# A Multi-layered Synergistic Approach to Motion Planning with Complex Goals

Amit Bhatia, Matthew R. Maly, Lydia E. Kavraki, and Moshe Y. Vardi

***Abstract*—This paper describes an approach for solving motion planning problems for mobile robots involving temporal goals. The temporal goals are described over subsets of the workspace (called propositions) using temporal logic. The approach uses an instantiation of a multi-layered synergistic planning framework that has been proposed recently. In this framework, a high-level planner constructs high-level plans using a discrete abstraction of the system, the temporal logic specifications and the low-level exploration information. A low-level sampling-based planner uses the suggested high-level plans and the dynamics of the system to explore the state-space of the system for feasible trajectories satisfying the specification. The construction and exploration of the discrete abstraction are critical issues that affect the overall performance of the approach. A geometry-based approach for constructing the abstraction, and a lazy high-level search technique for its exploration are discussed. The proposed techniques result in computational speedups of close to 10 times over earlier approaches for second-order nonlinear robot models in challenging workspace environments with obstacles and for a variety of temporal logic specifications.**

## I. Introduction

Traditional motion planning for mobile robotic systems involves the construction of a motion plan for the system that takes the system from an initial state to a set of goal states, while avoiding collisions with the obstacles at all times. The motion plan is also required to respect the dynamics of the system that are typically described by a set of differential equations. A wide variety of techniques have been proposed over the last two decades that try to solve such problems [1], [2].

Motivated by the desire to increase the capabilities and automation of robotic systems, and the success in solving conventional motion planning problems, there have been recent research efforts that try to answer the following question: *"Given state-of-the-art motion planning algorithms, how can we use them as a basic building block for solving motion planning problems involving complex goals and robots with complex dynamics?"* In Figure 1[1], we show an example of one such planning problem.

Clearly, planning problems involving complex goals are not a trivial extension of the traditional motion planning problem. A variety of ideas have been proposed from AI, control theory, formal methods, and hybrid systems communities for solving such problems [3]–[29]. Much of the earlier work has focused on discrete planning within the AI community [3]–[6]. The system is typically modeled as a graph and is called a transition system. The nodes represent states and the edges represent transitions between states and are labeled by actions that enable the transitions. The goal specification is described using formalisms like STRIPS [3] and Action Languages [4], [5]. Among many others, the approaches proposed in [7]–[11] try to combine motion planning with task planning. The work in [12] is one of the earlier attempts to formalize the concept of motion of robots using ideas from control theory. This has been followed by the notion of Motion Description Languages [13], [14], [19] and Maneuver Automaton [15], [19].

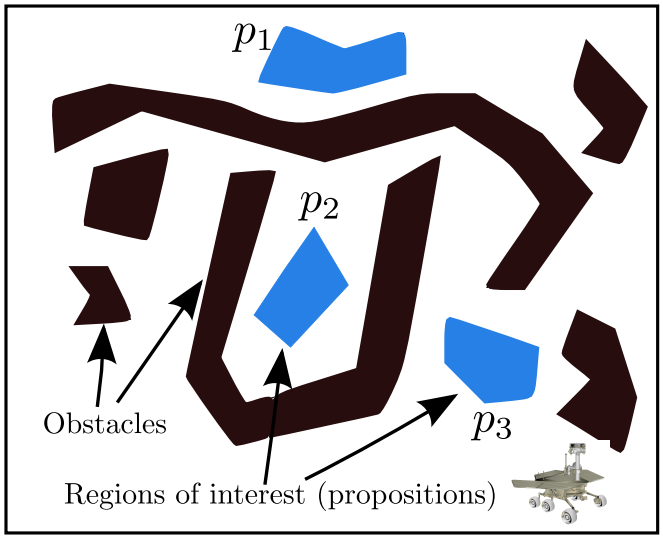



Fig. 1: An example of planning with complex goals. The sets shown in blue correspond to regions of interest. The sets shown in brown are obstacles. The goal specification is *"In future, visit the region where $p_1$ is true, and then visit a region where $p_2$ or $p_3$ is true."*

A class of complex goals impose temporal constraints on the trajectories of a given system. Such goal specifications are referred to as *temporal goals* in this paper. The goal specification described in Figure 1 is an example of such goals. A variety of model-checking inspired approaches have been proposed for solving planning problems involving mobile robots and temporal goals [17]–[30]. These approaches differ from some of the earlier works, e.g., [16], in that the planning problem is considered for mobile robots with dynamics. The proposed approaches are inspired by state-of-the-art techniques used for model-checking computer programs by the formal methods community [31], [32]. The temporal goals are described using a formal framework, e.g., Linear Temporal Logic (LTL, [33]), Computation Tree Logic (CTL, [34]), and $\mu-$calculus [35].

Amit Bhatia, Matthew Maly, Lydia Kavraki and Moshe Vardi are with the Department of Computer Science, Rice University, Houston, Texas, 77005, {abhatia,mmaly,kavraki,vardi}@rice.edu

[1]We recommend an online version for viewing all the figures in the paper.

While the idea of extending the discrete system semantics to continuous and hybrid systems has been investigated [36], [37], most of the approaches implement instantiations of the the following two-layer architecture. At a discrete level, a discrete plan is constructed using a discrete abstract model of the robot, and the formal specifications. Model-checking techniques are used to construct such a plan. The constructed plan is then used by the continuous layer to construct a physically feasible trajectory for the robot.

The specification language, the discrete abstraction of the robot model, and the planning framework depend on the particular problem being solved and the kind of guarantees required. One of the earliest works that used temporal logic as the specification language for the synthesis of controller programs for robotic and manufacturing tasks is [17]. The work in [18] investigates the problem of automatic controller synthesis for a team of mobile robots with high-level objectives described using LTL. LTL has also been used to solve multi-robot motion planning problems in [22], [23]. The issue of construction of a suitable discrete abstraction has been studied independently and extensively in the formal methods, robotics, hybrid systems, and control theory communities [19], [37]–[51]. Ideally, one would like to construct discrete abstractions that are *equivalent* to the exact model in terms of observed behaviors (i.e., bisimilar [41]). However such abstractions are known to exist only for very simple robot models where the dynamics are essentially linear [41]. For most models of interest, the exact equivalence is typically relaxed to an approximate one [46]–[51]. Furthermore, it is not clear if the geometric constraints arising due to robot geometry and the obstacles can be incorporated within such notions. An approach that uses local controllers for motion planning with temporal goals has also been proposed recently in [20], [21].

Inspired by the success of sampling-based algorithms in solving conventional motion planning problems [1], [2], a complementary set of techniques have been proposed over past few years that use sampling-based algorithms for safety analysis and motion planning of hybrid and robotic systems [24], [27]–[29], [52]–[59]. An important feature of sampling-based approaches is that the required controllers for feasible trajectories are automatically constructed as a result of the exploration process. Hence the approaches typically do not require the existence of a particular class of controllers.

The earliest work on using sampling-based algorithms for safety falsification [52] has been followed by more recent works that try to improve the scalability of such algorithms [57], [58]. The work in [53] has considered traditional motion planning problems but with hybrid robot dynamics. Extensions of a class of sampling-based algorithms for planning, control and verification of hybrid and robotic systems, have been proposed in [54]. The work in [54] has also proposed an extension of such algorithms for solving planning problems in discrete spaces. The approach proposed in [56] combines sampling-based algorithms with sensitivity analysis for verification of high dimensional nonlinear systems. The approaches proposed in [24], [27]–[29] use sampling-based algorithms within sophisticated frameworks for planning problems involving complex specifications. The approach proposed in [24] can be used to solve motion planning problems involving $\mu-$calculus specifications. An important contribution of this approach is that instead of relying on a fixed abstraction of the system, a sampling-based abstraction refinement technique is used. The refinement procedure is based on incremental sampling of the system trajectories. In contrast to single-layered [24] and hierarchical approaches [20]–[22], [25], [30], the framework proposed in [27] is a multi-layered synergistic framework that can be used for planning and safety falsification of hybrid and robotic systems with LTL specifications.

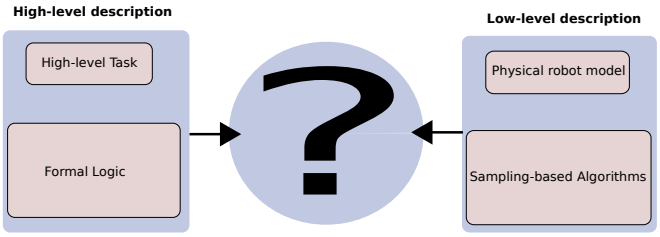



Fig. 2: Combining logic with sampling-based algorithms

The approaches proposed in [24], [27]–[29], [52]–[54], [56]–[59] trade strong completeness guarantees for scalability and efficiency. This means that while such approaches can deal with complexities arising due to model nonlinearities and geometric constraints, they may fail to find a solution in finite time (even if one exists). A class of sampling-based algorithms have also been proposed recently that provide stronger completeness guarantees for safety analysis [55], [60].

This paper presents a summary of the work in [28], [29]. The approach described in this paper is motivated by the following question (Figure 2): *"Given state-of-the-art model checking techniques, and sampling-based motion planning algorithms, how can the two be combined for solving motion planning problems involving complex goals and robots with complex dynamics?"* The focus is on solving problem instances involving nonlinear robot models with finite geometry, complex workspace environments, and high-level temporal goals. In this paper, recent research efforts towards solving such problems *efficiently* using a multi-layered synergistic framework that has been proposed recently for solving a variety of planning problems [27], [58], [59] are discussed. An important focus of the current work is on addressing the scalability issues, both in terms of the complexity of the robot model, and the complexity of high-level specifications.

## II. Multi-layered Synergistic Planning

The work on multi-layered synergistic planning is an instantiation of the recently proposed planning paradigm [27], [58], [59]. The paradigm has been used previously for safety analysis of hybrid systems with reachability specifications [58] and for conventional motion planning involving complex mobile robot models and environments [59]. The framework is inspired by earlier works [61], [62] that introduced a discrete search component for solving planning problems. The framework introduces a discrete component to the search procedure by synergistically utilizing the discrete structure present in the problem. The framework consists of the following steps: a) Construction of a discrete abstraction for the system, b)

High-level planning for the abstraction using the specifications and the exploration information from the low-level planner, c) Low-level sampling-based planning using the physical model and the suggested high-level plans. Note that there is a two-way exchange of information between the high-level and the low-level planning layers in steps b) and c). This kind of synergy helps to systematically convey information regarding physics of the problem from the low-level layer to the high-level layer. The constraints arising due to temporal goals are systematically conveyed to the low-level layer from the high-level layer using synergy.

The construction of the discrete abstraction and the two-way synergistic interaction between the layers are critical issues that affect the overall performance of the approach. It has been experimentally shown in [27], [28] that in the absence of synergy, the overall approach does not scale. As part of ongoing research, Bhatia, Kavraki and Vardi have been investigating the issue of construction of the discrete abstraction while trying to answer the following question: *"How should the discrete abstraction be constructed and explored in the high-level search layer such that the overall performance of the approach improves?"* While the idea of breaking the planning problem into multiple layers is not new, the proposed approach differs from related approaches [22], [24], [25], [61]–[63] in that there is a two-way, synergistic interaction between different layers of planning.

### A. Basic Framework

The instantiation of the multi-layered synergistic framework used for motion planning is shown in Figure 3. Below is a short description. For more details the reader is referred to [28], [29].

*a) Framework for system and specifications:* The robot is modeled as a dynamical system driven by exogenous inputs and is denoted by $H$. The model can be either continuous or hybrid. In all cases, the robot is assumed to have finite geometry. The state-space of the system is denoted by $S$. Let $\Pi = \{p_0, p_1, p_2, \ldots, p_N\}$ denote the set of boolean atomic propositions. Each proposition denotes a region of interest in the workspace for the robot. Every such region is referred to as a *propositional set*. $p_0$ denotes the free region of workspace where no other propositions are true. The planning problems considered in the work have a finite horizon. A particular class of LTL formulas called *co-safe* LTL formulas can be used to describe finite horizon specifications of the system [64]. Co-safe LTL formulas are the LTL formulas such that any good trace satisfying the formula has a finite good prefix. A finite good prefix for a formula is a finite prefix such that all its trace extensions satisfy the formula (cf. [64]). The co-safety formulas are the same as the *guarantee* formulas introduced in [65]. The temporal goals considered in the work are expressed as syntactically co-safe LTL formulas using atomic propositions. Syntactically co-safe LTL formulas are the LTL formulas that contain only the *Next* ($X$), *Until* ($U$), and *Finally* ($F$) operators, when written in positive normal form (i.e., the negation operator $\neg$ occurs only in front of atomic propositions, see [64] for more details). As an example, the specification described in Figure 1, *"In future, visit the region where $p_1$ is true, and then visit a region where $p_2$ or $p_3$ is true"* can be written as the LTL formula $\phi = F(p_1 \wedge F(p_2 \vee p_3))$. Given a syntactically co-safe LTL formula $\phi$, it has been shown that a Non-deterministic Finite Automaton (NFA) $\mathcal{A}_\phi$ can be constructed (with at most exponential blowup) that describes all the finite good prefixes satisfying $\phi$ [64]. For the LTL formula described above, the corresponding NFA is shown in Figure 4. It has been shown recently that using a minimized Deterministic Finite Automaton (DFA) for an NFA can offer significant computational speedups for model checking and falsification of temporal specifications for hybrid systems (cf. [27], [66]). In light of this result, a minimized DFA is used in the proposed approach.

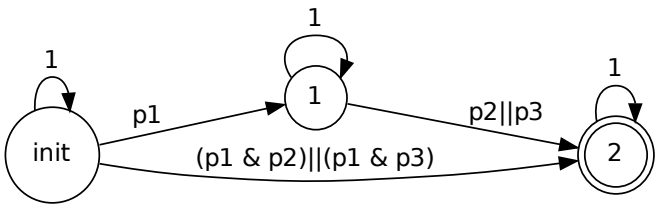



Fig. 4: NFA $\mathcal{A}_\phi$ describing all the finite good prefixes for the syntactically co-safe LTL formula $\phi = F(p_1 \wedge F(p_2 \vee p_3))$.

*b) High-level search layer:* This is the layer that handles most of the discrete nature of the problem. Given a robot model $H$, a discrete abstraction $M$ of the system is constructed. The abstraction $M$ contains a set $D$ of states and the transition relations between the abstract states. The abstraction techniques proposed in [28], [29] are based on decomposition of the workspace. However, alternate abstraction techniques that use additional information about the system (e.g., dynamics) can also be used in the framework. Given a decomposition of the workspace, the elements of the decomposition correspond to the states of the abstraction. The transition relations between abstract states are determined by using adjacency information among the elements of the decomposition. Every system state is mapped to an abstract state by first projecting it onto the workspace and then using the workspace decomposition to map it to the set $D$. Further details on the construction of the discrete abstraction are presented in Section II-B.

Given a specification $\phi$ defined over the set of propositions $\Pi$, the high-level layer searches the product $\mathcal{A}_\phi \times M$ for promising high-level guides. To construct such guides, ideas and tools from model checking and graph search algorithms are used [31]. The idea of using the product $\mathcal{A}_\phi \times M$ for discrete planning has also been used before [22], [25], [63]. An important difference in the proposed approach is that the vertices and edges in the graph representation of $\mathcal{A}_\phi \times M$ are assigned weights. These are used to synergistically convey the low-level exploration information to the high-level layer. Every high-level state $(z, d) \in \mathcal{A}_\phi \times M$ is assigned a feasibility estimate $\rho(z, d)$. Here $z$ is a state of $\mathcal{A}_\phi$ and $d$ is a state of the abstraction. An edge connecting a pair of high-level states $((z_i, d_i), (z_j, d_j))$ in $\mathcal{A}_\phi \times M$ is assigned the weight $(\rho(z_i, d_i) \cdot \rho(z_j, d_j))^{-1}$. The edge weight estimates the feasibility of transition between the corresponding high-level

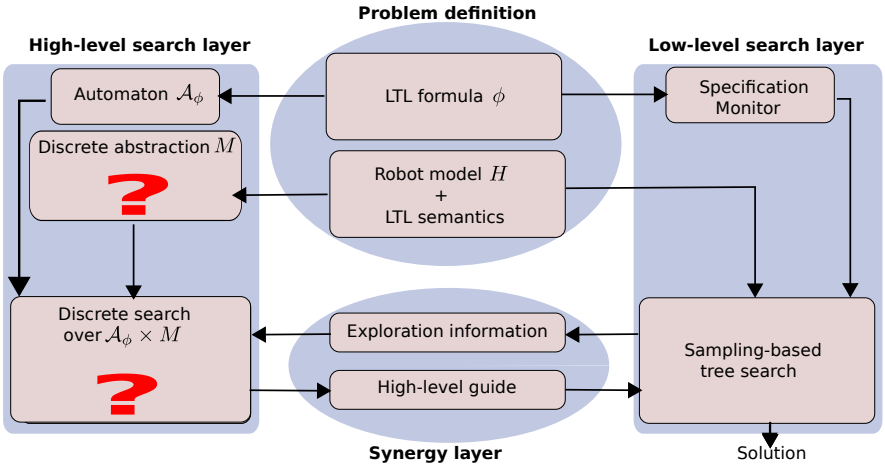



Fig. 3: Synergistic multi-layered approach for motion planning with LTL specifications

states $(z_i, d_i)$ and $(z_j, d_j)$ while respecting the dynamics of the robot. Further details on the real-valued function $\rho$ are discussed as part of the synergy layer. A high-level guide is computed as the shortest path from an initial state of $\mathcal{A}_\phi \times M$ to the set of accepting states using Dijkstra's algorithm. To account for the fact that the weights are an estimate of feasibility, the high-level search layer also computes a random path occasionally, which need not be the shortest one.

*c) Low-level search layer:* A high-level guide $\zeta$ constructed by the high-level search layer may or may not be feasible for the robot. This is checked incrementally at the low-level search layer by exploring the state-space $S$ of the robot. The suggested high-level guide $\zeta$ is used to bias the search such that the resulting exploration of $S$ improves the chances of finding a feasible trajectory satisfying the LTL specification $\phi$. The low-level exploration is done by repeatedly selecting a high-level state from the high-level plan $\zeta$. The probability of selecting a high-level state $(z, d)$ is proportional to its feasibility estimate $\rho(z, d)$ (described as part of the synergy layer). Let $\mathcal{V}(z, d)$ denote the set of tree vertices that belong to the high-level state $(z, d)$. A tree vertex $v \in \mathcal{V}$ is more likely to be selected for exploration if it has been explored a fewer number of times in the past. The search is done for a predetermined exploration time $t_{explore}$ using highly successful sampling-based algorithms that build an exploration tree in $S$ while keeping estimates for the coverage of $S$. The sampling-based algorithm used in the current work is similar to the one used in [27], but others, e.g., [52]–[54], [56], [57] can also be used. It is also possible to incorporate control-theoretic techniques (e.g., maneuver automaton for systems with symmetries [15]) into the low-level planning layer. The low-level search layer passes the exploration information to the synergy layer. For a given high-level state $(z, d)$, the exploration information comprises of the coverage estimate (REGION COVERAGE$(d)$), past exploration history (PAST HISTORY$(z, d)$), and the total volume of all system states that map to the high-level state $(z, d)$ (REGION VOLUME$(d)$). Computation of each of these terms is discussed in detail as part of the synergy layer. The low-level layer also uses the automaton $\mathcal{A}_\phi$ as a specification monitor to identify when a feasible trajectory has been found during the sampling-based exploration using the acceptance condition of the automaton and the sampling-based search tree. For further details, the reader is referred to [28], [29].

*d) Synergy layer:* The synergy layer is responsible for aggregating the information provided by the low-level search layer to information that can be used by the high-level search layer. A synergistic interaction between the different search layers is facilitated by using the feasibility estimate $\rho(z, d)$ associated with each high-level state $(z, d) \in \mathcal{A}_\phi \times M$. The feasibility estimates capture the constraints arising due to dynamics of the system and the LTL specification, through combination of the coverage estimate (REGION COVERAGE$(d)$), past exploration history (PAST HISTORY$(z, d)$), total volume of all system states that map to the high-level state $(z, d)$ (REGION VOLUME$(d)$), and the shortest distance of $z$ from an accepting state of the automaton $\mathcal{A}_\phi$ in terms of number of transitions (AUTOMATON COST$(z)$). Given a high-level state $(z, d)$, the feasibility estimate $\rho(z, d)$ is a real-valued function of the following form:

$$\rho(z, d) \propto \frac{\text{REGION COVERAGE}(d) \cdot \text{REGION VOLUME}(d)}{\text{AUTOMATON COST}(z) \cdot \text{PAST HISTORY}(z, d)}.$$

The functional form of the feasibility estimate is based on our investigations and previous experience of using such estimates. However, this is not the only possible instantiation. Further improvements could be made to the synergy layer, e.g., by accounting for the dynamics and the actuation limits of the system in the estimate. A variety of techniques to estimate coverage have been proposed in sampling-based motion planning literature, e.g., by overlaying a uniform grid over state-space [58] or using notions of dispersion [67], discrepancy [67]–[69], star discrepancy [57], and mutual distance [70]. In our work, we estimate coverage by imposing a grid on the workspace of the robot. The past-exploration history PAST HISTORY$(z, d)$ is estimated by evaluating the number of times the high-level state $(z, d)$ has been previously selected for exploration by the low-level search layer. The volume of a set of continuous states corresponding to the abstract state $d$ is computed either exactly or approximately. As discussed earlier, an edge connecting a pair of high-level states $((z_i, d_i), (z_j, d_j))$ in $\mathcal{A}_\phi \times M$ is assigned the weight

$(\rho(z_i, d_i) \cdot \rho(z_j, d_j))^{-1}$.

If a solution is found, the search stops and the solution is returned, else the low-level layer continues the exploration with a new high-level guide suggested by the high-level layer, based on updates to feasibility estimates from the last iteration. The overall search is conducted for a predetermined exploration time $t_{max}$.

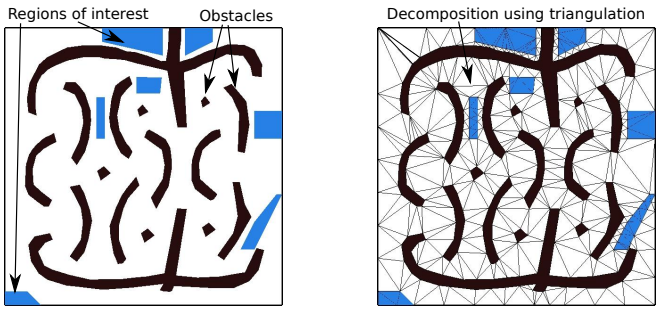
### B. Construction and Exploration of Discrete Abstraction

The construction and exploration of the discrete abstraction are critical issues that affect the overall performance of the approach. One of the main reasons for using a discrete abstract model for solving the problem is to make a quick guess of a possible solution using the discrete abstraction and then explore its feasibility for the robot model in the low-level search layer. In previous work [27], the discrete abstraction was treated as a user-supplied input. The benchmark problems considered did not involve geometric constraints, and the problem was set up such that the abstraction was easy to construct.

As part of ongoing work [28], [29], these issues have been investigated in detail. A geometry-based approach has been proposed for the construction of the discrete abstraction that is based on decomposition of the workspace of the robot. While there are many ways to decompose a given workspace, the effectiveness of the abstraction is directly related to the kind of properties it preserves from the exact robot model. A well-defined decomposition should ensure that the propositions are well defined for the abstraction (called as proposition-preserving decomposition, [25]). The simplest proposition-preserving decomposition is the one that is induced by the set of propositions and ignores the geometry of the specifications. Such a decomposition is called as *geometry ignoring*. Every element of $2^{\Pi}$ is represented by at most one element of the geometry-ignoring decomposition. The elements of the decomposition correspond to the states of the abstraction. The transition relations between abstract states are determined by checking intersection of propositional sets. While such an abstraction is relatively easy to construct, experimental results indicate that there is a significant computational advantage in using decompositions that take into account the geometry of the specifications.

The proposed approach for constructing the discrete abstraction uses the geometry of the specifications and the workspace[2]. The geometry of the workspace is used by triangulating it. We call such an abstraction *geometry using*. The workspace is given as a Planar Straight Line Graph (PSLG) to a mesh generation package (we use the Triangle package [71]). A PSLG is made of vertices and segments. Segments are edges whose endpoints are vertices in the PSLG, and whose presence in any mesh generated from the PSLG is enforced. Holes correspond to the regions that cannot be triangulated. To ensure that the resulting decomposition is proposition-preserving, the sets describing propositions are given as *segments*. The obstacles are given as *holes*. Based on previous experience in solving conventional motion planning problems [59], the workspace is triangulated using conforming Delaunay triangulation. One such decomposition for the example is shown in Figure 5.

[2]The geometry of specifications and the workspace is described by the sets describing the propositions and the obstacles.




(a) Workspace (b) Decomposition

Fig. 5: (a) Workspace with seven propositions. The sets shown in blue are labeled with propositions. (b) Triangulation-based decomposition of the workspace. The triangulation-based decomposition was done using conforming Delaunay triangulation and resulted in 618 elements in the decomposition. See also Section II-B and [28].

The dual graph of the triangulation is used to construct transition relations between abstract states. It must be remarked here that the idea of using a triangulation-based decomposition of the workspace has been used before (cf. [25], [45]). However, the abstractions used in [25], [45] need to satisfy the bisimilarity property [72], while in our work, this is not required.

The use of geometry for constructing the discrete abstractions results in abstractions that are larger in size. As an example, for the problem instance shown in Figure 5, the geometry-ignoring abstraction has 8 states while the triangulation-based decomposition shown in Figure 5 results in an abstraction with 618 states. To effectively utilize such abstractions in the high-level layer, it is also important to conduct the high-level search efficiently. The high-level search technique proposed in [27] (referred to as *reinitialized-search* technique) always starts the search from initial states of the automaton (say $z_0$) and the abstraction (say $d_0$). This can be expensive for the cases when the size of search space ($\mathcal{A}_\phi \times M$) is big. To reduce the time spent on high-level search, a *lazy-search* technique has been proposed in [29] that starts the search from $(z_0, d_0)$ only when other candidate high-level states that have been used previously do not look promising. Instead of reinitializing the search from $(z_0, d_0)$ every time a new high-level plan is being constructed, the lazy-search initializes search from previously explored high-level states. This effectively means that portions of previously explored plans are reused.

### C. Experiments

The proposed ideas have been tested for solving motion planning problems involving nonlinear robot models, complex environments with obstacles, and a variety of LTL specifications.

Second-order models of a car, a unicycle, and a differential drive, have been used as robot models in the experiments. These models are rich enough to capture the key aspects of the dynamics and have been extensively used for benchmarking

planning algorithms for mobile robots. The readers are referre to [28], [29] for more details.

To evaluate the effect of specifications on performanc three different types of LTL formulas have been considered the experiments. The first type of formulas are the coverag formulas $\phi_{\text{cov}}^{n_{\text{op}}}$. $n_{\text{op}} \in [1, 7]$ is the number of tempor operators in the formula. The coverage formulas deal wi visiting a given set of regions in the workspace witho imposing any constraints on the order of visits. The formu $\phi = Fp_1 \wedge Fp_2$ is a coverage formula with two tempor operators. A second set of formulas we have considered a the sequencing formulas $\phi_{\text{seq}}^{n_{\text{op}}}$. The sequencing formulas de with visiting a given set of regions in the workspace in a give order. The formula $\phi = F(p_1 \wedge F(p_2))$ is a sequencing formu with two temporal operators. A third class of formulas we ha considered are the strict sequencing formulas $\phi_{\text{st seq}}^{n_{\text{op}}}$. The strict sequencing formulas deal with visiting a given set of regions in the workspace in a given order that is to be respected strictly. The formula $\phi = F(p_1 \wedge ((p_0 \vee p_1) U p_2))$ is a strict sequencing formula with two temporal operators. Note that $U$ denotes the strict until operator in the strict sequencing formula. For constructing the automaton, we have used the tool scheck [73]. The coverage formulas typically result in the largest automaton and the sequencing formulas in the smallest automaton. As an example, $\phi_{\text{cov}}^{7}$ results in an automaton with 128 states and 2186 transitions, $\phi_{\text{seq}}^{7}$ results in an automaton with 8 states and 35 transitions and $\phi_{\text{st seq}}^{7}$ results in an automaton with 29 states and 569 transitions.

A comparison of performance obtained by using and ignoring geometry in the construction of the abstraction is shown in Figure 6. The results indicate that the performance of the approach is affected by not only the length of the formula but also the type of temporal operators in the formula. Problems involving coverage formulas ($\phi_{\text{cov}}^{n_{\text{op}}}$) with about 6-7 temporal operators take the longest to solve.

Even though the discrete abstraction can be constructed by ignoring the geometry of the specifications, there is a significant improvement in performance if the abstraction is constructed using the geometry of the specifications. For the case of sequencing ($\phi_{\text{seq}}^{n_{\text{op}}}$) and strict sequencing formulas ($\phi_{\text{st seq}}^{n_{\text{op}}}$), geometry-based abstractions result in speedup of up to 3-4 times. For the challenging case of coverage specifications ($\phi_{\text{cov}}^{n_{\text{op}}}$) with many temporal operators, the speedup obtained is about $50\%$. The results indicate that geometry-based abstractions certainly result in significant improvement in computational efficiency. The relatively modest improvement in performance for coverage formulas also motivates the need for an efficient high-level search. Although the lazy high-level search is not used in the experimental results shown in Figure 6, further experiments involving robot models with hybrid dynamics indicate that for best overall performance, geometry-using abstractions should be used together with lazy high-level search [29]. A brief discussion of related work is provided in Section II-D. We wish to remark here that the standard deviation in all our experimental results is high (similar magnitude as the means). We are currently investigating the underlying cause for the high variance in performance, and plan to address it as part of ongoing research.

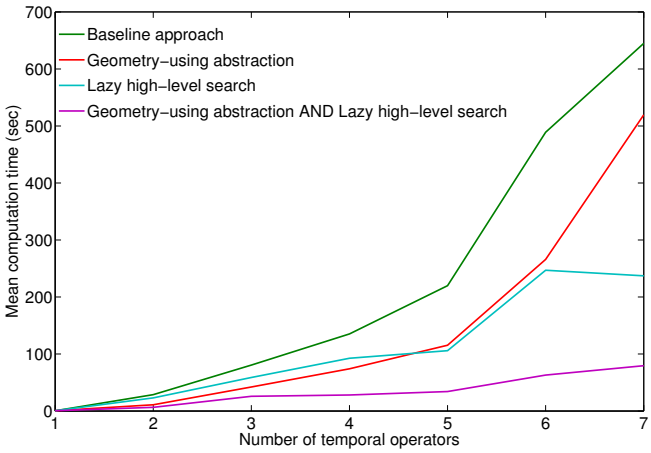


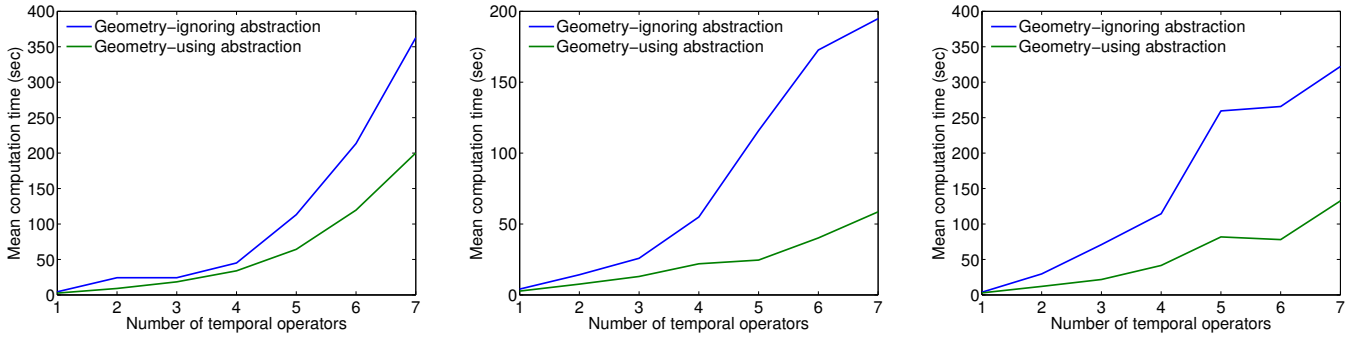

Fig. 7: A combination of geometry-using abstractions and lazy high-level search can result in speedups of close to 10 times when compared to the baseline approach. The baseline approach is the one that uses geometry-ignoring abstraction and reinitialized high-level search. See also Section II-D and [29].

### *D. Other Instantiations of the Multi-layered Framework*

Although the discussion so far has focused on motion planning problems with continuous robot dynamics and high-level goals described using LTL, this is by no means the only instantiation of the proposed multi-layered synergistic framework.

An instantiation of the framework for the case when the robot dynamics are hybrid has been proposed in [29], and the performance improvements obtained by using different combinations of geometry-using abstractions and lazy high-level search have been investigated. Computational results are shown in Figure 7 for coverage formulas. The hybrid robotic benchmark used in the experiments is as follows. The robot model has four discrete modes and three guard sets in each discrete mode. For each mode, the robot is modeled as either a second-order car, a unicycle or a differential drive with finite geometry. The workspace in each discrete mode has obstacles and the robot geometry is finite. The guards, invariants, obstacles, and the propositional sets are all polygons in the workspace. All the computation times are reported in seconds as average over 40 test runs. The maximum time allocated for each simulation $t_{max}$ was set to 1200 seconds. For the test runs when no solution was found, we have used the upper bound on simulation time $t_{max}$. The geometry-ignoring abstraction has 35 states while the geometry-using abstraction has 706 states. As in the case of continuous robot models, the standard deviation in our experimental results is high.

The baseline approach is the one that uses geometry-ignoring abstraction and reinitialized high-level search (see also Section II-B). The experimental results of Figure 7 and more generally [29] indicate that the best performance is obtained when geometry-using abstractions are used in combination with lazy high-level search. In fact, for the coverage formula with 7 temporal operators, the resulting speedup is close to 10 times. For further details the readers are referred to [29].




(a) Coverage (b) Sequencing (c) Strict Sequencing

Fig. 6: Performance comparison of the benefits of using geometry to construct the discrete abstraction. The workspace used in the experiments is shown in Figure 5. The geometry-ignoring abstraction had 8 states while the geometry-using abstraction had 618 states. All the computation times are reported in seconds as average over 40 test runs. The maximum time allocated for each simulation $t_{max}$ was set to 900 seconds. For the test runs when no solution was found, we have used the upper bound on simulation time $t_{max}$. See also Section II-C and [28].

The proposed ideas can also be used directly for safety analysis of hybrid and robotic system involving LTL specifications. In fact, the first instantiation of the framework in [27] has been used for safety falsification of hybrid systems with LTL specifications. Falsification studies the following problem: *"Can a feasible trajectory for the system be constructed such that the trajectory violates a given safety condition?"* Such a trajectory is called a counterexample trajectory. The safety conditions assert that nothing bad happens to the system. Falsification is often the focus of model checking in industrial applications [74]. Experience with industrial formal verification has shown that the ability to exhibit counterexample trajectories is often the most useful part of formal verification, since it provides designers with scenarios that they did not consider possible [75].

An instantiation of the multi-layered synergistic framework has been used for reachability-based falsification problems involving hybrid systems with a large number of discrete modes (up to a million) and nonlinear dynamics in each discrete mode in [58]. The approach significantly outperforms single-layered approaches (by orders of magnitude) .

The work on reachability-based falsification for hybrid systems in [58] has inspired an instantiation of the multi-layered synergistic planning paradigm for solving traditional motion planning problems in [59]. The approach treats the motion planning problem as a search problem in a hybrid space (consisting of both continuous and discrete components) instead of a purely continuous space. The continuous search is conducted in the state-space of the system. The discrete search is conducted over an abstraction that is constructed by decomposing the workspace of the robot using the geometry of the environment. The approach is shown to outperform existing single-layered sampling-based planners by up to two orders of magnitude.

## III. Discussion

In this paper, an approach for solving motion planning problems involving mobile robots with nonlinear hybrid dynamics and finite geometry, obstacles in the workspace, and high-level temporal goals has been described. The approach uses an instantiation of the multi-layered synergistic framework proposed in [27] while addressing two key issues: the construction of the discrete abstraction and its efficient exploration in the high-level search layer. Based on experimental results, the use of geometry for the construction of the discrete abstraction is advocated [28], [29]. For best performance, it is also recommended to explore such abstractions using efficient high-level search techniques.

As we discussed in Section II-C, an important area of concern in our current work is the high variance in performance results. As part of ongoing research, we are investigating this issue. There are several possible directions for future research. First, the specifications in [28], [29] are described over the workspace of the robot. Extensions of the proposed ideas to problem instances involving specifications in the state-space of the robotic system should be investigated. Second, as part of our current work we have advocated the use of geometry-based abstractions. Further significant improvements in performance could be obtained by taking into account the dynamics of the system. It may be possible to improve the performance of the approach even further by using pre-designed controllers in the low-level search layer. A third direction of research is on extending the framework to motion planning with optimal cost trajectories. Extension of the framework to a broader class of hybrid and robotic systems with a larger number of discrete modes remains an area for future research as well.

## IV. Acknowledgments

We would like to thank Y. Lustig and E. Plaku for many useful comments and suggestions. The research leading to this work was supported in part by NSF CCF 1018798, NSF IIS 0713623, NSF DUE 0920721, U.S. ARL W911NF-09-1-0383, NSF EIA-0216467, and a partnership between Rice University, Sun Microsystems, and Sigma Solutions.

## References

[1] H. Choset, K. M Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. 2005.

[2] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 1st edition, 2006.

[3] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189 – 208, 1971.

[4] Michael Gelfond and Vladimir Lifschitz. Action languages. *Electronic Transactions on AI*, 3, 1998.

[5] M. Ghallab, C. K. Isi, S. Penberthy, D. E. Smith, Y. Sun, and D. Weld. PDDL - the planning domain definition language, 1997.

[6] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers, 2004.

[7] K. Hauser and J.-C. Latombe. Integrating task and PRM motion plannning. In *International Conference on Automated Planning and Scheduling*, 2009. Workshop on Bridging the Gap between Task and Motion Planning.

[8] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation*, 2010. Workshop on Mobile Manipulation.

[9] E. Plaku and G. D. Hager. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *IEEE International Conference on Robotics and Automation*, pages 5002–5008, Anchorage, AK, 2010.

[10] J. Wolfe, B. Marthi, and S. J. Russell. Combined task and motion planning for mobile manipulation. In R. I. Brafman, H. Geffner, J. Hoffmann, and H. A. Kautz, editors, *International Conference on Automated Planning and Scheduling*, pages 254–258. AAAI, 2010.

[11] I. A. Şucan and L. E. Kavraki. Mobile manipulation: Encoding motion planning options using task motion multigraphs. In *IEEE International Conference on Robotics and Automation*, 2011.

[12] R.W. Brockett. On the computer control of movement. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 534 –540, April 1988.

[13] V. Manikonda, P. S. Krishnaprasad, and J. Hendler. A motion description language and a hybrid architecture for motion planning with nonholonomic robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2021 –2028, May 1995.

[14] M. Egerstedt, T. Murphey, and J. Ludwig. Motion programs for puppet choreography and control. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 190–202. Springer-Verlag, Berlin, Heidelberg, 2007.

[15] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005.

[16] Giuseppe De Giacomo and Moshe Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In Susanne Biundo and Maria Fox, editors, *European Conference on Planning*, volume 1809 of *LNCS*, pages 226–238. Springer, 1999.

[17] M. Antoniotti and B. Mishra. Discrete event models + temporal logic = supervisory controller:automatic synthesis of locomotion controllers. *IEEE International Conference on Robotics and Automation*, 2:1441 – 1446, 1995.

[18] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. In *IEEE Conference on Decision and Control*, volume 1, pages 153–158 Vol.1, 2004.

[19] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion: State of the art and grand challenges. *IEEE Robotics and Automation Magazine*, 14(1):61–70, March 2007.

[20] D. C. Conner, H. Kress-Gazit, H. Choset, A. Rizzi, and G. J. Pappas. Valet parking without a valet. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 572–577, San Diego, CA, October 2007. IEEE.

[21] Hadas Kress-Gazit, David C. Conner, Howie Choset, Alfred A. Rizzi, and George J. Pappas. Courteous cars: Decentralized multi-agent traffic coordination. *Robotics and Automation Magazine*, 15(1):30–38, 2008.

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

[23] S. Karaman and E. Frazzoli. Complex mission optimization for multiple-UAVs using linear temporal logic. In *American Control Conference*, pages 2003–2009, 2008.

[24] S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic $\mu$-calculus specifications. In *IEEE Conference on Decision and Control*, pages 2222–2229, 2009.

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

[26] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning for dynamical systems. In *IEEE Conference on Decision and Control*, pages 5997–6004, 2009.

[27] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Falsification of LTL safety properties in hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 5505 of *LNCS*, pages 368–382. Springer-Verlag, 2009.

[28] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *IEEE International Conference on Robotics and Automation*, pages 2689–2696, 2010.

[29] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Motion planning with hybrid dynamics and temporal goals. In *IEEE Conference on Decision and Control*, pages 1108–1115, 2010.

[30] M. Kloetzer and C. Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, 23:320–330, 2007.

[31] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *the VIII Banff Higher order workshop conference on Logics for concurrency : structure versus automata*, pages 238–266. Springer-Verlag New York, Inc., 1996.

[32] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. 2000.

[33] A. Pnueli. The temporal logic of programs. In *SFCS '77*, pages 46–57. IEEE Computer Society, 1977.

[34] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

[35] D. Kozen. Results on the propositional $\mu$-calculus. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 348–359, London, UK, 1982. Springer-Verlag.

[36] J. Davoren, V. Coulthard, N. Markey, and T. Moor. Non-deterministic temporal logics for general flow systems. In R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 280–295. Springer, 2004.

[37] J. M. Davoren and P. Tabuada. On simulations and bisimulations of general flow systems. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 529–542. Springer, 2007.

[38] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, April 1994.

[39] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *LNCS*, pages 209–229. Springer Berlin / Heidelberg, 1993.

[40] G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13:1–21, 2000.

[41] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971 –984, July 2000.

[42] P. Tabuada and G. J. Pappas. Hybrid abstractions that preserve timed languages. In M. D. Benedetto and A. L. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, pages 501–514. Springer, 2001.

[43] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical and control systems. *Electr. Notes Theor. Comput. Sci.*, 69, 2002.

[44] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theor. Comput. Sci.*, 342(2-3):229–261, 2005.

[45] C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21, 2005.

[46] P. Tabuada. Approximate simulation relations and finite abstractions of quantized control systems. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 529–542. Springer, 2007.

[47] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, 52(5):782–798, May 2007.

[48] A. Girard. Approximately bisimilar finite abstractions of stable linear systems. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 231–244. Springer, 2007.

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

[50] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[51] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computation and Control*, volume 4981 of *LNCS*, pages 201–214. Springer, 2008.

[52] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 142–156. Springer, 2004.

[53] J. Kim, J. M. Esposito, and V. Kumar. An RRT-based algorithm for testing and validating multi-robot controllers. In *RSS'05*, 2005.

[54] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control and verification of hybrid systems. *Control Theory and Applications, IEEE Proceedings* -, 153(5):575–590, 2006.

[55] A. Bhatia and E. Frazzoli. Sampling-based resolution-complete algorithms for safety falsification of linear systems. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computation and Control*, volume 4981 of *LNCS*, pages 606–609. Springer, 2008.

[56] T. Dang, A. Donze, O. Maler, and N. Shalev. Sensitive state-space exploration. In *IEEE CDC'08*, pages 4049–4054. IEEE, Dec. 2008.

[57] T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.

[58] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Hybrid systems: from verification to falsification by combining motion planning and discrete search. *Formal Methods in System Design*, 34(2):157–182, 2009.

[59] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics*, 26(3):469–482, 2010.

[60] P. Cheng and V. Kumar. Sampling-based falsification and verification of controllers for continuous dynamic systems. In *Algorithmic Foundations of Robotics VII*, 2006.

[61] R. Alami, J. P. Laumond, and T. Siméon. Two manipulation planning algorithms. In *Algorithmic Foundations of Robotics*, pages 109–125, 1995.

[62] C. L. Nielsen and L. E. Kavraki. A two level fuzzy PRM for manipulation planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1716–1721, 2000.

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

[64] O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19:291 – 314, 2001.

[65] E. Y. Chang, Z. Manna, and A. Pnueli. Characterization of temporal property classes. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, pages 474–486, London, UK, 1992. Springer-Verlag.

[66] R. Armoni, S. Egorov, R. Fraer, D. Korchemny, and M. Y. Vardi. Efficient LTL compilation for SAT-based model checking. In *Proceedings of the IEEE/ACM International Conference on Computer-aided design*, pages 877–884, Washington, DC, USA, 2005. IEEE Computer Society.

[67] S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research (to appear)*, 23:673–692, 2004.

[68] J. Matousek. *Geometric Discrepancy*. Springer-Verlag, Berlin, 1999.

[69] S. Tezuka. Quasi-Monte Carlo: The discrepancy between theory and practice. In K.-T. Fang, F. J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 124–140. Springer-Verlag, Berlin, 2002.

[70] S. R. Lindemann, A. Yershova, and S. M. LaValle. Incremental grid sampling strategies in robotics. In M. Erdmann, D. Hsu, M. Overmars, and A. F. van der Stappen, editors, *Algorithmic Foundations of Robotics, VI*. Springer-Verlag, Berlin, 2005.

[71] J. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering*, volume 1148 of *LNCS*, chapter 23, pages 203 – 222. Springer-Verlag, 1996.

[72] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971 – 984, 2000.

[73] T. Latvala. Efficient model checking of safety properties. In *Model Checking Software*, volume 2648 of *LNCS*, pages 74–88. Springer, 2003.

[74] F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and Moshe Y. Vardi. Benefits of bounded model checking at an industrial setting. In *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification*, pages 436–453, London, UK, 2001. Springer-Verlag.

[75] E. M. Clarke and H. Veith. Counterexamples revisited: Principles, algorithms, applications. In *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 208–224. Springer, 2003.