한국 퍼지 및 지능시스템학회 논문지 2000, Vol. 10, No. 6 (2000)548~556

BDI 에이전트 구조에서 충돌 해결을 위한 논리기반 협상 기법의 연구

A Study on Logic-based Negotiation Mechanism for Conflict Resolution in BDI Agents

이명진 · 김진상

Myung Jin Lee and Jin Sang Kim

계명대학교 컴퓨터공학과

요 약

멀티에이전트 시스템(MAS: Multi-Agent System)에서 에이전트는 각자의 목표 달성을 위해 주위 에이전트들과의 상호작용을 통해 목표의 충돌이 없는 일치 상황에 도달하도록 설계되어야 한다. 멀티에이전트 시스템에서 에이전트들 사이의 목표 충돌은 일반적으로 발생 가능한 상황이고, 어떤 에이전트가 다른 에이전트에 관한 모든 지식을 가진다는 것은 불가능하기 때문에 상대방에 관한 부분적인 지식만을 가진 상황에서 목표 충돌을 해결할 수 있는 협상은 중요하다. 본 논문은 멀티에이전트 시스템에서 믿음(Belief), 소망(Desire), 그리고 의도(Intention)를 에이전트 구조의 핵심 요소로 가정하고 이러한 구조를 가지는 BDI 에이전트를 논리 프로그래밍의 입장에서 표현한다. 또한 서로 다른 목표를 가진 BDI 에이전트들이 서로 협상하여 문제를 해결하는 과정에서 발생하는 에이전트들 상호간의 목표 충돌을 해결하는 방법을 제시하며, 이 방법의 효과성을 검증하기 위하여 JAVA와 PROLOG를 결합시킨 InterPROLOG 프로그래밍 언어로 구현하여 시험한다.

ABSTRACT

It is necessary that each agent in most multi-agent systems must be designed to reach an agreed state where the agent can avoid any goal conflicts stem from the requirement to resolve the tasks posed by its interdependence upon other agents. Many problematic situations, however, may arise such that an agent can meet goal conflicts among agents and cannot possesses a complete set of knowledge about others. Negotiation provides a solution to these problems even though agents' knowledge is incomplete. We present a logic programming framework in which agents are represented on the basis of three major components: their beliefs, desires, and intentions. Further, we suggest a negotiation mechanism to resolve goal conflicts accompanied with each agent's problem solving activities in a cooperative multi-agent system. To show the effectiveness of the mechanism, we implement and test a negotiation system in InterProlog that combines both Java and Prolog.

1. 서 론

멀티에이전트 시스템(MAS)은 주어진 문제를 해결하는 자율적인 에이전트들의 행위에 초점을 두고 있으며, 각 에이전트의 능력을 초월하는 문제를 해결하기 위해 서로 협동하는 문제 해결자(problem solver)들의 네트워크로 정의할 수 있다[8]. 에이전트라 부르는 이러한 문제 해결자는 자율성을 가져야 하는데, 여기서 말하는 자율의 개념은 에이전트가 스스로 어떤 목표를 선택할지 그리고 선택한 목표를 어떻게 달성해야 할지를 결정할 능력을 갖고 있음을 의미한다. 에이전트들 사이에 존재하는 고유한 상호 의존성 때문에 자율적인 에이전트들은 서로 상호작용 할 필요성이 있다. 이러한 상호작용을 다루는 기법 중에서 에이전트들이 어떤 문제 혹은 관점에 대해 충돌이 발생했을 때 서로 통신을 통해 상호 수용 가능한 일치 상황에 도달하는 과정을 협상이라 볼 수 있다. 협상은 분산 인공지능에서 강조되는 분야 중의 하나로써, 일반적인 목적은 각 에이전트의 계획의 수정과 상호작용이 가능한 상황의 확인에 있다. 특히 협상은 작업과 자원의 할당, 충돌의 인식, 전자상거래 시스템에서의 가격절충, 목표 불일치에 대한 해결, 그리고 에이전트들 상호간의 일관성 유지를 위해 사용되고 있다.

Davis와 Smith가 연구한 CNP(Contract Net Protocol)에서 협상은 작업과 문제 해결자를 효과적으로 연결하기 위해 사용하는 원리의 구성으로 간주하였다[4]. 그러나 CNP에서 협상을 수행하기 위해서는 에이전트들 사이에 전혀 목표의 충돌이 없어야 한다. Bussmann과 Muller는 협동적인 상황에서 에이전트들

이 사용할 수 있는 협상 모델과 알고리즘을 개발하기 위해 사회 심리학을 이용하였다[1]. 이 모델은 전통적인 MAS 모델보다는 발전하였지만 엄격한 이론적인 요소가 부족하고 에이전트들이 반드시 협동적이어야 한다는 한계점을 갖고 있다.

Maes, Guttman, 그리고 Moukas는 전자상거래에서 소비자의 구매 행위 모델에 기초하여 중개자로서의 에이전트의 역할에 관해 연구하였는데[7], 여기서는 협상을 트랜잭션의 조건과 가격을 결정하는 단계로 간주하였다. Parson, Sierra, 그리고 Jennings는 협박, 보상, 그리고 호소를 사용한 argumentation 시스템을 통해 설득력이 있는 협상을 수행할 수 있는 형식적인 프레임워크를 제안하였으며[11], 이 시스템은 전자상거래에서의 협상을 위한 Tete-a-Tete 프로젝트의 기초가 되었다[15]. Rosenshein과 Zlotkin은 게임 이론에 기초한 협상에 관해 연구하였는데[13], 에이전트들이 payoff matrix에 관한 완전한 지식을 가진다고 가정하였기 때문에 에이전트들은 다른 에이전트들의 기호에 관한 완전한 지식을 가진다는 강력한 가정을 필요로 한다.

그러나 MAS에서 에이전트들 사이의 목표 충돌은 불가피한 것이고, 에이전트가 다른 에이전트에 관한 완전한 지식을 가진다는 것 역시 불가능하다. 따라서 상대방에 관한 일부분의 지식을 갖고 있는 상황에서 목표가 충돌할 경우 협상이 어떤 과정으로 이루어질 수 있는가에 관한 연구는 매우 중요하다.

일반적으로 에이전트는 수행할 작업 혹은 달성할 목표를 갖고 있으며, 이러한 작업을 어떻게 수행할 것인가에 대해 결정을 내릴 필요가 있다. 이와 같은 요구를 만족시키는 에이전트 구조 중의 하나는 BDI (Belief-Desire-Intention) 접근방법이다. BDI 에이전트 구조는 에이전트의 믿음, 소망, 의도, 그리고 계획 라이브러리 등과 같은 네 가지의 자료구조를 가진다. 이 중에서 에이전트의 믿음은 에이전트가 세상에 관해 알고있는 정보에 해당하는데, 이러한 믿음은 본질적으로 불완전하고 부정확할 수 있다[14].

본 논문에서는 다른 에이전트에 관한 불완전한 지식을 가지는 BDI 에이전트 구조를 사용하여 MAS에서의 협상 과정을 기술한다. 본 연구에서는 BDI 에이전트들이 표현력이 높은 통신언어를 사용하여 서로의 믿음과 의도를 변화시키면서 협상을 수행할 수 있게 하는 BDI 에이전트를 위한 협상 기법을 제시한다. 제시된 기법의 효과성을 시험하기 위해 목표 달성 과정에서 충돌을 일으키는 BDI 에이전트들을 구성하고, 이들이 협상을 통해 목표 충돌을 해결하는 과정을 시험적으로 구현하였다. 구현에 사용한 언어는 PROLOG(XSB)와 JAVA를 결합시킨 InterPROLOG 이다.

본 논문의 구성은 다음과 같다: 2절에서는 협상을 위한 여러 프로토콜들과 여기서 사용하는 BDI 협상 프로토콜을 소개한다. 3절에서는 BDI 에이전트의 협상 모델과 관련된 에이전트의 구조, 통신언어, 협상 등을 소개하고, 4절에서는 협상 모델을 기초로 BDI 에이전트의 협상 과정을 예제를 통해 살펴본다. 5절에서는 연구 결과에 대해 논의하고, 마지막으로 6절에서는 BDI 에이전트의 협상에 관해 결론을 내리고 향후 연구과제에 대해 논의한다.

2. 협상을 위한 프로토콜들

협상 프로토콜은 협상 사이클을 시작하고 메시지에 응답할 가능성을 서술한다. 간단한 방법은 협상 프로토콜을 아래와 같이 주어진 일련의 협상 행위들로 정의하는 것이다[9].

(< *negotiation primitive* >, < *message contents* >)

여기서 *negotiation primitive*는 proposal, give, inform, accept, alternative 등과 같이 에이전트의 의도를 나타내며 *message contents*는 구체적인 전송 내용을 나타낸다. 다음으로 협상에 관해 지금까지 소개된 몇 가지 프로토콜들에 대해 간단히 살펴본다.

2.1 CNP(Contract Net Protocol)

Davis와 Smith의 CNP에서 에이전트들은 특정한 목표를 달성하기 위해 계약을 통해 그들의 활동을 조절한다[4]. 매니저 역할을 하는 에이전트는 자신의 계약(할당된 작업 혹은 문제)을 다른 가능성이 있는 계약자(에이전트)들에 의해 달성될 수 있는 작은 계약들로 나눈다. 매니저는 각각의 작은 작업들을 에이전트들의 네트워크에 알리게 되며 통보를 받은 에이전트들은 그 작업을 평가하게 된다. 적절한 자원, 지식, 그리고 정보를 가진 에이전트들은 통보된 작업을 달성할 수 있는 능력의 표시로 입찰을 하게 된다. 매니저는 수신된 입찰들을 평가하고 가장 적당한 에이전트(계약자)에게 그 작업을 맡기게 된다. 마지막으로 매니저와 계약자는 그 작업이 달성될 때까지 서로 정보를 교환한다. 그러나 CNP를 시작하기 위해서는 에이전트들 사이에 전혀 목표의 충돌이 없어야 된다. 계약자는 자신의 최소한의 조건을 전송하지 않고, 상호 일치는 궁극적으로 매니저의 결정에 의해 이루어진다. CNP는 협상 원리라기 보다는 오히려 표준화된 공조(coordination) 방법으로 간주할 수 있다.

2.2 Argumentation 시스템

Parsons, Sierra, Jennings가 제안한 argumentation 시스템은 협상 중인 에이전트들의 추론 과정을 서술하는 프레임워크를 사용한다[11]. 어떤 에이전트가 초기 제안을 전송하면 수신 에이전트는 전송된 제안에 찬성하는 argument 그리고 반대하는 argument를 만들어 그 제안을 평가한다. 만약 그 제안을 수용할 수 없다면, 초기 제안에 반대되는 argument나 새로운 대안을 만들어 원래 에이전트에게 전송한다. Argumentation 시스템은 BDI modality와 ACR(Argumentation Consequence Relation)을 사용한 시스템으로 예를 들어, ACR들 중의 하나인

$$\rightarrow\text{-}E\frac{\Delta\vdash_{ACR}(p,B)\quad \Delta\vdash_{ACR}(p\rightarrow q)}{\Delta\vdash_{ACR}(q,A\cup B)}$$

는 p와 $p\rightarrow q$에 대한 argument들로부터 q에 대한 argument를 만드는 것이다. 여기서는 지식베이스, A와 B는 각각 사실 p와 규칙 $p\rightarrow q$를 표시하는 레이블이다.

2.3 충돌 해결을 위한 협동 기법

Cammarate는 에이전트들의 그룹 계획 사이에서의 충돌 해결을 위한 협동 기법에 대해 연구하고 작업 중심(task centralization)이라 불리는 정책을 사용하였다[2]. 잠재적인 충돌 상황에 속해있는 에이전트들은 그 충돌을 해결하기 위해 충돌 상황 내에 포함되어 있는 에이전트들 중에 하나를 선택한다. 선택된 에이전트는 포함된 모든 계획들을 서술하는 멀티 에이전트 계획을 개발하는 계획자로서 작용한다. 에이전트들은 그들 중 어느 것이 가장 계획(planning)을 잘 할 수 있는가를 결정하기 위해 협상한다.

2.4 순환적인 협상 모델

Laasri는 정교한 협동적인 분산처리시스템(CDPS: Cooperative Distributed Processing System)을 구성할 때 다수의 전문가들, 관점, 혹은 추론 형태들 사이에 충돌 해결이 어디에서 필요한지를 분류하고 서술하는 순환적인 협상 모델(recursive negotiation model)을 제안했다[6]. 이 모델은 언제 그리고 어떻게 협상이 적용될 수 있는지를 정의하고 문제 해결을 네 가지 단계로 구성하였다: 문제 형식화, 초점의 대상, 에이전트에게 목표 혹은 작업의 할당, 목표 혹은 작업의 달성. 이 모델을 통해 Laasri는 협상이 영역 수준 그리고 제어 수준 문제 해결 모두에서 충돌을 해결하기 위해 사용되는 순환적이고 복잡한 과정일 수 있다고 강조하였다.

2.5 BDI 에이전트의 협상 프로토콜

본 논문에서는 BDI modality와 명확한 수용거부 조건을 가지는 시스템을 이용하고, 지식베이스와 규칙을 분리하고, 소켓을 통신수단으로 하는 멀티에이전트 모델을 사용한다. 수용거부의 조건으로 자신의 목표 달성을 위해 가지고 있어야 할 것을 상대방 에이전트가 요청한 경우와 아래와 같은 반대의 개념을 정의한다:

정의 **1.** s와 s'을 각각

$$M^1(p_1)\wedge\cdots\wedge M^n(p_n)\text{와 } M^{1'}(q_1)\wedge\cdots\ M^{m'}(q_m)$$

형태의 문장들이라 하고, M^i와 $M^{j'}(1\le i\le n,\ 1\le j'\le m)$을 믿음, 소망, 혹은 의도를 표현하는 modality라고 할 때 어떤 p_i와 q_j에 대해 $p_i=\neg q_j$이면 s는 s'을 반대한다.

마지막으로 본 논문에서 사용하는 협상 프로토콜은 그림 1과 같으며, 이 협상 프로토콜을 요약하면 다음과 같다: 어떤 에이전트가 초기 제안을 하는 경우 협상이 시작되며, 다른 에이전트는 수신된 제안을 수용하거나, 거부하거나, 혹은 역제안을 하게 된다. 이것에 따라 에이전트는 다시 제안을 수용하거나, 거부하거나, 새로운 제안을 만들거나, 혹은 전송한 에이전트의 목표를 해결할 수 있는 다른 방법 등을 보내게 된다. 이러한 과정은 제안 또는 역제안이 관련된 모든 에이전트들에게 수용 가능하게 될 때까지 혹은 일치에 이르지 못하고 협상이 결렬될 때까지 계속된다. 위와 같은 협상은 규칙과 지식베이스를 분리하여 사용함으로써 다른 환경의 협상에서도 쉽게 구현할 수 있을 것이다.

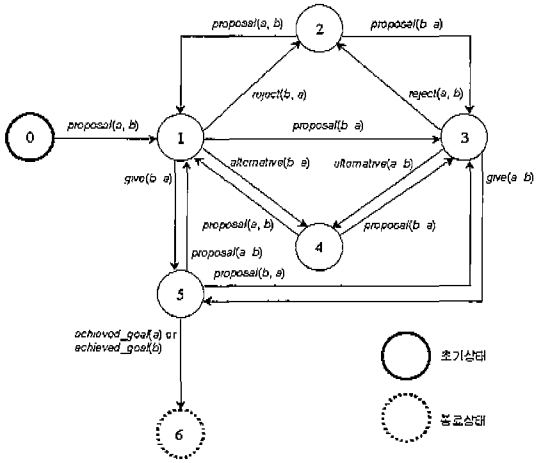

그림 1. BDI 에이전트의 협상 프로토콜

3. BDI 에이전트의 협상 모델

3.1 BDI 에이전트의 구조

MAS에서 에이전트는 일련의 지식들의 집합을 필요로 한다. 이러한 지식은 에이전트 자신의 믿음 혹은 능력에 관한 지식, 상호작용 할 수 있는 다른 에이전트에 관한 지식, 다른 에이전트와 어떻게 통신할 수 있는가에 관한 지식, 그리고 특정한 응용 영역에 관한 지식을 포함한다. 또한 에이전트는 여러 에이전트들과 상호작용해야 하기 때문에 다중 상호작용을 다루는 능력을 가져야 하며, 네트워크를 통해 분산되어 있는 에이전트들과 통신할 수 있는 능력을 가져야 한다. 따라서 에이전트 구현 언어는 지식을 표현할 수 있고, 추론 기법을 가지고, 다중 작업이 가능하고, 동기 혹은 비동기 통신이 가능하고, 그리고 네트워크 상에서 통신할 수 있어야 한다.

본 논문에서 사용하는 BDI 에이전트 구조는 아래와 같은 네 가지의 요소들을 포함하고 있다:

- 지식베이스 - 에이전트 자신의 믿음, 소망, 의도 혹은 능력, 다른 에이전트들에 관한 믿음 혹은 능력, 그리고 문제를 분해하는 규칙들을 가지고 있다.
- 계획 - 각각의 작업을 어떻게 해결해야 할지를 결정한다.
- 모니터 - 새로운 작업을 받아들이고 결과를 보고한다.
- 통신 - 전송하는/수신하는 메시지를 조작한다.

BDI 에이전트 구조에서 실제로 협상이 어떻게 이루어지는가를 보이기 위해서 다른 목표들과 다른 자원들을 가진 집안 일을 도와주는 두 에이전트들을 가정한다. 에이전트 *a1*은 그림을 걸려고 하고 그림, 나사못, 스크루드라이버, 그리고 망치를 갖고 있다. 에

```
b(my_name(a1))
b(have(a1, picture)).
b(have(a1, screw)).
b(have(a1, hammer))
b(have(a1, screwdriver)).
b(a2, b(have(a2 nail))).
it(do(a1, hang_picture)).
rule(b(can(W, hang_picture)),
        [b(have(X, hammer)), b(have(Y, nail)), b(have(Z, picture))])
rule(b(holdon(W, hammer)),
        [b(have(X, hammer)), b(have(Y, nail)), b(have(Z, picture))])
rule(b(can(W, hang_mirror)),
        [b(have(X, screwdriver)), b(have(Y, screw)), b(have(Z, mirror))]).
rule(b(do(X, Y))
        [it(do(X, Y)), b(can(X, Y))])
```

그림 2. 에이전트 *a1*의 지식베이스

```
b(my_name(a2))
b(have(a2 mirror))
b(have(a2, nail)).
b(a1, b(have(a1, hammer)))
b(a1, b(have(a1, screwdriver))).
b(a1, b(have(a1, screw))).
it(do(a2, hang_mirror)).
rule(b(can(W, hang_mirror)),
        [b(have(X, hammer)), b(have(Y, nail)), b(have(Z, mirror))])
rule(b(holdon(W, nail)),
        [b(have(X, hammer)), b(have(Y nail)), b(have(Z, mirror))]).
rule(a(do(X, Y)),
        [it(do(X, Y)), b(can(X, Y))]).
```

그림 3. 에이전트 *a2*의 지식베이스

이전트 *a1*은 에이전트 *a2*가 못을 갖고 있다고 믿고 있고, 그림을 거는 방법과 거울을 거는 방법을 알고 있다. 이와 같은 에이전트 *a1*의 지식을 논리 프로그래밍으로 표현하면 그림 2와 같다.

한편 에이전트 *a2*는 거울을 걸려고 하고, 거울과 못을 갖고 있고, 그리고 못을 사용하여 거울을 거는 방법을 알고 있다고 가정한다. 에이전트 *a2*의 지식을 논리 프로그래밍으로 표현하면 그림 3과 같다.

3.2 BDI 에이전트의 통신언어

일반적으로 에이전트들의 협상은 메시지의 교환을 통해 달성되며, 메시지의 실제적인 교환은 참여하는 에이전트들의 개인적인 필요와 목표에 의해 유도된다. 협상은 에이전트들 사이의 통신을 필요로 하고, 애매함을 없애기 위해서 각각의 에이전트는 고유한 식별자를 가져야 한다. 협상에 관련된 에이전트들의 식별자들의 집합을 Agents로 표시한다. BDI 에이전트들이 부족한 자원들에 관해 협상한다고 가정하면, 에이전트들은 자신의 목표를 달성하기 위해서 자신에게 필요한 부족한 자원의 할당을 요구한다. 목표들의 집합을 Goals로, 자원들의 집합을 Resources로 표시한다. 이러한 경우에 에이전트들이 사용하는 통신언어를 아래와 같이 간단히 정의할 수 있다:

정의 **2.** $a1, a2 \in Agents$, $g \in Goals$, 그리고 $r \in Resources$라 할 때 통신언어 CL을 다음과 같이 정의한다:

1) $proposal(a1, a2, g, r) \in CL$
2) $give(a1, a2, r) \in CL$
3) $reject(a1, a2, g, r) \in CL$
4) $alternative(a1, a2, subgoals) \in CL$
5) $achieved\_goal(a1, a2) \in CL$

여기서, $proposal(a1, a2, g, r)$은 "에이전트 $a1$이 목

표 *g*를 달성하기 위해서 에이전트 *a2*에게 부족한 자원 *r*을 제안한다", *give*(*a1*, *a2*, *r*)은 "에이전트 *a1*이 에이전트 *a2*에게 자원 *r*을 제공한다", *reject*(*a1*, *a2*, *g*, *r*)는 "에이전트 *a1*이 목표 *g*를 달성하기 위해 *a2*가 제안한 자원 *r*을 거절한다", *alternative*(*a1*, *a2*, *subgoals*)는 "에이전트 *a1*이 에이전트 *a2*에게 목표를 달성할 수 있는 다른 방법을 제공한다", 그리고 *achieved_goal*(*a1*, *a2*)는 "에이전트 *a1*이 자신의 목표를 달성한 후 에이전트 *a2*에게 이를 알린다"라고 해석할 수 있다. 그러나 위와 같은 통신언어는 특정 영역에 따라 여러 형태들로 확장할 수 있을 것이다.

3.3 BDI 에이전트의 협상

본 논문에서 사용하는 협상 기법의 특징은 BDI 에이전트 구조, 재계획, 소켓을 이용한 통신의 형식화란 점이다. 에이전트의 이론을 형식화할 때 요구되는 공리들은 일차논리의 부분집합인 Horn 절들의 집합으로 나타내며, 추론은 SLDNF를 이용한 연역 추론으로 이루어진다. 새로운 지식의 추가 혹은 기존 지식의 변경이 발생하는 경우에 에이전트는 이전의 목표 달성을 멈추고 새롭게 변경된 지식베이스를 기초로 재계획하여 다시 목표를 달성하여야 한다. 본 논문에서 사용된 예제는 서로 다른 목표를 추구하는 두 명의 에이전트들을 가정하고 수용거부가 발생하는 경우 충돌이 생긴 것으로 간주한다. 먼저 아래에서 협상을 위한 에이전트의 구조 각각에 대한 역할을 살펴본다.

3.3.1 에이전트의 계획

에이전트의 계획 부분은 문제 혹은 작업을 어떻게 해결해야 할지를 결정한다. 계획 부분은 모니터 부분으로부터 작업을 받거나 문제 분해의 결과로써 내부적으로 생기는 작업을 받는다. 먼저 계획 부분은 자신에게 주어진 작업을 완성하기 위해 자기 자신의 지식베이스에 질의하여 자신의 능력으로 해결하려 한다. 만약 그렇게 하지 못하는 경우에는 그 작업을 수행할 수 있는 다른 에이전트를 찾아서 그 에이전트에게 적당한 메시지를 보내고 통신 부분을 통해 응답을 기다린다. 전송하는 메시지의 형태는 *negotiation_primitive*(*message_contents*)이며, 예를 들어 *proposal*(*a1*, *a2*, *g*, *r*)일 때 *g*는 현재 에이전트가 달성하려는 목표를 나타내며 *r*은 목표 *g*를 달성하는데 필요한 상대방 에이전트의 자원이다. 만약 자신뿐만 아니라 다른 에이전트가 그 작업을 해결할 수 없다면 계획 부분은 작업을 분해하여 더 작은 작업을 수행하려고 시도한다. 이와 같은 에이전트의 계획은 그림 4와 같이 나타낼 수 있다.

3.3.2 에이전트의 모니터

에이전트의 모니터 부분은 작업 실행의 감시 그리고 메시지를 보낸 에이전트에게 결과를 보내주는 역할을 한다. 초기에 제안 혹은 요청은 통신 부분을 통해 외부 에이전트로부터 전송되어 오거나 내부적으로 발생하며 이런 제안은 계획 부분으로 전달된다. 구체적으로 *proposal*(*a1*, *a2*, *g*, *r*) 형태의 메시지를 받으면 먼저 자원 *r*이 수용거부되는가를 검사한다. 수용거부가 발생하면 목표 *g*를 달성할 수 있는 대안을 찾아 대안이 있다면 그 대안을 전송하고 응답을 기다린다. 만약 그렇지 않다면 자신의 목표를 달성하려 하는데 이 경우에는 자신의 목표를 계획 부분으로 보내게 되고 역제안이 전송된다. 수용거부가 발생하지 않는다면 상대방 에이전트의 제안을 수락하는 경우이며, 이 때는 자신의 믿음과 상대방 에이전트에 관한 믿음을 수정하고 요청한 자원을 넘겨주고 응답을 기다린다.

한편 *alternative*(*a1*, *a2*, *subgoals*) 형태의 메시지를 받을 때는 자신의 목표를 달성할 수 있는 다른 방법이 전송된 경우이다. 이 때는 지금까지 자신이 알고 있는 목표 달성 방법을 제거하고 새로운 목표 달성 방법을 지식베이스에 추가시킨다. 이상의 모니터 과정은 그림 5와 같다.

3.3.3 에이전트의 통신

에이전트의 통신 부분은 통신을 다루기 위해 소켓을 이용하여 다른 에이전트에게 메시지를 전송하고 수신된 메시지를 모니터 부분으로 보내는 역할을 한다. 메시지를 전송하는 경우 그리고 응답하는 경우 모두

```
plan(Task) :-
        Task
plan(Task) :-
        find_agent(Task, Receiver, Tool),
        find_goal(Task, Goal), b(my_name(Sender)),
        send_message('proposal', Sender, Receiver, Goal, Tool), ..
plan(Task) :-
        rule(Task, Subtasks), execute1(Subtasks).
```

그림 4. 에이전트의 계획

```
proposal(Sender, Goal, Tool) :-
        defeat(Tool), search_alternative(Goal, Subgoals), ...
proposal(Sender, Goal, Tool) :-
        defeat(Tool), it(do(X, Y)), call(solve(b(do(X, Y)))), !, ...
proposal(Sender, Goal, Tool) :-
        not(defeat(Tool)), .
solve(Task) :-
        plan(Task), !.
solve(Task) :-
        write('Cannot do '), writeln(Task)
```

그림 5. 에이전트의 모니터

```
send_message(Type, Sender, Receiver, Goal, Tool) :-
        ..., socket_send(Sock, Message, _).
wait_reply(Sock) :-
        ...,
        socket_recv(Sock, Message, _), ...,
        proposal(Sender, Goal, Tool), ...,
        alternative(Sender, Receiver, Goal, Tool), ...,
        give(Sender, '_', Tool), ...,
        achieved_goal(Sender, '_', '_'), ...,
        write('Message type mismatched: ').
```

그림 6. 에이전트의 통신

수신 에이전트의 이름을 알아낸 후 해당 에이전트에게 메시지 혹은 응답을 전송하는데 이를 논리 프로그래밍으로 표현하면 그림 6과 같다.

3.3.4 메시지의 생성

수신한 메시지를 어떻게 해석하고 송신할 메시지를 어떻게 만들어낼 것인가를 결정하기 위해서 에이전트가 사용하는 수단이 수용거부 조건이다. 에이전트는 현재의 지식베이스로부터 수용거부되지 않는 메시지를 받은 경우에 그 메시지를 해석하여 필요한 지식을 변경하여야 한다. 메시지의 해석은 지식베이스의 수정을, 메시지의 생성은 특별한 상황에서 취할 메시지 행위를 결정한다. 메시지의 실제적인 결과는 에이전트의 해석에 의존하며 이러한 해석 과정은 주로 특정 영역에 의존하고 또한 에이전트의 특정한 내부 구조에 의존하게 된다.

정의 **3.** 메시지의 해석과 생성: 에이전트 $a1$으로부터 메시지를 받은 에이전트 $a2$를 가정한다. 통신언어가 주어지면 에이전트 $a2$를 위한 메시지의 해석과 생성을 다음과 같이 정의할 수 있다:

1) 수용거부 조건을 만족하고, 목표를 달성할 대안이 있는 경우

$G(proposal(a1, goal, tool)) = alternative(a1, goal, subgoals)$

2) 수용거부 조건을 만족하고, 목표를 달성할 대안이 없는 경우

$G(proposal(a1, goal, tool)) = reject(a2, a1, goal', tool)$ or *make counterproposal*

3) 수용거부 조건을 만족하지 않는 경우

$G(proposal(a1, goal, tool)) = give(a1, tool)$

4) 목표 달성을 위한 대안을 받은 경우

$G(alternative(a1, goal, subgoals)) = make\ replanning$

5) 자원을 받은 경우

$G(give(a1, tool)) = continue\ planning$

6) 제안이 거부된 경우

$G(reject(a1, a2, goal, tool)) = search\ alternative$

7) 다른 에이전트의 목표 달성을 통보받은 경우

$G(achieved\_goal(a1)) = make\ replanning$

추가적으로 위의 정의에서 3), 4), 5), 7)의 경우 에이전트 $a2$는 자신의 지식베이스를 변경하여야 한다. 예를 들어, 3)의 경우 $a2$가 $give(a2, nail)$과 같은 메시지를 전송한다면 $a2$는 다음과 같이 자신의 지식베이스를 변경하여야 하고, 메시지를 받은 $a1$ 역시 자신의 지식베이스를 변경하여야 한다:

$a2$의 지식의 일부(변경 전)	$a2$의 지식의 일부 (변경 후)
$b(have(a2, nail))$	*retract* $b(have(a2, nail))$ *assert* $b(a1, b(have(a1, nail)))$

$a1$의 지식의 일부(변경 전)	$a1$의 지식의 일부 (변경 후)
$b(a2, b(have(a2, nail)))$	*retract* $b(a2, b(have(a2, nail)))$ *assert* $b(have(a1, nail))$

위와 같은 메시지의 해석과 생성을 포함하는 BDI 에이전트 구조에서 실제로 협상이 어떻게 진행되는가를 다음 절에서 예제를 통해 살펴본다.

4. BDI 에이전트의 협상 과정

에이전트들의 지식베이스에서 알 수 있듯이 에이전트 $a1$의 의도는

$it(do(a1, hang\_picture))$

이고, 에이전트 $a2$의 의도는

$it(do(a2, hang\_mirror))$

이다. 예제에서는 $a1$이 초기 제안을 하는 것으로 가정하지만 $a2$가 초기 제안을 하는 경우도 같은 결과에 이르게 된다. 그래서 $a1$은 자신의 의도를 달성하기 위해 질의어

$?\text{-}solve(b(do(a1, hang\_picture)))$.

를 해결하려 한다. 여기서 목표인 $b(do(a1, hang_picture))$는 에이전트의 모니터 부분으로 넘겨진 후 다시 계획 부분으로 넘겨진다. 계획 부분은 위 목표를 분해하여 자신에게는 없는 못을 $a2$에게 빌려줄 것을 제안하고 응답을 기다린다. 이 때 소켓을 이용하여 통신 부분을 통해 전송되는 실제 메시지는

$proposal(a1, a2, hang\_picture, nail)$

이며 이것을 간단히 표기하면

proposal(*sender*, *receiver*, *goal*, *tool*)

형태이며, 여기서 *tool*은 *goal*을 달성하기 위해 필요한 부족한 자원이다.

한편 위와 같은 제안을 받은 *a*2는 *tool*이 수용거부(필요한 자원의 충돌)되는지 검사한다. 불행히도 *tool*은 수용거부되어 *a*2는 이제 자신의 목표인

b(*do*(*a*2, *hang_mirror*))

를 달성하려 한다. 다시 *a*2는 위 목표를 분해하여 아래와 같이 망치를 *a*1에게 역제안하고 응답을 기다린다:

proposal(*a*2, *a*1, *hang_mirror*, *hammer*)

역제안을 받은 *a*1은 *hammer*가 수용거부되기 때문에 *hang_mirror*를 해결할 수 있는 대안(스크루드라이브, 스크루, 그리고 거울을 사용하는 방법)을 찾아 다음과 같은 메시지를 *a*2에게 전송한다:

alternative(*hang_mirror*, *screwdriver*, *screw*, *mirror*)

목표를 달성할 수 있는 대안을 받은 *a*2는 우선 자신이 알고 있는 *hang_mirror* 방법(망치, 못, 그리고 거울을 사용하는 방법)을 지식베이스에서 제거하고 전송받은 대안을 삽입한다. 다시 *a*2는 자신의 목표 달성을 재계획하여 거울을 걸려고 하고 필요한 스크루드라이브를 *a*1에게 요청한다:

proposal(*a*2, *a*1, *hang_mirror*, *screwdriver*)

*a*1은 스크루드라이브를 제공하는 것이 수용거부되지 않으므로 *a*1은 스크루드라이브를 *a*2에게 넘겨주고

give(*a*1, *a*2, *screwdriver*)

다시 *a*2는 스크루를 *a*1에게 요청하며

proposal(*a*2, *a*1, *hang_mirror*, *screw*)

*a*1은 스크루를 제공하는 것이 수용거부되지 않으므로 *a*1은 스크루를 *a*2에게 넘겨준다:

give(*a*1, *a*2, *screw*)

이제 *a*2는 자신의 목표가 달성되어 *a*1에게 목표 달성을 통보한다:

achieved_goal(*a*2)

*a*2의 목표 달성을 통보받은 *a*1은 상대방의 목표가 달성된 것으로 믿고 자신의 목표를 달성하기 위해 재계획하여 못을 *a*2에게 요청하게 된다:

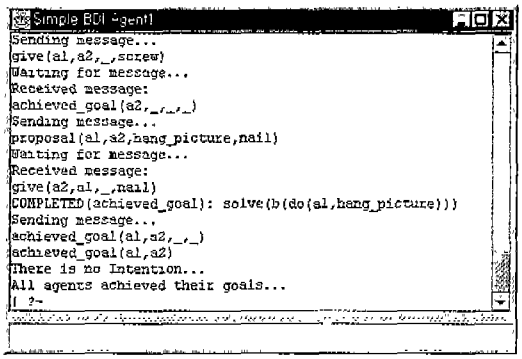

그림 7. 에이전트 *a*1의 협상 과정

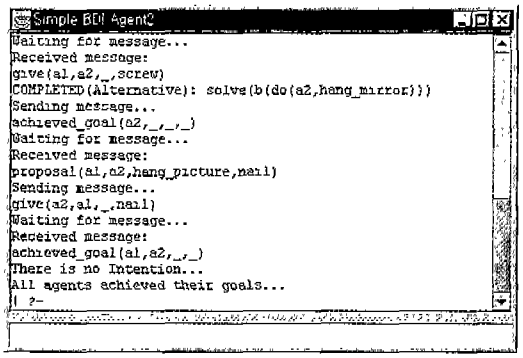

그림 8. 에이전트 *a*2의 협상 과정

proposal(*a*1, *a*2, *hang_picture*, *nail*)

이제는 못이 *a*2에 의해 수용거부되지 않기 때문에 *a*1에게 넘겨준다:

give(*a*2, *a*1, *nail*)

*a*1의 목표가 이제 달성되어 목표 달성을 *a*2에게 통보하고 협상을 종료한다:

achieved_goal(*a*1)

*a*1의 목표 달성을 통보받은 *a*2는 역시 협상을 종료한다. 이렇게 해서 *a*1과 *a*2는 그림 7 및 그림 8과 같은 협상을 통해 서로 수용가능한 일치에 도달하게 되고 각자의 목표를 달성한다.

5. 비교 및 평가

Rao와 Georgeff의 BDI 모델[12]은 modal 논리를 이용하여 에이전트의 믿음, 소망, 의도를 표현하는 모델들 중의 하나이다. 특히 이 모델에서는 에이전트가 포함될 수 있는 가능한 세상들의 집합을 믿음으로 표

현하기 때문에 세상에 관한 에이전트의 모델은 비현실적이다. 한편 Parson과 Giorgini는 BDI 에이전트를 모델링하기 위해 믿음, 소망, 의도를 각각 다른 요소로 설정하고 bridge 규칙을 사용하여 믿음, 소망, 의도 사이의 관계를 표현하였다[10]. 그리고 통신 요소로 작업의 종료가 전달되면(*C: done(e)*) 이 사실을 믿음의 집합에 추가하고(*B: B(done(e))*), 에이전트가 어떤 의도를 가졌을 때(*I: I(does(e))*) 이것을 통신 요소로 전달되게 하였다(*C: does(e)*). 본 연구에서는 에이전트들이 상호작용하여 그들의 믿음에 대해 일관성이 없음을 알았을 때 그들의 믿음을 변경할 수 있게 하여 가능한 한 믿음을 한정지으려 하였다. 그리고 BDI 에이전트의 구조를 지식베이스, 계획, 모니터, 통신과 같이 네 가지 요소들로 구성하여 논리 프로그래밍 환경에서 작업 수행을 보다 정교하게 만들려고 시도하였다.

멀티에이전트 시스템에서 에이전트는 특정한 문제를 해결하기 위해 협력 혹은 경쟁하는 엔터티로 볼 수 있는데, 본 연구에서는 서로 다른 목표를 달성하려 하는 BDI 에이전트를 가정하고 이들 사이의 협상에 관해 살펴보았다. 여러 가지의 에이전트 성질들 가령, 에이전트가 본질적으로 협상하려 하고 우호적이다는 가정 하에서 이루어진 연구이지만, 여기서 제안한 수용거부 조건과 상대방 에이전트에 관해 완전한 지식을 가지고 있지 않은 상황에서 에이전트 사이의 협상이 원만히 이루어짐을 보였다.

2절에서 언급한 몇몇 협상 프로토콜들과 본 연구 결과를 비교하면 다음과 같다: CNP는 에이전트들의 목표 충돌을 허용하지 않으며, 매니저 에이전트가 모든 협상을 담당하게 된다[4]. 여기서의 BDI 에이전트 협상은 CNP와는 달리 목표가 충돌하는 에이전트들을 가정하고, 매니저 역할을 담당하는 에이전트를 통하지 않고 에이전트 사이의 협상이 가능함을 보였다. Argumentation 시스템[11]은 진보한 협상 시스템이지만, 수신한 메시지에 포함된 상대방 에이전트의 argument를 자신의 입장에서 평가하는 과정을 거치게 된다. 이 과정을 통해 argumentation 시스템은 수신한 argument에 찬성 혹은 반대하는 argument를 만들어 수신한 argument를 평가한다. 따라서 argumentation 시스템은 메시지의 해석과 생성 작업에 많은 시간을 소비한다. 그리고 자신이 만든 argument를 메시지에 추가하여 전송하기 때문에 통신의 부담이 커지게 된다. 본 논문에서의 BDI 에이전트 협상은 argument를 사용하지 않고도 에이전트 사이의 협상이 가능함을 보였고, 따라서 통신의 부담 역시 argumentation 시스템과 비교할 때 훨씬 줄어든다.

에이전트의 협상 전략은 그가 속해있는 환경에 따라 다양하게 변화한다. 본 연구에서는 공동의 목표 달성과 부족한 자원의 할당에 관해 협상하는 환경을 가정하였다. 한편 LISP으로 구현된 전자상거래 시스템인 Kasbah에서는 사용자를 대신하는 구매 에이전트와 판매 에이전트를 생성하여 Kasbah 시장에서 이들 에이전트들이 협상한다[3]. 본 연구와 마찬가지로 이들 에이전트들도 충돌하는 목표들을 가지는데, 시간이 흐름에 따라 세 가지의 가격 decay(raise) 함수들을 사용하여 제안 가격을 변화시키면서 협상을 한다. 한편 Tete-a-Tete는 양방향 argumentation에 기초한 소매 상거래로의 특별한 협상 접근방법을 제공하는데, 협상은 XML에 기초한 제안, 비평, 역제안 등의 교환을 포함한다[15]. Tete-a-Tete에서 쇼핑 에이전트는 제안이 자신의 소유주의 기호를 얼마나 잘 만족시키는가를 기준으로 그 제안을 평가한다.

6. 결 론

본 논문에서는 MAS을 위한 BDI 에이전트의 믿음, 소망, 그리고 의도와 이를 바탕으로 한 협상 기법을 논리 프로그래밍의 입장에서 표현하고, 소켓을 사용하여 MAS에서 협상을 통한 목표 충돌이 어떻게 해결되는가를 보였다. 에이전트는 자신의 목표 달성에 방해되는 제안을 받은 경우에는 수용거부 조건을 검사해야 하는데, 본 논문에서 사용한 반대의 개념이 이러한 조건에 적합하였다. 간단한 예제를 통해서 다른 에이전트에 관한 완전한 지식을 갖지 않은 상황에서도 에이전트들의 협상이 가능함을 보였다.

본 연구는 여러 형태의 전문화된 에이전트와 협상 전략의 필요성을 보여주는데, 예를 들어 부족한 자원의 할당을 요구하는 에이전트, 소매 상거래 에이전트, 그리고 경매 에이전트의 협상 전략은 완전히 달라야 할 것이다. 또한 전자상거래에서의 협상 전략으로 이용할 수 있는 BDI 에이전트의 협상 전략을 개발해야 할 것이고, 컴퓨터 네트워크에서의 에이전트의 활동성과 작업 균형을 위해 모빌 멀티에이전트 BDI 시스템을 고려해야 할 것이다.

이러한 결과를 바탕으로 abductive 추론 기능을 BDI 에이전트 구조에 추가하여 다른 에이전트의 행위나 능력에 관해 일종의 가설적 추론을 행할 수 있다[5]. 즉, 어떤 에이전트의 행위로부터 그가 어떤 의도(혹은 목표)를 갖고 있는지, 혹은 어떤 지식을 갖고 있는지를 가설적으로 추론하여 에이전트 사이의 상호 협동이나 목표의 불일치로 인한 협상에 중요한 근거로 사용할 수 있을 것이다. 또한 논리적인 입장에서

전자상거래 환경에서 발생하는 협상의 한 가지 기법으로 BDI 에이전트를 활용할 수 있을 것이다.

참고문헌

[1] Bussmann, S., and Muller, H. J., A Negotiation Framework for Cooperating Agents, Proc. of the CKBS-SIG Conf., 1992.

[2] Cammarata, S., McArthur, D., and Steeb, R., Strategies of Cooperation in Distributed Problem Solving, Proc. J. Conf. Artif. Intell., 1983.

[3] Chavez, A., and Maes, P., Kasbah: An Agent Marketplace for Buying and Selling Goods, PAAM, 1996.

[4] Davis, R., and Smith, R. G., Frameworks for Cooperation in Distributed Problem Solving, Readings in Distributed Artif. Intell., Morgan Kaufmann, 1980.

[5] Kowalski, R. A., and Sadri, F., From logic programming towards multi-agent systems, Annals of Mathematics and Artif. Intell., 1999.

[6] Laasri, B., Laasri, H., Lander, S., and Lesser, V., A Generic Model for Intelligent Negotiating Agents, Int. *J. Intell. Coop. Inf. Syst*, 1(1), 1992.

[7] Maes, P., Guttman, R. H. and Moukas, A., Agents that Buy and Sell: Transforming Commerce as we Know It, CACM, 1999.

[8] Moulin, B., and Chaib-Draa, B., An Overview of Distributed Artificial Intelligence, Foundations of DAI, Wiley & Sons, 1996.

[9] Muller, H. J., Negotiation Principles, Foundations of DAI, Wiley & Sons, 1996.

[10] Parson, S., and Giorgini, P., On Using Degrees of Belief in BDI Agents, Proc. of the Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems, 1998.

[11] Parson, S., Sierra, C. and Jennings, N. R., Agents that reason and negotiate by arguing, J. of Logic and Computation, 8(3), 1998.

[12] Rao, A. S., and Georgeff M. P., BDI Agents: From Theory to Practice, ICMAS, 1995.

[13] Rosenschein, J. S., and Zlotkin, G., Rules of Encounter, MIT Press, 1994.

[14] Wooldridge, M., A Logic of BDI Agents with Procedural Knowledge, Proc. of 2nd Workshop of the MODELAGE Project. 1996.

[15] Zacharia, G., Moukas, A. and Maes, P., Agent-mediated Integrative Negotiation for Retail Electronic Commerce, Proc. of the Workshop on Agent Mediated Electronic Trading, 1998.

이 명 진 (Myung-Jin Lee)

1990년 : 대구대학교 수학과 이학사
1994년 : 계명대학교 컴퓨터공학과 공학석사
1999년 : 계명대학교 컴퓨터공학과 박사과정 수료
관심분야 : 인공지능, 에이전트 시스템, 에이전트 협상

김 진 상 (Jin-Sang Kim)

제 10 권 제 3 호 참조