Collaborative Multi-Robot Simulation using LTL Synthesis

Sunghae Kim, Gihwon Kwon

Dept of Computer Science, Kyonggi University, Korea
{tprover, khkwon}@kyonggi.ac.kr

Abstract. Synthesis is to automatically construct a system satisfying the given requirements specification. Therefore, it is not necessary to additional testing and verification in the system, as long as requirements specification should be written in a special class of temporal logic formulas. Consequently, it is essential to check precise translation and missing requirements. If you apply synthesis to robot task planning, it's very natural to extend multi robot task planning from single robot task planning. However, the LTLMoP simulator only supported single robot task planning, it was difficult to simulate multi robot task planning. In order to overcome such difficulty, we extent the LTLMoP simulator to simulate multi-robot task planning. Also, we verify and make sure that the simulator works correctly using multi-robot collaboration scenario.

Keywords: LTL, synthesis, multi-robot, simulator, LTLMoP

1 Introduction

LTL Synthesis [1] is a process to automatically construct a system satisfying the given requirements written in Linear Temporal Logic. From the underlying theory, the main benefit of synthesis is that the resulting system does not require additional testing and verification. However, the disadvantages of synthesis are, limitation of application area in reactive system, requirements specification written in LTL formulas, and requiring a lot of experience and repetitive tasks.

In this paper, we used LTLMoP simulator which supports robot task planning and its simulation using LTL synthesis. The LTLMoP takes as input robot task requirements of the form of Structured English, and converts it to LTL formulas internally. After that, it automatically generates robot task planning by using synthesis algorithm and simulates the continuous trajectories of robot.

Even though the robot task planning is automatically generated, it is essential step to check the trajectories of the robot using simulator. Because the LTLMoP only supported single robot task planning, it was difficult to simulate a collaborative multi-robot task planning. In this paper, we present a multi-robot task planning simulator based on LTLMoP to overcome this problem.

The remainder of this paper is organized as follow: Section 2 provides the reader with a necessary background in Linear Temporal Logic and synthesis to specify robot

task requirements. Section 3 describes the architecture of LTLMoP and how we extend it to simulate a multi-robot task planning In Section 4, we show that four robots collaborative task planning simulation as case study. Finally, we conclude the paper with future works.

2 Preliminaries

2.1 Linear Temporal Logic

Linear Temporal Logic (LTL) is a kind of modal logic. The syntax of LTL is given by the following abstract syntax equation:

$$0 ::= P \ ^{-1}0\ 01\ ^{V}\ 02\ I\ 00\ I\ 00\ I\ \square\ I\ 01\ U\ 02$$

It consists of the standard propositional logic with some temporal operators. The semantics of the temporal operators are as usual: next(0), future(0), always(E)•

In the above equation, let *AP* is a set of atomic propositions. LTL formula *0* is constructed from atomic propositions $p \in AP$ and temporal operators. Given negation() and disjunction(V), we can define conjunction(A), implication(and equivalence(<=>). For example, the conjunction 0_1 A 002 is equal to ¯'(-0i V ¯OA

The semantics of LTL formula *0* is defined on an infinite sequence a of truth assignments to the atomic proposition $p \in AP$. For a formal definition of the semantics we refer the reader to [2].

2.2 LTL Synthesis

Synthesis is to automatically construct a functional correct system from a behavioral description of the system. Because the resulting system is automatically constructed from the specification, so called "correct-by-construction", it is not necessary to additional testing and verification.

In this paper, we consider a special form of temporal logic formulas. These special formulas are LTL formulas of the form $_{g}s_{e}$ is the assumption about the environment, and represents the desired behavior of the system. More precisely, both O_e and 0_S have the following structure:

$$4=0,eAKAO;\ ,\ Os = A'\ A\ 0;$$

where 0_e represents the assumption of the environment, and it consists of the conjunction of formulas 0: , 0r , $0;$ and the meaning of these formulas are as follow:

- O^e : Non-temporal boolean formulas *(A*) constraining the initial value(s) for the environment.

- O_r : represents the possible state of the environment. It consists of the conjunction of formulas of the form DB, .
- O_g^e : represents the goal assumptions for the environment. It consists of the conjunction of formulas of the form DOB, .

The formulas $_{cs}$ represents the desired behavior of the system, and it consists of the conjunction of formulas , Or , 0; and the meaning of these formulas is similar to the meaning of environment, so we can omit it.

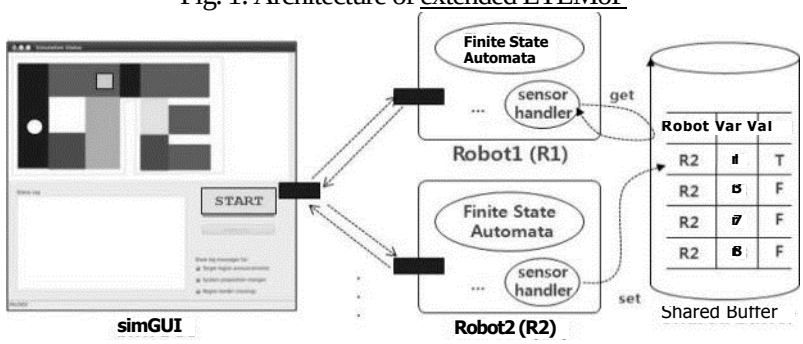
The formula $O_g e O_s$ which consists of the goal of the environment and the goal of system, is a *General Reactivity(1)(GR(1))* formula[1].

3 Extension of LTLMoP Simulator

If we apply LTL synthesis technique to robot task planning, it's very natural to extend multi-robot task planning from single-robot task planning. In multi-robot scenarios, one robot becomes part of the environment of another robot. Therefore, we need to just describe additional sensor propositions as another robot's behavior. In robot task planning, the LTLMoP is a Python-based toolbox for single robot simulation using LTL synthesis

However, the LTLMoP only supported to simulate single-robot task planning, it was difficult to simulate multi-robot task planning. To solve this problem, we extend the LTLMoP to support multi-robot task planning simulation. The following figure shows architecture of multi-robot task planning simulator.

Fig. 1. Architecture of extended LTLMoP



To simulate multi-robot task planning, it is necessary to execute FSA modules as the number of robots. Because each FSA module needs a unique udp port, we modified to use an individual port for each FSA module, and extend protocol to transfer the name of robot. Also, in multi-robot collaboration task planning, each robot need to recognize the location of another robot, we add a shared buffer which stores information about the robot location. Finally, we extend a simGUI module to store a position and velocity information of multi-robot.

4 Case Study

4.1 Suspect and Detective Scenario

The suspect and detective scenario is as follow: *"The suspect robot is moving in a clockwise direction in the workspace. If the suspect robot stop in some region, the detective robot moves as near as possible to it, then stay there. When the suspect robot begins to move in forth direction, the detective robot follows it. While the detective robot follows it, the detective robot should not stay in the same region with it, and should not pass it."*

Fig. 2. The workspace and initial regions

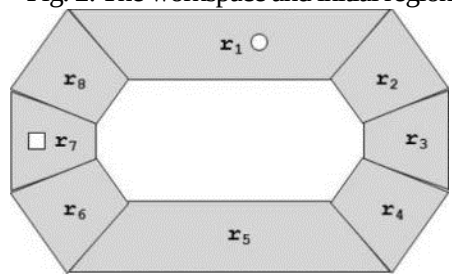


Fig. 2 shows that the workspace consists of eight regions labelled r_1, , r_8, and the region r_1 and r_7 are the initial region of robots. Also the circle represents the suspect robot, and the rectangle represents the detective robot.

4.2 Modeling the multi-robot collaboration

The desired behavior of the suspect robot is moving in workspace, sometimes stay in some region. First we consider one environment proposition, *[stop)*, which becomes true if the robot stop moving. Second, we define workspace shown in Fig. 2 as eight system propositions tr_1, r_2, , r_8), one for each region. Following these propositions, we can construct requirements of environment of the suspect robot 4:

$$Rs = —stop$$

The initial condition represents the suspect robot is moving.

$$R_S^s = \begin{cases} {}^{r}i^{A}A.\{2,...,8\}'^{1}; \\ Transitions \text{ A } Mutual\ Exclusion \\ Ostop = (\text{ Aio,...,8}) O^{r,} <=>^{0)} \\ \text{A}^{111} <> (ri \text{ v } stop) \end{cases}$$

The initial condition of system is that the robot 1 starts in r_5. The second line represents the possible *transitions* between regions, for example, from r_3 the robot can move to adjacent r_2, r_4 or stay in r_3. The *mutual exclusion* constraints that the robot at any step, stay in exactly one of regions. The third line sub formula is part of the desired specification and represents that if the suspect robot stops moving, it should remain in that region in the next step as well. The last line describes the desired be-

havior, the suspect robot has to infinitely often visit r_1 {1,...,8}, unless it has detected

stop signal. The *transitions* and *mutual exclusion* formulas about workspace are automatically generated in LTLMoP.

The desired behavior of the detective robot is to follow the suspect robot without overlap, so it depends on the location of suspect robot. For environment of detective robot, we define eight propositions: {r1, r2 , ,*r1}* indicating the location of suspect robot. The modeling of the environment of the detective robot is as follow:

$$R;_{)} = \left| \begin{array}{l} \left| \begin{array}{l} \text{V (Or}_{i} \\ 2 \\ 2 \\ \\ s \\ .\text{Jr}_{8} \end{array} \right| \end{array} \right| \quad \begin{array}{l} {}^{s_A}\bullet 1 \bullet \text{`iE}\{2,...,8\}^{s} \\ {}^{Sr_i s} \\ \\ \\ \\ \end{array} \quad \left| \begin{array}{l} ({}^{01}1\,A \qquad -10r_i^s) \\ {}^{s}\text{A} \qquad . \\ \text{e}\{1,...,8\},:* \end{array} \right.$$

The initial condition of environment represents the suspect robot starts in r_1 and constraints that the suspect robot at any step, stays in exactly one of regions.

$$R;_{)} \left| \begin{array}{l} r7^{\text{A}}{}_{\text{Aie}\{1,...,8\},\text{W7}}{}^{r}i \\ \textit{Transitions A Mutual Exclusion} \\ \textit{AiE}\{o,...,7\}(r(i \bmod 8)+1^{\text{A}}\, r(^{S}i+2)mod\ 8) \qquad (^{\text{CI}} \bmod 8)+1^{4\copyright,}\, r_{(i} \bmod \\ 8)+1) \end{array} \right.$$

$\text{Aie}\{1,...,8\}$ $\qquad\qquad$ ri

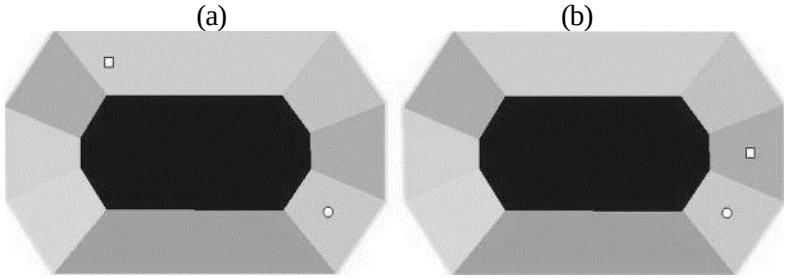
The initial condition of system is that the detective robot starts in r_7. The third and fourth lines represent the desired behavior; it stays in r_i if there is a suspect robot in r_{i+1} and infinitely often visit r_{iE}

4.3 Simulation of collaborative multi-robot

To verify the multi-robot simulator, we simulate suspect and detective scenario. In order to simulate multi-robot task planning, we have to execute individual FSA module for each robot.

Fig. 3. Detective follows suspect

(a) (b)



In Fig. 3, extended LTLMoP simulator shows that the suspect robot moves forth and stays in r_4, then the detective robot follows suspect as near as possible and stays r_3.

5 Conclusion

The main benefit of synthesis is to automatically construct a system satisfying the given requirements, however synthesis process requires a lot of experience and repetitive tasks, because the requirements should be written in logical formulas. Although the resulting system does not require additional testing and verification, it is essential to check precise translation of logical formulas and missing requirements using simulation tool. In this paper, we have presented the multi-robot task planning simulator based on LTLMoP, also we have shown that the simulator works correctly using multi-robot collaboration scenario. In the future work, we plan to implement a more generic version of multi simulator.

References

1. Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. "Synthesis of Reactive(1) Designs", In Proc, 7th International Conference on Verification, Model Checking and Abstract Interpretation, 2006
2. E. Allen Emerson. Tmporal and modal logic. In Handbook of theoretical computer science (vol. B): formal models and semantics, pp.995-1072. MIT Press, Cambridge, MA, USA, 1990.
3. LTLMoP, http://ltlmop.github.com/
4. Hadas Kress-Gazit, Georgios E. Fainekos and George J.Pappas. "Translating Structured English to Robot Controllers", 2008, Advanced Robotics Special Issue on Selected Papers from IROS 2007.
5. Hadas Kress-Gazit, Georgios E. Fainekos, George J. Pappas. "Where's Waldo? Sensor-Based Temporal Logic Motion Planning", ICRA 2007
6. A.Pnueli and R. Rosner. On the synthesis of a reactive module. In POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM Press, 1989.