Our main contribution is an efficient framework for control policy synthesis for stochastic nonlinear systems for syntactically co-safe LTL. Our framework is general in that it can utilize any technique that computes solutions to a stochastic constrained reachability problem. We take advantage of the case where the HJB equation is linear to specialize and increase the efficiency of our approach. This is done by exploiting the fact that solutions constructed for individual specifications may be superimposed to satisfy

richer specifications.

Our work is closely related to recent automata-guided approaches for control policy synthesis for discrete-time deterministic [32] and stochastic [28] systems subject to temporal logic specifications. Our work is also related to [11], which similarly encodes the relationship between sequential tasks via the boundary conditions to HJB equations. These approaches directly compute over the state space of the system. Our work differs in that we present a compositional approach based on superposition of solutions to linear PDEs. This allows solutions to be quickly computed, albeit with a degree of conservatism (see Section V).

II. Preliminaries

Notation: An *atomic proposition* is a statement that is either $True$ or $False$. The expectation of a random variable is denoted by $\mathbb{E}[\cdot]$, and the probability of an event is denoted by $\mathbb{P}[\cdot]$.

A. System Model

We consider continuous-time stochastic nonlinear systems that evolve with dynamics

$$dx_t = (f(x_t) + G(x_t)u_t)\,dt + B(x_t)H\,d\omega_t, \qquad (1)$$

with state $x_t \in X \subset \mathbb{R}^n$, control input $u_t \in \mathbb{R}^m$, Brownian motion $\omega_t \in \mathbb{R}^m$, and disturbance matrix $H \in \mathbb{R}^{n\times m}$. The functions $f(x_t)$, $G(x_t)$, and $B(x_t)$ are continuously differentiable, and the set X is compact.

Let AP be a finite set of atomic propositions. The labeling function $L : X \to 2^{AP}$ maps the state to the set of atomic propositions that are $True$. The set of states where atomic proposition $p \in AP$ holds is denoted by $[\![p]\!]$.

Definition 1. A *memoryless control policy* for a system of the form (1) is a map $\mu : X \to \mathbb{R}^m$. A *finite-memory control policy* is a map $\mu : X \times M \to \mathbb{R}^m \times M$ where the finite set M is called the memory.

The *trajectory* $\mathbf{x}(x_0, \mu, \boldsymbol{\omega}) = \mathbf{x} : R_{\geq 0} \to X$ represents a solution of (1) induced by an initial state $x_0 \in X$, a given control policy μ, and an instance of Brownian motion $\boldsymbol{\omega}$. A *word* of a trajectory $\mathbf{x}$ is an infinite sequence of labels $L(\mathbf{x}) = L(x_{t_0})L(x_{\tau_0})L(x_{t_1})L(x_{\tau_1})L(x_{t_2})\ldots$, such that $t_0 = 0$ and for all $i \geq 0$, $t_{i+1} \geq t_i$, $\tau_i \in [t_i, t_{i+1}]$, and $L(x_t) = L(x_{\tau_i})$ for all $t \in (t_i, t_{i+1})$. A word of trajectory $\mathbf{x}$ defines the behavior of $\mathbf{x}$ in terms of the label sequence. We assume that during any finite time interval, the label changes a finite number of times.

The set of words of system (1) with initial state $x_0 \in X$ induced by a control policy μ is denoted by $\mathbf{x}(x_0, \mu)$. The Brownian motion induces a probability measure over the trajectories of the system $\mathbf{x}(x_0, \mu)$, and thus the words.

B. Specification Language

We use syntactically co-safe linear temporal logic (sc-LTL) [17] to concisely and unambiguously specify desired system behavior over a finite horizon. In this paper, we only consider system behavior over a finite horizon due to the unbounded disturbances in equation (1). We give a brief introduction to the syntax and semantics of sc-LTL and refer the reader to [17] for details.

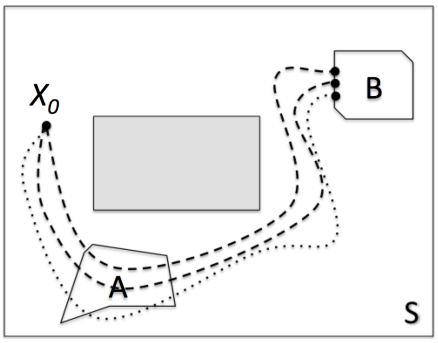


Fig. 1. A set of three example trajectories of a system satisfying the specification $\varphi = (S\ \mathcal{U}\ A)\ \wedge\ (S\ \mathcal{U}\ B)$, meaning visit regions A and B before leaving region S. The grey indicates a region outside of S. The word for the two dashed trajectories is $w = S(A\wedge S)S(B\wedge S)$, while the dotted trajectory re-enters A, giving the word $w = S(A\wedge S)S(A\wedge S)S(B\wedge S)$.

An sc-LTL formula is formed from the Boolean operators: $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and the temporal operators: $\mathcal{U}$ (until) and $\Diamond$ (eventually). An sc-LTL formula is written in positive normal form (i.e., negations are only allowed in front of atomic propositions). We do not include the $\bigcirc$ (next) temporal operator, which is ill-defined in continuous-time.

Definition 2. A syntactically co-safe LTL (sc-LTL) formula over a set of atomic propositions is inductively defined as follows:

$$\varphi ::= p \mid \neg p \mid \varphi_1\ \vee\ \varphi_2 \mid \varphi_1\ \wedge\ \varphi_2 \mid \varphi_1\,\mathcal{U}\,\varphi_2 \mid \Diamond\varphi,$$

where $p \in AP$ is an atomic proposition.

The semantics of an sc-LTL formula is defined over infinite words $w = w_0w_1w_2\ldots$ where $w_i \in 2^{AP}$. Informally, $\varphi_1\,\mathcal{U}\,\varphi_2$ means that φ_1 is $True$ until φ_2 is $True$ and $\Diamond\varphi$ means that φ eventually is $True$. More complex specifications can be defined by combining Boolean and temporal operators. The satisfaction of an sc-LTL formula is guaranteed in finite time [17]. An example of a temporal logic planning problem is shown in Figure 1.

There is a useful connection between sc-LTL formulae and deterministic finite automata.

Definition 3. A *deterministic finite automaton* (DFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with a finite set of states Q, a finite alphabet Σ, a transition function $\delta : Q \times \Sigma \to Q$, an initial state $q_0 \in Q$, and a set of accepting states $F \subset Q$.

An *accepting run* σ of an automaton $\mathcal{A}$ on a finite word $w = w_0 \ldots w_k$ over $\Sigma = 2^{AP}$ is a sequence of states $\sigma = q_0 \ldots q_{k+1}$ such that q_0 is the initial state, $q_{k+1} \in F$, and $q_{i+1} = \delta(q_i, w_i)$ for all $i = 0, \ldots, k$.

For any sc-LTL formula φ, there exists a deterministic finite automaton $\mathcal{A}_\varphi$ that accepts exactly the prefixes of all

satisfying words. Software exists for constructing such a deterministic finite automata from an sc-LTL formula [20].

III. Problem Statement

We now formally state our main problem. We begin by defining the probability of satisfaction of a specification by a stochastic system of the form (1).

The stochastic system (1) may have an infinite set of words $\mathbf{x}(x_0, \mu)$ for a given initial state x_0 and control policy μ. There is a well-defined probability measure over this infinite set of words [3], which gives rise to the notion of the probability that a specification is satisfied.

Definition 4. Let $\mathbf{x}(x_0, \mu)$ be a set of words of system (1) from initial state x_0 under control policy μ with an associated probability measure. Let φ be an sc-LTL formula over AP. Then, $\mathbb{P}(\mathbf{x}(x_0, \mu) \models \varphi)$ is the *probability that φ is satisfied* by system (1) under control policy μ.

Problem 1. Given a system of the form (1) and a syntactically co-safe LTL formula φ over AP, compute a control policy μ^* that maximizes the probability that φ is satisfied, i.e., $\mu^* \in \arg\max_\mu \mathbb{P}(\mathbf{x}(x_0, \mu) \models \varphi)$.

We give an efficient, yet conservative, solution to Problem 1. At a high level, we reduce Problem 1 to a series of stochastic constrained reachability (reach-avoid) problems, where the system attempts to reach a goal region while avoiding another region, e.g., obstacles. Each of these individual constrained reachability problems can be solved using stochastic optimal control techniques. We show how to chain these solutions together, via a dynamic programming argument, to solve temporal tasks specified by sc-LTL specifications. For complex specifications, many such constrained reachability problems must be solved. The resulting complexity typically stymies existing techniques. By leveraging the principle of superposition on the underlying repetitive task of solving optimal control problems over a common domain, we can scale to more complex tasks.

IV. Stochastic constrained reachability

We first define the *stochastic constrained reachability problem*. Such problems will later be composed to construct a control policy for a temporal logic planning problem. A solution to a constrained reachability problem is a control policy that maximizes the probability that the system reaches the boundary of set X_2 before reaching the boundary of set X_1, e.g., reach a goal region before running into obstacles. This stochastic reachability problem always has an optimal memoryless policy [4].

Problem 2 (Stochastic constrained reachability)**.** Given a system of the form (1) and sets $X_1, X_2 \subseteq X$, compute a control policy μ^* that maximizes the probability that the system reaches the boundary of set X_2 before reaching the boundary of set X_1.

Solving a stochastic constrained reachability problem is generally undecidable [7]. However, there exist numerous sound algorithms that compute solutions to constrained reachability problems using PDE-based methods [1], [14], [22], [31].

We make the standing assumption that there exists an oracle for computing a solution to a stochastic constrained reachability problem that under-approximates the probability of reaching the goal set. We denote this method by CstReach(X_1, X_2), with constraint set X_1 and reach set X_2. For a given query, CstReach returns a memoryless control policy μ^*. We now detail a method to solve the CstReach problem with an under-approximation for the case when the stochastic reachability problem has a particularly simple form.

A. The Linear Hamilton-Jacobi-Bellman Equation

In this section, we describe a technique for solving stochastic constrained reachability, i.e., Problem 2, for systems of the form (1), as a stochastic optimal control problem. The system has positive-valued costs $r(x_t, u_t) : X \times \mathbb{R}^m \to \mathbb{R}_{>0}$ accrued at time t according to

$$r(x_t, u_t) = q(x_t) + \frac{1}{2} u_t^T R u_t,$$

where $q(x)$ is a non-negative state dependent cost. The goal is to minimize the trajectory functional

$$J(x, u) = \phi_T(x_T) + \int_0^T r(x_t, u_t) dt,$$

where ϕ_T represents a state-dependent terminal cost. We consider the first-exit (or shortest path) problem, where the system continues to operate and accrue cost in the compact domain Ω until it reaches its boundary $\partial\Omega$ at time T. The choice of boundary may be due to the nature of the environment or desired operation, e.g. the presence of a goal region or an obstacle.

The construction of the constrained reachability value function $V_r(x)$ presented here follows the development and notation in [27]. The solution to the optimization, beginning from an initial point x_t at time t, is given by the value function

$$V_r(x_t) = \min_{u_{t:T}} \mathbb{E}\left[J(x_t)\right].$$

The Hamilton-Jacobi-Bellman equation associated with this problem is the nonlinear, second order partial differential equation (PDE) [13]

$$\begin{aligned} 0 \;=\; & q + (\nabla_x V_r)^T f - \frac{1}{2} (\nabla_x V_r)^T G R^{-1} G^T (\nabla_x V_r) \\ & + \frac{1}{2} Tr\left((\nabla_{xx} V_r) B \Sigma_\epsilon B^T \right), \end{aligned}$$

where $\Sigma_\epsilon = HH^T$. This nonlinear PDE is difficult to solve in general. However, it has recently been found [16], [27], [30] that under the assumption

$$\lambda G(x_t) R^{-1} G(x_t)^T = B(x_t) \Sigma_\epsilon B(x_t)^T \triangleq \Sigma_t, \qquad (2)$$

and the logarithmic transformation

$$V_r = -\lambda \log \Psi, \qquad (3)$$

one can obtain, after substitution and simplification, the linear PDE

$$0 = -\frac{1}{\lambda} q\Psi + f^T(\nabla_x \Psi) + \frac{1}{2} Tr\left((\nabla_{xx}\Psi)\,\Sigma_t\right). \quad (4)$$

The transformation of the value function V_r to Ψ (called the *desirability* [29]), provides a computationally appealing method to calculate the value function. Any solution calculated for (4) can easily be transformed to the value function via the bijection (3). We will therefore use the cost-to-go (value function) throughout. Note that any boundary conditions specified by Φ must be transformed to a boundary condition of (4) as $\Psi\,|_{\partial\Omega} = e^{-\frac{\phi}{\lambda}}$.

Remark 1. The condition prescribed by (2) relates the system's disturbances to the effect of its control input. Interestingly, the condition is satisfied when the dynamics are such that

$$dx_t = f(x_t)dt + B(x_t)\left(u_t dt + d\omega_t\right)$$

i.e. the noise enters with the control input, a common assumption in adaptive control [2]. Further analysis of the assumption is given in [29].

We are concerned about the ability for the system to satisfy the specification and not necessarily the cost to do so, giving the designer freedom with control effort R and state cost q. The state cost may be set to $q = 0$. However, it is not simple to minimize the effects of control cost. Examining constraint (2) and the transformation (3), we see that the control effort penalty R cannot be brought to zero naively. The cost-to-go thus includes a non-zero control cost, and is therefore a conservative approximation of the probability of satisfying the specification. Nonetheless, the degree of approximation introduced will be balanced by the computational gains present in this optimal control framework.

B. Numerical Methods for Solving Linear PDEs

In all but the most elementary of examples, the solution to (4) is difficult to calculate analytically, and numerical methods must be employed. Methods have also recently been developed to generate suboptimal, over-approximate solutions to (4) through the use of a relaxation on the partial differential constraints [14]. The Feynman-Kac lemma has also been used to determine the solution through sampling for high dimensional systems [26]. Recent results in the use of sparse tensor discretization have allowed for PDEs of dimension twelve and higher to be solved in a manner of minute [15]. As an alternative, it is possible to simply treat (4) as a general PDE and use existing numerical techniques such as Finite-Difference or the Finite Element Method.

V. Composition of Solutions to Linear PDEs

The superposition principle can be used to construct solutions to arbitrary specifications from the solutions of individual specifications over labeled regions. The method consists of two parts, the calculation of individual solutions for individual propositions in Algorithm 1, and the generation of solutions to a complete reachability problem in Algorithm 2.

Algorithm 1 Pre-processing of constituent solutions

Given compact domain Ω, and labeled regions $\mathcal{R} = \{R_i\}$:

1) **Set** $\partial\Phi = \{\partial\Omega, \{\partial R_i\}\}$
2) **Set** $\psi_0\,|_{\partial\Omega} = C$, null elsewhere
3) **For each** $R_i \in \mathcal{R}$
   a) Set $\psi_0\,|_{\partial R_i} = 0$
4) **Solve** for Ψ_d with PDE constraints (4) and boundary conditions $\Psi_d\,|_{\partial\Phi} = e^{-\psi_0}$
5) **For each** $R_i \in \mathcal{R}$
   a) **Solve** for Ψ_p^i according to (4) with boundary conditions $\Psi_p^i = 1$ on ∂R_i, $\Psi_p^i = 0$ on $\partial\Phi \backslash \partial R_i$
6) **Return** $\left\{\Psi_d, \left\{\Psi_p^i\right\}\right\}$

Algorithm 2 Generation of Value-maps

Given solution to Algorithm 1 $\left\{\Psi_d, \left\{\Psi_p^i\right\}\right\}$, and a set of penalties C_i for each region in $\mathcal{R}$:

1) **Initialize** value-map $\Psi^* := \Psi_d$
2) **For each** region $i \in \mathcal{I}$
   a) **Set** $\Psi^* := \Psi^* + C_i \cdot \Psi_g^i$
3) **Return** Ψ^*

The development of Algorithm 1 begins with the observation that the transformed Hamilton-Jacobi-Bellman PDE (4) is linear, which allows for superposition of solutions.

Theorem 1. *(Superposition Principle [12]) Given a pair of partial differential boundary value problems $P(\Psi_i) = 0$, $i = 1, 2$ on Ω, where P is an arbitrary linear differential operator, with boundary conditions $\Psi_1 = f$, $\Psi_2 = g$ on $\partial\Omega$, then $\Psi^* = \Psi_1 + \Psi_2$ is the solution to the boundary value problem with $\Psi^* = f + g$ on $\partial\Omega$.*

The method we propose is to solve a stochastic constrained reachability problem (i.e., Problem 2) separately for each labeled region. The activation of these regions as either goal regions or obstacles as part of a specification is then accomplished through superposition of the solutions of each individual activated region.

To begin, suppose we are given all labeled partitions as well as the domain. We first construct the *default solution* Ψ_d as that for which the boundary of the region is taken into account. The result is the boundary value problem

$$\begin{aligned} 0 &= -\frac{1}{\lambda} q\Psi_d + f^T(\nabla_x \Psi_d) + \frac{1}{2} Tr\left((\nabla_{xx}\Psi_d)\,\Sigma\right), \\ 1 &= \Psi_d\,|_{\partial\Omega}, \\ 0 &= \Psi_d\,|_{\partial\Phi\backslash\partial\Omega}\,. \end{aligned}$$

The next step is the calculation of a solution primitive Ψ_p for an individual labeled region R. Recall that such solutions will be added, and it is therefore necessary that the solution

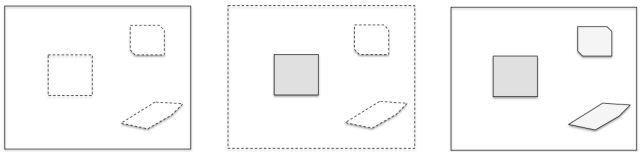


Fig. 2. Illustration of the construction of a solution primitive. On the left the solution is created for the boundary-only problem with the boundaries of the labeled regions set to zero, indicated by dashed lines. In the middle, the PDE is solved for one region activated, while all other regions and the boundary are set to zero. On the right, a complete solution is shown with the solutions added.

not alter the boundary values at other labeled regions. Thus, the boundary conditions for this problem are set to

$$\Psi_p \mid_{\partial\Phi \setminus \partial R} = 0.$$

Note that these conditions are also set for the domain boundary $\partial\Omega \subset \partial\Phi$. When solutions constructed in this manner are superimposed, the resultant boundary conditions then have the correct values. The approach is illustrated graphically in Figure 2.

Remark 2. We note that composing solution in this manner is not a form of averaging the solution primitives. The principle of superposition dictates that the solution will exactly match the composite problem.

VI. Temporal Planning Solution

In this section we present a method for (conservatively) solving Problem 1. We exploit the fact that every syntactically co-safe LTL formula can be represented by a deterministic finite automaton. A word, i.e., a labeled system trajectory, satisfies an sc-LTL formula if and only if the acceptance condition of the automaton holds. This reduces Problem 1 to computing a control policy that maximizes the probability that the system reaches an accepting state in the automaton. We modify the deterministic finite automaton by including stochastic transitions between modes, which accounts for uncertainty due to the stochastic system dynamics.

We use a deterministic finite automaton (DFA) corresponding to the sc-LTL specification to guide the computation of a control policy that (conservatively) solves Problem 1. Informally, the modes (i.e., states) in the DFA represent progress towards the completion of the task, and our goal is to compute a control policy that maximizes the probability that the system will reach an accepting mode of the automation from its initial state. However, transitions between modes in the automaton are not deterministic due to the system's stochastic noise. Thus, we construct an MDP corresponding to the DFA, where the actions correspond to memoryless control policies that are executed by the continuous system. A control policy selects the appropriate action (i.e., memoryless control policy) at each mode in the automaton.

A. Automata-Guided Task

At each node of the automaton, the specification is reduced to a set of constrained reachability tasks over the set of regions, with their reach or avoid nature indicated by the relevant propositions. Given the output $\mathcal{O}$ of Algorithm 1, Algorithm 2 generates the value-map for the current task specified by the node. The constrained reachability regions are selected, and based on their manifestation in the proposition, appropriately scaled by some constant C_i. For each region these scalings may be collected into a vector of coefficients $\mathcal{E} = \{C_i\}_{i=1,\dots,|\mathcal{R}|}$. The solutions $\Psi = \{\Psi_i\}$ to these individual solutions are then added and scaled appropriately to produce the desired solution

$$\Psi^* = \mathcal{E}^T \cdot \Psi$$

Once the goal region is reached, the current node on the graph is transitioned according to which goal was achieved, and the process repeats.

B. Dynamic Programming

As the completion of the specification of Problem 1 involves multiple individual steps, each corresponding to Problem 2, future goals must be weighted according to their future reward when planning in the current time step. By Bellman's Principle of Optimality, this suggests that at stage i the goals should be weighted according to their cost-to-go when beginning stage $i+1$. By proceeding from the accepting state, it is therefore possible to perform dynamic programming for this problem by setting the boundary conditions, i.e. the vector $\mathcal{E}$, for precedent problems according to their subsequent cost-to-go. Further analysis of the relationship between HJB boundary conditions and temporal planning is available in [11].

Dynamic programming in this setting therefore proceeds as follows. The final accepting states of a DFA representing the temporal specification are selected. For each accepting state, the corresponding regions of the domain are activated as goal regions with zero cost-to-go, and this final-stage stochastic reachability problem is then solved using Algorithm 2, with the other labeled regions inactive. This yields the solution to the final stage of the temporal planning problem.

The process then repeats recursively as follows. For each parent node of the automaton accepting states, the cost-to-go at the boundaries of the other labeled regions are selected. These values are then used as boundary conditions for the constrained reachability problem of the parent state of the automaton. The process is repeated, proceeding backwards in time, with the the boundary conditions of the labeled regions at each automaton state corresponding to the cost-to-go for those regions at the child automaton state. Each such constrained reachability problem is solved via superposition by weighting the boundaries according to their cost-to-go using Algorithm 2. This process is repeated until the beginning of the specification is reached.

C. Limitations of Our Approach

Several simplifying assumptions are necessary in the creation of the superposition framework that introduce a degree of conservatism. The first of these is that the boundary

conditions remain consistent for all reachability problems, requiring that boundary conditions for all regions be prescribed. Unfortunately, this prevents the elimination of some regions, for example if an obstacle is not necessarily to be avoided at some stages of the task. A straightforward method to fix the issue is to penalize the boundary conditions until they achieve a cost-to-go above the surrounding domain, i.e. such that $\frac{\partial \Psi}{\partial n} \mid_{O_i} \geq 0$ for n a normal vector on the surface of O_i. The effect is that the cost-to-go of reaching this deactivated region is higher than it would otherwise be. Although this will result in approximation error, the approach only overstates the expected cost of reaching the goal, resulting in a unnecessarily conservative but sound policy.

A second caveat is that the efficient construction of value functions also required boundary of each region be weighted according to a constant value, an unlikely situation. To facilitate this, we weight each region $\mathcal{R}$ according to its *worst-case* cost-to-go. This again results in a conservative but sound policy, with the solution treating some boundary values with a higher cost than is truly the case.

A last difficulty arises in that it is impossible to detect the exact cost-to-go from the boundary of the previous stage region. This is prevented by steps 2 and 3 of Algorithm 2, which prescribe boundary conditions along all labeled regions, preventing us from calculating what the cost-to-go would be if that region was not present. It is possible to raise the boundary condition from the region whose cost-to-go is to be approximated until it is greater than all neighboring states, i.e. such that the gradient away from the region is negative. This implies that the cost to go from the boundary of the region is greater than it would have been otherwise, and provides, again, an under approximation of the satisfaction probability.

D. Introduction of New Propositions

Once Algorithm 1 has been completed, it is still possible to add a newly created region. The need could arise from varying requirements over the course of execution, due to perhaps the introduction of a previously unseen obstacle, or if it desirable to completely eliminate a region.

Given an existing specification φ with solution Ψ, we wish to add a proposition a that an unlabeled region A. We first wish to adjust the value of the solution to match $\phi(x) \mid_{\partial A} = C$. The existing cost-to-go is captured from the boundary ∂A, and the boundary value problem is solved with boundary conditions

$$\begin{aligned} \phi \mid_{\partial A} &= \Phi(x), \\ \phi \mid_{\partial \Phi \setminus \partial A} &= 0, \end{aligned}$$

with solution denoted Ψ_e. This captures the effects of the existing solution along ∂A upon the rest of the solution. The process is repeated with the new boundary conditions

$$\begin{aligned} \phi \mid_{\partial A} &= 1, \\ \phi \mid_{\partial \Phi \setminus \partial A} &= 0, \end{aligned}$$

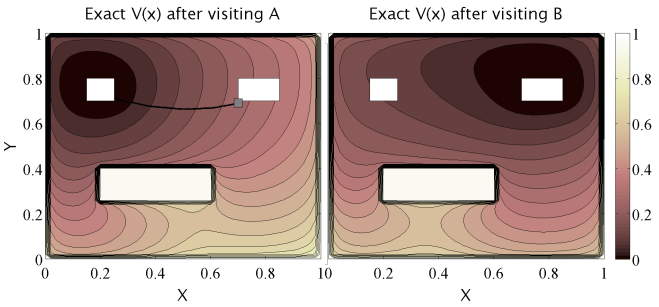


Fig. 3. Results for last stage cost-to-go of Example 1. The domain is $x, y \in [0,1]^2 \backslash C$ for $C = \{x \in [0.2, 0.6], y \in [0.25, 0.4]\}$. The goals are $A = \{x \in [0.7, 0.85], y \in [0.7, 0.8]\}$, $B = \{x \in [0.7, 0.8], y \in [0.15, 0.25]\}$. A trajectory of the closed loop system begins at the grey square.

producing solution Ψ_n. The new solution with the added boundary conditions may then be constructed. Note however that this new solution is particular to the boundary conditions prescribed on $\partial\Phi$, and cannot be scaled and used as part of the superposition framework.

E. Complexity

The primary cost in this framework is the calculation of the solution to the PDE (4), which must be computed once for each element of $\mathcal{R}$, and once more for all elements $\mathcal{R}$ set to zero. The Finite Difference Method requires $\mathcal{O}\left(\left(\frac{1}{h}\right)^d\right)$ for discretization length h and state space dimension d. Methods based on Monte Carlo sampling, such as Feynman-Kac, are known to have accuracy that scale independently of state space dimension at $\mathcal{O}(n^{\frac{1}{2}})$ where n are the number of samples used in the estimation process. More recently, methods to directly solve linear PDEs that scale linearly with dimension, but cubically in the tensor-rank of the solution, have also appeared [5], [15].

The computation of the deterministic finite automaton for the specification is worst-case doubly exponential [6]. The convergence of the value iteration for the specification MDP is guaranteed as the continuum of actions exists in a compact set, and the costs of the actions are non-negative [4]. Investigation of the specific convergence rate in this context is the subject of future research.

The gains of the framework presented lies in the composition of solutions. At each step in the automaton it is necessary to calculate the solution to a constrained reachability problem. This is done through simple vector addition of the primitives in Algorithm 2. As all primitives are used, this is an $\mathcal{O}(|\mathcal{R}|)$ operation, but with a quite small constant as vector addition is computationally negligible. Denoting the method of calculating an individual PDE solution as having complexity $\mathcal{O}(p)$, Algorithm 1 requires $\mathcal{O}(|\mathcal{R}|\, p)$ time.

VII. Examples

We illustrate the approach on two examples. The first example illustrates our method, and the second example shows how this compositional approach scales with task complexity. For simplicity, the finite difference method is used to solve

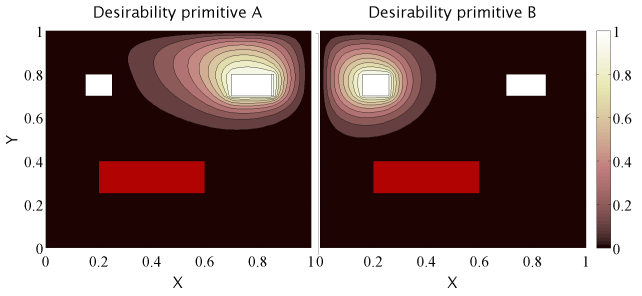


Fig. 4. Individual value function primitives for regions A, B on the left and right respectively.

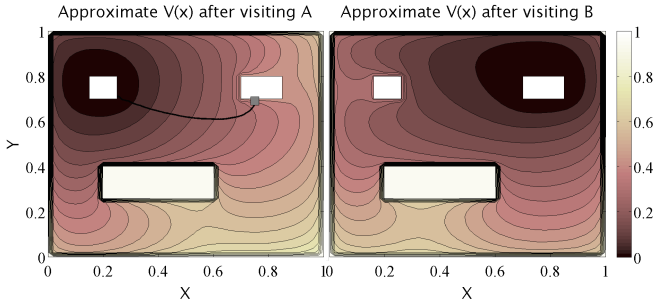
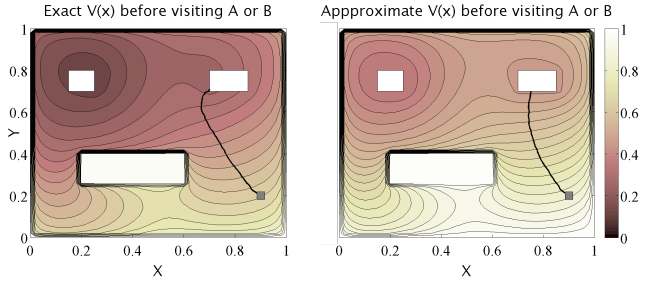


Fig. 5. Results of using composition method on the two region visit task of Example 1.

all PDEs. The standard approach to similar problems rely on a discrete abstraction with transition probabilities gained from Monte Carlo simulation [19]. These simulations create a high computational burden, taking as much as several hours, as well as approximation error in the model. We are able to avoid these issues in our approach, with computation time in the tens of seconds. In both of these examples we use a nonlinear two dimensional example, as it facilitates visualization, and demonstrates the generality of the method.

A. *Two goal problem*

The first example is to visit two goal regions A and B while remaining in an obstacle-free bounded domain S. The



(a) (b)

Fig. 6. Exact and approximate cost-to-go before either region is visited on the left and right respectively. A trajectory, beginning from the grey square is shown in black when following the induced policies. After visiting A the trajectories are continued in Fig. 3, 5.

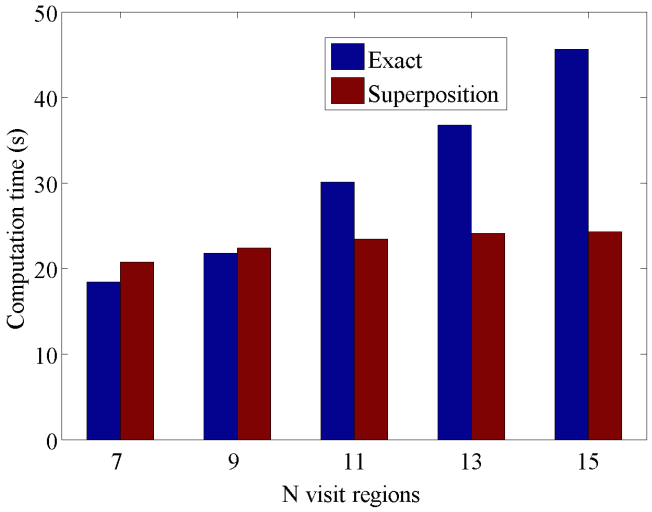


Fig. 7. Calculation times using the exact (blue) and superposition (red) methods for visitation problem with n regions to visit.

formal specification is $\varphi = (S \,\mathcal{U}\, A) \wedge (S \,\mathcal{U}\, B)$ and the system dynamics, taken from [24] are given by

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \left(\begin{bmatrix} -2x - x^3 - 5y - y^3 \\ 6x + x^3 - 3y - y^3 \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) dt + \begin{bmatrix} d\omega_1 \\ d\omega_2 \end{bmatrix}$$

The state cost is set to $q = 0.4$ with control penalty is $R = 0.05 I_{2\times 2}$. In Figure 3 the geometry is shown along with the cost-to-go in the last stage of the automaton after having visited one of the goals. The value function for the first stage is calculated and shown in Figure 6a.

The problem is then repeated using the compositional approach. The primitives are shown in Figure 4. These are superimposed to produce the solution from the last stage, shown in Figure 5, where the worst-case cost-to-go is applied to the last visited region. These worst-case values are then used as the boundary conditions for the first-stage value function, shown in Figure 6b.

B. *n−visit problem*

We expand upon the previous example, now scaling the number of goal regions n up to ten in a larger domain $\Omega = [0, 130]^2$ with discretization size $h = 1.0$, and goals equally spaced with width four. Specifically, the specification is $\varphi = (S \,\mathcal{U}\, A_1) \wedge \cdots \wedge (S \,\mathcal{U}\, A_n)$.

The problem can be solved exactly and using the method of superposition as before. As the exact approach requires the solution of a PDE for each edge of the automaton, the computation also scales with the size of the automaton. In contrast, using superposition the policy at each automaton node requires only vector additions and a maximization operation over a vector that describes the region boundaries. Calculation times for the two are shown in Figure 7.

VIII. Conclusions

We introduced a method for efficiently synthesizing control policies for stochastic nonlinear systems with syntactically co-safe LTL specifications. A certain structural assumption was made on the stochasticity of the system, allowing for the construction of a linear Hamilton Jacobi Bellman equation for individual stochastic constrained reachability problems. In turn, this allowed for temporal planning problems to be solved efficiently via the principle of superposition. The method relies on pre-computed primitives, each of which are solutions to individual stochastic optimal control problems related to the reachability of an individual region. Solutions to individual constrained reachability problems are cheaply constructed by simple vector addition of these pre-computed solutions, allowing for the individual stages of an automaton specified task to be solved quickly. The method relies on computation in the state space of the system and requires no a-priori discretization or cellular decomposition.

The drawbacks of this method are that the PDE utilized requires all boundary conditions to be specified. A region may be removed exactly, but this requires the solution of an additional PDE boundary value problem, and is not applicable when any of the other boundary conditions are changed. The need to incorporate all regions, it is possible to construct a sound but conservative solution with mild penalty on the inactive regions. While inexact, the method has many benefits, among which is the ability to rapidly adapt to new specifications over the existing labeled regions in real time.

IX. Acknowledgements

The authors thank the anonymous reviewers for their helpful comments. The first and second authors were supported by NSF GRFP and NDSEG fellowships, respectively. Additional support was provided by the Boeing Corporation.

References

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[2] K. J. Åström and B. Wittenmark. *Adaptive Control*. Courier Dover Publications, 2008.

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

[4] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16:580–595, 1991.

[5] G. Beylkin and M. J. Mohlenkamp. Algorithms for numerical analysis in high dimensions. *SIAM Journal on Scientific Computing*, 26(6):2133–2159, 2005.

[6] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *IEEE International Conference on Robotics and Automation*, 2010.

[7] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

[8] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[9] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. LTL control in uncertain environments with probabilistic satisfaction guarantees. In *Proc. of 18th IFAC World Congress*, 2011.

[10] A. D'Innocenzo, A. Abate, M. D. D. Benedetto, and S. Sastry. Approximate abstractions of discrete-time controlled stochastic hybrid systems. In *Proc. of IEEE Conf. on Decision and Control*, pages 221–226, 2008.

[11] P. M. Esfahani, D. Chatterjee, and J. Lygeros. Motion planning via optimal control for stochastic processes. *Preprint arXiv:1211.1138*, 2012.

[12] L. Evans. *Partial Differential Equations*. American Mathematical Society, 2010.

[13] W. H. Fleming and H. M. Soner. *Controlled Markov processes and viscosity solutions*, volume 25. Springer, 2006.

[14] M. B. Horowitz and J. W. Burdick. Semidefinite relaxations for stochastic optimal control policies. In *American Controls Conference (ACC)*, 2014.

[15] M. B. Horowitz, A. Damle, and J. W. Burdick. Linear Hamilton Jacobi Bellman equations in high dimensions. *IEEE Conference on Decision and Control (Submitted)*, 2014. arXiv:1404.1089.

[16] H. Kappen. Linear Theory for Control of Nonlinear Stochastic Systems. *Physical Review Letters*, 95(20):200201, Nov. 2005.

[17] O. Kupferman and M. Y Vardi. Model Checking of Safety Properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[18] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: verification of probabilistic real-time systems. In *Proceedings of 23rd International Conference on Computer Aided Verification*, 2011.

[19] M. Lahijanian, S. B. Andersson, and C. Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Trans. on Robotics*, 28:396–409, 2012.

[20] T. Latvala. Efficient model checking of safety properties. In *Model Checking Software*, pages 74–88. Springer, 2003.

[21] S. M. LaValle. *Planning Algorithms*. Cambridge press, 2006.

[22] I. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *J. Sci. Comput.*, 35:300–329, 2008.

[23] B. Oksendal. *Stochastic differential equations: an introduction with applications*. Springer-Verlag, New York, 1998.

[24] S. Prajna and A. Papachristodoulou. Analysis of switched and hybrid systems-beyond piecewise quadratic methods. In *Proceedings of the American Control Conference (ACC)*, volume 4, pages 2779–2784, 2003.

[25] C. Schwab and C. J. Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic pdes. *Acta Numerica*, 20:291–467, 2011.

[26] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 9999:3137–3181, 2010.

[27] E. Theodorou, F. Stulp, J. Buchli, and S. S. An iterative path integral stochastic optimal control approach for learning robotic tasks. In *18th IFAC World Congress*, volume 18, pages 11594–11601, 2011.

[28] I. Tkachev, A. Mereacre, J.-P. Koatoen, and A. Abate. Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems. In *Proc. of Hybrid Systems: Computation and Control*, pages 293–302, 2013.

[29] E. Todorov. Compositionality of optimal control laws. *Advances in Neural Information Processing Systems (NIPS)*, pages 1856–1864, 2009.

[30] E. Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009.

[31] C. J. Tomlin, I. M. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proc. IEEE*, 91:986–1001, 2003.

[32] E. M. Wolff, U. Topcu, and R. M. Murray. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *Proc. of Int. Conf. on Intelligent Robots and Systems*, pages 4332–4339, 2013.