HyKET에서 객체지향언어의 구현및 규칙기반 추론시스템의 설계에 관한 연구 *

송 은강, 김 진형
한국과학기술원, 전산학과

Object-Oriented Language and Rule System in HyKET :their design and implementation

Eunkang Song, Jin H. Kim
Dep. of Computer Science, KAIST.

요 약

HyKET는 Flavor 객체 지향언어를 기반으로한 프레임 시스템, 규칙기반 추론시스템, 사용자 인터페이스를 위한 그래픽 패키지, 윈도우 시스템, Common LISP 프로그래밍환경과 같은 인공지능 개발 기술들을 하나의 시스템 형태로 묶어놓은 복합형 인공지능 시스템이다 본 논문은 이 HyKET의 프레임 시스템을 설명하고 규칙기반 추론 시스템을 제안하였다. HyKET는 프레임 시스템으로 Flavor 언어를 기반으로 한다. 규칙기반 시스템은 Flavor언어와 결합하여 규칙 expressior에서 Flavor로 정의한 객체들을 사용할 수 있고 또한 규칙 수행중 객체들의 구성을 바꿀 수 있으며 순방향, 역방향 추론기능을 가지고 있다.

I 서 론

1980년대 들어서 각광을 받기 시작한 소프트웨어중의 하나가 인공지능용 소프트웨어이고 특히 전문가 시스템이 그렇다 이전의 문제 해결 시스템이 일반적이고 넓은 문제를 해결하려고 노력한 반면에 전문가 시스템은 전문가라는 단어가 의미하는 것처럼 보다 좁은 분야의 문제만을 해결하려고 한 것이 그 특징이다. DEC 컴퓨터를 구입하려는 손님의 요구를 분석하여 전체적인 컴퓨터 시스템을 구축해주는 *R1*을 비롯하여 몇몇의 시스템이 실험실과 현장에서 크게 성공하였으며 이러한 것이 자극이 되어 많은 기업과 학교에서 전문가 시스템을 개발하기 위해 보다 활발한 투자와 연구를 시작하게 되었다 현재 의학진단, 제정계획, 광물탐사, 법률문제, 고장진단, 조립계획 등 다양한 분야에 걸쳐서 상당한 성공을 거두고 있다[1]

전문가시스템은 기존의 프로그램 개발방법과는 다른 개발방법을 택하고 있다 기존의 개념으로서 프로그램을 한다는 것은 미리 알고있는 문제해결 절차나 알고리즘을 프로그램으로 바꾸는 것이다. 하지만 전문가시스템을 개발할 때에는 경험적 지식을 시스템내에 표현해야 하는데 이러한 개념들을 기존의 *PASCAL*이나 *C*와 같은 프로그래밍 언어를 이용하여 표현하는 것은 힘든 일이다. 더우기 전문가 자신도 어떻게 결론을 이끌어내는지에 대해 정확하게 알고있지 못한 경우에는 더욱 그렇다 따라서 전문가시스템의 개발은 여러번의 반복작업을 통하여 점점 향상된 시스템으로 변화시켜가는 개발방법을 요구하고 있다. 따라서 이와같은 개발기법들을 적용하기 위해 기존의 개발도구와는 다른 형태의 전문가시스템 개발도구가 필요하게 되었다 초창기의 전문가시스템 개발자들은 LISP과 같은 인공지능 프로그래밍 언어를 사용하였으며 이러한 노력의 결과 LISP 환경위에 전문가시스템을 용이하게 개발할 수 있는 특별한 도구들이 제작 되었다 이러한 개발도구들은 전문가의 지식을 편리하게 표현할 수 있고 표현된 지식을 이용하여 결론을 추론할 수 있는 기본적인 골격을 제공한다[2]

초기의 인공지능개발시스템들은 일차술어논리[3], 규칙(rule)[4], 프레임[5], 의미 네트워크, 스크립트(script)등과 같은 여러가지 지식표현 방법중 한가지만을 사용한 단순한 시스템이었다 그러나 인공지능시스템이 다루는 문제가 어느 한가지의 지식표현방법으로는 자연스럽게 여러 분야의 전문지식을 표현할 수 없다는 사실을 발견하게 되어 최근에는 다양한 지식표현방법을 제공하는 복합형 인공지능개발시스템이 많이 사용되고 있다 복합형 인공지능개발시스템은 여러가지의 지식표현방법을

* 이 논문은 1987년도 한국과학재단의 연구비에 의하여 연구되었음

사용자에게 제공함으로써 사용자가 전문지식을 자연스럽게 표현할 수 있도록 하였으며 이로 인하여 전문가시스템의 응용범위를 더욱 다양하고 복잡한 영역까지 확장할 수 있도록 하였다. 뿐만아니라 복합형 인공지능개발시스템은 윈도우시스템, 그래픽 시스템, 객체지향언어, LISP 개발도구 등으로 구성된 강력한 인공지능시스템 개발환경을 제공함으로써 인공지능 시스템을 개발하는데 드는 시간과 노력을 감소시켜 준다.

우리나라에서도 인공지능 연구가 활발해지고 있으며 그 투자도 늘어나고 있기 때문에 인공지능 개발 시스템의 수요가 증대될 것이다. 그러나 우리나라에서는 인공지능개발시스템에 대한 체계적인 연구가 부족했으며 다양한 지식표현과 그래픽, 윈도우 시스템등을 제공하는 인공지능개발시스템은 아직 개발된 것이 없다. 또한 외국산 시스템은 가격이 비싸서 구입하기에 많은 어려움이 있다. 따라서 국내에서도 강력한 인공지능시스템 개발환경을 갖춘 복합형 인공지능개발시스템이 필요하다.

이러한 요구를 만족시키고 인공지능 산업을 활성화 시키기 위하여 현재 KAIST에서 개발되고 있는 시스템이 HyKET이다[6]. 본 논문에서 다루고있는 것은 HyKET의 프레임 시스템과 규칙기반 시스템(Rule Based System)이다. HyKET은 Flavor 객체 지향 언어를 프레임 시스템으로 사용한다. 규칙 기반 시스템은 Flavor와의 인터페이스를 바탕으로하여 역방향 추론기능과 순방향 추론기능을 제공한다. 본 논문은 먼저 Flavor를 Common LISP위에 구현한 작업을 간단히 설명하고 규칙기반 시스템을 제안하였다.

II. HyKET

HyKET는 <그림 1>과 같이 프레임을 기반으로 한 객체 지향 프로그래밍환경, 규칙기반 추론시스템, 사용자 인터페이스를 위한 그래픽 패키지, 윈도우시스템, Common LISP 프로그래밍환경과 같은 인공지능 개발 기술들을 하나의 시스템 형태로 묶어놓은 복합형 인공지능시스템이다[6].

1. HyKET의 구성

가. LISP 개발환경

HyKET는 기호처리기능이 가능하고 개발환경이 뛰어난 LISP 프로그래밍 언어를 기본적인 프로그래밍 언어로 채택하며 모든 환경을 LISP 프로그래밍환경위에 구축한다.

LISP은 인공지능연구 및 인공지능시스템 개발에 가장 많이 사용되고 있는 프로그래밍 언어로서 객체(object)의 개념을 실현하기가 용이하고 절차적 지식을 표현하기에도 적합하다. HyKET에서는 현재 LISP의 표준안으로 채택되고 있는 Common LISP을 사용하고 편집기, 디버거, 인스펙터등으로 구성된 LISP개발환경을 구축한다. 그리고 한글 기호처리를 위한 한글 LISP환경을 구축하고 Common LISP으로는 KCL(Kyoto Common LISP)을 기본적으로 사용할 계획이다.

나. 객체지향적 프로그래밍 환경

프레임을 기반으로한 객체지향적 프로그래밍 환경을 HyKET에 구축한다. 이 환경에서는 여러가지 객체들을 표현할 수 있고 이러한 객체들이 계층적인 구조로 연결되어 상속관계를 갖는다. 또한 규칙 내에서도 객체의 슬롯값이나 속성값들을 사용할 수 있으며 그래픽 인터페이스도 프레임을 기반으로 구축하고 윈도우 시스템과도 연결을 시킨다. 결국 LISP머신의 Flavor 시스템과 유사한 객체 지향적 환경을 구축한다.

다. 규칙 기반 시스템

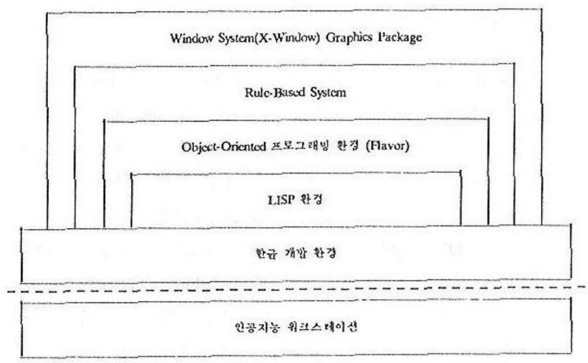
LISP 프로그래밍환경과 객체지향적 프로그래밍환경을 바탕으로 하여 규칙기반 시스템은 일반적인 규칙기반 시스템의 특성을 지니면서 프레임과 결합되어 다양한 지식표현 방법을 제공한다. 역방향 추론기능과 순방향 추론기능을 제공하며 규칙의 패턴으로 flavor의 인스턴스를 가질 수 있다. 또한 규칙의 각 문장에서 논리연산자를 표현할 수 있고 임의의 LISP 함수를 호출할 수 있다.

라. 윈도우시스템

기본적인 윈도우시스템의 기능을 가지며 규칙 시스템, flavor 시스템, LISP 환경과 결합되어 원활한 사용자 인터페이스 역할을 수행하고 효율적인 프로그래밍 개발환경을 제공하는 윈도우시스템을 구성한다. 현재 윈도우시스템은 세계적으로 X 윈도우로 표준화되고 있으며 특히 Common LISP 환경위의 윈도우시스템으로 CLX(Common LISP X Window)가 표준화되고 있다[7]. HyKET의 윈도우시스템은 CLX를 기본으로 KCL위에 구현할 것이다.

마. 그래픽 인터페이스

Flavor 시스템과 규칙 시스템의 내용을 그래픽으로 나타낼 수 있도록 하여 편집등을 용이하게 하고 이들 시스템과 결합되어 HyKET의 한 객체가 아이콘(icon)으로 표시될 수 있도록 한다. 즉 Active Image와 유사한 그래픽 인터페이스를 구축하며 주로 마우스를 사용할 수 있게 한다.



<그림 1> HyKET 의 구성

III. 프레임 시스템

HyKET는 프레임 시스템으로 Flavor 언어를 사용한다. 이 장에서는 Flavor 언어를 개요적으로 설명하고 Zetalisp Flavor를 Common LISP 환경으로 올리는 작업을 간단히 설명하였다.

1. Flavor 언어 개요 [8]

Flavor는 1979년에 MIT 대학의 LISP 머신 그룹에서 처음으로 개발되었다. 처음에는 윈도우 시스템을 만드는데 사용되

었으나 1981년에 Symbolics사가 Flavor를 더욱 효율적으로 구현한 이래로 LISP 머신의 입출력 프로그램, 네트워크 제어 프로그램, 디버거(debugger), 편집기, 사용자 인터페이스 프로그램 등과 같은 시스템 프로그램 개발에 많이 사용되고 있다

(가) flavor

Flavor에서는 객체가 하나의 flavor로써 표현되어진다 한 예를 들면 다음과 같다.

```
(defflavor ship
    (x-velocity y-velocity mass)
    () )
```

여기에서 defflavor는 하나의 flavor를 선언하는 함수이며 ship은 그 flavor, 즉 객체의 클래스가 가진 이름이다 또한 x-velocity, y-velocity, mass 등은 인스턴스 변수(instance variable), 즉 ship의 상태(state)와 특성(attribute)을 결정해주는 값들의 이름이다

(나) 인스턴스

실제 객체는 인스턴스로써 구현된다 flavor가 타입이라면 인스턴스 는 해당 객체로써 인스턴스 변수의 집합으로 구성된다 다음은 mass라는 인스턴스 변수의 값이 14인 ship의 인스턴스를 리턴(return)하는 예이다

```
(make-instance 'ship 'mass 14)
```

(다) method

어떤 flavor의 인스턴스위에서 하나의 오퍼레이션(operation)을 수행하는 Lisp 함수를 "method" 라고 부른다. Method의 body에서는 인스턴스 변수값을 변수명을 사용해서 액세스 할 수 있다.

```
(defmethod (ship speed) ()
  (sqrt (+ (expt x-velocity 2) (expt y-velocity 2))))
```

여기서 defmethod는 하나의 객체 ship에 speed란 이름의 method를 정의하는 함수이다 defmethod의 두번째 인자(argument)는 lambda 리스트이고 x-velocity는 ship이라는 flavor의 인스턴스 변수이다.

(라) Mixing flavors

전형적인 하나의 flavor는 여러개의 다른 flavor를 결합하여 구성할 수 있다 이 때 이렇게 결합되어지는 flavor를 컴포넌트 flavor(component flavor)라 하며, 기본 flavor는 이 컴포넌트 flavor의 내부변수를 상속한다 즉 컴포넌트 flavor의 내부변수를 자신이 정의한 것처럼 사용할 수 있다 이 때 컴포넌트 flavor는 컴포넌트 flavor 리스트를 통해서 정의되어진다 완전한 컴포넌트 flavor 리스트는 컴포넌트의 컴포넌트를 찾아가는 방법으로 구성되는 데 찾는 순서는 top-down, depth-first이다 예를 들면 다음을 보자

```
(defflavor flavor1 () (flavor2 flavor3))
(defflavor flavor2 () (flavor4 flavor5))
(defflavor falvor3 () (flavor4))
```

이 때 flavor1의 컴포넌트 flavor 리스트(list)는 다음과 같다

```
(flavor1 flavor2 flavor4 flavor5 flavor3)
```

이 때 flavor1으로 flavor2, flavor4, flavor5, flavor3에서 정의된 variable들과 method들이 상속되어 복합된 flavor를 생성한다 Method들은 상속될 때 약간의 변형이 있을 수 있는데 이것은 같은 이름의 method들이 여러개 있을 때 그들이 결합하는 형태에 따라 분류될 수 있다.

2 Flavor 구현

다른 porting작업과 마찬가지로 Flavor를 구현하는 데 가장 먼저 했던 일들은 ZETALISP과 GCLISP간의 인터페이스를 구현하는 일이었다 즉 ZETALISP에 있는 함수들 중에서 GCLISP에 없는 함수와 GCLISP과 이름이 같지만 다른 기능을 갖는 함수들을 정의하는 일이었다

그러나 Flavor 자체가 실행시간(Run time)에 method에서 인스턴스 변수를 액세스 하기위한 변수 검사를 하기위하여 LISP머신의 기계적 지원을 받는다. 이것을 소프트웨어로 그대로 구현하는 것은 LISP의 수행중 과중한 부담을 주게되므로 다른 방법을 사용하였다

본절에서는 GCLISP에 구현된 Flavor의 인스턴스 표현에 대해 설명하고 method에서 인스턴스 변수를 액세스하기위한 방법의 구현에 대하여 설명하였다.

1) 인스턴스 표현

하나의 flavor의 인스턴스는 스트럭춰(structure)의 인스턴스로 해결하였다 예를 들면 x-pos, y-pos, x-vel, y-vel을 모든 인스턴스 변수로 갖는 ship이라는 flavor가 정의되었다면 위의 변수들을 스트럭춰 변수로 가지는 스트럭춰가 가상적으로 선언되며 하나의 인스턴스가 만들어질 때마다 이 스트럭춰의 인스턴스가 만들어 진다

2) 인스턴스 변수 액세스

이 스트럭춰의 인스턴스 변수들이 method 수행시에 액세스되는 Flavor의 인스턴스 변수들과 연결될 수 있어야한다 연결시킨 방법은 다음과 같다

먼저 flavor 컴비네이션 후 method 컴비네이션 시에 모아지는 각 method의 body를 flavor 인스턴스 변수와 같은 이름의 변수들로 둘러싸게 되며 초기값으로는 현재 스트럭춰의 해당 변수의 값을 준다. body가 끝난 후에는 변수의 값을 스트럭춰의 해당 변수에 넣어 준다

위와같은 방법은 method의 body가 상대적으로 적은 변수들을 액세스하는 경우 불필요한 작업을 많이 하게되는 단점이 있으나 실행시간에 변수 검사를 할 필요가 없다는 장점이 있다.

IV. HyKET의 규칙기반 시스템

1 Flavor와의 인터페이스

Flavor가 제공하는 모든 flavor와 인스턴스들이 추론과정에서 사용되는 것은 아니다 그러므로 규칙기반 시스템에서 사용될 대상들을 미리 정의하는 것이 필요하다 이것을 정의하는 함수를 "defclass"라 하였다. 예를 들면 다음과 같다

```
(defclass ship
    :variables (x-velocity y-velocity)
    superclass moving-object . )
(defmethod (ship speed) ()
    (sqrt (expt x-velocity 2) (expt y-velocity 2)))
```

이때 "defclass"의 옵션으로는 flavor에서는 defflavor에서 허락하는 모든 옵션들이 그대로 허용된다 또한 Flavor와 마찬가지로 method들이 정의될 수 있으며 flavor와 거의 같은 기능을 한다 그러나 "defclass"로 정의된 것은 규칙의 패턴으로 사용될 수 있으며 "defclass"의 인스턴스들은 이름을 가지고 있어서 이름으로 인스턴스를 참조할 수 있다

2 규칙 Expression [9,10]

HyKET의 규칙 Expression은 일차술어논리의 명제와 같은 역할을 하며 또한 각 규칙에서 HyKET의 프레임기반 표현구조를 사용할 수 있도록 해준다

1) *(IN-CLASS <instance-name> <class>)*
IN-CLASS expression은 <instance-name>가 <class>라는 이름의 class의 인스턴스인 것을 서술한다

2) *(SUBCLASS-OF <class1> <class2>)*
SUB-CLASS expression은 <class2>가 <class1>의 superclass인 것을 서술한다.

3) *(SLOT-VALUE <slot-name> <instance-name> <value>)*
SLOT-VALUE expression은 <instance>의 한 슬롯 <slot-name>에 대하여 그 값이 <value>임을 서술한다. 이 expression의 assertion은 주어진 슬롯의 값을 바꾸는 것이 된다.

4) *(CREATE-INSTANCE <class> <instance-name>*
<slot-name> <value>
<slot-name> <value> ..).
(REMOVE-INSTANCE <class> <instance-name>)
이 두개의 expression은 <conclusion> 부분에 사용되며 인스턴스를 만들고 제거하는데 사용된다 여기서 <instance-name>은 <class>내에서 유일하게 존재하여야한다

3. 규칙의 형태 [4]

규칙을 외부형태(external form)로 표현하면 다음과 같다

```
(RULE rule-name
      IF <condition> <condition>  ..
      TEST <lisp function> ..
      THEN <conclusion> ...
      DO <lisp function> ...)
```

규칙은 위와같은 형태로 표현되며 <condition>, <conclusion> 부분에는 규칙 expression이 들어갈 수 있으며 <lisp function>에는 임의의 함수가 패턴변수를 인자로 가지고 들어갈 수 있다 패턴 변수는 ?로 시작하는 심볼로 표현된다 <condition>, <conclusion>부분의 각 expression들은 의미상 AND로 연결되어 있으며 의미를 명확하게 하기위하여 "AND"를 각 expression사이에 써줄 수 있다

4 규칙 함수 [10]

규칙 함수는 규칙과 관계하여 순방향 추론, 역방향 추론등을 시작하게 하는 함수들이다

1) *(ASSERT <rule expression> <rule expression>)*
Class 정의와 instance의 집합인 지식 베이스에 새로운 사실을 집어넣는 함수로써 대개 순방향 추론을 일으킨다

2) *(QUERY <rule expression> <rule expression>)*
지식 베이스에 있는 사실을 확인하고자 할때 사용하는 함수로써 내개 역방향 추론을 일으킨다

3) *(ASK-USER <rule expression> <rule expression>)*
여기에서 <rule expressions>의 경우에는 반드시 패턴 변수를 가지게 되는데 이 패턴변수의 참 값을 사용자에게 묻는 함수이다

4) *(RETRACT <rule expression> <rule expression> ..)*
지식 베이스로부터 사실들을 제거하는 함수이다

5 순방향, 역방향 추론시스템

순방향, 역방향 추론시에 규칙을 결정하기위한 전략을 conflict resolution 전략과 탐색전략(search strategy)이라 한다 HyKET에서는 conflict resoultion 전략으로 least-premise first, greatest-premise first를 제공하며 탐색 전략으로는 breadth first, depth first, best first등을 제공하여 사용자로 하여금 선택할 수 있도록한다

6 규칙기반 시스템의 구현

HyKET의 규칙기반시스템은 Flavor 언어를 사용하여 구현된다. 각 규칙은 flavor의 인스턴스로 표현되며 규칙 expression을 다루기위한 함수들이 method로 표현된다 그래서 규칙들이 쉽게 grouping될 수 있으며 각 규칙들의 group에서 다른 규칙 결정전략을 사용할 수 있다

V. 결 론

HyKET은 프레임기반 지식표현, 규칙기반 추론, 객체지향언어, 윈도우 시스템, 그래픽 기능등을 제공하는 복합형 인공지능 개발 시스템이다. 프레임 기반 지식표현과 객체지향언어를 위해 Flavor를 사용한다 규칙기반 시스템으로서는 Flavor와의 인터페이스를 제공하며 flavor의 인스턴스를 패턴으로 사용할 수 있는 시스템이며, 역방향, 순방향 추론기능을 제공한다 Flavor는 PC Common LISP의 일종인 GCLISP위에 구현되었으며 제안된 규칙기반 시스템도 곧 구현될 것이다

참 고 문 헌

[1] Daniel G Bobrow, Sanjay Mittal, and Mark J Stefik, "Expert Systems Perils and Promise," Communications of the ACM, vol. 29, no 9, September 1986, pp 880-894

[2] D A. Waterman, A Guide to Expert Systems, Addison-Wesley, Reading, MA, 1986

[3] M Genesereth and M Ginsberg "Logic programming," Communications of the ACM, vol 28, no 9, September 1985, pp. 933-941

[4] Frederick Hayes-Roth, "Rule-Based Systems," Communications of the ACM, vol 28, no 9, September 1985, pp 921-932.

[5] R Fikes and T Kehler, "The role of frame-based representation in reasoning," Communications of The ACM, vol 28, no 9, 1985, pp 904-920

[6] Y. W. Kim and J H Kim "Development of Hybrid Artificial Intelligence Development System," Proceedings of Joint Conference on Electronics and Communications, August 1987 pp 298-213

[7] Robert W Scheifler, "The X Window System," ACM Transaction on Graphics, Vol 5, No 2, April 1986, pp 79-109.

[8] D Weinreb and D Moon, Lisp Machine Manual, Symbolics Inc , 1981

[9] Mark H Richer, "An evaluation of expert system development tools," Expert Systems, vol 3, no 3, July 1986, pp 166-183

[10] SPERRY, Intellicorp KEE Software Development System User's Manual (KEE Version 2 1), July 18, 1985