에이전트 구조

경기대학교 김인철*

1. 서 론

일반적으로 에이전트 구조(agent architecture)는 에이전트가 몇 개의 구성모듈로 이루어지며, 이 구성모듈들이 서로 어떻게 상호작용을 하는지를 나타낸다. 그리고 이와 같은 한 에이전트의 구성모듈들과 그들의 상호작용(interaction)에 대한 명세는 그 에이전트가 실세계로부터 감지한 데이터(sensor data)와 자신의 내부상태(internal state)를 바탕으로 현재의 행동(action)과 후속 내부상태를 어떻게 결정하는지를 설명해줄 수 있다. 이러한 관점에서 Maes는 에이전트 구조를 실제적으로 에이전트를 구현하기 위한 하나의 특별한 방법론이라고 보았다[10]. 에이전트를 구현하기 위한 가장 전통적인 접근방법은 물리적 기호체계 가설(physical symbol system hypothesis)에 기초한 기호주의 인공지능(symbolic AI) 방법이다. 이 접근방법의 큰 특징은 각 에이전트가 자신과 환경에 대한 기호모델(symbolic model)을 내부에 명시적으로 표현, 저장하고서 이 모델 위에서 논리적 추론을 전개함으로써 자신이 취해야 할 행동들을 결정해 나가는 구조를 가지고 있다는 점이다. 이러한 에이전트 구조를 일반적으로 숙고형 에이전트 구조(deliberative agent architecture)라 부르며, 계획 알고리즘과 지식기반 시스템 등 많은 전통적인 인공지능 연구노력들이 공통적으로 이 숙고형 에이전트 구조의 실현에 초점을 맞추어 왔다. 반면에 숙고형 에이전트 구조와는 달리 실세계에 대한 어떠한 기호모델도 세움이 없이 환경의 변화라는 자극신호(stimulus signal)에 대해 곧바로 반응(response)을 행할 수 있는 에이전트 구조를 반응형 에이전트 구조(reactive agent architecture)로 분류한다. 본 논문에서는 숙고형 구조, 반응형 구조, 그리고 이들을 결합한 혼합형 구조의 개념을 차례로 소개하고 이들에 대한 이해를 돕기 위해 각 유형별로 몇 가지 대표적인 에이전트 구조들을 살펴보고 향후 진행될 연구에 대해 전망해 본다.

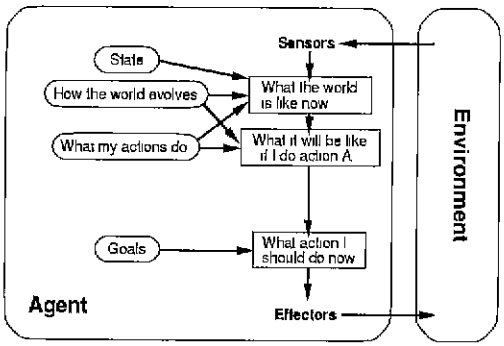

그림 1 숙고형 에이전트

2. 숙고형 에이전트 구조

그림 1에 나타낸 것과 같이 일반적으로 에이전트 자신이 놓여있는 환경(environment)과 추구하는 목표(goal), 그리고 가능한 행동(action)들에 대한 명시적인 기호모델(symbolic model)을 세우고 이 모델을 바탕으로 논리적 추론을 전개함으로써 현재 자신이 수행해야 될 행동을 결

* 정회원

정해 나가는 에이전트의 구조를 숙고형 에이전트 구조(deliberative agent architecture)라 부른다. 이러한 숙고형 에이전트 구조에서는 다음과 같은 중요한 두 가지 문제를 해결하여야 한다. 첫째는 끊임없이 변화하는 실세계의 상황을 어떻게 적절한 기호 표현으로 정확하게 옮길 수 있느냐 하는 변환 문제(transduction problem)이고, 둘째는 어떻게 하면 이러한 기호모델을 바탕으로 제한된 시간 내에 에이전트로 하여금 효과적으로 추론하게 만들 것이냐 하는 추론 문제(reasoning problem)이다. 전자의 문제는 로봇 비전, 자연어와 말 이해, 기계학습 등 주로 인식분야의 연구를 촉발시켰고, 후자의 문제는 지식 표현, 자동 추론, 자동 계획 등의 연구를 이끌어 왔다. 이와 같이 오랫동안 숙고형 에이전트에 관한 연구가 많이 진행되어 왔음에도 불구하고 효과적인 숙고형 에이전트의 구현이 어려운 이유는 정리 증명기(theorem prover)와 같은 기호 처리 알고리즘들은 근본적으로 주어진 시간 한계 내에 유용한 결과를 가지고 종료할 수 있다는 보장이 없다는 점이다. 따라서 끊임없이 변화하는 환경에 놓인 에이전트가 오직 이러한 기호처리에만 의존해 자신의 행동을 결정하려고 할 경우 시의 적절하게 환경에 반응할 수 없다는 것이다.

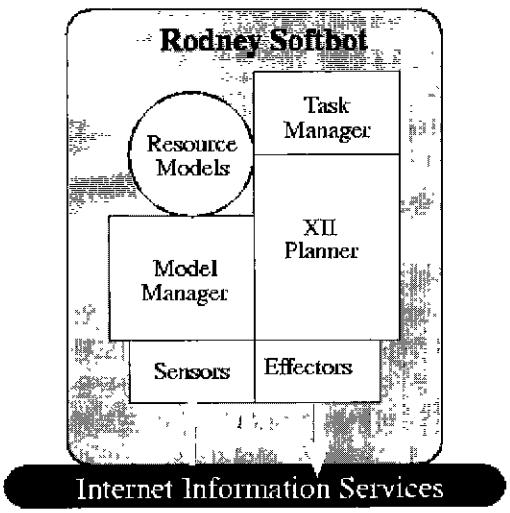

그림 2 Rodney Softbot의 구조

2.1 계획 에이전트 구조

1970년대 이후로 기호주의 인공지능 연구자들 사이에는 인공지능 계획시스템(AI planning system)이 어떤 형태의 에이전트에서든지 핵심 구성요소가 될 것이라는 가정을 사용해왔다. 가장 잘 알려진 초기 계획시스템인 STRIPS는 실세계의 초기상태와 목표상태, 그리고 전조건(pre-condition)과 후조건(post-condition)의 쌍으로 표현된 에이전트의 동작들이 기호로 주어지면, 이러한 기호모델을 바탕으로 목표상태에 도달할 수 있는 동작들의 순서를 찾아내려 했다. STRIPS 이후 단순한 STRIPS 알고리즘의 한계를 벗어나 보다 효율적인 계획 알고리즘을 찾고자 하는 노력이 계속되어 계층적 계획(hierarchical planning), 비선형 계획(nonlinear planning) 등 많은 계획방법과 계획시스템이 구현되었으나, 기호모델에 기초한 이러한 진보된 방법조차 제한된 시간 내에 효과적인 계획을 찾아내기 어렵다는 연구결과가 1980년대 중반 Chapman에 의해 발표되었다. 이러한 어려움에도 불구하고 계획기(planner)를 핵심 구성요소로 채용한 에이전트를 구축해보려는 연구가 계속되었다. 이러한 노력의 결과로 Ambros의 IPEM, Wood의 AUTODRIVE, Cohen의 PHEONIX 등이 개발되었다.

워싱턴 대학의 Etzioni와 Weld가 개발한 Rodney Softbot[5]는 UNIX 환경에서 스스로 계획을 세우고 행동함으로써 사용자를 위해 새로운 인터넷 인터페이스를 제공해주는 대표적인 소프트웨어 에이전트이다. Rodney Softbot는 그림 2와 같이 작업 관리자(task manager), XII 계획기(planner), 인터넷 자원모델(Internet resource model), 모델 관리자(model manager) 등 네 개의 주요 구성요소로 이루어져 있으며, 각각 전체 코드의 10%, 25%, 30%, 25%를 차지한다. 작업 관리자는 운영체제의 스케줄러와 같이 Rodney Softbot의 전반적인 활동을 제어하고, 인터넷 자원모델은 약 100 개의 인터넷 서비스 동작들을 UWL로 표현 저장함으로써 Rodney Softbot에게 인터넷 배경지식을 제공한다. UWL 동작표현의 특징은 기존 동작 표현법들과는 달리 에이전트의 감지동작(sensing action)과 정보수집 목표(information gathering goal)를 표현할 수 있다는 것이다. 한편 모델 관리자는 Rodney Softbot

가 실세계에 대해 관찰한 모든 것을 저장하는 특수한 데이터베이스이다. XII 계획기는 부분순서 계획 알고리즘을 기초로 UWL로 표현된 동작들에 대해 계획(planning)과 실행(execution)을 동시에 수행할 수 있는 통합구조이다. 가장 중심적인 역할을 하는 XII 계획기는 사용자가 원하는 작업에 대해 인터넷 자원 모델과 모델 관리자의 도움을 받아 스스로 작업계획을 세우고 실행하며 실행 결과에 따라 때로는 계획을 수정하기도 한다.

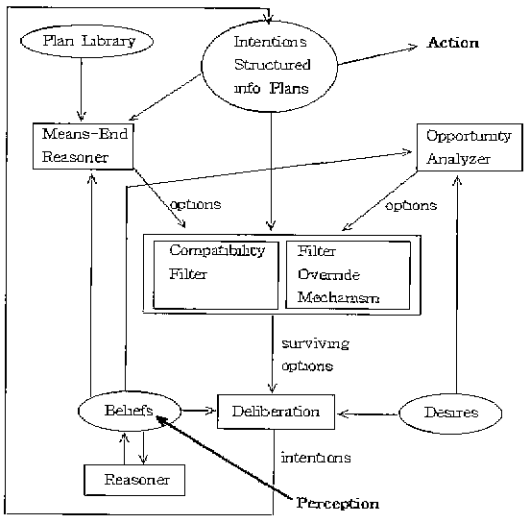

그림 3 IRMA의 구조

2.2 IRMA

IRMA(Intelligent Resource-bounded Machine Architecture)[2]는 믿음(belief), 소망(desire), 의도(intention)라는 세 가지 요소로 한 에이전트의 행위를 결정하는 BDI 에이전트 구조를 처음 구현한 것이다. 이 시스템은 자원에 대한 실시간 제약을 수용하여 이에 대한 적절한 반응을 보일 수 있는 대표적인 숙고형 에이전트 구조이다. 그림 3은 IRMA의 내부구조를 나타내고 있다. 그림에서 타원으로 표현된 것은 정보 저장소(information store)인 자료구조를, 사각형으로 표현된 것은 프로세스(process)를 각각 나타낸다. 그림에서 보듯이 IRMA는 계획 라이브러리, 믿음, 소망, 의도 등 기호로 표현되는 네 가지 핵심 자료구조를 내장하고 있다. 또한 IRMA는 이들 자료구조들을 처리하기 위한 다섯 가지 서로 다른 프로세스를 가지고 있다. 추론기(reasoner)는 실세계의 상황에 대한 추론을 담당하고, 수단-목적 분석기(means-ends analyser)는 에이전트의 현재 의도를 달성하기 위한 계획들을 결정하며, 기회 분석기(opportunity analyser)는 환경 변화를 지켜보다가 뜻하지 않은 상황이 발생하면 새로운 선택사항들을 제안하고, 필터링 프로세스(filtering process)와 숙고 프로세스(deliberation process)는 각각 에이전트의 현재 의도와 일치하는 행동(action)들을 가려내고 이들 중 하나를 선택하는 역할을 한다.

IRMA에서는 미리 프로그래머에 의해 주어지는 일반 계획들은 계획 라이브러리(plan library)에, 실제로 에이전트가 실행 중에 결정하는 구체적인 세부계획들은 의도 집합(intentions structured into plans)에 분리 저장된다. 일단 하나의 의도가 만들어지면, 수단-목적 분석기는 현재의 부분 계획(partial plan)들에 대한 세부 계획(subplan)들을 제안한다. 이와 같은 수단-목적 분석과정은 하나의 계획에 불일치성이 발생할 때까지 계속된다. 불일치성이 발생하면 이전 계획 상태로 돌아가 새로운 선택 사항(option)들이 제안된다. 한편 만약 뜻하지 않은 환경 변화가 일어났을 때는 기회 분석기가 새로운 선택 사항들을 제안한다. 수단-목적 분석기와 기회 분석기의 의해 제안된 선택 사항들은 필터링 프로세스로 전달된다. 호환성 필터(compatibility filter)는 제안된 선택 사항이 에이전트의 기존의 의도들과 호환성이 있는지를 검사하고, 호환성이 있다면 그것을 숙고 프로세스(deliberation process)으로 전달한다. 호환성이 없다고 판단된 선택 사항들은 필터 오버라이드 메카니즘(filter override mechanism)으로 전달된다. 여기에서 일정한 조건을 만족하는 것들에 한해 이미 의도된 계획들 중 일부를 중단하고 다른 선택 사항들을 고려할 수 있도록 비중을 높여준다. 숙고 프로세스로 전달된 선택 사항들은 에이전트의 소망과 현재의 믿음을 바탕으로 경쟁하게 되고 이 과정을 통해 선택된 선택 사항들에 따라 의도 집합내의 계획들을 확장하거나 변경한다. IRMA는 동적인 모의실험 환경인 Tileworld상에서 실험과 평가 작업이 이루어졌으며, 이를 통해 변화하는 환경 속

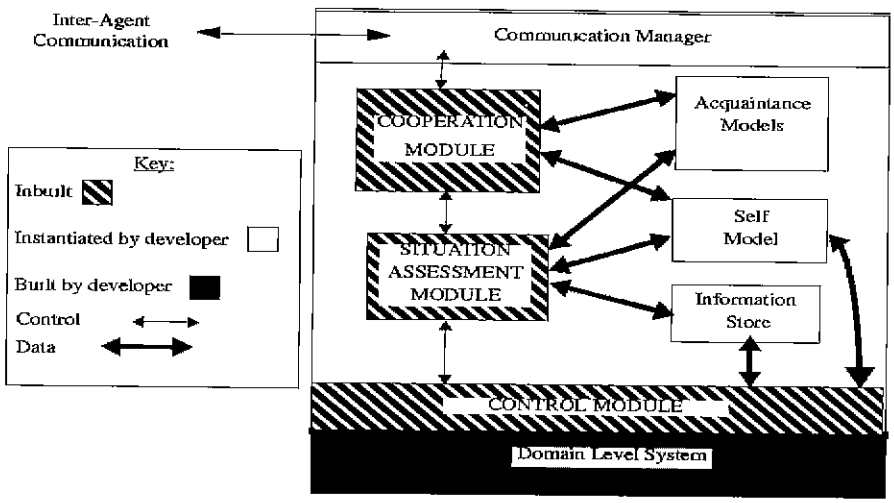

그림 4 GRATE 구조

의 한 에이전트에게 이미 결정된 계획에 대해 변경을 허용하는 것이 매우 중요한 전략임을 입증하였다.

2.3 GRATE

Jennings이 제안한 GRATE는 하나의 계층화된 구조로서, 에이전트의 모든 행위가 믿음(belief), 소망(desire), 의도(intention), 그리고 협동의도(joint intention)와 같은 내부적 속성들에 의해 유도되는 구조이다[8]. GRATE 에이전트 구조는 그림 4와 같이 영역-레벨의 계층(domain-level layer)과 협동 및 제어 계층(cooperation and control layer)으로 나뉜다. 전자는 주로 산업제어, 물류, 투자 등 구체적인 응용영역의 문제들을 푸는데 주력하는 반면, 후자는 영역 레벨의 계층 위에서 동작하며 영역-레벨의 활동이 다른 에이전트들의 활동과 조화를 이루도록 하는 메타-레벨의 제어기(meta-level controller)이다. 협동 계층은 제어 모듈(control module), 상황평가 모듈(situation assessment module), 그리고 협동 모듈(cooperation module) 등 세 개의 모듈들로 구성된다. 제어 모듈은 영역-레벨의 계층과의 인터페이스를 제공하고, 상황평가 모듈과 협동 모듈은 하나의 협동 책임(joint responsibility) 모델을 구현한다. 협동 책임 모델은 에이전트들이 협동적 문제 해결(cooperative problem solving)에 참여하고 있는 동안 다른 에이전트들에 대해 어떻게 행동해야 하는가를 명시한다. 전기 배전 관리 응용분야에서 GRATE 구조의 에이전트들을 개별 의도(individual intention)만을 가진 에이전트들이나 이기적인 방식으로 행동하는 에이전트들과의 비교를 통해 성능 평가를 실시한 결과 상황이 복잡해지고 변화가 심해질수록 GRATE 구조의 에이전트들이 뛰어난 성능 향상을 보였다.

3. 반응형 에이전트 구조

앞서 살펴본 숙고형 에이전트 구조와는 달리 에이전트 내부에 실세계에 대한 어떠한 기호모델도 갖지 않으며 복잡한 기호 추론과정도 사용하지 않는 구조를 반응형 구조(reactive architecture)라 한다. 반응형 에이전트 구조는 기호(symbol) 대신 실세계로부터 센서를 통해 감지한 신호(signal) 그 자체를 처리함으로써 곧바로 행동을 결정한다. 따라서 반응형 에이전트 구조는 환경 변화라는 자극(stimulus)에 대해 빠르게 반응(response)할 수 있다. Russell은 그림 5와 같이 어떤 내부 상태도 갖지 않고 단지 환경으로부터 감지된 현재 신호에 따라 행동을 결정하고 실행하는 유형의 에이전트를 단순 반응형 에이전트(simple reflex agent)로 규정하였다. 이러한 반응형 에이전트 구조는 들어오는 입력신

호를 인자로 취하는 간단한 함수나 회로로 구현 가능하므로 비교적 구현이 용이하다는 장점이 있다. 반응형 에이전트는 주로 빛이나 냄새에 따라 동작하는 곤충과 같은 단순한 지능을 가진 하급 동물들에 곧잘 비유된다.

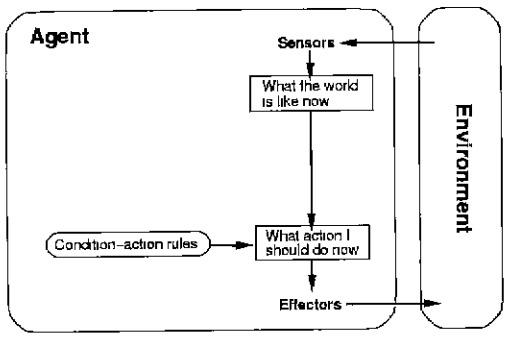

그림 5 반응형 에이전트

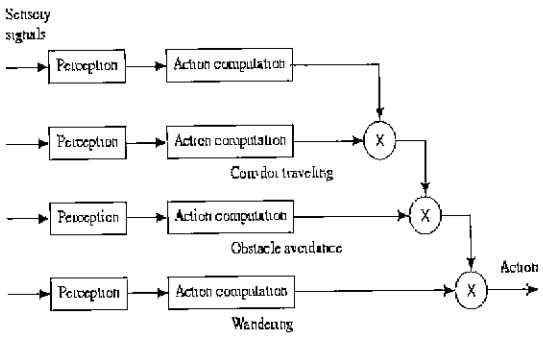

그림 6 포함구조

3.1 포함구조(subsumption architecture)

1985년 자율적인 이동 로봇을 제어하기 위한 전통적인 기호주의 인공지능 방법에 회의를 느낀 MIT의 Rodney Brooks 교수는 다음의 두 가지 점을 지적하며 새로운 에이전트 구조인 포함구조(subsumption architecture)[3]를 제안하였다. 첫째 진정한 지능은 정리 증명기(theorem prover)나 전문가 시스템(expert system)과 같이 실세계와는 유리된 몸체가 없는 시스템에 들어 있는 것이 아니라 실세계에 바로 놓여 있을 때 진정한 의미를 가질 수 있다. 둘째 지능적 행위란 에이전트가 자신의 환경과 상호작용을 가질 때 비로소 나타나는 것이며 따라서 환경 변화에 민감한 응급성(emergence)을 지니고 있다.

Rodney Brooks가 제안한 포함구조는 그림 6과 같이 에이전트의 행위에 대한 하나의 계층화된 제어구조(layered control system)이다. 이 구조에서 각 계층은 각기 비동기적으로 동작하는 모듈로 구성되며, 에이전트가 취해야 할 행동을 결정하기 위해 각 계층이 제안하는 행위들은 서로 경쟁관계에 놓이게 된다. 주로 하위 계층으로 내려갈수록 장애물을 피하는 등 비교적 기초적인 행위(primitive behavior)를 나타내며, 반면에 계층 레벨이 높아질수록 행위 능력(competence)은 증가한다. 상위 계층은 하위 계층의 출력을 억제함으로써 하위 계층의 역할을 포함(subsume)할 수 있다. 하지만 일반적으로 상위 계층들에 비해 하위 계층들에 우선권(precedence)을 부여한다. 이러한 구조에서는 비록 상위 계층들이 추가되더라도 계속해서 하위 계층들이 기능을 계속할 수 있으므로 매우 견고한 제어를 제공할 수 있다. 또한 이 구조는 기호주의 인공지능시스템에서 찾아볼 수 있었던 명시적인 추론과정이 없기 때문에 요구되는 계산 양의 면에서 매우 단순하다.

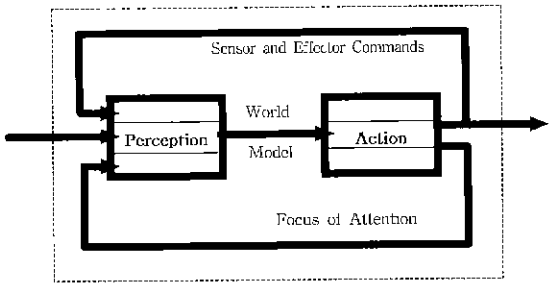

그림 7 상황 오토마타 에이전트 구조

3.2 상황오토마타(situated automata)

Brooks의 포함구조에서는 에이전트의 행위 제어시스템을 수평 분할(horizontal decomposition)하여 계층화하였으나 전통적인 접근방법에서는 인지(perception), 모델링(modeling), 계획(plannig), 실행(execution), 효과기 제어(effector control) 모듈들로 수직 분할(vertical decomposition)을 하였다. 포함구조와 같은 수평 분할은 프로그래머로 하여금 하위 레벨의 단순 행위들을 테스트한 다음에야 상위 레벨의 보다 복잡한 행위들을 구축할 수 있게 함으로써 시

스템을 점진적으로 구축하고 디버깅할 수 있도록 해준다. Rosenschein과 Kaelbling은 보다 지능적인 행위들의 구현하기 위해서는 센서로부터 들어오는 신호를 바로 이용하기 보다 이들로부터 별도의 인지과정(perception)을 거쳐 얻어진 정보를 이용하여야 한다고 생각하였다. 따라서 이들은 그림 7과 같이 한 에이전트의 내부구조를 인지 모듈과 행동 모듈로 크게 수직 분할을 하고, 포함구조와 같이 다시 이 각 모듈들을 수평 분할한 혼합구조를 제안하였다. 이때 행동모듈의 각 계층은 인지 모듈의 어떤 계층으로부터도 출력을 받아 이용할 수 있다.

Rosenschein과 Kaelbling은 이러한 구조의 에이전트를 구축하기 위한 방법으로 상황 오토마타를 제안하였다. 상황 오토마타는 에이전트를 하나의 선언적 고급언어(declarative high-level language)로 기술하면 이 명세를 만족하는 하나의 디지털 회로를 생성할 수 있는 프로그램으로 컴파일 해준다. 이 디지털 기계는 내부에 어떠한 기호처리도 포함하고 있지 않으므로 들어오는 입력에 매우 신속히 반응한다. 에이전트에 대한 명세는 형식논리(modal logic)를 이용하여 기술하며 에이전트의 구조에 맞춰 인식부와 행동부로 나누어 기술한다. 이 명세로부터 에이전트를 합성해내기 위해서는 RULER와 GAPPS라는 두 개의 프로그램을 사용한다.

4. 혼합형 에이전트 구조

많은 에이전트 연구자들은 완전한 숙고형 구조나 완전한 반응형 구조로는 효과적으로 에이전트를 구축할 수 없다고 판단하고 이 두 가지를 결합한 혼합형 구조를 제안하였다. 이 혼합형 구조에는 복잡한 추론과정 없이 환경변화에 바로 반응할 수 있는 반응형 모듈과 실세계에 대한 기호모델을 바탕으로 계획을 세우거나 의사결정을 내리는 숙고형 모듈을 함께 내포하고 있다. 그리고 숙고형 모듈에 비해 반응형 모듈에 더 우선권(precedence)을 부여함으로써 중요한 환경변화에 빠른 반응을 제공할 수 있도록 하였다. 따라서 대부분의 혼합형 구조는 TouringMachine이나 InteRRaP과 같이 하나의 계층 구조로 되어 있다. 이와 같이 계층화된 혼합형 에이전트 구조에서는 보다 상위 계층일수록 보다 추상화된 정보를 다룰 수 있다. 예컨대 최하위 계층에서는 가공되지 않은 감지데이터(raw sensor data)를 입력하여 곧바로 효과기(effector)를 통한 행동으로 대응시키는 반면, 최상위 계층에서는 기호모델을 바탕으로 에이전트의 궁극적인 목표(goal)를 달성하기 위한 계획을 수립한다. 혼합형 에이전트 구조에서 가장 중요한 문제는 어떤 제어 틀(control framework) 속에 숙고형 및 반응형 모듈들을 두고 이들간의 상호작용을 관리할 것인가 하는 것이다.

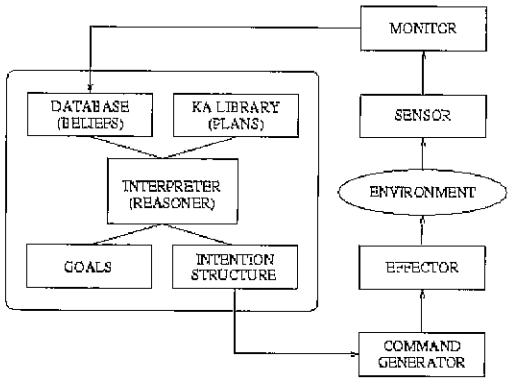

그림 8 PRS 구조

4.1 PRS

1987년 Georgeff와 Lansky에 의해 개발된 PRS(Procedural Reasoning System)[7]은 대표적인 BDI 에이전트 구조이다. 이 구조는 그림 8과 같이 내부에 기호로 명시된 믿음(belief), 소망(desire), 그리고 의도(intention)들의 집합과 계획 라이브러리(plan library)를 포함하고 있다. 믿음들은 외부 세계나 에이전트 내부 상태에 관한 사실(fact)들이다. 이 사실들은 전통적인 일차 서술 논리로 표현된다. 소망은 에이전트의 목표를 나타낸다. PRS의 계획라이브러리는 지식영역(Knowledge Area, KA)라 불리는 계획들의 집합을 포함하고 있다. 각 지식영역은 주어진 목표를 달성하거나 특수한 상황에 반응하기 위해 어떤 일련의 동작과 테스트를 하여야 하는 가를 나타내는 일종의 추상계획이다. 각 지식영역은 자신만의 활성조건(invocation condition), 문맥(context), 효과(effect), 목적(purpose), 몸체(body) 등 포함하고 있으며, 몸체부분에서는 목

적을 달성하기 위해 필요한 절차적 단계들을 기초함수(primitive function)나 부속목표(subgoal), 조건부 가지(conditional branch)들을 이용하여 기술하고 있다. 지식영역들은 활성조건이 만족되면 활성화되며 목표-주도(goal-driven) 방식이나 데이터-주도(data-driven) 방식으로 활성화될 수 있다. 한 에이전트의 현재 활성 중인 지식영역들의 집합은 그 에이전트의 의도(intention)들을 나타낸다. PRS에서 인터프리터(interpreter)는 믿음, 소망과 같은 구성요소들을 바탕으로 적절한 지식영역(계획)들을 선택하여 의도집합에 위치시킨 뒤 이들을 실행하는 역할을 수행한다. PRS는 실행 중인 지식영역들에 대해 명시된 바대로 문맥이 지켜지고 있는 지를 수시로 점검하여 문맥이 만족되지 않으면 곧바로 해당 지식영역의 실행을 중단함으로써 환경변화에 신속히 반응할 수 있다. PRS는 원래 Georgeff 등에 의해 LISP으로 구현되었으나 그 후 미시간 대학교에서 각각 C++와 Java로 재구현 되었다. PRS는 우주 왕복선을 위한 유지보수 절차에 관한 시뮬레이션, 이동 로봇 제어 등 다양한 응용분야에 적용되었다.

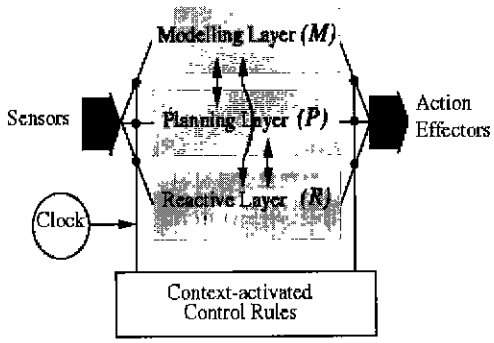

그림 9 TouringMachine 구조

4.2 TouringMachine

1992년 Ferguson은 TouringMachine[6]이란 혼합형 에이전트 구조를 발표하였다. 이 구조는 그림 9와 같이 환경과 직접 맞닿는 인지부(perception)와 행동부(action), 그리고 하나의 제어 틀(control framework)속에 포함되어 있는 세 개의 제어 계층들로 구성되어 있다. 각 제어 계층은 에이전트의 행동을 결정하기 위해 독립적으로 병행 수행되는 하나의 프로세스이며, 제어 틀은 이들 계층들간의 중재 역할을 수행한다. 반응층(reactive layer)에서는 다른 계층들에서 처리하기에는 너무 빠른 사건들에 대해 대응할 수 있는 동작들을 생성한다. 이 계층은 Brooks의 포함구조처럼 상황-행동 규칙(situation-action rule)들의 집합으로 구현되었다. 계획층(planning layer)에서는 목표를 달성하기 위한 계획을 세우고 실행할 동작을 결정한다. 이 계층은 계획기(planner)와 주의집중 메카니즘(focus of attention mechanism)으로 구성되어 있다. 계획기는 계획생성과 실행을 총괄하며, 계획수립을 위해 추상계획들의 라이브러리를 이용한다. 주의집중 메카니즘의 목적은 계획기가 다루어야 하는 정보의 양을 제한함으로써 효율을 높이는데 있다. 그리고 이것은 환경으로부터 직접 관련성이 없는 정보들을 여과해줌으로써 가능하다. 모델링층(modeling layer)에서는 다른 에이전트들의 인지상태를 기호로 표현하고, 이것을 바탕으로 에이전트간의 목표충돌(goal conflict)을 찾아내고 해결하는 기능을 수행한다. 이와 같은 세 계층들은 모두 하나의 제어 틀(control framework) 내에 포함되어 있으면서 메세지 교환을 통해 서로 교신할 수 있다. 이러한 제어 틀의 목적은 계층간의 중재 역할을 하는 것이며 특히 제어 규칙(control rule)들을 이용하여 서로 다른 계층들에서 제안하는 동작들간의 충돌을 처리하는 것이다.

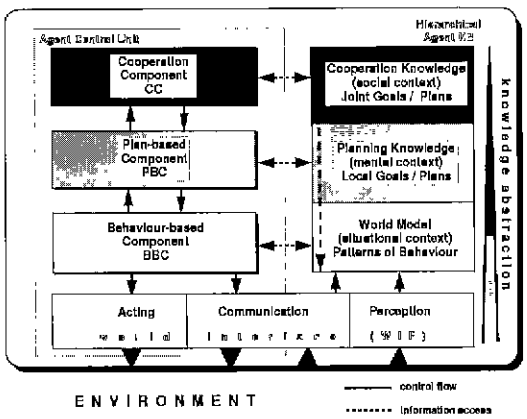

그림 10 InteRRaP 구조

4.3 InteRRaP

Muller에 의해 제안된 InteRRaP[11]은 앞서 살펴본 TouringMachine과 마찬가지로 하나의 계층구조이며, 연속적으로 놓여진 각 계층은 하위 계층보다 더 높은 추상화를 나타낸다. 하지만 InteRRaP에서는 그림 10과 같이 이들 각 계층들에 대해 다시 두 개의 수직 계층(vertical layer)들로 나누어, 그 중 하나는 지식베이스(knowledge base)들을 포함하는 층으로, 다른 하나는 그 레벨의 지식베이스와 상호 작용하는 다양한 제어 구성요소를 포함하는 층으로 두었다. InteRRaP에서 최하위 계층에는 실세계 인터페이스 제어부(world interface control componenet)와 그것에 대응되는 실세계 모델 지식베이스((world model knowledge base)가 놓인다. 실세계 인터페이스부는 에이전트와 환경간의 인터페이스를 담당하며, 따라서 동작 실행(acting)과 통신(communicating), 그리고 인식(perception) 등을 다룬다. 실세계 인터페이스부 바로 위에는 행위-기반 제어부(bahavior-based component)가 위치하며, 이 계층의 역할은 에이전트의 기본적인 반응 능력을 구현하고 제어하는 것이다. 이 계층에서는 다수의 행위패턴(behavior pattern)들을 다루는데, 하나의 행위패턴은 활성화되는 시기를 나타내는 전조건(pre-condition)과 그 행위패턴이 성공했거나 실패했던 환경여건을 나타내는 다양한 조건들, 그리고 하나의 후조건(post-condition)과 수행 가능한 본체(body)로 구성된다. 행위-기반 제어부 위에는 계획-기반 제어부(plan-based component)가 놓인다. 이 계층에는 행위-기반 제어부로부터 요청이 있을 때 단일 에이전트 계획(single- agent plan)들을 생성할 수 있는 하나의 계획기(planner)를 포함하고 있다. 이 계층의 지식베이스는 미리 주어진 계획라이브러리를 포함하여 다수의 계획들을 가지고 있다. InteRRaP에서 최상위 계층은 협동 제어부(cooperation component)이다. 이 계층에서는 하위 계층인 계획-기반 제어부의 요청이 있을 때 환경내의 다른 에이전트들을 고려하여 다수 에이전트들의 목표를 모두 만족시킬 수 있는 협동 계획(joint plan)을 생성한다. InteRRaP에서 제어는 데이터-주도 방식과 목표-주도 방식을 모두 가지고 있다. 일반적으로 실세계로부터 입력되는 인식데이터는 실세계 인터페이스부에 의해 처리되며 실세계 모델의 변화를 가져온다. 실세계 모델의 변화는 다양한 행위 패턴들을 활성화하거나 실행할 수 있다. 그리고 이러한 행위패턴들의 실행은 에이전트의 궁극적인 목표 달성을 위해 다시 상위 계층들인 행위-기반 제어부와 협동 제어부로 하여금 단일 에이전트 계획 또는 협동계획을 생성하도록 요청할 수 있다. 이와 같은 과정을 거쳐 최종적으로 다시 실세계 인터페이스부에 의해 기본 동작들을 실행하거나 메시지들을 전송하게 된다.

5. 결 론

지금까지 숙고형, 반응형, 그리고 혼합형 에이전트 구조의 특징과 대표적인 사례들을 간략히 살펴보았다. 최근 들어 많은 에이전트 연구자들 사이에 전통적인 숙고형 구조에 비해 환경변화에 빠른 반응을 보일 수 있는 반응형 구조나 혼합형 구조에 대한 관심이 날로 증가하고 있다. 하지만 이들 구조가 가지고 있는 중요한 문제점으로는 명확한 개발 방법론(development methodology)과 이론적 모델(theoretical model)이 아직은 부족하다는 것이다. 현재까지 구현된 많은 반응형 에이전트시스템들은 일정한 개발 방법론에 의존하지 못하고 개발자의 경험과 오랜 실험을 통해 개발되었다. 이러한 접근방식으로는 보다 복잡도가 높은 시스템들의 개발은 현실적으로 어려울 것이다. 그뿐만 아니라 현재 대부분의 반응형 구조와 혼합형 구조는 숙고형 구조에 비해 투명한 이론적 배경을 갖고 있지 못하다. 아무리 실세계에서 잘 동작하는 에이전트 구조라 하더라도 그 구조에 대한 이론적 모델을 갖지 못한다면 우리는 왜 그 시스템이 제대로 동작하는지 실제로 이해하지 못할 것이다. 따라서 향후 이들 반응형 구조와 혼합형 구조에 대한 이론적 연구와 개발 방법론에 관한 연구가 절실히 요구된다.

참고문헌

[1] Ambros-Ingerson, J. and Steel, S. Integrating planning, execution and

monitoring. In Proceedings of AAAI-88, pages 83-88, 1988.

[2] Bratman, M. E., Israel, D. J., and Pollack, M. E. Plans and resource-bounded practical reasoning. Computational Intelligence, 4:349--355. 1988.

[3] Brooks, R. A. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, 2(1):14-23. 1986.

[4] Chapman, D. Planning for conjunctive goals. Artificial Intelligence, 32:333-378. 1987.

[5] Etzioni, O., Lesh, N., and Segal, R. Building softbots for UNIX. In Etzioni, O., editor, Software Agents-Papers from the 1994 Spring Symposium, pages9-16. AAAI Press. 1994.

[6] Ferguson, I. A. TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents. PhD thesis, Clare Hall, University of Cambridge, UK. 1992.

[7] Georgeff, M. P. and Lansky, A. L. Reactive reasoning and planning. In Proceedings of AAAI-97, pages 677-682, 1987

[8] Jennings, N. R. Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving. Journal of Intelligent and Cooperative Information Systems. 2(3): 289-318. 1993.

[9] Kaelbling, L. P. A situated automata approach to the design of embedded agents. SIGART Bulletin, 2(4):85-88. 1991.

[10] Maes, P. The agent network architecture. SIGART Bulletin. 2(4):115-120. 1991.

[11] Muller, J. P., Pischel, M., and Thiel, M. A pragmatic approach to modelling autonomous interacting systems. In Wooldridge, M. and Jennings, N. R., editors. Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages, pages 226-240, Amsterdam, The Netherlands. 1994.

[12] Rao, A. S. and Georgeff, M. P. Modeling rational agents within a BDI-architecture. In Fikes, R and Sandewall, E., editors. Proceedings of KR&R-91, pages 473-484. 1991.

김 인 철

1985 서울대학교 수학과(학사)
1987 서울대학교 전산과학과(석사)
1995 서울대학교 전산과학과(박사)
1989 ~ 1995 경남대학교 전산통계학과 조교수
1996 ~ 현재 경기대학교 정보과학부 부교수
관심분야:지능형 에이전트, 계획시스템, 분산 인공지능
E mail:kic@kuic.kyonggi.ac.kr

● JCCI 2000 ●

- 일 자 : 2000년 5월 25 ~ 27일
- 장 소 : 경주
- 주 최 : 정보통신연구회
- 논문제출 양식 및 설문지 양식 : 홈페이지 참조
  홈페이지: http://ccl.cnu.ac.kr/jcci/2000
- 문 의 처 : 서강대학교 컴퓨터학과 최명화 교수
  Tel. 02-705-8495