네트워크 기반 로봇을 조종하기 위한 공통 명령 프로그래밍 언어(CCSLR)와 번역 시스템 구조

A Common Command-Scripting Language for network-based Robots (CCSLR) and Translator System Architecture

이 일 구[1] · 토 동[2] · 김 도 익[3]

Lee Il-Gu[1] · Dong To Nguyen[2] · Kim Doik[3]

Abstract A network-based robot [1] is a robot that explores service servers in the network environment for analyzing sensor data and making decision. Since network-based robot architecture was proposed, it's possible to reduce costs of robots. We hope robots would be all around at home environment. Therefore, normal users who are not experts need to be able to control those robots by using easy commands. We developed a scripting language, named CCSLR, to help users and developers who control various robots in ubiquitous environment. We focused on how to design the common language for various robots and how to translate a CCSLR script into a sequence of low-level commands of the target robot. In this paper, we propose scripting methods, with three layers. The CCSLR system reads the profile information from the knowledge representation database. Users don't have to know all about kinematical and mechanical details of a robot. Then again, the CCSLR system will use the profile information to translate the script into separated executable library commands. The CCSLR system manages robot's changing state every time a robot executes a command.

Keywords: Robot Programming, Scripting languages, Network-based humanoid architecture

1. 서 론

네트워크 기반 로봇[1]이 개발되면서 로봇의 일부 기능을 외부에 있는 서비스 서버에서 수행할 수 있기 때문에, 로봇이 점점 저렴해지고 있다. 따라서, 가까운 미래에 각 가정에서 로봇을 사용하게 될 것이다. 각 로봇은 여러 가지 작업을 수행할 수 있다. 게다가 로봇의 종류도 다양해서 현재의 기술로는 모든 로봇을 간단하게 조작하기가 쉽지 않다. 우리는 다양한 로봇을 공통된 방법으로 조종할 수 있도록 스크립트 언어를 제안한다. 제안하는 스크립트 언어, CCSLR은 일반 사용자와 로봇 전문가 모두가 사용할 수 있도록 개발되었다.

현재까지 로봇의 동작을 구성할 만한 프로그래밍 언어로 몇 가지가 제안되었다. 그런데, 이것들 대부분이 저수준 (low-level) 액츄에이터 제어에 치중하거나 고수준 (high-level)의 개략적인 로봇 동작을 구성하는 데에 치중되어 있다. 이런 두 가지 접근방식은 너무 특화되어 있어서 일반적으로 사용되기 어렵다.[2] 저수준 접근방식은 하드웨어에 의존할 수밖에 없기 때문에 여러 가지 로봇을 조종하는 데 적합하지 않다. 또한, 로봇에 대해 해박한 지식이 없는 일반 사용자가 배우기 어렵다. 반면, 고수준 접근방식으로 개발된 많은 프로그래밍 언어들이 웹에이전트(web agent)의 life-like character와 행동양식을 표현하기 위해 개발된 것들인데, 이것들을 실제 로봇에 적용시키기가 쉽지 않다.[3, 5] 실제 로봇에게는 많은 제약사항이 따르기 때문이다. 우리는 STEP[3] 언어와 비슷한 표현 방식과 문법의 언어를 사용하되, 몇 가지 새로운 특징들과 3 단계 번역 구조를 도입해서 고수준 언어 (high-level language)와 저수준 언어 (low-level language)의 결합점을 찾아 보았다.

먼저, 다양한 로봇에 적용할 만한 고수준 명령 집합을 만들기 위해서, 모든 명령은 로봇의 상태를 다음 상태로

※ 본 논문은 정통부 및 정보통신연구진흥원의 정보통신선도기반기술개발사업의 연구결과로 수행되었습니다.

[1] 과학기술연합대학원 대학교 HCI 및 로봇응용공학과 석사과정(E-mail : always19@skku.ac.kr)

[2] 과학기술연합대학원 대학교 HCI 및 로봇응용공학과 박사 (E-mail : todong@gmail.com)

[3] 한국과학기술연구원 인지로봇연구단 선임연구원 (E-mail : doikkim@kist.re.kr)

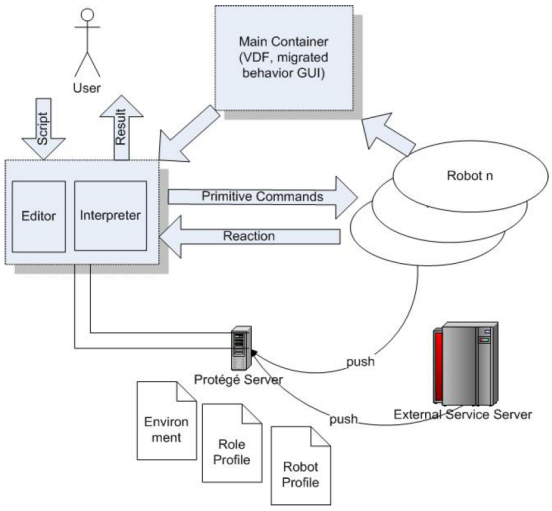

그림 1. Overview of the CCSLR System

변화시킨다고 간주했다. 기본 명령(Primitive Command)은 중간 과정이 없는 두 개의 상태를 연결하는 명령이다. 모든 명령은 두 상태를 직접적으로 연결하는 기본 명령으로 번역된다. 실제로는 두 상태 사이에 연속적으로 무한대인 상태들이 있지만, 이런 연속적인 상태들은 로봇을 조종하는 스크립트를 만들 때 사용할 필요가 없기 때문에 무시하기로 한다.

[그림1]은 사용자가 CCSLR 시스템을 사용해 로봇을 조종하는 과정을 개괄적으로 보여준다. 사용자가 여러 로봇이 수행할 일련의 작업들을 CCSLR 번역 시스템에 입력한다. CCSLR 번역 시스템은 편집기(Editor)와 번역기(Interpreter)로 되어있는데, 편집기는 사용자가 로봇 하드웨어에 대한 충분한 지식이 없어도 로봇이 수행할 명령을 편집할 수 있도록 정보를 제공한다. 이 때, CCSLR 번역 시스템은 Protégé Server와 Main Container를 이용한다. 그림에서 Main Container 안의 VDF [1]는 FIPA Standard의 DF [10]와 비슷한 역할을 한다. 사용자가 입력한 스크립트는 CCSLR 번역 시스템에 의해 기본 명령(Primitive Command)으로 번역되고, 기본 명령은 저수준 언어로 번역된다. CCSLR 시스템은 명령을 수행할 로봇을 선정해서 적당한 시기에 저수준 명령을 전송한다. 로봇은 명령을 실행하는 동안 CCSLR 시스템에 피드백을 보낸다. CCSLR 번역 시스템은 로봇이 명령을 끝마쳤을 때, 로봇의 변화한 상태를 저장하고, 그 결과를 사용자에게 알린다. 또한, 다음으로 수행할 명령을 다음 로봇에게 전송한다.

2. 관련 연구

Dance[2]는 휴머노이드의 동작을 “middle-level”로 추상화시켜서 표현하기 때문에, 다양한 용도로 확장하기가 용이하다 (scalable). Dance는 로봇의 반응을 고려해서 프로그래밍할 수 있게 해준다. 하지만, 일반 사용자가 배우기엔 다소 어렵다. 또한, 휴머노이드 로봇이 아닌 로봇들이 포함된 환경에서는 사용할 수 없다. 제안하는 CCSLR 언어는 3단계 번역시스템을 이용해서 Dance처럼 확장성을 확보하면서도 Dance의 단점을 극복했다. 제안하는 시스템에서는 일반 사용자라도 상위 단계의 CCSLR 명령을 이용하면 로봇에 대한 상세한 정보가 없더라도 로봇을 조종할 수 있다.

STEP[3]은 실체가 있는 에이전트(embodied agent)를 위한 스크립트 언어이다. 누구든 쉽게 배워서 스크립트를 만들 수 있도록 고안되었다. 3D 웹에이전트(web agent)의 제스쳐와 자세를 모델링하는 데에 중점을 두고 있다. 하지만, 실제 로봇에게 적용하기는 쉽지 않다. 그리고, 로봇의 반응을 고려하는 메커니즘이 부족하다.

MRL[4]은 robotic agent들을 조종하고 그것들이 통신하는 데에 사용하는 언어이다. robotic agent는 액츄에이터와 센서로 이루어진다. robotic agent들을 계층적인 모델로 구성해서 상위 에이전트(agent)와 하위 에이전트간의 통신방식을 정의했다. 각 에이전트는 MRL 메시지를 보낼 수 있고, 메시지 안에는 공동작업을 관리하는 공유 변수가 들어있다. 공유 변수를 통해 여러 로봇이 작업을 동기화할 수 있다. MRL은 동기화, 우선순위 제어에 뛰어나고, 협상 (negotiation)도 표현이 가능하다. 하지만, MRL은 저수준 로봇 명령에 능숙한 전문가만이 사용할 수 있다. CCSLR 언어는 처음 설계될 때부터, 로봇의 상태표현에 중심을 두었기 때문에, 공유변수를 이용하는 MRL의 동기화 제어 기법을 자연스럽게 사용할 수 있었다.

본 논문에서는 여러 가지 로봇을 동시에 조종할 수 있는 스크립트 언어를 제안한다. 제안하는 스크립트 언어는 다양한 로봇을 공통된 방법으로 조종하고, 일반 사용자부터 전문가까지 다양한 사용자들이 사용할 수 있도록 3단계 번역 구조를 갖는다.

3. 명령과 상태와의 관계

한 에이전트는 다른 에이전트의 상태를 변화시키기 위해 통신을 한다. 때로는, 자신의 상태를 알려서 다른 에이전트의 정보 상태를 변화하기도 한다. 통신은 다른 에

이전트의 행동에 영향을 미친다.

사람과 로봇과의 관계에서 이는 더욱 명확하다. 사람은 로봇에게 일방적으로 명령을 하고, 사람의 명령은 로봇의 상태를 변화시킨다. 예를 들어, 사용자가 로봇이 앞으로 가도록 명령을 내렸다면, 로봇의 이전 상태에서 변한 것은 오직 로봇의 위치 정보이다. 로봇 사용자는 로봇의 위치 변화 이외에는 아무것도 원하지 않았을 것이다. 로봇 자세의 변화나 남아있는 배터리의 양 등 다른 변화들이 일어나지만 로봇 사용자가 원하지 않는 부작용일 것이다. 로봇 사용자가 변하지 않기를 원했을 수도 있고, 신경을 쓰지 않는 것일 수도 있다. 우리는 변화한 위치 정보에만 관심을 갖기로 한다. 이런 생각은 무엇을 할 지 기술하되, 어떻게 할 지 기술하지 않는 functional programming의 사상과 닮아있다[2]. 우리는 로봇의 상태변화를 중심으로 명령을 설계한다면, 각종 로봇이 비슷한 구조의 공통된 명령을 가질 수 있다고 결론지었다.

게다가, 모든 명령이 완전히 실행되거나 아예 실행되지 않는 atomic 명령이라고 가정하면, 로봇을 유한 상태 오토마타(Finite state automata)로 가정할 수 있고, 방향성 있는 그래프로 표현할 수 있다. 로봇의 상태를 노드로 표현하고, 실행 가능한 명령을 방향성 있는 링크로 표현한다. 로봇의 상태전이 다이어그램 (state transition diagram)을 구성한다. 고수준인 공통의 CCSLR 명령을 구성했고, 이 명령들은 상태 변화를 기준으로 분류되었기 때문에, 다양한 로봇이 하드웨어에 무관하게, 같은 상태전이 다이어그램을 가진다. CCSLR 시스템은 각 로봇의 상태를 기억하고, 그것을 기반으로 적합한 다음 행동을 추론해서 planning에 이용할 수 있다. 사용자 또한, 로봇의 상태를 명령 실행 조건으로 이용하면, if-then rule이 가미된 로봇 프로그래밍이 가능하다.

4. CCSLR 번역 단계

CCSLR 시스템은 CCSLR 명령을 목적 로봇의 저수준 명령으로 번역한다. 저수준 명령으로 번역할 때까지는 몇 단계의 과정을 거친다. 번역 단계 1은 로봇의 역할과 행동에 따라 다른 CCSLR 스크립트를 출력하는 과정이다. 역할 프로파일은 사용자가 입력하는 로봇의 역할이나 행동에 따라 다른 스크립트로 번역되도록 대응 관계를 저장하고 있다 번역 단계 2는 CCSLR 명령으로 이루어진 스크립트를 CCSLR 기본 명령들로 바꾸는 작업이다. 모든 CCSLR 명령이 기본 명령인 것은 아니기 때문에, 일부 명령을 재배열하고 다른 명령으로 바꿀 필요성이 있다. 번역 단계 3은 저수준 언어와 일대일 대응되어있는 CCSLR 기본 명령을 번역하는 단계이다. CCSLR 기본 명령은 로봇의 상태를 바꾸는 최소단위이다. CCSLR 명령은 상태변수를 포함한다. 상태변수는 여러 명령을 동기화하는 데 사용된다. 재사용 가능한 스크립트를 다른 사용자에게 배포할 때에는 로봇의 역할(role)과 행동(behavior)를 부여한다.[그림 2].

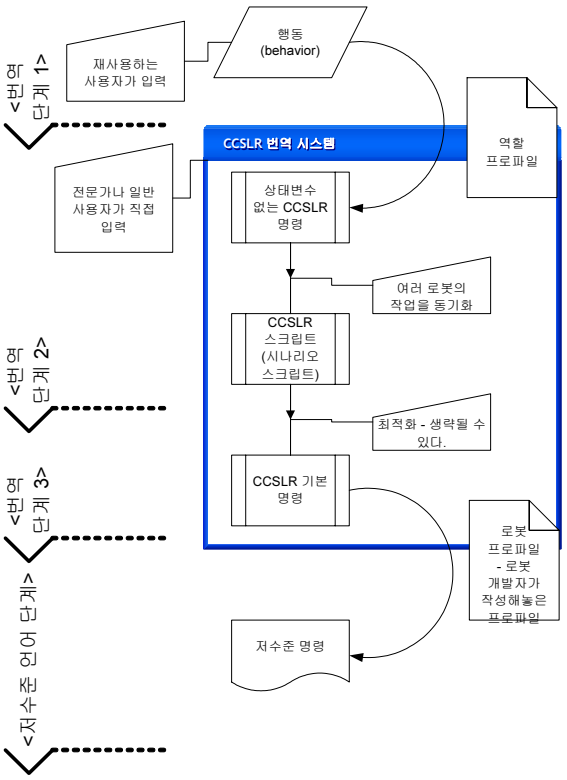

그림 2. CCSLR을 번역하는 과정

4.1 번역 단계 1

일반 사용자는 단순히 목적기반의 (Goal-based) 로봇의 행동만을 입력하는 것으로 로봇을 조종할 수 있다. 번역 단계 1은 로봇의 행동을 CCSLR 스크립트 (일련의 CCSLR 명령)로 바꾸는 과정이다. 시나리오 스크립트는 에이전트들의 행동양식과 순서를 CCSLR 명령으로 구성한 파일로써. 상태 변수를 포함하는 CCSLR 스크립트이다. 로봇 전문가는 시나리오 스크립트를 만들어서 다른 사용자에게 배포할 수 있다. CCSLR 번역 시스템은 시나리오 스크립트를 읽고, 어떤 로봇이 어떤 시각에 어떤 명령을 수행할 지 결정해서 CCSLR 명령어 집합을 결과로 내놓는다. 이 때, 여러 로봇이 각각 다른 시나리오 스크립트를 실행해야 하므로, 상태변수를 다시

구성해야 할 필요가 있다. 이 과정에서 로봇간의 공동수행 (coordination) 문제를 해결한다.

공동수행 문제는 공유 변수를 기반으로 명령의 실행순서를 결정하는 [4]의 아이디어로 해결한다. 모든 CCSLR 명령이 이전 상태와 다음 상태를 연결하기 때문에, 다음 명령은 로봇이 적합한 상태에 도달했을 때에만 실행하도록 만들 수 있다. 공동수행 문제 관점에서는 CCSLR에서의 로봇 상태가 [4]의 공유 변수에 해당한다.

'역할 프로파일'은 CCSLR 번역 시스템이 사용자가 원하는 로봇의 행동에 따라 알맞은 시나리오 스크립트로 선택할 때 사용한다. 각 로봇은 하나의 역할을 가지고 있고, 이 역할은 실행시간에 바뀔 수 있다. 시나리오에 어떤 로봇에게 같은 행동을 수행하게 표현되어 있더라도, 로봇의 역할이 다르다면 로봇은 다르게 동작한다. 때로는 사용자가 어떤 로봇에게 이전 로봇과 같은 역할을 주고, 이전 로봇이 행동했던 양식을 그대로 실행하게 할 수도 있다. 이는 CCSLR 명령이 로봇 하드웨어에 무관하게 실행되도록 디자인되었기 때문에 가능하다.

4.2 번역 단계 2

사용자는 CCSLR 언어를 이용해서 로봇 하드웨어에 독립적으로 로봇의 동작을 프로그래밍한다. 번역 단계 2는 사용자를 위해 개발된 다양한 CCSLR 명령을 저수준 명령으로 번역하기에 용이한 몇몇 CCSLR 기본 명령으로 바꾸는 과정이다. 이 단계에서는 일반적인 컴파일러가 최적화 단계에서 수행하는 Instruction Scheduling과 유사한 작업을 수행한다. 저수준 명령으로의 번역 규칙인 '로봇 프로파일'을 간단하게 만들기 위해서 CCSLR 기본 명령은 저수준 명령과 일대일 대응되어 있다. 따라서, 사용자가 입력한 CCSLR 명령들을 CCSLR 기본 명령들로 변환하고 다시 배열해야 할 필요성이 있다. 대부분의 작업은 통합작업이거나 분해작업이다. 만약, 사용자가 로봇의 팔을 같은 방향으로 2초 돌리는 명령을 연속적으로 두 번 실행시키려 한다면, 로봇의 팔을 4초 돌리는 하나의 명령으로 통합한다.

기본 명령이 아닌 CCSLR 명령은 로봇의 저수준 명령에 의존적이지 않기 때문에, 로봇의 저수준 명령에 맞춰서 여러 개의 기본 명령으로 바뀔 때도 있다. 로봇의 팔을 움직이는 저수준 명령 한 문장이 6축을 한꺼번에 돌리는 명령이라면, 로봇의 팔꿈치와 어깨를 동시에 돌리는 CCSLR 명령은 한 문장의 저수준 명령으로 표현되어야 한다. [그림 3] 한편, 사용자가 로봇의 팔꿈치를 오른쪽으로 돌리는 명령을 실행시키려 할 때, 해당 로봇의 팔꿈치를 오른쪽으로 꺾을 수 없고, 앞뒤 방향과 시계, 반시계

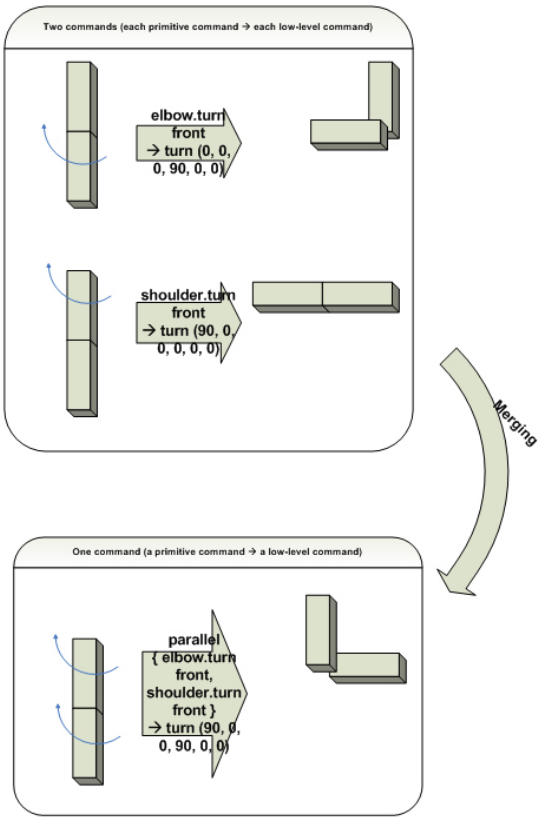

그림 3. CCSLR 명령 통합작업의 예

방향(위에서 봤을 때)으로밖에 회전할 수 없다면, 사용자가 입력한 하나의 명령을 두 개의 명령으로 바꿔서 동시에 실행해야만 사용자의 의도와 동등한 결과를 얻는다.

4.3 번역 단계 3

번역 단계 1은 저수준 언어와 일대일 대응되어있는 CCSLR 기본 명령을 번역하는 단계이다. 번역 단계 1에서 사용하는 번역 규칙은 '로봇 프로파일'에 저장되어 있다. 로봇 프로파일은 Protégé-OWL로 작성되어 있다. 로봇 개발자는 CCSLR 시스템이 알맞은 저수준 명령을 찾아낼 수 있도록 하나의 CCSLR 명령을 대응시켜 놓아야 한다. 번역 규칙을 포함하는 '로봇 프로파일'을 작성하려면 Protégé ontology 편집기로 '로봇 프로파일' 파일을 열고, 원하는 명령 파라미터에 해당하는 property의 치역에 적합한 CCSLR 명령 파라미터의 individual을 넣으면 된다.

Web ontology language(OWL)[9]는 물리적이거나 추상적인 개체를 모두 표현하기 위한 지식 표현 언어이다. 로봇 프로파일은 로봇에 대한 정보를 체계적이고 일반적으

로 표현하기 위해서 OWL을 사용했다. 그 구조는 CCSLR 번역 시스템을 만들 때부터 정해져 있고, 로봇 개발자는 이미 설계된 로봇 프로파일의 해당 부분에 자기 로봇에 대한 정보만 채워 넣으면 된다. 로봇 개발자는 Ontology와 지식 표현 기술에 익숙하지 않아도 된다.

5. 번역 과정

이 절에서는 사용자가 CCSLR 시스템을 이용해서 로봇을 조작하는 과정을 예를 들어 설명한다. 사용자는 2개의 로봇에게 "Greetings" 행동(behavior)를 실행시키려 한다. "Greetings" 행동을 입력하는 것은 단계 3의 입력이다. 각 역할 프로파일 안에는 Greetings에 해당하는 CCSLR script가 들어있다. 사용자는 Waiter 역할과 Boy 역할에 해당하는 행동을 실행시키려 한다. 다음은 역할 프로파일의 내용이다.

```
Boy's Greetings: { go front fast, wave_
      hands slow, go back fast }
Waiter's Greetings: { bow slow }
```

번역 단계 1을 수행하기 위해서 CCSLR 번역 시스템은 역할 프로파일에 있는 CCSLR 명령들을 확인하고, 이 역할을 수행할 알맞을 로봇을 선정한다. 이 때, Protégé-OWL로 작성한 로봇 프로파일을 확인해서 어떤 로봇이 해당하는 행동을 실행할 수 있는 지 검사한다. CCSLR 시스템은 Boy 역할에 해당하는 로봇으로 humanoid_01을 선정하고, Waiter 역할에 해당하는 로봇으로 mobile_01을 선정했다. 사용자가 원한다면, 사용자가 로봇을 선정하는 것도 가능하다.

다음은 시나리오에 맞게 명령의 실행순서를 정한다. Boy, Waiter 역할의 행동을 실행할 때, 가운데 명령을 동시에 실행하도록 정한다.

다음은 재구성된 CCSLR 스크립트이다.

```
humanoid.self.go front fast <0, 1>
humanoid.r_hand.wave_hands slow <1, 2>
humanoid.self.go back fast <2, 3>
mobile.self.bow slow <1, 4>
```

다음은 위 CCSLR 스크립트의 개략적인 문법을 BNF형식으로 작성한 것이다. 위 스크립트 중 한 문장(한 줄)이 '(명령문)'에 해당한다.

(명령문리스트) ::= (명령문) (명령문리스트)
(명령문리스트) ::= (명령문)
(명령문) ::= (명령)<(상태변수리스트),(상태변수)> \n
(상태변수리스트) ::= (상태변수), (상태변수리스트)
(상태변수리스트) ::= (상태변수)
(명령) ::= (에이전트).(부위).(함수명) (파라미터리스트)
(에이전트) ::= humnanoid | mobile | …
(부위) ::= self | r_hand | l_hand | …
(함수명) ::= bow | go | wave_hands | …
(파라미터리스트) ::= (파라미터) (파라미터리스트)
(파라미터리스트) ::= (파라미터)
(파라미터) ::= fast | slow | …

명령의 뒤에 따라오는 <(숫자), (숫자)>안에 들어있는 숫자들은 그 명령의 상태변수들이다. 로봇이 이 명령을 완전히 실행하면, 가장 뒤쪽에 있는 상태변수가 활성화된다. 앞쪽에 있는 상태변수들은 로봇이 이 명령을 실행할 시기인지 검사할 때 쓰이는 조건들이다. 위 예에서, 첫 번째 명령을 종료하면 로봇이 상태 1에 도달했다는 것을 CCSLR 시스템에 알린다. 상태 1은 <0, 1>에서 뒷부분에 있는 숫자에서 기인한다. CCSLR 시스템은 상태 1을 실행 조건으로 하는 명령들을 로봇에게 전송한다. 두 번째 명령과 네 번째 명령이 상태 1을 실행 조건으로 하기 때문에, 이 두 명령은 첫번째 명령이 완수된 직후에 실행된다.

즉, 모든 명령은 기본적으로 조건문이 된다. 한 명령의 실행 조건은 두 개 이상일 수도 있다. 상태 변수는 명령의 실행순서를 결정하는 중요한 요인이다.

이번 예제에서는 시나리오 스크립트에 있는 CCSLR 명령이 모두 기본 명령이어서, 번역 단계 2를 생략할 수 있었다.

마지막으로, 번역 단계 3에서는 각 명령은 로봇 프로파일을 참고해서 해당 로봇의 저수준 명령으로 변환된다. 예를 들어, 첫 번째 명령은 보행패턴함수와 양수인 step-size 파라미터이면서 빠른 걸음에 해당하는 0.09로 변환된다. 반면, 세 번째 명령의 back 파라미터는 음수인 step-size 파라미터(-0.06)로 변환된다. 다음은 저수준 언어 단계의 내용이다.

```
walk_forward 8 2 0.09 0.15 0 0 2
wave_hands 9 4
walk_backward 8 2 -0.06 0.15 0 0 2
bow 10 4
```

CCSLR 시스템은 0번 상태를 활성화시켜서 첫 번째 명령을 humanoid_01 로봇에게 전송한다. humanoid_01은 명령을 수행하면서 자신의 상태를 CCSLR 시스템에게 알린다. 첫 번째 명령을 무사히 수행했다는 것을 알게 되면, CCSLR 시스템은 상태 1을 활성화시키고 두 번째, 네 번째 명령을 해당 로봇에게 전송한다. 이 때, CCSLR 시스템은 상태 1에 해당하는 humanoid_01 로봇의 변화한 위치 상태를 저장하고 있다. 두 명령이 모두 끝난 것을 확인한 다음에는 상태 2를 활성화시켜서 세 번째 명령을 humanoid_01 로봇에게 전송한다.

6. 결 론

로봇 프로그래밍은 오랜 역사를 가진 연구분야이다. 최근에 네트워크기반 로봇의 연구 경향에 따라, 분산환경에서 로봇을 프로그래밍할 만한 방법이 필요하게 됐다. 이 논문에서는 네트워크기반 로봇을 프로그래밍할 만한 새로운 개념을 소개했다. 사용자는 간단한 방식으로 모르는 로봇에게 명령을 전달할 수 있다.

CCSLR은 관련 연구에서 소개한 세 논문의 장점을 하나씩 취했다. [2]에서는 어떻게 로봇을 변화시킬 지 기술하는 것이 아닌 무엇이 얼마나 변했는지 기술하는 방식, [3]에서는 사용자 편의성과 재사용성, 그리고 [4]에서는 공유 변수를 통한 공동작업 수행방식을 차용했다. 이 세 가지 장점을 확고히 하기 위하여 3 단계 번역 방식을 사용했다. 또한, 상태 표현 개념을 도입해서 명령과 상태 변화간의 관계를 명확히 하였다. 명령어를 번역할 때에도 Protégé-OWL 지식 표현방식을 사용했다.

CCSLR 로봇 프로그래밍 언어와 번역 시스템을 사용하면, 다음과 같은 장점을 얻을 수 있다. 첫째, 여러 가지 로봇을 일관된 방법으로 조종할 수 있다. CCSLR 번역 시스템은 CCSLR 언어를 각 목적 로봇의 저수준 언어로 번역해준다. 둘째, 일반 사용자가 쉽게 이용할 수 있다. 사용자는 로봇 하드웨어나 기계적인 구조를 모르더라도 로봇이 원하는 행동을 실행하도록 명령할 수 있다. 셋째, 작성한 스크립트는 재사용이 가능하다. 로봇 개발자나 전문가가 샘플 스크립트를 만들어서 배포할 수 있다. 다섯째, 번역 규칙을 실행시간에 변경할 수 있다. 마지막으로, 여러 로봇이 공동작업을 할 수 있다. CCSLR 스크립트의 상태변수는 조건문과 동기화를 가능하게 만든다. 따라서, CCSLR 번역 시스템은 스크립트에 따라 로봇이 실행할 명령의 순서를 조정할 수 있다.

참고문헌

[1] Dong To Nguyen, Do-Ik Kim, Bum-Jae You and Sang-Rok Oh, (2006), NBHA – a distributed network-based humanoid software architecture, in Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida, USA pp. 456-461

[2] Liwen Huang and Paul Hudak, (2003), Dance: A Declarative Language for the Control of Humanoid Robots – Tech Report # YALEU/DCS/RR-1253

[3] Zhisheng Huang, Anton Eliens and Cees Visser, STEP: A Scripting Language for Embodied Agents – Proceedings of the Workshop on Lifelike Animated Agents, 2002

[4] Hiroyuki Nishiyama, Hayato Ohwada and Fumio Mizoguchi, (1998), A Multiagent Robot Language for Communication and Concurrency Control, in Proceedings of the International Conference on Multi Agent Systems, 1998. 3-7 July 1998 pp. 206 - 213

[5] Helmut Prendinger and Mitsuru Ishizuka, Life-Like Characters, Springer, Berlin Heidelberg, 2004

[6] Robin R. Murphy, Introduction to AI Robotics, The MIT Press, Cambridge, Massachusetts, 2000

[7] Stuart Russell and Peter Norvig, Artificial Intelligence – A Modern Approach 2nd Edition, Prentice Hall, Upper Saddle River, New Jersey, 2002

[8] Protégé – Ontology editor and knowledge-base framework
http://protege.stanford.edu/

[9] OWL – Web ontology language, w3c recommend-dation
http://www.w3.org/TR/owl-guide

[10] FIPA 2000 - Foundation for Intelligent Physical Agents http://www.fipa.org.

이 일 구

2005 성균관대학교 정보통신공학부 (공학사)

2005~현재 과학기술연합대학원대학교 한국과학기술연구원 캠퍼스 HCI 및 로봇응용공학과 석사과정

관심분야 : 로봇 프로그래밍 언어

토 동

1995 하노이 공과대학교 전자통신공학과 (공학사)
1997 하노이 공과대학교 전자통신공학과 (공학석사)
2007 과학기술연합대학원 대학교 한국과학기술연구원 캠퍼스 HCI 및 로봇응용공학과 (공학박사)
관심분야 : 네트워크 기반 로봇 아키텍쳐

김 도 익

1995 포항공과대학교 기계공학과 (공학사)
1997 포항공과대학교 기계공학과 (공학석사)
2002 포항공과대학교 기계공학과 (공학박사)
2002~2004 한양대학교 공학기술연구소 연구원
2004~2005 한국과학기술연구원 Post-Doc 연구원
2005~현재 한국과학기술 연구원 선임연구원
관심분야 : 네트워크 기반 휴머노이드