다중온톨로지의 에이전트 통신

임동주*, 오창윤*, 배상현*

Agent Communication with Multiple Ontologies

Dong Ju Im, Chang Yun Oh, Sang Hyun Bae

요 약

본 논문에서는 이종 분산 지식기반 시스템을 구축하는데에 온톨로지가 어떤 역할을 하는지에 대해 논의한다. 먼저 다중 에이전트 시스템 기반 지식 공유와 재사용의 구조물인 지식사회에서 온톨로지와 에이전트간 관계를 논의하는데 온톨로지는 각 에이전트에 있어 지식사회에 연결되기 위한 최소한의 요구조건이다. 둘째로 지식사회에서 온톨로지가 어떻게 사용되고 있는가를 보여주기 위해 온톨로지에 의한 중재를 설명한다. 중재자라 불리는 특별한 에이전트는 주소가 없는 메시지들을 분석해서 온톨로지와 상의하고 온톨로지와 에이전트간 관계를 참고로 해서 수신후보 에이전트들을 추론한다. 셋째, 개념화 방식을 나타낼 수 있는 각 상황들의 결합으로서 온톨로지를 모델링한다. 상황통합을 의미하는 결합상황 혹은 상황선택을 의미하는 범주상황으로서 상황들이 결합된다. 상황에 의한 온톨로지는 실세계 현상에 대한 이종 다중 기술을 허용하기 때문에 이종 지식기반 시스템에게는 적합하다. 또한 다중상황을 해석하는 방식으로서 메시지 번역을 보여준다. 번역에이전트는 상황들의 종속성을 분석하여 어떤 상황을 가진 메시지를 다른 상황을 가진 메시지로 번역할 수 있다. 에이전트들을 쉽고 자연스럽게 구축하기 위해서 메시지의 중재와 번역은 중요한데 각 에이전트에 대하여 다른 에이전트들에 대한 지식이 덜 요청되기 때문이다.

ABSTRACT

In this paper, we discuss how ontology plays roles in building a distributed and heterogeneous knowledge-base system. First, we discuss relationship between ontology and agent in the Knowledgeable Community which is a framework of knowledge sharing and reuse based on a multi-agent architecture. Ontology is a minimum requirement for each agent to join the Knowledgeable Community. Second we explain mediation by ontology to show how ontology is used in the Knowledgeable Community. A special agent called mediator analyzes undirected messages and infer candidates of recipient agents by consulting ontology and relationship between ontology and agents. Third we model ontology as combination of aspects each of which can represent a way of conceptualization. Aspects are combined either as combination aspect which means integration of aspects or category aspect which means choice of aspects. Since ontology by aspect allows heterogeneous and multiple descriptions for phenomenon in the world, it is appropriate for heterogeneous knowledge-base systems. We also show translation of messages as a way of interpreting

multiple aspects. A translation agent can translate a message with some aspect to one with another aspect by analyzing dependency of aspects. Mediation and translation of messages are important to build agents easily and naturally because less knowledge on other agents is requested for each agent.

1. 서 론

실세계에서 인공지능 이론들이 작동할 수 있도록 하기 위해서는 대규모 지식베이스는 필수 불가결한 요소이다. 대규모 지식베이스를 실현시킬 수 있는 두 가지 접근방법이 있는데 하나는 Cyc[1][2] 같은 대규모 지식베이스 시스템을 구축하는 것이고 다른 하나는 다른 시스템간의 공통 언어와 온톨로지에 의한 지식공유와 재사용의 구조물을 제공하는 것이다.

지식사회라 불리는 프로젝트의 목적은 다중 에이전트 시스템 기반 지식 공유와 재사용의 구조물을 제공하는 것이다[3][4]. 지식 공유와 재사용 측면에서 서로 상이한 개념 구조들을 어떻게 같이 사용할 수 있는가는 가장 중요하지만 어렵다. 각 시스템이나 각 에이전트는 개념들이 어떻게 정의되어지고 서로 관련되어지는가에 대한 자신의 개념 구조를 선택한다. 각 개념구조는 개념들이 다루고 있는 영역과 영역을 개념들로서 어떻게 기술할 것인가에 대한 방침을 가지고 있다. 다른 개념 구조를 가지고 있는 두 개의 에이전트가 서로 통신하려고 할 때 개념 구조의 차이가 통신을 교란시킨다. 개념 구조의 차이에는 두 가지 이유가 있는데 하나는 어떤 개념들이 사용되고 있는지 개념들이 다른 에이전트에서는 어떻게 연결되고 있는지 각 에이전트가 안다는 것이 어렵다는 것이고 다른 하나는 개념 구조간의 관계를 안다 할지라도 각 에이전트가 다른 에이전트의 개념 구조를 해석하는 것이 어렵다는 점이다. 각 에이전트에서 사용된 개념들과 개념들의 관계가 첫 번째 문제를 해결할 수 있도록 하는 온톨로지들을 제공한다. 각 에이전트가 자신의 온톨로지를 명백하게 선언하도록 요구된다. 가능한 에이전트들이 주어진 주소가 없는 메시지들에 응답하도록 권할 수 있는 온톨로지를 사용한 중재 메커니즘을 실현시킨다. 두 번째 문제를 위하여 개념들에 대한 다중 상황을 허용하고 한 상황의 정보를 다른 상황의 정보로 번역하기 위해 사용되는 여러 상황들간의 관계를 제공한다.

본 논문의 2장에서는 대규모 지식베이스 시스템 기반 다중 에이전트 시스템을 구축하는 접근 방법을 보여준다. 특히 에이전트와 온톨로지간의 관계를 밝힌다. 3장은 온톨로지 사용의 응용으로서 주소가 없는 메시지의 중재를 보여준다. 3장에서는 단 하나의 온톨로지로 가정하지만 4장에서는 상황을 온톨로지의 구성 요소로 도입하고 온톨로지를 상황구조로 정의하는데 이러한 정의를 통해서 다중 온톨로지가 공존한다. 또한 상황들간의 관계를 정의하여 한 상황에서의 표현이 어떻게 다른 상황에서의 표현으로 번역될 수 있는지 보여준다. 마지막으로 5장에서 결론을 내린다.

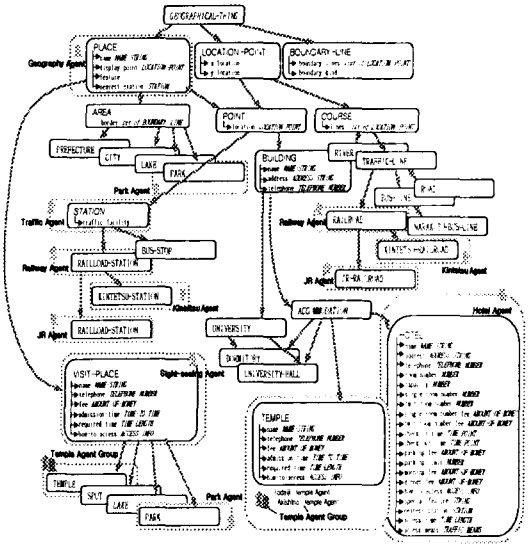

그림 1. 에이전트와 온톨로지 관계의 예

2. 지식사회 온톨로지

지식사회는 협력하는 에이전트들의 인위적 사회이다. 각 에이전트는 데이터베이스와 같은 어떤 계산능

* 조선대학교 전산통계학과
접수일자: 2000. 11. 7

력, 혹은 인간과의 상호작용, 혹은 두 가지 혼합된 형태를 나타낼 수 있다고 가정하자. 지식사회에서의 에이전트들은 다음과 같은 능력을 갖도록 요구된다.

- 통신능력
  각 에이전트는 다른 에이전트와 통신할 수 있다. 지식사회의 멤버가 되는 것은 최소한의 요구조건이다. 이러한 요구조건은 통신언어(유일한 언어가 요구되는 것은 아니다)를 공유해야 한다는 것을 의미한다.

- 개념구조의 선언
  각 에이전트는 개념구조를 선언해야 하는데 다른 에이전트들이 에이전트가 어떤 종류의 개념을 다룰 수 있는지를 추측할 수 있도록 하기 위해서다.

- 처리능력의 선언
  각 에이전트가 무엇을 할 수 있는지 혹은 적어도 무엇을 할 수 있을 것으로 기대되는지 선언해야 한다.

그림 1은 온톨로지의 한 예를 보여주고 있다. 그림에서처럼 각 에이전트가 온톨로지를 공유함으로써 즉, 온톨로지의 일부분을 가짐으로써 두 번째 포인트가 실현된다. 그러면 다른 에이전트들은 각자 가지고 있는 온톨로지의 일부분을 비교함으로써 자신과 다른 에이전트간의 관계를 추측할 수 있다. 본 논문에서는 온톨로지를 프레임 온톨로지로 제한한다. 즉 클래스, 관계, 그리고 클래스 계층구조로 제한한다.

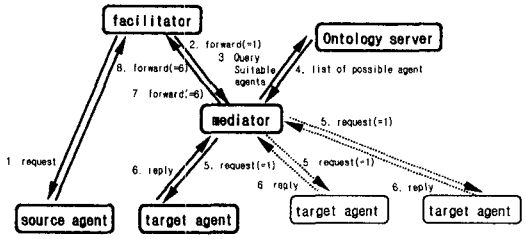

그림 2. 온톨로지에 의한 중재

그림 2는 원형 시스템 KC- Kansai에서 사용된 여행계획 온톨로지를 보여준다. 하나의 에이전트가 생성되어 지식사회에 추가될 때 온톨로지는 사용된다. 먼저, 에이전트 프로그래머가 새로운 에이전트를 프로그램 할 때 온톨로지와 상의한다. 프로그래머들이 의도한 것과 똑같은 혹은 유사한 개념들을 발견한다면 에이전트 프로그램에서 이러한 기존의 개념들을 사용할 수 있다. 둘째, 새로운 개념들과 그들과의 관계를 도입한 새로운 에이전트가 추가될 때 온톨로지의 단편 또한 기존 온톨로지에 추가된다. 지식사회에서 온톨로지의 관리인으로서, 그리고 온톨로지와 에이전트간 관계의 관리인으로서 온톨로지 서버를 제공하는데 이러한 서버의 주요 기능들은 다음과 같다.

1) 에이전트에 의해 사용된 에이전트와 온톨로지의 일부분을 수용한다.
2) 온톨로지를 일관성 있게 한다.
3) 요청된 온톨로지와 관련된 온톨로지의 일부분에 대하여 응답한다.
4) 요청된 에이전트와 관련된 온톨로지의 일부분에 대하여 응답한다.
5) 요청된 온톨로지와 관련된 에이전트들의 집합에 대하여 응답한다.

3. 단일 온톨로지와의 에이전트 통신

단 하나의 온톨로지만을 가정할 경우 이는 모든 에이전트가 온톨로지에 있는 모든 개념의 사용에 동의함을 의미하기 때문에 메시지를 해석하는데에 혼동이 없다. 즉, 모든 메시지에는 단 하나의 해석만이 존재할 뿐이다. 그러면 다음 문제는 메시지를 적절한 에이전트에게 중재하는 것이다. 모든 에이전트가 단 하나의 온톨로지 사용에 동의한다 할지라도 온톨로지의 모든 것을 해석할 수는 없고 다만 온톨로지의 어떤 부분만을 다룰 수 있을 뿐이다. 모든 메시지는 그 메시지의 온톨로지와 짝이 되는 에이전트의 온톨로지에 의해 처리되어야 한다. 반면에 에이전트가 다른 에이전트에 대한 지식을 반드시 가져야 한다는 것은 아니기 때문에 에이전트가 수신인이 정해져 있지 않은 메시지를 만드는 것은 자연스럽다. 따라서 지식사회가 그러한 주소 없는 메시지를 중재하는 것은 중요한 임무이며 모든 에이전트는 온톨로지 내에 배치되었기 때문에 온

톨로지를 사용하여 주소 없는 메시지를 위해 적절한 에이전트들을 결정할 수 있다. 온톨로지 서버는 온톨로지를 알고 온톨로지와 에이전트간의 관계를 알고 있으므로 가능한 에이전트들에 대한 개념 계층구조를 검색함으로써 주어진 메시지 속에 있는 개념과 관련 있는 후보 에이전트들을 답할 수 있다.

중재는 그림 2와 같이 촉진자[5][6], 중재자, 그리고 온톨로지 서버에 의해 수행되어진다. 모든 메시지 전송은 촉진자들을 통해서 이루어진다. 수신자가 정해지면 촉진자는 단지 그 메시지를 수신인에게 넘겨 줄 것이다. 그렇지 않으면 촉진자는 에이전트와 온톨로지에 대한 지식에 근거하여 수신인을 결정할 중재자라 불리는 중재 에이전트에게 그 메시지를 넘길 것이다. 중재자는 온톨로지 서버와 상의하여 가능한 수신 에이전트를 결정해서 응답 메시지가 성공적으로 돌아올 때까지 반복해서 그 메시지를 그 에이전트 각각에게 보낼 수 있다. 온톨로지 서버는 가능한 에이전트를 제의하기 위해 수신된 메시지를 분석한다. 먼저 메시지 내용 속에 서술부를 검사함으로써 메시지의 주 클래스들을 탐지한다. 그리고 나서 맨 처음 주 클래스들과 관련된 에이전트들을 탐색하고 그 다음에 그들의 하위 클래스들, 그 다음 순서로 그들의 상위 클래스들을 탐색한다. 마지막으로 중재자는 주어진 메시지와 관련된 후보 에이전트들의 리스트를 돌려준다. 그림 3은 메시지가 중재를 위하여 어떻게 넘겨지는가를 보여준다.

```
(ask-one :content (and (hotel ?x)
                       (name ?x "Nara-hotel" )
                       (nearest-station ?x ?y))
         :aspect ?y :language KIF)
```

(a) Message 1

```
(broker-all :content
  (ask-one :content (and (hotel ?x)
                         (name ?x "Nara-hotel")
                         (nearest-station ?x ?y))
           :aspect ?y :language KIF)
  :reply-with q1)
```

(b) Message 2

```
(recommend-all :content
  (ask-one :content (and (hotel ?x)
                         (name ?x "Nara-hotel")
                         (nearest-station ?x ?y))
           :aspect ?y :language KIF)
  :reply-with q1)
```

(c) Message 3

```
(ask-one :content (and (hotel ?x)
                       (name ?x "Nara-hotel")
                       (nearest-station ?x ?y))
         :receiver sight-seeing-agent
         :aspect ?y :language KIF)
:reply-with q1)
```

(d) Message 4

그림 3. 중재를 위한 순방향 메시지들

메시지는 KIF[7][8]와 KQML[9]로 표현되는데 이 그림에서 메시지 수는 그림 2와 일치한다. 최초 메시지의 내용이 변했는데 과정 중에 KQML 수행 타입만이 변했다는 것에 주목해야 한다.

4. 다중상황

본 장에서는 다중 온톨로지를 실현시키기 위해 상황을 온톨로지의 구성요소로서 도입한다.

4.1 상황

3장에서는 단일 온톨로지만을 가정했다. 하지만 비교적 큰 온톨로지들을 구축한다는 것은 쉽지 않다. 그 이유 중의 하나는 온톨로지들을 구축하려고 할 때 여러 개념화로부터 개념들을 가끔씩 혼동하기 때문이다. 예를 들어 그림 4는 temple이라는 개념이 여러 가지로 어떻게 모델링 되는가를 보여준다. 사람들은 temple을 역사책 속의 하나의 항목으로 생각할지도 모르며 역사적인 사건 등이 temple과 함께 사용된다. 또한 사람들은 temple을 종교장소로서 생각할 수 있으며 교리, 수도원장 등이 temple과 관련된 개념들이다. 여러 개념화로부터 오는 혼합된 개념들은 가끔씩 혼동되는데 더 많은 개념들이 모이면 모일수록 개념의 의미가 덜 명확해진다. 개념이 적절한 방법으로 사용되기만 하면 즉 동일한 개념화로부터 온 개념들과 함께 사용되기만 하면 각 개념의 의미는 분명하다. 이러한 것을

상황이라 칭한다. 상황은 개념화에 대한 일관된 생각이다.

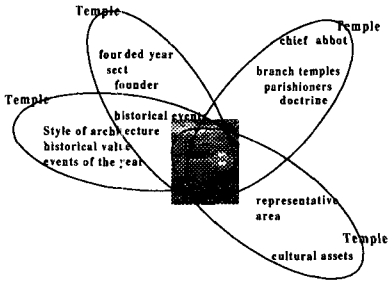

그림 4. 사원에 대한 온톨로지

온톨로지는 여러 상황들로 구성될 수 있다. 다양한 상황들이 사용되고 있는데 예컨대 엔지니어링 분야에서는 동력학, 운동학과 같은 상황을 사용하고 상식세계를 모델링하기 위해 교통상황을 사용할지 모른다. 상황에 대해 두 가지 문제가 있는데 그 하나는 상황의 대상이 무엇이어야 하는 가이다. 상황은 그 영역에 속한 현상을 기술할 수 있는 어휘를 가져야 하고 또한 어휘로 개념을 결합시키는 이론을 가져야 한다. 이론은 일관되어야 한다. 말하자면 상황은 일관성 있게 세상을 개념화시킬 수 있는 어떤 것이다. 또다른 하나는 다른 상황으로부터 상황을 어떻게 구성할 것인 가이다. 상황들간의 두 가지 형태의 기본적 연결을 제공하는데 첫 번째는 결합상황으로 이것은 단지 여러 영역에 대한 상황의 통합이다. 예를 들어 여행상황을 구축하는 방법중의 하나는 호텔상황과 교통상황을 결합하는 것이다. 이러한 경우에 교통상황에 있는 철로와 같은 개념은 호텔상황에는 존재하지 않는데 그 이유는 모델링 영역이 서로 다르기 때문이다. 여행상황에서는 양쪽 상황으로부터의 개념들을 이용해서 여행과 같은 개념이 정의된다. 두 번째는 범주상황으로 이것은 영역을 공유하지만 여러 개념화로부터 오는 상황들의 축적이다. 그림 4의 temple이 여러 가지로 모델링될 때 temple에 대한 범주상황이 있다는 것을 가정한다. 이러한 상황은 구성요소로서 역사책으로서의 temple에 대한 상황과 종교장소로서의 temple에 대한 상황과 같은 temple에 대한 구체적인 상황들을 가지고 있다. 구성요소 상황들이 영역을 공유하고 있기 때문에 여러 구성요소 상황 속의 개념들간에 관계가 있다는 것은 반드시 필수적이라 할수 없지만 합리적이다. 그러한 관계는 범주상황의 내용이다.

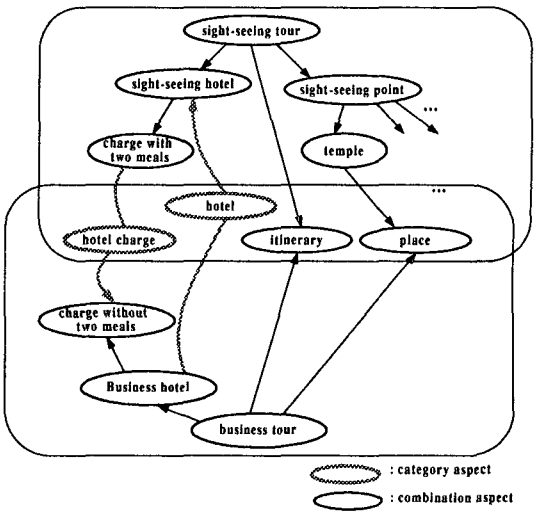

그림 5. 다중 및 공유 온톨로지

결합과 범주상황은 구성요소로서 다른 결합 혹은 범주상황을 이용할 수 있기 때문에 상대적으로 적은 상황들로부터 큰 상황들을 구축할 수 있다. 그러한 상대적으로 큰 상황들을 온톨로지라 부른다. 그림 5는 여러 온톨로지가 공유상황들을 가지고 어떻게 정의되어질 수 있는가에 대한 예를 보여준다. 두 개의 상황들은 그들의 구조 속에 있는 상황들을 공유할 수 있거나 혹은 범주상황에 의해 연결될 수 있다. 그러한 경우에 이러한 두 상황들이 양립한다고 할 수 있다. 즉, 그들은 정보를 공유하거나 서로에게 전달한다.

다음 절에서는 먼저 논리적 구조 속에 있는 상황을 기술하고 그 다음으로 프로그래밍 구조 속에 있는 상황을 기술하며 마지막으로 여러 상황들간에 통신이 어떻게 설정될 수 있는가를 보여준다.

4.2 상황이론

먼저 원자상황을 정의하고 그 다음에 복합상황을 정의하자. 원자상황은 실세계를 모델링하기 위한 일관된 견해를 정의하는 최소한의 이론이다.

원자상황 A는 상황이름 name(A)와 상황이론 T(A)로 이루어져 있다. 상황이론 T(A)는 일관된 논리식 집

합이다. 논의의 편의상 $T'(A)=T(A)\wedge name(A)$라고 정의하자. 복합상황은 자신을 정의하기 위해 두 개 이상의 상황을 사용하는 상황이다. 두 가지 형태의 복합상황, 즉 결합상황과 범주상황이 있다면 전자는 두 가지 이상의 이론들이 하나로 통합된 상황이다. 그것이 두 개의 다른 개념화 방법으로부터 온 개념들이 같이 사용된다는 것을 의미한다. 결합상황 $A_{COM}(A_1,\cdots,A_n)$의 상황이론은 다음과 같이 정의된다.

$$T(A_{COM}(A_1,\cdots,A_n))=T'(A_1)\wedge\cdots\wedge T'(A_n)\wedge I(A_1,\cdots,A_n)$$

여기서 이 정의는 일관되어야 한다. $I(A_1,\cdots,A_n)$는 $A_1,\cdots,A_n$ 사이의 상황간 이론이다. 반면에 범주상황은 훨씬 더 복잡한데 상황이론 둘 다 항상 참이어야 한다는 것을 의미하지 않기 때문이다.

범주상황을 표현하기 위해서 형식 연산자 □과 ◇를 도입하고 S4 형식 시스템을 가정한다. 범주상황 $A_{CAT}(A_1,\cdots,A_n)$은 다음과 같이 정의된다.

$$T(A_{CAT}(A_1,\cdots,A_n))=\Diamond T'(A_1)\wedge\cdots\wedge\Diamond T'(A_n)\wedge I(A_1,\cdots,A_n)$$

이것 또한 일관되어야 한다. 복합상황은 상황이론간의 관계를 기술하는 상황간 이론을 가지고 있다. 결합상황을 위한 상황간 이론은 통합 상황이론에 부가된 이론일 뿐이고 반면에 범주상황을 위한 상황간 이론은 한 상황에서 다른 한 상황에 매핑시키는 함수로서 작용한다. 그러면 양립성이나 포함과 같은 어떤 상황간 관계들을 정의할 수 있다. 상황이론에 대한 자세한 사항은 참고문헌 [10]을 참조하기 바란다.

4.3 상황언어

상황계산모델은 온톨로지 언어와 같은 온톨로지 정의의 확장이다[11]. 원자상황의 정의는 상황이름의 선언과 클래스, 관계, 그리고 함수의 정의로 이루어져 있다. 그림 6의 (d)와 (e)는 원자상황의 예이다. 결합상황의 정의는 (b)와 (c)에 나타난 바와 같이 원자상황의 정의이고 포함하는 상황의 선언이다. 범주상황의 정의는 번역공식의 집합으로 이루어져 있는데 번역공식은 그림 6의 (a)와 같이 범주상황 속에 있는 두 상황간에 정의되어지고 두 상황 속에 있는 개념간의 논리적 관계를 기술하는 정의-번역으로 정의되어진다. 공식에서 좌측 은 첫 번째 변수상황의 공식이고 우측은 두 번째 변수상황의 공식이다.

```
(define-category-aspect FEE (fee/a fee/b))
(define-translation FEE
     (=> (fee/A!fee ?fee) (fee/B!fee ?fee))
  ((:query-precedence nil
   :inform-precedence nil
   (-> (fee-value ?fee ?value)
        (and (adult ?fee ?fee1)
             (student ?fee ?fee2)
             (fee-value ?fee1 ?value)
             (fee-value ?fee2 ?value))))))

(define-translation FEE
     (=> (fee/B!fee ?f) (fee/A!fee ?f))
  ((:guery-precedence nil
   :inform-precedence nil
   (-> (and (adult ?fee ?fee1)
            (student ?fee ?fee2)
            (fee-value ?fee1 ?value1)
            (fee-value ?fee2 ?value2))))))
            (max ?value1 ?value2 ?max-value))
            (fee-value ?fee ?value))))
  ((:query-precedence nil
   :inform-precedence nil
   (-> (and (student ?fee ?fee2)
            (fee-value ?fee2 ?value))
        (fee-value ?fee ?value)))
)
```

(a) Category Aspect fee

```
(define-aspect temple/A (TEMPLE)
  (:use fee/A))
(in-aspect temple/A)
(define-class temple (?x)
  :def (and (has-one ?x name)
            (has-one ?x fee)))
(define-function name (?x)
  :-> ?n
  :def (and (temple ?x) (string ?n)))
(define-function temple-fee (?x)
  :-> ?f
  :def (and (temple ?x) (fee ?f)))
```

(b) Combination Aspect temple/A

```
(define-aspect temple/B (TEMPLE)
  (:use fee/B))
(in-aspect temple/B)
(define-class temple (?x)
  :def (and (has-one ?x name)
            (has-one ?x fee)))
(define-function name (?x)
  :-> ?n
  :def (and (temple ?x) (string ?n)))
(define-function temple-fee (?x)
  :-> ?f
  :def (and (temple ?x) (fee ?f)))
```

(c) Combination Aspect temple/B

```
(define-aspect fee/A (FEE)
(in-aspect fee/A)
(define-class fee (?fee)
  :def (has-one ?fee fee-value))
(define-function fee-value (?fee)
  :-> ?val
  :def (and (fee ?fee) (natural ?val)))
```

(d) Atomic Aspect fee/A

```
(define-aspect fee/B (FEE)
(in-aspect fee/B)
(define-class fee (?fee)
  :def (and (has-one ?fee adult)
            (has-one ?fee student)))
(define-class fee-elm (?elm)
  :def (has-one ?elm fee-value))
(define-function fee-value (?elm)
  :-> ?val
  :def (and (fee-elm ?elm) (natural ?val)))
(define-function adult (?fee)
  :-> ?elm
  :def (and (fee ?fee) (fee-elm ?elm)))
(define-function student (?fee)
  :-> ?elm
  :def (and (fee ?fee) (fee-elm ?elm)))
(define-function make-fee (?elm1 ?elm2)
  :-> ?fee
  :def (and (fee-elm ?elm1) (fee-elm ?elm2)
            (fee ?fee) (adult ?fee ?elm1)
            (student ?fee ?elm2)))
(define-function make-fee-elm (?val)
  :-> ?elm
  :def (and (natural ?val) (fee-elm ?elm)))
```

(e) Atomic Aspect fee/B
그림 6. 상황 정의의 예

1.4 에이전트 통신을 위한 상황간의 번역

상황간 관계를 다루는 접근은 다른 상황들을 가지고 있는 에이전트간의 메시지를 번역하는 것이다. 번역공식은 한 상황에서 다른 상황으로 정보를 변환하는 번역에이저트에 의해 사용된다. 3장에서 언급한 것과 같이 단 하나의 온톨로지만 있다면 메시지를 해석하는 데에 애매모호함은 없으나 다중상황이 허용될 경우 메시지가 근거하고 있는 상황들을 명시하는 것이 필요하고 가끔씩 한 상황으로부터 다른 상황으로 메시지를 번역하는 것이 필요하다. 따라서 번역 에이전트는 그림 7에서처럼 중재자와 목표 에이전트 사이에 삽입된다. 두 가지 형태의 메시지는 정보메시지와 질문메시지이다. 정보메시지를 위한 번역은 단지 주어진 메시지를 번역하는 것이고 반면에 질문메시지를 위한 번역은 그림 8에 나타난 바와 같이 주어진 메시지 이외에 응답메시지를 번역하는 것이다.

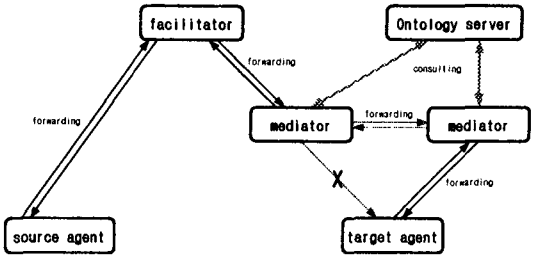

그림 7. A 지식사회에 있어서 번역에이전트

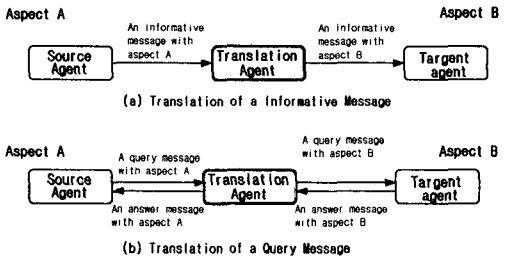

그림 8. 다중 상황을 가진 에이전트 통신

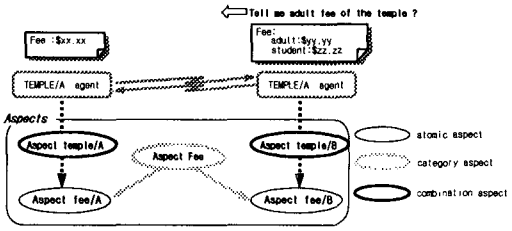

그림 9. 에이전트 사이 통신의 예

4.4.1 번역에이전트의 기능

번역에이전트의 기본 기능은 다음과 같다.

1) 번역에이전트는 중재자를 통해서 소스 에이전트로부터 목표 에이전트로 메시지를 수신한다.
2) 번역에이전트는 메시지를 분석해서 온톨로지 서버로부터 번역공식을 검색한다.
3) 번역에이전트는 번역공식을 참고해서 목표 에이전트의 상황으로 씌어진 새로운 메시지를 구성하고 목표 에이전트를 보내는데 메시지가 정보메시지이면 프로시저는 여기서 종료된다.
4) 번역에이전트는 목표 에이전트로부터 응답메시지를 수신할 때까지 대기한다.
5) 번역에이전트는 최초의 메시지를 구성하기 위해

사용된 동일한 번역공식을 참고해서 소스 에이전트의 상황으로 씌어진 응답메시지를 구성하고 그것을 소스 에이전트에게 보낸다.

진행 중에 번역에이전트는 소스와 목표 에이전트에 있는 두 개의 상황을 포함하는 범주상황을 이해할 수 있도록 행동한다.

4.4.2 번역절차

번역에이전트는 상황간의 온톨로지와 번역공식에 근거한 주어진 메시지를 다음과 같이 분석한다.

- **범주상황 탐색** : 먼저 번역에이전트는 주어진 메시지의 상황과 목표 에이전트의 상황을 결합할 범주상황을 찾기 위해 그것들을 분석한다. 번역에이전트는 포함관계를 추적함으로써 이러한 상황 속에 포함된 모든 상황들을 수집하여 만약 상황이 메시지의 상황과 목표 에이전트의 상황의 포함된 상황 속에 있는 두 가지 상황 모두를 포함하고 있다면 상황은 그것들을 결합할 범주상황이다. 그러면 번역에이전트는 이런 범주상황 속에 있는 번역공식을 검색한다.

- **메시지 내의 클래스 구별** : 번역에이전트는 메시지를 분석해서 메시지 안에 있는 각 용어의 클래스를 구별하는데 클래스 술부가 사용되면 변수용어의 클래스가 구별되며 그렇지 않을 경우 술부와 기능의 정의를 참고해서 용어의 클래스를 구별한다. 번역에이전트는 모든 용어에 대한 클래스를 메시지에 추가시킨다.

- **번역공식 적용** : 적절한 번역공식을 적용함으로써 메시지가 수정된다. 정보메시지의 경우에 공식의 왼쪽 부분이 메시지와 일치하면 메시지의 일치된 부분은 공식의 오른쪽 부분에 의해 대체된다. 질문메시지 경우에 공식의 오른쪽 부분이 적용되며 용어의 결합은 응답메시지의 번역을 위해 보존된다.

- **불필요한 문자들의 제거** : 목표 에이전트의 상황에 포함되지 않은 문자들은 제거시킨다.

```
(temple ?x)
(name ?x ?y)
(fee ?x ?f)
(adult ?f ?f1)
(student ?f ?f2)
(fee-value ?f1 ?v1)
(fee-value ?f2  ?v2)
(< ?v1 500)
(< ?v2 400)
```

(a) The given message

```
(temple ?x)
(name ?x ?y)
(fee ?x ?f)
(adult ?f ?f1)
(student ?f ?f2)
(fee-value ?f1 ?v1)
(fee-value ?f2 ?v2)
(< ?v1 500)
(< ?v2 400)
(string ?y)
(temple-fee ?f)
(temple-fee-elm ?f1)
(temple-fee -elm ?f2)
```

(b) Adding class definitions

```
(temple ?x)
(name ?x ?y)
(fee ?x ?fee)
(fee-value ?fee ?value)

(< ?value 500)
(< ?value 400)
(string ?y)
(temple-fee ?fee)
(temple-fee-elm ?fee1)
(temple-fee -elm ?fee2)
```

(c) Applying a translation formula

```
(temple ?x)
(name ?x ?y)
(fee ?x ?fee)
(fee-value ?fee ?value)

(< ?value 500)
(< ?value 400)
(string ?y)
(temple-fee ?fee)
```

(d) Removing unnecessary literals

그림 10. 번역의 예

그림 10은 번역이 메시지에 어떻게 적용되는지를 보여준다. 여기서 두 개의 에이전트 temple/A와 temple/B는 temple/A와 temple/B 상황들을 각각 사용한다. 그림 9를 보면 temple/A 에이전트에 입장료는 있지만 성인입장료는 없다는 것을 수용하지만 temple/B 에이전트는 temple/A 에이전트에게 temple의 성인 입장료를 요구한다. 이러한 예에서 상황 temple/B를 가진 질문메시지는 그림 10의 (a)와 같이 상황 temple/A를 가진 메시지로 번역되는 걸로 기대된다. 상황 temple/B와 temple /A는 그림 6의 (d)와 (e)처럼 상황 fee/B와 fee /A를 각각 이용하기 때문에 그림 6의 (a)와 같이 두 개의 상황을 포함하고 있는 범주상황 fee가 검색된다. 반면에 그림 10의 (b)에서는 용어의 클래

스 정의가 메시지에 추가된다. 그림 6의 (a)의 6~10번째 줄은 첫 번째 번역공식 메시지에 적용되어서 그림 10의 (C)에 있는 메시지로 번역되고 마지막으로 불필요한 글자들은 (d)와 같이 제거시킨다.

5. 결론

온톨로지가 이종 분산 지식베이스 시스템을 구축하는데에 어떠한 역할을 하는지 논의했다. 온톨로지는 지식베이스 시스템사회에 결합되기 위해 각 지식베이스 시스템에 대한 최소한의 요구조건의 하나이다. 이종 지식베이스 시스템을 위한 온톨로지는 이질적이어야 하는데 그 이유는 각 시스템의 견해에 따른 기술이 허용되어야 하기 때문이다. 온톨로지를 각각의 개념화 방식을 나타낼 수 있는 상황의 결합으로서 모델링했기 때문에 온톨로지는 실세계의 현상에 대한 이종 다중 기술을 허용한다. 따라서 온톨로지는 이종 지식베이스 시스템의 온톨로지로서 적합하다. 또한 메시지번역을 다중상황 해석방법으로서 보여주었는데 중재의 결합과 메시지의 번역은 상호간의 협력시스템의 구축을 용이하게 하는데 있어서 다른 시스템에 대한 지식이 비교적 덜 요구되기 때문이다.

Reference

[1] R.V. Guha. Representation of defaults in Cyc. In Proc. AAAI-90. pages 608-614, 1990.

[2] R.V. Guha and D.B. Lenat. Cyc: A midterm report. AI magazine, pages 32-59, Fall 1990.

[3] T. Nishida. Towards integration of heterogeneous knowledge for highly autonomous analysis of dynamical systems-preliminary report from the PSX project. Journal of Information Processing, 15(3):350-363, 1992.

[4] T. Nishida and H. Takeda. Towards the knowledgeable community. In Processings of International Conference on Building and Sharing of Very Large-Scale Knowledge bases '93, pages 157-166, Tokyo. 1993.

[5] R.S. Patil, R.E. Fikes, P.F. Patel-Schneider, D. McKay, T. Finin, T.R. Gruber, and R. Neches. The DARPA knowledge sharing effort: Progress report. In C. Rich, B. Nebel, and W. Swartout, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference. Morgan Kaufmann, 1992.

[6] M.R. Cutkosky, R.S. Engelmore, R.E. Fikes, M.R. Genesereth, T.R. Gruber, W.S. Mark, J.M. Tenenbaum, and J.C. Weber. PACT: An experiment in integrating concurrent engineering systems. IEEE Computer, January 1993:28-38, 1993.

[7] M. Genesereth. Knowledge interchange format. In Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, pages 599-600. Morgan Kaufmann, 1991.

[8] M. Genesereth and R.E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, June 1992.

[9] T. Finin, D. McKay, R. Fritzson, and R. McEntire. KQML: An information and knowledge exchange protocol. In K. Fuchi and T. Yokoi, editors, Knowledge Building and Knowledge Sharing. Ohm놈 and IOS Press, 1994.

[10] H. Takeda, K. Iino, and T. Nishida. Ontology-supported agent communication. In Working notes of 1995 AAAI Spring Symposium on Information Gathering in distributed environments, 1995.

[11] T.G. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Knowledge Systems Laboratory, Stanford University, 1992.

[12] H. Takeda and T. Nishida. Integration of aspects in design processes. In J.S. Gero and F. Sudweeks, editors, Artificial Intelligence in Design '94, pages 309-326. Kluwer Academic Publishers, 1994.

임 동 주(Dong-Ju Im)
1985년 : 전남대학교 영문학과 졸업 (문학사)
1993년 : 미국뉴욕주립대학교 대학원 전산학과 (이학석사)
1994년~1999년 : 대불대학교 근무
1999년 : 조선대학교 대학원 전산통계학과(이학박사)
※주관심분야 : 컴퓨터네트워크, 멀티미디어, 소프트웨어엔지니어링, 데이터베이스 등

오 창 윤(Chang-Yun Oh)
1992년 : 조선대학교 전산통계학과 졸업(이학사)
1994년 : 조선대학교 대학원 전산통계학과 (이학석사)
1994년~1996년 : ㈜아시아자동차 근무
2000년 : 조선대학교 대학원 전산통계학과(이학박사)
※주관심분야 : 컴퓨터네트워크, 전자상거래, 네트워크 보안, 멀티미디어 등

배 상 현(Sang-Hyun Bae)
1982년 : 조선대학교 전기공학과(공학사)
1984년 : 조선대학교 대학원 전기·전자공학과 (공학석사)
1988년 : 일본 동경도립대학 전자정보통신공학부 (공학박사)
1984년~1985년 : 일본동경공대 객원연구원
1995년~1996년 : 일본 NAIST 초빙교수
1999년~현재 : 조선대학교 전산통계학과 교수
※주관심분야 : 대규모 지식베이스, 인공신경망, 퍼지시스템, GIS, 전문가시스템, 지식처리