LTL Synthesis를 이용한 다중 로봇 시뮬레이터 개발*

김성회, 권령구, 권기현
경기대학교 컴퓨터과학과
e-mail:{tprover, rkay, khkwon}@kyonggi.ac.kr

Development of Multi-Robot Simulator using LTL Synthesis

Sunghae Kim, Ryoungkwo Kwon, Gihwon Kwon
Dept of Computer Science, Kyonggi University

요 약

Synthesis 기술은 요구사항을 입력 받아서 자동적으로 시스템을 생성한다. 그러므로 생성된 시스템에 대한 추가적인 테스팅과 검증을 요구하지 않는다는 장점이 있지만 자연어가 아닌 특별한 형태의 논리식으로 요구사항을 기술하기 때문에 정확하게 변환되었는지, 누락된 요구사항이 있는지 확인하는 과정이 반드시 필요하다. 이러한 Synthesis 기술을 로봇 계획 분야에 적용하면 다중 로봇의 작업 계획도 단일 로봇과 동일한 형태로 확장하는 것이 가능하다. 그러나 기존의 LTLMoP 시뮬레이터는 단일 로봇의 시뮬레이션만을 지원해서 다중 로봇의 시뮬레이션은 어려움이 많았다. 따라서 본 연구에서는 다중 로봇들의 작업 계획도 시뮬레이션 할 수 있도록 LTLMoP 시뮬레이터를 확장하였고 사례연구를 통하여 이를 확인하였다.

1. 서론

하드웨어의 개발 속도는 빠른 반면에 소프트웨어의 개발 속도는 이를 따라가지 못하는 소프트웨어 생산성의 문제가 지속되고 있다. 이러한 문제가 발생되는 다양한 원인들 중 하나는 요구사항 변경에 따른 소프트웨어의 변경과 이에 따른 테스팅 또는 검증 작업의 증가를 들 수 있다. 소프트웨어 생산성 문제를 해결하기 위한 방안으로 소프트웨어 재사용성, 표준화, 자동화 등이 연구되고 있다.

Synthesis[1]는 요구사항으로부터 시스템을 자동적으로 생성하는 기술로써 정확한 요구사항을 입력하면 자동으로 시스템을 생성하기 때문에 추가적인 테스팅 또는 검증작업을 필요로 하지 않으므로 생산성을 높일 수 있다는 장점이 있다. 그러나 적용 대상이 반응형 시스템과 같은 분야로 한정되며 시제 논리의 한 종류인 LTL(Linear Temporal Logic)[2]을 사용하여 요구사항을 기술해야하기 때문에 정확한 요구사항을 기술하기 위해서 많은 경험과 반복적인 작업이 필요하다.

본 논문에서는 반응형 시스템의 한 분야인 로봇 작업 계획(Robot Task Planning)에서 사용되는 LTLMoP[3]라는 Synthesis 및 시뮬레이터 도구를 사용하였다. LTLMoP는 단일 로봇 작업 요구사항을 Structured English[4][5]로 기술하면 내부적으로 LTL 식으로 변환한 후 Synthesis 알고리즘을 사용하여 자동으로 계획을 생성하고 시뮬레이터를 통하여 로봇의 움직임을 보여주는 도구이다. 로봇이 정확하게 동작하기 위해서는 요구사항으로부터 작업 계획을 생성한 후 시뮬레이터를 이용하여 로봇의 움직임을 확인하는 과정이 필요하다. 단일 로봇 작업 계획의 경우 시뮬레이터를 통하여 쉽게 확인이 가능하지만 다중 로봇 작업 계획의 경우 생성된 작업 계획을 확인하는 것이 어려웠다. 이러한 문제점을 보완하기 위하여 본 논문에서는 기존의 LTLMoP 시뮬레이터에서 다중 로봇 작업 계획을 시뮬레이션 할 수 있도록 확장하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로써 로봇의 요구사항 기술과 작업 계획을 자동으로 생성하는데 사용되는 LTL과 Synthesis에 대하여 설명한다. 3장에서는 LTLMoP 도구를 설명하고 시뮬레이터에서 다중 로봇 작업 계획을 시뮬레이션하기 위하여 확장한 부분을 설명한다. 4장에서는 적용 사례로써 4대의 로봇이 협업하는 시나리오와 시뮬레이터를 사용하여 확인한 결과를 설명한다. 마지막으로 5장에서는 결론 및 향후 연구 과제에 대하여 기술한다.

2. 관련 연구

2.1 LTL의 구문과 의미

LTL은 시제 논리의 한 종류로써 선형 시간을 기반으로 한 논리이며 구문은 다음과 같다.

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

LTL은 명제 논리에 시제 연산자가 추가된 구문으로 정의되며 추가된 시제 연산자의 의미는 다음과 같다.

- ○ : next state
- ◇ : some future states(Eventually)
- □ : All future states(Globally, Always)
- U : Until

* 본 연구는 경기도의 경기도지역협력센터사업의 일환으로 수행하였음.[2011-0215, 모바일 플랫폼 기반의 콘텐츠 응용 소프트웨어 기술 개발]

위의 식에서 $p \in AP$ 는 더 이상 나눌 수 없는 Atomic Proposition을 말하며 ϕ 는 p 와 논리 연산자, 시제 연산자로 이루어진 논리식을 말한다. 위의 구문에는 나오지 않았지만, Conjunction($\wedge$), Implication($\Rightarrow$), Equivalence($\Leftrightarrow$)는 각각 다음과 같이 정의할 수 있다.

- Conjunction : $\phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2)$
- Implication : $\phi_1 \Rightarrow \phi_2 = \neg\phi_1 \vee \phi_2$
- Equivalence : $\phi_1 \Leftrightarrow \phi_2 = (\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$

LTL 논리식 ϕ 의 의미는 $p \in AP$ 명제 진리값의 무한한 순서로 정의된다. LTL 논리식의 의미에 대한 정확한 정의는 [2]를 참고한다.

2.2 LTL Synthesis

Synthesis의 의미는 시스템에 대한 요구사항을 입력 받아서 자동으로 시스템을 생성하는 것이다. 이렇게 생성된 시스템은 요구사항으로부터 자동적으로 생성되었기 때문에 "correct by construction" 이라 하며 생성된 시스템이 요구사항에 따라서 올바르게 구현되었는지 확인하기 위하여 별도의 테스팅 또는 검증을 요구하지 않는다.

본 논문에서는 Synthesis를 위하여 시제 논리의 특수한 형태 식을 사용한다. 이 특수한 형태는 $\phi_e \Rightarrow \phi_s$ 와 같이 구성된다. 이 식에서 ϕ_e는 환경의 행위에 대한 가정이며, ϕ_s는 시스템에 대한 기대되는 행위를 표현하고 있다. 각각의 식 ϕ_e와 ϕ_s는 다음과 같이 좀 더 세분화 할 수 있다.

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

위의 식 중 환경에 대한 가정을 기술하는 ϕ_e는 ϕ_i^e , ϕ_t^e , ϕ_g^e 식의 논리곱으로 연결되며 각각의 의미는 다음과 같다.

- ϕ_i^e : 시제논리가 사용되지 않은 불리언 식(B_i)으로 환경에 대한 초기 값을 나타냄
- ϕ_t^e : 환경이 가질 수 있는 모든 가능한 상태들을 표현하고 있는 논리식으로써 $\Box B_i$ 형태를 따르며 이러한 논리식이 논리곱으로 연결됨
- ϕ_g^e : 환경에 대한 목표 가정으로 $\Box\Diamond B_i$형태를 따름

시스템에 대한 가정을 기술하는 ϕ_s는 ϕ_i^s , ϕ_t^s , ϕ_g^s 식의 논리곱으로 연결되며 각각의 의미는 다음과 같다.

- ϕ_i^s : 시제논리가 사용되지 않은 불리언 식(B_i)으로 시스템에 대한 초기 값을 나타냄
- ϕ_t^s : 시스템이 가질 수 있는 모든 가능한 상태들을 표현하고 있는 논리식으로써 $\Box B_i$ 형태를 따르며 이러한 논리식이 논리곱으로 연결됨
- ϕ_g^s : 시스템에 대한 목표 가정으로 $\Box\Diamond B_i$형태를 따름

환경과 시스템에 대한 목표 가정으로 구성되는 특수한 형태의 식 $\phi_g^e \Rightarrow \phi_g^s$을 General Reactivity(1)(GR(1)) formula[1]라고 한다.

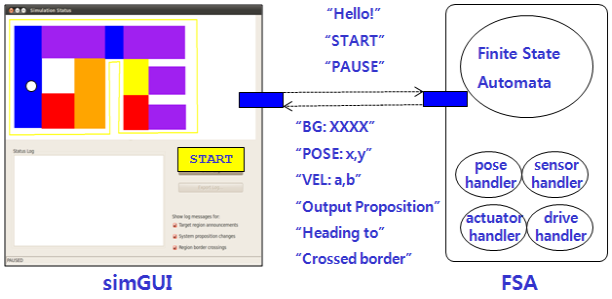
2.3 LTLMoP

LTLMoP(Linear Temporal Logic MissiOn Planning)는 로봇 계획 분야에서 사용되는 Synthesis 및 시뮬레이터 도구로써 로봇의 행위를 Structured English로 기술하면 내부적으로 LTL 식으로 변환 후 Synthesis 알고리즘[6]을 사용하여 자동적으로 로봇 계획을 생성하고 시뮬레이터를 통하여 로봇의 움직임을 보여준다.

3. LTLMoP 시뮬레이터 확장

3.1 LTLMoP 시뮬레이터 구조

LTLMoP 시뮬레이터는 GUI를 표현하는 simGUI와 자동 생성된 시스템 즉, 오토마타를 실행하는 FSA로 구성된다. FSA에서는 오토마타의 실행을 통하여 로봇의 다음 움직임을 계산하고 위치와 속도 정보를 UDP기반의 프로토콜을 사용하여 simGUI에 전송한다. simGUI에서는 좌표와 가속 정보를 수신하여 로봇이 시뮬레이션 되는 공간좌표에서 재계산한 후 로봇의 위치를 변경한다. LTLMoP 시뮬레이터 구조를 그림으로 설명하면 다음과 같다.

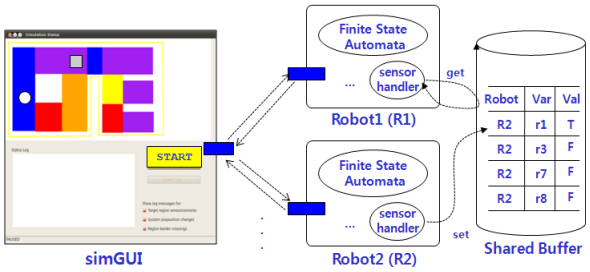


(그림 1) LTLMoP 시뮬레이터의 구조

3.2 다중 로봇 시뮬레이션을 위한 확장

LTL Synthesis를 사용하면 단일 로봇의 작업 계획을 확장하여 다중 로봇이 협업하는 작업 계획을 생성하는 경우에도 자연스럽게 표현할 수 있다. 왜냐하면 다중 로봇의 경우 각각의 로봇은 협력할 필요가 있는 다른 로봇의 정보를 환경 정보로써 인식하도록 추가로 기술하면 되기 때문이다. 그러나 LTL Synthesis를 사용하여 다중 로봇의 작업 계획을 생성하더라도 LTLMoP 시뮬레이터에서는 다중 로봇의 시뮬레이션을 지원하지 않기 때문에 정확하게 요구

사항대로 로봇이 협업하는지 확인하는 것이 어렵다. 따라서 기존의 LTLMoP 시뮬레이터를 확장하여 다중 로봇을 시뮬레이션 할 수 있도록 확장하였다. 다음 그림은 확장된 LTLMoP 시뮬레이터의 구조를 나타낸다.



(그림 2) 확장된 LTLMoP 시뮬레이터의 구조

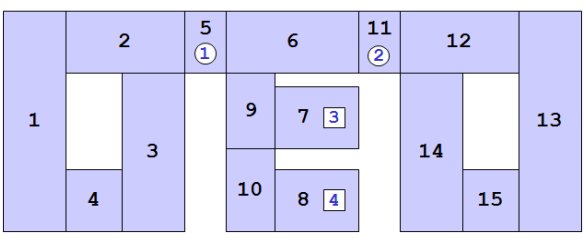
다중 로봇의 작업 계획을 시뮬레이션하기 위해서는 각각의 로봇에 대한 오토마타를 실행하는 FSA를 로봇의 개수만큼 실행해야 한다. 이때 simGUI와 로봇의 FSA는 UDP 포트를 사용하여 개별적으로 통신하므로 FSA마다 서로 다른 UDP 포트를 사용하도록 하였다. 이 Port 정보를 초기 "Hello" 프로토콜에 추가하여 정보를 교환하도록 하였고 로봇의 위치와 가속정보 프로토콜에 로봇의 이름을 추가하여 개별 로봇을 인식할 수 있도록 확장 하였다. 그리고 로봇들이 협업에 필요한 다른 로봇의 위치 정보를 파악할 수 있도록 Shared Buffer를 사용하여 로봇의 위치 정보를 공유할 수 있도록 하였다. 마지막으로 simGUI 내부에서 다중 로봇에 대한 위치와 가속 정보를 저장할 수 있도록 하였다.

4. 적용 사례

4.1 순찰 및 구조 시나리오

LTLMoP 시뮬레이터에서 다중 로봇의 작업 계획을 시뮬레이션하기 위한 순찰 및 응급구조 시나리오는 다음과 같다 : "로봇1과 로봇2는 위급한 사람이 있는지 확인하는 순찰 로봇으로써 로봇1은 지역 1, 2, 3, 4를 반복하여 순찰하고 로봇2는 지역 12, 13, 14, 15를 반복하여 순찰한다. 로봇3과 로봇4는 위급한 사람을 이동시키는 구조 로봇으로써 로봇3은 로봇1이 위급한 사람을 발견하는 경우 해당 위치로 이동하여 대응하고, 로봇4는 로봇2가 위급한 사람을 발견하는 경우 해당 지역으로 이동한다."

로봇의 요구사항 중 로봇3과 로봇4가 해당 지역에 도착하면 응급상황은 해제된다고 가정한다. 로봇의 작업 공간은 다음 그림과 같이 전체 15개의 구역으로 구성된다.



(그림 3) 로봇의 작업공간과 초기 위치

위의 그림에서 동그라미는 정해진 지역을 반복해서 순찰하는 로봇1과 로봇2를 나타내며 사각형은 응급환자를 이동시키는 로봇3과 로봇4를 나타낸다. 각각의 로봇이 있는 지역은 초기 시작 지역을 나타내는데 로봇1은 지역5에서, 로봇2는 지역11에서 시작하고 로봇3은 지역7에서 대기하고, 로봇4는 지역8에서 대기한다.

4.2 LTL을 이용한 모델링

로봇1의 요구사항을 LTL로 정의하는 것은 먼저 환경에 대한 가정과 로봇1의 행위를 정의하는 순서로 이루어진다. 위급한 사람을 감지하기 위한 로봇의 센서는 {*Help*}로 모델링하고 작업공간은 {r_1, r_2,..., r_{15}}로 모델링한다.

$$\phi_1^e = \begin{cases} \wedge\, \neg Help \\ \wedge\, \square(\neg(r_1 \vee r_2 \vee r_3 \vee r_4) \Rightarrow \neg \bigcirc Help) \\ \wedge\, \square\diamond True \end{cases}$$

환경의 초기 조건은 위급한 사람이 없다는 것과 r_1, r_2, r_3, r_4 이외 지역에서는 위급한 사람이 발견되지 않는다는 것이다.

$$\phi_1^s = \begin{cases} \wedge\, (r_5 \wedge_{i=2,\ldots,15} \neg r_i\ ) \\ \wedge\, Transitions \wedge Mutual\ \ Exclusion \\ \wedge_{i\in\{1,2,3,4\}}(\bigcirc Help \Rightarrow (\bigcirc r_i \Leftrightarrow r_i)) \\ \wedge_{i\in\{1,2,3,4\}}\square\diamond(r_i \vee Help) \end{cases}$$

시스템의 초기 조건은 로봇1이 r_5에서 시작한다는 것이며 *Transitions*은 로봇1이 이동 시 거쳐 가는 작업공간들의 연결 정보를 나타내고 *Mutual Exclusion*은 로봇1의 위치는 한순간에 오직 한곳에만 있을 수 있다는 제약조건을 나타낸다. 그리고 위급한 사람이 발견되면 그 지역에 멈춰 있어야 한다는 조건과 위급한 사람이 있는지 계속해서 감시하라는 조건이다. *Transitions*과 *Mutual Exclusion*은 LTLMoP에서 자동적으로 생성된다. 로봇2의 요구사항은 시작위치가 r_{11}이라는 점과 r_{12}, r_{13}, r_{14}, r_{15} 지역을 순찰한다는 것을 제외하면 로봇1과 동일하다. 구조 임무를 수행하는 로봇3의 요구사항 정의는 다음과 같다.

$$\phi_3^e = \begin{cases} \wedge\, \neg Help \\ \wedge_{i\in\{1,2,3,4\}}\square(r_i \Rightarrow \neg Help) \\ \wedge_{i\in\{1,2,3,4\}}\square((\neg r_i \wedge Help) \Rightarrow \bigcirc Help) \\ \wedge\, \square\diamond True \end{cases}$$

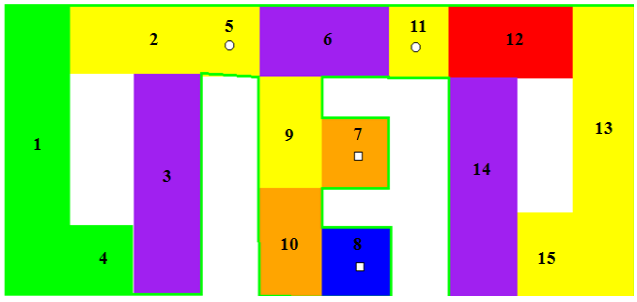
환경의 초기 조건은 위급한 사람이 없다는 것과 로봇3이 r_1, r_2, r_3, r_4 지역에 도착하면 위급한 상황이 해제되고 그 이외의 지역에서는 영향을 주지 않는다는 것이다.

$$\phi_3^s = \begin{cases} \bigwedge (r_7 \ \bigwedge_{i=1,\ldots,15} \neg r_i \ ) \\ \bigwedge Transitions \bigwedge Mutual\ \ Exclusion \\ \bigwedge \Box\Diamond(r_7 \Leftrightarrow \neg Help) \\ \bigwedge_{i\in\{1,2,3,4\}} \Box\Diamond(r_i \vee \neg Help) \end{cases}$$

시스템의 초기 조건은 로봇3이 r_7 지역에서 시작한다는 것이며 위급한 상황이 해결되면 r_7 지역으로 복귀하고 위급한 상황이 발생되면 해당 지역으로 이동한다는 조건이다. 로봇4의 요구사항은 시작위치가 r_8이라는 점과 r_{12}, r_{13}, r_{14}, r_{15} 지역으로 출동한다는 것을 제외하면 로봇3과 동일하다.

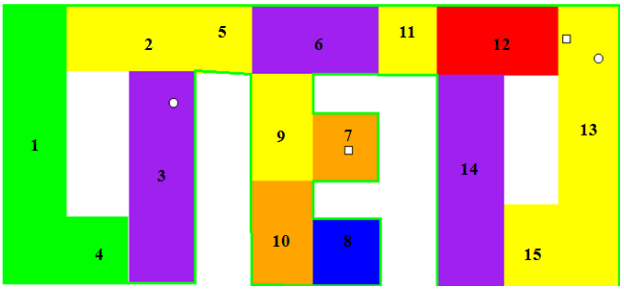
4.3 다중 로봇 시뮬레이션

확장한 LTLMoP 시뮬레이터를 사용하여 4대의 로봇이 작업하는 상황을 시뮬레이션 하였다. 시뮬레이션에서는 4대의 로봇 각각에 대하여 자동 생성된 오토마타가 실행되며 simGUI 시뮬레이터에서 순찰로봇은 원으로, 구조로봇은 사각형으로 표시된다.



(그림 4) 로봇의 초기 시작 위치

위의 그림은 로봇 4대의 초기 시작 위치를 나타낸다. 시뮬레이션이 시작되면 로봇1과 로봇2가 해당 지역으로 이동하여 순찰을 시작하게 된다. 만약 로봇1에서 *Help* 발생 시 로봇3이 해당 지역으로 이동하여 상황을 해결하게 되면 로봇1은 계속해서 순찰을 하게 되고 로봇3은 초기 위치 r_7 지역으로 복귀하게 된다.



(그림 5) 로봇4의 구조 활동 시뮬레이션

위의 그림은 로봇3이 13번 지역에서 위급한 사람을 발견한 경우 로봇4가 8번 위치에서 출발하여 13번 지역으로 이동한 시뮬레이션 결과를 나타낸다.

5. 결론 및 향후 연구

Synthesis는 요구사항을 입력받아 생성 가능한 경우 자동적으로 시스템을 생성해주는 장점이 있지만 자연어가 아닌 특별한 논리식을 사용해서 요구사항을 기술해야 하기 때문에 많은 경험이 필요하다. Synthesis를 통하여 자동 생성된 시스템은 추가적인 검증과 테스팅이 필요 없다고 할지라도 자연어로 기술된 요구사항을 논리식으로 변경하면서 요구사항이 정확하게 변환되지 못하거나 요구사항이 누락되는 상황이 발생될 수 있기 때문에 시뮬레이터를 사용하여 확인하는 과정이 필수적이다. 본 논문에서는 로봇 계획 분야의 도구인 LTLMoP 시뮬레이터를 확장하여 다중 로봇의 작업 계획을 시뮬레이션 할 수 있도록 확장하였고 4대의 로봇이 작업하는 상황을 시뮬레이션을 통하여 확인할 수 있었다. 향후 연구로는 로봇 작업 분야뿐만 아니라 다양한 분야에 대하여 Synthesis 기술을 적용하는 부분과 Synthesis 기술을 쉽게 적용하기 위해서 범용적인 환경에서 사용할 수 있는 다중 시뮬레이터 도구에 대한 연구가 필요하다.

참고문헌

[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. 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.

[3] 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.