2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Synthesis of an Asynchronous Communication Protocol for Search and Rescue Robots

Clemens Wiltsche, John Lygeros
Automatic Control Laboratory, ETH Zürich
Physikstrasse 3, CH-8092 Zürich, Switzerland
clemens.wiltsche@cantab.net,
lygeros@control.ethz.ch

Federico A. Ramponi
Dipartimento di Ingegneria dell'Informazione,
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, Italy
federico.ramponi@ing.unibs.it

***Abstract*— We present a protocol for reliable communication between search and rescue robots, which is synthesized from a high-level temporal logic specification. The protocol ensures provably correct data transmission on an asynchronous point-to-point link in the presence of an adverse environment. We synthesize the protocol into correct-by-construction transceiver controllers that can be included as building blocks in a larger design. The viability of our approach of synthesizing controllers with clearly defined interfaces and the validity of our protocol is demonstrated by implementing controllers for robots searching cooperatively for a moving target.**

I. Introduction

With increasing pervasiveness of technology in civilian, military and industrial applications, it has become of paramount importance to provide formal guarantees of safety and reliability for autonomous systems. The traditional "design and verify" paradigm has become a bottleneck in establishing trust in the safety of such systems: verification requires prototyping of the system, and hence the investment of valuable resources in a potentially deficient design. We therefore consider a "specify and synthesize" approach to control system design for efficient development of correct-by-construction controllers from high-level specifications. Specifically, we investigate the development of a reliable asynchronous communication protocol for autonomous agents that cooperate to achieve the common goal of performing a Search and Rescue (SAR) task.

In recent years, the development of robots assisting in disaster response in urban environments has increased in popularity, since robots can be deployed in hazardous areas where human SAR operations would not be possible. Early attempts at robot-assisted SAR already identify the need for cooperative search [6], [10]. The RoboCup-Rescue project, and the successful deployment of SAR robots during the September 11 attacks on the World Trade Center in 2001 initiated a surge of research into robot-assisted SAR [3], [18].

The development of several successful autonomous systems has provided a proof-of-concept of the potential benefits of robot-assisted SAR [5], [11], [12], [13]. However, currently the set of applications is limited by the inability of controllers to reliably react to a dynamically changing environment, required for complete autonomy. Especially since SAR involves interaction with humans in distress, confidence in the reliability of the autonomous systems is necessary. The main technological challenge is the trade-off between versatility and reliability in the presence of constraints, not only theoretical but also economic and operational.

Synthesis of correct-by-construction controllers from high level specifications tackles this trade-off by shifting the validation effort into a framework intuitively understandable by designers. In contrast to the traditional ad-hoc "design and verify" approach, specifications can be checked for inconsistencies and realizability, leading to a faster design cycle while ensuring correctness with mathematical rigor. Recent work by Piterman et al. demonstrated the computational feasibility of synthesis, and led to the development of frameworks for the control of autonomous systems [2], [9], [15].

The environment in a SAR task in disaster response is inherently unpredictable, posing adverse conditions to the searchers. Moreover, the individuals that are to be rescued do not need to remain stationary. It is therefore necessary for the searchers to cooperate and hence they have to communicate. A controller selects actions according to its past knowledge of the state of both the system and environment in order for the system to satisfy a specification, and maintains a continuous interaction with the environment and can therefore be considered as a *reactive system*.

Synthesis from formal specifications has gained significant popularity for the control of autonomous systems in the past years. We therefore provide a communication protocol that is easily integrated in this setting. Specifications in temporal logic can easily be refined, and so our protocol can be used as an off-the-shelf building block for autonomous systems — the protocol provides a well-defined service that can be relied upon by designers.

A major restriction in most current applications of temporal logic synthesis is that cooperating agents are assumed to be perfectly synchronized and that communication is neglected. Our protocol overcomes this restriction by taking asynchrony into account in transmitting data across a reliable point-to-point channel. Moreover, reliable communication is subject to a set of complicating factors: the network itself is susceptible to data loss; communicating peers might fail, leading to protocol violations and unknown delays; and lack of synchronization might lead to observed stuttering and missed glitches. This has to be taken into account when proving correctness of the distribution of the specification.

The main complication in implementing a distributed communication protocol is that several separate components

need to cooperate to satisfy a common specification such as "if there is data to be transmitted, it will eventually arrive." However, synthesis of distributed controllers cannot be done directly with traditional methods, as this is in general undecidable [17]. Thus, we manually distribute the specification, show that the asynchronous composition satisfies the specification, and synthesize each component separately.

We introduce specifications and synthesis in Section II and use them to develop our communication protocol in Section III. Lastly, we show how this protocol can be employed in the development of controllers for SAR robots in Section IV and conclude this paper in Section V.

II. Technical Approach

A. Specification Language

We use linear temporal logic (LTL) to specify the protocol. The syntax of LTL formulae φ over variables V is given by

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \varphi \,\mathcal{S}\, \varphi,$$

where p is an *atomic proposition* over a variable in V, e.g. "$u \leq 3$" for an integer variable $u \in V$. A *propositional formula* is formed of atomic propositions, negation ($\neg$) and disjunction ($\vee$). We also define conjunction ($\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$) and implication ($\varphi \rightarrow \psi = \neg\varphi \vee \psi$). LTL extends traditional propositional logic by introducing the temporal operators next ($\bigcirc$), until ($\mathcal{U}$), since ($\mathcal{S}$), eventually ($\Diamond \varphi = \mathrm{True}\,\mathcal{U}\,\varphi$), always ($\Box \varphi = \neg \Diamond \neg\varphi$), sometimes in the past ($\blacklozenge \varphi = \mathrm{True}\,\mathcal{S}\,\varphi$), and always in the past ($\blacksquare \varphi = \neg \blacklozenge \neg\varphi$).

B. Executable Controller Model and Composition

We define an executable model of a controller, which is considered as the "program" running on a microprocessor.

Definition 1: [15] A *fairness-free Fair Discrete System* (FDS) $\mathcal{M}$ is a tuple $(\mathcal{X}, \mathcal{Y}, Q, \Theta, \rho)$, where:

- $\mathcal{X}$ and $\mathcal{Y}$ (finite) are the *environment* and *system variables* respectively. We require $\mathcal{X} \cap \mathcal{Y} = \emptyset$. Denote the *state variables* $V = \mathcal{X} \cup \mathcal{Y}$, and D_V their domain.
- Q is the set of *states*. A state $q \in Q$ is a valuation of V with a unique identifier. We write $q[u]$ to denote the value of the variable $u \in V$ in the state q. $\mathcal{V}_{\mathcal{Z}}(q)$ is the tuple of valuations of the variables $\mathcal{Z} \subseteq V$ in state q. A state q *satisfies* a propositional formula φ, written $q \models \varphi$, if and only if φ holds when for each occurrence of u in φ the corresponding value $q[u]$ is substituted.
- Θ is the *initial condition*, a propositional formula characterizing the initial states $q \in Q$ for which $q \models \Theta$.
- $\rho : Q \rightarrow 2^Q$ is the (possibly nondeterministic) *transition relation* defining the behavior of the FDS.

An *execution* of an FDS $\mathcal{M} = (\mathcal{X}, \mathcal{Y}, Q, \Theta, \rho)$ is an infinite sequence $\sigma = s_0 s_1 s_2 \ldots$ over D_V such that there exists a sequence $q = q_0 q_1 q_2 \ldots$ over Q satisfying $q_0 \models \Theta$, $\forall j \,.\, q_{j+1} \in \rho(q_j)$ and $\forall j \,.\, \mathcal{V}_V(q_j) = s_j$. The FDS controls only the system variables $\mathcal{Y}$ and the choice of the next state is made *after* the values of the environment variables $\mathcal{X}$ are determined. In this paper we consider *open* systems that interact with their environment, modelled by $\mathcal{X} \neq \emptyset$. Moreover, two FDSs $\mathcal{M}_1 = (\mathcal{X}_1, \mathcal{Y}_1, Q_1, \Theta_1, \rho_1)$ and $\mathcal{M}_1 = (\mathcal{X}_2, \mathcal{Y}_2, Q_2, \Theta_2, \rho_2)$ communicate with each other via shared variables $\mathcal{T}_{2,1} = \mathcal{X}_2 \cap \mathcal{Y}_1$ from $\mathcal{M}_1$ to $\mathcal{M}_2$ and $\mathcal{T}_{1,2} = \mathcal{X}_1 \cap \mathcal{Y}_2$ from $\mathcal{M}_2$ to $\mathcal{M}_1$. Variables not controlled by any FDS are considered global environment variables $\mathcal{E}_1 = \mathcal{X}_1 \backslash \mathcal{T}_{1,2}$ and $\mathcal{E}_2 = \mathcal{X}_2 \backslash \mathcal{T}_{2,1}$. In order to reason about communication, we define the asynchronous composition of FDSs, which assumes no clock synchronization between controllers. The following definition is only stated for two FDSs, but can easily be generalized. Given two FDSs

$$\mathcal{M}_1 = (\underbrace{\mathcal{E}_1 \cup \mathcal{T}_{1,2}}_{\mathcal{X}_1}, \underbrace{\mathcal{S}_1 \cup \mathcal{T}_{2,1}}_{\mathcal{Y}_1}, Q_1, \Theta_1, \rho_1), \text{ and}$$
$$\mathcal{M}_2 = (\overbrace{\mathcal{E}_2 \cup \mathcal{T}_{2,1}}^{\mathcal{X}_2}, \overbrace{\mathcal{S}_2 \cup \mathcal{T}_{1,2}}^{\mathcal{Y}_2}, Q_2, \Theta_2, \rho_2)$$

their *asynchronous composition* is the composite FDS

$$\mathcal{M}_1 \,|\, \mathcal{M}_2 = (\underbrace{\mathcal{E}_1 \cup \mathcal{E}_2}_{\mathcal{X}}, \underbrace{\mathcal{Y}_1 \cup \mathcal{Y}_2}_{\mathcal{Y}}, \underbrace{Q_1 \times Q_2}_{Q}, \underbrace{\Theta_1 \wedge \Theta_2}_{\Theta}, \rho),$$

where the transition relation is $\rho : Q_1 \times Q_2 \rightarrow 2^{Q_1 \times Q_2}$ such that for all states $(q_1, q_2), (q_1', q_2') \in \rho(q_1, q_2)$ if and only if

$$q_1' \in \rho_1(q_1) \wedge (\mathcal{V}_{\mathcal{T}_{1,2}}(q_1') = \mathcal{V}_{\mathcal{T}_{1,2}}(q_2)) \wedge (q_2' = q_2) \vee$$
$$q_2' \in \rho_2(q_2) \wedge (\mathcal{V}_{\mathcal{T}_{2,1}}(q_2') = \mathcal{V}_{\mathcal{T}_{2,1}}(q_1)) \wedge (q_1' = q_1).$$

In order to model asynchronous transmission, the values of the transmission variables $\mathcal{T}_{1,2}$ and $\mathcal{T}_{2,1}$ are equated between q_1 and q_2 in a transition of $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively.

C. Specification Semantics

A *specification* of an FDS $\mathcal{M} = (\mathcal{X}, \mathcal{Y}, Q, \Theta, \rho)$ is an LTL formula over variables $\mathcal{X} \cup \mathcal{Y}$, with the following semantics:

Definition 2: Given an execution $\sigma = s_0 s_1 s_2 \ldots$ and an index $j \geq 0$, the satisfaction relation $\models$ is defined as follows:

$$\begin{aligned}
(\sigma, j) &\models p && \Leftrightarrow && s_j \models p \\
(\sigma, j) &\models \neg\varphi && \Leftrightarrow && (\sigma, j) \not\models \varphi \\
(\sigma, j) &\models \varphi \vee \psi && \Leftrightarrow && (\sigma, j) \models \varphi \;\text{ or }\; (\sigma, j) \models \psi \\
(\sigma, j) &\models \bigcirc \varphi && \Leftrightarrow && (\sigma, j+1) \models \varphi \\
(\sigma, j) &\models \varphi \,\mathcal{U}\, \psi && \Leftrightarrow && \exists k \geq j . (\sigma, k) \models \psi \wedge \\
& && && \quad (\forall i . j \leq i < k \Rightarrow (\sigma, i) \models \varphi) \\
(\sigma, j) &\models \varphi \,\mathcal{S}\, \psi && \Leftrightarrow && \exists k \leq j . (\sigma, k) \models \psi \wedge \\
& && && \quad (\forall i . k < i \leq j \Rightarrow (\sigma, i) \models \varphi).
\end{aligned}$$

A sequence σ is said to *satisfy* a formula φ, written $\sigma \models \varphi$, iff $(\sigma, 0) \models \varphi$ holds. An FDS $\mathcal{M}$ is said to satisfy a formula v, written $\mathcal{M} \models \varphi$, iff all executions σ of $\mathcal{M}$ satisfy φ.

D. Assumption/Guarantee Specifications

A controller is required to satisfy its guarantees only if the environment obeys certain assumptions. Such specifications are called Assumption/Guarantee (A/G) specifications, which are like contracts between the system and the environment, or between several subsystems [7], which suggests that subsystems satisfying A/G specifications can be composed to a larger system that satisfies a global specification [1].

An A/G specification is of the form $\varphi^e \rightarrow \varphi^s$, where φ^e and φ^s are the assumptions and guarantees respectively. We

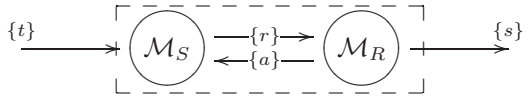


Fig. 1. Composition of FDSs $\mathcal{M}_S$ (sender) and $\mathcal{M}_R$ (receiver). Environment and system variables are on incoming and outgoing arrows resp.

use a particular class of A/G specifications called Generalised Reactivity(1) (GR(1)), see the work by Piterman et al. for a precise definition [15]. GR(1) specifications are attractive as they can be synthesized in polynomial time, and yet they are expressive enough for applications in robotics [20].

E. Synthesis

The *synthesis problem* is to find an FDS $\mathcal{M} = (\mathcal{X}, \mathcal{Y}, Q, \Theta, \rho)$ satisfying a given LTL specification φ. Since φ may contain variables both from $\mathcal{X}$ and $\mathcal{Y}$, the synthesis problem is considered as a turn-based two-player game between the controller and the environment [16]. The controller wins if it satisfies the specification. To synthesize the LTL specifications presented in this paper, we use TuLiP [21], which is based on the method by Piterman et al. [15].

Since communication is between two agents that are physically separated, this notion of synthesis of a two-player game against the environment has to be extended to allow the synthesis of several controllers that together satisfy one global specification by coordinating in order to win against the environment. Such distributed systems are specified by a *global specification* φ together with an architecture defining how many FDSs are to be synthesized and what their respective system and environment variables are. An example architecture is shown in Fig. 1.

The distributed synthesis problem has been shown to be undecidable [17]. We therefore consider distributing the global specification manually into several *local specifications* $\varphi_1, \varphi_2, \ldots$ according to the architecture, and then synthesizing each φ_i separately to the respective FDS $\mathcal{M}_i$. However, when doing so, we must show that the asynchronous composition $\mathcal{M}_1 \,|\, \mathcal{M}_2 \,|\, \cdots$ satisfies the global specification φ.

III. Communication Protocol

In this section we present a reliable asynchronous communication protocol, our main contribution in this paper. Below, we write $[m, n)$ for the range $\{i \in \mathbb{Z} | m \leq i < n\}$ and use $\epsilon = -1$ to indicate the "empty" symbol for better readability.

A. Formulation

We assume state-based transmission on reliable unidirectional links with unknown but finite delay, i.e. transmission is modelled via shared variables rather than via message-passing. These requirements and restrictions have to be taken into account when formulating the protocol specification.

We consider transmitting an integer $\delta \in [0, n)$ from a sender FDS $\mathcal{M}_S$ to a receiver FDS $\mathcal{M}_R$. A boolean environment variable t of $\mathcal{M}_S$ acts as a transmission "trigger". Similarly, a system variable $s \in [0, n)$ of $\mathcal{M}_R$ acts as a "sink" and stores the transmitted data δ after a completed transmission. Hence the global specification φ is $\square(t \rightarrow \lozenge(s = \delta))$, requiring that once a transmission is triggered by t, the data δ eventually arrives correctly at the sink s.

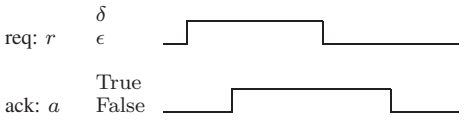


Fig. 2. Four-Phase Handshake Protocol.

$\mathcal{M}_S$ controls a request signal $r \in [\epsilon, n)$. If $r = \epsilon$, this indicates that no data should be transmitted. If $r \neq \epsilon$, the data to be transmitted is $\delta = r$. $\mathcal{M}_R$ controls a boolean acknowledgement signal a, and the request r appears as an environment variable at $\mathcal{M}_R$. Similarly, the acknowledgement a appears as an environment variable at $\mathcal{M}_S$. The architecture is visualized in Fig. 1. The composite FDS $\mathcal{M}_S \,|\, \mathcal{M}_R$ has environment variables $\mathcal{X} = \{t\}$ and system variables $\mathcal{Y} = \{a, r, s\}$. The trigger t may be controlled by a higher level entity using the communication protocol.

The four-phase handshake protocol can be informally described as follows, cf. Fig. 2: the sender $\mathcal{M}_S$ initiates a transfer by setting $r = \delta$. Once $\mathcal{M}_R$ detects this, it may use the transferred data, and then raises a. When $\mathcal{M}_S$ sees that a has been asserted, it resets $r = \epsilon$. After $\mathcal{M}_R$ detects this, it clears a, completing the transmission of δ.

B. Specifications

$\mathcal{M}_S$ assumes that the acknowledgement will be cleared eventually, allowing for a new request (I); there is no acknowledgement without a request (II); and the acknowledgement is only cleared if the request has been reset (III):

$$\begin{array}{ll} \text{I} & \varphi^e_{S,1} = \square(r = \epsilon \rightarrow \lozenge\, \neg a), \\ \text{II} & \varphi^e_{S,2} = \square(\neg a \wedge \bigcirc\, a \rightarrow r \neq \epsilon), \\ \text{III} & \varphi^e_{S,3} = \square(a \wedge \bigcirc\, \neg a \rightarrow r = \epsilon). \end{array}$$

$\mathcal{M}_S$ guarantees that as long as the request is not acknowledged, it stays at the same value (IV); the request may only be reset to ϵ (V); the sender responds to an acknowledgement by resetting the request (VI); and the data will eventually be sent once the transfer has been triggered by t (VII):

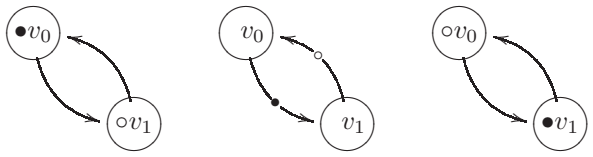
$$\begin{array}{ll} \text{IV} & \varphi^s_{S,1} = \bigwedge_{i \in [0,n)} \square(\neg a \wedge r = i \rightarrow \bigcirc(r = i)), \\ \text{V} & \varphi^s_{S,2} = \bigwedge_{i \in [0,n)} \square(r = i \wedge \bigcirc(r \neq i) \rightarrow \bigcirc(r = \epsilon)), \\ \text{VI} & \varphi^s_{S,3} = \square(a \rightarrow \bigcirc(r = \epsilon)), \\ \text{VII} & \varphi^s_{S,4} = \square(t \rightarrow \lozenge(r = \delta)). \end{array}$$

Note that guarantee VI also ensures that the sender may only initiate a transmission if the acknowledgement signal is low.

$\mathcal{M}_R$ assumes that an acknowledgement eventually results in the request to be cleared (VIII); the request reacts appropriately to the acknowledgement not initiating a transmission if the acknowledgement is still high (IX); and the request may only be cleared if it has been acknowledged (X):

$$\begin{array}{ll} \text{VIII} & \varphi^e_{R,1} = \square(a \rightarrow \lozenge(r = \epsilon)), \\ \text{IX} & \varphi^e_{R,2} = \square(r = \epsilon \wedge \bigcirc(r \neq \epsilon) \rightarrow \neg a), \\ \text{X} & \varphi^e_{R,3} = \square(r \neq \epsilon \wedge \bigcirc(r = \epsilon) \rightarrow a). \end{array}$$

$\mathcal{M}_R$ guarantees that each request will be acknowledged (XI); the resetting of the request results in the acknowledgement



(a) Initial configuration (b) Transition (c) Final configuration

Fig. 3. The target (∘) evading the SAR robot (•).

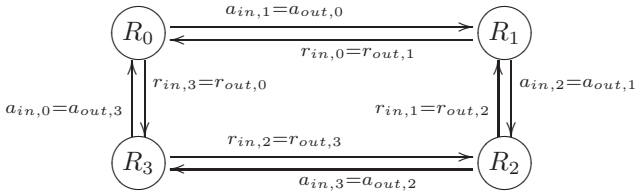


Fig. 4. Four robots communicating in the moving target search architecture. Note that only the local system and environment variables directly relevant for communication are shown.

to be cleared (XII); the acknowledgement may only be raised on a request (XIII); the acknowledgement may not be cleared unless the request is reset (XIV); and the data, once received, is stored in the sink s (XV):

$$\begin{array}{ll} \text{XI} & \varphi^s_{R,1} = \square(r \neq \epsilon \rightarrow \bigcirc a), \\ \text{XII} & \varphi^s_{R,2} = \square(r = \epsilon \rightarrow \bigcirc \neg a), \\ \text{XIII} & \varphi^s_{R,3} = \square(\neg a \wedge \bigcirc a \rightarrow r \neq \epsilon), \\ \text{XIV} & \varphi^s_{R,4} = \square(a \wedge \bigcirc \neg a \rightarrow r = \epsilon), \\ \text{XV} & \varphi^s_{R,5} = \square(r = \delta \rightarrow \Diamond(s = \delta)). \end{array}$$

The complete sender and receiver specifications are then

$$\varphi_S = \varphi^e_S \rightarrow \varphi^s_S, \quad \varphi^e_S = \textstyle\bigwedge_{j=1}^{3} \varphi^e_{S,j}, \quad \varphi^s_S = \bigwedge_{j=1}^{4} \varphi^s_{S,j};$$
$$\varphi_R = \varphi^e_R \rightarrow \varphi^s_R, \quad \varphi^e_R = \textstyle\bigwedge_{j=1}^{3} \varphi^e_{R,j}, \quad \varphi^s_R = \bigwedge_{j=1}^{5} \varphi^s_{R,j},$$

respectively. Both φ_S and φ_R are expressible as GR(1) specifications, and hence synthesizable by TuLiP, yielding $\mathcal{M}_S \models \varphi_S$ and $\mathcal{M}_R \models \varphi_R$. We can show that these FDSs in parallel satisfy the global specification.

Proposition 1: If $\mathcal{M}_S \models \varphi_S$ and $\mathcal{M}_R \models \varphi_R$ then $\mathcal{M}_S \,|\, \mathcal{M}_R \models \square(t \rightarrow \Diamond(s = \delta))$.

Proof: See the Appendix. ■

C. Usage

The four-phase handshake protocol can be used as follows: given specifications $\varphi^e_1 \rightarrow \varphi^s_1$ and $\varphi^e_2 \rightarrow \varphi^s_2$ of autonomous systems $\mathcal{M}_1$ and $\mathcal{M}_2$ that ought to communicate with each other, include the protocol specifications by refining the specifications to $\varphi^e_1 \wedge \varphi^e_S \rightarrow \varphi^s_1 \wedge \varphi^s_S$ and $\varphi^e_2 \wedge \varphi^e_R \rightarrow \varphi^s_2 \wedge \varphi^s_R$ for a one-way communication from $\mathcal{M}_1$ to $\mathcal{M}_2$.

We note that when using the four-phase handshake protocol in this form, it is always possible to use weaker assumptions, as long as the specification remains realizable. In particular, omitting or weakening conjuncts in φ^e_S or φ^e_R does not change the correctness of the protocol (although it may influence its realizability). However, the guarantees φ^s_S and φ^s_R must be provided in a specification. They need not occur in the same syntactic form, but can be strengthened without affecting the correctness of the protocol.

IV. Search and Rescue

In order to test the viability of our communication protocol, we synthesize controllers of SAR robots that search for moving targets. Due to limited space we only describe the idea behind using the protocol for cooperating SAR robots. For a detailed description, see [19]. We assume that the control of the robots cannot be centrally coordinated, and instead develop local specifications that together realize a reliable global search strategy. Interaction between robots is governed by our communication protocol, which is included in each robot's specification as a standardized building block.

A. Problem Setting

The movement of the robots in the SAR task is modelled as transitions between discrete "cells", which could represent rooms or corridors in a building [8], [9], [20]. Hence the robots can be viewed as moving on an underlying *topology*, represented by a digraph G. We require G to be strongly connected, i.e. it is possible to get from every cell to any other cell. We initialize all edges in G to be *contaminated*. If a robot moves along an edge $e = (v, v')$ from v to v' while another robot stays on (*guards*) vertex v, or v has no contaminated in-edges, we say that e is *cleared*. An edge (v, v') becomes recontaminated if v is no longer guarded and has at least one contaminated in-edge.

We assume that while a SAR robot moves between two cells, the targets may make an arbitrary number of moves (this is different from allowing targets to jump between nodes.) A robot *finds* a target if they are on the same vertex or move along the same edge in opposite directions. Depending on the topology G, a minimum number of cooperating SAR robots is required to reliably find the target. For example, in Fig. 3 a single robot is trying to find a moving target by moving on the same vertex, while the target can evade the robot by always moving to the vertex opposite of the robot.

Each robot is fitted with a transceiver that can both sense and set environment and system variables a_{in}, r_{in} and a_{out}, r_{out} respectively of the communication protocol. Also, each robot can reliably detect whether it finds a target, and whether out-edges or in-edges of its current vertex are cleared or contaminated. Lastly, the robot is equipped with actuators to move between vertices, one edge at a time.

B. Cops and Robbers Game

Trying to find a moving target can be seen as a *cops and robbers game*, in which the target (the robber) tries to evade the SAR robots (the cops) [14]. The objective of the game in its original formulation is to find the *cop number*, the minimum number of robots necessary to guarantee that the target is found [4]. However, our goal is to find a distributed winning strategy for the robots, given that sufficiently many SAR robots are available. Our starting point is the centralized strategies developed by Yang and Cao [22]. From a centralized strategy, which assumes that all robots can be

controlled independently, we develop specifications in LTL for distributed *local strategies*, in which each SAR robot controls only its own movements. The main challenge is to realize the necessary cooperation (cf. Fig. 3) between SAR robots, since they can access only local information: a robot only detects whether it has just found a target (and hence does not know where the targets are), and it may only gather information about the state of the topology in its vicinity.

C. Communication

We think of the robots as being connected in a circular manner with point-to-point links, see Fig. 4. These connections remain fixed over the entire search and are independent of the robots' positions. A robot may tell another to either stay in its current cell or move to a particular cell. Hence, each robot has a *master* and a *slave*, resulting in a circular structure of authority. Information exchange is governed by our communication protocol. Note that the circularity of the connections is not related to the structure of the graph G.

The local strategies consist of a combination of transmissions, receptions and moves between cells. The controller of a SAR robot R_i is then modelled as an FDS R_i with sensor inputs being environment variables and actuators being system variables. The environment variables $a_{in,i}$ and $r_{in,i}$ as well as the system variables $a_{out,i}$ and $r_{out,i}$ are added for communication, see Fig. 4. These are the only transmission variables shared between the FDSs.

D. Distributed Strategy

The local strategy for R_i is a GR(1) specification augmented by the communication protocol, cf. Sec. III-C. Each SAR robot R_i can be in one of four modes:

- G) R_i *guards* its current vertex by staying in it unless it receives a command from its master to move to a vertex v. In this case the master of R_i guards v and R_i enters mode C to clear all out-edges of v.
- S) R_i searches for an unguarded vertex v with at least one in-edge and all out-edges contaminated. Once such a vertex is found, R_i enters mode G and commands its slave to enter mode C to clear all out-edges of v.
- P) R_i searches for a vertex v with all in-edges cleared but at least one out-edge contaminated. Once such a vertex is found, R_i enters mode C and clears all out-edges of v. If no such vertex is found, mode S is entered.
- C) R_i clears a vertex by moving along all its out-edges, and once they are all cleared, mode P is entered.

All robots start at some arbitrary vertex in the graph. One robot is searching for a starting vertex (in mode S), all other robots are guarding their vertex (in mode G). The full LTL specifications are given in the technical report [19].

We synthesized controllers for SAR robots that cooperate using our communication protocol. Simulations show that they successfully perform moving target search in randomly generated strongly-connected graphs, given that sufficiently many robots are available. Note that this simplified strategy sometimes requires more robots than the cop-number.

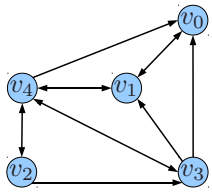


Fig. 5. Example graph to demonstrate distributed strategy.

E. Example

We show how this strategy clears all edges in an example graph of five vertices shown in Fig. 5. Once the entire graph is cleared, all targets must necessarily be found, since there is no more possibility of 'hiding'. Three robots (R_0, R_1 and R_2) is the minimum number required to clear the graph.

Let all robots start in v_0. R_0 is initialized in mode S and searches until it finds v_4, guards it and commands its slave R_1 to clear all out-edges. R_1 subsequently enters mode C and moves to v_4, then moves along the edge (v_4, v_0) to clear it, then moves back to v_4, moves along the edge (v_4, v_1), moves back to v_4 again and so on until it also cleared (v_4, v_3) and (v_4, v_2). R_1 then enters mode P, and searches until it finds v_2, which has no contaminated in-edges but two contaminated out-edges, (v_2, v_3) and (v_2, v_4). So R_1 enters mode C again, moves along these edges to clear them, enters mode P, finds v_3, enters mode C, and clears (v_3, v_4), (v_3, v_1) and (v_3, v_0). Now finally, after entering mode P no appropriate vertex is found, so R_1 enters mode S. R_1 then finds v_1 which is not guarded and has one contaminated in-edge, namely (v_0, v_1). Hence R_1 enters mode G and commands the third robot, R_2 to come to v_1. Note that R_0 still has to guard v_4 because (v_1, v_4) is contaminated. Now R_2 enters mode C and clears (v_1, v_4) and (v_1, v_0). Then R_2 enters mode P, finds v_0 with only the out-edge (v_0, v_1) contaminated, enters mode C and clears this edge. At this point the entire graph is cleared.

For a full report of the controller implementation and simulation results please see [19].

V. Conclusion

We developed a communication protocol for reliable point-to-point communication between asynchronous components. The correctness of the protocol has been verified, so higher-level control can rely on a guaranteed, synchronized interchange of information. This is illustrated by developing controllers for SAR robots where the correct distribution of the global strategy into local strategies for the individual robots is aided by establishing clear interfaces between the robots using our communication protocol.

While the protocol provides reliable communication services to higher level controllers, abstracting from the underlying asynchrony, no quality-of-service can be guaranteed. Moreover, the assumption of reliable finite-delay point-to-point links is not appropriate for wireless transceivers.

Acknowledgment

The authors thank Sean Summers at ETH Zürich for fruitful discussions and valuable comments. Research supported

by the European Commission under the MoVeS project, EU FP7/2007-2013, grant agreement 257005.

APPENDIX

Proof: [of Proposition 1] We use the Composition Theorem of Abadi and Lamport [1], which in our case is

$$\frac{\begin{array}{ll}(H1) & \models \mathfrak{C}(\varphi_S^s) \wedge \mathfrak{C}(\varphi_R^s) \rightarrow \varphi_S^e \wedge \varphi_R^e \\ (H2) & \models \mathfrak{C}(\varphi_S^s) \wedge \mathfrak{C}(\varphi_R^s) \rightarrow \mathfrak{C}(\Box(t \rightarrow \Diamond(s = \delta))) \\ (H3) & \models \varphi_S^s \wedge \varphi_R^s \rightarrow \Box(t \rightarrow \Diamond(s = \delta)) \\ (H4) & \mathcal{M}_S \models \varphi_S^e \rightarrow \varphi_S^s \\ (H5) & \mathcal{M}_R \models \varphi_R^e \rightarrow \varphi_R^s \end{array}}{\mathcal{M}_1 \,|\, \mathcal{M}_2 \models \Box(t \rightarrow \Diamond(s = \delta))},$$

where the *closure* $\mathfrak{C}(\varphi)$ of an LTL formula φ is the strongest safety formula implied by it [1]. Jonsson and Tsay give a syntactic definition of closure in LTL using past operators [7]:

$$\mathfrak{C}(\varphi) = \Box\left[\exists \bar{x} \,.\, \blacksquare(\bar{x} = x) \wedge \blacklozenge(\varphi[\bar{x}/x])\right],$$

where $\varphi[\bar{x}/x]$ is φ with values $\bar{x}$ substituted for variables x, and the existential quantifier $\exists \bar{x}.\varphi$ over finite domains $D_{\mathcal{X}}$ is defined as $\bigvee_{\bar{x} \in D_{\mathcal{X}}} \varphi[\bar{x}/x]$.

For the preconditions (H1) and (H2), consider the closures of φ_S^e and φ_R^e. Since the environment variables of $\mathcal{M}_S$ are t and a, $\mathfrak{C}(\varphi_S^e) = \Box(\exists \bar{t}, \bar{a} \,.\, \blacksquare(\bar{t} = t \wedge \bar{a} = a) \wedge \blacklozenge(\varphi_S^e[\bar{t}/t][\bar{a}/a]))$. Similarly, $\mathfrak{C}(\varphi_R^e) = \Box(\exists \bar{r} \,.\, \blacksquare(\bar{r} = r) \wedge \blacklozenge(\varphi_R^e[\bar{r}/r]))$, since the only environment variables of $\mathcal{M}_R$ is r. Note that $\models \Box\blacklozenge\Box\varphi \leftrightarrow \Box\varphi$ and $\models \Box\blacksquare\varphi \leftrightarrow \Box\varphi$ for any LTL formula φ, which helps to simplify the closures. We proceed to exhaustively consider the individual cases for values of $\bar{t}$, $\bar{a}$, and $\bar{r}$ to deal with the existential quantification.

Case 1. $\bar{t} = \text{True}$, $\bar{a} = \text{True}$: Evaluating the conjuncts for these valuations yields $\Box\bigcirc(r = \epsilon)$ from VI and $\Box\Diamond(r = \delta)$ from VII. These two formulae contradict each other; hence in this case $\mathfrak{C}(\varphi_S^s) = \text{False}$ and so both (H1) and (H2) hold.

Case 2. $\bar{r} \neq \epsilon$ and $\bar{r} \neq \delta$: Evaluating the conjuncts for these variables yields the conjuncts $\Box\bigcirc a$ from XI and $\Box\bigcirc\neg a$ from XII. These two conjuncts in the antecedent contradict each other, and so (H1) and (H2) are both satisfied.

Case 3. $\bar{a} = \text{True}$, $\bar{r} = \epsilon$: The conjunct in the closure resulting from XII is $\Box\bigcirc\neg a$ which contradicts the conjunct $\Box(a = \text{True})$ from the existential quantification.

Case 4. $\bar{a} = \text{True}$, $\bar{r} \neq \epsilon$: The conjunct in the closure resulting from VI is $\Box\bigcirc(r = \epsilon)$, but the existential quantification requires $\Box(r \neq \epsilon)$. This leads to a contradiction.

Case 5. $\bar{t} = \text{True}$, $\bar{a} = \text{False}$, $\bar{r} = \delta$: Evaluating the conjuncts in the closures for these variables yields $\Box\Diamond(s = \delta)$ for XV. Moreover, the conjuncts $\Box(t = \text{True})$, $\Box(a = \text{False})$, and $\Box(r = \delta)$ are in the closures from the existential quantification. For (H1) we need to deduce I–III and VIII–X from these conjuncts. I, IX and X follow from $\Box(r = \delta)$ while II, III and VIII follow from $\Box(a = \text{False})$.

For (H2) we need to verify the individual terms of the closure of $\Box(t \rightarrow \Diamond(s = \delta))$, which is $\Box(\exists \bar{\bar{t}} \,.\, \blacksquare(\bar{\bar{t}} = t) \wedge \blacklozenge\Box(\bar{\bar{t}} \rightarrow \Diamond(s = \delta)))$. In order to resolve the existential quantification, we again consider both truth values of $\bar{\bar{t}}$. However, since we are now proving the closure as a consequent, we only need to exploit one value of $\bar{\bar{t}}$ for which the closure of $\Box(t \rightarrow \Diamond(s = \delta))$ is implied by the antecedents. Let $\bar{\bar{t}} = \text{True}$. Then the closure becomes $\Box\blacksquare(t = \text{True}) \wedge \Box\blacklozenge\Box\Diamond(s = \delta)$. The first conjunct follows simply from $\Box(t = \text{True})$ of the closure of φ_S^s. The second conjunct follows from XV, i.e. $\Box\Diamond(s = \delta)$.

Case 6. $\bar{t} = \text{False}$, $\bar{a} = \text{False}$, $\bar{r} = \epsilon$ or $\bar{r} = \delta$: To prove (H1) we again need to be able to deduce I–III and VIII–X from the conjuncts of the closures. I–III and VIII follow from $\Box(a = \text{False})$ while IX and X follow from both $\Box(r = \epsilon)$ and $\Box(r = \delta)$. For (H2) consider again the closure $\mathfrak{C}(\Box(t \rightarrow \Diamond(s = \delta))) = \Box(\exists \bar{\bar{t}} \,.\, \blacksquare(\bar{\bar{t}} = t) \wedge \blacklozenge\Box(\bar{\bar{t}} \rightarrow \Diamond(s = \delta)))$. Taking $\bar{\bar{t}} = \text{False}$, this closure is implied by $\Box(t = \text{False})$.

(H3) follows from VII and XV. (H4) and (H5) are satisfied by synthesis. The Composition Theorem yields the result. ■

REFERENCES

[1] M. Abadi and L. Lamport. Conjoining specifications. *TOPLAS*, 17(3):507–535, 1995.

[2] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion: State of the art and grand challenges. *Robot. Autom. Mag.*, 14(1):61–70, 2007.

[3] J. Casper and R.R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *Trans. Syst., Man, Cybern., Syst.*, 33(3):367–385, 2003.

[4] T. Chung, G. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31:299–316, 2011.

[5] I. Erkmen, A.M. Erkmen, F. Matsuno, R. Chatterjee, and T. Kamegawa. Snake robots to the rescue! *Robot. Autom. Mag.*, 9(3):17–25, 2002.

[6] J.S. Jennings, G. Whelan, and W.F. Evans. Cooperative search and rescue with a team of mobile robots. *ICAR*, 1997.

[7] B. Jonsson and Y.K. Tsay. Assumption/Guarantee specifications in linear-time temporal logic. *Theor. Comp. Sci.*, 167(1&2):47–72, 1996.

[8] M. Kloetzer and C. Belta. LTL planning for groups of robots. In *ICNSC*, pages 578–583. IEEE, 2006.

[9] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-logic-based reactive mission and motion planning. *Trans. Robot.*, pages 1370–1381, 2009.

[10] M.J. Mataric, M. Nilsson, and K.T. Simsarin. Cooperative multi-robot box-pushing. In *IROS*, volume 3, pages 556–561. IEEE, 1995.

[11] F. Matsuno and S. Tadokoro. Rescue robots and systems in Japan. In *ROBIO*, pages 12–20. IEEE, 2004.

[12] R. Murphy, J. Kravitz, S. Stover, and R. Shoureshi. Mobile robots in mine rescue and recovery. *Robot. Autom. Mag.*, 16(2):91–103, 2009.

[13] R.R. Murphy. Marsupial and shape-shifting robots for urban search and rescue. *Intell. Syst. App.*, 15(2):14–19, 2000.

[14] R.J. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2–3):235–239, 1983.

[15] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of Reactive(1) designs. In *VMCAI*, volume 3855, pages 364–380. Springer, 2006.

[16] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *SIGPLAN-SIGACT*, pages 179–190. ACM, 1989.

[17] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *FOCS*, pages 746–757. IEEE, 1990.

[18] S. Tadokoro, H. Kitano, T. Takahashi, I. Noda, H. Matsubara, A. Shinjoh, T. Koto, I. Takeuchi, H. Takahashi, F. Matsuno, M. Hatayama, J. Nobe, and S. Shimada. The RoboCup-Rescue project: a robotic approach to the disaster mitigation problem. In *ICRA*, volume 4, pages 4089–4094. IEEE, 2000.

[19] C. Wiltsche. Automated synthesis of controllers for search and rescue from temporal logic specifications. arXiv:1304.6898, 2012. Master Thesis.

[20] T. Wongpiromsarn, U. Topcu, and R.M. Murray. Receding horizon control for temporal logic specifications. In *International Conference on Hybrid Systems: Computation and Control*, pages 101–110, 2010.

[21] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R.M. Murray. TuLiP: a software toolbox for receding horizon temporal logic planning. In *HSCC*, pages 313–314. ACM, 2011.

[22] B. Yang and Y. Cao. Standard directed search strategies and their applications. *J. Comb. Opt.*, 17:378–399, 2009.