Korea-Japan Joint Workshop on ICT (Pohang, Korea, 20-22 September 2012), Paper #02

Mitigating the state explosion problem of synthesized multi-robot controller by decentralized model*

Sunghae Kim Gihwon Kwon
{tprover, khkwon}@kyonggi.ac.kr

Abstract: There are many interests to synthesize controller according to the advancement of IT convergence. In robotics domain, synthesis has advantages to automatically generate high-level task planning. Normally, it is possible to apply synthesis technique to single robot controller. However in collaboration of multi-robot, the full state space of multi-robot is to be Cartesian product of each robot's state space. To mitigate this problem, we tried to apply divide-and-conquer technique to multi-robot task planning. Also, through multi-robot simulator developed by under this study, we verified the desired behavior of multi-robot.

Keywords: state explosion problem, LTL, synthesis, multi-robot, simulator

1. Introduction

As controllers are present in a large proportion in the IT convergence environment, there are many interests in synthesizing controllers to operate correctly. For example, it's essential to synthesis robot controllers for controlling robot motion to satisfy requirements in robotics domain.

Linear Temporal Logic (LTL) synthesis is one of technique to generate controller[2]. In this technique, system requirements should be described using LTL formulas, and then check whether LTL formulas are realizable or not. As a synthesis result, controller in the form of automata is generated, if LTL formulas are realizable. Researches on synthesis techniques to generate robot controllers are attracting interest according to the advancement of H/W such as hybrid-controller[3].

Normally, it is possible to apply synthesis technique to single robot controller. However in collaboration of multi-robot, we encountered state explosion problem. Because the full state space of the multi-robot controller is to be Cartesian product of each robot's state space. Even though LTL formulas are realizable, if the state explosion problem occurred, then it takes a lot of time to do synthesis, so that it is necessary to minimize or mitigate this problem. There are several approaches to reduce the state explosion problem, such as abstraction, symmetry, divide-and-conquer and so on [4].

In this study, we tried to use divide-and-conquer to reduce state explosion problem. In other words, we specified each robot's requirements specification separately, and added shared minimum information which enables each robot to collaboration. In shortly, to reduce the state space, we used decentralized collaboration model rather than centralized collaboration model. Also we applied this model to Poli-Heli scenario.

The remainder of this paper is organized as follows: Section 2 provides the reader with a necessary background. Section 3 describes process of synthesizing controllers with Poli-Heli scenario. In section 4, we show that the architecture of multi-robot simulator and how we simulate a collaborative multi-robot task planning. Finally, we conclude the paper with future works.

* This works is supported by Gyonggi Regonal Research Center, Gyonggi-Do [2012, Application Software Development Technique based on Mobile Platform]

2. Preliminaries

In this section, we describe background of this study. For more detailed information, see [5]. First, LTL provides operators used in propositional logic such as $\neg$, $\vee$, as well as some temporal operators such as $\bigcirc$, $\Diamond$, $\Box$.

DEFINITION (Syntax of LTL)

Let *AP* is a set of atomic propositions. The syntax of LTL formula is given by the following BNF rule.

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi \mid \Diamond\phi \mid \Box\phi$$

In the above rule, *true* is a constant that represents always true, *p* is any propositional variable belongs to *AP* and $\neg$, $\vee$ are propositional operators which represent negation and disjunction respectively. And also, $\bigcirc$, $\Diamond$, $\Box$ are temporal operators which represent next, future, always respectively. In this paper, we introduced only a subset of operators, but we can define other operators using these operators, for example $\phi_1 \Rightarrow \phi_2$ is equal to $\neg\phi_1 \vee \phi_2$.

DEFINITION (Semantics of LTL)

The semantics of LTL formula ϕ is defined over an infinite sequence σ of truth assignments to the atomic proposition $p \in$ AP. For the convenience, we first define some notations for infinite sequence. For infinite sequence σ, we denote set of truth propositions of i-th in σ as $\sigma(i)$. Also we denote that ϕ is satisfied on i-th of infinite sequence σ as $\sigma, i \models \phi$. Using these notations, we can define semantics of each operator as follows:

$$
\begin{array}{lll}
\sigma, i \models true & & \\
\sigma, i \models p & \textit{iff} & p \in \sigma(i) \\
\sigma, i \models \phi_1 \vee \phi_2 & \textit{iff} & \sigma, i \models \phi_1 \text{ or } \sigma, i \models \phi_2 \\
\sigma, i \models \bigcirc\phi & \textit{iff} & \sigma, i+1 \models \phi \\
\sigma, i \models \Diamond\phi & \textit{iff} & \exists_{j \geq i} \bullet \sigma, j \models \phi \\
\sigma, i \models \Box\phi & \textit{iff} & \forall_{j \geq i} \bullet \sigma, j \models \phi
\end{array}
$$

As following above definition, LTL formula $\bigcirc\phi$ means that ϕ is true in next state, also $\Diamond\phi$ means that ϕ is true in some future states, finally $\Box\phi$ means that ϕ is true in all future states.

Using these syntax and semantics, system requirements can be defined as LTL formulas. LTL synthesis is a technique that takes as input LTL formulas, and then generates controller which meets LTL formulas [5]. It is necessary to limit input formulas for synthesis to work well. In other words, if we use full LTL formula, we can get a expressive power but synthesis algorithm is to be more complex, so that we should describe requirements using LTL fragment which has normal expressive power but synthesis algorithm is to be less complex.

Normally, the system interacts with environment continuously. Therefore, when we write requirements, we should consider not only system but also environment, so that requirements should be described in following form:

DEFINITION (LTL formula representing requirements)

The requirements should be described using LTL formulas of the form $\phi_e \Rightarrow \phi_s$. ϕ_e is an assumption about the environment, and ϕ_s represents the desired behavior of the system. More precisely, both ϕ_e and ϕ_s have the following structure:

$$\phi_e = \phi_i^e \wedge \phi_t^e \wedge \phi_g^e \ , \ \phi_s = \phi_i^s \wedge \phi_t^s \wedge \phi_g^s$$

where ϕ_e represents an assumption of the environment, and it consists of the conjunction of formulas ϕ_i^e, ϕ_t^e, ϕ_g^e and the meaning of these formulas is as follows:

ϕ_i^e : non-temporal boolean formulas (B_i) constraining the initial value(s) for the environment.

ϕ_t^e : represents the possible states of the environment. It consists of the conjunction of formulas of the form $\Box B_i$.

ϕ_g^e: represents the goal assumptions for the environment. It consists of the conjunction of formulas of the form $\Box\Diamond B_i$.

The formulas ϕ_s represents the desired behavior of the system, and it consists of the conjunction of formulas ϕ_i^s, ϕ_t^s, ϕ_g^s and the meaning of these formulas is similar to the meaning of environment, so we can omit it.

The formula $\phi_g^e \Rightarrow \phi_g^s$ which consists of the goal of the environment and the goal of system, is a Generalized Reactivity(1) formulas (GR(1)) [1].

As shown before, requirements are described in the assume-guarantee form $\phi_e \Rightarrow \phi_s$. In other words, the system guarantees desired behaviors only when the environment acts as expected. As the system doesn't need to consider all circumstances, this restriction of the environment simplifies synthesis algorithm.

3. Poli-Heli Robots

3.1 Centralized Collaboration Model

As a case study of collaborative multi-robot, we show that Poli-Heli scenario. The requirements of Poli-Heli are as follows: *"Poli and Heli are patrol robots. Poli moves clockwise from r_1 in r_1, r_2, r_3 and r_4, when Poli detects injured people then stay there. Heli moves clockwise from r_5 in r_5, r_6, r_7 and r_4, and does as Poli. Because only one robot can be in r_4 at a time, if one robot enters in r_4 already, then another robot needs to wait until the robot has passed in r_4".*

The following figure shows that workspace seven connected regions. In figure 1, circle denotes Poli robot and rectangle denotes Heli robot. The r_4 is a critical section where one robot can pass only.

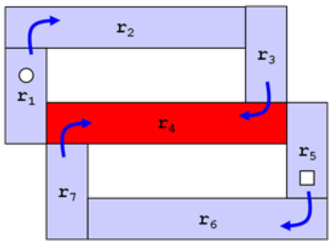

Fig. 1. The workspace and initial regions

We describe the process of specifying above requirements into LTL formulas. At first, for generating centralized controller, we specified requirements as one LTL specification.

First, Poli should know the information $\{help^P, r_4^H\}$ from environment. Variable $help^P$ indicates that Poli detects an injured people in patrol, and r_4^H indicates that Heli is passing in r_4. Also, Heli should know the information $\{help^H, r_4^P\}$ similar to Poli. Using these variables, we can specify centralized model of environment's requirements as follows:

$$\neg help^P \wedge \neg r_4^H \text{ ----------------- } (1)$$

$$\neg(r_1^P \vee r_2^P \vee r_3^P \vee r_4^P) \Rightarrow \neg\bigcirc help^P \text{ --- } (2)$$

$$\Box\Diamond\neg r_4^H \text{ ---------------------- } (3)$$

$$\neg help^H \wedge \neg r_4^P \text{ ----------------- } (4)$$

$$\neg(r_5^H \vee r_6^H \vee r_7^H \vee r_4^H) \Rightarrow \neg\bigcirc help^H \text{ -- } (5)$$

$$\Box\Diamond\neg r_4^P \text{ ---------------------- } (6)$$

The formula (1) represents that Poli doesn't detect injured people and Heli is not in r_4. The formula (2) is a transition relation condition of environment, and represents that Poli doesn't detect injured people in other regions excepts r_1, r_2, r_3 and r_4, and the formula (3) represents that if Heli is passing in r_4, then Heli will have passed in r_4 eventually.

The formulas (4), (5), and (6) are same formulas with Poli's except that robot is changed into Heli. The centralized model of environment consists of the conjunction of formulas from (1) to (6) without same variables.

The centralized model of system's requirements is as follows:

$$r_1^P \wedge \neg r_2^P \wedge \neg r_3^P \wedge \neg r_4^P \text{ ------------- } (7)$$

$$\bigwedge_{i=1}^{4} \Box(r_i^P \Rightarrow r_{(i+1)\mathrm{mod}4}^P) \text{ ------------- } (8)$$

$$\Box\begin{pmatrix} (r_1^P \wedge \bigwedge_{i\neq 1} \neg\bigcirc r_i^P) \\ \vee (r_2^P \wedge \bigwedge_{i\neq 2} \neg\bigcirc r_i^P) \\ \vee (r_3^P \wedge \bigwedge_{i\neq 3} \neg\bigcirc r_i^P) \\ \vee (r_4^P \wedge \bigwedge_{i\neq 4} \neg\bigcirc r_i^P) \end{pmatrix} \text{ ------------- } (9)$$

$$(r_3^P \wedge \bigcirc r_4^H) \Rightarrow (\bigcirc r_3^P \Leftrightarrow r_3^P) \text{ ------- } (10)$$

$$\bigwedge_{i\in\{1,2,3,4\}} \bigcirc help^P \Rightarrow (\bigcirc r_i^P \Leftrightarrow r_i^P) \text{ ---- } (11)$$

$$\bigwedge_{i\in\{1,2,3,4\}} \Box\Diamond(r_i^P \vee help^P) \text{ ---------- } (12)$$

$$r_5^H \wedge \neg r_6^H \wedge \neg r_7^H \wedge \neg r_4^H \text{ ------------ } (13)$$

$$\bigwedge_{i=1}^{4} \Box(r_{i+3}^{H} \Rightarrow r_{(i \bmod 4)+4}^{H}) \quad \text{-----------} \tag{14}$$

$$\Box \begin{pmatrix} (r_4^H \wedge \bigwedge_{i\neq 4} \neg\bigcirc r_i^H) \\ \vee (r_5^H \wedge \bigwedge_{i\neq 5} \neg\bigcirc r_i^H) \\ \vee (r_6^H \wedge \bigwedge_{i\neq 6} \neg\bigcirc r_i^H) \\ \vee (r_7^H \wedge \bigwedge_{i\neq 7} \neg\bigcirc r_i^H) \end{pmatrix} \quad \text{-----------} \tag{15}$$

$$(r_7^H \wedge \bigcirc r_4^P) \Rightarrow (\bigcirc r_7^H \Leftrightarrow r_7^H) \quad \text{------} \tag{16}$$

$$\bigwedge_{i\in\{4,5,6,7\}} \bigcirc help^H \Rightarrow (\bigcirc r_i^H \Leftrightarrow r_i^H) \quad \text{---} \tag{17}$$

$$\bigwedge_{i\in\{4,5,6,7\}} \Box\Diamond(r_i^H \vee help^H) \quad \text{--------} \tag{18}$$

The formula (7) is initial condition of system, and represents Poli starts from r_1. The formulas (8), (9), (10) and (11) are transition relation of system. The formula (8) represents robot's moving relationship between regions, and the formula (9) represents that Poli should be in one region only at a certain time. The formula (10) represents that if Poli is in r_3 and Heli is in r_4, then Poli should stay in r_3, also the formula (11) represents that if Poli detects an injured people in some regions, then stay there. Finally, the formula (12) is the goal of system, represents that Poli should infinitely often patrol r_1 from r_4 unless there is a need for help.

The formulas from (13) to (18) represent initial condition, transition relation and goal, and have the same meaning with Poli's specification. As explained before, the conjunction of formulas from (7) to (18) is centralized model of system's requirements.

We got a controller as automata form from JTLV(Java Temporal Logic Verifier)[6] which took as input centralized model of Poli-Heli specification of the form of LTL formulas, and checked realizable internally. It took a lot of time to generating automata which have 1,385 states. The following figure shows fragment of generated automata.

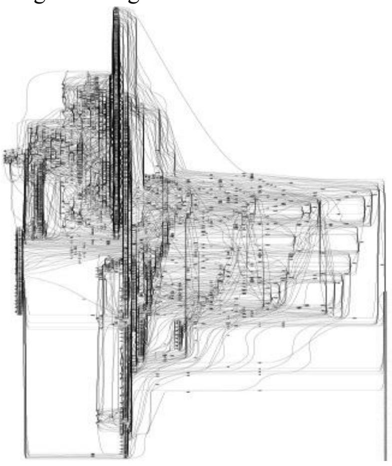

Fig. 2 Fragment of centralized controller

3.2 Decentralized Collaboration Model

To reduce or mitigate state explosion problem, we tried to specify Poli and Heli individually, and minimize information sharing for collaboration. In other words, Poli regards Heli as environment, and Heli regards Poli as environment.

As centralized model, Poli should know $\{help, r_4^H\}$ from environment. In decentralized model, variable duplication problem doesn't happen, so that we don't need to use superscript notation to *help* variable. Using these variables, we can specify environment's requirements of Poli as follows:

$$\neg help \wedge \neg r_4^H \quad \text{----------------} \tag{1}$$

$$\neg(r_1^P \vee r_2^P \vee r_3^P \vee r_4^P) \Rightarrow \neg\bigcirc help \quad \text{---} \tag{2}$$

$$\Box\Diamond\neg r_4^H \quad \text{------------------} \tag{3}$$

The environment's requirements of Poli are conjunction of formulas from (1) to (3). On the other hands, the system's requirements of Poli are as follows:

$$r_1 \wedge \neg r_2 \wedge \neg r_3 \wedge \neg r_4 \quad \text{-----------} \tag{4}$$

$$\bigwedge_{i=1}^{4} \Box(r_i \Rightarrow r_{(i+1) \bmod 4}) \quad \text{----------} \tag{5}$$

$$\Box \begin{pmatrix} (r_1 \wedge \bigwedge_{i\neq 1} \neg\bigcirc r_i) \\ \vee (r_2 \wedge \bigwedge_{i\neq 2} \neg\bigcirc r_i) \\ \vee (r_3 \wedge \bigwedge_{i\neq 3} \neg\bigcirc r_i) \\ \vee (r_4 \wedge \bigwedge_{i\neq 4} \neg\bigcirc r_i) \end{pmatrix} \quad \text{----------} \tag{6}$$

$$(r_3 \wedge \bigcirc r_4^H) \Rightarrow (\bigcirc r_3 \Leftrightarrow r_3) \quad \text{-------} \tag{7}$$

$$\bigwedge_{i\in\{1,2,3,4\}} \bigcirc help \Rightarrow (\bigcirc r_i \Leftrightarrow r_i) \quad \text{----} \tag{8}$$

$$\bigwedge_{i\in\{1,2,3,4\}} \Box\Diamond(r_i \vee help) \quad \text{----------} \tag{9}$$

The formula (4) is initial condition of system, and represents Poli starts from r_1. The formulas (5), (6), (7) and (8) are transition relation of system. Finally, the formula (9) is the goal of system, represents that Poli should infinitely often patrol r_1 from r_4 unless there is a need for help. The conjunction of formulas from (4) to (9) is requirements of Poli.

The requirements of Heli are the same with Poli, excepts that environment variables are $\{help, r_4^H\}$ and initial region and patrol regions of Heli are r_5 and r_5, r_6, r_7, r_4 respectively. Because requirements of Heli are the same with Poli except above variables, we describe Poli's requirements only.

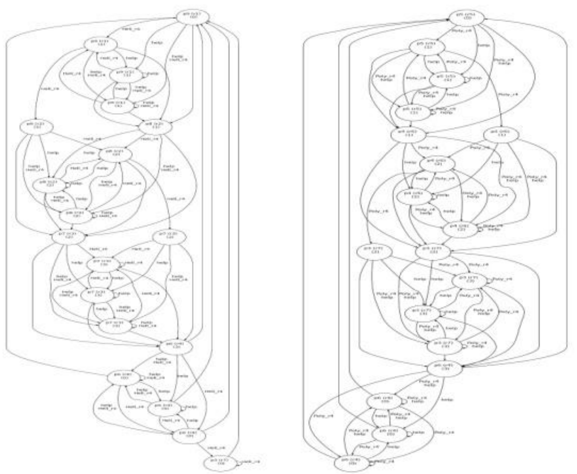

Fig. 3 Decentralized Poli-Heli controllers

The JTLV decided that realizability of decentralized model of Poli-Heli is true. So we got two controllers as automata form, each controls motion of Poli and Heli to satisfy requirements.

As shown in figure 3, the automaton of Poli has 19 states, and Heli has 18 states. Comparing the number of state of centralized model, the sum of states of two automata is much less than centralized model. Because Poli considered to critical section r_4 only, rather than considered whole regions, therefore Poli got a reduced automaton. Also, Heli was the same with Poli.

4. Collaborative Multi-Robot Simulator

In robot task planning, the LTLMoP[7] is a Python-based toolbox for robot simulation. However, the LTLMoP supported to simulate single-robot task planning only, so it was difficult to simulate collaborative multi-robot task planning. To overcome this problem, we extended the LTLMoP to simulate collaborative multi-robot task planning. The following figure shows architecture of multi-robot simulator.

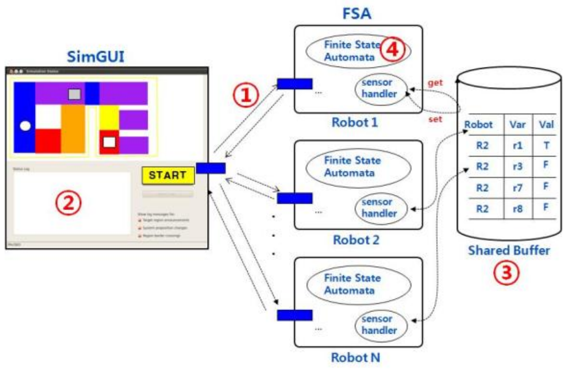

Fig. 4 The architecture of multi-robot simulator

To simulate multi-robot task planning, it is necessary to run FSA modules as the number of robots. As each FSA module needs a unique udp port, we modified to use an individual port for each FSA module, and extended protocol to transfer robot name. Also, in collaborative multi-robot task planning, each robot needs to recognize current region of other robots, we added shared buffer which stores current region of each robots. Finally, we extended simGUI module to store position and velocity of multi-robot.

To verify decentralized model of Poli-Heli, we simulated Poli-Heli scenario using extended LTLMoP. The figure 5 shows that the result of simulation using extended LTLMoP simulator. In this figure, circle denotes Poli, and rectangle denotes Heli. If Poli moves in r_4 already, Heli stops in r_7. After Poli passes in r_4, then Heli moves in r_4.

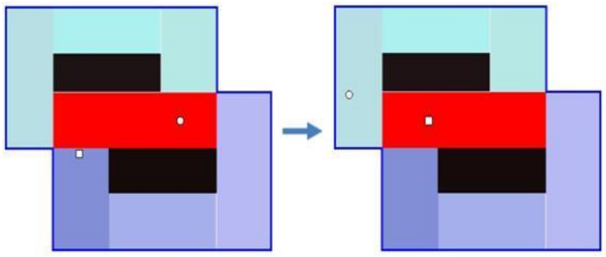

Fig. 5 Simulation of Poli-Heli collaboration

5. Conclusion

Now a day, there are many interests to controllers which control robots, automobiles, even medical devices dealing with human life according to the advancement of IT convergence. The LTL synthesis is a technique to generate correct and robust controller automatically. It takes as input requirements of LTL formulas, and checks realizability, if result is true then synthesis generates target controller as automata form.

It is possible to apply LTL synthesis to single robot controller. However in collaboration of multi-robot, the full state space of multi-robot controller is to be Cartesian product of each robot's state space. To mitigate this problem, we tried to use divide-and-conquer technique, so that we specified each robot's requirements individually, and to minimize shared information which enables each robot to collaboration. To verify this technique, we applied it to Poli-Heli scenario. As a result, we got the much reduced states automata. Also, we extended LTLMoP to simulate collaborative multi-robot. The extended LTLMoP helps to understand formulas and debug multi-robot specification through visualization of multi-robot task planning.

Reference

1. N. Piterman, A. Pnueli, and Y. Sa'ar. "Synthesis of Reactive(1) Designs", In Proc, 7th International Conference on Verification, Model Checking and Abstract Interpretation, 2006
2. T. Wongpiromsarn, U. Topcu and R. Murray, "Automatic Synthesis of Robust Embedded Control Software" In Proc, AAAI 2010.
3. Hadas Kress-Gazit, "Transforming High Level Tasks to Low Level Controllers, PhD Dissertation, 2008.
4. C. Baier and J. P. Katoen, Principles of Model Checking, Massachusetts Institute of Technology, 2008.
5. E. Allen Emerson. Temporal and modal logic. In Handbook of theoretical computer science (vol. B): formal models and semantics, pp.995-1072. MIT Press, Cambridge, MA, USA, 1990
6. A. Pnueli, Y. Sa'ar and L. D. Zuck. "JTLV: A Framework for Developing Verification Algorithms" In Proc, CAV, LNCS 6174, pp. 171-174, 2010.
7. LTLMoP, http://ltlmop.github.com/

Sunghae Kim
1996 : received B.S. in Computer Science from Kyonggi University
1998 : received M.D. in Computer Science from Kyonggi University
2008 ~ : under Ph.D. course in Computer Science from Kyonggi University
Research Area : Model Checking, Synthesis

Gihwon Kwon
1985 : received B.S. in Computer Science from Kyonggi University
1987 : received M.D. in Computer Science from Chung-Ang University
1991 : received Ph.D. in Computer Science from Chung-Ang University
1991 ~ : Full professor in the Dept. Computer Science, Kyonggi University
Research Area : Software Engineering, Formal Methods, Model Checking, Synthesis