2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

On the Need for Communication in Distributed Implementations of LTL Motion Specifications

M. Kloetzer, S. Itani, S. Birch, and C. Belta

Abstract— **We revisit the problem of automatic deployment of robotic teams from temporal logic specifications over regions of interests in the environment. In our previous work, we developed an algorithm that could accommodate arbitrary communication constraints, but had two main limitations: (1) it only allowed for communicating robots to move, and (2) it was computationally very expensive. In this paper, we present two approaches to address these limitations. First, we show that if identical robots are allowed to communicate for all times, then the computation is cheaper. Second, we develop an algorithm to test if a given global specification can be implemented by the robots without the move-only-when-communicate constraint.**

I. Introduction

The goal in robot motion planning and control is to specify a motion task in a rich, high-level language and have the robot(s) automatically convert this specification into low level primitives, such as feedback controllers and communication protocols, to accomplish the task [1]. This work is motivated by two limitations of the current approaches to robot motion planning and control. First, in the "classical" motion planning problem, the specification is given simply as "go from A to B and avoid obstacles", where A and B are two regions of interest in the environment. This is not rich enough to accommodate missions that might require the attainment of either A or B, surveillance ("reach A and then B infinitely often"), or the satisfaction of more complicated temporal and logic conditions about the reachability of regions of interest (*e.g.,*, "Never go to A. Don't go to B unless C or D were visited."). Second, the problem of automatic deployment of robotic teams (*i.e.,* generation of control and communication strategies) from such rich, global specifications is not well understood.

In our previous work [2], we considered a purely discrete scenario, in which the environment was modeled as a (partition quotient) finite graph, and a team of robots moved among the vertices of the graph. Any two robots were constrained to communicate only when in particular pairs of vertices, which were specified as an arbitrary relation over the vertices of the graph. We developed an algorithm for the automatic generation of motion and communication strategies from arbitrary specifications given as Linear Temporal Logic (LTL) formulas over the vertices of the graph. The approach had two main limitations: (1) it only allowed for communicating robots to move, and (2) it was computationally very expensive.

In this paper, we present two approaches to address these limitations. First, we assume that the robots can communicate from all vertices and that they are identical. While the first assumption trivially eliminates limitation (1) from above, the second assumption allows for significant decrease in computation time. Second, we assume that the robots can communicate only when in adjacent vertices of the graph and develop an algorithm to test if a given global specification can be implemented by the robots without inter-robot communication except for to ensure collision avoidance. We illustrate the methods with simulation and experimental results in our Robotic Urban-Like Environment (RULE) (see `hyness.bu.edu/rule/`).

The use of temporal logic for task specification and controller synthesis in mobile robotics has been advocated as far back as [4], and recent works include [5], [6], [7], [8], [9]. As opposed to [8], [9], here we consider teams, rather than just single agents. The closest related works are [6], [7]. In [6], the specifications are given as formulas of timed CTL and control strategies are generated using the UPPAAL model checker. In [7], the policy for producing robot actions is obtained by applying a game-theoretic framework. While this allows for accommodating environmental events, the specification language is limited to formulas in the GR(1) fragment of LTL. This work is also related (and draws inspiration from) recent results in concurrency theory [10].

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 (or observations), and $\vDash \subseteq Q \times \Pi$ is a satisfaction relation.

For $q \in Q$, let $\Pi_q = \{\pi \in \Pi \,|\, q \vDash \pi\}$, $\Pi_q \in 2^{\Pi}$, denote the set of all 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* over the set 2^{Π}, $w = w(1)w(2)w(3)\ldots$, where $w(i) = \Pi_{r(i)}$. The set of all words of T is called the (ω-) language of T.

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

This work was partially supported at Boston University by the NSF CNS-0834260, ARO W911NF-09-1-0088, and AFOSR FA9550-09-1-0209.

M. Kloetzer is with the Technical University of Iasi, Romania, E-mail : kmarius@ac.tuiasi.ro
S. Itani is with Massachusetts Institute of Technology, Cambridge, MA, USA, E-mail: itani@mit.edu
S. Birch is with the BU Academy, Boston, MA, E-mail: Samuel_Birch@buacademy.org
C. Belta is with Boston University, MA, USA, E-mail: cbelta@bu.edu
C. Belta is the corresponding author

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 $T = (Q, Q_0, \rightarrow, \Pi, \vDash)$ 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 $T/_\sim$ is called a *bisimulation quotient* of T, and the transition systems T and $T/_\sim$ are called *bisimilar*. Bisimilar systems are equivalent with respect to the satisfaction of properties specified as formulas of temporal logics such as LTL, CTL, and CTL* [11].

Here we consider motion specifications given as formulas of a fragment of Linear Temporal Logic (LTL) [12] called LTL_{-X}, which we will simply denote by LTL. Informally, its formulas are recursively defined over a set of propositions Π, by using the standard boolean operators and a set of temporal operators. The boolean operators are $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and some temporal operators that we use include $\mathcal{U}$ (standing for "until"), $\Box$ ("always"), $\Diamond$ ("eventually"). LTL formulas are interpreted over infinite words over the set 2^Π, like those generated by transition system T. The expressivity of LTL makes it suited for specifying motion tasks, such as reachability ("reach π_1 eventually", written as $\Diamond\pi_1$), reachability and obstacle avoidance ("reach π_1 eventually, while always avoiding π_2", written as $\Diamond\pi_1 \wedge \Box\neg\pi_2$), convergence tasks ("reach π_1 eventually and stay there for all future times" - $\Diamond\Box\pi_1$), etc.

Checking whether the language of a transition system T satisfies an LTL formula ϕ over Π is called model checking. Available off-the-shelf packages for LTL model checking include NuSMV [13] and SPIN [14]. In short, given a transition system T and an LTL formula ϕ over its set of propositions, a model checker will return the initial states for which the language satisfies the formula. If the language generated from a state does not satisfy the formula, the model checker returns a counterexample. Among the several runs of T satisfying a formula ϕ, there are some with a particular structure of a *prefix* followed by infinitely many repetitions of a *suffix*. If there exists a run of T satisfying ϕ starting from an initial state, then there always exists a run with the above particular prefix-suffix structure from that state [14]. Such a run is of particular interest to us.

In this paper, the problem of finding runs of T satisfying a formula ϕ is of special interest. To this goal, one can use an off-the-shelf model checker. Indeed, by feeding T and $\neg\phi$ into a model checker, if the formula is not satisfied at a state, the model checker will return that initial state together with a counterexample, which is a run satisfying ϕ. However, the user does not have full control over the structure of the produced counterexamples. Even though some model checkers (e.g., SPIN) allow to retrieve the "shortest" counterexample, this might not satisfy a particular structure of interest. An attractive alternative is to use the tool proposed in [9], in which we construct the set of all satisfying runs, and allow the user to choose runs with specific structure. In particular, we can look for runs in the prefix-suffix form, for runs which are "minimal" with respect to pre-defined transition costs, for runs that do not have finitely many consecutive repetitions of a symbol, etc.

III. Definitions and Problem Formulation

Let

$$G = (P, \rightarrow_G) \tag{1}$$

be a graph, where $P = \{p_1, \ldots, p_k\}$ is the set of vertices and $\rightarrow_G \subset P \times P$ is a relation modelling the set of edges. For example, G can be the quotient graph of a partitioned environment, where P is a set of labels for the regions in the partition, and $\rightarrow_G$ is the corresponding adjacency relation. In the particular case study considered in this paper, G is a graph representation of a city, where P is a set of labels for roads, intersections, and parking spaces, and $\rightarrow_G$ shows how these are accessible from each other. Assume we have a team of n, $n < k$, robots that can move between adjacent vertices of G. We assume that they have identical communication capabilities, which are induced by the environment. Explicitly, the set of communication constraints is defined as

$$C = (P, \rightarrow_C), \tag{2}$$

where $\rightarrow_C \subset P \times P$ is a symmetric and reflexive relation. In other words, if two robots are at p_i and p_j, respectively, then they can directly communicate if and only if $(p_i, p_j) \in \rightarrow_C$. We assume that the communication is instantaneous, and each robot can act as a communication relay. In other words, any two robots in a connected component of C can instantaneously communicate.

We model the motion capabilities of each robot i, $i = 1, \ldots, n$ on the graph G using a transition system T_i, defined as follows:

$$T_i = (Q_i, q_{0_i}, \rightarrow_i, \Pi_i, \vDash_i),\ i = 1, \ldots, n, \tag{3}$$

where

- $Q_i = P$ is the set of states;
- $q_{0_i} \in Q_i$ is the initial location of agent i (a singleton);
- $\rightarrow_i \subset P \times P$ is a reflexive transition relation satisfying $\rightarrow_i \subseteq \rightarrow_G \cup_{j=1}^{k} (p_j, p_j)$ and $\rightarrow_i \subseteq \rightarrow_C$;
- $\Pi_i = P$;
- $\vDash_i$ is the trivial satisfaction relation $(q, \pi) \in \vDash_i$ if and only if $q = \pi$.

In other words, the motion of robot i among the vertices of G is restricted by the transition relation $\rightarrow_i$. We assume that each robot can stay at a vertex and can only move between adjacent vertices. However, it is not necessary that a robot can move between any adjacent vertices. In addition, in order to be able to avoid collision with other robots, we assume that a robot can only transit from a current vertex to a vertex with which communication is possible.

Note that the only differences between the transition systems T_i are given by their initial states and possibly by their transitions (for agents with different movement capabilities). Also, for this particular definition of a transition system, trajectories are equivalent to words. A *motion* of robot i on the graph G is such an (infinite) word produced by T_i. The occurrence of a vertex in the motion of robot i means that the vertex is visited. Infinitely many consecutive repetitions of a vertex means that the robot stays at that vertex for all future times. A *control strategy* for robot i is an algorithm that, for each state $q \in Q_i$, determines what transition the robot should take, and what the robot should communicate with the other robots in its communication range.

Definition 3: The *behavior* of the team of robots is an infinite sequence of sets $\{p_1^1, \dots, p_n^1\}, \{p_1^2, \dots, p_n^2\}, \dots$, where $p_i^j \in P$ and $p_i^j \neq p_k^j$ for $i \neq k$ and $j = 1, 2, \dots$. Each entry $\{p_1^j, \dots, p_n^j\}$, $j = 1, 2, \dots$ denotes the set of vertices of G occupied by the n robots. The first entry corresponds to the initial positions of the robots $\{p_1^1, \dots, p_n^1\} = \{q_{0_1}, \dots, q_{0_n}\}$. A set $\{p_1^j, \dots, p_n^j\}$, $j \geq 2$ is added to the sequence whenever at least one robot changes its position. An infinite number of identical entries $\{p_1^j, \dots, p_n^j\}$ is added to the sequence if the set of vertices $\{p_1^j, \dots, p_n^j\}$ is reached and never left.

According to Section II, the semantics of LTL formulas over P can be defined over team behaviors. We are now ready to formulate the main problem:

Problem 1: Given a team of n agents with motion capabilities (3) and communication constraints (2) on a graph (1), their initial non-overlapping positions, and a task specified as an LTL formula ϕ over P, find individual control strategies such that the behavior of the team satisfies the specification. In addition, there should be no collisions among the robots.

IV. Approach

Definition 4: The transition system $T_g = (Q_g, Q_{g0}, \rightarrow_g, \Pi_g, \vDash_g)$ capturing the behavior of the group of n agents is defined as:

- $Q_g \subset Q_1 \times \ldots \times Q_n$, where $(q_1, \dots, q_n) \in Q_g$ if and only if $q_i \neq q_j$ for $i \neq j$,
- $Q_{g0} = (q_{0_1}, \dots, q_{0_n})$,
- $\rightarrow_g \subset Q_g \times Q_g$ is defined by $(q, q') \in \rightarrow_g$, with $q = (q_1, \dots, q_n)$ and $q' = (q'_1, \dots, q'_n)$, if and only if (1) $(q_i, q'_i) \in \rightarrow_i$, $i = 1, \dots, n$, (2) $\forall i, j = 1, \dots, n$ with $i \neq j$, if $q'_i = q_j$, then $q'_j \neq q_i$,
- $\vDash_g \subset Q_g \times \Pi_g$ is defined by $((q_1, \dots, q_n), \pi) \in \vDash_g$ if $\pi \in \{q_1, \dots, q_n\}$.

In other words, the states of the transition system T_g capture all possible ways in which the k vertices of G can be occupied by the n robots. The configurations in which two agents overlap (occupy the same vertex) are excluded. The possible motions of the team are modelled by the transition relation $\rightarrow_g$. A transition of T_g occurs when all agents synchronously take allowed transitions (requirement (1) of Definition 4), and we exclude the case when two agents swap positions, since this could cause collision (requirement (2)). Finally, each team configuration is equipped with n predicates enumerating the locations occupied by the agents (satisfied propositions), without explicitly specifying the exact position of each agent.

In [2], we proposed a solution to Problem 1 starting from a definition of a group transition system T_g resembling Def. 4, but with the additional requirement that only communicating robots were allowed to move. The group transition system T_g was then pruned to produce a set of reduced transition systems T_r, which had the property that all their runs were implementable by the robots in a distributed manner, *i.e.,* while satisfying the communication constraints (Eqn. 2). Implementability consisted in the additional "transitivity" constraint, which required switching communicating groups to share a robot, which in return led to a token and message-passing communication protocol. Runs of T_r satisfying the LTL specification were then projected to individual runs of T_i, which were then implemented in each robot. This approach had two main disadvantages. First, only communicating robots were allowed to move at a given time. Informally, a resulting motion could be seen as starting with a connected (with respect to the communication graph (2)) group of robots. As the group moved, robots could leave or join the moving group. At all times, there was a robot (not necessarily the same) that kept track of the evolution of the team through the use of a token-based communication algorithm, and which decided the control strategy of all the robots communicating with it. Second, because of the construction of the reduced transition systems T_r , which were necessary to "keep track" of the team evolution, the method was very expensive.

V. Identical robots with no communication constraints

In this section, we assume that the robots are all identical and can all communicate with each other from all vertices of G. Let a denote the map taking an (ordered) n-tuple and producing the corresponding set, *i.e.,* $a((q_1, \dots, q_n)) = \{q_1, \dots, q_n\}$. Over the states of T_g, we define an equivalence relation $\sim_a \subset Q_g \times Q_g$ by

$$((q_1, \dots, q_n), (q'_1, \dots, q'_n)) \in \sim_a \text{ if } a((q_1, \dots, q_n)) = a((q'_1, \dots, q'_n)) \quad (4)$$

It is obvious that $\sim_a$ is a proposition preserving equivalence relation. In addition, we can prove that the transition system $T_g/_{\sim_a}$ is a bisimulation quotient (see Def. 2). The proof is omitted due to space constraints (a related proof can be found in [15]).

Note that there is a significant decrease in the number of states from T_g to $T_g/_{\sim_a}$, because $T_g/_{\sim_a}$ has only $k!/((k-n)!n!)$ states (*i.e.,* $n!$ times fewer than T_g). Since $T_g/_{\sim_a}$ is a bisimulation quotient, T_g and $T_g/_{\sim_a}$ are equivalent with respect to the satisfaction of LTL formulas (see Section II). Therefore, we can use the tool developed in [9] for producing runs of $T_g/_{\sim_a}$ satisfying formula ϕ. Then, from these we will produce runs of the original T_g, and finally runs of each T_i.

Specifically, let us denote by $r = r(1)r(2)...$ the obtained run (if it exists) of $T_g/_{\sim_a}$ starting from $r(1) = a((q_{0_1},\ldots,q_{0_n})) = \{q_{0_1},\ldots,q_{0_n}\}$. 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 $r_{T_g}(1) = (q_{0_1},\ldots,q_{0_n})$. 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 $a(p) = r(2)$. The bisimulation relation between T_g and $T_g/_{\sim_a}$ 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 (for avoiding robots spending energy in unnecessary movements). We apply this method iteratively to find a run r_{T_g} with a prefix-suffix structure as defined above. To find the robot specific runs, we simply project each state of r_{T_g} on states of T_i, using n trivial projection maps $\gamma_i : Q_g \rightarrow Q_i$, $\gamma_i((q_1,\ldots,q_n)) = q_i$, $i = 1,\ldots,n$. Note that the runs of T_i and the run r_{T_g} of T_g inherit the same prefix-suffix structure of run r of $T_g/_{\sim_a}$. To guarantee that the individual runs map to the satisfying run of T_g, communication among the robots is necessary anytime they transit from a region to another. However, the communication protocol consists in the simple exchange of a synchronization signal. An example for this case is presented in Sec. VII (Case Study 1).

VI. Independent distributed implementations

In this section, we present an approach allowing us to determine if the robots can satisfy the team LTL task by executing individual plans, without synchronization (as in Sec. V) or explicit communication through token and message passing (as in [2]). The robots should only be able to sense whether an adjacent region is occupied by another robot in order to avoid collisions. The main idea is the following. First, we find a run of T_g satisfying the LTL formula. Second, we find the set of all possible executions that can be generated by the agents while they follow the individual projections of the run of T_g, but without synchronizing (as they did in Sec. V). This will be described as the language of a transition system. Then we test if this language violates the LTL formula (by using our model checking tool [9]). If there is no word violating the formula, we conclude that synchronization is not necessary and the individual agent runs provide a solution to Problem 1. If the formula is violated by at least one word, then we decide that synchronization is necessary, and the general approach from [2] has to be involved to find a solution.

We now describe the above ideas formally. Assume that by using the tool from [9] with inputs T_g and ϕ we obtain a run r of T_g starting from the initial state and satisfying the LTL formula ϕ. If no such run exists, we conclude that Problem 1 is unfeasible (by using the approach from [2] we will also get no run, since the language of the reduced transition system T_r is included in the language of T_g). Furthermore, for simplicity of exposition, we assume that the run r has a suffix of length one (if this is not the case, the definitions for Ts_i and T_p given below become more complicated).

If the run r is of form $r = (q_1^{(1)}, q_2^{(1)},\ldots, q_n^{(1)})(q_1^{(2)}, q_2^{(2)},\ldots,q_n^{(2)}) \ldots (q_1^{(s)}, q_2^{(s)},\ldots,q_n^{(s)})\ldots$, where

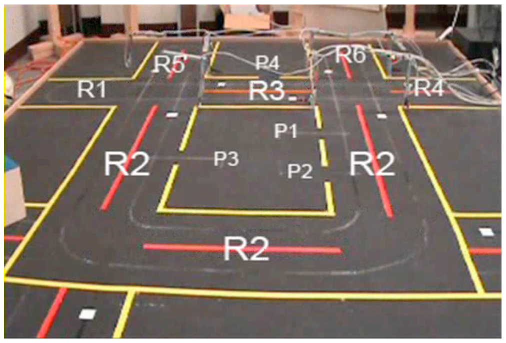

Fig. 1. The topology of the city and the road, intersection, and parking labels for the two case studies from Sec. VII.

state $(q_1^{(s)}, q_2^{(s)},\ldots,q_n^{(s)})$ is the suffix and is infinitely repeated, then the individual run of agent i is $r_i = q_i^{(1)}, q_i^{(2)},\ldots,q_i^{(s)}, q_i^{(s)},\ldots$, $i = 1,\ldots,n$. Next, in each individual run, we collapse all finite successive repetitions of the same symbol (except for the suffix, which consists of infinitely many repetitions) into a single occurrence (*i.e.*, if $\exists i \in \{1,\ldots,n\}, j \in \{1,\ldots,s-1\}$ such that $q_i^{(j)} = q_i^{(j+1)}$, then we remove $q_i^{(j+1)}$ from r_i, and repeat until no such i and j can be found). This is motivated by the fact that the agents do not synchronize while moving, and the removed repetitions correspond to synchronizations in run r modeling an agent that waits in the same location until other agents reach some specific locations. For simplicity, let each individual run (without successive repetitions) be denoted by $r_i = q_{i_1}, q_{i_2},\ldots,q_{i_{s_i}},\ldots$, where $q_{i_{s_i}}$ is infinitely repeated (the runs might now have prefixes of different lengths, because of the performed collapsing).

Next we construct the set of all possible n-tuples that can be generated by the team of agents, while each agent i individually (without synchronization) follows run r_i. For this, we first construct a small transition system Ts_i for each agent, and then we construct a special kind of product automaton of all these transition systems, with the guarantee that the set of words generated by the obtained product will be exactly the desired set of tuples.

Definition 5: Transition system $Ts_i = (Qs_i, Qs_{i0}, \rightarrow_{s_i}, \Pi_{s_i}, \vDash_{s_i})$, $i = 1,\ldots,n$, is defined as:

- $Qs_i = \{q_{i_1}, q_{i_2},\ldots,q_{i_{s_i}}\} \subset P$ is the set of states,
- $Qs_{i0} = q_{i_1} = q_{0_i}$ is the initial location of agent i,
- $\rightarrow_{s_i} \subset Qs_i \times Qs_i$ is defined by $(q_{i_j}, q_{i_j}) \in \rightarrow_{s_i}$, $\forall j \in \{1,\ldots,s_i\}$, and $(q_{i_j}, q_{i_{j+1}}) \in \rightarrow_{s_i}$, $\forall j \in \{1,\ldots,s_i - 1\}$,
- $\Pi_{s_i} = P$,
- $\vDash_{s_i} \subset Qs_i \times \Pi_{s_i}$ is the trivial satisfaction relation $(q,\pi) \in \vDash_{s_i}$ if and only if $q = \pi$.

Each transition system Ts_i, $i = 1,\ldots,n$, corresponds to agent i following run r_i (transitions exist only between successive states of r_i, together with self-loops in any state). The self-loops are included to correctly create the product transition system from Definition 6. Informally, agent i takes a self transition if it is slower than other moving agents - this is necessary since there is no synchronization.

Definition 6: The product of Ts_i, $i = 1,\ldots,n$ is defined as $T_p = (Qp, Qp_0, \rightarrow_p, \Pi_p, \models_p)$, where:

- $Qp \subset Qs_1 \times \ldots \times Qs_n$ is defined by $(q_1, \ldots, q_n) \in Qp$ if and only if $q_i \neq q_j$ for $i \neq j$,
- $Qp_0 = (q_{1_1}, \ldots, q_{n_1})$,
- $\rightarrow_p \subset Qp \times Qp$ is defined by $(q, q') \in \rightarrow_p$, with $q = (q_1, \ldots, q_n)$ and $q' = (q'_1, \ldots, q'_n)$, if and only if (1) $(q_i, q'_i) \in \rightarrow_{s_i}$, $i = 1, \ldots, n$, and (2) $q \neq q'$ or $q = q' = (q_{1_{s_1}}, \ldots, q_{n_{s_n}})$,
- $\Pi_p = P$,
- $\models_p \subset Qp \times \Pi_p$ is defined by $((q_1, \ldots, q_n), \pi) \in \models_p$ if $\pi \in \{q_1, \ldots, q_n\}$.

Transition system T_p captures all possible transitions that can appear in any Ts_i, *i.e.,* all possible configurations that can be attained by the team, while each robot follows its run on its own (without synchronization). This fact is guaranteed by requirement (1) from the transition relation of T_p. Requirement (2) from the transition relation of T_p eliminates self-loops in all states of T_p, except for the state corresponding to suffixes of runs r_i. This is because we use LTL_{-X} formulas and there is no need to capture finitely many successive repetitions of the same tuple in the generated word.

Note that, to achieve collision avoidance, in the construction of Qp in Definition 6, we exclude the situations in which two or more agents occupy the same location. This requires all agents to be able to sense if the next (adjacent) location is free before moving from the current location. If the next location is occupied, the agent needs to wait until the location becomes free. This sensing capacity can, of course, be implemented if the agents can communicate when in adjacent regions. Since there is no synchronization when different agents change locations, the probability of two agents taking transitions to exactly the same location at exactly the same time is zero, and thus the described simple protocol guarantees collision avoidance. Such a protocol was not necessary in the approaches from [2] and Sec. V, since collision avoidance was guaranteed by the global construction of the team run. Finally, note that the "move-only-when-communicate" restriction from the run r of T_g disappears as a result of collapsing identical successive repetitions from its projections to individual agents.

By construction, the language of T_p contains all and only the words that can be generated by the unsynchronized team movement when each agent i follows its run r_i, $i = 1, \ldots, n$. If the language of T_p satisfies formula ϕ, then we conclude that the unsynchronized movement (with collision avoidance, as mentioned above) is a solution to Problem 1. If this is not the case, we conclude that synchronization is necessary, and therefore the approach developed in [2] is necessary. An illustrative example is presented in Sec. VII (Case Study 2).

VII. Deployment of teams of autonomous cars in the Robotic Urban-Like Environment (RULE)

Our urban environment is an easily reconfigurable collection of roads, intersections, traffic lights, and parking spaces. An example is given in Fig. 1. In our previous work [3], we showed that a simple set of composition rules allows for the construction of any Manhattan-type urban environment from a small number of "modules". The "cars" in our city are Khepera III robots (see [3] or `hyness.bu.edu/rule/` for implementation details). Robot deployment is achieved through a user-friendly graphical interface.

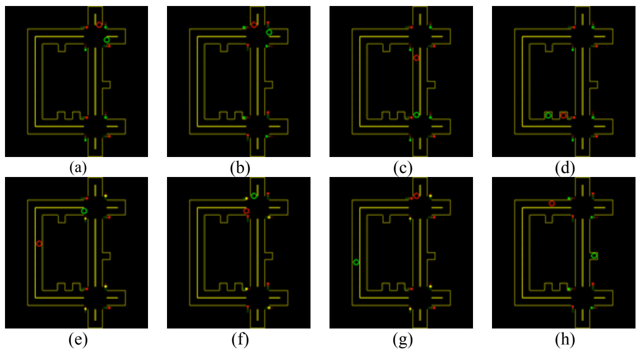

Fig. 2. Snapshots from our RULE simulator showing the car motions satisfying the specification given by the LTL formula in Eqn. (5). Initially, C_1 (red) is on road R_1 and C_2 (green) is on road R_5. Both cars cross intersection I_1 and drive through R_3 ((b), (c)) and R_2 before parking at P_1 and P_2 ((d)) (part $(P_1 \wedge P_2) \vee (P_2 \wedge P_3)$ of the specification). They both leave and take R_2 ((e)), before C_2 goes back on R_1 ((f)), and then back on R_2 ((g)), and through R_3 goes to P_4 and parks there (h)) (part $\Diamond\Box P_4$ of the specification). Car C_1 keeps on moving on the route R_1 - R_2 - R_3, therefore satisfying the $\Box\Diamond(R_1 \wedge \Diamond R_2)$ part of specification ϕ_3 (the suffix configurations are not shown).

Formally, an urban environment is a graph (Eqn. (1)), where P is a set of labels for roads, intersections, and parking spaces, and $\rightarrow_G$ shows how these features are connected. The motion of a car in the urban environment can be described as a transition system T_i (Eqn. (3) and the accompanying text). Note that, in reality, the transition system describing the motion of a robot has inputs, *i.e.,* each transition is enabled by a symbol from an input set. These inputs can be generated by the environment (*e.g.,* the color of a traffic light) or by the robot itself (*e.g.,* turn left in an intersection, park). However, since (by construction) this transition system with inputs is deterministic, *i.e.,* at each state, an input determines a unique transition to another state, the inputs can be dropped and the transition system takes the form in Eqn. (3). In [3], we show that under reasonable "liveness" assumption about environmental events, a transition system of the type (3) captures the motion of each robot correctly.

We present two case studies illustrating the two deployment methods described in Secs. V and VI. Both case studies refer to the city topology from Fig. 1. We assume two robots (cars) are available, which are labeled by C_1 and C_2.

Case Study 1: Consider the following specification:

"*Park at P_1 and P_2 or at P_2 and P_3. Then keep on driving in between R_1 and R_2. Park at P_4 and stay there for all future times.*"

The specification translates to the following LTL formula:

$$\Diamond((P_1 \wedge P_2) \vee (P_2 \wedge P_3)) \wedge \Box\Diamond(R_1 \wedge \Diamond R_2) \wedge \Diamond\Box P_4 \quad (5)$$

By using the method described in Sec. VI, we find that an independent distributed solution is not possible. We therefore

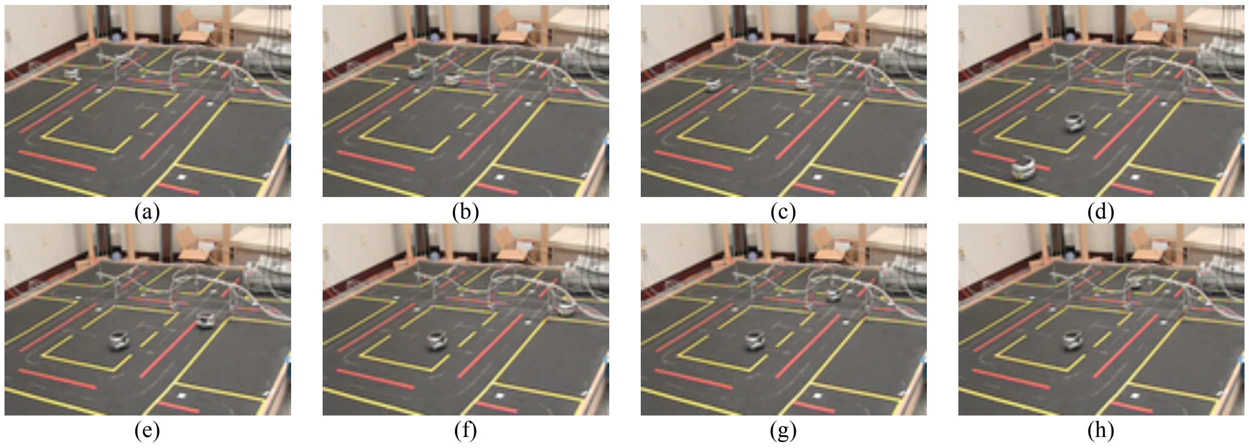

Fig. 3. The car motions satisfying the specification from Eqn. (6). Initially, C_1 is on road R_1 and C_2 is on road R_5. A simple inspection of all snapshots shows that R_6 is always avoided (the $\neg\Box R_6$ part of the specification). Car C_2 visits both R_2 ((c), (d), (e)) and R_4 ((f)), which corresponds to $\Diamond R_2 \wedge \Diamond R_4$. Car C_1 parks in P_2 ((e) - (h)) and then car C_2 parks in P_4 ((h)), which guarantee that $\Diamond(P_2 \wedge P_4)$ is satisfied.

allow the robots to communicate for all times (through the wireless network) and apply the method from Sec. V, which is possible since the cars are identical. The produced motion of the team (in the RULE simulator) is described in Fig. 2 and the corresponding caption. The movie of the corresponding experiment is available at `hyness.bu.edu/rule`.

Case Study 2: Consider the following specification:

"*Never use road R_6. Visit roads R_2 and R_4 - the order does not matter. Parking spaces P_2 and P_4 should be both occupied at some time in the future.*"

The specification translates to the following LTL formula:

$$\Box\neg R_6 \wedge \Diamond R_2 \wedge \Diamond R_4 \wedge \Diamond(P_2 \wedge P_4) \qquad (6)$$

By applying the method described in Sec. VI, we find that an independent distributed solution is possible. The produced motion of the team (in the RULE experimental setup) is described in Fig. 3 and the corresponding caption.

VIII. Conclusion and Discussion

We presented some results on automatic deployment of robotic teams from LTL specifications over features of interest in a partitioned environment. This work can be seen as extension of our previous results from [2]. In particular, we show that if we allow identical robots to communicate for all times, we can decrease the computational complexity. In addition, we develop an algorithm that can test when a global specification can be executed by the team with no need for communication, except for inter-robot collision detection. We illustrate the two methods through experimental trials in our Robotic Urban-Like Environment (RULE).

References

[1] S. M. LaValle, *Planning algorithms*. Cambridge, UK: Cambridge University Press, 2006.

[2] M. Kloetzer and C. Belta, "Distributed implementation of global temporal logic motion specifications," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, 2008.

[3] M. Lahijanian, M. Kloetzer, S. Itani, C. Belta, and S. B. Andersson, "Automatic deployment of autonomous cars in a robotic urban-like environment (rule)," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[4] 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.

[5] 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.

[6] 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*, New Orleans, LA, April 2004, pp. 4417–4422.

[7] H. K. Gazit, G. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *IEEE Conference on Robotics and Automation*, Rome, Italy, 2007.

[8] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: a temporal logic approach," in *Proceedings of the 2005 IEEE Conference on Decision and Control*, Seville, Spain, December 2005.

[9] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[10] M. Mukund, "From global specifications to distributed implementations," in *Synthesis and control of discrete event systems*. Kluwer, 2002, pp. 19–34.

[11] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science: Formal Models and Semantics*, J. van Leeuwen, Ed. North-Holland Pub. Co./MIT Press, 1990, vol. B, pp. 995–1072.

[12] E. M. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[13] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "Nusmv version 2: An opensource tool for symbolic model checking," in *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, ser. LNCS, vol. 2404. Copenhagen, Denmark: Springer, July 2002.

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

[15] M. Kloetzer and C. Belta, "LTL planning for groups of robots," in *IEEE International Conference on Networking, Sensing, and Control*, Ft. Lauderdale, FL, 2006.