The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

A Hierarchical Approach to Automatic Deployment of Robotic Teams with Communication Constraints

Yushan Chen, Samuel Birch, Alin Stefanescu, and Calin Belta

***Abstract*— We consider the following problem: GIVEN (1) a set of service requests occurring at known locations in an environment, (2) a set of temporal and logical constraints on how the requests need to be serviced, (3) a team of robots and their capacities to service the requests individually or through collaboration, FIND robot control and communication strategies guaranteeing the correct servicing of the requests. Our approach is hierarchical. At the top level, we check whether the specification, which is a regular expression over the requests, is distributable among the robots given their service and cooperation capabilities; if the answer is positive, we generate individual specifications in the form of finite state automata, and interaction rules in the form of synchronizations on shared requests. At the bottom level, we check whether the local specifications and the synchronizations can be implemented given the motion and communication constraints of the robots; if the answer is positive, we generate robot motion and service plans, which are then mapped to control and communication strategies. We illustrate the method with experimental and simulation results.**

I. Introduction

While the "classical" motion planning problem is given simply as "go from A to B" [1], [2], missions typically require much more complicated tasks. Consider, for example, an emergency response scenario in the miniature Robotic Urban-Like Environment (RULE) shown in Fig. 1 and Fig. 2, where a firefighter autonomous vehicle needs to get to location P_3, where there is fire. Before getting there, the vehicle needs to get water, which is available on road R_1 or R_2. The vehicle is not allowed to cross intersection I_3, and it is required to obey the traffic rules at all times. Such a "rich" specification cannot be trivially converted to a sequence of "go from A to B" primitives. When several vehicles are available, the problem becomes even more interesting and challenging. Assume that several service requests occur at different locations in the city, and they need to be serviced subject to some temporal and logical constraints. Some of these requests can be serviced by one (possibly specific) vehicle, while others require the collaboration of two or more (possibly specific) vehicles. With reference to the same urban emergency response scenario, assume that the task is to help an injured person located at P_1. A medical emergency vehicle and a traffic escort vehicle need to eventually get there. In order to provide medical support, the emergency vehicle needs to pick medical supplies that can be found at P_3 and P_4 before going to P_1. Can we generate provably-correct individual control and communication strategies from such rich, global specifications? This is the problem that we address in this paper.

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

Y. Chen is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA, E-mail: yushanc@bu.edu

S. Birch is with the Boston University Academy, Boston, MA, USA, E-mail: sbirch@bu.edu

A. Stefanescu is with SAP Research, Darmstadt, Germany, E-mail: alin.stefanescu@sap.com

C. Belta is with the Department of Mechanical Engineering, Division of Systems Engineering, and the Bioinformatics Graduate Program, Boston University, MA, USA, E-mail: cbelta@bu.edu

Y. Chen is the corresponding author

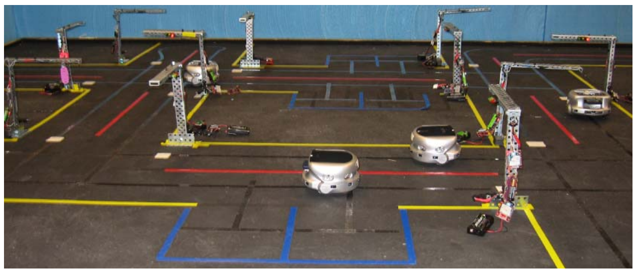
Fig. 1: Robotic Urban-Like Environment (RULE). Khepera III car-like robots move autonomously on streets while staying in their lanes, obeying traffic rules, and avoiding collisions.

It has recently been suggested that temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [3], can be used as expressive specification languages for mobile robots [4], [5], [6], and model checking algorithms can be adapted for motion planning and controller synthesis from such specifications. Some related works show that such techniques can be extended to multi-agent systems through the use of parallel composition [7], [8] or reactive games [9]. However, such bottom-up approaches are expensive and can lead to state-space explosion even for relatively simple problems. As a result, there is a need for top-down approaches, in which "rich", global specifications can be decomposed into local (individual) specifications, which can then be used to automatically synthesize robot control and communication strategies.

In this paper, we draw inspiration from the area of distributed formal synthesis [10] to develop such a top-down approach. We consider a team of robots that can move among the regions of a partitioned environment, and which have known capabilities of servicing a set of requests that can occur in the regions of the partition. Some of these requests can be serviced by a robot individually, while some require the cooperation of groups of robots. We present an approach that allows for the fully automatic synthesis of robot control and communication strategies from a task specification given as a regular expression over the set of requests. For simplicity of presentation, we model the (partitioned) environment as a graph and the robots as agents that can move between adjacent vertices and can communicate only when at particular vertices. This framework is quite general and can be used in conjunction with cell decomposition motion planning techniques [1]. In particular, by using feedback controllers for facet reachability polytopes [11], [12], this scenario can be immediately extended to robots with continuous dynamics moving in environments with polytopic partitions. For illustration, we apply the developed computation framework to our Robotic Urban-Like Environment (RULE) (Fig. 1).

II. Preliminaries

For a set Σ, we use $|\Sigma|$ to denote its cardinality. Given a set Σ, a collection of subsets $\Delta = \{\Sigma_i \subseteq \Sigma, i \in I\}$ is called a *distribution* over Σ if $\cup_{i\in I}\Sigma_i = \Sigma$, where I is an index set. A *word* w over a set Σ is a sequence of symbols from Σ. We use Σ^* to denote the set of all finite words over Σ. A *language* is a set of words.

Definition 1: A transition system (TS) is a tuple $T = (S, s_0, \rightarrow, \Pi, \vDash)$, where S is a finite set of states, $s_0 \in S$ is the initial state, and $\rightarrow \subseteq S \times S$ is the transition relation, Π is a finite set of atomic propositions (observations), and $\vDash \subseteq Q \times \Pi$ is a satisfaction relation.

A transition $(s, s') \in \rightarrow$ is also denoted by $s \rightarrow s'$. A *run* of T is a sequence $r(1)r(2)r(3)\ldots r(n)$ with the property that $r(1) = s_0$, $r(i) \in S$, and $(r(i), r(i+1)) \in \rightarrow, \forall i \geqslant 1$.

Definition 2: A finite state automaton (FSA) is a tuple $A = (Q, q_0, \Sigma, \rightarrow, F)$, where Q is the set of states, $q_0 \in Q$ is the initial state, Σ is the set (alphabet) of actions, $\rightarrow \in Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states.

We also write $q \xrightarrow{\sigma} q'$ to denote $(q, \sigma, q') \in \rightarrow$. In this paper, we assume that all the FSAs are deterministic. The *language* accepted by an FSA A, denoted by $\mathcal{L}(A)$, is the set of all finite words accepted by A. The language accepted by an FSA is called a *regular language*, which can be concisely represented by a *regular expression* (RE). Given an RE, an FSA accepting the language satisfying the RE can be constructed by using an off-the-shelf tool, such as JFLAP [13].

Definition 3: The synchronous product (SP) of n FSAs $A_i = (Q_i, q_{0_i}, \Sigma_i, \rightarrow_i, F_i)$, denoted by $\|_{i=1}^{n} A_i$, is an FSA $(Q, q_0, \Sigma, \rightarrow, F)$, where $Q = Q_1 \times Q_2 \times \ldots \times Q_n$, $q_0 = (q_{0_1}, q_{0_2}, \ldots, q_{0_n})$, $\Sigma = \cup_{i=1}^{n}\Sigma_i$, and $F = F_1 \times F_2 \times \ldots \times F_n$. The transition relation $\rightarrow \subseteq Q \times \Sigma \times Q$ is defined by $q \xrightarrow{\sigma} q'$ iff $\forall\ i \in I_\sigma : q[i] \xrightarrow{\sigma}_i q'[i]$ and $\forall\ i \notin I_\sigma : q[i] = q'[i]$, where $q[i]$ denotes the ith component of q and $I_\sigma = \{i \in \{1, \ldots, n\} \mid \sigma \in \Sigma_i\}$.

For a word $w \in \Sigma^*$ and a subset $S \subseteq \Sigma$, we denote by $w\upharpoonright_S$ the *projection* of w onto S, which is obtained by erasing all actions $\sigma \notin S$ in w. For a language $L \subseteq \Sigma^*$ and a subset $S \subseteq \Sigma$, we denote by $L\upharpoonright_S$ the projection of L onto S, which is given by $L\upharpoonright_S := \{w\upharpoonright_S \mid w \in L\}$. Starting from the observation that the projection of a regular language is a regular language, the projection of an FSA A on a subset $S \subseteq \Sigma$ is an FSA $A\upharpoonright_S$, which can be constructed from A through ε-closure and optional minimization ([14]).

Definition 4: Given a distribution Δ, the product of a set of languages L_i over Σ_i is denoted by $\|_{i\in I} L_i$ and defined as $\|_{i\in I} L_i := \{w \in \Sigma^* \mid w\upharpoonright_{\Sigma_i} \in L_i, \forall i \in I\}$. A *product language* over Δ is a language L such that $L = \|_{i\in I} L_i$, where $L_i = L\upharpoonright_{\Sigma_i}, \forall i \in I$.

Definition 5: (Adapted from [5]) The product automaton (PA) $P = T \times A$ between the transition system $T = (S, s_0, \rightarrow, \Sigma, \vDash)$ and the FSA $A = (Q, q_0, \Sigma, \rightarrow_A, F)$ is defined as the tuple $P = (Q_P, q_{0_P}, \Sigma, \rightarrow_P, F_P)$, where $Q_P = S \times Q$, $q_{0_P} = (s_0, q_0)$ is the initial state, and $F_P = S \times F$ is the set of accepting states. The transition relation $\rightarrow_P \subseteq Q_P \times \Sigma \times Q_P$ is defined as $(s, q) \xrightarrow{\sigma}_P (s', q')$ if and only if $s \rightarrow s', (s, \sigma) \in \vDash$ and $q \xrightarrow{\sigma}_A q'$,

It is easy to see that a word $w \in \Sigma^*$ is accepted by P ($w \in \mathcal{L}(P)$) only if w is accepted by A ($w \in \mathcal{L}(A)$).

III. Problem Formulation

Let

$$\mathcal{E} = (V, \rightarrow_\mathcal{E}) \tag{1}$$

be an environment graph, where V is the set of vertices and $\rightarrow_\mathcal{E} \subseteq V \times V$ is a relation modeling the set of edges. For example, $\mathcal{E}$ can be the quotient graph of a partitioned environment, where V is a set of labels for the regions in the partition, and $\rightarrow_\mathcal{E}$ is the corresponding adjacency relation. In particular, V can be a set of labels for the roads, intersections, and parking spaces in an urban-like environment and $\rightarrow_\mathcal{E}$ can show how these are connected (see Sec. VII). Assume we have a team of robots (moving agents) $\mathcal{A}_i$, $i \in I$, whose motions are restricted by $\mathcal{E}$, where I is a set of robot labels. We assume that they have identical communication capabilities. The set of communication constraints is defined as an undirected graph

$$\mathbb{C} = (V, E_\mathbb{C}), \tag{2}$$

where $E_\mathbb{C} \subseteq V \times V$ is a symmetric relation modeling a set of communication edges. In other words, $(v_i, v_j) \in E_\mathbb{C}$ if and only if a robot located at v_i can directly communicate with another robot located at v_j.

Let Σ be a set of the requests that can occur at the vertices of $\mathcal{E}$. The locations of the requests are modeled as a relation $loc \subseteq V \times \Sigma$ with the following meaning: $(v, \sigma) \in loc$ means that service request σ occurs at vertex v.

We model the capacity of the robots to service requests as a distribution Δ, where $\cup_{i\in I}\Sigma_i = \Sigma$. For a given request $\sigma \in \Sigma$, we define $I_\sigma = \{i \in I \,|\, \sigma \in \Sigma_i\}$, *i.e.,* I_σ is the set of labels of all the robots that can service request σ. The semantics of this distribution is defined as follows. For an arbitrary request σ, if $|I_\sigma| = 1$ (*i.e.,* there is only one robot that owns it), the robot can (and should) service the request by itself, independent of the other robots. This kind of request is called *independent request.* If $|I_\sigma| > 1$, all the robots $\mathcal{A}_i$ with $i \in I_\sigma$ must service the request simultaneously. This kind of request is called *shared request.*

The meaning of servicing a request σ by the robotic team depends on whether σ is an independent or a shared request. If $|I_\sigma| = 1$, then the request is serviced if the only robot that owns σ visits one vertex v that satisfies $(v, \sigma) \in loc$. If $|I_\sigma| > 1$, then all the robots $\mathcal{A}_i$ with $i \in I_\sigma$ must service the request simultaneously. Therefore, all the robots that own σ should be in the same connected component of the communication graph $\mathbb{C}$ to service σ (*i.e.,* they need to communicate to service σ together). Depending on the topology of $\mathcal{E}$ and $\mathbb{C}$, the cardinality of I_σ, the motion capabilities of the robots (see below), and the location(s) of σ, there are multiple ways in which σ can be serviced. Choosing a (optimal) solution is one of the fundamental problems in this paper, which is treated in Sec. VI.

We model the motion capabilities of each agent $\mathcal{A}_i$, $i \in I$ on the environment graph $\mathcal{E}$ using a transition system:

$$T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \vDash_i),\ i \in I, \tag{3}$$

where $v_{0_i} \in V$ is the initial location of $\mathcal{A}_i$, $\rightarrow_i \subseteq V \times V$ is a relation satisfying $\rightarrow_i \subseteq \rightarrow_{\mathcal{E}} \cup_{v \in V} \{(v, v)\}$, $\Pi = \Sigma \cup \{\epsilon\}$, ϵ is the empty request, and $\vDash_i \subseteq V \times \Pi$ is a relation where $(v, \epsilon) \in \vDash_i, \forall v \in V$ and $(v, \sigma) \in \vDash_i$, $\sigma \in \Pi$ iff $(v, \sigma) \in loc$.

In other words, the motion of robot $\mathcal{A}_i$ is restricted by the relation $\rightarrow_i$, which captures motion (actuation) constraints in addition to $\rightarrow_{\mathcal{E}}$. The locations of the requests in the environment are captured by relation $\vDash_i$. As it will become clear later, each vertex satisfying ϵ captures that a robot can pass through a vertex without servicing any request.

Definition 6: A motion and service plan (MS plan) for the robot $\mathcal{A}_i$, $i \in I$ is a word $ms_i \in (V \cup \Sigma_i)^*$ that satisfies the following conditions: (1) $ms_i(1) = v_{0_i}$, (2) if $ms_i(j) \in \Sigma_i$, then $ms_i(j-1) \in V$ and $(ms_i(j-1), ms_i(j)) \in loc, \forall j > 1$ and (3) if $ms_i(j) \in V$ and $ms_i(j-1) \in V$, then $(ms_i(j-1), ms_i(j)) \in \rightarrow_i, \forall j > 1$.

The semantics of an MS plan is as follows. A vertex entry $ms_i(j) \in V$ means that the corresponding vertex should be visited. A request entry $ms_i(j) \in \Sigma_i$ means that robot $\mathcal{A}_i$ should service request $ms_i(j)$ at vertex $ms_i(j-1)$. Before servicing request $ms_i(j) \in \Sigma_i$, the robot needs to arrive at vertex $ms_i(j-1) \in V$, where $ms_i(j)$ occurs (*i.e.* $(ms_i(j-1), ms_i(j)) \in loc$). *Wait-and-leave* protocol need to be used for synchronization on shared requests. A shared request entry $ms_i(j) \in \Sigma_i$ where $|I_{ms_i(j)}| > 1$ following a vertex entry $ms_i(j-1) \in V$ triggers a *wait-and-leave* protocol: while at $ms_i(j-1)$, robot $\mathcal{A}_i$ broadcasts request $ms_i(j)$ and listens for broadcasts of $ms_i(j)$ from all agents $\mathcal{A}_j$, $j \in I_{ms_i(j)} \setminus \{i\}$. When all these are received, $\mathcal{A}_i$ moves to the next vertex specified in $ms_i(j+1)$. We say that a word $w_i \in {\Sigma_i}^*$ can be *implemented* by the robot $\mathcal{A}_i$ if there exists a MS plan ms_i such that $ms_i \upharpoonright_{\Sigma_i} = w_i$.

Given a set of MS plans $\{ms_i, i \in I\}$ for the robotic team, there may exist many possible sequences of requests serviced by the team due to parallel executions of individual robots (we do not assume that we know the time it takes for each robot to service requests). For a given set of MS plans, we denote

$$L_{MS}^{team}(\{ms_i, i \in I\}) := \|_{i \in I} \{ms_i \upharpoonright_{\Sigma_i}\}, \tag{4}$$

(Def. (4)) as the set of all possible sequences of requests (*i.e.* language over Σ) serviced by the team of robots $\mathcal{A}_i$ while they follow their individual MS plans ms_i. For simplicity of notations, we usually denote $L_{MS}^{team}(\{ms_i, i \in I\})$ as L_{MS}^{team} when there is no ambiguity. We say that the motion of the team with MS plans $\{ms_i, i \in I\}$ satisfies a specification given as an RE ϕ over Σ if $L_{MS}^{team} \neq \emptyset$ (see Remark 1) and all words in L_{MS}^{team} satisfy ϕ (*i.e.* $L_{MS}^{team} \subseteq \mathcal{L}(A)$, where A is an FSA accepting only the words satisfying ϕ).

Remark 1: A deadlock may occur when $L_{MS}^{team} = \emptyset$. By the definition of product of languages, this will happen when there exists no word $w \in \Sigma^*$ such that $w \upharpoonright_{\Sigma_i} = ms_i \upharpoonright_{\Sigma_i}$, $\forall i \in I$. In practice, the case that $L_{MS}^{team} = \emptyset$ corresponds to a scenario where one (or more) agent waits indefinitely for other agents to service a request σ that is shared among these agents. For example, if a robot $\mathcal{A}_i$ that shares a request σ with another robot $\mathcal{A}_j$ chooses to service σ while $\mathcal{A}_j$ chooses not to service σ, then $\mathcal{A}_i$ will be stuck in a "deadlock" state and wait indefinitely. When such a deadlock scenario occurs, the motion of the team does not satisfy the specification.

Given a MS plan, a robot generates a *control and communication strategy*, which is a finite sequence of control primitives, interrupts, and communication protocols. To guarantee the uniqueness of this strategy, we assume that each robot is equipped with a set of motion primitives (feedback controllers), such that the selection of a motion primitive at a vertex uniquely determines the next vertex, given that the robot is properly initialized and the history of visited vertices is known. In other words, the robot $\mathcal{A}_i$, $i \in I$ can follow any run of T_i. We are now ready to formulate the main problem:

Problem 1: Given a team of robots $\mathcal{A}_i$, $i \in I$ with motion capabilities T_i and communication constraints $\mathbb{C}$ on a graph $\mathcal{E}$, a set of service requests Σ, a task specification ϕ in the form of an RE over Σ, a distribution Δ of requests over the robots, find a set of MS plans $\{ms_i,\ i \in I\}$ such that the corresponding motion of the team satisfies ϕ.

IV. Synthesis of individual strategies

A. From global to local specifications

Our approach proceeds with the conversion of the RE ϕ over Σ to a global FSA $A = (Q, q_0, \Sigma, \rightarrow, F)$ with input alphabet Σ, which accepts exactly the language over Σ that satisfies ϕ. To find individual (local) task specifications for each robot according to the distribution Δ, we draw inspiration from the emerging area of distributed formal synthesis [10]. Our goal is to construct a set of local specifications in the form of FSAs over Σ_i, $i \in I$, such that when they synchronize, they are equivalent to the global specification represented by A. Specifically, we consider the following problem:

Problem 2: Given a global FSA $A = (Q, q_0, \Sigma, \rightarrow, F)$ and a distribution $\Delta = \{\Sigma_i, i \in I\}$ of the requests over the robots, construct a set of local FSAs $\{A_i = (Q_i, q_{0i}, \Sigma_i, \rightarrow_i, F_i), i \in I\}$ such that $\mathcal{L}(A) = \mathcal{L}(\|_{i \in I} A_i)$.

Note that Prob.2 has a solution iff $\mathcal{L}(A)$ is a product language [14]. Moreover, using the characterization of product languages (Def. 4), the solution to Prob. 2 is a set of local

FSAs $\{A_i, i \in I\}$ such that $\mathcal{L}(A) \upharpoonright_{\Sigma_i}$. We can check if $\mathcal{L}(A)$ is a product language by applying an extension of the approach proposed in [14], which reduces the problem to the (non-)reachability problem for 1-safe Petri nets [15].

If a solution to Prob. 2 (*i.e.* $\{A_i, i \in I\}$) exists, then for any set of words $\{w_i \in \mathcal{L}(A_i), i \in I\}$, we have $\|_{i\in I} \{w_i\} \subseteq \mathcal{L}(A)$. Therefore, these sets of words $\{w_i, i \in I\}$ can be used to generate a solution to Prob. 1. However, we also need to ensure that (1) $\{w_i, i \in I\}$ can be implemented by the robots $\mathcal{A}_i$, (2) the communication constraint $\mathbb{C}$ is satisfied by the team of robots (*i.e.* the robots need to communicate to service a shared request), and (3) the set of MS plans $\{ms_i, i \in I\}$ generated from $\{w_i, i \in I\}$ satisfies $L_{MS}^{team} \neq \emptyset$ (*i.e.* $\|_{i\in I} \{w_i\} \neq \emptyset$) (see Remark 1). We show how to satisfy these requirements in Secs.V and VI.

V. Construction and Reduction of Product Automaton P_i

To find the solution to Prob. 1, we need to construct a product automaton P_i that captures all the words $w_i \in \mathcal{L}(A_i)$ that can be implemented by the robot $\mathcal{A}_i$ (*i.e.* w_i can be used to obtain a MS plan ms_i such that $ms_i \upharpoonright_{\Sigma_i} = w_i$). We present an approach inspired by model checking techniques [3], which takes as input a local FSA A_i, a transition system T_i (Eqn. 3), and returns a product automaton P_i. Then, a reduced version R_i of P_i is constructed, which will be further used in the next section to generate a set of MS plans such that the corresponding motion of the team satisfies the communication constraints and it is deadlock-free (in the sense described in the previous section).

To construct P_i, we first construct a new FSA $\widehat{A}_i$ by adding a new action ϵ and self-transitions (q, ϵ, q) to each state $q \in Q_i$. It is important to note that these self transitions do not affect the semantics of FSA A_i since ϵ means no request is served by robot $\mathcal{A}_i$. Given a word w accepted by the new FSA, we can obtain a word $w_A = w \upharpoonright_{\Sigma_i}$ accepted by A_i. The FSA $\widehat{A}_i$, $i \in I$, can now be defined as:

$$\widehat{A}_i = (\widehat{Q}_i, \widehat{q}_{0_i}, \widehat{\Sigma}_i, \widehat{\rightarrow}_{A_i}, \widehat{F}_i), \tag{5}$$

where $\widehat{Q}_i = Q_i$, $\widehat{q}_{0_i} = q_{0_i}$, $\widehat{\Sigma}_i = \Sigma_i \cup \{\epsilon\}$, $\widehat{\rightarrow}_{A_i} = \rightarrow_{A_i} \cup_{q \in Q_i} (q, \epsilon, q)$, and $\widehat{F}_i = F_i$.

By noting that the set of inputs $\widehat{\Sigma}_i$ of $\widehat{A}_i$ is a subset of the observations Π of T_i, we can construct the product automaton $P_i = T_i \times \widehat{A}_i$. A transition $(v, q) \xrightarrow{\sigma}_P (v', q')$ of P_i exists iff $(v, v') \in \rightarrow_i$ and request σ occurs at vertex v. Transitions with input ϵ mean that a robot is moving from v to v' (v may be equal to v') without servicing any request. According to Def. 5 and the accompanying text, P_i will accept exactly the runs of T_i satisfy the words accepted by $\widehat{A}_i$. In particular, this implies that for i, $i \in I$, there exists a run of T_i satisfying a word $w_i \in \mathcal{L}(A_i)$ iff P_i has a nonempty language. Moreover, this run of T_i and the corresponding word w_i can be used to generate a MS plan for $\mathcal{A}_i$.

Next, we introduce a reduced version R_i of P_i constructed above, whose accepted runs can always be refined to obtain runs accepted by P_i. The main idea behind the reduction is that individual requests can (and should) be serviced by a robot on its own; only shared requests require communication. Each R_i will be composed of "collapsed" minimal paths (with respect to possible costs associated to the transitions of T_i) corresponding to individual requests, and transitions relating to shared requests, which have to satisfy the communication constraints. In Sec. VI, we approach this problem through a synchronization process. Mapping each P_i to an R_i keeps the synchronous product at a manageable size.

To construct the reduced automaton R_i for robot $\mathcal{A}_i$, we keep the initial state of P_i and all the states and transitions that are related to shared requests. We say that states q_{P_i}, q'_{P_i} and transition $(q_{P_i}, \sigma, q'_{P_i})$ are related to shared requests if $(q_{P_i}, \sigma, q'_{P_i}) \in \rightarrow_{P_i}$ and $|I_\sigma| > 1$. To preserve the connectivity of P_i in R_i, we check if paths with only independent requests exist between remaining states. If more than one such path exists, we can use Dijkstra's algorithm to find the shortest path. If such a path exists, we say there is a transition between the corresponding states q_{R_i} and q'_{R_i} in R_i, and the input of this transition is denoted as $path(q_{R_i}, q'_{R_i})$. We denote the set of all states and the set of all transitions related to shared requests by $Q^S_{P_i}$ and $\rightarrow^S_{P_i}$. Moreover, we can reduce the number of accepting states and generate a subset of accepting states denoted as F'_{P_i} by applying the following two steps: (1) for each remaining state which is not an accepting state, we check if there exist accepting states that can be reached from this state by paths including only independent requests, and (2) if more than one paths exist, we choose the path with the lowest cost and the corresponding accepting state and add it to F'_{P_i}. Formally, the reduced product automaton R_i for robot $\mathcal{A}_i$, $i \in I$ is defined as:

$$R_i = (Q_{R_i}, q_{0_{R_i}}, \Sigma_{R_i}, \rightarrow_{R_i}, F_{R_i}) \tag{6}$$

where $Q_{R_i} = \{q_{0_{P_i}}\} \cup F'_{P_i} \cup Q^S_{P_i}$, $q_{0_{R_i}} = q_{0_{P_i}}$, $\Sigma_{R_i} = \Sigma^S_i \cup_{q,q' \in Q_{R_i}} \{path(q, q')\}$, $\rightarrow_{R_i} = \rightarrow^S_{P_i} \cup_{q,q' \in Q_{R_i}} \{(q, path(q, q'), q')\}$, and $F_{R_i} = F'_{P_i} \cup (F_{P_i} \cap Q^S_{P_i})$.

VI. Construction of Satisfying MS plans

To complete the solution to Prob. 1, we need to find $\{ms_i, i \in I\}$, such that the corresponding motion of the team satisfies the global specification and the communication constraints $\mathbb{C}$. Given an ms_i, robot $\mathcal{A}_i$ services independent requests regardless of the behaviors of other robots. To service shared requests, it needs to cooperate with other robots. All possible motions of the team with synchronizations on shared requests can be captured by the synchronous product (SP) of the reduced automata R_i, $i \in I$. To make sure that the motion of the team satisfy $\mathbb{C}$, we need to take into account these constraints in the SP. Finally, we find a run accepted by the SP, map it into a run accepted by P_i, and then use it to generate an MS plan for robot $\mathcal{A}_i$.

We now describe the above ideas in more details. Given $\mathbb{C}$ (Eqn. 2), we introduce the concept of communication components for all shared requests. We denote $\mathbb{C}_k$ as the kth connected component of graph $\mathbb{C}$. We can compute all connected components of graph $\mathbb{C}$ in linear time using either breadth-first search or depth-first search. For each shared request σ, we define $\mathbb{C}^\sigma_k = \{v \in \mathbb{C}_k \mid (v, \sigma) \in loc\}$. Recall that the robots can only communicate when they are at the vertices of the same communication components and they need to synchronize before servicing σ.

To construct a SP of the set of reduced automata R_i, $i \in I$ that captures the motion of the team and the communication constraints, we use Def. 3, to which we impose additional communication constraints. Assume that there are n agents in the team and therefore $I = \{1, \ldots, n\}$. We define the SP of the reduced automata R_i, $i \in I$ with communication constraints $\mathbb{C}$ as

$$P_g = (Q_g, q_{0_g}, \Sigma_g, \rightarrow_g, F_g) \tag{7}$$

where $Q_g = Q_{R_1} \times \ldots \times Q_{R_n}$, $q_{0_g} = (q_{0_{R_1}}, \ldots, q_{0_{R_n}})$, $\Sigma_g = \cup_{i=1}^{n} \Sigma_{R_i}$, and $F_g = F_{R_1} \times \ldots \times F_{R_n}$. The transition relation $\rightarrow_g \subset Q_g \times \Sigma_g \times Q_g$ is defined by $(q_g, \sigma_g, q_g') \in \rightarrow_g$, iff (1) for all shared requests σ_g: $q_g[i] \xrightarrow{\sigma_g}_i q_g'[i]$ and $\exists \mathbb{C}_k^{\sigma_g}, v[i] \in \mathbb{C}_k^{\sigma_g}$, for all $i \in I_{\sigma_g}$, and $q_g[i] = q_g'[i]$, for all $i \notin I_{\sigma_g}$, and (2) for all σ_g which are optimized paths generated in Sec. V, $q_g[i] \xrightarrow{\sigma_g}_i q_g'[i]$ if this optimized path belongs to the reduced automaton R_i and $q_g[i] = q_g'[i]$ otherwise, where $q_g[i]$ denotes the ith component of q_g and $v[i]$ denotes the second component of $q_g[i] = (q_i, v_i)$.

In other words, the communication product P_g captures the synchronization of a team of robots, while making sure that the robots can communicate before servicing shared requests. The configurations in which robots occupy disconnected vertices in the communication graph $\mathbb{C}$ before servicing shared requests are excluded. The possible motions of the team for all shared requests and optimized paths are modeled by the transition relation $\rightarrow_g$. A transition with a shared request in P_g occurs when all robots owning σ_g synchronously take allowed transitions. A transition with input σ_g that is an optimized path in R_i only changes the state of robot $\mathcal{A}_i$ without affecting other robots.

We then find an accepted word $w_g = w_g(1) \ldots w_g(m)$ of P_g and refine it to obtain a set of MS plans as follows. We first restore the vertex and service entries by replacing all the optimized paths with the sequence of corresponding vertices and independent requests. Then, for each shared request $w_g(l)$, $1 \leq l \leq m$, we insert a sequence of vertices $v[i]$ before $w_g(l)$, where $i \in I_{w_g(l)}$. We denote w_o as the obtained word. Finally, we obtain an MS plan ms_i for each robot $\mathcal{A}_i$ simply by deleting all the requests and vertices from w_o which are not related to $\mathcal{A}_i$. Note that $ms_i \upharpoonright_{\Sigma_i}$ is equal to $w_o \upharpoonright_{\Sigma_i}$, which is accepted by A_i. Since $\mathcal{L}(\|_{i \in I} A_i) = \mathcal{L}(A)$, $L_{MS}^{team} \subseteq \mathcal{L}(A)$. Moreover, since $w_o \upharpoonright_{\Sigma} \in L_{MS}^{team}$, $L_{MS}^{team} \neq \emptyset$. Hence, the motion of the robotic team with the obtained MS plans $\{ms_i, i \in I\}$ satisfies the global specification.

VII. Automatic Deployment in the RULE

In this section, we show how our solution to Prob. 1 can be immediately used to deploy a team of robots from a rich specification about service requests occurring in a miniature city. Our Robotic Urban-Like Environment (RULE) shown in Fig. 2 is a collection of roads, intersections, and parking lots, which are connected following a simple set of rules, *e.g.,* a road connects two (not necessarily different) intersections. Each intersection has traffic lights that are synchronized in the usual way. Each parking lot consists of several parking spaces, where each parking space can accommodate exactly one car, and each parking lot has enough parking spaces to accommodate all the robots at the same time. The city is easily reconfigurable through re-taping and re-placement of the wireless traffic lights at the intersections.

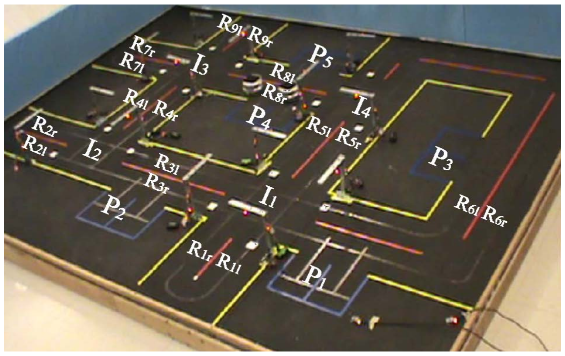

Fig. 2: The topology of the city for the case study from Sec. VII and the road, intersection, and parking lot labels

The robots are Khepera III miniature cars. Each car can sense when entering an intersection from a road, when entering a road from an intersection, when passing in front of a parking lot, when being correctly parked in a parking space, and when an obstacle is dangerously close. Each car can distinguish the color of a traffic light and different parking spaces in the same parking lot. All the cars can communicate through Wi-Fi with a desktop computer and with other cars.

Robot deployment is achieved through a user-friendly graphical interface. The image of the city obtained from four overhead cameras, which are connected to the desktop computer, is converted into a schematic representation. Labels are automatically generated for roads, intersections, and parking spaces and presented to the user, who can specify service requests, their locations, and a task as an arbitrary regular expression over the service requests. The desktop computer performs all the computation and sends the control strategy to the robots through Wi-Fi. Then, the robots execute the task autonomously by interacting with the environment and by communicating with each other, if necessary. The user has the option to simulate the team deployment before trying it on the actual platform (see hyness.bu.edu/rule/ for implementation details).

Modeling RULE using the framework described in Sec. III proceeds as follows. The set of vertices V of the environment graph $\mathcal{E}$ is the set of labels assigned to the roads, intersections, and parking lots (see Fig. 2). The edges in $\rightarrow_{\mathcal{E}}$ show how these regions are connected. We assume that communication is only possible when both of the robots park in the same parking lot, or when one of the robots parks in P_1 and the other one parks in P_2. In other words, the edges of the communication graph $\mathbb{C}$ are given by $E_{\mathbb{C}} = \{(P_1, P_1), (P_2, P_2), (P_3, P_3), (P_4, P_4), (P_5, P_5), (P_1, P_2), (P_2, P_1)\}$. We can then calculate $\mathbb{C}_1 = \{P_1, P_2\}$, $\mathbb{C}_2 =$

$\{P_3\}$, $\mathbb{C}_3 = \{P_4\}$ and $\mathbb{C}_4 = \{P_5\}$.

The motion capabilities of the (identical) robots are captured by a transition system T_i (Eqn. (3)). Note that, in reality, each vertex of T_i has associated a set of motion primitives, and each transition is triggered by a Boolean combination of interrupts. Note that, by selecting a motion primitive available at a vertex, the robot can correctly execute a run of T_i, given that it is initialized on a road. Indeed, only one motion primitive (*follow_road*) is available on a road. At an intersection, the choice of a motion primitive uniquely determines the next vertex given the road that the robot entered the intersection from. For example, there are four motion primitives available at I_1: *turn_right*, *turn_left*, *U_turn*, and *go_straight*. By selecting *turn_right* at I_1, the robot goes to R_{1r} given that it came from R_{3r}. This justifies our assumption from Sec. III that runs of T_i can be executed by the robots. In other words, MS plans defined in Sec. III and derived as described in Sec. VI can be immediately implemented by a robot. It is easy to see that, under some reasonable liveness assumptions about environmental events (*e.g.,* the traffic lights will eventually turn green), such a transition system captures the motion of each robot correctly.

Assume that two robots (cars), labeled as C_1 and C_2, are available for deployment in the city with the topology as shown in Fig. 2. Assume that the set of service requests is $\Sigma = \{H_1, H_2, L_1, L_2, L_3\}$, where L_i, $i = 1, 2, 3$ are "light" requests, which require only one robot, and therefore should be serviced in parallel, while H_i, $i = 1, 2$ are "heavy", and require the cooperation of the two robots. Assume that C_1 can service L_1 and C_2 can service L_2 and L_3, *i.e.,* the set of requests is distributed as $\Sigma_1 = \{L_1, H_1, H_2\}$ and $\Sigma_2 = \{L_2, L_3, H_1, H_2\}$ between the two robots. Assume that the requests occur at the parking lots as given by the location relation $loc = \{(P_1, H_1), (P_1, L_1), (P_2, H_1), (P_2, L_1), (P_2, L_2), (P_3, L_3), (P_4, H_2), (P_5, H_2)\}$. From this, we can easily compute $\mathbb{C}_1^{H_1} = \{P_1, P_2\}$, $\mathbb{C}_3^{H_2} = \{P_4\}$, $\mathbb{C}_4^{H_2} = \{P_5\}$ and $\mathbb{C}_2^{H_1} = \mathbb{C}_3^{H_1} = \mathbb{C}_4^{H_1} = \mathbb{C}_1^{H_2} = \mathbb{C}_2^{H_2} = \emptyset$.

Finally, assume that the global task specification is to first service H_1, then both L_1 and L_2 in an arbitrary order, then H_2, and finally both L_1 and L_3 in an arbitrary order. The specification translates to the following RE:

$$H_1\ (L_1L_2\ +\ L_2L_1)\ H_2\ (L_1L_3\ +\ L_3L_1).$$

By applying the method described in Sec. IV, we find that this global specification is a product language. The local specifications for car C_1 and C_2 are $H_1L_1H_2L_1$ and $H_1L_2H_2L_3$, respectively. Two accepting MS plans are ms_1 : $R_{1l}I_1R_{6r}P_1H_1P_1L_1R_{6r}I_4R_{8l}P_5H_2R_{8l}I_3R_{4l}I_2R_{3r}P_2L_1$ and ms_2 : $R_{2l}I_2R_{3r}P_2H_1P_2L_2R_{3r}I_1R_{5r}I_4R_{8l}P_5H_2R_{8l}R_{8r}$ $I_4R_{6l}P_3L_3$.

The above MS plans are then mapped to control and communication strategies (Sec. III) through the use of motion and communication primitives, and interrupts as described above. The deployment of the robots on the RULE platform is shown in the video accompanying the paper, which is also available at hyness.bu.edu/rule/media.html. We also include footage of the RULE simulator for another case study in the accompanying video.

VIII. Conclusion

We presented a framework for automatic deployment of a robotic team from a specification given as a regular expression over a set of service requests occurring at known locations of a partitioned environment. We assume that several requests can occur at each region, and the robots are subject to environment-induced communication constraints. Given the robot capabilities to service the requests, and the possible cooperation requirements for some requests, we first determine whether the specification is distributable among the robots, and generate individual specifications in the form of finite state automata and synchronization rules among the automata. We use these to generate a set of reduced product automata, which we then employ to look for motion and service plans that satisfy the communication constraints. These plans are eventually mapped to robot control and communication strategies. We illustrate the method with experimental and simulation results in our Robotic Urban-Like Environment (RULE).

References

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

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

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

[4] 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.

[5] 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.

[6] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *IEEE Conference on Decision and Control*, Shanghai, China, 2009.

[7] 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.

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

[9] 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.

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

[11] L. Habets, P. Collins, and J. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Trans. Aut. Control*, vol. 51, pp. 938–948, 2006.

[12] C. Belta and L. Habets, "Control of a class of nonlinear systems on rectangles," *IEEE Transactions on Automatic Control*, vol. 51, no. 11, pp. 1749 – 1759, 2006.

[13] S. H. Rodger and T. W. Finley, *JFLAP: An Interactive Formal Languages and Automata Package*. Sudbury, MA: Jones and Bartlett Publishers, 2006.

[14] A. Stefanescu, "Automatic synthesis of distributed transition systems," Ph.D. dissertation, 2006.

[15] K. Heljanko, "Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets," *Fundamenta Informaticae*, vol. 37, no. 3, pp. 247–268, 1999.