Regular paper | Journal of the KIECS. pp. 709-714, vol. 18, no. 4, Aug. 31. 2023, t. 120, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2023.18.4.709

그래프 데이터베이스 기반 자동 PDDL Planning 시스템

문지윤*

Automated PDDL Planning System using Graph Database

Ji-Youn Moon*

요 약

유연한 planning system은 로봇이 다양한 임무를 수행하기 위해서 중요한 요소이다. 본 논문에서는 변화하는 환경에 대응할 수 있는 automated planning system architecture를 소개한다. 심볼릭 기반의 task planning을 위해 PDDL을 활용하였으며 실시간 환경 정보 업데이트를 위해 그래프 데이터베이스를 이용한다. 제안한 구조는 시나리오 기반 실험을 통해 검증하였다.

ABSTRACT

A flexible planning system is an important element for the robot to perform various tasks. In this paper, we introduce an automated planning system architecture that can deal with the changing environment. PDDL is used for symbolic-based task planning, and a graph database is used for real-time environment information updates for automated PDDL generation. The proposed framework was verified through scenario-based experiments.

키워드

Automated Planning System, Graph Database, PDDL, Symbolic Planning
자동 Planning 시스템, 그래프 데이터베이스, PDDL, 심볼릭 작업 계획

Ⅰ. 서 론

산업용 로봇이 등장한 이후로 로봇에 관한 연구는 현재까지도 활발하게 이루어지고 있다 [1, 2]. 초창기의 산업용 로봇은 인간이 수행하기에 위험하지만 단조로운 반복 작업에 주로 사용되었다. 최근에는 IT 기술과 인공지능의 발달로 로봇의 적용 분야 또한 점차 확대되었고, 지능형 로봇의 시대가 도래했다 [3, 4]. 자동차·전자제품 조립, 폭발물 제거와 같은 산업용 로봇부터 가사 지원, 교육, 수술·재활, 환경 분야에 응용할 수 있는 지능형 로봇까지 로봇의 활동 영역은 더 넓어졌다 [5, 6, 7].

심볼릭 접근법의 PDDL (Planning Domain Definition Language) 기반 시스템은 Planner가 물체의 행동을 계획하는 것을 효율적으로 하기 위한 표준 언어로 객체, 술어, 초기 상태, 목표 상태, 행동 등의 정보를 포함하고 있다. PDDL 시스템을 사용하려면 사람이 직접 모든 Domain과 Problem을 설계해야 한다. 이렇게 사람이 직접 작업하는 도메인 모델 습득을 수동 핸드 코딩이라고 한다. 하지만 데이터가 많아지

* 조선대학교 전자공학부(jymoon@chosun.ac.kr)
교신저자 : 조선대학교 전자공학부
• 접 수 일 : 2023. 06. 27
• 수정완료일 : 2023. 07. 20
• 게재확정일 : 2023. 08. 17
• Received : Jun. 27, 2023, Revised : Jul. 20, 2023, Accepted : Aug. 17, 2023
• Corresponding Author : Ji-Youn Moon
Dept. of Electronics Engineering, Chosun University
Email : jymoon@chosun.ac.kr

고, 로봇이 수행하는 task가 복잡해 짐에 따라 사람이 직접 모든 것을 설계하는 것은 어렵기 때문에 동적 환경에서 PDDL을 이용한 시스템을 적용하기는 어렵다. 이러한 수동 핸드 코딩의 단점을 개선하고자 최근에는 도메인 모델을 자동 생성하는 연구가 이루어지고 있다.

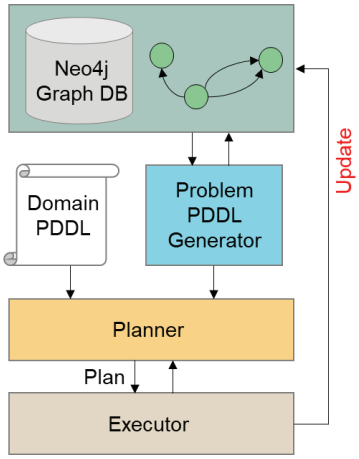

그림 1. 전체 구조
Fig. 1 Overall architecture

NL-to-PDDL은 자연어 입력 데이터를 사용하여 자동화하는 방식으로, 자연어로 작성된 데이터를 입력으로 받아들여 해당 PDDL 모델을 출력한다. 2가지 단계로 나뉘는데, 1단계는 언어 모델을 사전 훈련시켜 전달 학습을 활용한다. 2단계는 추출된 계획에서 PDDL 모델을 원샷 학습으로 유지하는 객체 중심 알고리즘을 사용한다. 이와 같이 자연어 입력을 사용하는 방법 외에 구조화된 데이터들을 이용하여 자동화하는 planning system에 대한 연구도 진행되고 있다. AMAN은 노이즈가 많은 plan에서 행동 모델을 학습하는 접근 방식이다. 먼저 도메인의 물리학을 설명하기 위한 그래픽 모델을 구축한 뒤, 그래픽 모델의 매개 변수를 학습하고 학습된 매개 변수를 기반으로 도메인 모델을 획득한다.

본 논문에서는 그래프 데이터베이스를 활용한 problem.pddl 자동 생성 방법을 제안한다. 전체 프레임워크는 그림 1을 통해 확인할 수 있다. Neo4jDB [8] 및 PDDL 2.1 [9, 10]을 사용하여, Neo4j를 통해 triple data 저장에 적합한 그래프 데이터베이스를 구축하고 problem PDDL generator program을 설계하여 problem.pddl 파일을 자동 생성한다. 제안한 방법을 통해 DB를 변화하는 상황에 따라 update 하여 자동으로 problem.pddl을 생성한다. 그 결과 우리는 자동적으로 replanning을 수행할 수 있으며 동적으로 변하는 환경에 실시간으로 대응가능하다. 2장에서는 자동 PDDL Planning 시스템 개발 방법에 대해 3장에서는 실험내용에 대해 설명한다.

II. 자동 PDDL Planning 시스템 개발

본 논문에서는 symbolic 접근법 기반의 planning system에서 많이 활용되는 PDDL 2.1을 활용한다. PDDL 2.1은 시간의 개념이 들어간 PDDL이다. PDDL을 이용한 planning 과정은 다음과 같다. 먼저 neo4j, problem PDDL generator program을 이용해 생성한다. 자동으로 생성한 problem 파일에는 시작 상태, 적용할 수 있는 행동, 원하는 목표 상태가 들어있다. problem 파일과 domain 파일을 AI planner를 통해서 추론한다. AI planner란 PDDL을 읽고 problem을 분석해서 추론하는 것으로 본 연구는 forwards-chaining temporal planner인 OPTIC planner와 POPF planner를 사용했다. planner를 통한 추론 과정은 다음과 같다. 시작 상태를 root로 하고, 목표를 달성하기 위해 필요한 모든 fact를 가지는 tree에서 root와 목표까지 깊이가 최단거리인 상태를 찾기 위해 검색한다.

그래프 데이터베이스는 기본적으로 개체를 나타내는 노드(node)와 노드 간의 관계를 지정할 수 있는 엣지(edges)로 구성된다. 노드, 엣지를 이용하여 직관적인 모델링이 가능하며, 데이터를 운용하기 유연하다. 오늘날 사용하는 데이터가 다양해지고 그 수가 증가함에 따라 그래프 데이터베이스를 택하는 사용자들이 많아지는 추세이다. 우리는 problem.pddl을 자동 생성하기 위한 그래프DB 시스템으로 neo4j를 이용한다. neo4j는 노드와 관계로 그래프를 나타낼 수 있는데, 속성(property)이라는 데이터 값을 추가할 수 있으며 노드를 분류하는 단위로 라벨(labels)이 존재한다. 또한 선언적 질의어를 사용하며, 인덱스 및 노드

탐색을 지원한다. 쿼리 결과를 시각화할 때는 노드와 관계를 포함한 그래프로 표현할 수 있다.

problem 파일 자동 생성을 위해 DB구성은 다음과 같이 한다. problem 파일에는 계획 수립을 위해 환경에 존재하는 물체들을 뜻하는 object와 object들간의 관계를 표현하는 predicate가 있다. 본 연구에서는 이 object들과 predicate를 분류하고 데이터베이스를 구축한 뒤, python과 연동해 원하는 데이터를 가져와 problem을 생성하는 방법으로 접근한다. problem 파일의 init은 초기 상태를 뜻하며, goal은 목표값을 의미한다. init과 goal은 주어-서술부-목적부의 트리플(triple) 형태로 나타낸다. neo4j는 노드-관계-노드의 형태로 db를 구성하여 init과 goal의 각각의 트리플에 해당하는 요소들을 대응하여 표현한다. 선언적 그래프 쿼리 언어인 Cypher를 활용해 problem PDDL generator program에서 neo4j 그래프 데이터베이스로부터 필요한 data를 가져온다.

는 neo4j와 Cypher를 활용하였다. 쿼리를 입력해 DB를 실시간으로 update하여 replanning이 필요한 때에 자동으로 problem 파일을 생성할 수 있도록 한다. 이 방법으로 replanning을 수행할 수 있으며, 동적인 환경에 적용할 수 있다는 강점이 있다.

III. 실험

본 연구에서 사용할 시나리오는 다음과 같다. 세 개의 로봇과, 물병, 씨앗 두 개, 각 지점이 8개 있다. 초기 상태는 robot1이 wp0에 위치하고, robot1은 bottle1을 가지고 있지 않다. 또한 bottle1은 wp3에 seed1은 wp5에 seed2는 wp7에 위치한다. 로봇은 물병이 있는 지점으로 이동하여 물병을 줍는다. 그 뒤 씨앗이 있는 지점으로 이동하여 씨앗에게 물을 주는

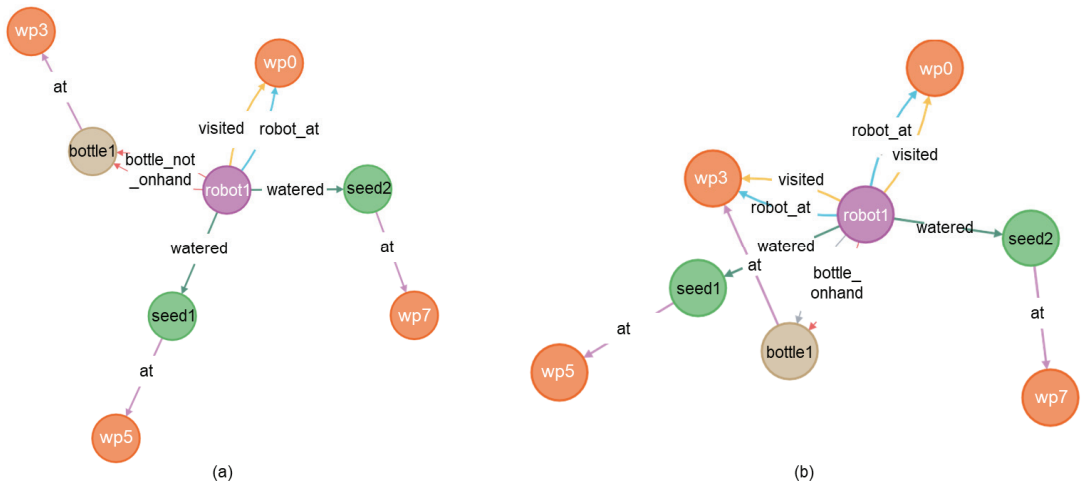

그림 2. 자동으로 생성된 그래프 데이터 (a) 첫 번째 생성된 그래프 데이터 (b) 두 번째 생성된 그래프
Fig. 2 Automatically generated graph database (a) First generated graph database (b) Second generated graph database

본 논문에서 설계한 problem PDDL generator program에서는 로봇이 planning 결과 출력한 plan을 수행하는 동안 실시간으로 해당 정보를 그래프 DB에 업데이트한다. 그래프 데이터 업데이트를 위해서 우리

것이 목표이다.

neo4j를 이용한 데이터베이스 구성은 다음처럼 한다. problem 파일의 object에 robot1, bottle1, seed1, seed2, wp0, …, wp7를 저장하기 위해 이 개체들을

neo4j의 노드로 설정하였다. 노드에서 로봇들은 robot 라벨을 지정해 분류하였고, 지점에 해당하는 wp0, …, wp7은 waypoint 라벨을 지정해 분류하였다. 또한 problem 파일의 predicate에 robot_at, visited를 저장하기 위해 이들을 neo4j에서 노드와 노드를 이어주는 관계로 설정하였다. 각 관계는 <어떤 로봇이 어떤 지점에 있다>를 뜻하므로 로봇 노드와 지점 노드 사이를 연결하는 관계로 지정하였다.

```
(define (problem demo)
(:domain demo)
(:objects
   robot1 - robot
   bottle1 - bottle
   seed1 seed2 - seed
   wp0 wp1 wp2 wp3 wp4 wp5 wp6 wp7 -
   waypoint
)
(:init
   (bottle_not_onhand robot1 bottle1)
   (visited robot1 wp0)
   (robot_at robot1 wp0)
   (at bottle1 wp3)
   (at seed1 wp5)
   (at seed2 wp7)
)
(:goal (and
   (bottle_not_onhand robot1 bottle1)
   (watered robot1 seed2)
   (watered robot1 seed1)
)))
```

그림 3. 첫 번째 생성된 problem PDDL
Fig. 3 First generated problem PDDL

problem 파일의 초기 상태에 “robot1이 wp0에 위치한다”를 저장하기 위해 각 neo4j에서 저장하려는 문장에 해당되는 관계 부분에 속성으로 ‘init’을 추가했다. robot1이 wp0에 위치한다면 그 지점을 방문한 것이므로 “robot_at wp0” 과 “robot1 visited wp0”가 ‘init’을 추가할 부분에 해당한다. 같은 방법으로 “robot2가 wp3에 위치한다”, “robot3이 wp6에 위치한다”도 초기 상태로 저장한다.

```
(define (problem demo)
(:domain demo)
(:objects
   robot1 - robot
   bottle1 - bottle
   seed1 seed2 - seed
   wp0 wp1 wp2 wp3 wp4 wp5 wp6 wp7 -
   waypoint
)
(:init
   (bottle_onhand robot1 bottle1)
   (robot_at robot1 wp0)
   (visited robot1 wp3)
   (visited robot1 wp0)
   (at bottle1 wp3)
   (at seed1 wp5)
   (at seed2 wp7)
)
(:goal (and
   (bottle_not_onhand robot1 bottle1)
   (watered robot1 seed2)
   (watered robot1 seed1)
)))
```

그림 4. 두 번째 생성된 problem PDDL
Fig. 4 Table.2 Second generated problem PDDL

마지막으로 problem 파일의 목표 상태에 “robot1이 wp1에 위치한다”를 저장하기 위해 문장에 해당되는 관계 부분에 속성으로 ‘goal’을 추가했다. 같은 방법으로 “robot1이 wp2에 위치한다”, “robot2가 wp4에 위치한다”,“robot2가 wp5에 위치한다”,“robot3이 wp7에 위치한다”도 목표 상태로 저장하였다. 이렇게 데이터베이스를 구축한 후에 다음과 같이 쿼리를 사용하였다. ″match (p) return Distinct labels(p)″ 쿼리는 라벨을 중복없이 리턴한다. object들을 가져오기 위한 것으로, 앞에서 생성한 robot1~robot3과 wp0~wp7 노드가 해당된다. python을 적용하면 각각의 라벨에 해당하는 노드의 이름들을 출력한다.

″match (a)-[r]->(b) where r.state='init' return type(r),a,b″ 쿼리는 state가 ‘init’인 관계들을 리턴하고, 연결된 노드들도 함께 출력한다. problem 파일의 init 상태에 해당하는 데이터를 가져오기 위함이다. type(r)은 관계를 출력하고, a, b는 노드를 출력한다. robot_at

robot1 wp0 과 같이 트리플 형태로 나타낼 수 있다. "match (a)-[r]->(b) where r.state='goal' return type(r),a,b" 쿼리는 state가 'goal'인 관계들을 리턴하고, 연결된 노드들도 함께 출력한다. problem 파일의 goal 상태에 해당하는 데이터를 가져오기 위함이다. type(r)은 관계를 출력하고, a, b는 노드를 출력한다. init과 마찬가지로 트리플 형태로 나타낼 수 있다.

이러한 쿼리를 응용하여 자동으로 데이터를 업데이트 할 수 있다. 따라서 우리는 초기에 설정한 상황이 변화하는 경우에 대해서도 제안한 방법을 통해 자동으로 problem 파일생성이 가능함을 확인할 수 있었다. 그림 2를 통해 초기 및 업데이트 된 그래프 DB 설정 상태를 확인할 수 있으며 그림 3을 통해 자동으로 생성한 problem PDDL 파일을 확인할 수 있다. 그림 4에서는 planner를 이용해 자동으로 생성한 problem PDDL을 이용하여 생성한 plan을 표 1과 표2를 통해 확인할 수 있다. 실험 결과 우리는 동적인 환경 변화에 대응하여 planning을 수행할 수 있었음을 확인할 수 있었다.

표 1. 첫 번째 생성된 problem PDDL의 plan 결과
Table 1. Plan for first generated problem PDDL

No	Action
1	(move_wp_without robot1 bottle1 wp0 wp3)
2	(grab robot1 bottle1 wp3)
3	(move_wp_with robot1 bottle1 wp3 wp5)
4	(water robot1 bottle1 seed1 wp5)
5	(move_wp_with robot1 bottle1 wp5 wp7)
6	(water robot1 bottle1 seed2 wp7)
7	(putdown robot1 bottle1 wp7)

표 2. 두 번째 생성된 problem PDDL의 plan 결과
Table 2. Plan for second generated problem PDDL

No	Action
1	(move_wp_with robot1 bottle1 wp3 wp5)
2	(water robot1 bottle1 seed1 wp5)
3	(move_wp_with robot1 bottle1 wp5 wp7)
4	(water robot1 bottle1 seed2 wp7)
5	(putdown robot1 bottle1 wp7)

Ⅳ. 결 론

기술이 발달하고 로봇 산업에 대한 관심이 증가해 로봇이 다양한 분야의 작업에 이용된다. 따라서 로봇이 가장 효율적으로 작업을 수행하도록 설계하는 것이 중요한데, 사람이 매 상황마다 행동을 설계해주는 방법은 비효율적이고 동적인 환경에 적용하기 어렵다. 이에 우리는 자동으로 환경정보를 업데이트해 planning할 수 있는 방법을 제안한다. 본 논문에서는 그래프 데이터 베이스 및 PDDL planning system을 이용하여 problem.pddl 파일을 자동 생성한다. Neo4jDB 및 PDDL 2.1을 사용하였으며, Neo4j를 통해 데이터베이스를 구축하고 직접 설계한 problem PDDL generator program으로 problem.pddl 파일을 생성한다. 실험을 통해 구축한 db를 실시간으로 update하여 replanning을 성공적으로 수행함을 확인할 수 있었다.

감사의 글
이 논문은 2023학년도 조선대학교 학술연구비의 지원을 받아 연구되었음.

References

[1] Y. S. Moon, Y. C. Bae, H. R. Cha, , S. H. Roh, J. K. Park, "The Development of Ecobot Robot for Friendly Environment Smart Home Appliance Application System," The *J. of the Korea Institute of Electronics Communications Sciences*, vol. 5, no. 4, 2010, pp. 480-485.

[2] J. H. Bong, S. H. Lee, D. J. Koh, , N. B. Kim, E. S. Park, D. R. Jeon, " Mobile Robot for Indoor Air Quality Monitoring," The *J. of the Korea Institute of Electronics Communications Sciences*, vol. 17, no. 3, 2022, pp. 537-542.

[3] J. Fan, Z. Pai, L. Shufei, "Vision-based holistic scene understanding towards proactive human–robot collaboration," *Robotics and Computer-Integrated Manufacturing* vol. 75, pp. 102304, 2022.

[4] J. Moon, "Plugin Framework-Based Neuro-Symbolic Grounded Task Planning for

Multi-Agent System," *Sensors,* vol. 21, no. 23, 2021, pp. 7896.
[5] S. Miglani, and N. Yorke-Smith, "Nltopddl: One-shot learning of pddl models from natural language process manuals," *In ICAPS'20 Workshop on Knowledge Engineering for Planning and Scheduling,* 2020.
[6] J. Moon, J. Moon, and S. Bea, "Control for Manipulator of an Underwater Robot Using Meta Reinforcement Learning," *J. of the Korea Institute of Electronics Communications Sciences*, vol. 16, no. 1, 2021, pp. 95-100.
[7] J. Fan, Z. Pai, and L. Shufei, "Vision-based holistic scene understanding towards proactive human‐robot collaboration," *Robotics and Computer-Integrated Manufacturing,* vol. 75, 2022, pp. 102304.
[8] J. Webber, "A programmatic introduction to neo4j," I*n proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity,* pp. 217-218, 2012.
[9] H. A. Geffner, "PDDL 2.1: Representation vs. computation," *Journal of Artificial Intelligence Research,* vol. 20, pp, 139-144, 2003.
[10] M.. Fox and D. Long. "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains," *Journal of Artificial Intelligence Research,* AI Access Foundation, December 1, 2003.

저자 소개

문지윤 (Ji-Youn Moon)

2014년 광운대학교 로봇학부 졸업(공학사)
2020년 서울대학교 대학원 전기정보공학부 졸업(공학박사)
2020년~현재 조선대학교 전자공학부 조교수
※ 관심분야 : 일반인공지능, 인지 로보틱스, 뉴로-심볼릭, 강화 학습, 임베디드 AI 컴퓨팅