Reactive Switching Protocols for Multi-Robot High-Level Tasks

Vasumathi Raman[1]

Abstract— **This paper considers a team of cooperative, homogeneous robots operating in a nondeterministic environment, performing complex high-level tasks. We concurrently solve the problems of dynamic task assignment and reactive planning in a centralized fashion, allowing goal reassignment as needed in response to changes in the environment. To this end, we model the team of robots as a switched system whose various modes must be activated to satisfy high-level specifications, and describe a formal framework for synthesizing switching protocols for this system. The robots' continuous dynamics under a specific task allocation, as well as their operating environment, are encoded as formulas on a discrete abstraction, in the GR(1) fragment of linear temporal logic. The synthesized protocols ensure that the multi-robot task is fulfilled even in adversarial environments, and the presented abstraction provides a significant computational improvement over previous approaches. We illustrate these ideas in simulation, with applications to intrusion detection and surveillance.**

I. Introduction

We consider multi-robot applications where teams of robots must operate in a nondeterministic environment to accomplish complex high-level behaviors. Examples include search-and-rescue and surveillance missions, where individual robots must be deployed to different parts of a building or other workspace to search for victims or look for intruders. In most cases, the specific task assigned to each robot is unimportant, as long as every task is accomplished; we thus consider a team of cooperative, homogeneous robots. Our main contribution is a framework for protocol synthesis for such multi-robot systems performing high-level tasks described in Linear Temporal Logic (LTL).

Previous work on multi-robot task allocation, including market-based approaches [5], does not address the problem of nondeterminism in the environment. Recent work has solved the task allocation and planning problem concurrently in a centralized fashion [12], but does not consider tasks involving nondeterminism, patrol-type behaviors and reaction to external events. The authors in [3] provide a partially decentralized solution to multi-robot coordination in partially-known environments, where the problems of task assignment, trajectory planning and safe control are concurrently solved in the presence of communication constraints. Our work extends the class of high-level robot tasks accommodated, allowing reactive temporal logic specifications rather than just a goal configuration for the team. In contrast with [3], we explore a centralized approach, but discuss possible steps towards decentralization in future work.

Unlike previous temporal logic synthesis approaches such as [7], [6], which deal with multi-robot path planning, we consider reactive behaviors, wherein the system must respond to gathered sensory information. Reactive synthesis for LTL enables us to handle a wide variety of tasks, beyond mere safety and reachability. While LTL synthesis is in general prohibitively expensive, the Generalized Reactivity (1) (GR(1)) fragment admits synthesis with manageable computational complexity [9]. This paper builds on recent work on synthesis of reactive switching protocols [8], where the objective was to determine a control strategy for switching amongst a family of differential equations to satisfy a GR(1) specification. Similar to [8], the only control input in our work is the mode of the switched system, which codifies the task allocation. However, while the abstractions considered in [8] are over-approximations of the dynamics, we focus on deterministic abstractions where every discrete transition can be implemented in the physical domain.

By construction, the existence of a solution to the discrete synthesis problem we present guarantees the existence of a switching protocol that, when implemented at the continuous level, ensures that the multi-robot system behaves as specified at runtime. The synthesized protocols can react to possibly adversarial environment events, including exogenous disturbances on the continuous dynamics of the robots. We describe how we can combine our high-level plans with existing multi-robot feedback controllers such as those in [3], [2], and see that concurrent task reassignment and planning allows the solution of otherwise infeasible problems. We illustrate these ideas in simulation, with applications to region surveillance in response to the detection of an intruder.

II. Problem Statement

We first introduce the notation and models used, and formalize the problem addressed. Much of the notation mirrors [8] and [3].

A. Linear Temporal Logic (LTL)

Syntax: Let Π be a set of atomic propositions. LTL formulas are defined by the recursive grammar:

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

where $\pi \in \Pi$, $\neg$ is negation, $\vee$ is disjunction, $\bigcirc$ is "next", and $\mathcal{U}$ is a strong "until". Conjunction ($\wedge$), implication ($\Rightarrow$), equivalence ($\Leftrightarrow$), "eventually" ($\Diamond$) and "always" ($\Box$) are derived from these operators.

*This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

[1]V. Raman is with the California Institute of Technology, Pasadena, CA 91125, USA vasu@caltech.edu

Semantics: The truth of an LTL formula is evaluated over infinite sequences of states, which can be thought to represent executions of a finite state machine representing the system. A state corresponds to an assignment of truth values to all propositions $\pi \in \Pi$. Given an infinite sequence of these states σ, the statement $\sigma \models \varphi$ denotes that σ satisfies the formula φ at the first position. The statement $\sigma \models \Delta\varphi$ for $\Delta = (\bigcirc, \Box, \Diamond, \Box\Diamond)$ denotes that φ is satisfied at the second position, at every position, at some position, and infinitely often in σ, respectively. Let A be a finite state machine whose states are labeled with truth assignments to Π. Then A is said to satisfy φ if, for every execution σ of A, $\sigma \models \varphi$. The reader is referred to [4] for a formal definition of the semantics.

B. Multi-Robot Switched Systems

Consider a team of N robots $\mathcal{A} = \{a_i \mid i = 1, ..., N\}$ in a shared, bounded, nondeterministic environment with N distinct tasks. The team has the configuration or state $x(t) = [x_1 x_2 ... x_N], x_i \in X_i \subset \mathbb{R}^d$, with dynamics

$$\dot{x}_i(t) = u_i(t), i = 1, ..., N. \tag{1}$$

Here $x(t) \in X \subseteq \mathbb{R}^{Nd}$ is the system state at time t.

Let $\mathcal{G}$ be a set of goal configurations; for simplicity, we assume that $|\mathcal{G}| = N$ goals, and at any given time, each robot is assigned a unique goal[1]. We represent the task assignment as a permutation $p \in \mathcal{P}_N$, where $\mathcal{P}_N$ is the set of all bijections of $\{1, ..., N\}$ onto itself: $p = p_1 p_2 ... p_N$ assigns task $p_i \in \{1, 2, ..., N\}$ to agent i. We allow all permutations of the tasks on the team of robots (i.e. any robot can complete any task). The resulting switched system dynamics are

$$\dot{x}(t) = f_{\sigma(t)}(x(t)), \tag{2}$$

where $f_{\sigma(t)}$ refers to a distinct piecewise smooth control input for each permutation $\sigma(t)$.

Here, we view σ as a switching signal taking values in $\mathcal{P}_N$; $\sigma(t)$ may depend on $x(t)$ as well as t. Given a set of initial states $X_0 \subseteq X$ and initial modes $\mathcal{P}_0 \subseteq \mathcal{P}_N$, *solutions* to (2) are pairs (x, σ) that satisfy (2) for all $t \geq 0$ and $(x(0), \sigma(0)) \in X_0 \times \mathcal{P}_0$. In the above formulation, σ is the only controllable variable. Each mode represents a set of pre-designed behaviors for the robot team, i.e. motion towards pre-specified goals. Our objective is to automatically synthesize a multi-robot switching protocol such that solutions of the resulting switched system (2) satisfy (by construction) a given LTL specification. If a change in the environment is detected, the system can react by possibly switching modes; the resulting behavior should satisfy the LTL specification.

We now define the satisfaction of LTL formulas by continuous signals, following the notation in [8]. Let X be a set of states and Π_X be a finite set of propositions: each $\pi_X \in \Pi_X$ is characterized by a subset of X in which it is true, and the set of all possible subsets of Π_X is denoted 2^{Π_X}. Let $h_X : X \to 2^{\Pi_X}$ be an *observation map* such that $h_X(x) = \{\pi_X \in \Pi_X : x \in \pi_X\}$, i.e., $h_X(x)$ is the set of propositions in Π_X that hold in x. A *continuous-time signal* is a function $s : \mathbb{R}^+ \to X$. Intuitively, the *word* generated by a signal is the sequence of sets of propositions satisfied as it evolves over time, as obtained from the observation map h_X; see Definition 4 in [8] for a formal definition. The signal s satisfies an LTL formula φ, written $s \models \varphi$, if and only if the word it produces satisfies φ.

[1]If there are more goal regions than robots, then the task assignment is no longer a permutation, but an injection from robots to goals. If there are fewer goal regions than robots, we can add uniquely indexed but identical new goals representing the disjunction of all the regions in the workspace. Both cases are easily accommodated in our approach

C. Environment

In this paper, we use "environment" to refer to the factors that are relevant to the operation of the multi-robot system, but are not part of the dynamics modeled in (1) or (2). These include potentially adversarial factors not controlled by the system, e.g., hazards, doors, traffic lights, etc. Formally, we assume that the environment state can take values in some set Z, and that there exists an observation map $h_Z : Z \to \mathcal{E}$ which maps states in Z to some finite set $\mathcal{E}$ of observations. Real-time properties of the environment are captured using *environment signals*, i.e. functions $\zeta : \mathbb{R}^+ \to Z$.

Following [8], we define an *environment transition system* $\mathcal{T}_e := (\mathcal{E}, \mathcal{E}_0, \to)$ that captures all possible changes in the environment; this will let us encapsulate our knowledge about the allowable environment behavior in the specification as an environment assumption. Here $\mathcal{E}$ is the aforementioned finite set of observations, $\mathcal{E}_0 \subseteq \mathcal{E}$ is a set of initial observations, and $\to \subseteq \mathcal{E} \times \mathcal{E}$ is a transition relation such that $e \to e'$ if and only if $e \neq e'$ and there exists some environment signal ζ and some $\tau > 0$ such that $h_Z(\zeta(\tau)) = e'$ and $\lim_{t \to \tau^-} h_Z(\zeta(t)) = e$. We also assume that we are given a set of environment propositions $\Pi_\mathcal{E}$ and a labeling $\mathcal{L}_e : \mathcal{E} \to 2^{\Pi_\mathcal{E}}$; intuitively, $\mathcal{L}_e$ labels $e \in \mathcal{E}$ with propositions that are true in that state.

We can now define the trajectory of a multi-robot switched system operating in and reacting to its environment:

Definition 1 (Definition 6 in [8]): A *trajectory* $s : \mathbb{R}^+ \to X \times Z \times \mathcal{P}_N$ of the multi-robot switched system (2) and its environment is a tuple $s = (x, \zeta, \sigma)$, where (x, σ) are solutions to (2) and ζ is an environment signal.

The finite set of atomic propositions relevant to this system is $\Pi := \Pi_X \times \Pi_\mathcal{E} \times \mathcal{P}_N$, where Π_X is a finite set of propositions defined over the system state space X; the observation map is defined by $h(x, z, p) = (h_X(x), \mathcal{L}_e(h_Z(z)), \{p\})$.

Problem 1 (Multi-Robot Switching Synthesis): Given a family of multi-robot controllers satisfying (1), a set of goals $\mathcal{G}$, permutations $\mathcal{P}_N$, an environment transition system $\mathcal{T}_e$, and an LTL specification φ of the form $\varphi = (\varphi_e \Rightarrow \varphi_s)$, synthesize a reactive switching protocol σ that generates only trajectories $s = (x, \zeta, \sigma)$ such that $s \models \varphi$.

In the above formulation, φ_e encodes the assumptions on admissible environment behavior, and φ_s encodes the desired behavior of the system. Note that this problem statement is in contrast with [3], where the authors seek a finite sequence of switches that will guarantee that the system will achieve a desired goal configuration: Problem 1 requires a reactive

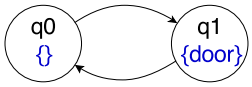


Fig. 1: Transition system $\mathcal{T}_e$ for an environment with one door, which is initially open and can close or reopen at any time thereafter (there are implicit self-loops on each state). Here $\mathcal{E} = \{q_0, q_1\}$, $\mathcal{E}_0 = \{q_0\}$, $\rightarrow = \{(q_0, q_0), (q_0, q_1), (q_1, q_0), (q_1, q_1)\}$,$\mathcal{L}_e(q_0) = \{\}$, and $\mathcal{L}_e(q_1) = \{\text{door}\}$

protocol whose infinite executions fulfill an LTL formula, including safety conditions in addition to the goals.

In Section III, we solve Problem 1 via the following steps:

- defining a discrete abstraction of the system dynamics (1) for each permutation $p \in \mathcal{P}_N$, and transitions between them based on which switches can be implemented,
- formulating a discrete synthesis problem on this finite abstraction and the LTL task specification,
- synthesizing a switching protocol by solving the discrete synthesis problem, and
- implementing this switching protocol using continuous controllers for each mode.

III. Approach

In this section, we describe our solution to Problem 1, elaborating on the steps listed in Section II, and discussing its correctness and computational complexity. We demonstrate the stages of the synthesis procedure using an example of robots patrolling a workspace for intruders.

A. Discrete Abstraction

The relevant features of the continuous multi-robot control problem are abstracted using a finite set of Boolean propositions. To capture the motion of the robots using the discrete formalism of LTL, the workspace is partitioned into regions, and Boolean propositions introduced to indicate each robot's location. Additional propositions capture other properties of and events in the environment including, for example, the presence of obstacles such as doors.

Given an environment model $\mathcal{T}_e$, a set of workspace regions $\mathcal{R}$, sets of robot agents $\mathcal{A}$ and goal regions $\mathcal{G} \subseteq \mathcal{R}$ each indexed by N, the discrete abstraction consists of:

- every environment proposition $\pi_e \in \Pi_{\mathcal{E}}$. Consider the example environment transition system in Fig. 1. Then proposition π_{door} is true if and only if the door is closed (i.e. the environment is in state q_1).
- π_{i_r} to indicate the presence of robot i in region $r \in \mathcal{R}$ (e.g., $\pi_{1_{hall}}$ is true if and only if robot 1 is in the hall) – here $\Pi_X = \{\pi_{i_r} \mid i \in \{1, ..., N\}, r \in \mathcal{R}\}$.
- π_p for every permutation $p \in \mathcal{P}_N$ (e.g., if there are 3 goal regions and 3 robots, then π_{123} is true if and only if each robot i is assigned to region i for $i = 1, 2, 3$).

We partition these propositions into a set of environment propositions $\mathcal{X}$, and a set of robot-controlled propositions $\mathcal{Y}$. The only robot-controlled variables in this setting are the permutations $p \in \mathcal{P}_N$. Thus, $\mathcal{Y} = \mathcal{P}_N$, and $\mathcal{X} = \Pi_{\mathcal{E}} \cup \Pi_X$.

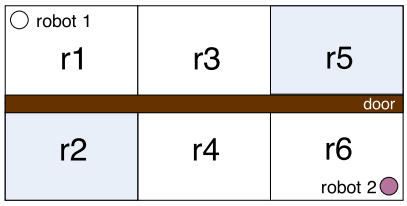


Fig. 2: Workspace for Example 1 with initial position of the robots marked and regions of possible intrusion shaded blue

Note that, as proposed in [10], we treat the location of each robot as a sensed event, controlled by the environment. We will add assumptions on how the environment can evolve given an input (mode); nondeterministic disturbances in the robot dynamics can be accommodated here.

We present a simple example with two robots for clarity of presentation – the method extends to larger teams of robots.

Example 1: Consider the six-room workspace depicted in Fig. 2, where the rooms are labeled r_1−r_6. There is a door running down the length of the workspace, separating the top rooms from the bottom. There are two robots, indicated by circles: robot 1 starts in r_1 and robot 2 starts in r_6. Each robot has a sensor, that can sense whether the door is open; if the door is closed, the robots cannot move from one side of it to the other. In addition, each robot can sense the presence of an intruder in each of r_2 or r_5, which are both designated "intrusion-sensitive" regions. The task specified is that when an intruder is detected in one of these two regions, either one of the robots goes to that region to investigate.

For simplicity, we assume that the door and intrusion-detection sensors of both robots are identical, i.e. they can be modeled by a single variable. There are two goal regions in this task: $\mathcal{G} = \{r_2, r_5\}$, so $\mathcal{P}_N = \{12, 21\}$. Here,

- $\mathcal{X} = \{\pi_{intruder_2}, \pi_{intruder_5}, \pi_{door}\}$
  $\cup \{\pi_{i_r} \mid i \in \{1, 2\}, r \in \{1, 2, 3, 4, 5, 6\}\}$
- $\mathcal{Y} = \{\pi_p \mid p \in \{12, 21\}\}$: e.g. 12 indicates that robot 1 is assigned goal r_2 and robot 2 is assigned goal r_5.

B. Task Specification

Given a discrete abstraction, a high-level task is specified using an LTL formula over $\mathcal{X} \cup \mathcal{Y}$. The specification governs which modes the system can activate in each state, and assumptions on how the environment-controlled variables evolve. A specification can contain two types of properties: *safety* properties, which guarantee that "something bad never happens," and *liveness* conditions, which state that "something good (always eventually) happens". We consider tasks specified as GR(1) formulas, i.e. of the form $\varphi_e \Rightarrow \varphi_s$, where φ_s represents the desired multi-robot behavior, and φ_e encodes assumptions on the environment, as perceived by the robots' sensors. Each of φ_e and φ_s contains initial $(\varphi_e^i, \varphi_s^i)$, safety $(\varphi_e^t, \varphi_s^t)$ and liveness conditions $(\varphi_e^g, \varphi_s^g)$.

The following is an excerpt[2] of the user-defined LTL specification for the task in Example 1:

[2]Some parts of the specification have been eliminated for clarity. E.g., note that since the progress each robot's motion in the workspace is independent, in order for the above specification to be realizable, the door is assumed to never close while the robots are on the same side of it.

$(\varphi_{1_{r_1}} \wedge \varphi_{2_{r_6}})$ #Initial (Environment)
(Robot 1 starts in r1, Robot 2 in r6)
$(\neg\pi_{intruder_2} \wedge \neg\pi_{intruder_5} \wedge \neg\pi_{door})$ #Initial (Environment)
(Initially no intruders, open door)
(π_{12}) #Initial (System)
(Robot 1 is initially assigned goal r_2,
Robot 2 is initially assigned goal r_5)
$\wedge \quad \square(\varphi_{1_{r_1}} \wedge \bigcirc \pi_{door} \Rightarrow \bigcirc \neg\varphi_{1_{r_2}})$ #Safety (Environment)
(If door closed, Robot 1 can't move from r_1 to r_2)
$\wedge \quad \square(\varphi_{1_{r_2}} \wedge \bigcirc \pi_{door} \Rightarrow \bigcirc \neg\varphi_{1_{r_1}})$ #Safety (Environment)
(If door closed, Robot 1 can't move from r_2 to r_1)
...
$\wedge \quad \square(\varphi_{2_{r_5}} \wedge \bigcirc \pi_{door} \Rightarrow \bigcirc \neg\varphi_{2_{r_6}})$ #Safety (Environment)
(If door closed, Robot 2 can't move from r_5 to r_6)
$\wedge \quad \square(\varphi_{2_{r_6}} \wedge \bigcirc \pi_{door} \Rightarrow \bigcirc \neg\varphi_{2_{r_5}})$ #Safety (Environment)
(If door closed, Robot 2 can't move from r_6 to r_5)
$\wedge \quad \square\diamondsuit(\pi_{intruder2} \Rightarrow \bigcirc(\varphi_{1_{r_2}} \vee \varphi_{2_{r_2}}))$ #Liveness (System)
(If an intruder is detected in r_2,
either Robot 1 or 2 should go to r_2)
$\wedge \quad \square\diamondsuit(\pi_{intruder5} \Rightarrow \bigcirc(\varphi_{1_{r_5}} \vee \varphi_{2_{r_5}}))$ #Liveness (System)
(If an intruder is detected in r_5,
either Robot 1 or 2 should go to r_5)

Since each robot can be in exactly one location at any given time, the formula $\varphi_{i_r} = \pi_{i_r} \wedge \bigwedge_{r' \neq r} \neg\pi_{i'_r}$ represents robot i being in region r. In addition to the user-defined specification, there are several constraints automatically generated depending on the multi-robot motion controllers in each mode of the switched system. These components of the specification are now described in detail.

1) Multi-Robot Motion Controllers: The robot team's motion in the workspace is governed by the availability of controllers to drive it to its current goal configuration. We can use the approach presented in [3], [2] to synthesize navigation functions for driving the team of robots from any current configuration to the goal configuration for each permutation $\sigma \in \mathcal{P}_N$, based on the location of the other robots and obstacles in the workspace. The approach involves decomposing the configuration space into obstacle-free polytopes (where each polytope is associated with a region combination based on where each robot is), and generating local smooth feedback laws that drive the team of robots from one cell to an adjoining one. These local controllers are then sequenced using discrete graph search methods like A* or incremental D* to reach the goal. The method is complete, i.e. it is guaranteed to find a solution if one exists, i.e. if every robot can achieve its current goal. These atomic controllers can also be switched between immediately in response to changes in the environment.

2) Sensor Safety Assumptions: The progress of the linear feedback controllers for each robot is reflected in the discrete abstraction in the form of safety constraints on where the robots can move, given which mode is activated. So activating the mode that drives robot 1 to goal r_2 and robot 2 to goal r_5 will give rise to constraints on the robot's current and next location while the motion controller for this mode is active. Therefore, given a goal configuration for the team of robots, the atomic controller corresponding to the current mode results in constraints on the discrete environment-controlled variables indicating the region for each robot. Regardless of the mode, we include the constraint that, at each time step, each robot can only move between adjacent rooms or stay in place. There are also proximity constraints that enforce that the robots are not in the same room as each other. The following safety conditions would result for the workspace in Fig. 2:

$$\begin{aligned}\varphi_{motion} = \quad & \text{\# adjacency for robot } i,\ i \in \{1,2\} \\ & \square(\varphi_{i_{r_1}} \Rightarrow (\bigcirc \varphi_{i_{r_1}} \vee \bigcirc \varphi_{i_{r_2}} \vee \bigcirc \varphi_{i_{r_3}})) \\ & \wedge \square(\varphi_{i_{r_2}} \Rightarrow (\bigcirc \varphi_{i_{r_1}} \vee \bigcirc \varphi_{i_{r_2}} \vee \bigcirc \varphi_{i_{r_4}})) \\ & \wedge \square(\varphi_{i_{r_3}} \Rightarrow (\bigcirc \varphi_{i_{r_1}} \vee \bigcirc \varphi_{i_{r_3}} \vee \bigcirc \varphi_{i_{r_4}} \vee \bigcirc \varphi_{i_{r_5}})) \\ & \wedge \square(\varphi_{i_{r_4}} \Rightarrow (\bigcirc \varphi_{i_{r_2}} \vee \bigcirc \varphi_{i_{r_3}} \vee \bigcirc \varphi_{i_{r_4}} \vee \bigcirc \varphi_{i_{r_6}})) \\ & \wedge \square(\varphi_{i_{r_5}} \Rightarrow (\bigcirc \varphi_{i_{r_3}} \vee \bigcirc \varphi_{i_{r_5}} \vee \bigcirc \varphi_{i_{r_6}})) \\ & \wedge \square(\varphi_{i_{r_6}} \Rightarrow (\bigcirc \varphi_{i_{r_4}} \vee \bigcirc \varphi_{i_{r_5}} \vee \bigcirc \varphi_{i_{r_6}})) \\ & \text{\# constraints forbidding robot co-location} \\ & \wedge \square \bigvee_{i=1}^{6} (\varphi_{1_{r_i}} \wedge \bigvee_{j \neq i} \varphi_{2_{r_i}})\end{aligned}$$

In addition, the activation of a specific mode enforces progress toward the current goal configuration:

$$\begin{aligned}\varphi_{mode\_trans_{12}} = \quad & \text{\# each robot's motion under mode } p = 12 \\ & \square(\varphi_{1_{r_1}} \wedge \pi_{12} \Rightarrow (\bigcirc \varphi_{1_{r_1}} \vee \bigcirc \varphi_{1_{r_2}})) \\ & \wedge \square(\varphi_{2_{r_6}} \wedge \pi_{12} \Rightarrow (\bigcirc \varphi_{2_{r_6}} \vee \bigcirc \varphi_{2_{r_5}})) \wedge ...\end{aligned}$$

The entire formula $\varphi_{mode\_trans_{12}}$ is omitted for space reasons, but the above excerpt says that if mode $p = 12$ is activated, then robot 1 is being driven towards r_2 and robot 2 is being driven towards r_5, so the environment variables indicating the robot's sensed location must change accordingly. In the construction of multi-robot motion controllers using the approach described in [2], these constraints can be automatically generated from the discrete path used to sequence local controllers. Note that if we wanted to model nondeterministic disturbances that would cause the robots to drift away from their path, we could incorporate these by relaxing these assumptions on the environment.

3) Fairness Assumptions: In addition to the above safety conditions, additional constraints on the environment are required to ensure that if a robot is moving towards a specific region, it will eventually get there (unless the door is closed). This ensures that in the absence of adversarial obstacles, every atomic controller eventually completes, i.e. that the robots' goal configuration is either changed or achieved. In Example 1, we add the fairness conditions

$$\begin{aligned}\varphi_{fair_{12}} = \quad & \square\diamondsuit(\varphi_{1_{r_1}} \wedge \pi_{12} \wedge \neg\pi_{door} \Rightarrow \bigcirc \varphi_{1_{r_2}}) \\ & \wedge \square\diamondsuit(\varphi_{2_{r_6}} \wedge \pi_{12} \wedge \neg\pi_{door} \Rightarrow \bigcirc \varphi_{2_{r_5}}) \wedge \ldots\end{aligned}$$

This corresponds to the assumption that the atomic motion controller corresponding to mode $p = 12$ will eventually drive the robot from r_1 to r_2 as long as the door is not closed. Again, these assumptions can be automatically constructed from the discrete path between regions for each mode, incorporating obstacles that may prevent robot motion.

The final addition to the system specification is the mutual exclusion between the modes in $\mathcal{P}_N$; in Example 1, $\varphi_{mode\_mutex} = \square((\pi_{12} \vee \pi_{21}) \wedge (\neg\pi_{12} \vee \neg\pi_{21}))$

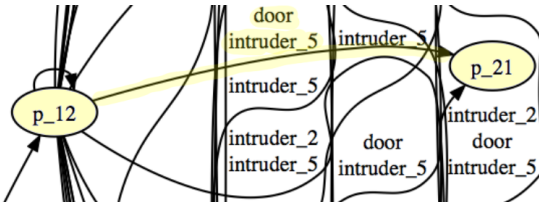


Fig. 3: Excerpt of synthesized automaton for Example 1

Given a task specification $\varphi = (\varphi_e \Rightarrow \varphi_s)$ of the form described in Section III-B, the new GR(1) specification used to synthesize a controller (with added multi-robot motion constraints, sensor assumptions and fairness conditions) is:

$$\varphi_{new} = (\varphi_e \wedge \varphi_{motion} \wedge \bigwedge_{p \in \mathcal{P}_N} \varphi_{mode\_trans_p} \wedge \bigwedge_{p \in \mathcal{P}_N} \varphi_{fair_p}) \\ (\Rightarrow \varphi_s \wedge \varphi_{mode\_mutex})$$

C. Synthesis

An LTL formula φ is *realizable* if there exists a finite state strategy that, for every finite sequence of truth assignments to the environment-controlled propositions, provides an assignment to the system-controlled propositions such that every infinite sequence of truth assignments generated in this manner satisfies φ. The synthesis problem is to find a deterministic finite state automaton (if one exists) that encodes this strategy, i.e. whose executions satisfy φ. When restricted to LTL formulas of type GR(1), such as those described in Section III-B, the algorithm introduced in [9] permits synthesis in time polynomial in the size of the state space; this synthesis algorithm was extended in [11] to accommodate a wider class of specifications with the same polynomial time complexity.

Fig. 3 depicts a single transition of the automaton synthesized for the above specification using the GR(1) synthesis algorithm in [11] using the SLUGS synthesis tool[3]; the full automaton has 310 states and took 0.93s to synthesize on a 1.3 GHz Intel Core i5 processor with 8GB of RAM. Each state of the automaton is labeled with the truth assignment to the system-controlled propositions in that state, and each transition is labeled with the truth assignment to environment-controlled propositions required for that transition to be enabled. Incoming transitions therefore determine the truth value of the environment propositions for each state. In the state labeled with p_{12}, robot 1 is heading towards goal r_2 and robot 2 is heading towards goal r_5. In the depicted transition, we see the system switch modes from p_{12} to p_{21} when the door is closed and an intruder is sensed in r_5 (as indicated by incoming edge labels `door` and `intruder_5`).

D. Continuous Execution

If synthesis succeeds, a controller that implements the corresponding continuous behavior is constructed by viewing the resulting automaton as a hybrid controller, with a transition between two states achieved by the activation of one or more low-level continuous controllers corresponding to each proposition. The atomic controllers used must satisfy the bisimulation property [1], which ensures that every change in the discrete system model can be implemented in the continuous domain. As described in Section III-B, the atomic controllers for each mode are obtained using a construction such as the one in [2], and the specification is augmented with environment safety and liveness assumptions derived from a discrete transition graph representing the behavior of these atomic controllers. Therefore, every change in the discrete model can by design be implemented in the continuous domain (i.e., the atomic motion controllers are guaranteed to drive each robot towards the current goal configuration regardless of initial state).

Given an automaton synthesized using the above approach, the multi-robot motion controller corresponding to the current mode is invoked to initiate the robot team's motion towards the current goal configuration. Note that a switch to any mode is possible at any time. Transitions in the automaton are instantaneous, as they correspond to activation of a new mode or a discrete event in the environment, including the arrival of a robot in a new region. Moreover, every continuous state maps to a unique discrete state, and so the mode to be activated in response to an environment event is always determined for a given strategy. As the automaton is correct by construction, this continuous execution ensures that the LTL specification is met on the resulting executions.

E. Computational Complexity

We use a computationally expensive centralized approach in order to guarantee fulfillment of the specified task in a reactive environment, but still save computation over previous approaches. For example, the desired behavior could be obtained by using the approach in [10], where the control input is not just a switching mode, but the activation of a controller that drives the team from some initial configuration to some goal configuration. However, that approach adds one variable to $\mathcal{Y}$ for the activation of a motion controller to each region for each robot, rather than just one for each permutation of robot goals – this is a total of $N * |\mathcal{R}|$ variables in contrast with one for each of $N!$ modes. The strategy synthesis scales exponentially in the number of sensor propositions[4], i.e. by a factor of $2^{|\mathcal{Y}|}$. If the number of goals N is small compared to the number of regions $|\mathcal{G}|$ (as is usually the case), the approach presented in this paper yields a significant reduction in computational complexity. The example in the next section will demonstrate this difference.

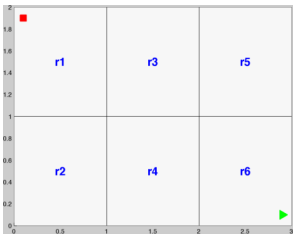
IV. Example

Fig. 4 summarizes a MATLAB simulation of the automaton synthesized for Example 1; a video accompanies this paper. As the purpose is to illustrate the high-level switching control, the naive controllers used to simulate motion for each robot set the robot's velocity towards the centroid of the next region on the shortest discrete path to the goal. In future work, these will be replaced in physical experiments with atomic controllers such as those constructed in [2], [3].

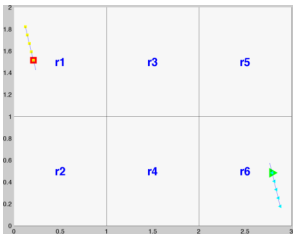
[3]available at https://github.com/ltlmop/slugs

[4]this can be reduced in practice using bit vectors to encode mutually exclusive propositions

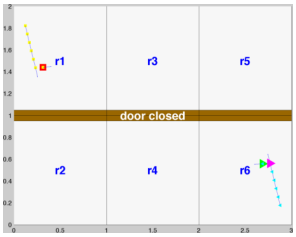
Robots 1 and 2 are represented by a red square and green triangle respectively. Larger markers on the trajectories indicate points where the robots change state, either by moving to a new region, or as the result of a sensor event. Fig 4(a) shows the initial condition, with robot 1 in r_1 and robot 2 in r_6. The door is initially open. In Fig 4(b), the robots have both started moving towards their goals, r_2 and r_5, respectively. In Fig 4(c), the door has closed before the robots achieve their goals. In Fig. 4(d), an intruder is sensed in r_2. Since robot 1 is now on the wrong side of the door, the system switches modes, assigning robot 2 the goal of r_2 instead. With these new goals, robot 2 starts moving towards r_2 through r_4, while robot 1 moves to r_5 through r_3, as in Fig. 4(e). In Fig 4(f), the robots have arrived in their new goal regions. The ability of the two robots to switch goals is critical to meeting this specification. If the door were reopened, each robot would be assigned the closest goal, making the synthesized protocol robust to an adversarial environment that repeatedly opens and closes the door.



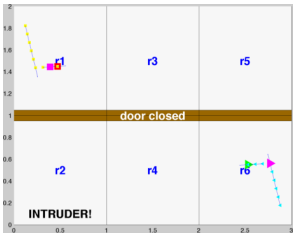
(a) Initial state – robot 1 is in r_1, robot 2 is in r_6



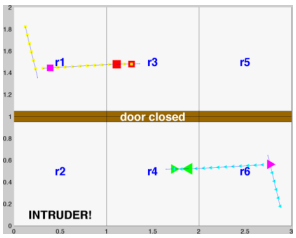
(b) Robots 1 and 2 move towards their respective goals, r_2 and r_5



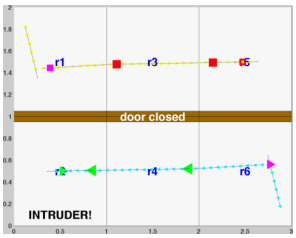
(c) The door closes before the robots can attain their respective goals. The robots stop moving towards the door.



(d) An intruder appears in r_2. The robots switch goals and start moving towards their new goals



(e) Robot 1 moves towards r_5, Robot 2 moves towards r_2.



(f) Robot 1 arrives in r_5, Robot 2 arrives in r_2.

Fig. 4: Simulation of controller synthesized for Example 1

Our solution uses $|\mathcal{Y}| = 2$, while the approach in [10] uses $|\mathcal{Y}| = 2 * |\mathcal{R}| = 2 * 6 = 12$. This represents a reduction in the state space for synthesis by a factor of 2^{10}. Moreover, our approach still allows robots to move simultaneously, succeeding in the cases described in [10] where requiring that at most one robot move at a time is too conservative.

V. Conclusion

We have demonstrated a reactive synthesis approach to controlling a team of homogeneous robots in a centralized fashion. Our approach allows dynamic goal reassignment via switching modes, and guarantees that the high-level task specification is met despite an adversarial environment. For this work, we assumed that all robots could communicate. Future work will consider the case where only robots that are within a specified, known communication range can communicate, thereby allowing centralized control only within these ranges. This will reduce computation for large teams of robots in large environments, where the communication range is much smaller than the size of the workspace.

In modeling a multi-robot system as a switched system, we have made explicit the separation between the physical controllers and the robots themselves. Each mode of the system maps a set of abstract or "virtual" robots to physical robots, providing the associated controllers. This can be likened to the mapping of virtual machines to physical ones for computation, which allows migration of resources as needed to address new tasks. Here, the abstraction of a switched system of virtual robots allows migration of physical controllers in order to achieve a high-level behavior. Virtual machines have revolutionized the way data centers and large computer clusters are managed, and greatly improved their efficiency and ability to react to changes in the operating environment. We contend that similar advantages apply for systems of interchangeable robots, and intend to explore our approach on larger systems in the future.

References

[1] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[2] Nora Ayanian, Vinutha Kallem, and Vijay Kumar. Synthesis of feedback controllers for multiple aerial robots with geometric constraints. In *IROS*, pages 3126–3131, 2011.

[3] Nora Ayanian, Daniela Rus, and Vijay Kumar. Decentralized multirobot control in partially known environments with dynamic task reassignment. In *NecSys*, pages 311–316, Santa Barbara, CA, 2012.

[4] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[5] M Bernardine Dias, Robert Michael Zlot, Nidhi Kalra, and Anthony (Tony) Stentz. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Pittsburgh, PA, April 2005.

[6] M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *Robotics, IEEE Transactions on*, 26(1):48–61, 2010.

[7] Marius Kloetzer and Calin Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, 23(2):320–330, 2007.

[8] Jun Liu, Necmiye Ozay, Ufuk Topcu, and Richard M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Trans. Automat. Contr.*, 58(7):1771–1785, 2013.

[9] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, pages 364–380. Springer, 2006.

[10] Vasumathi Raman and Hadas Kress-Gazit. Synthesis for multi-robot controllers with interleaved motion. In *ICRA*, pages 4316–4321, 2014.

[11] Vasumathi Raman, Nir Piterman, and Hadas Kress-Gazit. Provably correct continuous control for high-level robot behaviors with actions of arbitrary execution durations. In *ICRA*, pages 4075–4081, 2013.

[12] Matthew Turpin, Nathan Michael, and Vijay Kumar. Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In *ICRA*, pages 842–848, 2013.