A Formal Approach to Deployment of Robotic Teams in an Urban-Like Environment

Yu Shan Chen, Xu Chu Ding, Alin Stefanescu, and Calin Belta

Abstract We present a computational framework for automatic synthesis of control and communication strategies for a robotic team from task specifications given as regular expressions about servicing requests in an environment. Our approach is based on two main ideas. First, we extend recent results from formal synthesis of distributed systems to check for the distributability of the task specification and to generate local specifications, while accounting for the service and communication capabilities of the robots. Second, by using a technique inspired from LTL model checking, we generate individual control and communication strategies. We illustrate the method with experimental results in our Robotic Urban-Like Environment.

1 Introduction

The goal in robot motion planning and control is to be able to specify a motion task in a rich, high level language and have the robot(s) automatically convert this specification into a set of low level primitives, such as feedback controllers and communication protocols, to accomplish the task [5, 12]. In most of the existing works, the motion planning problem is simply specified as "go from A to B while avoiding obstacles" [12]. However, there are situations in which this is not enough to capture the nature of the task. Consider, for example, the miniature Robotic Urban-Like Environment (RULE) shown in Fig. 1, where a robot might be required to "Visit Road R_1 or Road R_2 without crossing Intersection I_3, and then park in an available parking space," while at same time obeying the traffic rules. Such a "rich" specification cannot be trivially converted to a sequence of "go from A to B" primitives.

Y. S. Chen, X. C. Ding and C. Belta
Boston University, Boston, MA, US, e-mail: yushanc, xcding, cbelta@bu.edu

A. Stefanescu
Department of Computer Science, University of Pitesti, 110040 Pitesti, Romania, e-mail: alin@stefanescu.eu

When several robots 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) robot, while others require the collaboration of two or more (possibly specific) robots. For example, assume that the task is to assemble a piece of machinery in location P_1 or P_2 from two components that can be found at P_3 and P_4. The assembly requires the cooperation of two robots, and the collection of the components needs to be performed in parallel. 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.

It has been advocated as far back as [1] and more recently in [13, 7, 19] that temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [6], can be used as "rich" specification languages in mobile robotics. All of the above works suggest that the corresponding formal verification (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 [11] or reactive games [8]. However, such bottom-up approaches are expensive and can lead to state-space explosion even for relatively simple problems. As a result, one of the main challenges in the area of motion planning and control of distributed teams based on formal verification is to create provably-correct, top-down approaches in which a global, "rich" specification can be decomposed into local (individual) specifications, which can then be used to automatically synthesize robot control and communication strategies. In such a framework, the construction of the parallel composition of the individual motions is not necessary, and therefore the state-space explosion problem is avoided.

In this paper, we draw inspiration from the area of distributed formal synthesis [15] 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 algorithm 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

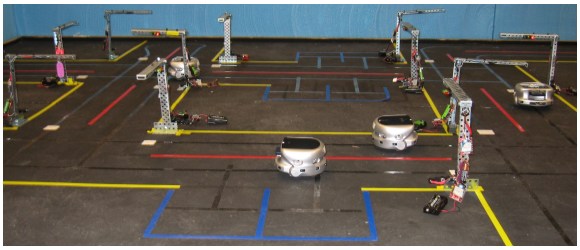

Fig. 1 Robotic Urban-Like Environment (RULE). Khepera III car-like robots move autonomously on streets while staying in their lanes, obeying traffic rules, and avoiding collisions.

of presentation, we model the 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 [5]. In particular, by using feedback controllers for facet reachability polytopes [9, 2], this scenario can be extended to robots with continuous dynamics moving in environments with polytopic partitions.

The contribution of this work is threefold. First, we develop a top-down computational framework for automatic deployment of mobile agents from global specifications given as regular expressions over environmental requests. This is a significant extension of our recent work [3] by enlarging the class of specifications for which a solution exists. Second, we provide a relaxation to the standard problem of distributed synthesis modulo synchronous products and language equivalence [15]. Specifically, we show how a satisfying distributed execution can be found when the global specification is only a traced-closed language, rather than a product language. This extends our previous work [18], in which we provided two heuristics for the case of asynchronous automata. Third, we implement and illustrate the computational framework in our Khepera-based Robotic Urban-Like Environment (Fig. 1). In this experimental setup, the robots can be automatically deployed from specifications given as regular expressions over requests occurring at regions in the city.

2 Preliminaries

Throughout this paper, we assume that the reader is familiar with automata and trace theory [10, 14]. In this section, we review some concepts and introduce notations.

For a set Σ, we use $|\Sigma|$ and 2^{Σ} to denote its cardinality and power set, respectively. A collection of subsets $\Delta = \{\Sigma_i \subseteq \Sigma, i \in I\}$ is called a *distribution* over Σ if $\bigcup_{i\in I} \Sigma_i = \Sigma$, where I is an index set. A word is a sequence of symbols from Σ. We denote Σ^* as the set of all finite words over Σ. A *language* is a set of words.

Definition 1. A finite state automaton (FSA) is a tuple $A = (Q, q_0, \Sigma, \rightarrow_A, F)$, where Q is the set of states, $q_0 \in Q$ is the initial state, Σ is the set (alphabet) of actions, $\rightarrow_A \in Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. We also write $q \xrightarrow{\sigma}_A q'$ to denote $(q, \sigma, q') \in \rightarrow_A$.

We denote $\mathscr{L}(A)$ as the language accepted by an FSA A. The language of an FSA is called a *regular language*, which can be concisely represented by a *regular expression* (RE). Given an RE, an FSA accepting all and only the words satisfying the RE can be constructed by using an off-the-shelf tool, such as JFLAP [16].

Definition 2. The synchronous product (SP) of n FSAs $A_i = (Q_i, q_{0_i}, \Sigma_i, \rightarrow_{A_i}, F_i)$, denoted by $\|_{i=1}^{n} A_i$, is an FSA $A = (Q, q_0, \Sigma, \rightarrow_A, F)$, 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 $\rightarrow_A \subseteq Q \times \Sigma \times Q$ is defined by $q \xrightarrow{\sigma}_A q'$ iff $\forall\, i \in I_\sigma : q[i] \xrightarrow{\sigma}_{A_i} q'[i]$ 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 convenience, in the particular case when $\Sigma_1 = \Sigma_2 = \Sigma$, we use $A_1 \times A_2$ to denote $\|_{i=1}^{2} A_i$. Moreover, $\mathscr{L}(A \times B) = \mathscr{L}(A) \cap \mathscr{L}(B)$ ([20]). An FSA $\neg A$ is defined as an FSA that accepts the language $\overline{\mathscr{L}(A)}$ where $\overline{\mathscr{L}(A)} := \Sigma^* \setminus \mathscr{L}(A)$.

For a word $w \in \Sigma^*$ and a subset $S \subseteq \Sigma$, we denote by $w \restriction_S$ the *projection* of w onto S, which is obtained by erasing all actions σ in w that do not belong to S. For a language $L \subseteq \Sigma^*$ and a subset $S \subseteq \Sigma$, we denote by $L \restriction_S$ the projection of L onto S, which is given by $L \restriction_S := \{w \restriction_S \mid w \in L\}$. 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 another FSA (denoted by $A \restriction_S$) accepting the language $\mathscr{L}(A) \restriction_S$, through ε-closure, determinization and minimization ([17]).

Definition 3. Given a distribution Δ of Σ, the product of a set of languages L_i over Σ_i is denoted by $\|_{i \in I} L_i$ and defined as $\|_{i \in I} L_i := \{w \in \Sigma^* \mid w \restriction_{\Sigma_i} \in L_i \text{ for all } i \in I\}$. A *product language* over a distribution Δ of Σ is a language L such that $L = \|_{i \in I} L_i$, where $L_i = L \restriction_{\Sigma_i}$ for all $i \in I$.

Definition 4. Given a distribution Δ of Σ and $w, w' \in \Sigma^*$, we say that w is trace-equivalent to w' ($w \sim_\Delta w'$) iff $w \restriction_{\Sigma_i} = w' \restriction_{\Sigma_i}, \forall i \in I$. We denote by $[w]_\Delta$ the trace-equivalence class of $w \in \Sigma^*$. A *trace-closed language* over a distribution Δ of Σ is a language L such that for all $w \in L$, $[w]_\Delta \subseteq L$.

3 Problem Formulation and Approach

Let

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

be an environment graph, where V is the set of vertices and $\rightarrow_{\mathscr{E}} \subseteq V \times V$ is a relation modeling the set of edges. For example, $\mathscr{E}$ can be the quotient graph of a partitioned environment, where V is a set of labels for the regions in the partition, and $\rightarrow_{\mathscr{E}}$ 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 $\rightarrow_{\mathscr{E}}$ can show how these are connected (see Fig. 2). Assume we have a team of mobile robots $\mathscr{A}_i$, $i \in I$, whose motions are restricted by $\mathscr{E}$, where I is a set of robot labels.

Let Σ be a set of service requests, or actions to be performed at the vertices of $\mathscr{E}$. The locations of the service requests are defined as a function $a : \Sigma \to V$ (*i.e.,* different requests can occur at the same vertex but vertices do not share requests). There may be no request at some vertices of $\mathscr{E}$.

We model the capacity of the robots to service requests and cooperation among robots as a distribution Δ over Σ (*i.e.* $\cup_{i \in I} \Sigma_i = \Sigma$). Σ_i is the set of requests that can be serviced by the robot $\mathscr{A}_i$. For a given request $\sigma \in \Sigma$, we define $I_\sigma = \{i \in I \mid \sigma \in \Sigma_i\}$, *i.e.,* I_σ is the set of labels of all the agents 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 agent that owns it), the agent can (and should) service the request by itself, independent of the other agents. If $|I_\sigma| > 1$, all the agents $\mathscr{A}_i$ with $i \in I_\sigma$

must service the request simultaneously. An agent is said to service a request σ if it visits the vertex $a(\sigma)$. We assume that two or more robots can communicate only when they are at vertices at which shared requests occur.

We model the motion capabilities of each agent $\mathscr{A}_i$, $i \in I$ on the environment graph $\mathscr{E}$ as a transition system T_i, defined as follows:

$$T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \models_i),\ i \in I, \tag{2}$$

where $v_{0_i} \in V$ is the initial state representing the initial location of $\mathscr{A}_i$, $\rightarrow_i \subseteq V \times V$ is a reflexive transition relation satisfying $\rightarrow_i \subseteq \rightarrow_{\mathscr{E}} \cup_{v \in V} \{(v,v)\}$, $\Pi = \Sigma \cup \{\varepsilon\}$ is a finite set of observations, ε is the empty request, and $\models_i \subseteq V \times \Pi$ is a satisfaction relation where $(v,\varepsilon) \in \models_i, \forall v \in V$ and $(v,\sigma) \in \models_i$, $\sigma \in \Sigma_i$, iff $a(\sigma) = v$. A transition $(v,v') \in \rightarrow$ is also denoted by $v \rightarrow v'$. For an arbitrary state $v \in V$, we define $\Pi_v = \{\pi \in \Pi \mid (v,\pi) \in \models\} \in 2^{\Pi}$ as the set of all observations satisfied at v. A *trajectory* of T_i is a sequence $v(0)v(1)\ldots v(n)$ with the property that $v(0) = v_{0_i}$, $v(i) \in V$, and $(v(i) \rightarrow v(i+1))$, $\forall i \geqslant 0$. We say a trajectory $v = v(0)v(1)\ldots v(n)$ of T_i satisfies a word $w = w(0)w(1)\ldots w(n)$ if $w(i) \in \Pi_{v(i)}$, $\forall i \geqslant 0$. In other words, the motion of robot $\mathscr{A}_i$ is restricted by the transition relation $\rightarrow_i$, which captures motion constraints in addition to $\rightarrow_{\mathscr{E}}$. The locations of the requests in the environment are captured by relation $\models_i$. Each vertex satisfying ε captures that a robot can pass through a vertex without servicing any request.

Definition 5. A *motion and service plan* (or MS plan for short) for robot $\mathscr{A}_i$, $i \in I$ 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 $ms_i(j) \in \Pi_{ms_i(j-1)}$ (*i.e.* $ms_i(j-1) = a(ms_i(j))$), for all $j > 1$ and (3) if $ms_i(j) \in V$ and $ms_i(j-1) \in V$, then $(ms_i(j-1), ms_i(j)) \in \rightarrow_i$, for all $j > 1$. A *motion plan* for robot $\mathscr{A}_i$, $i \in I$, defined as $m_i = ms_i \restriction_V$, can be obtained from the MS plan by deleting all request entries $ms_i(j) \in \Sigma_i$. Similarly, a *service plan* for robot $\mathscr{A}_i$, $i \in I$, is defined as $s_i = ms_i \restriction_{\Sigma_i}$, can be obtained from the MS plan by deleting all motion entries $ms_i(j) \in V$.

The semantics of an MS plan is as follows. A vertex entry $ms_i(j) \in V$ means that the vertex $ms_i(j)$ should be visited. A request entry $ms_i(j) \in \Sigma_i$, means that robot $\mathscr{A}_i$ should service request $ms_i(j)$ at vertex $ms_i(j-1)$. A 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 $\mathscr{A}_i$ broadcasts request $ms_i(j)$ and listens for broadcasts of $ms_i(j)$ from all agents $\mathscr{A}_j$, $j \in I_{ms_i(j)} \setminus \{i\}$. When they are all received, the request $ms_i(j)$ is serviced and then $\mathscr{A}_i$ moves to the next vertex.

Note that by the definition (conditions (2) and (3)) of an MS plan ms_i, the motion plan $m_i = ms_i \restriction_V$ is a trajectory of T_i satisfying a word $w_i \in (\Sigma_i \cup \varepsilon)^*$, where its corresponding service plan $s_i = ms_i \restriction_{\Sigma_i}$ is equal to $w_i \restriction_{\Sigma_i}$. We say that a word s_i can be *implemented* by the robot $\mathscr{A}_i$ if there exists a MS plan ms_i such that $ms_i \restriction_{\Sigma_i} = s_i$.

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

$$L_{MS}^{team}(\{ms_i, i \in I\}) := \|_{i \in I} \{s_i\}, \text{ where } s_i = ms_i \restriction_{\Sigma_i}, \tag{3}$$

(see Def. (3)) as the set of all possible sequences of requests serviced by the team of robots $\mathscr{A}_i, i \in I$ 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 $L_{MS}^{team} \neq \emptyset$ and all words in L_{MS}^{team} satisfy ϕ (*i.e.* $L_{MS}^{team} \subseteq \mathscr{L}(A)$, where A is an FSA accepting only the words satisfying ϕ). We are now ready to formulate the main problem:

Problem 1. Given a team of agents $\mathscr{A}_i$, $i \in I$ with motion capabilities T_i (Eqn. (2)), a set of service requests Σ, a task specification ϕ in the form of an RE over Σ, and a distribution Δ over Σ, find a set of MS plans $\{ms_i, i \in I\}$ such that the motion of the team satisfies ϕ.

Remark 1. For a set of MS plans, the corresponding L_{MS}^{team} could be an empty set by the definition of product of languages (since there may not exist a word $w \in \Sigma^*$ such that $w \restriction_{\Sigma_i} = s_i, \forall i \in I$). In practice, this case 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 σ does not appear in the service plan of one of the agent who owns σ but it appears in the service plans of some other agents, then all those agents 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.

In the case that Prob. 1 has a solution, for each MS plan ms_i, 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, we assume that $\mathscr{A}_i$ can follow any trajectory of T_i (see Sec. 5).

Our approach to solve Problem 1 can be summarized as follows. We first generate an "implementable" FSA for each robot, which captures all the possible service plans that can be implemented by the robot. Then, if the language satisfying the global specification ϕ is trace-closed, we generate a solution to the problem. Otherwise, we attempt to construct an FSA whose language is trace-closed and satisfies the global specification. If we succeed (the language of this FSA is not empty), then we use it to generate a solution. Our overall approach is summarized as a provably-correct algorithm in Sec. 4.

In our previous work [3], we provided a solution to Prob. 1 by following the "standard" approach to distributed synthesis modulo synchronous products and language equivalence [15]. As a result, our approach was conservative, since we could only generate a solution for the particular case when the language satisfying ϕ was a product language (Def. 3). In this paper, we show that we can find a solution to Prob. 1 if the language satisfying ϕ is trace-closed. Since trace-closed languages

are less restrictive than product languages ([17]) (*i.e.* product languages are trace-closed but not vice versa), we significantly reduce the conservatism from our previous approach. In addition, our current approach is less expensive. Indeed, checking whether a language is trace-closed is linear in the size of the FSA accepting the language, while checking whether a language is a product language is PSPACE-complete [17].

4 Synthesis of local MS plans from the global specification

We omit all the proofs in this section due to space limitations. They are available in our detailed technical report [4].

We begin with the conversion of the specification ϕ over Σ to a minimal and deterministic FSA $A = (Q, q_0, \Sigma, \rightarrow, F)$, which accepts exactly the language over Σ that satisfies ϕ (using JFLAP [16]). We call A the *global* specification. Given the distribution Δ, we assign requests to each agent. Specifically, we construct a set of projected FSAs $A_i = (Q_i, q_{0_i}, \Sigma_i, \rightarrow_{A_i}, F_i)$ whose languages are the projections of $\mathscr{L}(A)$ onto the local alphabets Σ_i, $i \in I$ (see Sec. 2). The projected FSAs are used as a starting point to find a solution to Prob. 1 because of the following proposition.

Proposition 1. *If a set of MS plans $\{ms_i, i \in I\}$ is a solution to Prob. 1, then its corresponding service plans $s_i = ms_i \upharpoonright_{\Sigma_i}$ are accepted words of A_i (i.e. $s_i \in \mathscr{L}(A_i), \forall i \in I$).*

However, to provide a provably correct solution for Prob. 1, it is not sufficient to simply choose an arbitrary accepted word from the projected FSAs A_i to be a service plan s_i. We need to make sure that (1) the service plan s_i can be implemented by robot $\mathscr{A}_i$ and (2) all possible sequences of request serviced by the robotic team following their MS plans satisfy the global specification. To satisfy the first requirement, we aim to find an FSA A_i^E for each $i \in I$ such that $\mathscr{L}(A_i^E)$ equals all the accepted words of A_i that can be implemented by the agent $\mathscr{A}_i$ in the environment.

To obtain A_i^E, we construct a new FSA $\widehat{A}_i$ from A_i by adding the action ε to Σ_i and self-transitions (q, ε, q) to each state $q \in Q_i$. For a robot, the action ε means that no request is serviced. We denote the set of all these self transitions by $\rightarrow_{\varepsilon_i}$. The FSA $\widehat{A}_i$, $i \in I$, can now be defined as: $\widehat{A}_i = (\widehat{Q}_i, \widehat{q}_{0_i}, \widehat{\Sigma}_i, \rightarrow_{\widehat{A}_i}, \widehat{F}_i)$, where $\widehat{Q}_i = Q_i$, $\widehat{q}_{0_i} = q_{0_i}$, $\widehat{\Sigma}_i = \Sigma_i \cup \{\varepsilon\}$, $\rightarrow_{\widehat{A}_i} = \rightarrow_{A_i} \cup \rightarrow_{\varepsilon_i}$, and $\widehat{F}_i = F_i$.

Given a word $\widehat{w} \in \mathscr{L}(\widehat{A}_i)$, we can obtain a word $w = \widehat{w} \upharpoonright_{\Sigma_i} \in \mathscr{L}(A_i)$. Note that the input ε corresponds to the observation ε in the transition system T_i and the set of inputs $\widehat{\Sigma}_i$ of $\widehat{A}_i$ is a subset of the observations Π of T_i. To restrict the trajectories of a TS T_i with a set of observations Π to the language accepted by an FSA with a set of actions $\widehat{\Sigma}_i \subseteq \Pi$, we define the following product automaton, which is inspired from LTL model checking [6]:

Definition 6. (Adapted from [7]) The product automaton $P_i = T_i \otimes \widehat{A}_i$ between the transition system $T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \vDash_i)$ and the FSA $\widehat{A}_i = (\widehat{Q}_i, \widehat{q}_{0_i}, \widehat{\Sigma}_i, \rightarrow_{\widehat{A}_i}, \widehat{F}_i)$, is

an FSA $P_i = (Q_{P_i}, q_{0_{P_i}}, \Sigma_{P_i}, \rightarrow_{P_i}, F_{P_i})$, where $Q_{P_i} = V \times \widehat{Q}_i$, $q_{0_{P_i}} = (v_{0_i}, \widehat{q}_{0_i})$ is the set of initial states, $\Sigma_{P_i} = \widehat{\Sigma}_i \subseteq \Pi$ is the set of inputs and $F_{P_i} = V \times \widehat{F}_i$ is the set of accepting (final) states. The transition relation $\rightarrow_{P_i} \subseteq Q_{P_i} \times \Sigma_{P_i} \times Q_{P_i}$ is defined as $(v,q) \xrightarrow{\sigma_{P_i}}_{P_i} (v',q')$ iff $v \rightarrow_i v', q \xrightarrow{\sigma_{P_i}}_{\widehat{A}_i} q'$ and $\sigma_{P_i} \in \Pi_v$.

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 vertex v to v' without servicing any request. $r_{P_i} = (v_i(0), \widehat{q}_i(0)) \ldots (v_i(n), \widehat{q}_i(n))$, where $\widehat{q}_i(j) \in \widehat{Q}_i$, $v_i(j) \in V$ and $j \in \{1, \ldots, n\}$ is a run of P_i. We define the projection of r_{P_i} onto T_i as $\gamma_{T_i}(r_{P_i}) = v_i(0) \ldots v_i(n)$. The next proposition highlights a useful property of P_i.

Proposition 2. *Given any word $w_{\widehat{A}_i} \in \mathscr{L}(\widehat{A}_i)$, there exist at least one trajectory of T_i satisfying $w_{\widehat{A}_i}$ iff $w_{\widehat{A}_i} \in \mathscr{L}(P_i)$.*

Finally, we obtain A_i^E that captures $\mathscr{L}(P_i)$ by removing environment information stored in P_i. To achieve this, we collapse the states of P_i, by taking ε-closure, determinizing, and minimizing P_i. The interested readers are referred to [10] for more details about these procedures. Thus, given a word $w \in \mathscr{L}(A_i^E)$, there exists a word $w' \in \mathscr{L}(P_i)$ such that $w' \restriction_{\Sigma_i} = w$. Using this fact, the following proposition shows that A_i^E captures the largest subset of $\mathscr{L}(A_i)$ which can be implemented by the robot $\mathscr{A}_i$ in the environment.

Proposition 3. *A word $w_i^E \in \mathscr{L}(A_i)$, $i \in I$, can be used to generate a MS plan ms_i for $\mathscr{A}_i$, such that $ms_i \restriction_{\Sigma_i} = w_i^E$, if and only if $w_i^E \in \mathscr{L}(A_i^E)$.*

To address the second requirement mentioned eariler (*i.e.* all possible sequences of requests serviced by the team satisfy ϕ), we aim to find a set of service plans $\{s_i, i \in I\}$ such that $\|_{i \in I} \{s_i\} \subseteq \mathscr{L}(A)$ and $\|_{i \in I} \{s_i\} \neq \emptyset$. The following proposition shows that a trace-closed language is sufficient to satisfy this requirement:

Proposition 4. *Given a language L and a distribution Δ of Σ, if L is a trace-closed language and $w \in L$, then $\|_{i \in I} \{w \restriction_{\Sigma_i}\} \subseteq L$.*

Our approach aims to construct an FSA A_G whose language is both trace-closed and included in $\mathscr{L}(A)$. By Prop. 4, an arbitrary word accepted by A_G can be used to generate a set of service plans satisfying the desired requirement by projecting this word onto the given distribution Δ. Furthermore, we use the synchronous product (SP) of the local implementable specifications generated previously to ensure that the obtained service plans can be implemented by individual agents. This is achieved by taking product of automata, which produces intersection of regular languages.

Specifically, to find A_G, we first check if $\mathscr{L}(A)$ is trace-closed. An algorithm that checks this property for an arbitrary FSA can be found in [4]. If $\mathscr{L}(A)$ is trace-closed, then we define $A_G = A \times \|_{i \in I} A_i^E$. Otherwise, we define $A_G = \neg(\|_{i \in I} B_i) \times \|_{i \in I} A_i^E$, where $B_i = B \restriction_{\Sigma_i}$ and $B = \|_{i \in I} A_i^E \times (\neg A)$. In this second case, A_G is constructed specifically to remove words $w \in \mathscr{L}(\|_{i \in I} A_i^E)$ that cannot be used to generate desired individual service plans for the robots (*i.e.* $\|_{i \in I} \{s_i = w \restriction_{\Sigma_i}\} \nsubseteq \mathscr{L}(A)$). The following proposition shows that A_G satisfies the desired requirement.

Proposition 5. *$\mathscr{L}(A_G)$ is a trace-closed language and $\mathscr{L}(A_G) \subseteq \mathscr{L}(A)$.*

If $\mathscr{L}(A_G)$ is not empty, then a solution to Prob. 1 can be found by picking any accepted word of A_G. In this paper, we obtain this word w_g by using a backward reachability search starting from the set of accepting states and ending at the initial state. In a particular application, any optimization criterion can be used. Once obtained, w_g is projected onto the given distribution to generate a set of MS plans.

The overall approach proposed in this section is summarized in Alg. 1. In the following theorem, we show that the solution obtained by Alg. 1 is provably correct.

Algorithm 1 : Construction of a set of MS plans from a global specification

Input: A RE ϕ, a distribution Δ, and a set of TS $\{T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \models_i), i \in I\}$

1: Convert ϕ to a deterministic and minimal FSA A and construct $\{A_i, i \in I\}$ $(A_i = A \restriction_{\Sigma_i}, \forall i \in I)$
2: Construct $\{\widehat{A}_i, i \in I\}$, $\{P_i = \widehat{A}_i \otimes T_i, i \in I\}$, $\{A_i^E, i \in I\}$ and then $\|_{i\in I} A_i^E$
3: **if** $\mathscr{L}(\|_{i\in I} A_i^E) = \emptyset$, **return** no solution
4: **if** $\mathscr{L}(A)$ is trace-closed, $A_G = A \times \|_{i\in I} A_i^E$, **else** $A_G = \neg(\|_{i\in I}((\|_{i\in I} A_i^E \times (\neg A)) \restriction_{\Sigma_i})) \times \|_{i\in I} A_i^E$
5: **if** $\mathscr{L}(A_G) = \emptyset$, **return** no solution
6: Find a word $w_g \in \mathscr{L}(A_G)$ and obtain a set of local words $w_i^{loc} = w_g \restriction_{\Sigma_i}$
7: Construct $\{\widehat{A}_i^{loc}, i \in I\}$ such that $\mathscr{L}(A_i^{loc}) = w_i^{loc}$, and then construct $\{P_i^{loc} = \widehat{A}_i^{loc} \otimes T_i, i \in I\}$
8: Obtain $\{r_{T_i} = \gamma_{T_i}(r_{P_i^{loc}}) = v_i(0) \dots v_i(n+1),\ i \in I\}$, where $r_{P_i^{loc}}$ is the accepted run of P_i^{loc}, and $\{w_i = w_i(0) \dots w_i(n), i \in I\}$, which is the corresponding accepted word
9: **return** $\{ms_i = v_i(0)w_i(0) \dots v_i(n)w_i(n) \restriction_{V \cup \Sigma_i},\ i \in I\}$

Theorem 1. *If $\mathscr{L}(A_G) \neq \emptyset$, then Alg.1 returns a solution to Prob. 1, i.e. , a set of MS plans $\{ms_i, i \in I\}$ such that $L_{MS}^{team} \subseteq \mathscr{L}(A)$ and $L_{MS}^{team} \neq \emptyset$.*

Remark 2 (Completeness). In the case that $\mathscr{L}(A)$ is trace-closed, our approach is complete in the sense that we find a solution to Prob. 1 if one exists. This follows directly from Prop. 3 and the definition of product of languages. If $\mathscr{L}(A)$ is not trace-closed, a complete solution to Prob. 1 requires one to find a non-empty trace-closed subset of $\mathscr{L}(A)$ if one exists. This problem is undecidable (the proof is in [4]). Therefore, our overall approach to Prob. 1 is not complete.

5 Automatic Deployment in the RULE

In this section, we show how our solution to Prob. 1 can be used to deploy a team of robots using a rich specification to 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.,* the parking lots can only be located on the side of (each bound of) a road). Each intersection has traffic lights that are synchronized in the usual way. A desktop computer is used to remotely control the traffic lights. Each parking lot has enough parking spaces to accommodate all the robots at the same time.

The robots are Khepera III miniature cars. Each car can drive in the miniature city, obey the traffic rule and sense when an obstacle is dangerously close. Each car is programmed with motion and communication primitives allowing it to safely drive on a road, turn in an intersection, park, and communicate with other cars. All the cars can communicate with a desktop computer, which is used as an interface to the user (*i.e.,* to enter the global specification) and to perform all the computation necessary to generate the individual control and communication strategies. Once computed, these are sent to the cars, which execute the task autonomously by interacting with the environment and by communicating with each other, if necessary. We assume that the communication protocol is deadlock-free.

Modeling RULE using the framework described in Sec. 3 proceeds as follows. The set of vertices V of the environment graph $\mathscr{E}$ is the set of labels assigned to the roads, intersections, and parking lots (see Fig. 2). The edges in $\rightarrow_{\mathscr{E}}$ show how these regions are connected. We assume that inter-robot communication is possible only when the robots are in the same parking lot. The motion capabilities of the (identical) robots are captured by a transition system T_i (Eqn. (2)) which has 27 vertices and 42 transitions. 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. For example, at vertex R_{5l}, only one motion primitive *follow_road* is available, which allows the robot to drive on the road. It is important to 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. In other words, MS plans derived as described in Sec. 4 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.

In the rest of this section, we present a case study. Assume that two robots (cars), labeled as C_1 and C_2, are available for deployment in the city with the topology from Fig. 2. Assume the set of service requests is $\Sigma = \{H_1, H_2, L_1, L_2, L_3, L_4, L_5\}$, where L_i, $i = 1,2,3,4,5$ 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 L_4 and C_2 can service L_2, L_3 and L_5, *i.e.,* the set of requests is distributed as $\Sigma_1 = \{L_1, L_4, H_1, H_2\}$, $\Sigma_2 = \{L_2, L_3, L_5, H_1, H_2\}$. between the two agents. Assume the requests occur at the parking lots as given by the assignment function $a(L_1) = P_1$,

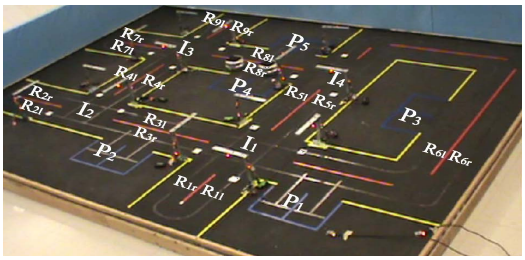

Fig. 2 The topology of the city for the case study from Sec. 5 and the labels of the roads, intersections, and parking lots.

$a(L_2) = P_2$, $a(L_3) = P_3$, $a(L_4) = P_4$, $a(L_5) = P_1$, $a(H_1) = P_4$, and $a(H_2) = P_5$. Finally, assume that the global task specification is to service L_4 and then L_5 or 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. Formally, this specification translates to the following RE over Σ: $L_4L_5 + H_1\ (L_1L_2\ +\ L_2L_1)\ H_2\ (L_1L_3\ +\ L_3L_1)$.

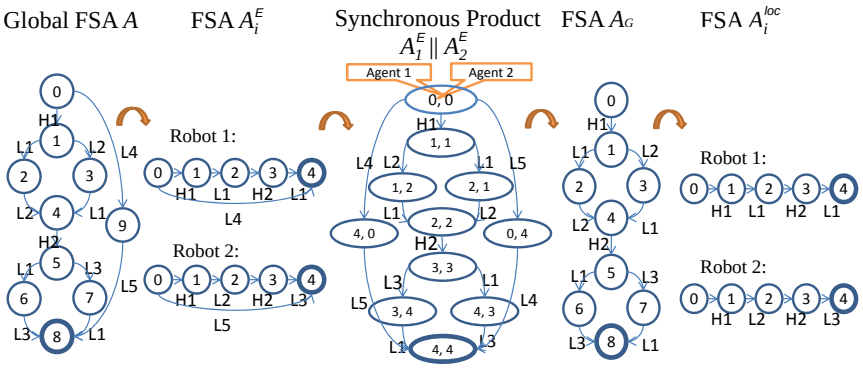

Fig. 3 The FSAs generated by applying Alg. 1.

Using Alg. 1, we generate a set of FSAs A_i^E. Since RULE is fully connected, all the words accepted by A_i can be implemented. In this example, $\mathscr{L}(A)$ is neither a product language nor a trace-closed language (*e.g.*, for $w = L_4L_5$, we have $[w]_\Delta = \{L_4L_5, L_5L_4\}$ and hence, $[w]_\Delta \not\subseteq \mathscr{L}(A)$). Therefore, the FSA A_G is obtained as described in Sec. 4. We choose $w_g = H_1L_1L_2H_2L_1L_3 \in \mathscr{L}(A_G)$. The corresponding service plans for C_1 and C_2 are $s_1 = H_1L_1H_2L_1$ and $s_2 = H_1L_2H_2L_3$, respectively. The FSAs generated by Alg. 1 are shown in Fig. 3. Finally, we generate the MS plans for C_1 and C_2. By assuming that C_1 and C_2 start in R_{2l} and R_{1l} respectively, the two MS plans are

$$ms_1 : R_{2l}I_2R_{4r}I_3R_{8r}P_4H_1R_{8r}I_4R_{5l}I_1R_{6r}P_1L_1R_{6r}I_4R_{8l}P_5H_2R_{8l}I_3R_{8r}I_4R_{5l}I_1R_{6r}P_1L_1$$
$$ms_2 : R_{1l}I_1R_{3l}I_2R_{4r}I_3R_{8r}P_4H_1R_{8r}I_4R_{5l}I_1R_{3l}I_2R_{3r}P_2L_2R_{3r}I_1R_{5r}I_4R_{8l}P_5H_2R_{8l}I_3R_{8r}I_4R_{6l}P_3L_3$$

The above MS plans are then mapped to control and communication strategies through the use of motion primitives and interrupts as described above. The movie of the actual deployment in the RULE platform is available at http://hyness.bu.edu/dars.

6 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. Given the robot capabilities to service the requests, and the possible cooperation requirements for some requests, we find local control and communication strategies such that the global behavior of the system satisfies the given specification. We illustrate the proposed method with ex-

perimental results in our Robotic Urban-Like Environment (RULE). As future work, we will consider expanding the set of global specifications to formulas of temporal logics, such as Linear Temporal Logic, and extensions to probabilistic frameworks.

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. Control of a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11):1749 – 1759, 2006.
3. Y. S. Chen, S. Birch, A. Stefanescu, and C. Belta. A hierarchical approach to automatic deployment of robotic teams with communication constraints. In *International Conference on Intelligent Robots and Systems*, 2010 (to appear).
4. Y. S. Chen, X. C. Ding, A. Stefanescu, and C. Belta. A formal approach to deployment of robotic teams in an urban-like environment. Technical report, Boston University, 2010. (available at hyness.bu.edu/dars).
5. H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, MA, 2005.
6. E. M. Clarke, D. Peled, and O. Grumberg. *Model checking*. MIT Press, 1999.
7. 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.
8. H. Kress 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.
9. L.C.G.J.M. Habets, P.J. Collins, and J.H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Trans. Aut. Control*, 51:938–948, 2006.
10. J.E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2007.
11. M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics*, 26(1):48–61, 2010.
12. S. M. LaValle. *Planning algorithms*. Cambridge University Press, Cambridge, UK, 2006.
13. 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.
14. A. Mazurkiewicz. Introduction to trace theory. In *The Book of Traces*, pages 3–41. WorldSci-Book, 1995.
15. M. Mukund. From global specifications to distributed implementations. In *Synthesis and control of discrete event systems*, pages 19–34. Kluwer, 2002.
16. S. H. Rodger and T. W. Finley. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones and Bartlett Publishers, Sudbury, MA, 2006.
17. A. Stefanescu. *Automatic synthesis of distributed transition systems*. PhD thesis, 2006.
18. Alin Stefanescu, Javier Esparza, and Anca Muscholl. Synthesis of distributed algorithms using asynchronous automata. In *Proc. of the 14th Int. Conf. on Concurrency Theory (CONCUR'03)*, volume 2761 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2003.
19. 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.
20. Sheng Yu. Regular languages. In *Handbook of formal languages, vol. 1: word, language, grammar*, pages 41–110. Springer-Verlag New York, Inc., New York, NY, USA, 1997.