객체지향 구조를 이용한 개념 지식 베이스 구축 도구의 개발

진 정 수,° 박 종 희, 조 유 제
경북대학교 전자공학과

Development of Tool for Making Conceptural Knowledge-Base Using Object-Oriented Structure

Jung-Su Jin,° Jong-Hee Park, You-Ze Cho
Dept. of Electronics, Kyungpook National University

요 약

동적인 지식을 처리하기 위하여 객체지향방법을 이용하여 클래스와 객체들을 생성시키고 객체지향 구조에서 제공하는 분류학적 구조와 조직 구조이외에 사용자가 클래스 계층구조 자체를 정의할 수 있게 함으로써 생성된 클래스와 객체들간의 관계를 효율적으로 표현할 수 있게 하는 지식획득 시스템을 제안한다. 규칙 베이스에 내장된 여러 가지 규칙들을 사용하여 사용자의 지식 베이스 구축시 사용자가 보다 쉽게 지식 베이스를 구축할 수 있도록 도와준다.

I. 서 론

지식베이스[1,2]는 대상 영역의 지식을 모형화한 것으로 볼 수 있다. 이러한 지식베이스 구축에 있어서 중요한 것은 지식을 얼마나 효율적으로 획득할 것인가와 획득된 지식을 어떻게 효율적으로 표현할 것인가 하는 것이다.

지식 표현[5,6,7]은 영어와 같은 자연어를 컴퓨터가 이해할 수 있도록 프로그램화하려는 필요에 의해 생겨났다. 지식 표현 방법[3,4]은 인간의 일상언어와 컴퓨터언어와의 표현 구조 사이의 어느 중간에서 타협점을 결정하게 된다. 만약 인간의 언어, 즉 자연언어로만 표현한다면 아직 컴퓨터에 의한 자연어의 처리가 완벽하지 못하므로 구현이 불가능하고, 반면 컴퓨터의 입장에서 지식을 컴퓨터언어의 알고리즘과 자료구조로만 표현하면 그 또한 인간이 이해하기가 어렵기 때문이다.

지식 획득[1,8]이란 여러 형태의 지식 원천으로부터 필요한 지식을 추출하여 이를 구조적으로 조직화하는 과정을 말한다.

본 논문에서 지식획득 방법으로 사용하는 객체지향 방법이란 대상 영역에서의 특정 객체에 관한 지식을 파헤침으로써 구체적인 객체들의 공통적인 속성을 추출하여 집단화함으로써 객체들의 클래스를 형성시키는 방법이다. 우리는 이러한 객체지향 방법을 사용함으로써 다양한 지식을 표현하고 획득할 수 있다.

지금까지 발표된 여러 가지 객체지향 시스템들을 살펴보면 기본적인 객체지향 지식구조인 분류학적 구조와 구성요소들 사이의 조직 구조[9,10]외의 관련성들은 구조자체에 표현할 수 있는 요소가 주어져 있지 않았다[11,12]. 그리고 상황에 따라 변하는 동적인 지식을 적절하게 표현할 수가 없었다[7]

본 논문에서는 동적인 지식을 처리하기 위하여 객체지향 방법[9,10]을 이용하여 클래스와 객체들을 생성시키고 객체지향 구조에서 제공하는 분류학적 구조와 조직 구조이외에 사용자가 클래스 계층구조 자체를 정의할 수 있게 함으로써 생성된 클래스와 객체들간의 관계를 효율적으로 표현할 수 있게 하는

지식획득 시스템을 제안한다. 본 논문에서 제안한 지식획득 시스템은 규칙 베이스에 내장된 여러 가지 규칙들을 사용하여 사용자의 지식 베이스 구축시 사용자가 보다 쉽게 지식 베이스를 구축할 수 있도록 도와준다. 또한 GUI를 사용하여 구현함으로써 사용자의 직관에 의한 사용이 가능하도록 하였다.

II. 클래스의 개념 분류

본 논문에서는 지식획득의 수단이 되는 클래스로서 객체형 개념 클래스와 관계형 개념 클래스를 두고 객체형 개념 클래스와 관계형 개념 클래스의 생성을 위하여 동작형 개념 클래스, 상태형 개념 클래스, 관계 상태형 개념 클래스, 객체 지시형 개념 클래스, 객체 선행형 개념 클래스의 다섯 가지 개념 클래스를 두어 모두 일곱 가지 개념 클래스로 분류하였다[7,9].

2.1 객체형 개념 클래스

객체형 개념 클래스는 그 자신이 어떠한 주체(논리적이든지 물리적이든지)가 되는 것, 행위의 대상이 되고 어떠한 기능을 할 수 있는 것, 그리고 어떤 하나의 현상이나 논리적인 개념을 나타내는 개념 범주에 속하는 객체들의 클래스이다. 지식 베이스 구축자는 특정 객체에 관한 지식을 파헤쳐서 구체적인 객체들의 공통적인 속성을 추출하여 집단화함으로써 클래스를 형성할 수 있다. 그리고 이러한 클래스를 분석하여 확연히 구분되는 속성들을 갖는 하위 클래스들로 나누게 된다.

객체형 개념 클래스의 클래스 계층구조는 지식 베이스 구축자에 따라 조금씩 상이한 구조를 가질 수 있다[13,14]. 객체형 개념 클래스는 객체의 개념에 따라크게 physical object, logical concept, 그리고 phenomenon의 세 가지 최상위 슈퍼클래스로 구분하였다. Physical Object 클래스에는 물리적인 형태를 가지고 있는 객체들, 예를 들면 의자, 나무, 꽃과 같은 객체,이 속한다. Logical Concept 클래스에는 인간의 관념에 의해 생성된 개념들, 예를 들면 학문, 정치체계와 같은 객체들,이 속한다. Pheno- monon 클래스에는 자연현상들을 나타내는 객체들, 예를 들면 날씨나 계절,이 속한다. 객체의 특수한 경우로서 대표객체가 있다. 대표객체는 클래스에 속하는 객체가 하나만 존재하고 클래스의 이름과 객체의 이름이 같은 객체를 가리킨다. 즉 객체가 클래스 자체를 나타낸다. 대표객체는 클래스와 객체와의 관계가 아닌 슈퍼클래스와 서브클래스의 분류학적 구조 관계를 가능하게 함으로써 클래스와 객체 생성에 유연성을 부여한다.

객체형 개념 클래스에 속하는 객체들은 클래스 계층구조로서 슈퍼클래스와 서브클래스 구조인 분류학적 구조와 속성, 도메인 구조인 조직 구조외에 사용자 정의 구조를 가진다. 사용자 정의 구조는 클래스들 사이에 자주 나타나는 관계 구조들을 사용자 정의 관계 개념 클래스에 정의하고 정의된 관계들을 이용하여 분류학적 구조와 조직 구조 외에 사용자가 새롭게 정의할 수 있는 클래스 계층구조이다. 자주 나타나는 속성들을 사용자 정의 관계 개념 클래스에 정의함으로써 객체에 대한 정보를 더욱 효율적으로 나타낼 수 있다.

객체형 개념 클래스내의 각각의 클래스에는 단지 스키마만 정의되어 있고 객체에 실제의 값이 저장되어 있다. 객체는 클래스가 가진 속성과 메소드외에 특성과 상태를 가진다. 객체에만 특성과 상태를 정의한 것은 데이터의 상속성과 은폐성을 실현하기 위해서이다.

속성과 메소드가 분류학적 구조에서 슈퍼클래스에서 서브클래스로 상속이 되는 공용 데이터인 반면에 객체에만 정의되어 있는 특성과 상태는 서브클래스로 상속이 되지 않는 사적 데이터이다.

2.2 관계형 개념 클래스

관계형 개념 클래스는 사용자 정의 관계형 개념 클래스와 객체 관계형 개념 클래스의 두 가지 클래스로 분류한다.

사용자 정의 관계형 개념 클래스는 객체들 사이에 자주 발생하는 관계들만을 사용자가 정의하여 모아놓은 개념 클래스이다. 기본적인 객체 지향 구조에서는 분류학적 구조와 조직 구조의 두 가지 클래스 계층구조만 허용된다. 하지만 이 두 가지 클래스 계층구조 이외에 사용자가 정의할 수 있는 클래스 계층구조를 허용함으로써 클래스 사이의 관계 정보를 더욱 효율적으로 나타낼 수 있다.

객체 관계형 개념 클래스는 지식 표현의 정확성을 위해서 클래스와 객체 사이의 관계만을 모아놓은 클래스이다. 지식은 미묘한 문제에 있어서 정확하게 표현하기가 어렵다. 또한 지식은 정적인 것이 아니라 동적인 것이므로 처리가 상당히 어렵다. 따라서 지식 표현의 정확성을 위해서는 주어진 문제에 있어서 어떤 대상물 사이의 상관관계를 정확하게 표시할 수 있어야 한다[7]. 대상물 사이의 상관관계를 표시하기 위해서 클래스와 객체 사이의 관계를 설정해 주며 지식 표시의 정확성을 위해서 주체와 대상사이의 관

계가 변하는가 변하지 않는가에 따라 순간 관계와 지속 관계로 나눈다. 순간 관계란 클래스와 객체들 사이의 관계가 변할 수 있는 관계를 말하고 지속 관계란 클래스와 객체들 사이의 관계가 변하지 않는 관계를 말한다. 기존의 속성과 도메인 구조에서는 클래스내의 모든 객체에 동일하게 속성과 도메인이 적용되어야 하지만 객체 관계형 개념 클래스는 클래스에 소속된 객체와는 무관하게 다른 클래스와의 관계를 설정할 수 있을 뿐만 아니라 클래스내의 특정한 객체에만 적용시킬 수 있다.

관계 설정시 대상물 사이의 상관관계를 나타내기 위해서 주체 객체와 대상 객체, 그리고 두 객체 사이의 관계명을 정의해 준다. 여기서 주체 객체는 관계의 중심이 되는 객체이고 대상 객체는 주체 객체와 관계를 맺고 있는 객체를 말한다.

지식 표현의 정확성에 관련되는 분야로서 중요한 것 중 하나는 지식표현의 정도(Level)을 선택해야 한다는 것이다. 지식표현의 정도란 주어진 문제에 대하여 어느 정도 상세히, 정확하게 표현할 것인가 하는 것을 말한다. 지식표현의 정확성 측면에서 볼 때는 낮은 차원(low-level), 즉 덜 세분화되고 덜 정확하게 표현하는 방법을 사용할 수 있고, 높은 차원(high-level), 즉 더욱 세분화되고, 더욱 정확하게 표현하는 방법을 사용할 수도 있다[7].

간단한 예를 들어 보면, "I love Ms.Moon very much."를 낮은 차원 및 높은 차원의 지식형태로 표현해 보면 다음과 같다.

- 낮은 차원(low-level) 표현 : love(I, Ms.Moon).
-높은 차원(high-level) 표현 : love(I, Ms.Moon, very-much).

위의 예와 같이 정확하고 세부적인 지식표현이 가능한 높은 차원의 표현을 가능하게 하기 위하여 객체의 관계 설정에 있어서 관계 레벨(Relation Level)을 둔다. 관계 레벨은 최종 주체와 관계를 맺고 있는 대상 클래스나 객체들 사이에 몇 단계의 관계가 존재하는 가를 나타낸다.

2.3 동작형 개념 클래스

동작형 개념 클래스는 어떠한 객체의 동작을 나타내는 개념 범주에 속하는 개념 객체들의 클래스이다. 여기에서 개념 객체란 것은 속성만 가지고 있고 메소드는 가지고 있지 않지만 객체 자체가 어떠한 메소드를 표현하는 객체를 말한다. 동작형 개념 동작형 개념 클래스의 개념 객체는 객체형 개념 클래스의 동작 이름으로 사용되기도 하고 객체의 관계를 맺어 줄 때에도 사용된다.

2.4 상태형 개념 클래스

상태형 개념 클래스는 객체형 개념 클래스내 객체들의 성질이나 상태를 나타내는 개념 객체들을 모아 놓은 개념 클래스이다. 상태형 개념 클래스는 객체형 개념 클래스의 상태 이름이 되기도 하고 동작형 개념 클래스처럼 객체간의 관계를 나타내기도 한다. 하지만 동작형 개념 클래스와 달리 단순히 객체간의 관계만 맺어주는 것이 아니라 객체간의 비교를 가능하게 한다.

2.5 관계 상태형 개념 클래스

관계 상태형 개념 클래스는 클래스와 객체들간에 동작형 개념 객체를 사용하여 관계를 설정할 때 관계 정보를 더욱 자세하게 나타내는 역할을 한다.

2.6 객체 지시형 개념 클래스

객체 지시형 개념 클래스는 객체형 개념 클래스를 가리키는 여러 가지 가상객체들을 모아둔 개념 클래스이다. 여기에서 가상 객체란 실제로 속성과 메소드는 정의되어 있지 않지만 가상 객체가 가리키는 실제 객체의 속성과 메소드를 나타내고 있는 객체이다.

예를 들어 객체 지시형 개념 클래스에 That 가상 객체, 객체형 개념 클래스에 Rose 객체, 동작형 개념 클래스에 Is 객체가 존재할 때 Rose와 That의 관계 설정이 Is로 되었을 경우를 생각해 보자.

That --- <Is> --- Rose

이 경우에 That 가상 객체는 Rose 객체를 가리키는 가상 객체이다.

2.7 객체 선행형 개념 클래스

객체 선행형 개념 클래스는 클래스와 객체간의 관계 설정시에 관계 정보를 더욱 상세히 나타내는 클래스이다. 하지만 관계 상태형 개념 클래스가 동작형 개념 객체와 결합하여 관계 정보를 상세히 나타내는 것과는 달리 객체 선행형 개념 클래스는 동작형 개념 클래스외의 개념 클래스와도 결합하여 관계 정보를 나타낼 수 있다.

III. 지식 베이스 시스템의 구성

본 절에서는 개념에 따라 분류된 객체들을 개념

지식 베이스에 접가하고 이 객체들 사이의 의미론적 관계를 설정해 주는 시스템 구조를 제안한다

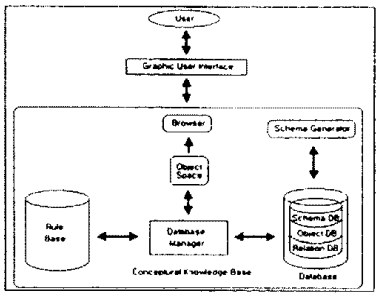

그림 1. 개념 지식 베이스 시스템 구성

개념 지식 베이스 시스템은 그림 1과 같이 모두 여섯 개의 부분으로 구성되어 있다.

3.1 데이터베이스(Database)

모든 클래스와 객체들, 그리고 이들간의 관계(Relation) 데이터를 저장하고 있는 스키마 데이터베이스(Schema DB), 객체 베이스(Object Base), 관계 데이터베이스(Relation DB)의 세 가지 데이터베이스가 있다.

3.2 스키마 발생기

스키마 발생기는 객체형 개념 클래스 내에 사용자가 새로운 클래스를 생성할 때나 규칙 베이스에 내장된 규칙의 적용을 받아 클래스 계층구조 재조정시 시스템에서 새로운 클래스를 만들 때 슈퍼클래스로부터 속성과 동작, 그리고 사용자 정의 관계 구조를 새로운 클래스에 상속시킨다.

사용자나 시스템이 새로운 클래스를 생성시킬 때 사용자 입력의 유무에 따라 디폴트 클래스와 사용자 정의 클래스의 두 가지 클래스를 만들 수 있다. 사용자에 의한 입력이 없이 순전히 상속에 의해서만 만들어진 클래스를 디폴트 클래스라 하고 상속에 의해서 뿐만 아니라 사용자가 직접 속성, 동작, 사용자 정의 관계를 입력한 클래스를 사용자 정의 클래스라 한다.

3.3 규칙 베이스

규칙 베이스는 사용자가 새로운 클래스를 생성시킬 때 각각의 개념 클래스에 적용되는 여러 가지 규칙들을 저장하고 있다. 시스템은 규칙 베이스에 저장되어있는 여러 가지 규칙들을 이용해서 입력의 적정 여부를 판단하거나, 클래스 계층구조를 재조직화 하는 등의 작업을 수행한다. 본 시스템에서 제공하는 규칙은 여섯 가지가 있다.

첫 번째 규칙인 최적 입력 확인 규칙은 객체형 개념 클래스의 생성시에 적용되는 규칙이다. 사용자가 객체형 개념 클래스에 새로운 클래스를 생성시키면 입력 단계에서 사용자가 생성한 클래스 계층구조에 오류가 발생했는지를 검사한다.

두 번째 규칙인 클래스 계층구조 자기 재조직화 규칙은 기존의 클래스 계층구조와 새로 생성되는 클래스의 계층구조를 비교하여 클래스 계층구조를 최적화 시킨다. 최적 입력 확인 규칙과 클래스 계층구조 자기 재조직화 규칙은 객체형 개념 클래스 생성시 적용되는 규칙이다.

세 번째 규칙인 원형 검색 규칙은 동작형 개념 클래스 생성시 사용자가 동작형 개념 클래스의 시제형을 입력했을 때 원형 이외의 시제형에 대하여 시스템에서 원형을 찾아주는 규칙이다.

네 번째 규칙은 관계 상태형 검색 규칙으로 상태형 개념 클래스 생성시 연관되는 관계 상태형 클래스를 찾아주고, 만약 연관된 클래스가 없을 경우 새로 생성시켜주는 규칙이다.

다섯 번째 규칙은 비교형 검색 규칙으로 상태형 개념 클래스의 비교급과 최상급을 생성시켜 주는 규칙이다.

여섯 번째 규칙은 상태형 검색 규칙으로 관계 상태형 개념 클래스 생성시 연관되는 상태형 클래스를 찾아주고, 만약 연관된 클래스가 없을 경우 새로 생성시켜주는 규칙이다.

일곱 번째 규칙은 객체 지시형 문법 정보 확인 규칙으로 객체 지시형 개념 클래스 생성시 클래스의 문법적인 정보를 검색해 주는 규칙이다.

3.4 객체 공간(Object Space)

객체 공간 클래스와 객체들을 처리하는 환경이라고 할 수 있다. 모든 클래스와 객체들은 객체 공간에서 모든 처리가 행해진다. 클래스 테이블은 클래스를 처리하고 객체 테이블은 객체들을 처리한다.

3.5 데이터베이스 관리자

데이터베이스 관리자는 규칙 베이스, 데이터베이스, 객체 공간 사이에서 사용자의 작업과정을 처리 감독한다. 사용자가 어떤 데이터에 대한 요구를 하면

데이터베이스 관리자는 규칙 베이스, 데이터베이스, 객체 공간 사이에서 사용자의 작업과정을 처리 감독한다. 사용자가 어떤 데이타에 대한 요구를 하면 데이터베이스 관리자는 객체 클래스 테이블로 해당 클래스나 객체들을 불러들인다. 그리고 사용자의 요구에 따라 데이터를 처리한다.

3.6 브라우저(Browser)

사용자는 브라우저 기능을 통하여 클래스나 객체를 편리하게 생성시키고 관리할 수 있다. 사용자는 클래스의 스키마를 볼 때, 그리고 객체에 저장되어 있는 값들을 볼 때 이외에도 객체의 속성값과 특성값, 동작, 상태 입력 시에도 브라우저를 이용할 수 있다. 브라우저의 하위 메뉴로서 스키마 뷰, 객체 속성 뷰, 객체 특성 뷰, 객체 동작 뷰, 객체 상태 뷰, 그리고 객체 뷰가 있다.

IV. 결 론

지금까지 객체지향구조에 바탕을 둔 개념 지식 베이스 시스템의 구현에 관해서 설명을 하였다. 본 논문에서 구현한 개념 지식 베이스 시스템은 객체지향 구조에서 기본적으로 제공하는 기존의 분류학적 구조와 조직 구조외에 사용자가 클래스 계층구조를 직접 정의할 수 있는 사용자 정의 관계 구조를 도입하여 클래스 계층구조를 더욱 세분화하였으며 동적인 지식의 표현을 위하여 클래스와 객체들 사이의 상관관계를 나타낼 수 있는 객체형 관계를 정의하였다. 또한 관계 레벨을 이용하여 더욱 세분화되고 정확한 지식 표현을 가능하게 하였다. 규칙 베이스에 내장된 여러 가지 규칙들을 사용함으로써 시스템은 사용자의 개념 지식 베이스 구성을 돕는다. 특히 정형화된 클래스 계층구조를 사용자의 입력에 따라 자기 재조직화 할 수 있는 규칙들을 시스템에서 지원해 줌으로써 최적의 클래스 계층구조가 유지되도록 하였다.

본 논문에서 구현된 개념 지식 베이스는 윈도우 95 상에서 Visual C++를 사용하여 구현하였다. 따라서 모든 처리과정은 대화상자를 통하여 이루어진다. 사소한 입력상의 오류나 처리상의 오류가 발생한다고 할지라도 항상 사용자에게 경고 메시지를 발생시켜 사용자의 주의를 환기시키고 사용자에게 자신의 입력을 재확인하도록 유도한다. 만약 오류가 발생한다면 항상 오류의 원인을 사용자에게 제시함으로써 다른 실수의 재발을 방지하였다. 그리고 사용자가 다음 단계에 해야 할 작업에 관한 간단한 메시지를 보여줌으로써 작업의 진행 단계에 대한 이해를 돕고 있다.

한편 본 시스템에서 보완되어야 할 점은 질의어 처리문제와 객체의 상태 표현시 더욱 효과적인 방법의 개발, 그리고 에니메이션으로 표현할 수 없는 메소드의 표현 방법의 개발이라 할 수 있다. 사용자의 요구에 적절히 대응할 수 있는 질의어를 개발하고 객체의 메소드와 상태를 더욱 효과적으로 표현할 수 있다면 보다 나은 시스템이 될 수 있을 것이다.

참 고 문 헌

[1] Adrian A. Hopgood, " Knowledge-Based Systems for Engineers and Scientists," CRC Press, 1993.

[2] Jeffrey D. Ullman, "Principles of Database and Knowledge-Base Sysem," Computer Science Press, 1988.

[3] William A. Woods, "What's Important About Knowledge Representaion," IEEE COMPUTER, October. 1983.

[4] Gordon McCalla, Nick Cercone, "Approa- ches to Knowledge Representation," IEEE COMPUTER, October. 1983.

[5] 이재규, 최형림, 김현수, 서민수, 주석진, 지원철, "전문가 시스템," 법영사, 1996.

[6] 이윤배, "전문가 시스템, " 홍릉과학출판사, 1992.

[7] 김화수, 조용범, 최종욱, "전문가 시스템," 집문당, 1995.

[8] Sabrina Sestito, T. S. Dillon "Automated Knowledge Acquisition," Prentice Hall, 1994.

[9] Won Kim, "Object-Oriented Database :Definition and Research Directions," IEEE Transactions on Knowledge and Data Engineering VOL. 2, NO. 3, September. 1990.

[10] Bogdan Czeido, Christoph F. Eick and Makolm Taylor, "Integration Sets, Rules, and Data in an Object-Oriented Environment," IEEE, February. 1993.

[11] Kazuyuki Tsuda, Kensaku Yamamoto, Masahito Hirakawa, Minoru Tanaka, Tadao Ichikawa, "MORE : An Object-Oriented Data Modal with a Facility for Chaning Object Struc- tures," IEEE Transactions on Kno wledge and Data Engineering, VOL. 3, NO. 4, December. 19991.

[12] Paul Butterworth, Allen Otis, Jacob Stein, "Gemstone," Communications of the ACM,

VOL. 34, NO. 10, October, 1991.

[13] 박영자, "단어의 개념 정보를 이용한 한국어 문장 분석기에 대한 연구," 연세대학교 졸업논문, 1992.

[14] Carlo Batini, Stefano Ceri, Shamkant B.Navathe, "Conceptual Database Design," The Benjamin/Cummings Publishing Company, 1992.