Hierarchical decomposition of LTL synthesis problem for nonlinear control systems

Pierre-Jean Meyer, Dimos V. Dimarogonas

***Abstract*—This paper deals with the control synthesis problem for a continuous nonlinear dynamical system under a Linear Temporal Logic (LTL) formula. The proposed solution is a top-down hierarchical decomposition of the control problem involving three abstraction layers of the problem, iteratively solved from the coarsest to the finest. The LTL planning is first solved on a small transition system only describing the regions of interest involved in the LTL formula. For each pair of consecutive regions of interest in the resulting accepting path satisfying the LTL formula, a discrete plan is then constructed in the partitioned workspace to connect these two regions while avoiding unsafe regions. Finally, an abstraction refinement approach is applied to synthesize a controller for the dynamical system to follow each discrete plan. The second main contribution, used in the third abstraction layer, is a new monotonicity-based method to over-approximate the finite-time reachable set of any continuously differentiable system. The proposed framework is demonstrated in simulation for a motion planning problem of a mobile robot modeled as a disturbed unicycle.**

***Index Terms*—Hierarchical decomposition, LTL planning, abstraction-based synthesis, mixed-monotone systems, reachability analysis.**

I. Introduction

Control synthesis and planning for continuous dynamical systems under high-level specifications, such as Linear Temporal Logic (LTL) formulas [2], usually cannot be solved directly on the continuous dynamics. The classical solutions to this problem thus rely on a two-step approach, where we first create a finite abstraction (the *abstract* or *symbolic model*) of the continuous dynamical system (the *concrete model*), and then leverage formal methods from the field of computer science to synthesize a controller for the abstraction to satisfy the high-level specifications. Provided that the abstraction was created to obtain some behavioral relationship (such as alternating simulation [25] or feedback refinement relation [21]) between the concrete and abstract models, the controller obtained on the abstraction can then be concretized into a controller for the concrete model to satisfy the desired specifications.

This topic recently received significant interest resulting in various abstraction methods such as designing local feedback controllers between any two neighboring cells of a state space partition to guarantee the creation of a deterministic abstraction [9], [4], considering infinite-time reachability analysis of neighboring cells [20], [29], or fixed and finite-time reachability analysis [5], [21], which we consider in this paper. While the combined results of all these approaches cover a wide range of dynamical systems and control objectives, when taken separately most of these approaches (as well as others in the literature) are restricted to particular classes of systems (e.g. multi-affine [9], mixed-monotone [5]) and subsets of LTL formulas (e.g. reach-avoid-stay [20], [29], co-safe LTL [9]). However, providing a framework capable of solving the synthesis problem for any dynamical systems under general LTL formulas remains a challenging problem.

This can particularly be observed when considering the main software toolboxes in the literature aimed at addressing such high-level control problems on dynamical systems, which can be split in two categories. On one side are tools such as TuLiP [28], conPAS2 [26] and LTLMoP [8] which can handle general LTL specifications (conPAS2) or the large subset of GR(1) formulas (TuLiP, LTLMoP) but are restricted to simpler dynamical systems such as fully actuated (LTLMoP) and piecewise affine models (TuLiP, conPAS2). On the other side, switched or nonlinear dynamical systems are handled by tools such as PESSOA [14], CoSyMa [19] and SCOTS [22], but only for combination of safety and reachability specifications.

To overcome these limitations, in this paper we propose a 3-layer hierarchical decomposition of the control problem aimed at addressing general control synthesis for nonlinear dynamical systems under LTL specifications. As opposed to the 2-step bottom-up symbolic control approach presented above which starts by computing an abstraction of the dynamical system before synthesizing a controller on this abstraction, we rather take inspiration from top-down hierarchical decomposition in the field of artificial intelligence [23]. In this approach, we have several granularities of abstraction of the control problem and we first solve the problem on the most abstract layer, then iteratively refine this result by going down to a more detailed layer whose subproblem consists of realizing the solution of the above layer. Given an initial partition of the state space (possibly containing unsafe regions) and an LTL formula defined over a set of regions of interest, each corresponding to a single cell of this partition, the proposed hierarchical decomposition proceeds to the following three steps.

1) Solve the LTL planning problem on a finite transition system that only represents the regions of interest, and obtain a resulting infinite sequence of regions to visit.
2) Find a discrete plan in the partitioned state space connecting each pair of consecutive regions in this sequence

This work was supported by the H2020 ERC Starting Grant BUCOPHSYS, the EU H2020 AEROWORKS project, the EU H2020 Co4Robots project, the Swedish Foundation for Strategic Research, the Swedish Research Council and the KAW Foundation.

P.-J. Meyer is with the Department of Electrical Engineering and Computer Sciences at University of California, Berkeley, CA 94720-1770, USA (email: pjmeyer@berkeley.edu).

D. V. Dimarogonas is with the Department of Automatic Control at KTH Royal Institute of Technology, 10044 Stockholm, Sweden (email: dimos@kth.se).

while avoiding unsafe regions.
3) Synthesize a controller for the dynamical system to follow each discrete plan in a sampled-time manner.

The main contribution of this paper is the 3-layer hierarchical structure allowing to tackle control problems for nonlinear systems under general LTL specifications, without restricting each layer to specific tools. The first two layers can easily be solved by classical methods for LTL model checking on finite systems [2] and graph searches [6], respectively. For the third layer, we consider the recent abstraction refinement approach in [17] which is applicable to any system associated with a method to over-approximate its finite-time reachable sets. The second contribution of this paper is thus the definition of a new reachability analysis method relying on the monotonicity property [24] but applicable to any continuously differentiable system without any monotonicity assumption.

This paper is structured as follows. In Section II, our main contributions are compared to existing work in the literature. Section III formulates the control problem and introduces the 3-layer hierarchical decomposition of LTL control problems on nonlinear dynamical systems. The new reachability analysis for any continuously differentiable system is presented in Section IV alongside an overview of the abstraction refinement algorithm in which it is used. Finally, Section V presents a numerical implementation of the proposed approach to a motion planning problem for a unicycle robot with disturbances.

II. Related work

The abstraction method in the third step uses a finite-time reachability analysis of the dynamical system to compute the non-deterministic transitions of the abstraction. In this paper, we propose a new reachability analysis approach relying on a monotonicity property [24] but applicable to dynamical systems which are not monotone. A first abstraction-based control approach relying on the monotonicity property was introduced in [18] for monotone systems and then extended in [5] for the larger class of mixed-monotone systems. An extension of the sufficient conditions for mixed-monotonicity from [5] to any continuously differentiable system was recently introduced in [30] for another type of abstraction [29]. Starting from systems satisfying the mild conditions in [30], our contribution is to define a new finite-time reachability analysis approach inspired by, yet strictly more general than, the one in [5], thus opening the use of monotonicity-based abstraction approaches to any continuously differentiable system.

To further compare the proposed approach with the previously mentioned works, we can first note that the introduction of the intermediate layer in our hierarchical decomposition allows the consideration of more general control objectives than PESSOA [14], CoSyMa [19] or SCOTS [22] by translating a general LTL specification into a sequence of reachability problems. In addition, the new contribution on monotonicity-based reachability analysis for any continuously differentiable nonlinear system opens this approach to a much wider class of systems than those considered in TuLiP [28], conPAS2 [26] and LTLMoP [8]. Regarding other tools also covering nonlinear systems, it should be noted that PESSOA [14] does not natively handle those systems and requires the user to manually provide a Matlab function computing an over-approximation of the reachable set, while an over-approximation method is included by default in our approach. The consideration of nonlinear systems in CoSyMa [19] is based on an incremental stability assumption, which is relaxed in this paper. Finally, SCOTS [22] simply uses a different over-approximation method based on Lipschitz arguments to create a growth bound on the nonlinear system's reachable set. Although we compare our contributions to existing tools as they are good indicators of the generality of results that can be covered in this field, this paper mainly focuses on providing the initial theoretical results and structure for a possible future development of a general and fully reusable tool.

Among other relevant work, 2-layer top-down structures are proposed in [13], [27] for fully actuated and piecewise affine systems respectively, where the first layer of the present paper is skipped to look directly for a discrete plan satisfying the LTL formula in the partitioned environment, then a continuous controller is designed to realize this discrete plan. Similarly, the 3-layer top-down decomposition mentioned in [3] also skips the first layer but splits the second one in two components: first finding all discrete plans satisfying the LTL formula in the partitioned environment without obstacle, then picking an optimal plan based on obstacle avoidance. The third layer in [3] uses a deterministic abstraction approach similar to [9] to implement this discrete plan on a robot modeled by an affine system. Another 3-step hierarchical decomposition of an LTL control problem is presented in [7], but for a bottom-up decomposition whose steps are significantly different from our approach as they consist in first abstracting the dynamical system into a fully actuated model (similarly to our second step), then robustifying the specification to compensate for the mismatches with the initial system and finally solving the new LTL problem on the robust specification.

As a summary, the main theoretical contributions of the proposed framework compared to existing tools for abstraction-based synthesis and other multi-layer approaches is the ability to handle general LTL control problems (layer 1) on any nonlinear system (new result on reachability analysis combined with abstraction refinement in layer 3) at a reduced computational cost (decomposing the LTL planning in two steps with layer 2). Such general problems cannot be handled by any of the approaches mentioned above.

III. Hierarchical decomposition of LTL problems

Let $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}_0^+$ and $\mathbb{R}_0^-$ be the sets of positive integers, reals, non-negative reals and non-positive reals, respectively. For $a, b \in \mathbb{R}^n$, the interval $[a, b] \subseteq \mathbb{R}^n$ is defined as $[a, b] = \{z \in \mathbb{R}^n \mid a \leq z \leq b\}$ using componentwise inequalities.

A. System description

We consider a class of continuous-time nonlinear control systems subject to disturbances and modeled by:

$$\dot{z} = f(z, u, d), \tag{1}$$

where $z \in \mathcal{Z} \subseteq \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^p$ and $d \in \mathcal{D} \subseteq \mathbb{R}^q$ are the state, bounded control input and bounded disturbance input, respectively. Throughout this paper, the vector field f of (1) is assumed to be continuously differentiable. We denote as $\Phi(t, z, \mathbf{u}, \mathbf{d})$ the state (assumed to exist and be unique) reached by (1) at time $t \in \mathbb{R}_0^+$ from initial state $z \in \mathcal{Z}$, under the piecewise continuous control $\mathbf{u} : \mathbb{R}_0^+ \to \mathcal{U}$ and disturbance functions $\mathbf{d} : \mathbb{R}_0^+ \to \mathcal{D}$. We use $\Phi(t, z, u, d)$ with $u \in \mathcal{U}$ and $d \in \mathcal{D}$ in the case of constant input functions $\mathbf{u} : \mathbb{R}_0^+ \to \{u\}$ and $\mathbf{d} : \mathbb{R}_0^+ \to \{d\}$.

Given a sampling period $\tau \in \mathbb{R}_0^+$, a sampled version of system (1) can be described as a non-deterministic (due to the disturbance) infinite transition system $S_\tau = (X_\tau, U_\tau, \delta_\tau)$ where: $X_\tau = \mathcal{Z}$ is the set of states; $U_\tau = \mathcal{U}$ is the set of control inputs; the transition relation $\delta_\tau : X_\tau \times U_\tau \to X_\tau$ is such that $z' \in \delta_\tau(z, u)$ if there exists a disturbance $\mathbf{d} : [0, \tau] \to \mathcal{D}$ such that $z' = \Phi(\tau, z, u, \mathbf{d})$, i.e. z' can be reached from z exactly in time τ by applying the constant control u on $[0, \tau]$. While, to the best of our knowledge, there exists very few results involving the choice of the sampling period for abstraction-based approaches [4], some guidelines are provided in Section V-B3 for a unicycle model and in [15] for systems with additive control input ($\dot{z} = f(z, d) + u$).

B. Hierarchical decomposition of an LTL control problem

We consider a high-level control problem on the sampled-time system S_τ evolving in the workspace $X_\tau = \mathcal{Z} \subseteq \mathbb{R}^n$ associated to a uniform partition $\mathcal{P} \subseteq 2^{\mathcal{Z}}$ into intervals (for compatibility with the reachability analysis introduced in Section IV-A). The description of this workspace also includes a set $Obs \subseteq \mathcal{P}$ of unsafe regions (referred to as *obstacles* in this section) and a set $\Pi \subseteq \mathcal{P} \backslash Obs$ of *regions of interest*. The control specification is described by a Linear Temporal Logic (LTL) formula φ defined over the set of regions of interest Π. The reader is referred to [2] for an introduction on the LTL framework. We thus aim at solving the following problem.

Problem 1. *Find a controller $C : X_\tau \to U_\tau$ such that the closed-loop sampled system S_τ with transitions $\delta_\tau(z, C(z))$ satisfies the LTL formula φ while avoiding the obstacles $Obs \subseteq \mathcal{P}$.*

To solve Problem 1, we propose a hierarchical control structure involving three different abstraction layers of the dynamical system and its environment, each of which successively addresses one aspect of the control problem as sketched in Figure 1. The evolution of the control objectives (highlighted in red in Figure 1) is obtained through the following three steps, each applied on a different abstraction layer (in blue).

1) LTL planning: We first solve the LTL planning problem on a transition system $S_\Pi = (\Pi, \delta_\Pi)$ whose states are the regions of interest in the finite set $\Pi \subseteq \mathcal{P} \backslash Obs$ and its transition relation is $\delta_\Pi \subseteq \Pi^2$. The user can freely choose between defining $\delta_\Pi = \Pi^2$ (i.e. each region of interest can reach any other) to disregard the workspace geometry or manually creating $\delta_\Pi \subsetneq \Pi^2$ to consider physical constraints. Any standard LTL model checker (see e.g. [11]) can then be used with S_Π and φ to obtain an *accepting path* in S_Π satisfying the LTL formula φ. Let $\bar{\pi} = \pi^0 \pi^1 \pi^2 \dots$ denote this accepting path (if it exists) as a (possibly infinite) sequence of regions of interest in Π.

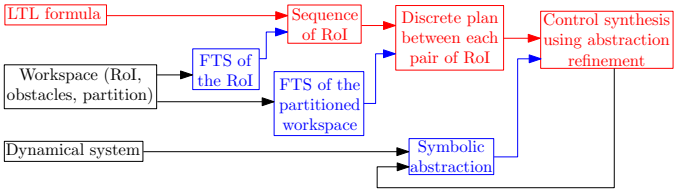

Fig. 1. Hierarchical structure of the problem solution (RoI = *Regions of Interest*, FTS = *Finite Transition System*).

2) Discrete plan and obstacle avoidance: Next we focus on obtaining a discrete plan in the workspace partition $\mathcal{P}$ realizing the accepting path $\bar{\pi}$. Let $\mathcal{N}(\sigma) \subseteq \mathcal{P}$ be the set of neighbors of a cell $\sigma \in \mathcal{P}$ (i.e. the partition elements having a common facet with σ), with the assumption that $\sigma \in \mathcal{N}(\sigma)$. The second abstraction layer thus describes possible motion in the physical environment while disregarding the system dynamics and is represented by the transition system $S_\mathcal{P} = (\mathcal{P}, \delta_\mathcal{P})$ whose set of states (or *cells*) is $\mathcal{P}$ and its transition relation $\delta_\mathcal{P} \subseteq \mathcal{P}^2$ is such that $(\sigma, \sigma') \in \delta_\mathcal{P}$ for all $\sigma \in \mathcal{P} \backslash Obs$, $\sigma' \in \mathcal{N}(\sigma) \backslash Obs$. As a result, any behavior of $S_\mathcal{P}$ induced by the above transition relation $\delta_\mathcal{P}$ is guaranteed to satisfy the obstacle avoidance.

Then for each pair $(\pi^i, \pi^{i+1}) \in \Pi^2$ of consecutive regions of interest in the accepting path $\bar{\pi}$, we look for a plan $\Lambda^i = \sigma_0^i \sigma_1^i \dots \sigma_{r_i}^i$ in $\mathcal{P} \backslash Obs$ connecting the two cells $\sigma_0^i = \pi^i$ and $\sigma_{r_i}^i = \pi^{i+1}$. Since Π is finite, there is necessarily a finite number of such pairs appearing in an infinite accepting path $\bar{\pi}$, which means that even an LTL formula with infinitely repeating properties is translated into a finite set of finite plans. The search for the plan $\Lambda^i = \sigma_0^i \sigma_1^i \dots \sigma_{r_i}^i$ is done through classical graph search algorithms on $S_\mathcal{P}$ (see e.g. [6]), such as a Breadth-First Search or a Dijkstra algorithm to consider weighted transitions (e.g. to penalize transitions going to a cell neighboring an obstacle), which are guaranteed to find such plans Λ^i as long as π^i and π^{i+1} can be connected in $S_\mathcal{P}$.

Remark 2. *$\Pi \subseteq \mathcal{P}$ ensures that each pair (π^i, π^{i+1}) in $\bar{\pi}$ only needs one plan Λ^i as above. The more general case with $\Pi \subseteq 2^\mathcal{P}$ can be handled at a greater computational cost, since a plan in $\mathcal{P}$ needs to be found for each pair in $\pi^i \times \pi^{i+1} \subseteq \mathcal{P}^2$.*

3) Control synthesis: After applying the first two layers as above, Problem 1 can then be solved by obtaining a solution to the following problem for each plan Λ^i from Section III-B2.

Problem 3. *Given a plan $\Lambda^i = \sigma_0^i \sigma_1^i \dots \sigma_{r_i}^i$ in $\mathcal{P}$, find a controller $C^i : X_\tau \to U_\tau$ such that the closed-loop sampled system S_τ follows this plan, i.e. for any trajectory $z_0 \dots z_{r_i}$ of S_τ with $z_0 \in \sigma_0^i$ and $z_{k+1} \in \delta_\tau(z_k, C^i(z_k))$ for all $k \in \{0, \dots, r_i - 1\}$, it holds that $z_k \in \sigma_k^i$.*

As for the other layers, any tool able to solve Problem 3 can be used in this third layer. One possible example is the approach in [9] where an abstraction of S_τ is created as a finite transition system obtained by designing feedback controllers deterministically driving the system between any two neighbor cells in $\mathcal{P}$. While the obtained deterministic abstraction leads

to a straightforward solution for Problem 3, such approach is limited to the class of multi-affine systems.

For the highest generality of the proposed hierarchical structure, the suggested solution for the third layer (detailed in Section IV) is another abstraction-based approach relying on reachability analysis of system (1) and which has the advantage of being applicable to any continuously differentiable system. Unlike [9] above, the drawback of such approach is that the obtained abstraction is a non-deterministic transition system (see e.g. [5]) which is thus unlikely to deterministically follow the plan Λ^i in Problem 3. Instead of manually looking for abstractions of finer granularity on which the control problem is feasible, we consider an abstraction refinement approach as in [17] where an abstraction is created on the initial coarse partition $\mathcal{P}$ and then iteratively refined by re-partitioning the cells where the control synthesis fails.

4) General comments: The main contribution of this section is the new 3-layer hierarchical framework to solve Problem 1 for nonlinear systems under general LTL specifications. However as mentioned above, each layer can be tackled by any existing tool able to address the corresponding subproblem, and the examples mentioned in this section are not claimed to be the optimal choices nor to be new contributions of this paper. Unlike the first two layers relying on well established tools, the abstraction refinement approach proposed for the third layer is a more recent result [17], and although its algorithm is not new, its applicability to any continuously differentiable system is the second main contribution of this paper which we thus describe in more detail in Section IV.

The proposed structure combining the first and second layers has the advantage of providing a solution to the LTL planning on $\mathcal{P}$ at a significantly lower computational cost than if this problem were to be solved in a single step, directly on $S_{\mathcal{P}}$. As will be seen in the numerical example of Section V, the main computational bottleneck is on the control synthesis in the third layer, which is the trade-off for the generality offered by the proposed abstraction refinement approach. As mentioned above, the computational complexity of layer 3 can be reduced at the cost of generality by using more efficient tools which are only applicable to smaller classes of systems.

Finally, we provide some guidelines on how to handle infeasibility of each layer's subproblem. In layer 1, if the LTL specification φ is infeasible, *specification revision* methods [12] can be considered to find a new specification satisfiable by S_Π and as close to φ as possible. In layer 2, if regions of interest $\pi^i, \pi^j \in \Pi$ cannot be connected in $\mathcal{P}\backslash Obs$, the first abstraction layer S_Π needs to be updated with the physical constraints: $(\pi^i, \pi^j) \notin \delta_\Pi$ and $(\pi^j, \pi^i) \notin \delta_\Pi$. In layer 3, if we fail to synthesize a controller within reasonable refinement iterations, the abstraction refinement algorithm can be combined with the *plan revision* approach in [15] to look for alternative plans Λ^i.

IV. Generalized abstraction refinement

The abstraction refinement approach considered in the third layer and initiated in [17] is applicable to any system (1) whose finite-time reachable sets can be efficiently over-approximated. In Section IV-A, we thus introduce the second main contribution of this paper: a new reachability analysis approach applicable to any continuously differentiable system (1). For self-containment of the paper, Section IV-B then provides an overview of the abstraction refinement algorithm from [17].

A. General monotonicity-based reachability analysis

Monotone systems are systems whose trajectories preserve some partial orders as below. A formal definition of a partial order is omitted in this paper but can be found in [1].

Definition 4. *System (1) is monotone with respect to partial orders $\preceq_z$, $\preceq_u$ and $\preceq_d$ on the state, control and disturbance inputs respectively, if for all time $t \in \mathbb{R}_0^+$, initial states $z, z' \in \mathcal{Z}$, control functions $\mathbf{u}, \mathbf{u}' : [0,t] \to \mathcal{U}$ and disturbance functions $\mathbf{d}, \mathbf{d}' : [0,t] \to \mathcal{D}$ we have: $z \preceq_z z', \mathbf{u} \preceq_u \mathbf{u}', \mathbf{d} \preceq_d \mathbf{d}' \Rightarrow \Phi(t, z, \mathbf{u}, \mathbf{d}) \preceq_z \Phi(t, z', \mathbf{u}', \mathbf{d}')$.*

In this section, we provide a new reachability analysis approach relying on the monotonicity property but without any monotonicity assumption on (1). The recent results in [30] extend the sufficient conditions for mixed-monotonicity (see e.g. [5]) to any continuous-time system whose Jacobian matrices are bounded over the considered sets of states and inputs. Our new contribution in this section starts from systems satisfying the very mild sufficient conditions in [30] and defines a new reachability analysis approach which is inspired by but is strictly more general than the one in [5].

In what follows, several steps need to be identically applied to all three variables $z \in \mathcal{Z} \subseteq \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^p$ and $d \in \mathcal{D} \subseteq \mathbb{R}^q$. When this is the case, we will use generic notations with variable $c \in \{z, u, d\}$ and dimension $m \in \{n, p, q\}$ such that $c \in \mathbb{R}^m$. We first denote as a_{ij}^c and b_{ij}^c the bounds of the partial derivatives of the vector field f as follows: for all $z \in \mathcal{Z}$, $u \in \mathcal{U}$, $d \in \mathcal{D}$, $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$, $\frac{\partial f_i}{\partial c_j}(z, u, d) \in [a_{ij}^c, b_{ij}^c]$. The values of these bounds lead us to consider the 4 cases below, covering all possibilities for the sign of each partial derivative, as in [30]:

(C1) $a_{ij}^c \geq 0$: positive,
(C2) $a_{ij}^c \leq 0 \leq b_{ij}^c$ and $|a_{ij}^c| \leq |b_{ij}^c|$: mostly positive,
(C3) $a_{ij}^c \leq 0 \leq b_{ij}^c$ and $|a_{ij}^c| \geq |b_{ij}^c|$: mostly negative,
(C4) $b_{ij}^c \leq 0$: negative.

We then define the function g such that for all $z, z^* \in \mathcal{Z}$, $u, u^* \in \mathcal{U}$, $d, d^* \in \mathcal{D}$ and $i \in \{1, \dots, n\}$ we have

$$g_i(z, u, d, z^*, u^*, d^*) = f_i(Z_i, U_i, D_i) + \alpha_i^z(z - z^*) + \alpha_i^u(u - u^*) + \alpha_i^d(d - d^*), \quad (2)$$

where the components of $Z_i = (z_{i1} \dots z_{in})^\top$, $U_i = (u_{i1} \dots u_{ip})^\top$, $D_i = (d_{i1} \dots d_{iq})^\top$ and $\alpha_i^c = (\alpha_{i1}^c \dots \alpha_{im}^c)$ are defined according to cases (C1)-(C4) for $\frac{\partial f_i}{\partial c_j}$ as follows with $c \in \{z, u, d\}$ in all notations below:

$$c_{ij} = \begin{cases} c_j & \text{if (C1) or (C2),} \\ c_j^* & \text{if (C3) or (C4),} \end{cases} \qquad \alpha_{ij}^c = \begin{cases} -a_{ij}^c & \text{if (C2),} \\ b_{ij}^c & \text{if (C3),} \\ 0 & \text{otherwise.} \end{cases}$$

The above definition of g is a straightforward extension, to non-autonomous systems with control and disturbance inputs, of the one introduced in [30]. In what follows, we provide our main contribution on this topic which describes how

to compute an interval over-approximation of the finite-time reachable set for any continuously differentiable system (1), without needing any additional assumption. For this we define the following dynamical system evolving in $\mathcal{Z}^2$:

$$\begin{pmatrix}\dot{z}\\ \dot{z}^*\end{pmatrix} = h(z,u,d,z^*,u^*,d^*) = \begin{pmatrix}g(z,u,d,z^*,u^*,d^*)\\ g(z^*,u^*,d^*,z,u,d)\end{pmatrix} \quad (3)$$

and similarly to Φ we denote the trajectories of (3) as $\Phi_h(\cdot,z,\mathbf{u},\mathbf{d},z^*,\mathbf{u}^*,\mathbf{d}^*):\mathbb{R}_0^+\to\mathcal{Z}^2$, where bold variables are piecewise continuous input functions. Let Φ_h^1 and Φ_h^2 denote the first n and last n components of Φ_h, respectively. Then, a single successor of (3) can be used to compute an over-approximation of the finite-time reachable set of (1) as follows.

Theorem 5. *For all bounds $\underline{z},\overline{z}\in\mathbb{R}^n$, $\underline{u},\overline{u}\in\mathbb{R}^p$, $\underline{d},\overline{d}\in\mathbb{R}^q$ and for all $t\in\mathbb{R}_0^+$, $z\in[\underline{z},\overline{z}]$, $\mathbf{u}:[0,t]\to[\underline{u},\overline{u}]$ and $\mathbf{d}:[0,t]\to[\underline{d},\overline{d}]$ we have (using componentwise inequalities):*

$$\Phi_h^1(t,\underline{z},\underline{u},\underline{d},\overline{z},\overline{u},\overline{d}) \le \Phi(t,z,\mathbf{u},\mathbf{d}) \le \Phi_h^2(t,\underline{z},\underline{u},\underline{d},\overline{z},\overline{u},\overline{d}).$$

Proof. From the terms $\alpha_i^c(c-c^*)$ in (2), we can show similarly to [30] that for all variables $c\in\{z,u,d\}$ (with $c\in\mathbb{R}^m$, $m\in\{n,p,q\}$), $i\in\{1,\dots,2n\}$ and $j\in\{1,\dots,m\}$ we have

$$\frac{\partial h_i}{\partial c_j}\begin{cases}\ge 0 & \text{if } i\le n,\\ \le 0 & \text{if } i\ge n,\end{cases} \quad\text{and}\quad \frac{\partial h_i}{\partial c_j^*}\begin{cases}\le 0 & \text{if } i\le n,\\ \ge 0 & \text{if } i\ge n.\end{cases}$$

Then from [1], (3) is monotone as in Definition 4 with the partial orders $\preceq_z$, $\preceq_u$ and $\preceq_d$ on the spaces $\mathbb{R}^{2n}$, $\mathbb{R}^{2p}$ and $\mathbb{R}^{2q}$, respectively, as defined below. For all variables $c\in\{z,u,d\}$ with $c\in\mathbb{R}^m$, the partial order $\preceq_c$ is characterized by the orthant $(\mathbb{R}_0^+)^m\times(\mathbb{R}_0^-)^m$ of space $\mathbb{R}^{2m}$ as follows: for all $c^1,c^2,c^3,c^4\in\mathbb{R}^m$, $\begin{pmatrix}c^1\\ c^2\end{pmatrix}\preceq_c\begin{pmatrix}c^3\\ c^4\end{pmatrix}\Leftrightarrow\begin{cases}c^1\le c^3,\\ c^2\ge c^4,\end{cases}$ where $\le$ and $\ge$ are the componentwise inequalities on $\mathbb{R}^m$. For all $c\in[\underline{c},\overline{c}]\subseteq\mathbb{R}^m$ we thus have $\begin{pmatrix}\underline{c}\\ \overline{c}\end{pmatrix}\preceq_c\begin{pmatrix}c\\ c\end{pmatrix}\preceq_c\begin{pmatrix}\overline{c}\\ \underline{c}\end{pmatrix}$ and we can then use Definition 4 for system (3) to obtain the following over-approximation: for all $t\in\mathbb{R}_0^+$, $z\in[\underline{z},\overline{z}]$, $\mathbf{u}:[0,t]\to[\underline{u},\overline{u}]$ and $\mathbf{d}:[0,t]\to[\underline{d},\overline{d}]$ we have

$$\begin{cases}\Phi_h(t,\underline{z},\underline{u},\underline{d},\overline{z},\overline{u},\overline{d})\preceq_z\Phi_h(t,z,\mathbf{u},\mathbf{d},z,\mathbf{u},\mathbf{d}),\\ \Phi_h(t,z,\mathbf{u},\mathbf{d},z,\mathbf{u},\mathbf{d})\preceq_z\Phi_h(t,\overline{z},\overline{u},\overline{d},\underline{z},\underline{u},\underline{d}).\end{cases}$$

From (2), $g(z,u,d,z,u,d)=f(z,u,d)$ which implies $\Phi_h(t,z,\mathbf{u},\mathbf{d},z,\mathbf{u},\mathbf{d})=\begin{pmatrix}\Phi(t,z,\mathbf{u},\mathbf{d})\\ \Phi(t,z,\mathbf{u},\mathbf{d})\end{pmatrix}$. Then by symmetry of (3), we have $\Phi_h^1(t,\overline{z},\overline{u},\overline{d},\underline{z},\underline{u},\underline{d})=\Phi_h^2(t,\underline{z},\underline{u},\underline{d},\overline{z},\overline{u},\overline{d})$ finally giving the result in Theorem 5. □

Theorem 5 thus provides a method to obtain over-approximations of the finite-time reachable sets for any continuously differentiable system (1) by computing a single successor state $\Phi_h(t,\underline{z},\underline{u},\underline{d},\overline{z},\overline{u},\overline{d})$ of system (3).

B. Abstraction refinement algorithm

For each plan Λ^i obtained in the second layer (Section V-B2), we abstract the sampled system S_τ by the finite transition system $S_a^i=(X_a^i,U_a,\delta_a^i)$, where: the set of states (or *symbols*) X_a^i is a partition of the workspace $\mathcal{Z}\subseteq\mathbb{R}^n$ into intervals, i.e. any symbol $s\in X_a^i$ is also an interval $s=[\underline{s},\overline{s}]\subseteq\mathcal{Z}$ of the workspace; the set of inputs U_a is a finite subset of control values in $\mathcal{U}$; a transition $s'\in\delta_a^i(s,u)$ between symbols $s\in X_a^i$ and $s'\in X_a^i$ with input $u\in U_a$ exists if $s'\cap[\Phi_h^1(\tau,\underline{s},u,\underline{d},\overline{s},u,\overline{d}),\Phi_h^2(\tau,\underline{s},u,\underline{d},\overline{s},u,\overline{d})]\neq\emptyset$.

Remark 6. *The over-approximation of the reachable set of (1) from Theorem 5 with the constant control value u used above to define δ_a^i can also be obtained with the less conservative local bounds for the Jacobians $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial d}$ over the set of possible states only during the time period $[0,\tau]$.*

Since the transition relation δ_a^i is non-deterministic and the objective of Problem 3 is to deterministically follow the plan Λ^i in $\mathcal{P}$, we rely on an abstraction refinement approach by considering the initial coarse partition $X_a^i=\mathcal{P}$ which is then iteratively refined by re-partitioning the elements of X_a^i that are responsible for preventing the synthesis of a controller. Below, we provide an overview of the considered refinement algorithm. For further details, the reader is referred to [17].

We first define the function $P_a^i:\mathcal{P}\to 2^{X_a^i}$ such that $P_a^i(\sigma)=\{s\in X_a^i\mid s\subseteq\sigma\}$ corresponds to the projection of a cell $\sigma\in\mathcal{P}$ onto the given finer partition X_a^i. Then, Algorithm 1 is centered around the computation of *valid sets* defined as a function $V^i:\mathcal{P}\to 2^{X_a^i}$ by proceeding backwards along the plan $\Lambda^i=\sigma_0^i\sigma_1^i\dots\sigma_{r_i}^i$. The final cell $\sigma_{r_i}^i$ of Λ^i is considered as valid and the valid set function is thus initialized with $V^i(\sigma_{r_i}^i)=\{\sigma_{r_i}^i\}$. Other valid sets $V^i(\sigma_k^i)$ are then iteratively defined as the subset of symbols in σ_k^i which can be driven towards the valid set $V^i(\sigma_{k+1}^i)$ of the next cell for at least one control input in U_a: $V^i(\sigma_k^i)=\left\{s\in P_a^i(\sigma_k^i)\;\middle|\;\exists u\in U_a,\ \delta_a^i(s,u)\subseteq V^i(\sigma_{k+1}^i)\right\}$. The controller $C_a^i:X_a^i\to U_a$ is simultaneously defined by associating to each valid symbol $s\in V^i(\sigma_k^i)$ the *first* of such control values. Given a cell σ_k^i and a targeted valid set $V^i(\sigma_{k+1}^i)$, the function $\texttt{ValidSet}(\sigma_k^i,V^i(\sigma_{k+1}^i))$ in Algorithm 1 denotes the computation of $V^i(\sigma_k^i)$ and C_a^i as above.

Data: $\mathcal{P}$, $\Lambda^i=\sigma_0^i\sigma_1^i\dots\sigma_{r_i}^i\in\mathcal{P}^{r_i+1}$, $P_a^i:\mathcal{P}\to 2^{X_a^i}$.
1 **Initialization:** $X_a^i=\mathcal{P}$, $V^i(\sigma_{r_i}^i)=\{\sigma_{r_i}^i\}$
2 **for** *k from r_i-1 to 0* **do**
3 $\{V^i(\sigma_k^i),\ C_a^i\}=\texttt{ValidSet}\ (\sigma_k^i,V^i(\sigma_{k+1}^i))$
4 **while** $V^i(\sigma_k^i)=\emptyset$ *or* $V^i(\sigma_0^i)\neq P_a^i(\sigma_0^i)$ **do**
5 $j=\texttt{Pick}(k,r_i-1)$
6 **forall** $s\in P_a^i(\sigma_j^i)\backslash V^i(\sigma_j^i)$ **do**
7 $X_a^i=(X_a^i\backslash\{s\})\cup\texttt{Split}(s)$
8 **for** *l from j to k* **do**
9 $\{V^i(\sigma_l^i),\ C_a^i\}=\texttt{ValidSet}\ (\sigma_l^i,V^i(\sigma_{l+1}^i))$

Output: $\{X_a^i,\ V^i:\mathcal{P}\to 2^{X_a^i},\ C_a^i:X_a^i\to U_a\}$
Algorithm 1: Abstraction refinement algorithm.

If $V^i(\sigma_k^i)=\emptyset$ for some k (line 4), we first pick a cell σ_j^i with $j\in\{k,\dots,r_i-1\}$ to be refined (line 5), split each of its invalid symbols $s\in P_a^i(\sigma_j^i)\backslash V^i(\sigma_j^i)$ into a set of subsymbols $\texttt{Split}(s)$ and update the partition X_a^i accordingly (lines 6-7), and finally update the valid sets and controller for all cells from σ_j^i to σ_k^i whose valid sets may be expanded after this

refinement (lines 8-9). The refinement procedure is repeated until $V^i(\sigma_0^i) = P_a(\sigma_0^i)$ (line 4), i.e. when starting from any subsymbol in σ_0^i, S_a^i can be controlled to reach $\sigma_{r_i}^i$ exactly in r_i steps. For each plan Λ^i, Algorithm 1 then returns the refined partition X_a^i, the valid set function V^i and the associated controller C_a^i. The definition of both functions `Pick` and `Split` can be arbitrary but some guidelines and examples are provided in [15], [16], [17] and Section V-B3.

Lemma 7 ([17])**.** *If Algorithm 1 terminates for a plan Λ^i, then the controller $C^i : X_\tau \to U_\tau$ such that $C^i(z) = C_a^i(s)$ for all $z \in s$ solves Problem 3, i.e. the closed loop of S_τ with transitions $z' \in \delta_\tau(z, C^i(z))$ follows Λ^i when starting in σ_0^i.*

The proof of Lemma 7 in [17] is independent of the over-approximation method associated with system (1). A solution to the main LTL control problem then immediately follows.

Corollary 8. *If Algorithm 1 terminates for all plans Λ^i derived in Section III-B2, then the controller $C : \mathbb{N} \times X_\tau \to U_\tau$ defined by $C(i, z) = C^i(z)$ solves Problem 1.*

V. Application to non-holonomic motion planning

In this section, we consider a high-level motion planning problem for a mobile robot evolving in an office environment. The robot is modeled by disturbed unicycle dynamics:

$$\dot{z} = f(z, u, d) = \begin{pmatrix} v\cos(\theta) + d_1 \\ v\sin(\theta) + d_2 \\ \omega + d_3 \end{pmatrix} \tag{4}$$

where $z = (x, y, \theta) \in \mathcal{Z} \subseteq \mathbb{R}^3$ is the state (2D position and orientation), $u = (v, \omega) \in \mathcal{U} \subseteq \mathbb{R}^2$ is the control input (linear and angular velocities) and $d = (d_1, d_2, d_3) \in \mathcal{D} \subseteq \mathbb{R}^3$ is the disturbance. We further assume that the disturbance take its values in an interval $\mathcal{D} = [\underline{d}, \overline{d}]$ of $\mathbb{R}^3$. We first apply the proposed reachability analysis results to the unicycle (4) in Section V-A. Section V-B then describes the considered motion planning problem and the associated simulation results.

A. Reachability analysis

We first define function $g : \mathcal{Z} \times \mathcal{U} \times \mathcal{D} \times \mathcal{Z} \times \mathcal{U} \times \mathcal{D} \to \mathbb{R}^3$ as in (2) for the unicycle model (4). Since all partial derivatives of f_3 are non-negative ($a_{3j}^c \geq 0$ as in (C1) of Section IV-A for all $c \in \{z, u, d\}$), we thus have $\alpha_3^z = \alpha_3^d = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$, $\alpha_3^u = \begin{pmatrix} 0 & 0 \end{pmatrix}$, $Z_3 = z$, $U_3 = u$ and $D_3 = d$, leading to:

$$g_3(z, u, d, z^*, u^*, d^*) = f_3(z, u, d) = \omega + d_3. \tag{5}$$

For abstraction S_a^i in Section IV-B, the over-approximation is computed with a known and constant control value u over the time period $[0, \tau]$. The signs of $\frac{\partial f_i}{\partial v}$ for $i \in \{1, 2\}$ thus have no influence on this over-approximation since we always have $u = u^*$ in (2), implying $U_1 = U_2 = u$ and $\alpha_i^u(u - u^*) = 0$. Since for $i, j \in \{1, 2\}$ and $k \in \{1, 2, 3\}$ the partial derivatives $\frac{\partial f_i}{\partial z_j}$ and $\frac{\partial f_i}{\partial d_k}$ are non-negative, we have $\alpha_{i1}^z = \alpha_{i2}^z = 0$, $\alpha_i^d = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$ and $D_i = d$, and we thus obtain for $i \in \{1, 2\}$:

$$g_i(z, u, d, z^*, u, d^*) = f_i(Z_i, u, d) + \alpha_{i3}^z(\theta - \theta^*), \tag{6}$$

where Z_1, Z_2, α_{13}^z and α_{23}^z are defined as in Section IV-A from the values of the four bounds a_{13}^z, b_{13}^z, a_{23}^z and b_{23}^z of the remaining two partial derivatives whose signs are not constant:

$$\frac{\partial f_1}{\partial \theta} = -v\sin(\theta) \in [a_{13}^z, b_{13}^z], \quad \frac{\partial f_2}{\partial \theta} = v\cos(\theta) \in [a_{23}^z, b_{23}^z]. \tag{7}$$

Considering $\theta \in (-\pi, \pi]$ would result in too conservative global bounds $\frac{\partial f_1}{\partial \theta}, \frac{\partial f_2}{\partial \theta} \in [-v, v]$. Instead, we follow Remark 6 to find a subset of possible orientations on each sampling period $[0, \tau]$ and thus obtain tighter local bounds in (7). Given an interval of initial orientations $[\underline{\theta_0}, \overline{\theta_0}] \subseteq (-\pi, \pi]$ and a known angular velocity ω, (4) gives $\dot{\theta} \in [\omega + \underline{d}_3, \omega + \overline{d}_3]$, and thus the orientation $\theta(\tau)$ at time $\tau > 0$ is bounded as $\theta(\tau) \in [\underline{\theta_0} + \tau(\omega + \underline{d}_3), \overline{\theta_0} + \tau(\omega + \overline{d}_3)]$. Over the whole sampling period $[0, \tau]$, we obtain the following set $[\underline{\theta}, \overline{\theta}]$ of possible orientations: $\theta([0, \tau]) \in [\underline{\theta}, \overline{\theta}] = [\underline{\theta_0} + \min(0, \tau(\omega + \underline{d}_3)), \overline{\theta_0} + \max(0, \tau(\omega + \overline{d}_3))]$.

When $v \geq 0$, the bounds in (7) can thus be computed by:

$$\begin{aligned} a_{13}^z &= -v \max_{\theta \in [\underline{\theta}, \overline{\theta}]}(\sin(\theta)), & a_{23}^z &= v \min_{\theta \in [\underline{\theta}, \overline{\theta}]}(\cos(\theta)) \\ b_{13}^z &= -v \min_{\theta \in [\underline{\theta}, \overline{\theta}]}(\sin(\theta)), & b_{23}^z &= v \max_{\theta \in [\underline{\theta}, \overline{\theta}]}(\cos(\theta)) \end{aligned} \tag{8}$$

with swapped $\min$ and $\max$ operators when $v < 0$. Since $[\underline{\theta}, \overline{\theta}] \cap (-\pi, \pi] \neq \emptyset$, the extrema of the $\cos$ and $\sin$ are:

$$\min_{\theta \in [\underline{\theta}, \overline{\theta}]}(\cos(\theta)) = \begin{cases} -1 & \text{if } \{-\pi, \pi\} \cap [\underline{\theta}, \overline{\theta}] \neq \emptyset \\ \min(\cos(\underline{\theta}), \cos(\overline{\theta})) & \text{otherwise} \end{cases} \tag{9}$$

with similar equations replacing $\{-\pi, \pi\}$ by $\{-2\pi, 0, 2\pi\}$, $\{\frac{-5\pi}{2}, \frac{-\pi}{2}, \frac{3\pi}{2}\}$ and $\{\frac{-3\pi}{2}, \frac{\pi}{2}, \frac{5\pi}{2}\}$ for $\max(\cos(\theta)) = 1$, $\min(\sin(\theta)) = -1$ and $\max(\sin(\theta)) = 1$, respectively.

The bounds a_{13}^z, b_{13}^z, a_{23}^z and b_{23}^z computed from (8)-(9) for each sampling period lead to function g defined in (5)-(6) followed by the duplicated dynamical system (3) with vector field h and trajectories Φ_h as in Section IV-A. Theorem 5 with state interval $[\underline{s}, \overline{s}] \subseteq \mathcal{Z}$ and constant control $u \in \mathcal{U}$ gives the over-approximation $[\Phi_h^1(\tau, \underline{z}, u, \underline{d}, \overline{z}, u, \overline{d}), \Phi_h^2(\tau, \underline{z}, u, \underline{d}, \overline{z}, u, \overline{d})]$ as used for the abstraction refinement in Section IV-B.

B. Problem description and simulation results

We consider a high-level motion planning problem for a mobile robot evolving in a 33×20 square meters office environment. This 2D workspace is formed by four rooms and a central hallway, as sketched in Figure 2 uniformly partitioned into 20×12 cells and where the black cells represent static obstacles (walls). The four regions of interest (in blue) denoted as π_1 to π_4 correspond to the cells in which the observation tasks of each room are to be carried out.

The robot is modeled as a unicycle (4) where the state $z = (x, y, \theta)$ evolves in $\mathcal{Z} = [0, 33] \times [0, 20] \times (-\pi, \pi]$, the control inputs $u = (v, \omega)$ are picked in the discrete set $U_a = \{-0.5, -0.25, 0, 0.25, 0.5\} \times \{-0.3, -0.15, 0, 0.15, 0.3\}$ and the disturbances lie in $\mathcal{D} = [-0.05, 0.05] \times [-0.05, 0.05] \times [-0.03, 0.03]$. The initial state of (4) is taken in the cell π_1.

The control objective is expressed by the LTL formula $\varphi = \square\lozenge\pi_2 \wedge \square\lozenge\pi_4 \wedge \lozenge\pi_3 \wedge \neg\pi_3 \mathcal{U} \pi_4$, whose first two elements are a surveillance task in π_2 and π_4 (visit each infinitely often) and the last two elements mean that we also want to eventually visit π_3 but not before π_4 has been visited at least once. The

obstacle avoidance is not included in φ since it is automatically handled in the second layer of the hierarchical decomposition.

The simulation results are obtained on a laptop with a 1.7 GHz CPU and 4 GB of RAM (on Matlab for steps 2 and 3).

1) First layer - LTL planning: As in Section III-B1, we first define the finite transition system $S_\Pi = (\Pi, \delta_\Pi)$, where $\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$ is the set of regions of interest and $\delta_\Pi \subseteq \Pi^2$ represents the office structure in Figure 2 such that room 1 can only be reached through room 2: $\delta_\Pi = \Pi^2 \backslash \{(\pi_1, \pi_3), (\pi_3, \pi_1), (\pi_1, \pi_4), (\pi_4, \pi_1)\}$. The LTL planning is solved in 2 milliseconds (due to the small size of S_Π) by the model checker P-MAS-TG [10] resulting in an infinite accepting path $\bar{\pi} = \pi_1\pi_2\pi_4\pi_3(\pi_2\pi_4)^\omega$ where the prefix $\pi_1\pi_2\pi_4\pi_3$ is followed once and the suffix $\pi_2\pi_4$ is repeated infinitely often. This infinite path can then be handled by applying the next two layers to only five pairs of regions of interest: $\pi_1 - \pi_2$, $\pi_2 - \pi_4$, $\pi_4 - \pi_3$, $\pi_3 - \pi_2$ and $\pi_4 - \pi_2$.

2) Second layer - Physical environment: The transition system $S_\mathcal{P} = (\mathcal{P}, \delta_\mathcal{P})$ is then defined as in Section III-B2 for the planning and obstacle avoidance in the 2D workspace (partitioned in 20×12 *cells*) while disregarding the system dynamics (4). For each of the above 5 pairs $\pi_i - \pi_j$, one of the shortest discrete plan $\Lambda^{ij} = \sigma_0^{ij}\sigma_1^{ij}\dots\sigma_{r_{ij}}^{ij}$ in $\mathcal{P}$ connecting $\sigma_0^{ij} = \pi_i$ and $\sigma_{r_{ij}}^{ij} = \pi_j$ is obtained by applying a Breadth-First Search algorithm [6] on $S_\mathcal{P}$, with an average computation time of 46 milliseconds per pair $\pi_i - \pi_j$.

3) Third layer - Control synthesis: Taking inspiration from the guidelines in [15], we pick the sampling period $\tau = 1.2 * \max(33/20, 20/12)/0.5 = 4$ seconds approximating the minimal time to translate vertically or horizontally a cell in $\mathcal{P}$ to one of its neighbors using maximal linear velocity $u = (0.5, 0)$. The factor 1.2 multiplies this value to account for (4) not always keeping axis-aligned orientations $\theta \in \{-\pi/2, 0, \pi/2, \pi\}$ or using the maximal linear velocity.

Since the control objective Λ^{ij} for this third layer is in the 2D workspace while system (4) has a 3D state, we consider a modified definition for the valid set of a 2D cell $\sigma \in \mathcal{P}$: $V_{2D}^i(\sigma) = \{s \subseteq \sigma \mid \exists s_\theta \subseteq (-\pi, \pi], s \times s_\theta \in X_a^i \cap V^i(\sigma \times (-\pi, \pi])\}$, i.e. $V_{2D}^i(\sigma)$ contains any 2D projection s of a 3D symbol $s \times s_\theta$ belonging to both the refined partition X_a^i and the valid set of the 3D cell $\sigma \times (-\pi, \pi]$ as in Section IV-B. As a result, the abstraction refinement still works on the partition X_a^i of the 3D state space but becomes more permissive since the 2D control synthesis (using V_{2D}^i) disregards the validity of the orientation. The drawback is that after each application of the controller, we may need to rotate the robot to reach a valid orientation (which always exists by definition of V_{2D}^i).

Function `Pick` in Algorithm 1 is chosen similarly to [16] as a queue picking the oldest cell of Λ^{ij} added to the queue among the least refined ones. The orientation interval $(-\pi, \pi]$ is initially partitioned into 4 identical intervals. Function `Split`(s) in Algorithm 1 then takes a uniform partition of the 3D symbol s into 8 subsymbols (2 per dimension).

For the 5 plans Λ^{12}, Λ^{24}, Λ^{43}, Λ^{32} and Λ^{42} from layer 2, the control synthesis takes between 18 and 58 minutes (39 minutes per plan on average) with a number of refinement iterations ranging from 40 to 76 (54 per plan on average).

4) Simulation results: Figure 2 provides a visualization of the abstraction refinement results for plan Λ^{32}, where the finer black grid is the 2D projection of the refined partition X_a^i and the red area is the 2D valid sets $V_{2D}^i(\sigma)$ (not represented on cells π_2 and π_3). In this particular case, the valid sets happen to cover the whole cells: $V^i(\sigma_k^i) = P_a^i(\sigma_k^i)$ for all σ_k^i.

The disturbed unicycle (4) in closed-loop with the global controller C from Corollary 8 is then simulated from an initial state z_0 randomly picked in the 3D cell $\pi_1 \times (-\pi, \pi]$. At each time step, the robot measures its position $z = (x, y, \theta)$ and finds the corresponding 3D symbol $s_{3D} = s_{2D} \times s_\theta \in X_a^i$ such that $z \in s_{3D}$. Since the control synthesis was successful for the current 2D cell $\sigma_{2D} \in \mathcal{P}$ with $s_{2D} \subseteq \sigma_{2D}$, then by definition of $s_{2D} \in V_{2D}^i(\sigma_{2D})$ there exists a valid 3D symbol $s'_{3D} = s_{2D} \times s'_\theta \in X_a^i \cap V^i(\sigma_{2D} \times (-\pi, \pi])$. If $s_{3D} \neq s'_{3D}$, we apply a constant rotation $u = (0, \omega)$ until the system reaches a new state $z' = (x, y, \theta') \in s'_{3D}$ (assuming that we have $d_1 = d_2 = 0$ during such rotations only). Whether a rotation was done or not, we finally apply the constant control value $C_a^i(s_{2D})$ for $\tau = 4$ seconds to go to the next cell of the current plan and repeat this procedure with a new measurement of the state. The closed-loop trajectory for plan Λ^{32} is displayed in green in Figure 2. The snaps in this trajectory correspond to rotations before applying the next control, while smoother sections over several cells mean that no rotation was needed.

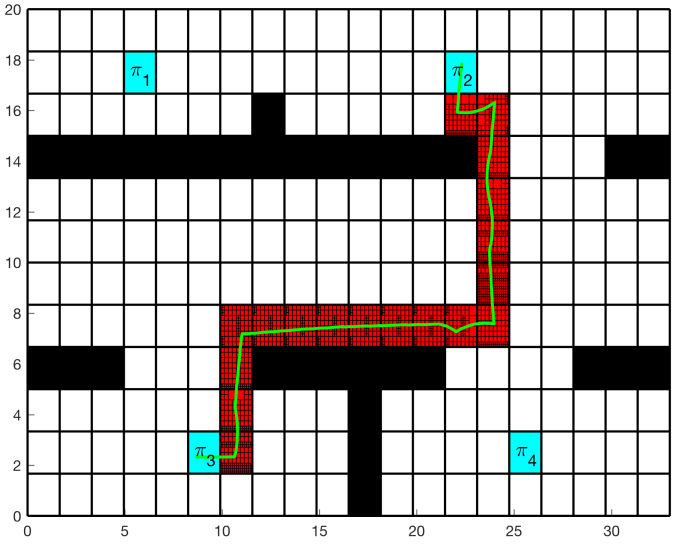

Fig. 2. Partitioned office environment with obstacles (black) and four regions of interest (blue) as the center of each room. For plan Λ^{32} from π_3 to π_2, we also display the refined partition (finer black grid), the valid symbols (red) and the closed-loop trajectory (green).

5) Final comments: As mentioned in Section III-B, the solver of each layer can be substituted by any existing tool designed to tackle the same subproblem. For the example of this section, we could also consider replacing layers 1 and 2 by conPAS2 [26] for a fully actuated system $\dot{z} = u$, or layers 2 and 3 by the unicycle example provided in PESSOA [14]. On the other hand, none of the existing tools mentioned in Section I are capable of solving the whole control problem tackled in this section, because they do not handle either nonlinear systems [28], [26], [8] or general LTL

specifications [14], [19], [22]. As a consequence, the 3-layer hierarchical framework introduced in Section III-B is strictly more general than these tools and thus cannot be compared with them on the case study of this section for the disturbed unicycle (4) under the LTL specification φ.

To highlight the novelty of the proposed approach, we can however discuss the complexity for approaching the same problem with abstraction-based methods without relying on the hierarchical decomposition or abstraction refinement. With no synthesis of an accepting path $\bar{\pi}$ for the LTL formula and of the discrete plans Λ^{ij} realizing this path (as done in layers 1 and 2), a symbolic abstraction needs to be created for the whole state space. The creation of such abstraction takes over 43 hours (compared to 3 hours in our framework) when using the finest partition granularity reached in the abstraction refinement layer to ensure that the controller obtained above could be reproduced in this setting. Further intensive computation would also result from attempting a controller synthesis with respect to the LTL specification on this abstraction containing over 98 millions state-input pairs. This last step could not even be attempted as the abstraction variable weighted over 4GB which could not be stored in Matlab.

VI. Conclusion

The first contribution of this paper is a three-layer hierarchical decomposition of a high-level control problem under a Linear Temporal Logic formula by iteratively solving finer versions of the problem: first solve the problem only on the regions of interest involved in the LTL formula, then realize the obtained sequence of regions by finding discrete plans in the partitioned workspace, finally synthesize a controller for the dynamical system to follow these plans. This framework enables the consideration of general LTL control problems for nonlinear systems and the subproblems defined at each layer can be solved through various existing tools. The second contribution is a new method to over-approximate the finite-time reachable set of any continuously differentiable system, relying on an auxiliary monotone system obtained by using Jacobian bounds to compensate the non-monotone components of the initial system. For generality of the hierarchical framework to continuously differentiable systems, an implementation for layer 3 is proposed using this new reachability analysis result within an abstraction refinement algorithm.

The main goal for future work is to strengthen the cohesion in the implementation of the three control layers to distribute this framework as a coherent and publicly available tool.

References

[1] D. Angeli and E. D. Sontag. Monotone control systems. *IEEE Transactions on Automatic Control*, 48(10):1684–1698, 2003.

[2] C. Baier, J.-P. Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

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

[4] D. Boskos and D. V. Dimarogonas. Decentralized abstractions for multi-agent systems under coupled constraints. *European Journal of Control*, 45:1–16, 2019.

[5] S. Coogan and M. Arcak. Efficient finite abstraction of mixed monotone systems. In *Hybrid Systems: Computation and Control*, pages 58–67. 2015.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.

[7] G. E. Fainekos, A. Girard, and G. J. Pappas. Hierarchical synthesis of hybrid controllers from temporal logic specifications. In *Hybrid Systems: Computation and Control*, pages 203–216. Springer, 2007.

[8] C. Finucane, G. Jing, and H. Kress-Gazit. Ltlmop: Experimenting with language, temporal logic and robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1988–1993. IEEE, 2010.

[9] E. A. Gol and C. Belta. Time-constrained temporal logic control of multi-affine systems. *Nonlinear Analysis: Hybrid Systems*, 10:21–33, 2013.

[10] M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[11] G. Holzmann. *The Spin model checker: primer and reference manual*. Addison-Wesley Professional, 2003.

[12] K. Kim, G. Fainekos, and S. Sankaranarayanan. On the minimal revision problem of specification automata. *The International Journal of Robotics Research*, 34(12):1515–1535, 2015.

[13] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics*, 25(6):1370–1381, 2009.

[14] M. Mazo Jr, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *International Conference on Computer Aided Verification*, pages 566–569. Springer, 2010.

[15] P.-J. Meyer and D. V. Dimarogonas. Abstraction refinement and plan revision for control synthesis under high level specifications. In *Proceedings of the* 20^{th} *IFAC World Congress*, pages 9664–9669, 2017.

[16] P.-J. Meyer and D. V. Dimarogonas. Compositional abstraction refinement for control synthesis under lasso-shaped specifications. In *Proceedings of the American Control Conference*, pages 523–528, 2017.

[17] P.-J. Meyer and D. V. Dimarogonas. Compositional abstraction refinement for control synthesis. *Nonlinear Analysis: Hybrid Systems*, 27:437–451, 2018.

[18] T. Moor and J. Raisch. Abstraction based supervisory controller synthesis for high order monotone continuous systems. In *Modelling, Analysis, and Design of Hybrid Systems*, pages 247–265. Springer, 2002.

[19] S. Mouelhi, A. Girard, and G. Gössler. Cosyma: a tool for controller synthesis using multi-scale abstractions. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 83–88. ACM, 2013.

[20] P. Nilsson and N. Ozay. Incremental synthesis of switching protocols via abstraction refinement. In *53rd IEEE Conference on Decision and Control*, pages 6246–6253. IEEE, 2014.

[21] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2016.

[22] M. Rungger and M. Zamani. Scots: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 99–104. ACM, 2016.

[23] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition, 2009.

[24] H. L. Smith. *Monotone dynamical systems: an introduction to the theory of competitive and cooperative systems*, volume 41. American Mathematical Soc., 1995.

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

[26] J. Tumova, B. Yordanov, C. Belta, I. Černá, and J. Barnat. A symbolic approach to controlling piecewise affine systems. In *49th IEEE Conference on Decision and Control*, pages 4230–4235. IEEE, 2010.

[27] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning for dynamical systems. In *48th IEEE Conference on Decision and Control held jointly with the 28th Chinese Control Conference*, pages 5997–6004. IEEE, 2009.

[28] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray. Tulip: a software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 313–314. ACM, 2011.

[29] L. Yang, A. Karnik, B. Pence, M. T. B. Waez, and N. Ozay. Fuel cell thermal management: Modeling, specifications and correct-byconstruction control synthesis. In *Proceedings of American Control Conference*, 2017.

[30] L. Yang and N. Ozay. A note on some sufficient conditions for mixed monotone systems. Technical Report 2027.42/136122, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA, 2017.