Communication Policies in Heterogeneous Multi-Agent Systems in Partially Known Environments under Temporal Logic Specifications [⋆]

Christoforos Keroglou [*] Dimos V. Dimarogonas [*]

[*] *Division of Decision and Control Systems, School of Electrical Engineering and Computer Science,*
KTH Royal Institute of Technology, Stockholm, Sweden,
e-mail: {keroglou,dimos}@kth.se

Abstract: In this paper, we explore communication protocols between two or more agents in an initially partially known environment. We assume two types of agents (A and B), where an agent of Type A constitutes an information source (e.g., a mobile sensor) with its own local objective expressed in temporal logic, and an agent of Type B constitutes an agent that accomplishes its own mission (e.g., search and rescue mission) also expressed in temporal logic. An agent of Type B requests information from an agent of Type A to update its knowledge about the environment. In this work, we develop an algorithm that is able to verify if a communication protocol exists, for any possible initial plan executed by an agent of Type B.

Keywords: Formal Methods, Multi-agent Systems, Temporal Logic, Distributed Contol, Discrete Event Systems

1. INTRODUCTION

The motivation of this work is derived from problems in search and rescue missions, where there is great uncertainty, and the management of the available resources (e.g., limited power, or limited communication, or time) is crucial. In the considered setup, restrictions are imposed on the initial knowledge that the agents have about the environment. In this work formal methods are employed to define a motion planning problem for each agent in a multi-agent system. Based on such specification definition, high-level control policies are synthesized after processing local information and information communicated between agents of different types. The objectives of the agents are formulated as temporal logic specifications Fainekos et al. (2009); Kress-Gazit et al. (2009); Lahijanian et al. (2012). Specifically, task specifications are given as syntactically co-safe LTL formulas (scLTLs) which are able to express finite horizon specifications.

Motion planning in partially known environments under temporal logic specifications has recently attracted interest from many researchers Ayala et al. (2013); Hoxha and Fainekos (2016); Meng Guo et al. (2013); Nenchev and Belta (2016). Specifically, in Ayala et al. (2013), motion planning for a single agent is devised, where the agent updates its information about the environment by exploring the area. Our setup is very similar to the one presented in that work, with the important difference that our setup is applied to a multi-agent system. In particular, consider an agent B with no sensing abilities. That agent relies on another agent's sensing ability to update its own knowledge about the surrounding environment. Initially, agent B has a partial knowledge of the environment, which means that it is probable that an initial motion planning could end in a problematic situation, where the agent can not accomplish its mission. For that reason, communication is introduced between the agents, and a communication protocol is designed.

The design of the communication protocol is formulated as a decision-making problem in a decentralized discrete event system Debouk et al. (2000, 2003); Keroglou and Hadjicostis (2018); Yin and Lafortune (2018). In that case, the communication is event-driven, and formal verification along with control techniques are used in the discrete event formalism. Specifically, supervisory control Cassandras and Lafortune (2008) is applied to guarantee specifications expressed in a temporal logic formulation, which in our case (i.e., scLTL specification) is translated into an equivalent finite automaton. The major contribution of this work is the proposal of an algorithm that verifies the existence or not of a communication policy, which guarantees the local objectives for the agents.

The remaining sections of this paper are organized as follows. In Section II, we revisit notions from Languages and Automata, needed later in the paper. In Section III, we formulate the problem. In Section IV, we verify the allowed behavior of a mobile sensor, according to its scLTL specification. In Section V, we develop the algorithm that checks, if there exists a communication policy, which

[⋆] This work was supported by the H2020 ERC Starting Grant BUCOPHSYS, the EU H2020 Research and Innovation Programme under GA No. 731869 (Co4Robots), the SSF, Sweden COIN project, the Swedish Research Council (VR), Sweden and the Knut och Alice Wallenberg Foundation, Sweden.

allows a robotic vehicle, to move without restrictions to its workspace. This is also the main contribution of the paper. In Section VI, we conclude the paper, and we propose extensions of the current work.

2. NOTATION AND BACKGROUND

2.1 Languages and Automata

Let Σ be an alphabet (set of events) and denote by Σ^* the set of all finite-length strings of elements of Σ (sequences of events), including the empty string ε (the length of a string s is denoted by $|s|$ with $|\varepsilon| = 0$). A language $L \subseteq \Sigma^*$ is a subset of finite-length strings in Σ^* (i.e., sequences of events with the convention that the first event appears on the left). Given strings $s, t \in \Sigma^*$, the string st denotes the concatenation of s and t, i.e., the sequence of events captured by s followed by the sequence of events captured by t. For a string s, $\bar{s}$ denotes the *prefix-closure* of s, and is defined as $\bar{s} = \{t \in \Sigma^* \mid \exists t' \in \Sigma^* \{tt' = s\}\}$. For two strings s and t, we also define $t \in s$, if $\exists t_1, t_2 \in \Sigma^*$, such as $t_1 t t_2 = s$.

Definition 1. (Finite State Automaton (FSA)). A finite state automaton is captured by $G = (X, \Sigma, \delta, X_0, F)$, where $X = \{1, 2, \ldots, N\}$ is the set of states, Σ is the set of events [1], $\delta : X \times \Sigma \to 2^X$ is a nondeterministic state transition function (in the simpler case, where $\delta : X \times \Sigma \to X$, then we call it a deterministic transition function, and the FSA is a deterministic finite automaton or DFA), $X_0 \subseteq X$ is the set of possible initial states [2], and F is the set of Accept states.

For a set $Q \subseteq X$ and $\sigma \in \Sigma$, we define $\delta(Q, \sigma) = \cup_{q \in Q} \delta(q, \sigma)$; with this notation at hand, the function δ can be extended from the domain $X \times \Sigma$ to the domain $X \times \Sigma^*$ in the usual recursive manner: $\delta(x, \sigma s) := \delta(\delta(x, \sigma), s)$ for $x \in X$, $s \in \Sigma^*$ and $\sigma \in \Sigma$ (note that $\delta(x, \varepsilon) := \{x\}$). The behavior of G is captured by $L(G) := \{s \in \Sigma^* \mid \exists x_0 \in X_0 \{\delta(x_0, s) \neq \emptyset\}\}$. We use $L(G, x)$ to denote the set of all traces that originate from state x of G (so that $L(G) = \bigcup_{x_0 \in X_0} L(G, x_0)$).

Definition 2. (Refined Observer). Consider an FSA $G = (X, \Sigma, \delta, X_0, F)$. The *Refined Observer* $G_{obs} = (X_{obs}, \Sigma, \delta_{obs}, X_{obs,0}, F_{obs})$ is constructed as follows:

- The initial state $X_{obs,0} = \{X_0\}$;
- Each state of G_{obs} is associated with a unique subset of states of the original NFA G (so that there are at most $2^{|X|} = 2^N$ states);
- $X_{obs} \subseteq 2^{|X|}$ is the set of states reachable in G_{obs} from the set of initial states $X_{obs,0}$;
- At any state Z' ($Z' \subseteq X$), the next state ($Z = \delta_{obs}(Z', e)$) upon observing an event $e \in \Sigma$ is
  - · $Z = \cup_{x \in Z'} \delta(x, e)$, if $(\forall x \in Z') \to (\delta(x, e) \neq \emptyset)$
  - · Z is undefined, otherwise.
- $F_{obs} = \{Z' \in X_{obs} : (\forall x \in Z') \to (x \in F)\}$

Remark 1. Note that the construction of the Refined Observer presented in this paper, is a modified version of the Observer that is presented in Cassandras and Lafortune (2008). In our case, all events are observable. Moreover, the next state ($\delta_{obs}(Z', e)$) is defined if and only if $\delta(x, e)$ is defined for all states $x \in Z'$. Also, the Accept states of our Observer (F_{obs}) are different than the states presented in Cassandras and Lafortune (2008). In our case, we want all different paths going to the same state $Z \in X_{obs}$ to reach Accept states ($x \in Z \subseteq F$).

3. PROBLEM FORMULATION

Consider a heterogeneous multi-agent team, which consists of two types of agents, Type A: mobile sensors, and Type B: ground mobile robots deployed in a partially known environment partitioned into a grid of finite number of cells. In our setup, an element from the set of labels $\{C, NC, SA, UN\}$, holds true for each cell. The labels describe i) critical areas (C), ii) non-critical areas (NC), iii) safe areas (SA), and iv) unknown areas (UN). In Critical areas (C), one can find human survivors, while in non-critical areas (NC) there is no presence of human survivors. While the mobile sensor moves, it could reveal the label of one or more unknown areas (which can be either noncritical NC, or critical areas C).

Definition 3. (Observation function o). Given a finite number of cells $|X|$, we define the observation map of a cell (i.e., a state $x \in X$), as the function $o : X \to L$, where L is the set of labels, and $o(x)$, for any $x \in X$, is the label of state x.

Assumption: The agents can move from their current cell to any of the adjacent neighboring cells provided that this move is possible on a Finite Transition System, which is a finite abstraction of the dynamics of each agent Ayala et al. (2013). Each agent needs to satisfy a local objective expressed in a syntactically co-safe linear temporal logic formula (scLTL), defined below.

Definition 4. (scLTL syntax)Belta et al. (2017) A (propositional) syntactically co-safe linear temporal logic (scLTL) formula ϕ over a set of observations L (given in Definition 3) is recursively defined as $\phi = \top \mid l \mid \neg l \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \bigcirc \phi \mid \phi_1 \; U \; \phi_2$, where $l \in L$ is a label and ϕ, ϕ_1, and ϕ_2 are scLTL formulas.

Initially agents of type A, and B, have partial knowledge of the environment. Assuming agents of Type B have perfect localization, they can perfectly determine their current cell. Agents of Type B gain information about their environment communicating with an agent of Type A (a mobile sensor). The problem is presented in a simple setup, where there is only one agent A (of Type A), and one agent B (of Type B). Agent A is described below.

3.1 Agent A (of Type A)

Agent A is a mobile sensor, which can identify the presence of survivors in an area. The finite abstraction of the dynamics of agent A is given by the following Finite Transition System.

Definition 5. (Finite Transition System for agent A). $T_A = (X_{T,A}, \Sigma_A, \delta_{T,A}, X_{0,T,A}, L, \mathbf{o_A})$, where $X_{T,A}$ is the

[1] Usually the set Σ of events is partitioned into the set Σ_o of observable events and the set Σ_{uo} of unobservable events. For convenience and simplicity we assume throughout the paper that $\Sigma = \Sigma_o$.

[2] In the next sections we will use, without loss of generality, a unique state x_0 as initial state of an FSA.

set of states, Σ_A is the set of inputs (controls or actions), which in our case is the set of possible movements $\Sigma_A = \{W_1, E_1, S_1, N_1\}$ (meaning accordingly, moving West, East, South, or North), $\delta_{T,A} : X_{T,A} \times \Sigma_A \to 2^{X_{T,A}}$ is the transition function, L is the set of labels of the states $L = \{C, NC, SA, UN\}$ and $\mathbf{o_A}$ is a row vector $\mathbf{o_A} = [o_{A,1}\ o_{A,2}\ \ldots\ o_{A,|X_{T,A}|}]$, where $o_{A,i} = o(x_i) \in L$ is the label of state $x_i \in X_{T,A}$.

Agent A needs to satisfy an scLTL formula (ϕ_a), which is expressed as an FSA Ayala et al. (2013).

Definition 6. The finite state automaton, that expresses the valid behavior conforming to the specification ϕ_a is the tuple $A^{\phi_a} = (Y_{\phi_a}, L, \delta_{\phi_a}, Y_{0,\phi_a}, F_{\phi_a})$, where Y_{ϕ_a} is the finite set of states, $Y_{0,\phi_a} \in Y_{\phi_a}$ is the initial state, L is the set of labels $\{NC, C, SA, UN\}$), $\delta_{\phi_a} : Y_{\phi_a} \times L \to Y_{\phi_a}$ is the transition function, and F_{ϕ_a} is the set of Accept states.

Example 1. Let agents A (mobile sensor) and B (robotic vehicle), moving in the workspace, with finite abstractions of their dynamics given by Finite Transition Systems T_A and T_B drawn in Fig. 1. Each agent (A and B), needs to satisfy an scLTL formula (ϕ_a and ϕ_b) respectively (see Fig. 2).

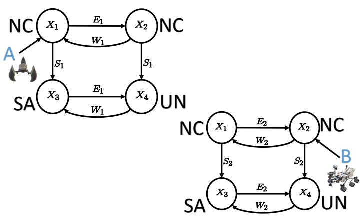

Fig. 1. Finite Transition Systems, T_A (upper left) and T_B (lower right) modeling the dynamics of agents A and B.

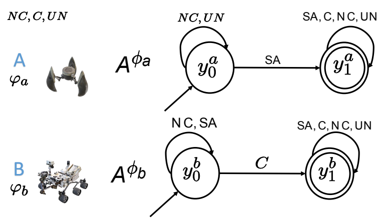

Fig. 2. Finite Automaton A^{ϕ_a} expressing the scLTL formula $\phi_a := (\neg C)\, U(SA)$ for agent A (i.e., agent A should not reach a state with label C before reaching a state with label SA), and Finite Automaton A^{ϕ_b} expressing the scLTL formula $\phi_b := (\neg UN)\, U(C)$ for agent B (i.e., agent B should not reach a state with label UN before reaching a state with label C).

3.2 Update of the Observation function

For an agent of type A, every time a new sensor reading is obtained, the labels of all the cells within the sensor's range are updated. Therefore, agent A is able to identify with accuracy the areas labeled as UN (unknown) corresponding to each cell within the sensor's range. The sensor range function (SR) of an agent of Type A is defined below.

Definition 7. (Sensor Range SR). Given a Finite Transition System $T_A = (X_{T,A}, \Sigma_A, \delta_{T,A}, x_{0,T,A}, L, \mathbf{o_A})$, which captures the dynamics of an agent of Type A, the sensor range of agent A is defined as the function $SR : X_{T,A} \to 2^{|X_{T,A}|}$, where $SR(x)$, for any $x \in X_{T,A}$, is the set of reachable states from state x, after the execution of a single event, $e \in \Sigma_A$. Therefore, we have, $SR(x) = \{x' \in X_{T,A} : \exists e \in \Sigma_A \text{ s.t. } x' \in \delta_{T,A}(x, e)\}$.

The observation function (o) is updated after the execution of a single event ($e \in \Sigma_A$). In general, the updated observation row vector $\mathbf{o^t_A} = [o^t_{A,1}\ o^t_{A,2}\ \ldots\ o^t_{A,|X_{T,A}|}]$ is defined for a sequence of events $t = e_1 e_2 \ldots e_{|t|}$, where for $j \in \{1, \ldots |t|\}$, we have $e_j \in \Sigma_A$. The updated observation $o^t_{A,i}$ describes the updated label for state $x_i \in X_{T,A}$, after the execution of t. The updated observation $o^t_{A,i}$ is different than $o_{A,i}$ (i.e., $o^t_{A,i} \neq o_{A,i}$), if and only if

(1) $o_{A,i} = UN$,
(2) $\exists s$, s.t. $t = ss'$, with $x_s = \delta_{T,A}(x_{0,T,A}, s)$,
(3) and $x_i \in SR(x_s)$.

The updated label will then be $o^t_{A,i} \neq UN$, and it depends on the sensor readings. Finally, the observation function is updated ($o(x_i) \leftarrow o^t_{A,i}$). In any other case, the observation function is not updated ($o(x_i) \leftarrow o_{A,i}$).

Example 1. (continued) We have $\Sigma_A = \{S_1, N_1, E_1, W_1\}$ and the initial knowledge regarding the labels of the states is $\mathbf{o_A} = [NC\ NC\ SA\ UN]$, because for state X_1 we have $o(x_1) = o_{A,1} = NC$, for X_2 we have $o(x_2) = o_{A,2} = NC$, for X_3 we have $o(x_3) = o_{A,3} = SA$, and for state X_4 we have $o(x_4) = o_{A,4} = UN$. Thus, the only label that needs to be updated is the label of state X_4. Assuming that agent A moves South (i.e., executing S_1), then the current state of agent A, is $\delta_{T,A}(X_1, S_1) = X_3$. We also have $SR(X_3) = \{X_3, X_4\}$, which means that the sensor reading is able to update the label of X_4. In our case, the sensor gives us $o^{S_1}_{A,4} = C$, which means that the observation function is updated for state X_4 from $o(X_4) = UN$ to $o(X_4) = C$.

Remark 2. The number of different observation maps $\mathbf{o^t_A}$ is finite, and in the worst case the number of all different observation maps is $|O|^{|X_{T,A}|}$. In our case, one state is labeled as unknown, with two possible updated labels $\{C, NC\}$, which means that there are two possible updated observation maps.

3.3 Agent B (of Type B)

Agent B is a ground mobile robot, which can rescue human survivors, but it cannot sense its environment. The finite abstraction of the dynamics of agent B is given by the following Finite Transition System.

Definition 8. (Finite Transition System for agent B). $T_B = (X_{T,B}, \Sigma_B, \delta_{T,B}, X_{0,T,B}, L, \mathbf{o_B})$, where $X_{T,B}$ is the set of states, $\Sigma_B = \{W_2, E_2, S_2, N_2\}$ is the set of possible movements (meaning accordingly, moving West, East, South, or North), $\delta_{T,B} : X_{T,B} \times \Sigma_B \to 2^{|X_{T,B}|}$ is the transition function, L is the set of labels of the states $L = \{C, NC, SA, UN\}$ and $\mathbf{o_B}$ is a row vector $\mathbf{o_B} = [o_{B,1}\ o_{B,2}\ \ldots\ o_{B,|X_{T,B}|}]$, where $o_{B,i} = o(x_i) \in L$ is the label of the state $x_i \in X_{T,B}$.

Agent B needs to satisfy an scLTL formula (ϕ_b).

Definition 9. The finite state automaton, that expresses the valid behavior conforming the specification ϕ_b is the tuple $A^{\phi_b} = (Y_{\phi_b}, L, \delta_{\phi_b}, Y_{0,\phi_b}, F_{\phi_b})$, where Y_{ϕ_b} is the finite set of states, $Y_{0,\phi_b} \in Y_{\phi_b}$ is the initial state, L is the set of labels, $\delta_{\phi_b} : Y_{\phi_b} \times L \to Y_{\phi_b}$ is the transition function, and F_{ϕ_b} is the set of accepting (final) states.

3.4 Communication between agents A and B

The communication is initiated by agent B and it is event-driven. Therefore, agent B initiates communication with agent A only immediately after the execution of an event (e.g., agent B moves West (W_2)). The decision of the initiation (or not) of a communication with agent A is formulated as an action from the set $Com = \{R, \bar{R}\}$, where R denotes the initiation of communication (a request) and $\bar{R}$ denotes the absence of communication. Agents of type B, initiate a request according to a set of communication protocols which are designed to work locally for each agent of type B. Agent B requests information from A in order to establish the objective of its mission (i.e., rescuing survivors, expressed as ϕ_b). The requested information will be i) the current state of agent A, and ii) the information about the environment, acquired by agent A (i.e., identification of the existence of survivors for areas scanned by agent A). Therefore, agent B requests the updated observation function of agent A, for states ($x_i \in X_{T,B}$), of which it does not have information ($o_{B,i} = UN$ and $o^t_{A,i} \neq UN$).

3.5 Objective

In our setup, one wants to verify if a communication protocol that guarantees the local specification always exists. Exploring this setup, one can verify if there are guarantees about the satisfaction of the objective of agent B. Therefore, one needs to verify that for any finite planned trajectory for agent B, agent B will always be able to achieve its local objective, by acquiring information from agent A. Therefore, one can be certain, that any *a priori* finite planned trajectory, cannot lead agent B to a state, where its local objective will not be feasible (i.e., infeasibility occurs if there are no future trajectories that could establish the objective). This case is important, because irregardless of an initial planning, one can always find a trajectory in an unknown environment, without needing to worry about the possibility of reaching states where the objective is infeasible. It also indicates how robust a solution is, in case of uncertainty (e.g., planning a move W_2 and due to uncertainty, having instead a move E_2). The objective is formally specified as follows: for any sequence of actions $t = t_1 t_2 ... t_n \in L(T_B)$ of agent B of any finite length $n > 0$, there always exists a communication policy $\pi_t \in Com^*$, where $Com = \{R, \bar{R}\}$ s.t. $(\exists t' = t'_1...t'_k$ with $tt' \in L(T_B)) \wedge (\exists \pi_{t'} \in Com^*)$ such that $\delta_{\phi_B}(Y_{0,\phi_B}, o^{\pi_t}_t o^{\pi_{t'}}_{t'}) \in F_{\phi_B}$, where $o^{\pi_t}_t = o^{\pi_t}_{t_1} ..., o^{\pi_t}_{t_n}$, with $o^{\pi_t}_{t_j}$ being the updated observation function $(o(x_j))$ for the state $x_j = \delta_B(x_{0,B}, t_1...t_j)$, after the application of the communication policy π_t ($o^{\pi_{t'}}_{t'}$ is defined similarly).

4. VALID BEHAVIOR OF AGENT A

A product is constructed of i) the Finite Transition system which describes the abstraction of the dynamics of the movements of agent A on the workspace, and ii) the finite automaton which expresses the behavior that validates the syntactically co-safe specification ϕ_a, for agent A. Constructing the product automaton, the behavior of agent A is controlled and the only allowed behavior guarantees the required local specification.

Definition 10. (Controlled Finite State Product Automaton). Given Finite Transition System $T_A = (X_{T,A}, \Sigma_A, \delta_{T,A}, x_{0,T,A}, L, \mathbf{o_A})$ and FSA $A^{\phi_a} = (Y_{\phi_a}, Y_{0,\phi_a}, L, \delta_{\phi_a}, F_{\phi_a})$, we construct $A^P = (X_A, X_{0,A}, \Sigma_A, \delta_A, F_A)$, where

- $X_A = X_{T,A} \times Y_{\phi_a} \times \mathbf{o^t_A}$, is the product of the set of states $x \in X_{T,A}$, $y \in Y_{\phi_a}$, and $\mathbf{o^t_A} = [o^t_{A,1}\ o^t_{A,2}\ ...\ o^t_{A,|X_{T,A}|}]$, after the execution of a sequence of events t.
- $X_{0,A} = X_{0,T,A} \times Y_{0,\phi_a} \times \mathbf{o}^{\varepsilon}_{\mathbf{A}}$, where $\mathbf{o}^{\varepsilon}_{\mathbf{A}}$ are the initial labels, before the execution of any event ($t = \varepsilon$).
- Σ_A is the input alphabet,
- $\delta_A : X_A \times \Sigma_A \to 2^{|X_A|}$ is the transition map, where for $x_A = (x_k, y, \mathbf{o_A})$, we have $\delta_A((x_A, \sigma) = (x_l, y', \mathbf{o'_A}) \in \mathbf{X_A}$ such that $x_l \in \delta_{T,A}(x_k, \sigma), y' = \delta_{\phi_a}(y, o_{A,k}))$, and $\mathbf{o'_A}$ is the updated row vector for the labels for all states $x_j \in X_{T,A}$, where $o'_{A,j} = o_{A,j}$ if $o_{A,j} \in \{C, NC, SA\}$, otherwise, if $o_{A,j} = UN$, and $x_j \in SR(x)$, then $o'_{A,j} \in \{C, NC\}$, is updated according to the observation of x_j after the sensor reading. In case that, $o_{A,j} = UN$, and $x_j \notin SR(x)$, then $o'_{A,j} = UN$.
- $x_{F,A} = (x, y, \mathbf{o_A}) \in F_A$, for $x \in X_{T,A}$ and $y \in F_{\phi_a}$.

4.1 Refining the behavior of agent A according to local specification ϕ_a

Having constructed the product automaton A^P, the behavior that satisfies the local specification ϕ_a is identified. First, the nondeterministic finite automaton A^P is converted to the deterministic Refined Observer $D^P = (X_D, X_{0,D}, \Sigma_A, \delta_D, F_D)$ (see Definition 2). We then take the Co-accessible automaton of D^P Cassandras and Lafortune (2008), which refines the language of D^P removing all states, from which there are no reachable accept states, and is given by $A^{P'} = CoAc(D^P) = (X'_A, X_{0,A}, \Sigma_A, \delta'_A, F'_A)$. Finally, the states of $A^{P'}$ are unfolded to produce its language equivalent nondeterministic automaton, which is called A^{r_A}. Automaton $A^{r_A} = (X^{r_A}, X_{0,A}, \Sigma_A, \delta^{r_A}, F_A)$ is a refined version of A^P, with less states ($X^{r_A} \subseteq X_A$) and transitions (for a state $x \in X^{r_a}$ and $e \in \Sigma_A$, always exists $\delta^{r_A}(x, e) \subseteq \delta_A(x, e)$).

5. VERIFICATION OF THE EXISTENCE OF A COMMUNICATION PROTOCOL

Agent's B specification is also expressed in scLTL, and described as a finite automaton (see Fig. 2). Therefore, because of the partial knowledge about the environment, and lack of sensing abilities, agent B needs to request information from agent A in order to fulfill its own objective. For that reason, a verification algorithm is developed, that answers to the question: "does there exist a communication policy, such that agent B can move without restrictions in its workspace, and at the same time guarantees that there always exists a path that satisfies ϕ_b?"

5.1 Verification Algorithm

A Verifier is constructed, to identify if there exists a communication policy under which agent B, can establish its own objective. There are two phases and types of states. In the first phase, agent B controls the initiation of communication. The movement of agent B is not considered as a controlled behavior in this case (although it is an observable behavior), because we are interested in proving the existence or not of a communication policy, for any possible movement of agent B. In the second phase, agents A and B execute a move. Our objective is to verify that for any possible action sequence (movements of agents A and B), one can find a communication policy, so that the objective of agent B is validated. The construction of the Verifier is presented below.

Definition 11. (Verifier). The Verifier $VI = (X_v, \Sigma_v, \delta_v, X_{o,v}, F_v)$, is described below:

- X_v is a set formed by the Cartesian product $X_A \times \mathbb{P}(X_A) \times X_B \times \Sigma_v\{r, m\}$ and $x_v \in X_V$ is a quintuple of the form $(x_A, x_A^B, x_B, e_B, turn)$ such that
  - · $x_A \in X^{r_A}$ is the current-state of agent A
  - · $x_A^B \in \mathbb{P}(X_A)$ is the estimate of agent B about the current-state of agent A.
  - · $x_B = (x_m, y^B, \mathbf{o_B})$ is the current-state of agent B. In case that $y^B \in F_{\phi_B}$, then $x_v \in F_v$.
  - · $e_B \in \Sigma_B$ is the event that will be executed next by agent B.
  - · $turn \in \{r, m\}$, where r indicates the first phase (rectangle-shaped states, drawn in Fig. 3) and m indicates the second phase (elliptical-shaped states drawn in Fig. 3).
- $X_{0,v}$ describes the initial set of states, where $(x_A, x_A, x_B, e_B, r) \in X_{0,v}$ if $x_A \in X_{0,A}$, $x_B = (x_m, y^B, \mathbf{o_B})$, where $x_m \in X_{0,T,B}$, $y^B \in Y_0^B$, $\mathbf{o_B} = \mathbf{o_A^\epsilon}$ and $e_B \in \Sigma_B$ s.t. $\exists x'_B = (x_n, y'^B, \mathbf{o'_B})$ with $x_n \in \delta_{T,B}(x_m, e_B)$, and $y'^B \in \delta_{\phi_A}(y^B, o_{B,m})$.
- The event set is $\Sigma_v = \{R, \bar{R}, p\}$, where R or $\bar{R}$ describes the initiation or not of a communication request, and p describes all possible moves for agent A (a move is an event in Σ_A).

- Step 1. Initiation or not of communication by agent B: The Verifier moves via event R or $\bar{R}$. The execution of these events formulate the initiation or not of communication by agent B. In case that agent B executes R (initiation of communication), this is formally expressed as $\delta_v((x_A, x_A^B, x_B, e_B, r), R) = ((x_{A'}, x_A^{B'}, x_{B'}, e_{B'}, m)$, where
  - · $x_{A'} = x_A = (x_k, y^A, \mathbf{o_A})$
  - · $x_{A'}^B = \{x_A\}$
  - · $e_{B'} = e_B$
  - · $x_{B'} = (x_n, y^{B'}, \mathbf{o_{B'}})$, with
    $x_n = x_m$
    $y^{B'} = y^B$
    $\mathbf{o_{B'}} = \mathbf{o_A}$

  In case that agent B executes $\bar{R}$ (agent B does not initiate communication), this is formally expressed as $\delta_v((x_A, x_A^B, x_B, e_B, r), \bar{R}) = ((x_{A'}, x_A^{B'}, x_{B'}, e_{B'}, m)$, where
  - · $x_{A'} = x_A = (x_k, y^A, \mathbf{o_A})$
  - · $x'^B_A = \cup_{x^{r_A} \in X_A^B} R(x^{r_A})$, where $R(x^{r_A})$ is simply the set of reachable states in A^{r_A}, starting from state x^{r_A}.
  - · $e_{B'} = e_B$
  - · $x_{B'} = x_B$.
- Step 2. Agents A and B move in parallel, where agent A moves with all possible sequences of moves $s \in \Sigma_A^*$ and agent B is making a single move $e_2 \in \Sigma_B$. All these pairs of moves by both agents, are described symbolically by a single event $p \in \Sigma_v$ in the Verifier. This is formally expressed as $\delta_v((x_A, x_A^B, x_B, e_B, m), p) = ((x_{A'}, x_A^{B'}, x_{B'}, e_{B'}, r)$, where
  - · $x_{A'} \in R(x_A)$
  - · $x'^B_A = x_A^B$
  - · $x'_B = (x_n, y'^B, \mathbf{o'_B})$, with
    $x_n \in \delta_{T,B}(x_m, e_B)$
    $y'^B \in \delta_{\phi_B}(y^B, o_{B,m})$
    $\mathbf{o'_B} = \mathbf{o_B}$
  - · $e'_B \in \{e \in \Sigma_B : (\delta_{T,B}(x_m, e) \neq \emptyset) \wedge (\delta_{\phi_B}(y^B, o_{B,m}) \neq \emptyset)\}$
- Step 3. Steps 1 and 2, are repeated until we finish the construction of the Verifier. □

The verification continues by removing the "blocking states". The algorithm runs recursively to find all blocking states, starting from the set of immediately blocking states (i.e., the set of states with no outgoing transitions at all, or states where there is no path reaching an Accept state from that state). Having formulated the problem into a supervisory control one, the problem of characterizing and pruning the "blocking states" is solved as an instance of the Basic Supervisory Control Problem-Nonblocking Case (BSCP-NB), in the terminology of Cassandras and Lafortune (2008). In the end, after the removal of all blocking states, the Verifier should satisfy that "the initial states of the Verifier are not "blocking states". The proof is a straightforward outcome of the way that the blocking states are defined and removed applying the BSCP-NB formulation.

Remark 3. A simpler Verifier could model only the initiations of communication requests ($C = \{R\}$) after any possible move of agent B. However, the Verifier presented here can be easily extended to capture the complexity of a general multi-agent system. We note that our Verifier cannot be used to synthesize a communication protocol (other than the protocol which initiates all communication requests after the execution of any event). In order for the Verifier to be used to synthesize a protocol, it needs to combine all possible states of agent A into a set of states. This extension of the Verifier is left for a future work.

Example 2. Example 1 is now modified in order to illustrate a case, where one can choose a communication policy initiated by agent B, for every possible plan of action for both agents A and B. Example 1 illustrates another scenario for the unknown labeled areas. In that case, agent A i) starts at the initial state x_2 and ii) knows a priori, that the unknown areas cannot be noncritical areas. However, if agent A cannot pinpoint the exact location of the survivors in an unknown area, then it does not update the label of this area, because it cannot provide useful information to an agent B, responsible for the rescue mission. Therefore, taking $A^{P'}$ which is the co-accessible part of the Refined Observer of A^P, and unfolding it, we have A^{r_A}. The solution of the problem continues by constructing the Verifier drawn in Fig. 3.

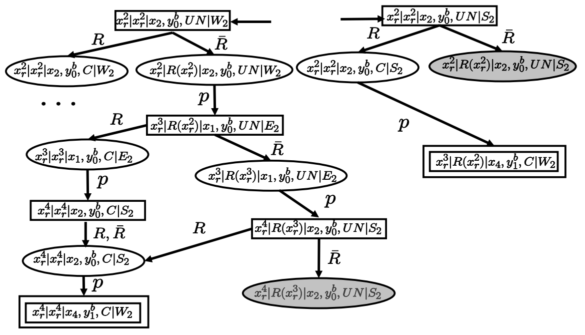
$x_r^2|x_r^2|x_2, y_0^b, UN|W_2$
R
$\bar{R}$
$x_r^2|x_r^2|x_2, y_0^b, C|W_2$
$x_r^2|R(x_r^2)|x_2, y_0^b, UN|W_2$
$\ldots$
p
$x_r^3|R(x_r^2)|x_1, y_0^b, UN|E_2$
R
$\bar{R}$
$x_r^3|x_r^3|x_1, y_0^b, C|E_2$
$x_r^3|R(x_r^3)|x_1, y_0^b, UN|E_2$
p
$x_r^4|x_r^4|x_2, y_0^b, C|S_2$
p
$x_r^4|R(x_r^3)|x_2, y_0^b, UN|S_2$
$R, \bar{R}$
R
$\bar{R}$
$x_r^4|x_r^4|x_2, y_0^b, C|S_2$
$x_r^4|R(x_r^3)|x_2, y_0^b, UN|S_2$
p
$x_r^4|x_r^4|x_4, y_1^b, C|W_2$
$x_r^2|x_r^2|x_2, y_0^b, UN|S_2$
R
$\bar{R}$
$x_r^2|x_r^2|x_2, y_0^b, C|S_2$
$x_r^2|R(x_r^2)|x_2, y_0^b, UN|S_2$
p
$x_r^3|R(x_r^2)|x_4, y_1^b, C|W_2$

Fig. 3. A part of the Verifier for Example 2, before removing the "immediately blocking states" (the states in gray), which are the only "blocking states" in this Example.

Finally, there are different communication protocols that agent B can implement based only on local information. Let us say that the initial state is $(x_r^2, x_r^2, (x_2, y_0^b, UN), W2)$, this means that the next executed event by agent B, will be W_2 (moving West). In this example, one can synthesize a protocol, even without knowing the exact state of agent A (Remark 3 does not apply in this case), because agent A has a complete knowledge of its surrounding environment from its initial state x_r^2 (i.e., the state of agent A is not important). Therefore, agent B will take the same decision for any possible state that agent A lies in. Let us assume that agent B decides to not request information from agent A, which means that $\bar{R}$ is executed. The Verifier goes to state $(x_r^2, R(x_r^2), (x_2, y_0^b, UN), W_2)$ from which the pair (t_1^1, e_2^1) is executed, where $t_1^1 \in \Sigma_1^*$, and $e_2^1 = W_2$ are symbolically represented as p in the Verifier. After the execution of p, the Verifier is reaching a state of the form $(x_r^k, R(x_r^2), (x_1, y_0^b, UN), E_2)$, where $x_r^k \in R(x_r^2)$. For illustration purposes let us say that $x_r^k = x_r^3$ (in Fig. 3, w.l.o.g., the only case which is drawn, is the one where the next executed event is E_2). Again, agent B does not request any information from agent A, so $\bar{R}$ is executed, and the Verifier reaches the state $(x_r^3, R(x_r^3), (x_1, y_0^b, UN), E_2)$. Then again some p is executed, which symbolically captures the pairs (t_1^2, e_2^2), where $t_1^2 \in \Sigma_1^*$, and $e_2^2 = E_2$, and the state $(x_r^4, R(x_r^3), (x_2, y_0^b, UN), S_2)$ is reached. From that state, agent B requests information from agent A, because otherwise the Verifier would be reaching a blocking state (the state in grey, drawn in Fig. 3). Finally, for any executed sequence of events by agent A, $t = t_1^1 t_1^2 t_1^3 \in \Sigma_1^*$, and for $W_2 E_2 S_2$ executed by agent B, there exists a legal sequence of communication requests which is given as $\bar{R} \underbrace{p}_{(t_1^1, W_2)} \bar{R} \underbrace{p}_{(t_1^2, E_2)} R \underbrace{p}_{(t_1^3, S_2)}$.

6. CONCLUSION

We have considered the problem of verifying the existence or not of a communication policy in a multi-agent system to guarantee temporal logic specifications, in the form of an scLTL formula. We proposed a verification algorithm and we provided appropriate constructions from which a communication policy can be extracted for all possible initial motion plans for each agent. For the future, we are interested in developing a method to obtain an optimal communication policy according to a given criterion (e.g., a cost function). We are also interested in extending the aforementioned problem in various directions, i) extending it to a more general temporal logic formulation (e.g., LTL), ii) extending it to a probabilistic setting (Markov decision processes), and iii) having timed constraints synthesizing a both time and event-driven communication protocol.

REFERENCES

Ayala, A.I.M., Andersson, S.B., and Belta, C. (2013). Temporal logic motion planning in unknown environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5279–5284.

Belta, C., Yordanov, B., and Aydin Gol, E. (2017). Formal methods for discrete-time dynamical systems. In *Studies in Systems, Decision and Control*, volume 89. Springer.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems.* Springer Publishing Company, Incorporated, 2nd edition.

Debouk, R., Lafortune, S., and Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, (12), 33–86.

Debouk, R., Lafortune, S., and Teneketzis, D. (2003). On the effect of communication delays in failure diagnosis of decentralized discrete event systems. *Discrete Event Dynamic Systems: Theory and Application*, (13), 263–289.

Fainekos, G.E., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2), 343 – 352.

Hoxha, B. and Fainekos, G. (2016). Planning in dynamic environments through temporal logic monitoring. In *AAAI Workshops.*

Keroglou, C. and Hadjicostis, C.N. (2018). Distributed fault diagnosis in discrete event systems via set intersection refinements. *IEEE Transactions on Automatic Control*, 63(10), 3601–3607.

Kress-Gazit, H., Fainekos, G.E., and Pappas, G.J. (2009). Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6), 1370–1381.

Lahijanian, M., Andersson, S.B., and Belta, C. (2012). Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics*, 28(2), 396–409.

Meng Guo, Johansson, K.H., and Dimarogonas, D.V. (2013). Revising motion planning under linear temporal logic specifications in partially known workspaces. In *2013 IEEE International Conference on Robotics and Automation*, 5025–5032.

Nenchev, V. and Belta, C. (2016). Receding horizon robot control in partially unknown environments with temporal logic constraints. In *2016 European Control Conference (ECC)*, 2614–2619.

Yin, X. and Lafortune, S. (2018). Minimization of sensor activation in decentralized discrete-event systems. *IEEE Transactions on Automatic Control*, 63(11), 3705–3718.