LTL Planning for Groups of Robots

Marius Kloetzer, *Student Member, IEEE*, and Calin Belta, *Member, IEEE*

***Abstract*— We approach the general problem of planning and controlling groups of robots from logical and temporal specifications over regions of interest in 2D or 3D environments. The focus of this paper is on planning, and, enabled by our previous results, we assume that the environment is partitioned and described in the form of a graph whose nodes label the partition regions and whose edges capture adjacency relations among these regions. We also assume that the robots can syncronize when penetrating from a region to another. We develop a fully automated framework for generation of robot plans from robot abstract task specifications given in terms of Linear Temporal Logic (LTL) formulas over regions of interest. Inter-robot collision avoidance is guaranteed, and the assignment of plans to specific robots is automatic. The main tools underlying our framework are model checking and bisimilarity equivalence relations.**

I. Introduction

The standard navigation problem for one robot in an environment with obstacles consists of finding a motion of the robot from configuration A to configuration B. There are several approaches to this problem, which include deterministic and probabilistic roadmap methods, potential fields [17], [18], navigation functions [16], cellular decompositions, etc. In several robotics applications, the specification "go from A to B" is either too explicit, or simply does not capture the nature of the task. For example, there might exist two regions A and B in the environment so that the accomplishment of the mission requires the attainment of either A or B. Robotic tasks might also include temporal specifications, such as in reachability tasks ("reach A eventually"), convergence tasks ("reach A eventually and stay there for all future times"), sequencing tasks ("reach A, and then B, and then C"), surveillance tasks ("reach A and then B infinitely often"), etc. Moreover, if more robots are available, the attainment of disjoint regions at the same time might be of interest, as in "reach A and B eventually".

Such specifications containing both logical and temporal operators translate naturally to temporal logic. Several types of temporal logics have been developed [8] for verifying the correctness of computer programs, which can be seen as continuously operating, (concurrent) reactive systems [20]. However, due to their resemblance to natural language, their expressivity, and the existence of off-the-shelf algorithms for model checking, temporal logic has the potential to impact several other areas of engineering. The use of temporal logic for task specification and controller synthesis in mobile robotics has been advocated as far back as [1]. More recently, robot controllers satisfying Linear Temporal Logic (LTL) formulas were constructed by composition of navigation functions in [19]. The UPPAAL model checker was used for motion planning of robots with trivial kinematics from Computational Tree Logic (CTL) formulas in [22]. Finally, based on continuous controllers developed in [4], [2], a control strategy for a fully actuated planar robot moving in a triangulated polygonal environment was developed in [9] from specifications given in terms of LTL formulas over the triangles in the partition.

We address the general problem of planning and controlling a group of robots from specifications given in terms of LTL formulas over predicates in the coordinates of the environment. In our previous work, we developed controllers for fully actuated robots moving in triangulated polygonal environments [4], and for fully actuated robots and unicycles operating in 2D and 3D environments with rectangular partitions [14]. In both cases, we derived necessary and sufficient conditions for the existence of controllers for driving the robot from any initial point in a region to an adjacent region in finite time, and for keeping the robot in a given region for all times.

In this paper, we only focus on the discrete planning problem. Specifically, we consider the following problem: *given a set of n robots, and given an arbitrary LTL formula over the quotient of a partitioned environment, generate motions for all robots so that the formula is satisfied.* Central to our approach is model checking and a certain notion of equivalence, called *bisimulation* [21]. Enabled by our previous results on robot control in partitioned 2D and 3D environments, we make some simplifying assumptions, which allow us to model the environment as a graph (whose nodes label the regions in the partition and whose edges capture adjacency relations) and the set of all possible motions of the robots as transition systems over this graph. A robot motion will correspond to a run of the corresponding transition system. In order to find runs satisfying the formula, we perform model checking (using Büchi automata [10]) on the transition system capturing the set of all possible motions of the team. To this goal, starting from the observation that the task is robot abstract (*i.e.,* it only contains logical and temporal statements about attainment of regions, regardless of the performing robot(s)), we first reduce the size of the problem by showing that the quotient produced by the equivalence relation induced by robot permutations is a bisimilarity quotient. To choose among several motions of the team satisfying a given formula, we select trajectories corresponding to a minimum number of regions visited by the whole team.

From all related works cited above, this paper is closest

The authors are with the Center for Information and Systems Engineering, Boston University, Brookline, MA 02446, USA (phone: 617-353-9586; fax: 617-353-5548; email: {kmarius,cbelta}@bu.edu).

This work is partially supported by NSF CAREER 0447721 and NSF 0410514.

related to [9], [22]. As in [9], we assume a partitioned environment and use LTL formulas as task specifications. However, we consider a team of robots as opposed to just one robot, and automatically generate individual robot plans from robot-abstract specifications. In [22], timed automata are used to model robots moving in an environment partitioned in square cells. Their controllers are synchronized through a central controller, which is again modelled as a timed automaton. The specifications are given as CTL formulas over robot-cell assignments, and UPPAAL is used for model checking. As opposed to [22], in addition to accommodating robot abstract LTL specifications, we model the robots as transition systems. Such models are general enough to accommodate discrete, continuous, and hybrid dynamics, and provide a formal framework for the inclusion of realistic robot continuous and hybrid controllers in future extensions of this work.

II. Preliminaries

Definition 1: A transition system is a tuple $T = (Q, Q_0, \rightarrow, \Pi, \vDash)$, where Q is a set of states, $Q_0 \subseteq Q$ is a set of initial states, $\rightarrow \subseteq Q \times Q$ is a transition relation, Π is a finite set of atomic propositions, and $\vDash \subseteq Q \times \Pi$ is a satisfaction relation.

In this work, we assume that the transition system is *finite* (Q is finite). For an arbitrary proposition $\pi \in \Pi$, we define $[[\pi]] = \{q \in Q | q \vDash \pi\}$ as the set of all states satisfying it. Conversely, for an arbitrary state $q \in Q$, let $\Pi_q = \{\pi \in \Pi \,|\, q \vDash \pi\}$, $\Pi_q \in 2^{\Pi}$, denote the set of all atomic propositions satisfied at q. A *trajectory* or *run* of T starting from q is an infinite sequence $r = r(1)r(2)r(3)\ldots$ with the property that $r(1) = q$, $r(i) \in Q$, and $(r(i), r(i+1)) \in \rightarrow$, for all $i \geq 1$. A trajectory $r = r(1)r(2)r(3)\ldots$ defines a *word* $w = w(1)w(2)w(3)\ldots$, where $w(i) = \Pi_{r(i)}$. The set of all words generated by the set of all trajectories starting at $q \in Q$ is called the *language* of q and is denoted by $L(q)$. The language of the transition system T is defined as $L(T) = \bigcup_{q \in Q_0} L(q)$.

An equivalence relation $\sim \subseteq Q \times Q$ over the state space of T is *proposition preserving* if for all $q_1, q_2 \in Q$ and all $\pi \in \Pi$, if $q_1 \sim q_2$ and $q_1 \vDash \pi$, then $q_2 \vDash \pi$. A proposition preserving equivalence relation naturally induces a quotient transition system $T/_\sim = (Q/_\sim, Q_0/_\sim, \rightarrow_\sim, \Pi, \vDash_\sim)$. $Q/_\sim$ is the quotient space (the set of all equivalence classes). The transition relation $\rightarrow_\sim$ is defined as follows: for $P_1, P_2 \in Q/_\sim$, $P_1 \rightarrow_\sim P_2$ if and only if there exist $q_1 \in P_1$ and $q_2 \in P_2$ such that $q_1 \rightarrow q_2$. The satisfaction relation is defined as follows: for $P \in Q/_\sim$, we have $P \vDash_\sim \pi$ if and only if there exist $q \in P$ such that $q \vDash \pi$. $Q_0/_\sim$ is the set of all $P \in Q/_\sim$ containing at least one $q \in Q_0$.

Definition 2: A proposition preserving equivalence relation $\sim$ is a bisimulation of a transition system $T = (Q, Q_0, \rightarrow, \Pi, \vDash)$ if for all states $p, q \in Q$, if $p \sim q$ and $p \rightarrow p'$, then there exist $q' \in Q$ such that $q \rightarrow q'$ and $p' \sim q'$.

If $\sim$ is a bisimulation, then the quotient transition system $T/_\sim$ is called a *bisimulation quotient* of T, and the transition systems T and $T/_\sim$ are called *bisimilar*. Bisimilar systems have equivalent reachability properties, or more generally, preserve properties specified in terms of temporal logics such as LTL, CTL, and CTL* [8]. In this paper, we are only concerned with LTL_{-X}, a subclass of LTL, which we will call for short *linear temporal logic* (LTL).

Definition 3: [Syntax of LTL_{-X} formulas] A linear temporal logic LTL_{-X} formula over Π is recursively defined as follows:

- Every atomic proposition π_i, $i = 1, \ldots, K$ is a formula, and
- If ϕ_1 and ϕ_2 are formulas, then $\phi_1 \vee \phi_2$, $\neg\phi_1$, $\phi_1 \mathcal{U} \phi_2$ are also formulas.

The semantics of LTL_{-X} formulas are given over words in the language of the transition system T.

Definition 4: [Semantics of LTL_{-X} formulas] The satisfaction of formula ϕ at position $i \in \mathbb{N}$ of word w, denoted by $w(i) \vDash \phi$, is defined recursively as follows:

- $w(i) \vDash \pi$ if $\pi \in w(i)$,
- $w(i) \vDash \neg\phi$ if $w(i) \nvDash \phi$,
- $w(i) \vDash \phi_1 \vee \phi_2$ if $w(i) \vDash \phi_1$ or $w(i) \vDash \phi_2$,
- $w(i) \vDash \phi_1 \mathcal{U} \phi_2$ if there exist a $j \geq i$ such that $w(j) \vDash \phi_2$ and for all $i \leq k < j$ we have $w(k) \vDash \phi_1$

A word w satisfies an LTL formula ϕ, written as $w \vDash \phi$, if $w(1) \vDash \phi$. The transition system T satisfies formula ϕ, written as $T \vDash \phi$, if $w \vDash \phi$, $\forall w \in L(T)$.

The symbols $\neg$ and $\vee$ stand for negation and disjunction. The boolean constants $\top$ and $\bot$ are defined as $\top = \pi \vee \neg\pi$ and $\bot = \neg\top$. The other Boolean connectors $\wedge$ (conjunction), $\Rightarrow$ (implication), and $\Leftrightarrow$ (equivalence) are defined from $\neg$ and $\vee$ in the usual way. The *temporal operator* $\mathcal{U}$ is called the *until* operator. Formula $\phi_1 \mathcal{U} \phi_2$ intuitively means that (over a word) ϕ_2 will eventually become true and ϕ_1 is true until this happens. Two useful additional temporal operators, "eventually" and "always" can be defined as $\Diamond\phi = \top \mathcal{U} \phi$ and $\Box\phi = \phi \mathcal{U} \bot$, respectively. Formula $\Diamond\phi$ means that ϕ becomes eventually true, whereas $\Box\phi$ indicates that ϕ is true at all positions of w. More expressiveness can be achieved by combining the temporal operators. Examples include $\Box\Diamond\phi$ (ϕ is true infinitely often) and $\Diamond\Box\phi$ (ϕ becomes eventually true and stays true forever).

III. Problem formulation and approach

Enabled by our previous results on robot control in partitioned 2D and 3D environments [4], [14], we make the following simplifying assumptions, which allow us to focus on the discrete motion planning problem: (1) the environment is assumed to be already partitioned, (2) the task is specified in terms of an LTL formula over the partition quotient, (3) the robots are assumed to be able to syncronize when penetrating from a region to another, and (4) the robots are identically actuated. Assumptions (1) and (2) are not restrictive, since as we show in [15], an arbitrary LTL specification over arbitrary linear predicates can automatically generate a partition satisfying assumption (2). Also, on assumption (3), synchronization can be easily achieved by endowing the robots with a 'wait' mode (enabled at the boundaries of the regions) and a shared syncronization input. Assumption (4) will allow us to solve the problem in a computationally feasible way, by using a bisimilarity equivalence relation.

To model the environment, we assign a symbol to every region of the partition and assume that the adjacency among regions is known. In other words, we describe the environment as a graph

$$E = (Q, A), \tag{1}$$

where Q is a finite set of symbols (one for each element in the partition), and $A \subset Q \times Q$ is a symmetric relation capturing adjacency of regions. Such partitions can be obtained by triangulation [4], rectangular grids [14], or polytopal decompositions [15].

The motion of each robot is abstracted to the sequence of regions reached by the robot, *i.e.,* to a sequence of symbols from Q. A robot can move from region q_1 to region q_2 if the regions are adjacent and there exists a feedback controller driving the robot from any initial point in q_1 to q_2 in finite time. Similarly, a robot can stay in a region q for all future times if there exists a feedback controller making the region an invariant for the robot, independent of its initial position. Such controllers for triangular partitions are described in [2], for rectangular partitions in [3] and for partitions in convex polytopes in [12], [15]. Formally, we model the i-th robot as a transition system

$$T_i = (Q_i, q_{i0}, \rightarrow_i, \Pi_i, \vDash_i),\ i = 1, \ldots, n, \tag{2}$$

where $Q_i = Q$, $q_{i0} \subseteq Q$ is the initial position of the robot (a singleton), $\rightarrow_i \subseteq A \cup_{q \in Q} (q, q)$ is the transition relation which captures our capability to design controllers for robot i so that it moves in finite time between adjacent regions or stays inside a region for all times, $\Pi_i = Q$, and $\vDash$ is the trivial satisfaction relation $(q, \pi) \in \vDash$ if and only if $q = \pi$. In other words, the proposition attached to symbol q is q itself.

A motion of robot i is a word produced by a run of T_i from equation (2) as defined in Section II. We assume that the robots do not overlap initially, *i.e.,* $q_{i0} \neq q_{j0}$, for all $i, j = 1, \ldots, n$, $i \neq j$. Assumption (3) above means that the transitions of T_i, $i = 1, \ldots, n$ occur at the same time, [1] while assumption (4) implies that $\rightarrow_i = \rightarrow_j$, for all $i, j = 1, \ldots, n$.

We are now ready to formulate the main problem:

Problem 1: Given a task specified as an arbitrary LTL formula ϕ over an arbitrary subset of partition symbols Q, determine a motion for each robot i (run of T_i), $i = 1, \ldots, n$ so that the formula ϕ is satisfied.

Before we start to outline our approach, some comments are in order. First, note that the task specification is "robot-abstract", *i.e.,* the accomplishment of the task requires that certain Boolean combinations of regions are attained in certain order during the motion of the team, while the exact assignment robot-region is not of interest. Second, note that the satisfiability of the formula might depend on the number of the robots in the team. For example, a formula of the type "$\Diamond q$" can be satisfied by a robot that eventually reaches region q. However, "$\Diamond(q_1 \wedge q_2)$" cannot be satisfied by one robot, since the regions q_1 and q_2 are distinct, and one robot cannot be in q_1 and q_2 at the same time. Third, in general there might exist several robot runs satisfying a given formula.

Our solution to Problem 1 starts with the definition of a transition system capturing all possible motions of the team. Since the formula is robot abstract, we declare its states that are related through robot permutations equivalent, and show that the obtained quotient transition system is a bisimilarity quotient (Section IV). We then find runs of the bisimulation quotient satisfying the formula using Büchi automata and generate the individual robot runs (Sections V, VI). To choose among the several possible runs, we select those runs consisting of the smallest number of robot transitions (without counting transitions for staying inside a region).

IV. The motion of the group of robots

To define the motion of the team of synchronized robots, we define a team (group) transition system. In what follows, we use the usual notations $(q_1, \ldots, q_n)$ to denote the n-tuple formed by the elements $q_1, \ldots, q_n$ (ordered set) and $\{q_1, \ldots, q_n\}$ to denote the (unordered) set formed by $q_1, \ldots, q_n$.

Definition 5: The transition system $T_g = (Q_g, Q_{g0}, \rightarrow_g, \Pi_g, \vDash_g)$ capturing the behavior of the group of n robots is defined as

- $Q_g \subset Q_1 \times \ldots \times Q_n$ is defined by $(q_1, \ldots, q_n) \in Q_g$ if and only if $q_i \neq q_j$ for $i \neq j$,
- $Q_{g0} = (q_{10}, \ldots, q_{n0})$,
- $\rightarrow_g \subset Q_g \times Q_g$ is defined by $((q_1, \ldots, q_n), (q'_1, \ldots, q'_n)) \in \rightarrow_g$ if and only if $(q_i, q'_i) \in \rightarrow_i$ and for all $i, j = 1, \ldots, n$ with $i \neq j$, if $q'_i = q_j$, then $q'_j \neq q_i$,
- $\Pi_g = Q$,
- $\vDash_g \subset Q_g \times \Pi_g$ is defined by $((q_1, \ldots, q_n), \pi) \in \vDash_g$ if $\pi \in \{q_1, \ldots, q_n\}$.

In other words, the states of the transition system T_g capture all possible ways in which the regions labelled with symbols from Q are occupied by the n robots. Configurations in which two robots overlap (occupy the same region) are excluded. The possible motions of the team are modelled by the its transitions. A transition of T_g occurs when all robots take synchronously allowed transitions, and we exclude the case when two robots in adjacent regions switch positions, since this could cause collision. Finally, each team configuration is equipped with n predicates enumerating the regions occupied by the robots, without explicitly specifying the exact position of each robot.

Let a denote the map taking an n-tuple and producing the corresponding set, *i.e.,* $a((q_1, \ldots, q_n)) = \{q_1, \ldots, q_n\}$.

[1]Syncronization of transition systems is usually achieved through shared actions, or inputs. Definition 1 can be easily modified to include actions. T_g from Definition 5 then becomes a synchronous product. However, since we assume that all robots take transitions at the same time, this is equivalent to all of them sharing the same action, and we omit writing the action explicitly, since there is no ambiguity.

Over the states of T_g, we define an equivalence relation $\sim_a \subset Q_g \times Q_g$ by

$$((q_1, \ldots, q_n), (q'_1, \ldots, q'_n)) \in \sim_a \text{ if } \\ a((q_1, \ldots, q_n)) = a((q'_1, \ldots, q'_n)) \tag{3}$$

It is obvious that $\sim_a$ is a proposition preserving equivalence relation.

Proposition 1: The transition system $T_g/_{\sim_a}$ is a bisimulation quotient.

Proof: We need to show that $\sim_a$ is a bisimulation relation, as in Definition 2. Definition 5 implies that for any state $p = (q_1, \ldots, q_n) \in Q_g$ there are $n!$ states $q \in Q_g$ (including $q = p$) such that $(p, q) \in \sim_a$. From all these possible permutations of symbols $q_1, \ldots, q_n$, choose for simplicity of notation $q = (q_n, \ldots, q_1)$. For any transition $(p, p') \in \rightarrow_g$ with $p' = (q'_1, \ldots, q'_n) \in Q_g$, there exists $q' = (q'_n, \ldots, q'_1) \in Q_g$ such that $(p', q') \in \sim_a$. Since $(p, p') \in \rightarrow_g$, $(q_i, q'_i) \in \rightarrow_i$, $i = 1, \ldots, n$. Using $\rightarrow_i = \rightarrow_j$, for all $i, j = 1, \ldots, n$ we obtain that $(q_{n-i+1}, q'_{n-i+1}) \in \rightarrow_i$, $i = 1, \ldots, n$, or $(q, q') \in \rightarrow_g$, which proves the Proposition. ■

Note that there is a significant decrease in the number of states from T_g to $T_g/_{\sim_a}$. Indeed, if there are N elements in Q, then T_g has $N!/(N-n)!$ states, while $T_g/_{\sim_a}$ has $N!/((N-n)!n!)$ states. However, based on Theorem 1, T_g and $T_g/_{\sim_a}$ are equivalent with respect to the satisfaction of LTL formulas. Therefore, in Section V, we will produce runs of $T_g/_{\sim_a}$ satisfying formula ϕ. From these, in Section VI, we will produce runs of the original T_g and and finally runs of each T_i.

V. Producing runs of $T_g/_{\sim_a}$

Formula ϕ from Problem 1 is over symbols in the set Q. The set of predicates of $T_g/_{\sim_a}$ is $\Pi_g = Q$ (the same as the set of predicates of T_g, in accordance with Section II). Therefore, the semantics of formula ϕ can be interpreted over words in the language of $T_g/_{\sim_a}$.

In this section, we outline a procedure for finding runs of $T_g/_{\sim_a}$ satisfying an arbitrary LTL formula ϕ over Π_g. We start by translating ϕ into a Büchi automaton $\mathcal{B}$. Then we take the product of $T_g/_{\sim_a}$ with $\mathcal{B}$ to obtain a product automaton $\mathcal{A}$, whose accepted runs will only include trajectories of $T_g/_{\sim_a}$ that satisfy formula ϕ. We use standard algorithms for graph traversing on $\mathcal{A}$ and eventually project back to find the desired runs of $T_g/_{\sim_a}$.

To simplify the notation and keep the discussion at a general level, in the rest of this section we focus on an arbitrary transition system $T = (Q, Q_0, \rightarrow, \Pi, \vDash)$ and an arbitrary LTL formula ϕ over Π.

Definition 6 (Büchi automaton): A Büchi automaton is a tuple $\mathcal{B} = (S, S_0, \Sigma, \rightarrow_{\mathcal{B}}, F)$, where

- S is a finite set of states,
- $S_0 \subseteq S$ is the set of initial states,
- Σ is the input alphabet,
- $\rightarrow_{\mathcal{B}} \subseteq S \times \Sigma \times S$ is a nondeterministic transition relation,
- $F \subseteq S$ is the set of accepting (final) states.

The semantics of a Büchi automaton is defined over infinite input words. Let $\omega = \omega_1\omega_2\omega_3...$ be an infinite input word of automaton $\mathcal{B}$, $\omega_i \in \Sigma$, $\forall i \in \mathbb{N}^*$. We denote by $\mathcal{R}_{\mathcal{B}}(\omega)$ the set of all initialized runs of $\mathcal{B}$ that can be generated by ω:

$$\mathcal{R}_{\mathcal{B}}(\omega) = \{r = s_1s_2s_3...|s_1 \in S_0, \\ (s_i, \omega_i, s_{i+1}) \in \rightarrow_{\mathcal{B}}, \forall i \in \mathbb{N}^*\} \tag{4}$$

Definition 7 (Büchi acceptance): A word ω is accepted by the Büchi automaton $\mathcal{B}$ (the word satisfies the automaton) if and only if $\exists r \in \mathcal{R}_{\mathcal{B}}(\omega)$ so that $inf(r) \cap F \neq \emptyset$, where $inf(r)$ denotes the set of states appearing infinitely often in the run r.

In [5], it was proved that, for any LTL formula ϕ over a set of atomic propositions Π, there exists a Büchi automaton $\mathcal{B}$ with input alphabet $\Sigma \subseteq 2^{\Pi}$ accepting *all and only* the infinite strings over Π satisfying formula ϕ. In this paper, we use the conversion algorithm described in [10] and its freely downloadable implementation, LTL2BA.

Definition 8 (Product automaton): The product automaton $\mathcal{A} = T \times \mathcal{B}$ between the transition system $T = (Q, Q_0, \rightarrow, \Pi, \vDash)$ and the Büchi automaton $\mathcal{B} = (S, S_0, \Sigma, \rightarrow_{\mathcal{B}}, F)$ with $\Sigma \subseteq 2^{\Pi}$ is defined as the tuple $\mathcal{A} = (S_{\mathcal{A}}, S_{\mathcal{A}0}, \rightarrow_{\mathcal{A}}, F_{\mathcal{A}})$, where:

- $S_{\mathcal{A}} = (Q \cup \{q_0\}) \times S$ is the finite set of states,
- $S_{\mathcal{A}0} = \{q_0\} \times S$ is the set of initial states,
- $\rightarrow_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times S_{\mathcal{A}}$ is the transition relation, defined as: $\{(q_i, s_j), (q_k, s_l)\} \in \rightarrow_{\mathcal{A}}$ if and only if $(q_i, q_k) \in (\rightarrow \cup (\{q_0\} \times Q_0))$ and $(s_j, \Pi_{q_k}, s_l) \in \rightarrow_{\mathcal{B}}$,
- $F_{\mathcal{A}} = Q \times F$ is the set of accepting (final) states.

In the above definition, q_0 is a dummy initial state. Product automaton $\mathcal{A}$ can be seen as a Büchi automaton without an input alphabet (*i.e.*, the transitions have no guards). The acceptance condition of $\mathcal{A}$ is formulated similar to Definition 7, but with respect to runs instead of input words. Explicitly, a run of $r_{\mathcal{A}}$ of $\mathcal{A}$ is accepted if it satisfies the transitions $\rightarrow_{\mathcal{A}}$ and $inf(r_{\mathcal{A}}) \cap F_{\mathcal{A}} \neq \emptyset$. The product automaton in Definition 8 can be regarded as a match between the states of T and the transitions of $\mathcal{B}$, therefore $\mathcal{A}$ is sometimes referred to as the synchronous product [13].

For any initialized run $r_{\mathcal{A}} = (q_0, s_{j1})(q_{i1}, s_{j2})(q_{i2}, s_{j3})...$ of automaton $\mathcal{A}$, we define the projection $\gamma_T(r_{\mathcal{A}}) = q_{i1}q_{i2}...$, which maps $r_{\mathcal{A}}$ to the corresponding run of T. In [13], it was proved that, if ϕ is an arbitrary LTL formula over Π and $\mathcal{B}$ is a corresponding Büchi automaton, then the projection $\gamma_T(r_{\mathcal{A}_\phi})$ of any accepted run $r_{\mathcal{A}}$ of $\mathcal{A} = T \times \mathcal{B}$ is a run of T satisfying LTL formula ϕ. Thus, a run of T satisfying specification ϕ exists if and only if $\mathcal{A}$ has an accepted run [11], [9], which in turn is equivalent to the existence of a strongly connected component of the directed graph corresponding to $\mathcal{A}$ reachable from at least one initial state and containing at least one accepting state [13].

Any accepted run of $\mathcal{A}$ starts from the initial state and contains an infinite number of occurrences of one or more final states. To find such runs, we used Dijkstra's algorithm [7], [6]. We only consider accepted runs $r_{\mathcal{A}}$ of $\mathcal{A}$ composed of two parts: (1) a *prefix*, containing a trajectory from an initial state to an accepting state (excluding it) and (2) a *suffix*, containing a trajectory starting and ending at the above accepting state (excluding the last occurrence of this final state if the suffix contains more than one state). The actual run $r_{\mathcal{A}}$ consists of the concatenation of the prefix and an infinite number of suffix repetitions.

After projection, the corresponding run $r = r(1)r(2)r(3)\dots$ of T inherits the prefix-suffix structure. In other words, there exist n_p and n_s such that for any $i > n_p + n_s$, $r(i) = r((i - n_p - 1) \bmod n_s + n_p + 1)$. n_p and n_s are the number of states in prefix and suffix of r, respectively and thus the run r contains at most $n_p + n_s$ different states. In addition, it can be proved [15] that in a run r of T, none of the states can be succeeded by itself, except for the state of a suffix of length one (case in which this state will be infinitely repeated). In other words, $r(i) \neq r(i+1)$, $\forall i \in \mathbb{N}^*$, $i \neq n_p + k\,n_s + 1$, $k \in \mathbb{N}$. Moreover, if $n_s \geq 2$, $r(i) \neq r(i+1)$, $\forall i \in \mathbb{N}^*$.

Due to space constraints, we do not give here an explicit description of the algorithm returning runs for transition system T satisfying an LTL formula ϕ and we refer the interested reader to [15]. However, in contrast with [15], we need to point out that we associate weights (costs) with transitions of T ($T_g/_{\sim_a}$). These costs are equal with the minimum number of transitions from T_i, $i = 1, \dots, n$ that guarantee a transition of $T_g/_{\sim_a}$. Then, for a given initial state of $T_g/_{\sim_a}$, we find a prefix with minimum cost and a corresponding suffix with minimum cost.

On the complexity of the computation, let us denote by ν_1 the number of predicates appearing in the LTL formula. Then an upper limit for the number of states of the Büchi automaton $\mathcal{B}$ is $\nu_1 2^{\nu_1}$. However, this limit is almost never achieved. The number of states of the product automaton $\mathcal{A}$ (say ν_2) equals the product of the number of states of $T_g/_{\sim_a}$ and the number of states of the Büchi automaton $\mathcal{B}$, therefore an upper limit for ν_2 is $\nu_1 2^{\nu_1} N!/((N-n)!n!)$. Finally, the complexity of Dijkstra's algorithm is $O(\nu_2^2)$.

VI. Robot motion plans

In order to determine individual robot plans (*i.e.,* runs of T_i starting from q_{i0}, $i = 1, \dots, n$), we first need to find a run of T_g starting from $(q_{10}, \dots, q_{n0})$. Let $r = r(1)r(2)...$ be a run of $T_g/_{\sim_a}$ starting from $r(1) = a((q_{10}, \dots, q_{n0})) = \{q_{10}, \dots, q_{n0}\}$ and satisfying formula ϕ (found as described in Section V). Let us denote the run of T_g that we want to find by $r_{T_g} = r_{T_g}(1) r_{T_g}(2)...$, with $r_{T_g}(1) = (q_{10}, \dots, q_{n0})$.

For finding $r_{T_g}(2)$, we find all states $p \in Q_g$ such that $(r_{T_g}(1), p) \in \rightarrow_g$ and $a(p) = r(2)$. Proposition 1 guarantees that there exists at least one such state p. If there are more, we choose one corresponding to the minimum number of robots leaving their currently occupied region. We iteratively use this method $(n_p + n_s - 1)$ times and we find a prefix-suffix structure of r_{T_g}, similar with the one of r. The minimum cost of prefix and suffix of r, together with the way we match transitions of $T_g/_{\sim_a}$ with transitions of T_g, implies that there will be no robots spending energy in unnecessary movements.

Further, we easily find the robot specific runs by projecting each state of r_{T_g} on states of T_i, using n trivial projection maps $\alpha_i : Q_g \to Q_i$, $\alpha_i((q_1, \dots, q_n)) = q_i$, $i = 1, \dots, n$. Note that the runs of both T_g and all T_i, $i = 1, \dots, n$ inherit the prefix-suffix structure of runs from $T_g/_{\sim_a}$, with the same number of states in prefix (n_p) and suffix (n_s). This particular structure is appropriate for real implementations, because of the way it manages an infinite run by storing a finite number of its elements.

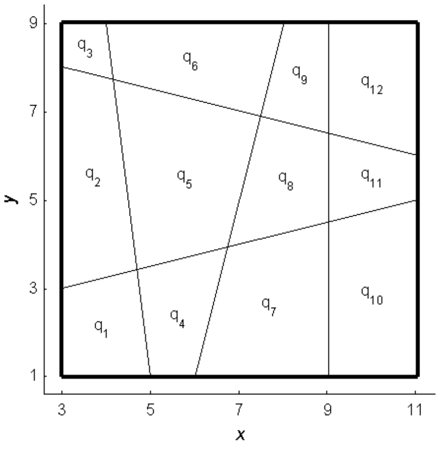

Fig. 1. Partitioned planar environment with 12 regions.

VII. Case study

Consider the partitioned planar environment shown in Figure 1 (such "polytopal" partitions can be automatically constructed from linear predicates in spaces of arbitrary dimension using the algorithms presented in [15]). Assume we have $n = 2$ robots in the team and the task is specified as the following LTL formula:

$$\phi = \diamond(q_{10} \wedge q_{12} \wedge \Box\diamond(q_5 \wedge q_8 \wedge \diamond(q_6 \wedge q_8))) \wedge \Box\neg(q_4 \vee q_7 \vee q_9) \tag{5}$$

Formula 5 represents a reachability followed by surveillance task, involving obstacle avoidance, which can be informally read as: "eventually reach regions q_{10} and q_{12}, after that infinitely often reach q_5 and q_8, then q_6 and q_8, and always avoid regions q_4, q_7, and q_9".

The transition system T_g has 132 states, its bisimulation quotient $T_g/_{\sim_a}$ has 66 states, and the corresponding Büchi automaton has 6 states. There are 36 initial states of $T_g/_{\sim_a}$ starting from which formula ϕ can be satisfied, which correspond to 72 robot-specific configurations. We chose the initial configuration (q_1, q_3) and obtained a corresponding run with prefix of length $n_p = 9$ and suffix of length $n_s = 2$. Figure 2 shows the resulted motion of the team, which corresponds to the smallest number of robot transitions.

VIII. Conclusion

In this paper, we present a fully automated framework for planning of a team of robots from task specifications

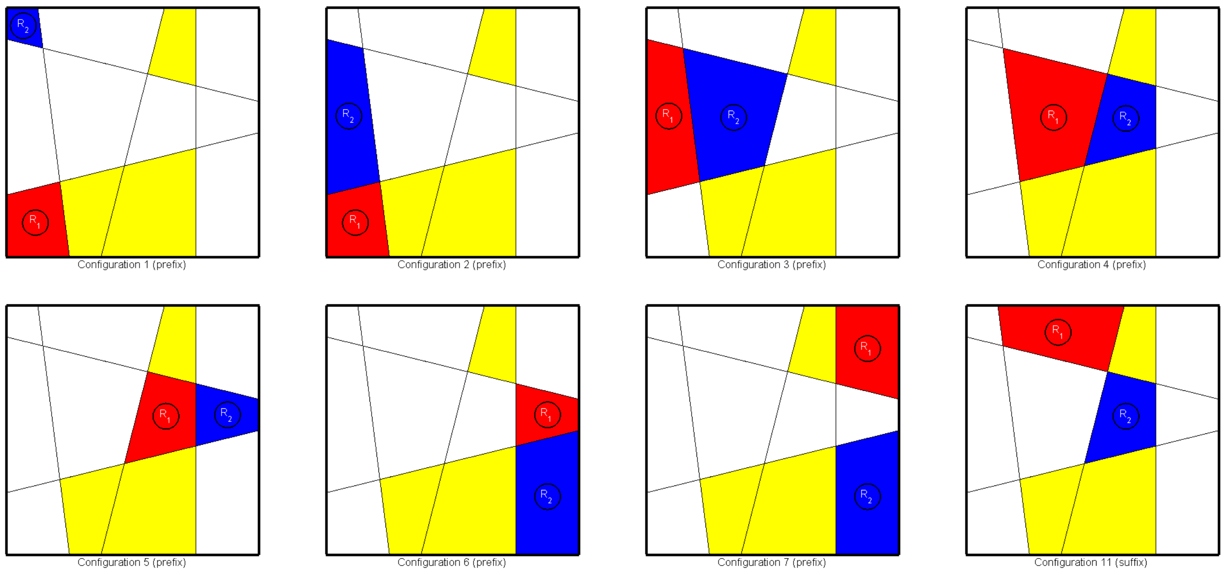

Fig. 2. All different configuration reached by R_1 and R_2 during the motion satisfying formula ϕ. The region occupied by R_1 is shown in red, the region occupied by R_2 is shown in blue, while the regions corresponding to obstacles are shown in yellow. The 8^{th} configuration from prefix is identical with the 6^{th} one, the 9^{th} configuration from prefix is identical with the 5^{th} one, and the 10^{th} configuration (first one from suffix) is identical with the 4^{th} one from prefix. Configurations 10 and 11 are repeated infinitely often.

given in terms of LTL formulas over the quotient of a partitioned environment. We model the robots as transition systems, and use model checking techniques to generate robot plans satisfying a formula. We also exploit the robot-abstract structure of the specifications to reduce the size of the model checking problem by using bisimulations.

REFERENCES

[1] M. Antoniotti and B. Mishra. Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers. In *IEEE International Conference on Robotics and Automation*, May 1995.

[2] C. Belta and L.C.G.J.M. Habets. Constructing decidable hybrid systems with velocity bounds. In *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004.

[3] C. Belta and L.C.G.J.M. Habets. Control of a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 2005. to appear.

[4] C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot planning and control in polygonal environments. *IEEE Trans. on Robotics*, 21(5):864–874, 2005.

[5] J. R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel et al., editor, *Proceedings of the International Congress on Logic, Methodology and Philosophy of Science 1960*, Stanford, CA.

[6] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, Cambridge, Massachusetts and New York, 2nd edition, 2001.

[7] E.W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, volume 1, pages 269–271. Mathematisch Centrum, Amsterdam, The Netherlands, 1959.

[8] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 995–1072. North-Holland Pub. Co./MIT Press, 1990.

[9] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005.

[10] P. Gastin and D. Oddoux. Fast ltl to büchi automata translation. In H. Comon G. Berry and A. Finkel, editors, *Proceedings of the 13th Conference on Computer Aided Verification (CAV'01)*, number 2102, pages 53–65, 2001.

[11] G. De Giacomo and M.Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *Proceedings of the 5th European Conference on Planning (ECP '99)*, volume 1809, pages 226–238, 1999.

[12] L.C.G.J.M. Habets and J.H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004.

[13] G. Holzmann. *The Spin Model Checker, Primer and Reference Manual*. Addison-Wesley, Reading, Massachusetts, 2004.

[14] M. Kloetzer and C. Belta. A framework for automatic deployment of robots in 2d and 3d environments. In *International Conference on Robotics and Automation*, Orlando, FL, 2006. submitted.

[15] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from ltl specifications. In *The 9th International Workshop on Hybrid Systems: Computation and Control*, Santa Barbara, CA, 2006. to appear.

[16] D. E. Koditschek. The control of natural motion in mechanical systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, 113(4):548–551, 1991.

[17] J.C. Latombe. *Robot Motion Planning*. Kluger Academic Pub., 1991.

[18] S. M. LaValle and M. S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *Workshop on the Algorithmic Foundations of Robotics*, Nice, France, 2002.

[19] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multiagent motion tasks based on ltl specifications. In *43rd IEEE Conference on Decision and Control*, December 2004.

[20] Z. Manna and A. Pnueli. *The temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, 1992.

[21] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[22] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi. Multi-robot motion planning: A timed automata approach. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, page 44174422, New Orleans, LA, April 2004.