1960년대 후반, 인공지능 연구가 한동안 침체되었다가 1970년대에 들어서 기존 문제 해결의 탐색이나 추론을 중심으로 한 연구로부터 새롭게 시스템 안에 지식을 도입하여 고도의 전문적인 작업을 수행하는 인공지능 시스템의 연구개발이 성행하게 되었다. 그와 함께 이 시기에 여러 가지 지식표현 방법이 개발되었다.

대표적인 지식 표현의 종류로서 앞에서 설명한 기호 논리로 표현하는 것이나 의미 네트워크, 프레임, 프로덕션 규칙, 온톨로지 등이 열거된다. 본 장에서는 의미 네트워크와 온톨로지에 대하여 설명한다.

6.1 지식의 표현 형식

6.1.1 지식의 유형

인공지능에서 지식 표현을 크게 두 가지 타입으로 분류하면 다음과 같이 된다.

* 절차적 지식(procedural knowledge)

“만약 ~이라면 ~이다”와 같이 문제-해결의 절차를 기술한 지식. 영어로 표현하면 How에 해당하고, 프로그램 언어로 표현하면 if~then 규칙에 해당한다.

* 선언적 지식(declarative knowledge)

“ …이다”와 같이 어떤 사상을 선언적으로 기술한 지식. 영어로 표현하면 What에 해당하고, 프로그램 언어로 표현하면 데이터와 같은 것에 해당한다.

6.1.2 지식의 성질

지식의 성질로 본 종류들로 다음과 같다. 영국의 철학자이자 물리화학자인 마이클 폴러니(Michael Polanyi)가 구분한 지식의 한 종류이다. 폴러니는 지식을 암묵지(암묵적 지식)와 명시지 또는 형식지(形式知)으로 구분하였는데, 암묵지는 학습과 경험을 통하여 습득함으로써 개인에게 체화되어 있지만 언어나 문자로 표현하기 어려운, 겉으로 드러나지 않는 지식을 말한다. 명시지(명시적 지식)는 암묵지와 상대되는 개념으로서 언어나 문자를 통하여 겉으로 표현된 지식으로서 문서화 또는 데이터화된 지식이라고 할 수 있다.

* 전문지식(expert knowledge/ expertise)

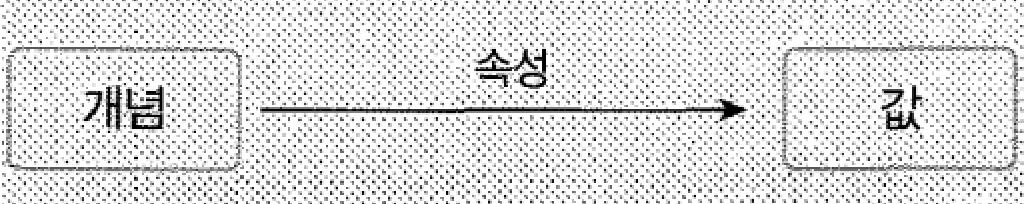
명시지 또는 형식지(explicit knowledge)라고도 하며 지식으로 명시화하는 것이 가능한 것이 여기에 해당한다. 예를 들어, 특정분야의 전문가가 문제 해결을 수행할 때 가지고 있는 전문적인 지식 등을 말한다. 어떤 상태 하에서 대처방법 등의 지식도 포함되기 때문에 절차적 지식으로 표현하기 쉽다.

* 상식 (common sense): 암묵적 지식

암묵지 (tacit knowledge) 라고도 하며, 만인이 공유하고 있어 특별히 명시화되지 않는 것이 이에 해당된다. 명시화되지 않기 때문에 표현하기가 곤란하고 상식을 추론에서 사용하는 것이 지적인 시스템 개발에 중요하지만 그것을 다루는 것은 여전히 문제가 되고 있다.

6.1 .2 의미 네트워크

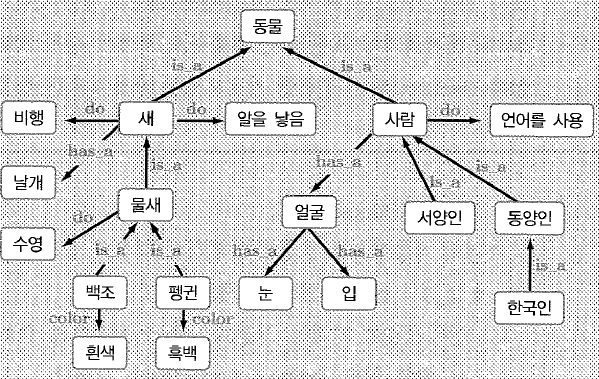
의미 네트워크 (semantic network) 란 1960년대 후반에 콜린스(A. M. Collins) 와 퀼리언 (M. R. Quillian) 에 의해 고안된 지식표현이다. 지식을 개념 (concept)과 그들을 연결하는 관계 (relation)로 기술하여 표현한다. 표현된 지식은 라벨 붙은 방향성 그래프 (labeled directed graph) 로 표현된다. 아래 그림은 의미 네트워크의 기본 구성을 보였다.



6.1.2.1 개념이 가지는 성질

개념은 그 성질을 나타내는 속성 (property / attribute) 정보를 가지며, 속성이 구체적인 값을 가짐으로써 개념이 실체(instance)로서 존재한다. 개념은 계층구조(hierarchy)로 체계화되어 표현되며 계층구조 하에서 상위 개념의 속성이 하위개념으로 상속(inheritance) 되는 구조를 가짐으로써 효율적으로 복잡한 지식표현을 가능하게 한다.

개념을 기술하는 데에는 실제로 그 개념이 어떤 대상영역에서 사용되고 있는가라는 관점이 중요해진다. 그래서 구조적 속성에서 정의 속성과 성질 속성에 대한 확고한 기준이 존재하지 않고 대상영역에 따라 개념이 기술된다고 볼 수 있다. 아래는 의미 네트워크의 예이다.



* 계층구조(hierarchy)

개념의 계층구조를 표현하기 위하여 다음과 같은 관계가 이용된다.

* + is\_a: 상위-하위와 관계를 표현.

상위개념이 가지는 성질(속성)은 기본적으로 하위개념에도 이어받게 된다는 속성 상속 (inheritance of property)을 가진다.

예) “동물”은 “새”의 상위개념이고， 속성 상속에 의해 “새”는 “동물”의 속성을 공유한다. “갈매기”는 “새”의 하위개념이므로 “새”의 속성을 상속한다.

* + has\_a: 부분-전체의 관계를 표현.

개념을 구성하는 요소와 그 개념과의 관계를 나타낸다. 속성 상속은 존재하지 않는다.

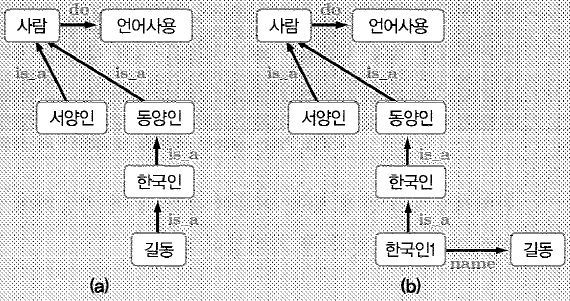
예) “손”은 “인간”의 부분이고， “손가락”은 “손”의 부분이다.

* 구조적 속성 (structural property)

개념의 속성은 다음 두 종류의 속성으로 분류된다.

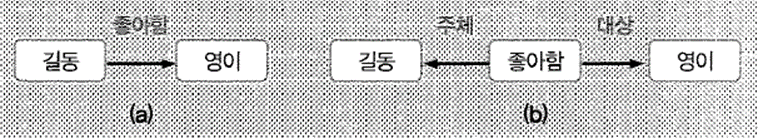
* + 정의 속성: 통상 어떤 개념을 설명하기 위하여 필요한 속성을 나타내는 속성. 개념이 필연적으로 가지는 속성이므로 미리 속성의 값을 부여해 둘 수가 있다. 이러한 값을 디폴트값 (default value) 라고 한다. 예를 들어 “호텔” 등의 개념에서는 위치, 숙박비 등이 정의 속성으로 고려된다. 그리고“비즈니스호텔”이라면 숙박비용이 “약 7만원 전후” 등과 같은 디폴트값을 부여해 둘 수가 있다.
  + 성질 속성: 어떤 사상을 개념을 이용하여 설명할 때 필요한 속성에 해당한다. 예를 들어， “펭귄” 등의 개념을 나타낼 때는 펭귄이 가지고 있는 성질로서 깃털의 색깔 등이 있다. 이것은 펭귄을 설명하고자 할 때 필요한 경우에 한해서 사용되는 속성이므로 성질 속성으로 간주된다.
* 개념과 실체(인스턴스: instance)

“길동은 한국인이다”라는 지식을 의미 네트워크로 기술하면 아래 그림에서 나타내는 (a) 또는 (b) 의 기술방법을 생각할 수 있다 is←a 관계에는 엄밀히 분류하면 개념의 “상위-하위”의 관계와 “개념-실제” 관계가 포함되어 있어 주의해야 한다. (a) 의 표기라면 “길동”은 “한국인”의 하위개념이라는 해석이 되어버릴 가능성이 있다. 한편 (b)는 개념 “한국인”의 하나의 실체로서“어떤 한국인”이라는 의미로 “한국인1" 이 is→a로 연결되어 있다. 이와 같은 표현은 “길동”이 개념 “한국인”의 실체(인스턴스) “한국인1"의 이름이 “길동”이라는 사람이라는 것을 명시하고 있다.



6.1.2.2 문장의 의미내용 표현

의미 네트워크는 문장의 의미내용을 표현하기 위한 수단으로도 이용된다. 문장의 의미내용을 표현할 때는 문장의 술어부률 관계로 간주하고, 그 관계를 이용하여 문장 내에 나타나는 개념을 연결하여 표현한다. 예를 들어， “길동은 영이를 좋아한다”라는 의미내용은 아래 그림 (a)와 같이 표현된다. 이 표현의 경우, 술어부 “좋아하다”가 개념 “길동”과 “영이”를 연결하는 관계가 된다. 그리고 그림 (b) 에서는 문장 내의 격구조(case structure)를 이용하여 술어부의 개념 “좋아하다”를 중심으로 하여 그것과 다른 개념의 관계를 나타낸 개념기술로 되어 있다. 그림 (b)와 같이 술어부(여기서는 “좋아하다")를 관계가 아니라 개념으로 다룸으로써 술어부도 개념관계를 가지고 있음을 알 수 있으며, 술어부의 개념에 관해서도 is\_a 관계 등을 이용한 추론 등이 가능하게 된다. 이 점이 술어부를 관계로 기술하는 것과 큰 차이점이 된다.

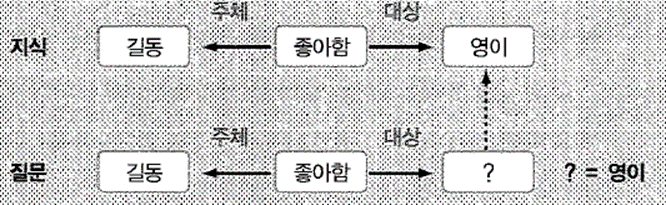


6.1.2.3 의미 네트워크를 이용한 추론

의미 네트워크를 이용한 추론에는 크게 나누어 “직접 대조에 의한 추론”과“간접 대조에 의한 추론” 그리고 “개념의 계층관계를 이용한 추론” 등이 있다. 아래에 순서대로 설명한다.

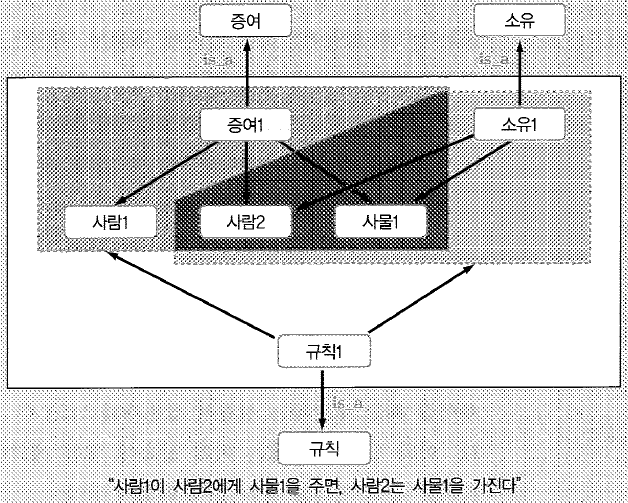
* 직접 대조에 의한 추론:

질문의 의미 네트워크에 대조하는 의미 네트워크를 지식베이스 안에서 검색함으로써 해를 구한다. 아래 그림에 나타내는 것과 같이 지식으로서 의미 네트워크가 상단에 주어져 있다. 이에 대해 질문내용을 나타내는 의미 네트워크가 주어지면 네트워크 구조의 대조에 기반하여 질문내용에 결여되어 있는 정보를 추출한다. 이것을 의미 네트워크의 “직접 대조에 의한 추론”이라고 한다.



* 간접 대조에 의한 추론:

주어진 사실로부터 추론규칙 형식의 의미 네트워크를 이용하여 새로운 사실을 이끌어냄으로써 수행되는 추론 방법을“간접 대조에 의한 추론”이라고 한다. “길동은 영이에게 꽃을 선물하였다”라는 사실이 있다고 하자. 이에 대해 “영이는 무엇을 받았습니까?"라는 질문이 주어졌다고 하자. 이 때 “증여와 소유”에 관한 인과 관계가 아래 그림과 같은 의미 네트워크로 주어져 있다고 하자. 전건부에는 “길동이 영이에게 꽃을 선물하였다”라는 사실이 직접 대조에 의해 매칭된다. 따라서 후건부로서 “영이는 꽃을 받았다”라는 사실이 도출된다.



* 개념의 계층관계를 이용한 추론:
  + 상속(inheritance):

개념의 계층관계를 이용한 추론으로 “백조는 새인가?"라는 질문에 대해서 is\_a 관계를 따라감으로써 개념 “백조”의 상위개념으로 “새”가 발견된다면 질문내용이 참이라고 답할 수 있다. 또한 “백조는 날 수 있는가?"라는 질문에 대해서 “새”가 상위개념이라는 것이 판별되면 속성 상속에 의해 “백조는 날 수 있다”라는 것을 알 수 있다.

* + 예외(exception):

펭귄은 새이므로 “펭귄은 날 수 있는가?"라는 질문에 대하여 속성 상속을 수행하면 “날 수 있다”라는 답을 얻을 수 있다. 그러나 펭귄은 새이지만 날 수가 없기 때문에 예외로 처리할 필요가 있다. 이 경우에는 펭귄의 개념에 상위개념(여기서는 “새")으로부터의 속성 상속에서 예외를 기술해둔다. 이것을 예외처리 (exception handling) 라고 한다.

* + 다중 상속(multiple inheritance):

하나의 개념이 복수의 상위개념으로부터 속성 상속이 이루어지는 것을 다중 상속이라고 한다. 다중 상속이 이루어질 때는 모순되는 성질이 상위개념으로부터 상속될 가능성이 존재한다. 이와 같은 경우에는 미리 상속에 대한 제약이나 개념의 계층관계를 따지는 등의 처리를 하는 등 주의해야 한다.

* 의미 네트워크의 특징
  + 장점:

1. 방향성 그래프에 의한 네트워크 표시에 의해 시각적으로 지식을 표현할 수 있기 때문에 사람이 직관적으로 이해하기 쉽다.
2. 지식의 추가, 갱신이 비교적 쉽다.
3. 개념의 계층관계를 정의함으로써 개념이 가지는 속성 상속을 계층관계에 의해 실현할 수 있기 때문에 복잡한 지식의 구조화를 실현할 수 있다.

* 단점:

1. 추론규칙을 대상영역마다 준비해야 할 필요가 있다.
2. 지식의 양이 많아지면 관리가 어렵게 된다.
3. 개념이나 관계의 정의가 임의로 정해지기 때문에 지식표현으로써 통일성이 보장되지 않는다.

6.1.3 온톨로지

온톨로지 (ontology)란 원래 철학에서 존재론을 가리키는 것으로 사물이 “존재한다”라는 것에 대한 철학적 고찰이다. 이에 대해 인공지능 분야에서는 “온톨로지”라는 단어가 나타내는 의미는 철학에서의 “존재론”과는 달리 대상세계에 대한 시각으로써， 사물이 거기에 어떻게 존재하는가에 착안하여 “그 사물의 개념을 명시적으로 기술한 사양”을 말한다. 그 배경에는 온톨로지로 기술된 개념을 여러 시스템에서 공유와 재사용을 가능하도록 하는 목적이 존재한다.

인공지능 연구가 온톨로지 연구에 이르는 경위를 돌아보자. 1970년대에 시작된 지식을 포함시킨 시스템 개발에 의한 인공지능 연구의 부흥으로부터 많은 전문가 시스템이 구축되었다. 당시에 전문가 시스템 개발자들이 항상 문제로 삼고 있었던 것이 전문가의 지식을 추출하여 지식표현으로 만드는 것에 막대한 비용

이 지불됨에도 불구하고 구축된 지식이 그 시스템에서만 이용되고 다른 시스템에서 재사용성이 낮다는 것이었다. 그와 같은 배경으로부터 지식이나 데이터를 공유하는 기술의 필요성이 많이 제기되어 왔고 그 흐름은 1980년대 후반에 지식의 공유와 재사용을 목표로 하는 온톨로지 공학의 탄생으로 연결되었다.

최근에는 웹 상에서의 메타데이터 (metadata)로서 사용되는 것에 주목하여 효율이 좋은 정보검색 · 관리를 실현하기 위해 반드시 필요한 기술로 인식되고 있다.

6.1.3.1인공지능에서의 온톨로지 정의

위에서 온톨로지란 “사물의 개념을 명시적으로 기술한 사양”이라고 기술하였으나， 엄밀하게 정의하기는 어렵다. 문헌 [15J 에 의하면 “온톨로지는 인간이 대상세계를 어떻게 보고 있는가라는 근원적인 문제의식을 가지고 사물의 내력을 풀어헤쳐서 그것을 컴퓨터와 인간이 이해를 공유할 수 있도록 기록한 것”이라고 설명하고 있다. 그리고 W3C의 문헌 [48J 에 의하면 “온톨로지란 어떤 영역의 지식을 기술하고 표현하기 위하여 사용되는 어휘를 정의하는 것이며, 정보의 공유가 필요한 사람, 데이터베이스, 어플리케이션 시스템 등에 의해 사용되는 것이다. 온톨로지는 컴퓨터에 의해 재사용 가능한 그 영역의 기본적인 개념과 그것들의 관계의 정의를 포함한다”고 되어 있다.

어느 쪽 생각도 배경에 존재하는 암묵적인 정보를 명시화하는 것을 목적으로 하여 그 명시화되는 정보가 어떻게 의미를 가지는가를 정의하기 위한 틀임을 알 수 있다. 명시적으로 정의된 개념(공통어휘)을 공유· 재사용함으로써 인간이나 시스템 간의 지식의 상호운용성(interoperability )을 높이는 것이 가능하다.

앞에서 다룬 의미 네트워크의 속성에 대한 정의에서는 많은 임의성이 남겨져 있음을 설명하였다. 그 임의성은 시스템 개발에 있어서는 제약이 적다는 장점이 있지만, 다른 시스템과의 연계를 고려한 실용적인 시스템의 개발에는 큰 장애가 된다. 따라서 온톨로지는 그와 같은 속성 정의도 공유할 수 있는 표준이 되는 지식의 표현 방법을 엄격하게 탐구하는 기술로서 큰 역할을 담당하고 있다.

6.1.3.2지식기술에서 온톨로지의 장점

지식을 온톨로지에 기반하여 기술하는 데에는 다음과 같은 장점이 존재한다.

1. 어떤 관점으로 대상을 다루는가가 규정되어 있기 때문에 지식의 표현이 쉽게 된다.
2. 지식을 표현하기 위해 이용하는 개념과 어휘가 사전에 준비되어 있다.
3. 온톨로지로 정의된 지식과 주어진 정보를 대조함으로써 정보의 제약 위반을 자동적으로 체크할 수 있다.
4. 기술된 지식의 “질”이 향상된다.
5. 지식을 공유 · 재사용할 수 있다.
6. 개념의 통일이나 관점의 일관성이 보장된다.
7. 다른 사람이 기술한 지식을 이해할 때 도움이 된다.

6.1.3.3온톨로지의 종류

온톨로지를 기술대상과 기술력의 차이에 따라 분류하면 다음과 같이 나눌 수 있다.

* 기술대상의 차이에 의한 구분
* 상위 온톨로지〈upper ontology):

철학의 온톨로지 에 가까운 것으로 만인이 공유할만한 개념체계를 가리킨다. 공학적으로는 그와 같은 개념체계의 구축은 어렵기 때문에 가능한 한 많은 사람이 받아들일만한 객관성이 큰 개념체계를 구축하는 것을 목표로 하여 개발되고 있다.

* 도메인 온톨로지 (domain ontology):

비즈니스, 법률, 정보과학 등 특정한 대상영역 (domain) 의 개념체계를 가리킨다.

* 태스크 온톨로지(task ontology):

진단， 설계， 학습지원 등 문제 해결 과정 자체를 대상으로 하는 개념체계를 가리킨다. 중심이 되는 개념은 “처리”와 “처리대상”이 기본이 된다.

* 기술력의 차이에 의한 구분
  + 라이트 웨이트 온톨로지(light weight ontology):

Yahoo! 등의 검색엔진이 가지고 있는 토픽의 계층 등의 메타데이터와 관련한 어휘집합이나 개념계층 등이 여기에 해당한다. 엄격하게 정의되어 있어서 지식을 기술하기 위한 개념이라기 보다는 어휘의 유사성 등을 다루는 사전이 갖춰져 있는 것과 같은 개념체계에 가까운 것을 말한다.

* + 헤비 웨이트 온톨로지(heavy weight ontology):

지식을 기술하기 위한 개념제계를 가리킨다. 개념이나 관계 하나하나가 세밀하게 정의되어， 엄

격하고 형식적인 정의와 의미제약기술 하에서 개념체계가 구축된다.

6.1.3.4온톨로지 구축을 위한 지침

온톨로지 구축이 어려운 가장 큰 원인은 “어떻게 온톨로지를 구축하면 좋을지 모른다”는 것이라고 생각된다. 실제로 온톨로지는 개념을 명시화하기 위한 명확한 방침이 없는 경우에는 대상이 되는 사물을 정확하게 명시화하는 개념을 정의하는 것이 불가능하다고 말해도 과언이 아니다. 여기서는 온톨로지 구축의 기본이 되는 개념 정의의 지침에 대하여 설명한다.

온톨로지 구축은 개념분류체계(taxonomy)를 구축하여 거기서 정의된 클래스(개념)를 이용하여 여러 가지 지식표현을 기술한다. 이를 위해서는 적절한 기준에 의해 클래스를 분류하여 분류된 클래스의 계층관계에서 상위-하위의 일관성 혹은 상속 등의 성질이 적절하게 표현된 체계를 구축한다.

1. 클래스 분류

어떤 클래스의 서브클래스 분류의 기준은 그 서브클래스의 성질에 기반하여 명확한 차이를 고려함으로써 수행된다. 클래스로서 정의할 때 동일성이 변화되는 것은 클래스로 정의해서는 안된다. 예를 들어 출세물고기(성장함에 따라 이름이 바뀌는 물고기)와 같이 성장함에 따라 부르는 이름이 달라지는 경우에는 개체의 동일성이 변하지 않는 것에 반해 이름이 바뀔 뿐이기 때문에 이름을 개념으로 정의하지 않고 이 경우에는 “물고기”를 개념으로 정의하는 것이 올바르다.

1. 클래스 계층으로서의 분류의 일관성

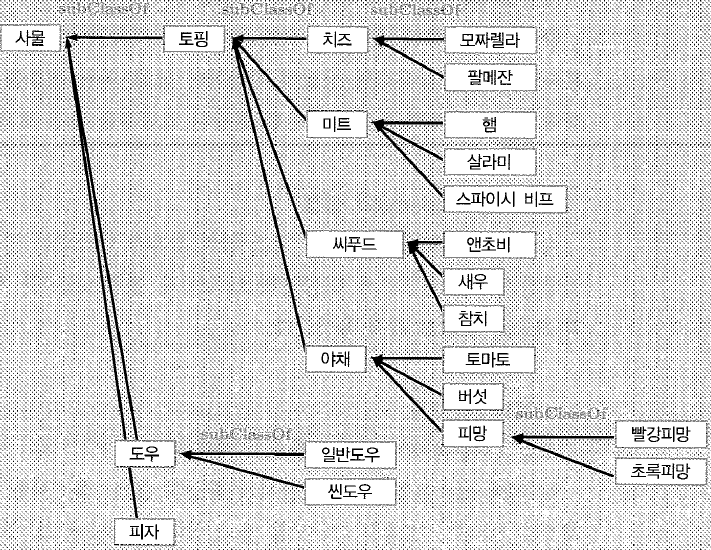
클래스로 분류를 진행하면 클래스 계층이 구축된다. 클래스 계층은 클래스를 분류할 때 서브클래스의 성질에 기반하여 상위 하위 계층을 구성하는데, 이 때 클래스 계층에서의 속성에 기반한 분류의 일관성을 가칠 필요가 있다.

1. 속성(property)의 성질

속성은 서로 다른 클래스의 요소 간의 관계를 나타내는 이항관계로 표현된다. 이항관계에서 인수로서 대상영역 (domain)과 범위 (range)를 사용하며, 클래스의 각 요소가 값으로 각각 들어간다.

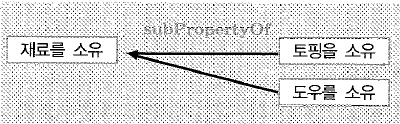
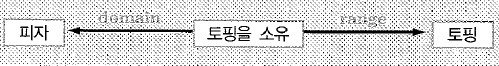
* + - 개념 분류 체계

아래 그림에 피자에 대한 온톨로지 예를 나타낸다. 이 예에서는 온톨로지를 기술하는 최상위 클래스를 “사물”로 하고， 그 하위 클래스로 “토핑 “도우 “피자”를 들고 있다. 이것들은 그 속성에서 명확한 차이를 가지고 있다. 그리고 토핑의 계층클래스를 보면 토핑의 종류에 따라 클래스 분류가 명확하게 이루어져 있음을 알 수 있다.



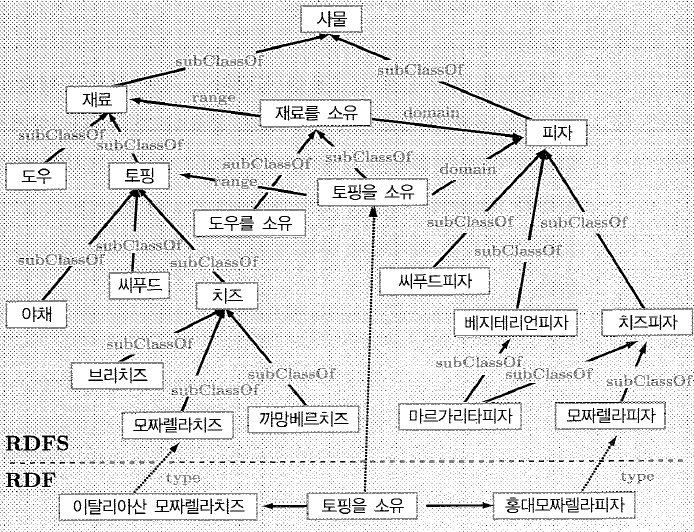
* + - 속성

속성도 클래스와 마찬가지로 계층구조를 가지고 있어서 속성의 상속도 계층 구조에 따라 이루어진다. 아래 그림의 좌측은 피자 온톨로지를 대상으로 한 속성 계층의 예를 나타낸다. “토핑을 소유” 속성에서 도메인(대상)과 범위 설정을 수행하면 아래 그림의 우측과 같이 된다.

“a는 토핑 b를 소유한다”라는 명제가 주어졌다면, 속성 “토핑을 소유”는 위 우측 그림에서 클래스 “피자”를 도메인으로 하고 있기 때문에 a는 클래스 “피자”의 요소라고 추론하는 것이 가능하고, 마찬가지로 b는 클래스 “토핑”의 요소라고 추론할 수 있다.

아래 그림에 속성도 포함하는 피자 온톨로지를 보였다. 이 속성을 포함하는 개념체계는 RDFS(Resource Definition Framework Schema)라는 기본적인 요소(어휘)를 제공하는 개념체계로써 표현된다. 이 개념체계로부터 실제화된 젓이 RDF(Resource Definition Framework)로 표현된다.



6.1.3.5온톨로지와 의미 네트워크의 차이점

의미 네트워크에서는 구조적 성질에 대해서는 상당한 임의성이 반해서 온톨로지는 개념, 계층구조가 명확한 지침에 의해 설계되어 있다. 그리고 다른 시스템과 개념을 공유하고 상호운용성을 수행하는 입장에서 지식이 표현되어 있다는 점에서 온톨로지와 의미 네트워크는 구별된다.

6.1.3.6온톨로지의 활용

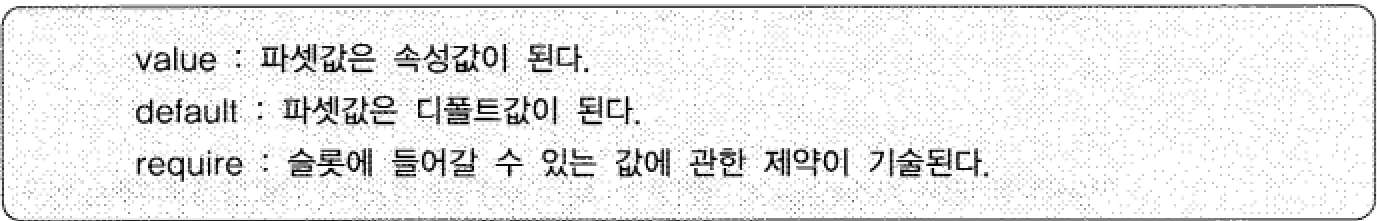
웹 상에 있는 문서에 내용을 설명하는 의미 정보인 메타 데이터를 부가함으로써 더욱 정확한 정보검색이나 정보 활용이 가능하다. 여기서 메타 데이터를 정의하기 위해 사용하는 공통의 어휘체계로서 온톨로지가 필요하게 된다. 그 뿐만 아니라 제품이나 시스템 개발 등에서 기능이나 구조 등을 온톨로지로 기술함으로써 설계자들 사이에서 공통 문제의 발견이나 시스템의 버그 발견 등에도 공헌한다.

6.2 프레임 이론과 오브젝트 지향

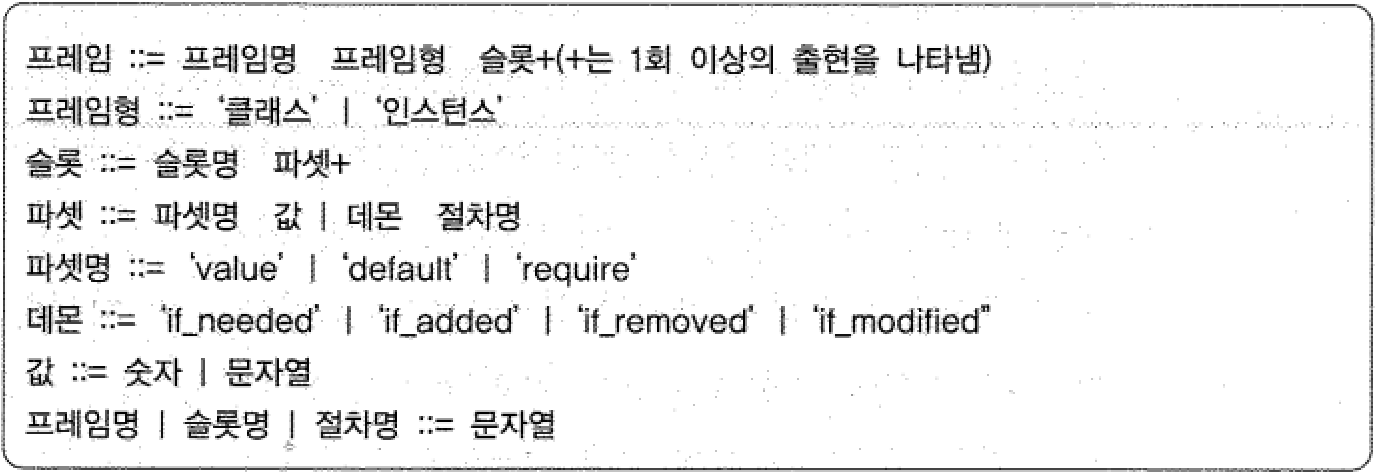
프레임 이론(frame theory)은 1957년에 민스키에 의해 제안한 것으로 인간이 가지고 있는 어떤 대상에 대한 전형적인 이미지를 나타내는 지식표현 방법이다. 인간은 어떤 대상을 전형적인 이미지의 형태로 기억하고 있어서 인간이 처음 접하는 상황에서도 그 상황에 대한 전형적인 장면 (scene)의 지식을 이용함으로써 쉽게 이해할 수가 있다. 예를 들어, 어떤 학교 교실에 처음 들어갔다고 하면, 그 사람은 교실에 대한 전형적인 장면 지식으로써 책상, 의자, 철판, 벽, 천정 등이 존재하는 공간이라는 지식을 가지고 있으므로 그 교실을 금방 이해할 수 있게 된다. 이와 같은 지식을 표현하는 틀이 프레임이다. 프레임은 의미 네트워크와 마찬가지로 상위-하위 계층관계에 의해 체계화되어 복잡한 지식을 표현할 수 있다.

6.2.1 프레임의 기본 형식

프레임 (frame)도 의미 네트워크와 마찬가지로 속성과 값의 쌍에 의한 표현 형식을 그 기본단위로 하고 있다. 프레임에서는 기본단위를 슬롯(slot) 이라고 하고， 프레임에 의한 지식 표현은 슬롯의 집합으로 표현된다. 슬롯의 속성에 해당하는 부분을 슬롯명 (slot name) 이라고 하고 이 부분은 의미 네트워크의 개념에 해당한 다. 그리고 값에 해당하는 부분을 슬롯값(slot value)이라고 하고, 슬롯값은 파셋 (facet) 이라는 부분 구조를 가진다. 파셋은 파셋명과 파셋값의 조합으로 기술되어， 파셋명에는 아래에 나타내는 파셋값이 취하는 값에 관한 여러 가지 정보가 기술 된다.



다음에 프레임의 기본적인 표현을 배커스-나우어 기법 (Backus-Naur form) 으로 나타낸다.



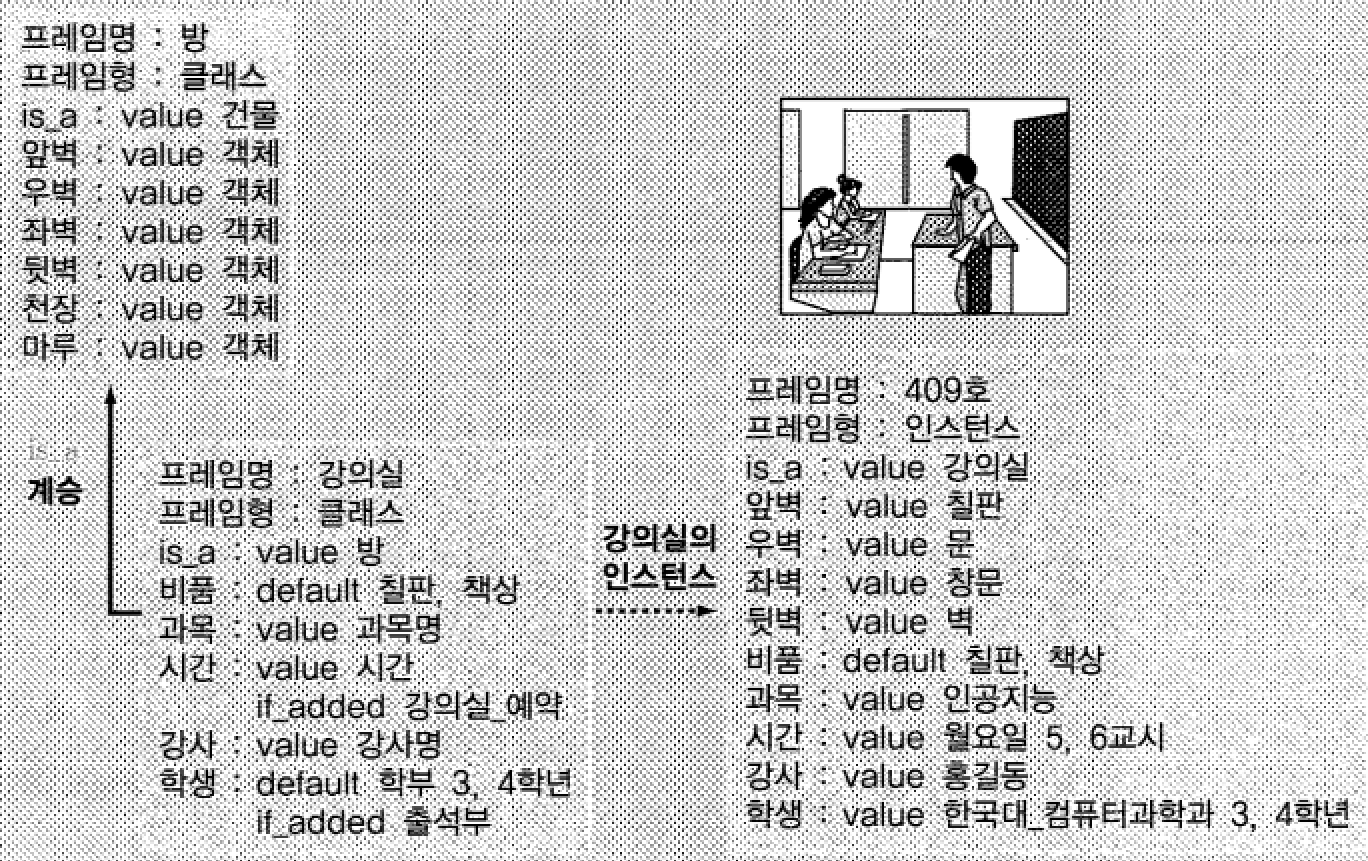
* 데몬

프레임에서는 슬롯값으로 데몬(demon)이라는 절차적인 지식을 기술할 수 있다. 데몬은 지정된 조건이 만족되면 기동되어 아래의 (l) ~(4)에 나타내는 여러 가지 절차가 실행된다. 이와 같이 프레임은 선언적인 지식과 절차적인 지식이 융합되어 있는 지식표현으로 되어 있다. 이것이 프레임의 가장 큰 특징이다.

1. if\_needed: 슬롯값을 구할 때 기동하는 데몬.
2. if\_needed: 슬롯값이 정해졌을 때 기동하는 데몬.
3. if\_needed: 슬롯값이 제거되었을 때 기동하는 데몬.
4. if\_needed: 슬롯값이 변경되었을 때 기동하는 데몬.

예) 교실 프레임

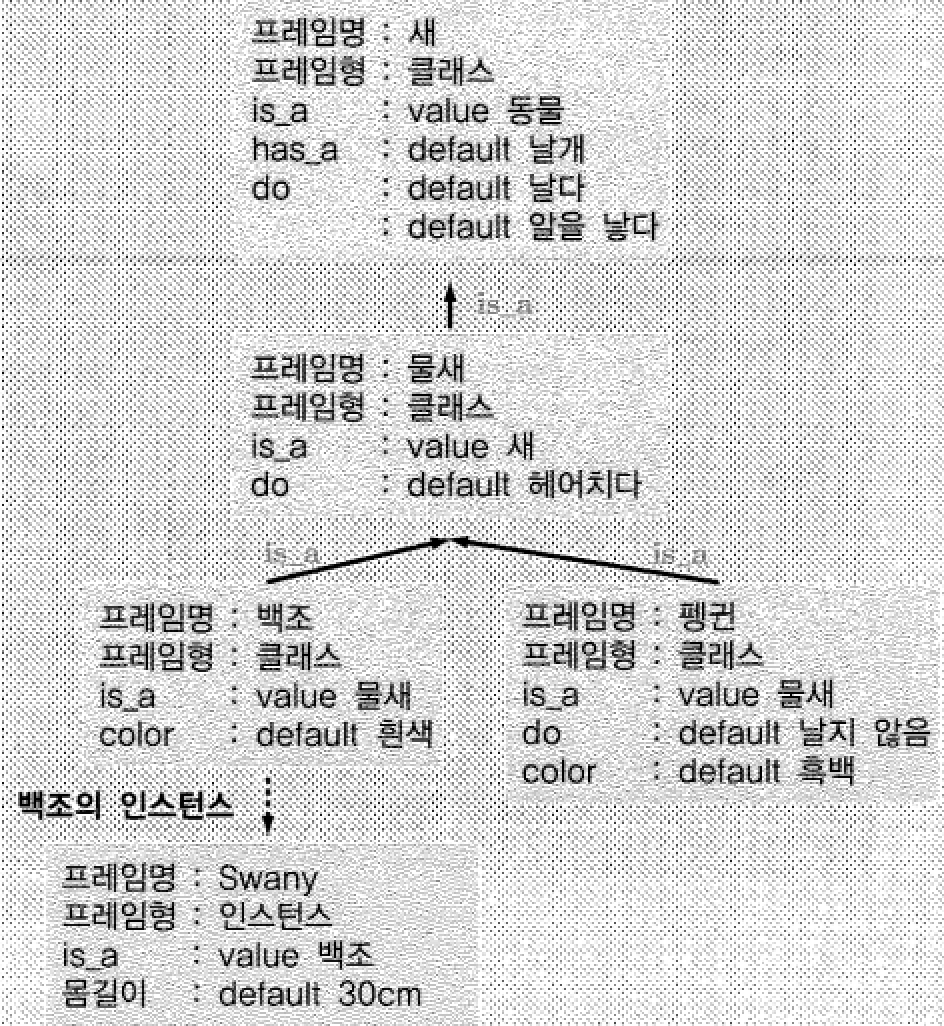
교실의 이미지로써, 교실은 방이므로 창, 문, 천정, 벽, 바닥 등이 있고, 반드시 칠판, 책상이 있는 장면이 떠오른다. 그 밖에 교실의 성질을 기술하는 정보로서 과목명, 시간, 교사, 학생 등이 있다. 이것들을 슬롯으로 각각의 프레임이 표현된다. 우선 전형적인 방의 지식이 클래스 레벨의 프레임으로 표현되고， 방의 하위 프레임으로 교실 프레임이 존재한다. 그리고 강의 시간이 정해지면 교설 예약을 수행하는 if\_ added 데몬과 필요에 따라 수강자 리스트를 가져오는if\_ needed 데몬이 사용되기도 한다. 아래 그림에 교실 프레임의 예를 나타낸다.



예) 새의 종류를 나타나는 프레임

새의 종류 프레임을 아래 그림에 보였다. 이것은 앞에서 다룬 의미 네트워크에서 표현된 새의 지식을 프레임으로 표현한 것이다.

의미 네트워크와 마찬가지로 상위 개념이 가지는 속성의 계승을 수행할 수 있다. “백조는 알을 낳는가?"라는 질문에 대해서는 “백조”의 상위개념인 “새”가 가지고 있는 속성 “알을 낳는다”가 계승되어 “낳는다”라고 답할 수가 있다.



* 의미 네트워크와 프레임

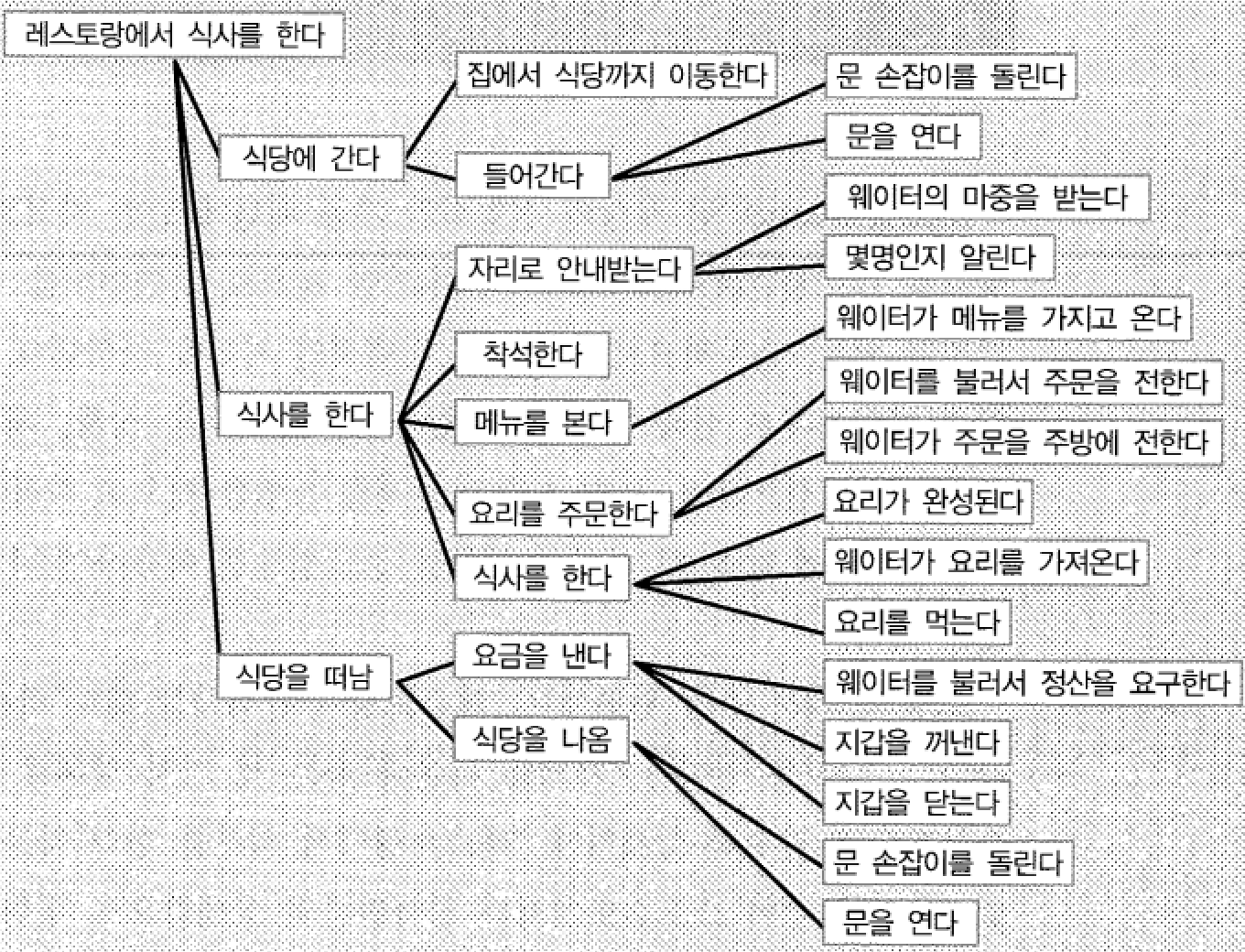
의미 네트워크와 프레임은 양쪽 모두 인간의 지식을 표현하는 방법이다. 같은 것을 대상으로 하기 때문에 양쪽의 표현력은 거의 같다고 봐도 좋다. 지식 표현에 대한 큰 방침의 차이에서는 의미 네트워크는 어떤 사상을 설명하는 지식표현 형식으로 고안되었다는 것에서 자연언어 문장의 의미내용을 표현하는 것에 많이 이용되고 있다. 한편 프레임은 인간의 기억 속에 존재하는 전형적인 장면을 기술하기 위한 지식표현으로 고안되었다는 것에서 인간의 장기기억 등 상식 적으로 다루어지는 지식을 미리 시스템에 넣어두기 위하여 많이 이용되고 있다.

양쪽 모두 계층 관계에 기반하는 성질의 계승 등의 기본적인 추론 메커니즘은 같고, 프레임은 거기에 절차적 지식인 데몬이 포함되어 있다는 점이 의미 네트워크와 다르다.

6.2.2 스크립트

스크립트(script)란 사상의 전형적인 기동 패턴에 관한 지식 표현이며, 1977년에 상크와 에이블슨(Abelson) 이 제안한 것으로 자연언어를 이해할 때 인간들이 보이는 순차적으로 수행하는 전형적인 지식을 나타낸다. 스크립트도 프레임과 마찬가지로 반복적으로 경험한 전형적인 장면, 상황, 개념 등을 나타내기 위한 지식표현으로써, 특히 시간적 순서를 도입한 것이 다르다고 볼 수 있다.

레스토랑에 갔을 때 사람의 행동을 스크립트를 이용하여 나타내면 아래 그림과 같이 된다.



그림의 가장 왼쪽을 보면 “레스토랑에서 식사를 한다”라는 행위를 “가게로 간다 “식사를 한다 “가게를 나온다”라는 세 가지 행위가 기술되고 있으며, 오른쪽으로 갈수록 상세하게 “레스토랑에서 식사를 한다”라는 행위를 기술하고 있다. 이와 같이 스크립트에서는 얼마만큼의 상세함으로 행위의 절차를 기술할 것인지 명확한 기준이 없고 필요에 따른 각각의 판단에 따라 상세함을 결정한다.

그리고 이야기가 들어있는 문장을 컴퓨터에게 이해시키기 위해서 이해를 위한 배경지식을 스토리의 전형적인 전개 스크립트로서 부여한다. 거기에서 이탈 하는 것에 대해서는 개념 지식의 추론을 이용하여 문장에서 드러나 있지 않은 내용의 이해를 지원하는 것과 같은 상식적인 지식으로 이용할 수 있다.

이와 같이 상식적인 지식의 일부로 이용하기 위하여 스크립트는 인간이 가지고 있는 장기기억의 하나인 에피소드 기억 (episodic memory)의 지식표현 형식으로 정의된다. 상크는 스크립트 개념을 기반으로 인간의 지식에서의 스카마에 대해 더욱 깊이 고찰하여 1980 년에는 부분적인 기억 단위로서의 지식인 MO P(Memory Organization packet)를 제창하고 있다.

6.2.3 오브젝트 지향

오브젝트 지향(object oriented)이란 오브젝트(객체) 자체가 각각 처리절차를 구비 하고 있어서 자율적으로 동작하면서 다른 오브젝트와 협조하여 전체적인 처리(자율, 분산, 협조(autonomous, distributed, cooperative) 라고 부름)를 수행한다는 개념이다.

오브젝트의 동작은 그것의 추상화된 표현인 클래스 (class)에 의해 표현된다. 하나의 클래스는 데이터와 그것에 관한 조작을 하나의 묶음으로 하는 캡슐화(encapsulation)로 표현된다. 따라서 각 오브젝트가 가지고 있는 데이터에 대한 처리는 거기에 정의되어 있는 조작(메소드 method)을 통해서만 수행될 수 있도록 하여 외부로부터 정보 은닉 (information hiding) 이 되도록 정의된다. 이러한 특정을 통하여 각 오브젝트의 처리능력 또는 자율성이 표현된다. 클래스는 상위-하위의 계층 관계를 형성하여 계층 간에 메소드의 계승이 이루어진다. 오브젝트는 클래스를 예시화(instantiation)함으로써 생성된다.

6.2.3.1 오브젝트 지향과 프레임

앞서 보인 것처럼 오브젝트 지향에 의한 표현은 프레임에 의한 표현과 매우 유사하다. 기본적인 차이점으로는 오브젝트 지향은 소프트웨어 개발에서 프로 그램의 모률성이나 개발 효율성을 높이기 위해 고안된 것에 반해, 프레임은 이미지로 얻어지는 인간의 개념을 대상으로 한 지식 표현이라는 것이다. 또한 프레임에는 데몬이라는 계산 처리의 메커니즘이나 디폴트값 등이 포함되지만, 오브젝트 지향에는 그것에 상당하는 것이 없다. 한편 오브젝트 지향은 정보 은닉의 기능이 있지만, 프레임은 그러한 기능을 가지고 있지 않다.

6.X 문제

6.3 프로덕션 시스템

6.3.1 프로덕션 시스템의 기본 구성

6.3.2 프로덕션 시스템의 동작 확인

6.3.3 전방향 추론과 역방향 추론

6.3 프로덕션 시스템

초기 인공지능 연구에서는 추론이나 탐색에 의한 문제-해결을 중심으로 연구가 진행되어 왔지만, 보다 실제적인 문제에서는 그것들의 방법을 적용시켰을 때 조합의 폭발이 발생하는 등 방법의 한계를 깨닫게 되었다. 이와 같은 배경으로부터 1970년대에 들어와서 우리의 경험에서 얻어지는 지식을 문제-해결에 활용 하는 연구가 증가해 왔다. 이 흐름이 지식공학(knowledge engineering) 이 라는 연구 분야를 탄생시키게 되었다.

이 장에서는 인공지능 기술에서 대표적인 지식표현 중 하나인 프로덕션 규칙 (production rule)과 그것을 이용한 문제 해결 시스템인 프로덕션 시스템 (production system) 에 대하여 설명한다.

프로덕션 시스템은 우리의 일상경험에서 얻은 문제-해결에 필요한 절차적인 지식

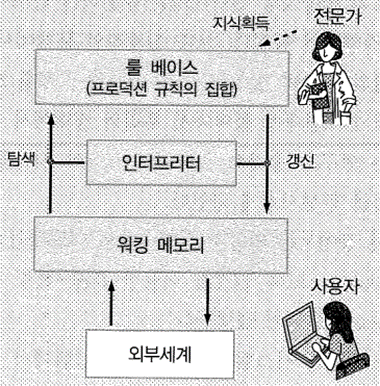
“만약 ~이라면, ~한다”(if 조건 then 행동)

에 의해 표현되는 지식을 가지고 그것을 운용함으로써 문제를 해결하는 시스템이며， 이 기술은 그 후에 전문가 시스템 (expert system) 의 본격적인 연구에 기반이 되었다.

감염증 진단을 수행하는 MYCIN이나 화학자가 수행하는 것과 같은 과정으로 미지의 유기화합물을 규명하는 DENDRAL을 시작하여 많은 전문가시스템에 도입 되어 인공지능 시스템의 중요한 기반기술 중 하나가 되었다.

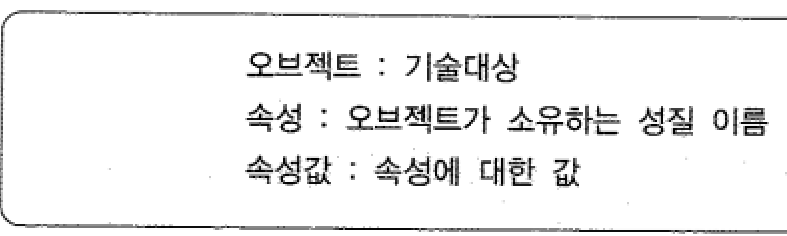
6.3.1 프로덕션 시스템의 기본 구성

프로덕션 시스템의 기본 구성은 워킹 메모리(작업 기억), 룰 베이스, 인터프리터의 세 가지로 되어 있다. 아래 에 프로덕션 시스템의 기본 구성을 보였다.



6.3.1.1 워킹 메모리

워킹 메모리 (working memory)는 사용자로부터 받은 문제를 일시적으로 보존하고, 이 문제를 해결하기 위하여 룰 베이스에 액세스하여 문제-해결을 위한 지식을 적용하는 것을 반복하면서 중간결과를 기억하고 최종적인 해를 발견한다.

워킹 메모리 상에서 표현되는 문제를 팩트(fact)라고 한다. 팩트의 데이터 형식은 오브젝트· 속성· 속성값 (Object-Attribute-Value)의 세 종류의 데이터로 되어 있으며 이것을 OAV 형식 (OAV form) 이라고 한다.

OAV 형식으로 표현되는 팩트의 일반형은 다음과 같다.

(오브젝트 속성1: 속성값1 … 속성n: 속성값n)

위의 일반형에 따라 기술대상이 “대학”이고 속성으로 이름에 “한국”, 종류에 “여자대학” 장소에 “도심”을 가지는 팩트의 예를 다음에 나타낸다.

(대학 이름: 한국 종류: 여자대학 장소: 도심)

6.3.1.2 룰 베이스

룰 베이스 (rule base)는 절차적 지식 표현인 프로덕션 규칙의 집합으로 이루어져 있다.

일반적으로 프로덕션 규칙은 다음의 형식으로 표현된다.

**if** C 1, C 2 , C 3, …, C n **then** A1, A2, A3, …, An

여기서 C 1, C 2, C 3, …, C n 은 조건부(LHS : 1eft-Hand-Side)라고 하며, 하나 이상의 조건의 조합에 의해 구성되어 있다. 문제-해결 작업에서 워킹 메모리 안에 존재하는 팩트와 프로덕션 규칙의 조건부가 일치할 때 행동부 (RHS: Right-Hand-Side) A1, A2, A3, …, An 을 실행한다(행동부는 일반적으로 지식의 유지관리 면에서도 하나의 행동으로 구성되는 경우가 많다). 각 조건 C i 는 팩트와 같은 데이터 형식을 취하여,

Ci = (오브젝트 속성1속성값1 … 속성n: 속성값n}

와 같은 OAV 형식으로 표현된다.

위 식의 조건부에서 각 조건C 1, C 2, C 3, …, C n이 동시에 참이 되었을 때， 즉 AND 조건식 전체

C1∧C2∧C3∧…∧ C n

이 참이 되었을 때 “if 조건 then 행동”의 규칙은 실행가능이 된다.

프로덕션 규칙의 조건부는 구성하는 명제와 논리연산자에 따라 다양한 조건을 표현하는 것이 가능하다. 그 전형적인 패턴을 다음에 나타낸다.

* 프로덕션 규칙의 조건부 패턴
  + 조건부 AND형:

조건부를 구성하는 명제C 1, C 2, C 3,…, C n전부가 성립하는 조건 하에서 행동부 명제 A1이 성립할 때, 다음과 같이 조건부 명제를∧(AND)로 연결한 프로덕션 규칙 이 만들어진다.

C1∧C2∧C3∧…∧ C n→A1

* + 조건부 OR형: 조건부를 구성하는 명제C1, C 2, C 3,…, Cn중 어떤 것이 성립하는 조건 하에서 행동부 명제 A1이 성립할 때, 다음과 같이 조건부 명제를 ∨(OR)로 연결한 프로덕션 규칙이 만들어진다.

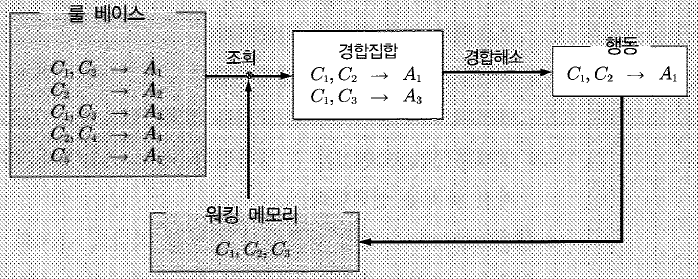
C1∨C2∨C3∨…∨ C n→A1

* + 조건부 AND/OR복합형: 조건부 명제를∧와∨를 섞어서 표현하는 것도 가능하다. 아래 식이 발화 되기 위해서 조건부의 명제(C1∧C2) 혹은C3가 성립하면 된다.

(C1∧C2)∨C3→A1

6.3.1.3 인터프리터

인터프리터(interpreter)는 워킹 메모리에 주어진 팩트 형식의 문제에 대해 프로덕션 규칙(이하, 규칙)의 조건에 맞는 것을 검색(조건 대조)하고, 복수의 규칙에 조건이 일치하였을 경우에 가장 적절한 규칙을 선택(경합 해소)하고, 규칙을 실행(행동)하는 단계를 문제-해결이 될 때까지 반복한다. 인터프리터에 의한 이 동작을 인식-행동 사이클(recognition–act cycle)이라 하고, 아래 그림에 그 행동의 개요를 나타낸다.



다음에 조건 대조, 경합 해소, 행동에 대하여 상세히 나타낸다.

* + 조건 대조(matching):

워킹 메모리의 팩트와 룰 베이스에 있는 프로덕션 규칙의 조건부의 데이터 형식을 대조하여, 팩트와 프로덕션 규칙의 조건부가 일치하는 규칙을 찾아내는 작업을 가리킨다.

* + 경합 해소(conflict resolution):

조건 대조에서 유일한 규칙이 선택되는 것이 이상적이지만, 통상적으로 복수의 규칙이 선택된다. 팩트와 조건이 일치하는 규칙의 집합을 경합집합(conflict set)이라 하고 경합집합 중에서 가장 적절한 규칙을 선택해야만 한다. 이를 위한 작업을 경합 해소라고 한다. 경합 해소는 문제-해결 능력을 크게 좌우하는 작업이며 경합 해소 전략(conflict resolution strategy)이라는 방법들을 아래에 보였다.

* + - 적용제한(rule application limitation): 무한루프가 발생하는 것을 방지하기 위하여 이전에 선택되었던 규칙의 사용을 제한한다.
    - 첫매치(first match): 룰 베이스 내에서 최초로 매치된 규칙을 선택한다.
    - 규칙우선순위 (rule priority): 각 규칙에 미리 우선순위를 부여해 두고 우선순위가 높은 규칙을 선택한다.
    - 최신우선 (recency): 경합집합의 규칙 중에서 가장 최근에 워킹 메모리에서의 조작이 이루어진 규칙을 선택한다.
    - 상세 우선(specificity): 경합집합 내의 규칙에서 가장 상세한 조건부를 가진 규칙을 선택한다.
  + 행동(action):

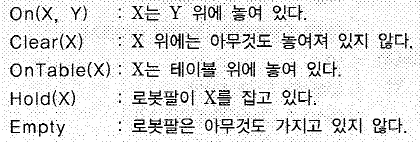
경합 해소에서 선택된 하나의 규칙의 행동부를 실행한다. 이것을 ‘규칙을 발화(fire)한다’고 한다. 행동부는 인터프리터의 인식-행동 사이클을 유지하는 작업인 워킹 메모리 내의 팩트 갱신, 추가, 제거를 수행하고, 외부환경과의 데이터 입출력 등도 수행한다. 워킹 메모리 내에서 팩트의 조작에 관한 행동부 A1의 예를 아래에 기술하였다.

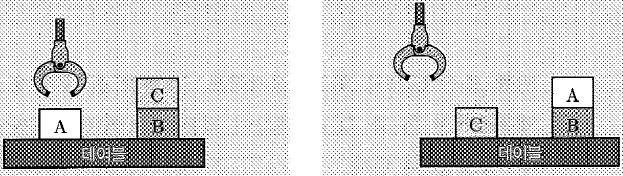
* (make (오브젝트 속성1: 속성값1 …속성n: 속성값n)): 워킹 메모리 안에 팩트를 삽입 한다.
* (modify i속성j: 속성값j): 프로덕션 규칙의 조건부에서 i번째 조건 Ci와 일치하는 워킹 메모리의 팩트의 “속성j”의 값을 새로운 “속성값j”로 수정한다.
* (add i속성j : 속성값j): 프로덕션 규칙의 조건부에서 i번째 조건 Ci와 일치하는 워킹 메모리의 팩트에 “속성j : 속성값j”을 추가한다.
* (delete i속성j: 속성값J): 프로덕션 규칙의 조건부에서 i번째 조건 Ci와 일치하는 워킹 메모리의 팩트에 “속성j : 속성값j”을 제거한다.

6.3.2 프로덕션 시스템의 동작확인

간단한 프로덕션 시스템을 설정하여 그 동작을 확인해보자.

아래 그림과 같이 테이블 위에 놓여진 나무 블록 A, B, C가 있다. 하나의 손을 가진 로봇팔로 좌측 그림의 블록을 이동시켜 우측에 주어진 그림과 같이 블록의 배치를 변경하고 싶다. 단, 테이블 위에서의 블록위치는 상관없고, 작업영역은 테이블 위로 한정한다. 블록 쌓기의 상태를 기술하는 술어를 다음과 같이 정의하자.





풀이)

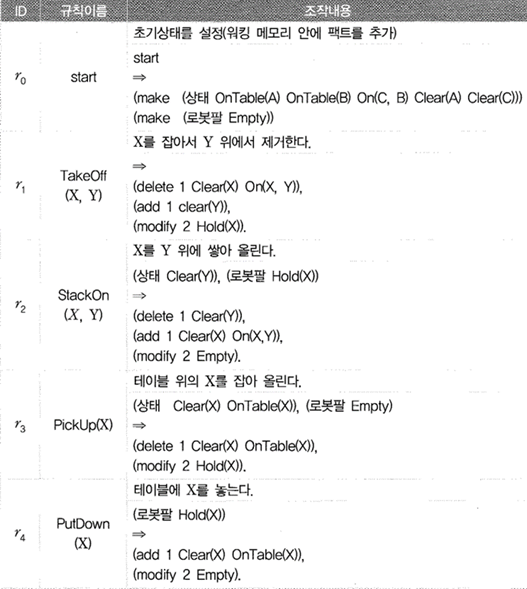
* 초기 상태(좌)의 로봇팔의 상태는 다음과 같다.

(상태 onTable(A) On Table(B) On (C , B) Clear(A) Clear(C))

(로봇팔 Empty)

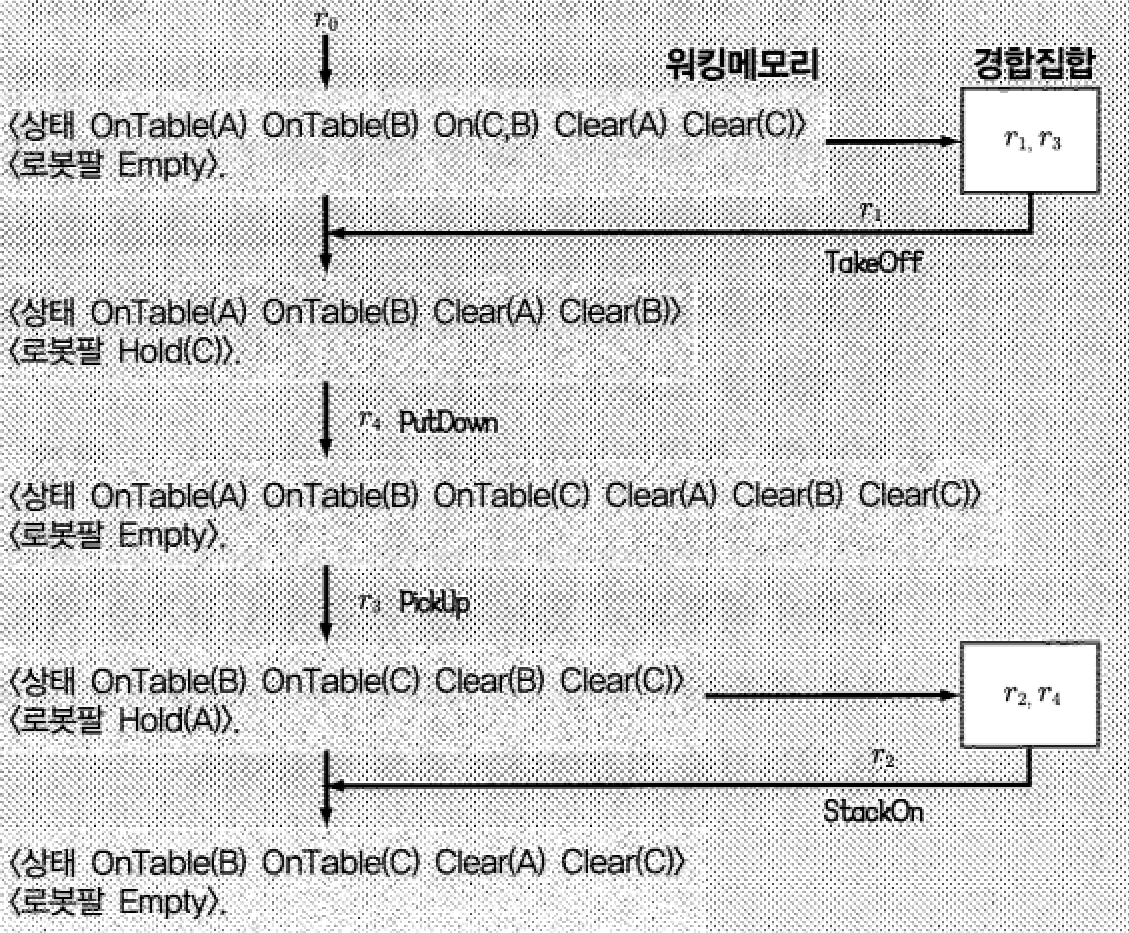
* 아래와 같이 프로덕션 규칙을 설정한다.

이 때 오브젝트 “상태”를 첫번째 조건 C 1 으로 하고， 오브젝트 “로봇팔”을 두번째 조건 C 2 로 한다. 따라서 행동부에서 “1" 과 ‘'2" 를 대상이 되는 오브젝트를 나타낸다고 하면 각각 “상태” 와 “로봇팔”이 된다.



* 아래 그림은 인식-행동 사이클에 의한 워킹 메모리의 변화를 나타낸다.

이 때 경합집합이 존재하면 경합 해소 전략에 의해 적용되는 규칙이 선택되는 것으로 한다. 최초의 경합 해소에는 첫매치에 의한 경합 해소가 이루어지고 있다고 하고, 다음 경합 해소에는 적용 제한에 의해 이미 적용된 규칙의 적용은 피한다고 하자.



6.3.3 전방향 추론과 역방향 추론

프로덕션 규칙을 사용하는 추론 방법을 크게 나누면 전방향 추론과 역방향 추론, 두 가지가 있다. 각각에 대해 설명하기로 한다.

* 전 방향 추론(forward reasoning):

데이터 구동형 추론 (data-driven reasoning) 이라고도 한다. 특정한 데이터나 사실에 대하여 프로덕션 규칙을 적용하고, 얻어진 결과에 대해 다시 프로덕션 규칙을 반복해서 적용해 나감으로써 최종적인 결론을 이끌어내는 추론 방법이다. 사실을 제공하여 결론을 도출하는 탑다운 처리를 적용하고 있다. 한편 규칙 적용이 부적절하면 결론을 도출하기 어렵게 된다.

* 역방향 추론 (backward reasoning):

목표 구동형 추론 (goal-driven reasoning) 이라고도 한다. 목표가 주어져서 그 목표를 달성하기 위해 필요한 조건이 만족되고 있는지를 프로덕션 규칙을 역으로 추론(행동부로부터 조건부를 추론)함으로써 그 목표가 주어져 있는 지식(프로덕션 규칙)으로 입증될 수 있는지를 이끌어내는 추론 방식이다.

6.4 문제 9장 프로덕션 시스템