Correct High-level Robot Behavior with Unexpected Events in a Shared Workspace (Extended Abstract)

Kai Weng Wong and Hadas Kress-Gazit

Abstract—Provably-correct robot controllers, which can be automatically synthesized from high-level task specifications, have the advantage of providing correctness guarantees during execution. These controllers, however, are prone to failures during execution, if the assumptions made about the environment in the specifications are violated. We consider this problem of unexpected environment behaviors during execution and develop mechanisms that allow the robot to continue its mission when the correctness guarantees can be restored and maintained.

I. INTRODUCTION

Synthesis of provably-correct robot controllers from high-level task specifications has gained an increasing interest in the robotics community (e.g. [2], [5], [9]). These controllers, if successfully generated, can provide guarantees on robot behaviors under different environments. When the user writes a high-level task specification using linear temporal logic (LTL), he or she makes assumptions about the environment in order to create a controller. This assumption, however, may not hold during execution due to modeling errors or unforeseen anomalies. In this case, the controller cannot guarantee the correctness of the robot’s behavior.

We propose a two-step solution for the problem of failure due to environment assumption violations: (a) We modify the controller synthesis algorithm in [3] to allow transient anomalies in the synthesized controller. During controller execution, our algorithm automatically records any new environment behavior into the specification and resynthesizes the controller. (b) If the environment includes another robot with another task working in the same workspace, we can automatically modify the environment robot behavior through communication and achieve the goals of the two robots.

II. PROBLEM DEFINITION

In our work, we consider controllers, in the form of finite state machines, synthesized from high-level task specifications written in LTL formulas. The specifications include assumptions about the operating environment of the robot. For example, when two robots share the same workspace, each robot can be treated as the environment for the other one. The user may then assume one robot in the shared workspace not in the way of the other robot, and this assumption may not hold during execution. Fig. 1 shows two robots both heading to the square region. The orange robot can prevent the blue robot from reaching square, if it is assumed that the two robots cannot be in the same region at all times.

We are interested in the problem arose during execution when the environment assumptions are violated. We develop mechanisms to construct controllers that allow the robot to continue its mission while at the same time provide correctness guarantees if possible.

III. RESULTS

First, we tackle the problem without involving any communication with the surrounding robot. This includes an offline step, recovery, and an online step, environment characterization. The two steps were first introduced in RSS 2014 [7].

A. Recovery

In the recovery step, we modify the controller synthesis algorithm proposed by [3] and include into the controller all the transitions between states that are non-failure-causing. A transition is non-failure-causing if after the robot has taken this transition to a new state, the robot can reach its goals safely starting at this next state in the controller.

In this step, we assume the environment assumption violations are transient and the violations will be removed eventually.

B. Environment Characterization

In the environment characterization step, during execution, if the environment acts differently than the assumptions made by the user, our algorithm will automatically append the newly observed environment behavior into the assumption portion of the specification and create a new controller. The new controller generated takes this new environment behavior into account and allows the robot to continue its task accordingly.

If the step cannot resolve the environment assumption violation, i.e., no controller is successfully synthesized for the robot, and if the environment involves another robot working in the same workspace, the robot can then initiate a conversation with the other robot and try to resolve the problem together. This is the Integrative Negotiation step, introduced in ICRA 2015 [8].

Fig. 1: The orange robot prevents the blue robot from entering the square region.
C. Integrative Negotiation

The integrative negotiation step involves two robots working in a shared workspace. Consider one of the two robots, here referred to as the system robot, cannot synthesize a new controller with the current environment behavior. Since the other robot in the shared workspace is treated as environment of the system robot, the system robot will initiate a communication with the other robot to exchange parts of its specification. This modifies the behaviors of the other robot to accommodate the tasks of the system robot.

The system robot will send its assumptions about the environment behaviors, including the expected behaviors of the other robot, and its goals to the other robot. The other robot is asked to incorporate these information into its specification and to synthesize a new controller.

If a new controller is successfully synthesized, i.e., the other robot can ensure that the system robot will finish its task, then the execution of the two robots resumes. Otherwise, the other robot will ask the system robot to incorporate its assumptions instead, and send the system robot parts of its specification.

This step allows the two robots to finish their tasks without a priori information about the other robot. With this step, the orange robot moves to the tunnel region so that the blue robot can enter the square region, as shown in Fig. 2.

Fig. 2: The orange robot leaves and the blue robot can now enter the square region

IV. Future Work

We will relax assumptions such as mutual exclusion of robots in the same region or the environment information being globally available in our work. We also want to extend the work here to involve more agents.

V. Related Work

Li et al. [6] consider the unrealizable specification problem. Based on an unrealizable specification, their work creates a realizable specification, from which a controller can be synthesized. They tackle the problem through mining environment assumptions, in the form of LTL formulas, that renders the specification realizable, from the analysis of counter-strategies obtained from the synthesis procedure. Our proposed algorithm modifies the specification based on the observed environment behavior instead.

In [4], the authors consider decentralized control of robots with each given an independent specification. During execution, each agent verifies mutual satisfiability upon meeting with the other robots, through an exchange of their automata. In our work, we also consider two agents having independent specifications and we automatically resolve any conflicts arose during execution through rewriting the specification of the robots.

REFERENCES