한국지능정보시스템학회논문지 제9권 제3호 2003년 12월(pp.49~63)

지식관리 시스템을 수반한 전문가 시스템 구축 도구

서의현
목원대학교 컴퓨터 공학과
(ehsuh@mokwon.ac.kr)

본 논문은 효율적이고 신뢰성있는 전문가시스템을 구축하기 위한 도구를 제안하고 구현한다. 전문가 시스템에서 추론은 특정 전문분야의 지식베이스에 저장된 지식을 기반으로 행해진다. 이 때 전문가 시스템이 신뢰할 수 있는 추론의 결과를 얻기 위해서는 다양한 형태의 지식들이 이용되고, 지식의 정확성 및 일관성이 유지되어야 한다. 이러한 관점에서 본 논문은 지식이 지식베이스에 첨가되기 전에 지식의 오류를 점검함으로써 오류가 없는 지식들을 선택적으로 지식베이스에 첨가하여, 지식의 정확성 및 일관성을 유지하는 지식관리 시스템을 구축했다. 아울러 본 논문은 전문가 시스템이 추론과정에서 다양한 지식을 이용하도록 절차적 지식과 데이터베이스에 저장된 선언적 지식을 호출하여 사용할 수 있도록 했다.

논문접수일 : 2003년 3월 게재확정일 : 2003년 11월 교신저자 : 서의현

1. 서론

전문가 시스템은 한 특정 분야의 전문가를 대신하여 문제를 해결하는 시스템으로서 기본적으로 추론엔진 및 제어 모듈, 지식베이스와 사용자 인터페이스로 구성된다. 이 시스템은 규칙과 사실들의 매칭에 의한 일련의 규칙을 실행하여 추론을 수행하는 과정에서 유용한 형태의 지식들을 생성하여 사용자에게 제공한다. 이렇게 생성된 지식이 유용하기 위해서는 지식 공학자가 해당 분야의 많은 전문지식을 모아서 원하는 형태의 추론을 제공할 수 있는 지식 베이스가 구축되어야 한다. 또한 지식의 효율적인 추론을 위한 메카니즘과 전문가 시스템을 편리하게 이용하기 위한 인터페이스가 제공되어야 한다.

이러한 맥락에서 전문가 시스템을 쉽게 구축할 수 있는 전문가 시스템 쉘이 제공되고 있다. 많이 사용되는 쉘에는 OPS5, KES, EXPERT, CLIPS(Riley 1997)등이 있다. 이러한 쉘들은 추론 메카니즘을 제공하며 지식베이스를 쉽게 구축하고 유지할 수 있는 개발 환경을 제공하고 있어 지식 공학자의 노력을 많이 절감시켜준다. 그러나 이러한 쉘에서 전문가 시스템의 신뢰도의 성능을 좌우하는 지식베이스 내의 지식의 검증은 극히 기본적인 것에 불과하며 지식의 일관성 및 정확성 여부는 전적으로 지식 제공자에 의하여 결정된다.

그런데 지식의 양은 방대한 경우가 많기 때문에 지식 제공자(전문가)나 지식 공학자가 내용 전체를 파악하기 어렵고, 전문가는 때때로 직관

적으로 생각하므로 추론에서의 많은 과정을 생략한 채 지식을 지식베이스에 삽입할 수 있다. 또한 지식베이스는 점진적인 방식으로 오랜 기간에 걸쳐 구축되고 여러 명의 전문가가 지식을 함께 저장하기 때문에 지식의 중복, 순환과 모순의 문제가 야기될 수 있다. 이러한 문제들을 해결하기 위해서는 전문가 시스템이 지식베이스를 사용하기 전에 정확성을 검증하여 발견된 오류를 자체 수정하거나 자체 수정이 불가능할 경우 전문가에게 문의하여 수정하는 시스템이 요구된다.

그러나 생성규칙으로 표현된 지식의 일관성 검증 문제는 NP-complete 문제로서 시간복잡도가 문제의 크기에 대해 지수승의 형태로 나타난다. 이 중 쌍 검증 방식은 경우에 따라서는 가치있는 규칙이 생략되고 의미 없는 규칙이 취급되는 일이 발생할 수 있으며 규칙의 수가 많아질 경우 시간이 많이 걸리는 단점이 있다.

따라서 본 논문은 확실한 특성의 리스트와 가능한 특성의 리스트를 생성하고 검증 방법 및 단계를 개선하는 형태로 쌍 검증 방식을 보완한다. 이 리스트들과 개선된 검증 방법은 오류를 검색할 때 전방 추론이나 후방 추론을 해야 하는 과정의 대부분을 생략할 수 있게 함으로써 시간이 많이 걸리는 단점을 보완함과 동시에 오류 점검 단계에서 모든 가능성을 고려함으로써 생략되는 규칙이 없어 정확한 오류 점검을 수행한다. 즉, 보완된 쌍검증 방식을 이용하여 새로운 지식이 기존의 지식베이스에 첨가되기 전에 새로운 지식의 정확성 및 일관성을 검증한 후 일관성이 유지될 경우에만 지식을 지식베이스에 첨가하는 지식관리 시스템을 구축하였다. 이와 함께 지식베이스 및 데이터베이스의 지식을 호출할 수 있는 전문가 시스템 개발도구를 구축하여 지식 관리 시스템과 연계시켰다. 지식 관리 시스템의 모든 하위 시스템들은 C++언어와 Oracle SQL을 사용하여 Sun Sparc 5.5에 구축되었다.

이를 위하여 본 논문은 2장에서 기존 시스템의 문제점과 분석을 소개한 후 3장에서는 지식 및 데이터베이스와 사용자 인터페이스 등 시스템의 구성요소에 대해 소개한다. 4장에서는 지식관리 시스템에 대해서 자세히 설명하고 5장에서는 구현된 시스템을 실행한 예를 보인다. 마지막으로 결론에서는 지식 관리 시스템을 갖춘 전문가 시스템을 개발 도구를 구축함으로써 확인된 결과를 요약하고 향후 과제를 다룬다.

2. 기존 시스템의 문제점 및 분석

NP-복잡성 문제인 생성 규칙으로 표현된 지식베이스의 일관성 검증을 위한 많은 시도가 있었다.

지식 기반 경영 시스템(Knowledge Based Management System: KBMS) 내에 일관성을 제약하는 조건을 제시하고, 제약 조건을 위반했을 경우 예외 처리를 실행하기 위하여 실행 형태 제어 규칙(Activation Pattern Controlled Rule)을 사용한 방식이 있다(Eick 1993). 그런데 이 방식은 부분적인 일관성만을 자동으로 처리할 수 있을 뿐이다.

의존 관계표를 사용하여 일관성과 완결성을 점검하는 방법(Nguyen 1985)은 열거 방식으로 처리하기 때문에 시간이 많이 걸리는 단점이 있다.

결정표(Decision Table)를 사용하는 방식(Cragun 1987)에서는 간혹 오류의 존재만을 발견하고 오류의 위치를 정확하게 찾아내지 못하는 단점이 있다. 또한 복잡한 관계를 가지는 규칙들은 추론 과정 중 연결된 규칙이 많아 지수 함수

적인 성능을 나타낸다(Nazareth 1989).

그래프 표현 방식은 프레미스의 개념적인 의존 관계를 나타내기 쉬운 표현법으로써 방향 그래프 (Directed Graph)(Nazareth 1991), 추론 그래프 (Inference Graph)(Nguyen 1987), 하이퍼그래프(Hypergraph)(Valiente 1993)등의 방법이 있다. 그러나 이러한 방법들은 복합절과 단순절 사이의 의존 관계를 나타내기 어렵다(Botten 1989)(Ramaswamy 1997). 따라서 그래프 표현 방식은 노드를 첨가하거나 전이를 묘사하고 또는 하이퍼 아크를 사용하여 복합절을 묘사함으로써 중복과 포함을 정확히 발견하여 수정한다. 그렇지만 모순과 불완전성은 여전히 완벽하게 처리되지 못하는 단점이 있다. 방향 하이퍼 그래프 (Directed Hypergraph)(Ramaswamy 1997)는 이러한 그래프의 단점을 보완하였지만 고려해야 할 하이퍼 노드의 조합의 수가 많아지면 실제로 이들을 다 묘사하지 못함으로써 오류가 생성될 수도 있다. 또한 이 방식은 구조적 오류만을 검색할 수 있을 뿐이며 규칙이 많아질 경우에는 시간이 많이 소요된다. 개념 그래프(Conceptual Graph)는 계층적 구조로써 지식을 체계적으로 분류하여 계층적으로 저장하고 편집 방법을 통해 중복을 제거한다(Cyre 1997)(Ellis 1995). 그러나 이 방식은 표현의 다양성 때문에 극히 제한적인 일관성만을 검진할 뿐 여러 곳에 있는 비일관성 오류를 점검하기 어렵다.

페트리 네트(Petri Net)를 이용한 방식(Liu 1996)(Nazareth 1993)은 일관성 검진이 정해진 실행 순서로 진행되어야 하는데 형식이 완전하지 못하면 제약 조건이 적용되지 못하고 수행시간도 많이 걸리는 단점이 있다.

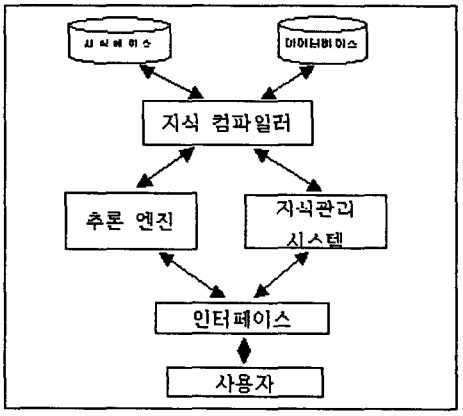
지역 탐색 방식(Brisoux 2001)은 매우 큰 지식베이스에서 일관성 검증에 효율적인 성능을 보이지만 전체의 탐색 영역을 모두 고려하지 못하고 지식베이스가 비일관성 상태에 있다는 것을 직접 증명할 수 없다는 단점이 있다.

쌍 검증(Pairwise Checking)에서는 생성된 절들이 적용 가능한 규칙들과 비교된다(LeBeux 1986) (Morizet-Mahoudeaux 1991). 쌍검증 방식에 계층적 구조를 첨가하거나, 비일관성의 새로운 정의를 기초로 하여 확장된 쌍검증 방식이 소개되었다.(Koczkidaj 1993) (Koczkidaj 1997). 이 때 경우에 따라서는 가치 있는 규칙이 생략되고 의미 없는 규칙이 취급되는 일이 발생할 수 있으며 규칙의 수가 많아질 경우 시간이 많이 걸리는 단점이 있다(Liu 1996).

따라서 본 논문은 전체 지식베이스의 영역을 고려하면서도 수행 시간을 단축하기 위하여 기본 구조 및 검증 방법을 개선시킨 쌍 검증 방식을 이용한 지식 관리 시스템을 제안한다.

3. 시스템 개요

시스템은 <그림 1>과 같이 구성된다.



<그림 1> 시스템 구조

3.1 지식베이스

지식베이스의 기본 요소는 생성 규칙이다. 생성규칙은 규칙들을 모듈화하여 나타낼 수 있고, 비절차적 방식으로도 규칙들로 표현된 지식을 사용할 수 있을 뿐 아니라 여러 종류의 지식을 쉽게 표현할 수 있기 때문이다. 지식은 조건 부분과 실행 부분으로 표현되며 선언적 지식과 절차적 지식도 같은 방식으로 나타낼 수 있도록 하였다. 이러한 생성 규칙들의 실행 효율을 증가시키기 위하여 내부 표현 구조는 네트워크 구조로 표현하였다. 절차적 지식에는 데이터 베이스내의 지식을 호출, 삽입, 제거 및 수정할 수 있는 기능도 포함되어 있다.

3.1.1 규칙의 표현

규칙의 표현은 <그림 2>와 같다.

```
<Knowledge Base> ::= <rule> | <rule><Knowledge Base>
<rule> ::= rule <rule-name> (<certainty factor>) <rule-class-name>
          if <condition> then <action>
<rule-name> ::= <string>
<certainty factor> ::= 0.0 ····· 1.0
<rule-class name> ::= <string>
<condition> ::= {<proposition>}_1^7

<action> ::= <proposition>
<proposition> ::= <text> <value-type> {<operator> <value>}
<text> ::= <string>
                                                              255
<string> ::=<string_type> <question_type> {(<letter> | <digit>)}_1
<string_type> ::= i | n | f
                  [참고] i : initial
                     f : goal
                     n : non-initial
<ques_type> ::= 1 | 2 | 3
                  [참고] 1 : ·····isn't it ?
                     2 : What is····· ?
                     3 : ·····doesn't it ?
<letter> ::= A| ··· |Z|a|···|z
<digit> ::= 0|1|···|9
<operator> ::= '=' | '!=' | '<' | '<=' | '>' | '>=' | <indicator_function_result>
<indicator_function_result> ::= n | y
                  [참고] n : 결과 없음.
                     y : 결과 존재.
<value> ::= <string> | <number> | <function-name>
<number> ::= <integer> | <real>
<function-name> ::= <string>
<value_type> ::= t | f | s | n | r | x | i
        [참고] t : true,      f : false,    s : string,    n : integer,
            r : real,         x : function, i : interface of Database.
```

<그림 2> 규칙의 표현

3.1.2 절차 부가(Procedural Attachment)

생성 규칙 표현법의 장점들 중 하나는 다양한 지식을 같은 형태로 표현할 수 있다는 것이다. 전문가 시스템은 필요한 경우 함수를 호출하여 비결정적인 값을 계산할 수 있어야 한다.

예를 들면,

```
rule1 (0.9) <modeler>
  if  type of model, s=triangle
      surface, x=calcul_surface_tri(base, height)
      surface, r >= 100.0
  then triangle_model_no = 1
```

이 경우 함수는 0개 이상의 매개변수를 가진다.

추론 도중 이러한 지식을 만나면 먼저 함수 지식 인터프리터가 이 문장을 분석하여 함수명과 매개 변수의 개수 및 매개 변수를 분리한 뒤 해당 함수를 진행시킨다.

3.1.3 데이터베이스의 지식에 대한 호출

전문가 시스템에서 추론 엔진은 특정한 영역의 지식과 사실(fact)들을 가지고 추론을 행한다. 그러나 경우에 따라서는 사실의 양이 많아지고 데이터베이스에 저장되어 있기 때문에 데이터베이스 내에 저장된 사실들을 호출해야 할 경우가 있다.

따라서 본 시스템은 데이터베이스 내의 정보의 호출, 추가, 제거 및 수정할 수 있는 기능을 갖추었다. 관계(relation), 엔티티(entity), 키(key), 레코드(record)와 호출된 값을 보관할 수 있는 변수의 이름들을 가지고 조작할 수 있다.

예를 들면,

```
rule2 (0.8) <modeler>
  if    eq1, i y a(relation_name, equation, key_
        col:edge1_id, return_value)
        eq2, i y a(relation_name, equation,
        key_col:edge2_id, return_value)
        calcul_state, x = calcul_parall(eq1, eq2)
        calcul_state, s = intersection
  then delete_edge, i n s(relation_name, key_
        col, edge2_id)
```

이것은 절차 부가의 일종이므로 수행과정은 절차 부가와 동일하다.

3.2 데이터베이스

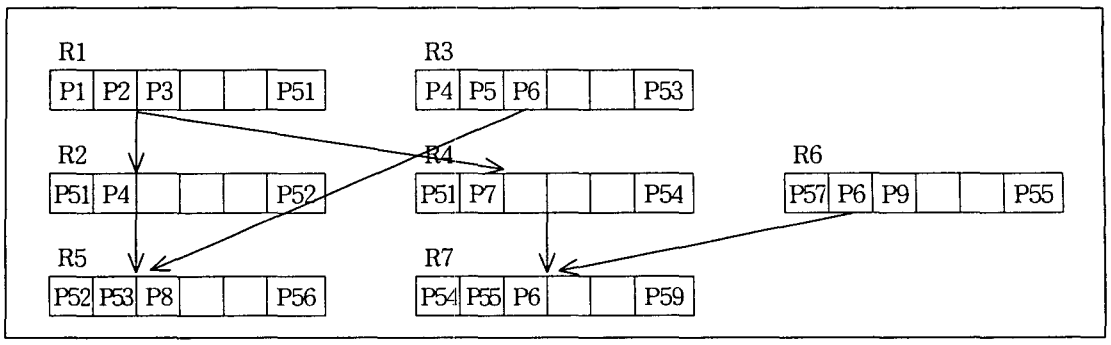
관계형 데이터베이스로써 Oracle 7.3.3을 사용하여 일반적인 정보들을 표현하였다. 전문가 시스템은 필요한 경우 데이터베이스의 정보들을 호출할 수 있도록 하였다. 사실들에 대한 정보의 양이 많을 경우, 데이터베이스를 이용하여 정보를 관리하는 것이 바람직하기 때문이다.

3.3 추론 시스템

추론 시스템의 주요 구성 요소는 추론 엔진인데, 생성 규칙 기반 시스템에서 많이 사용하는 추론 엔진의 기법은 전방 추론(forward chaining)과 후방 추론(backward chaining)이다(박창현 1992). 본 시스템에서는 전방 및 후방 추론을 구축하였다. 추론 방식은 사용자가 선택할 수 있다. 단, 후방 추론일 경우 가정을 반드시 설정해 주어야 한다.

3.4 지식 컴파일러

지식 베이스에 저장된 규칙은 지식 컴파일러에 의해 네트워크 방식의(<그림 3>) 내부 표현 구조로 나타내어진다. 이는 추론 도중 가능성이 있는 규칙만을 탐색함으로써 추론 엔진의 추론



<그림 3> 지식의 네트워크 표현

속도를 증가시키려는 데 그 목적이 있다. 또한 하나의 규칙 내부에 이 규칙과 연결되어지는 규칙의 정보를 포함하고 있는 네트워크로 표현함으로써 도달될 수 없는 규칙, 더 이상 연결되지 않는 규칙들을 쉽게 찾아낼 수 있는 장점을 가지기 때문이다. 지식은 오류를 점검하는 과정을 거쳐 오류가 없으면 네트워크 구조에 추가되며 최종적으로 지식베이스에 저장된다.

3.5 지식 관리 시스템

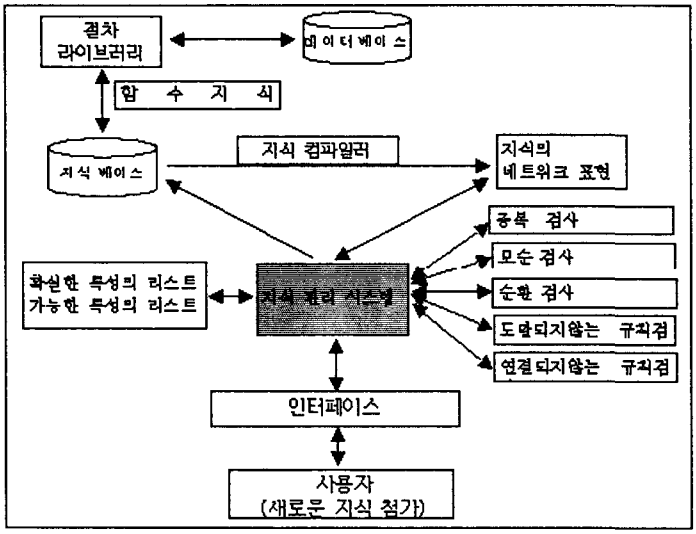
지식 관리 시스템은 새로운 지식을 지식베이스에 추가하고자 할 경우, 지식이 추가되기 전에 지식의 정확성 및 일관성을 검증한다. 이는 만약 오류가 존재한다면 오류를 자체 수정하거나 자체 수정이 불가능할 경우 전문가에게 문의하여 오류를 수정하도록 한 후 지식이 첨가되도록 하는 시스템이다. 이 시스템의 목표는 지식의 정확성 및 일관성을 유지시켜 전문가 시스템의 신뢰도를 높이는 것이다. 지식관리 시스템은 <그림 4>와 같다.

3.6 사용자 인터페이스

사용자가 전문가 시스템을 쉽게 이용할 수 있도록 대화형의 인터페이스가 구축되었다. 전문가 시스템이 사용자에게 질문을 하면 사용자는 주어진 답안 중에서 하나를 선택하거나 간단한 스트링이나 숫자를 입력하든지 혹은 예(true), 아니오(false) 형태로 대답하면 된다.

4. 지식 관리 시스템

전문가 시스템은 규칙과 사실들을 매칭시켜 일련의 규칙을 실행시키는 방식으로 추론을 수행하는데 이 때 추론된 지식의 신뢰도를 높이기 위해서는 지식의 정확성 및 일관성이 유지되어야 한다. 따라서 이 시스템에서는 확실한 특성의 리스트와 가능한 특성의 리스트가 생성되고 이들을 이용하여 지식이 첨가되기 전 중복, 모순, 순환 및 도달될 수 없는 규칙과 더 이상 연결되지 않는 규칙 등의 오류가 검사된다. 만약 오류가 있다면 지식 제공자의 승인을 거쳐 오류를 제거한 뒤 새로운 규칙이 삽입된다.



<그림 4> 지식 관리 시스템

4.1 확실한 특성의 리스트와 가능한 특성의 리스트

각 프레미스(premise)의 확실한 특성과 가능한 특성은 지식 관리 시스템에 의해 구해진 후 리스트 구조로 표현된다. 이 때 확실한 특성에는 상식과 범용 지식이 포함된다.

■ 정의 1) 네트워크의 규칙을 R_i, i=1..n으로 표시하고, 하나의 규칙이 조건 부분(P)과 실행부분(A)으로 표현되면 규칙 R_i의 조건부분의 모든 프레미스는 Set R_i라 정의한다.

■ 정의 2) $R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_n$이고 R_n의 실행부분이 A인 일련의 연속된 규칙들을 A의 경로라 하며 T(A)라 정의한다. 만약 A가 조건으로만 사용된다면 A의 경로가 없다.

■ 정의 3) 만약 E의 실행이 P의 실행을 내포하면 P는 E의 확실한 특성(Certain(E))이라 정의한다. 표현 방법은 $E \rightarrow P$이다. 다음 3조건 중 하나를 만족하면 P는 E의 확실한 특성이다.

① $\forall K \in [1..q], \exists i \in [1..n], R_i \in T_k(E)$ and $P \in$ Set R_i.

즉 $Certain(E) = \bigcap_{i=1}^{q} Antecedent(R_i)$

$Antecedent(R_i) = \bigcup_{j=1}^{m} Certain(P_{ji}) \cup \{P_{1i}, P_{2i}, \ldots P_{mi}\}$

$Certain(P_{ji}) = \emptyset$ 만약 P_{ji}에 관련된 경로가 없다면.

② 실행 부분이 P인 규칙이 존재해서 그 규칙의 모든 조건 부분이 E의 확실한 특성일 경우.

③ 하나의 조건 not(P)와 실행부 A를 가지는 규칙이 존재하고 not(A)가 E의 확실한 특성일 경우.

■ 정의 4) 가능한 특성은 일어날 수도 있고 그렇지 않을 수도 있는 특성을 말한다. 만약 P의 실행이 E의 실행에 기여한다

면, P는 E의 가능한 특성(Eventual (E))이라 정의한다. 만약 아래의 조건 중 하나를 만족하면 P는 E의 가능한 특성이다.

① $\exists K \in [1..q]$, $\exists i \in [1..n]$, $R_i \in T_k(E)$ and $P \in$ Set R_i

즉, $Eventual(E) = \bigcap_{i=1}^{q} Antecedent(R_i)$

$Antecedent(R_i) = \bigcup_{j=1}^{m} Eventual(R_{ji}) \cup \{P_{1i}, P_{2i}, \cdots P_{mi}\}$

$Eventual(P_{ji}) = \varnothing$: 만약 Pji에 관계된 경로가 없다면.

② 실행 부분이 P인 규칙이 존재해서 그 규칙의 모든 조건 부분이 E의 가능한 특성일 경우.

③ 하나의 조건 not(P)와 실행부 A를 가지는 규칙이 존재하고 not(A)가 E의 가능한 특성일 경우.

결국 다음과 같은 결과가 얻어진다.

Certain(E) ⊂ Eventual(E).

4.2 지식베이스의 오류

본 논문에서는 지식의 중복, 모순, 순환, 도달되지 않는 규칙 그리고 더 이상 연결되지 않는 규칙을 오류라 칭한다.

4.2.1 중복

지식의 중복은 다음과 같이 점검된다.

■ 구조적 중복과 수정

- Set R_1 = Set R_2 and $A_1 = A_2$: 하나의 규칙은 제거한다. 단, A는 실행부를 나타낸다.
- Set R_1 ⊂ Set R_2 and $A_1 = A_2$: R_1이나 R_2 둘 중 하나의 규칙만 선택하는데 R_2를 선택하는 것이 바람직하다.
- $A_1 \in$ Set R_2 and #(Set R_1 ∩ Set R_2) >= 1 : Set R_2 = Set R_2 - Set R_1

■ 의미적 중복과 수정

- 하나의 규칙 안에서 조건 부분에 프레미스 X를 첨가하려는데 X의 확실한 특성 중 하나가 이미 소개되어진 프레미스 P인 경우 프레미스 P를 제거할 수 있다. 그렇지 않으면 상황을 열거한다.
- Set R_1 = Set R_2 이거나 Set R_1 ⊂ Set R_2 이고 A_1 과 A_2중 어느 하나가 다른 하나의 확실한 특성 또는 가능한 특성의 관계에 있을 때, 확실한 특성 또는 가능한 특성을 실행부로 가진 규칙을 제거한다.

4.2.2 모순

■ 구조적 모순과 수정

- Set R_1 = Set R_2 and $A_1 \neq A_2$: 만약 A_1 과 A_2가 논리적 모순 관계이면 규칙 중 하나를 버려야 한다. 이 경우 지식 제공자에게 수정하라는 메시지를 주어 Set R_1 혹은 Set R_2를 수정하거나 두 규칙 중 하나를 버린다. 단, 논리적 모순 관계란 해당되는 프레미스의 반의어거나 반의어의 가능한 특성 혹은 확실한 특성을 말한다.
- Set R_1 ⊂ Set R_2 and $A_1 \neq A_2$: 만약 A_1과 A_2가 논리적 모순관계이면 전문가의 지시대로 규칙 중 하나는 버려야 한다. 그렇지 않을 경우 규칙 R_1은 그대로 두고 규칙 R_2를 수정하여 첨가한다. 즉 Set R_2 = (Set R_2 - Set R_1) ∪ {A_1}.

■ 의미적 모순과 수정

이미 소개되어진 프레미스 P의 확실한 특성과는 반대 의미로써 묘사되어진 프레미스 X를 첨가하려 할 때 시스템은 모순의 근원을 알려준다. 만일 X가 확증되었다면 프레미스는 기록되고 X가 P의 확실한 특성이 아니다라는 사실이 기억된다. 반대의미는 확실한 특성의 리스트에 포함되어져 있다. 예를 들면 P의 확실한 특성이 ~X이면 P와 X는 서로 반대라는 사실을 알 수 있다.

4.2.3 순환

하나의 규칙 내에 프레미스 P를 첨가하려는데 실행부 A가 P의 가능한 특성 중 하나인 경우, 시스템은 사용자에게 경고하고 프레미스를 첨가하지 않고 규칙의 첨가자가 선택하도록 한다.

4.2.4 더 이상 연결되지 않는 규칙

한 규칙의 실행부가 목표가 아니고, 이 규칙의 실행부가 다른 규칙의 조건 부분에 포함되어 있지도 않다면, 그 규칙은 더 이상 연결되지 않는 규칙이다. 이 규칙은 지식베이스에 포함되지 말아야 하거나, 추론할 때 이 규칙의 실행부를 사용하는 다른 규칙이 빠진 경우이므로 지식 제공자에 의해 빠진 규칙이 보완되어야 한다. 네트워크 구조에서 다른 규칙으로의 링크가 없는 경우이므로 쉽게 찾아질 수 있다.

4.2.5 도달될 수 없는 규칙

한 규칙의 조건부가 입력 변수가 아니고 다른 규칙의 실행부도 아닐 때, 즉 규칙의 진입 차수가 0일 때 이 규칙의 실행부는 도달될 수 없는 절이다. 이 규칙은 필요 없는 규칙이거나 이 규칙의 조건 부분에 도달되기 위한 규칙이 빠진 경우이기 때문에 빠진 규칙이 보완되어야 한다.

4.3 새로운 지식의 첨가 및 제거

4.3.1 지식을 첨가하기 전의 오류점검

새로운 지식을 첨가하기 전에 오류의 점검 과정을 차례로 기술하면 다음과 같다.

단계 1 : 지식베이스의 규칙들 R_i와 첨가하려는 규칙 R을 비교한다. 모든 R_i에 대해 만약 Set $R_i \subset$ Set R, $A_i \notin$ Set R 이면 Set R=Set R $\cup \{A_i\}$ 이다.

단계 2 : 첨가하려는 규칙 내부에서 조건 부분들 사이에 의미적 모순 또는 순환을 검증하고 제거한다. 조건부의 프레미스들을 서로 비교하여 모순 관계를 점검하고 만약 모순이 발견되면 전문가에게 수정을 요구한다. 조건부와 실행부에 같은 프레미스가 있으면 순환이다.

단계 3 : 불가피한 프레미스만 남겨두고 Set R을 감축시킨다. 즉 지식의 구조적 및 의미적 중복을 제거하는 것이다. Set R′ = Set R - $\cup \{$가능한 프레미스(P_i)$\}$

단계 4 : 첨가하려는 R은 조건 부분들이 같은 규칙들과 비교된다. 즉 구조적 중복과 모순을 제거하는 것이다. 만약 하나의 규칙 R_i가 R과 같은 실행부를 가지면 R은 제거된다. 만약 R_i와 R의 실행 부분이 다르다면 시스템은 어느 것이 모순된 것인지 혹은 둘 다 받아들일 것 인지를 검사한다.

단계 5 : 규칙 R은 실행 부분이 같은 규칙들과 비교된다. 즉 구조적 중복과 모순을 제거하는 것이다. 만약 조건 부분이 다른

R_i의 Set(R_i) 중 Set(R)과 모순되는 것이 있으면 모순을 지적하고 전문가에게 수정을 요구한다.

4.3.2 규칙의 첨가와 네트워크의 수정

규칙은 위의 5단계를 거쳐 오류가 제거되고 수정되어 지식베이스에 첨가될 수 있다. 그러나 이 새로운 규칙이 첨가됨으로써 지식베이스의 기존의 규칙들을 더욱 간소화시킬 수 있는지를 먼저 점검한 후 새로운 규칙이 첨가되도록 한다.

단계 6 : 새로 삽입될 규칙의 실행부가 다른 규칙의 조건부라면 새로운 규칙을 삽입함으로써 기존의 규칙들이 모순을 일으키지 않는지를 점검한다. 즉 구조적 및 의미적 모순을 제거하는 것이다. 이 때 모순이 발견되면 전문가에게 알린다. 또한 첨가할 규칙 R의 Set(R) $\subset$ Set(R_i)인 R_i 의 조건부를 Set R_i =(Set R_i - Set R) $\cup$ {A}로 수정하여 구조적 중복을 제거한다. 그리고 진입 차수와 진출 차수도 수정한다.

단계 7 : 새로 첨가할 규칙의 진입 차수와 진출 차수를 구한다.

단계 8 : 규칙의 진입 차수가 0인 규칙은 도달될 수 없는 절이고 더 이상 연결되지 않는 절들은 네트워크 구조에서 진출 링크가 없는 경우이다.

새로운 규칙은 위의 8단계를 모두 거친 후 지식 베이스에 첨가된다.

4.3.3 규칙의 제거

규칙의 제거는 규칙 R은 물론 R의 선행자의 링크까지 제거되는 것이다. 이때 도달될 수 없는 규칙과 더 이상 연결되지 않는 규칙이 나타나면 전문가에게 알려 수정하도록 한다.

4.3.4 지식 관리 시스템의 실행 예

구축된 지식 관리 시스템에 의해 다음과 같은 규칙을 첨가시키고 실행시켜 나타난 시스템의 처리 결과는 다음과 같다.

R_1 : P_1 , P_2 → A_1

R_2 : P_1 , P_3 , P_4 → A_2

R_3 : P_1 , P_5 , P_6 → A_3

R_4 : A_1 , P_7 , P_8 → A_4

R_5 : A_3 , A_4 , P_9 → A_5

지식베이스에 위와 같은 지식이 있고 다음과 같이 차례로 규칙을 첨가해본다.

① R_6 : P_1 , P_2 , P_7 → A_6의 첨가 : R_6는 다음과 같이 변경되어 첨가되고 R_4도 변경된다.

R_6 : A_1, P_7 → A_6

R_4 : A_6 , P_8 → A_4

② R_7 : A_3 , P_5, P_9 → A_7의 첨가 : R_7는 다음과 같이 변경되어 첨가되고 R_5도 변경된다.

R_7 : A_3 , P_9 → A_7

R_5 : A_4 , A_7 → A_5

③ R_8 : A_4 , P_{10} → A_1의 첨가 : 이 경우 첨가 단계1에 의해 A_4는 A_1, P_7, P_8로 대치되므로 R_8은 A_1, P_7, P_8 P_{10} → A_1 되나 단계 2에서 조건부와 실행부에 A_1이 같이 존재하므로 순환이라 명시하며 첨가되지 않는다.

④ R_9 : P_{11} → A_8의 첨가 : 수정 없이 첨가된다.

⑤ R_{10} : ~P_{12} → A_9의 첨가 : 수정 없이 첨가된다.

⑥ R_{11} : A_9, P_{12}, P_{13} → A_{10}의 첨가 : R_{10}에 의해, R_{11}은 ~P_{12} , P_{12} , P_{13} → A_{10}이 된다. 이때 조건절에 P_{12}와 ~P_{12}가 있는 경우이므로 모순임을 알리며 첨가되지 않는다.

⑦ R_{12} : P_1 , P_5 , P_6 → A_{11}의 첨가 : R_3와 R_{12}의 조건절이 같아서 A_3과 A_{11}의 관계를 점검한 후 모순이 발견되지 않으므로 지식 베이스 내에서의 모순이 없음을 알리며 받아들일 것인지를 묻는다. 받아들이기를 원하면 A_3 → A_{11}로 저장된다.

⑧ R_{13} : P_1 , ~P_8 → A_1의 첨가 : P_8과 P_2의 관계가 모순관계는 아니지만, R_{13}을 삽입하면 첨가과정의 단계 6에서 R_4 : A_6 , P_8 → A_4가 모순인 규칙으로 발견되므로 시스템은 지식의 제공자에게 R_{13}을 첨가할 경우 R_4가 모순으로 변함을 알리고 R_{13}을 삽입할 것인지를 묻는다.

위 예의 각 과정에서 더 이상 연결되지 않는 규칙과 도달될 수 없는 규칙의 리스트를 보여준다. ⑧의 첨가가 끝난 후, R_2, R_5, R_9, R_{10}은 더 이상 연결되지 않는 규칙이고 진입 차수가 0인 R_1, R_2, R_3, R_9, R_{10} 등은 입력 변수가 아니면 도달될 수 없는 규칙임을 알린다.

4.4 지식 관리 시스템의 성능 분석

이 논문에서 제시한 지식 관리 시스템의 기본 알고리즘의 시간 복잡도에 영향을 미치는 요인은 다음과 같다.

n : 지식베이스내의 규칙의 수
l : 확실한 특성의 리스트의 수
m : 가능한 특성의 리스트의 수
k : 하나의 규칙 내에서 조건부의 최대 수
h : 하나의 리스트에서 항의 최대 수

다음은 최악의 경우를 고려하여 분석해 본다. 즉 새로운 규칙이 모든 지식에 영향을 미친다고 가정하자. 이때 각 단계의 비교 횟수는 다음과 같다.

1단계 : $O(n \cdot k)$
2단계 : $O(k^2 \cdot h)$
3단계 : $O(k \cdot h)$
4단계 : $O(n \cdot k)$
5단계 : $O(n) + O(k^2 \cdot h)$
6단계 : $O(n) + O(k^2 \cdot h) + O(n \cdot k)$
7단계 : $O(n \cdot k)$이다.
8단계 : $O(n)$이다.

따라서 8단계까지 각 단계를 모두 합한 비교 횟수는 $O(4nk + 3k^2h + kh + 3n) = O(n(4k+3) + kh(3k+1)) \approx O(nk+k^2h)$이며 k 와 h는 보통 작은 상수이다. 그러나 기존의 쌍 검증 방식에서는 일관성 검증을 위해 전방추론 혹은 후방추론을 해야 하므로 최악의 경우 지수승의 특성을 보인다(Brisoux 2001).

이와 같이 지수승의 성능을 보이는 NP-복잡성 문제인 일관성 검증 문제를 본 시스템은 $O(nk+k^2h)$정도의 효율적인 검증 절차를 거쳐 오류가 없을 경우에만 지식의 삽입과 제거 연산을 수행한다.

5. 지식관리시스템을 수반한 전문가 시스템 개발 도구의 구현 및 예제

시스템은 C++언어와 Oracle SQL을 사용하여 구축되었다. 그리고 15개의 프레미스와 8개의 규칙으로 구성된 지식을 사용하여 테스트해 보았다.

5.1 프레미스

다음은 스트링의 집합이다. 11개의 레코드가 있으며 첫 번째 필드는 순번이다. 두 번째 필드의 값 중 i는 initial, f는 final, n은 그 이외의 노드를 가리킨다. 세 번째 필드의 값 중 1은 질문할 때 isn't it?, 2는 what is, 3은 doesn't it을 붙인다.

```
11
1i1 This is a flower.
2n2 the color(red, blue, green).
3n2 the number of petal.
4n1 The number of petal is known.
5n2 the result of calculation.
6i3 It has roots.
7f1 This is a rose.
8f1 This is a lily_of_the_valley.
9n2 the name of flower.
10n1 The calculation is known.
11n1 The color is known.
```

다음은 프레미스로서 15개가 있다. 각 프레미스의 첫 번째 필드는 순번이고 둘째 필드는 스트링의 번호이다. 셋 째 필드의 값 중 t는 true, s는 string, x는 절차부가, i는 데이터베이스 참조를 나타낸다. 넷째 필드는 연산자를 나타내며 값 중 y는 리턴 값이 있는 경우이고 n은 리턴 값이 없는 경우를 나타낸다. 5번째 필드는 값을 나타낸다.

```
15
1  1 t
2  2 s = red
3  4 s = known
4  5 x y calcule_f
5  3 n > 0
6  5 n > 0
7  6 t
8  7 t
9  2 s = blue
10 8 x n print(lily_of_the_valley)
11 9 x y demand_name_of_type
12 2 i y r(flower, 1 a_flower, 1 a_name:s:v:
   t9, a_color)
13 10 t
14 11 t
15 8 t
```

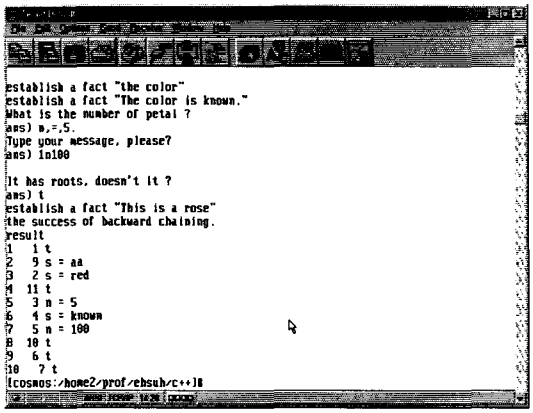
5.2 규칙

다음은 전문가에 의해 제공된 지식이다. 앞에서 정의한 프레미스 들을 이용하여 규칙을 정의한다.

```
rule1 (1.0) k1 if 1t 14t 2t 3t then 4t
rule2 (1.0) k1 if 5t then 3t
rule3 (1.0) k1 if 13t 6t 7t then 8t
rule4 (1.0) k1 if 4t then 13t
rule5 (1.0) k1 if 1t then 11t
rule6 (1.0) k1 if 11t then 12t
rule7 (1.0) k1 if 12t then 14t
rule8 (1.0) k1 if 1t 14t 9t 3t then 10t
```

5.3. 실행

다음은 후방 추론을 통해서 추론을 수행하는 과정을 예를 통하여 보여준다. 이 예에서는 8t를 목표로 설정하였다. 추론을 위해 시스템은 사용자에게 질문을 하고 ans)는 사용자가 대답한 부분이다.



<그림 5> 시스템의 실행 예

6. 결론

본 논문은 효율적이고 신뢰성있는 전문가 시스템을 구축하는 도구를 설계하고 구현하였다. 즉, 이 시스템을 이용하여 사용자가 특정분야의 지식을 저장하기만 하면 그 분야의 전문가 시스템이 구축된다. 이 때 추론된 지식의 신뢰도를 높이기 위해서는 저장된 지식의 정확성과 일관성이 유지되어야 한다. 따라서 본 논문에서는 전문가 시스템 구축 도구에 지식 관리 시스템을 첨가했다. 지식 제공자가 지식을 저장할 때 저장될 지식은 기존의 지식 베이스에 있던 지식과의 중복, 모순, 순환, 도달되지 않는 규칙과 더 이상 연결되지 않는 규칙 등의 오류를 구조적 및 의미적 측면에서 판별한 뒤 오류가 없는 경우에만 지식 베이스에 첨가된다. 지식 관리 시스템은 오류 판별 시 확실한 특성의 리스트와 가능한 특성의 리스트를 사용하고 개선된 쌍 검증 방식을 이용함으로써 기존의 쌍검증 방식보다 시간적인 면에서 더욱 효율적일 뿐 만 아니라 오류 점검 단계에서 오류 가능성을 대폭 확대 점검하여 정확한 오류 점검을 수행한다. 또한 다양한 지식을 이용하도록 절차적 지식과 데이터베이스의 선언적 지식을 첨가, 제거 및 호출할 수 있도록 하였다. 이 도구를 특정분야에 적용함으로써 이 도구의 실용성을 확보하는 것이 향후 과제이다.

참고문헌

박창현, 유석인, "지식 기반 시스템에서의 추론 Browser의 설계", *한국정보과학회논문지*, vol19, no.1 (1992), 80-93.

Botten, N., A. Kusiak and T. Raz, "Knowledge Bases : Integration, Verification and Partitioning", *European j. Operational Research*, vol 42(1989), 111-128.

Brisoux, L., Eric Gregoire, Lakhdar Sais, "Checking Depth-Limited Consistency and Inconsistency in Knowledge-Based Systems", International Journal of Intelligent Systems, vol. 16 (2001), 319-331.

Cragun, B. and H. Steudel, "A Decision-Table Based Processor for Checking Completeness and Consistency in Rule-Based Expert Systems", *Int'l j. Man-Machine Studies*, vol 26, no. 5(1987), 633-648.

Cyre, W. R., "Capture, integration, and analysis of digital system requirements with conceptual graphs", *IEEE Trans. on Knowledge and data engineering*, vol 9, no.1(1997), 8-23.

Eick, C. F., "Rule-Based Consistency Enforcement for Knowledge-Based systems", *IEEE Trans. on Knowledge and data engineering*, vol 5, no.1(1993), 4-13.

Ellis, G., "Compiling Conceptual Graphs", *IEEE trans. on Knowledge and data engineering*, vol 7, no 1(1995), 68-81.

Koczkodaj, W. W., "A New Definition of

Consistency of Pairwise Comparisons", Mathematical and Computer Modelling, vol 18, no.7(1993), 79-84.

Koczkodaj, W. W., Michael W. Herman, "Using Consistency-driven Pairwise Comparisons in Knowledge-based Systems", Proceedings of the 6th International Conference on Information and Knowledge Management, (1997), 91-96.

LeBeux, P., D. Fontaine, "Un systeme d'acquisition des connaissances pour systemes experts", *Technique et Science Informatique,* vol 5, no.1(1986), 7-20.

Liu, N. K., "Formal Verification of Some Potential Contradictions in Knowledge Base Usion a High Level Net Approach", *Applied Intelligence*, vol.6, no. 4(1996).

Morizet-Mahoudeaux, P., "Maintaining Consistency of Database During Monitoring of an Evolving Process by a Knowledge-Based System", *IEEE Trans. on systems, man and cybernetics*, vol 21, no. 1(1991), 47-60.

Nazareth, D. L., "Issues in the Verification of Knowledge in Rule-Based Systems", *I.J.Man-Machine Studies*, vol. 30(1989), 255-271.

Nazareth, D. L., and M.H.Kennedy, "Verification of Rule-Based Knowledge Using Directed Graphs", *Knowledge Acquisition*, vol.3(1991), 339-360.

Nazareth, D. L., "Investigating the Applicability of Petri Nets for Rule-Based System Verification", *IEEE Trans. on Knowledge and data engineering*, vol 5, no. 3(1993), 402-415.

Nguyen, T. A., N.A.Perkins, T.J.Laffey and D.Pecora, "Checking an Expert system knowledge base for consistency and completeness", *Proc. 9th IJCAI*, LA, Califonia(1985), 375-378.

Nguyen, T. A., W.A.Perkins, T.J.Laffey and D.Pecora, "Knowledge Based Verification", *AI Magazine,* vol 8, no 2(1987), 69-75.

Popescu, I., "Combining Constraints and Consistency Techniques in Knowledge-Based Expert System", Lecture notes in computer Science, issue, 2000, 191-200

Ramaswamy, M., S. Sarkar, Ye-Sho Chen, "Using directed hypergraphs to verify rule-based expert systems", *IEEE Trans. on Knowledge and data engineering*, vol. 9, no. 2(1997), 221-237.

Riley, G., Expert Systems : Principles and Programming, PWS Publishing Company ed., (1997).

Valiente, G., "Verification of knowledge based redundancy and subsumption using graph transformations", *Int'l J. Expert System*, vol.6, no.3(1993), 341-355.

Abstract

A Tool for Implementation of Expert System with Knowledge Management System

Euy-hyun Suh*

This paper proposes and implements a tool for the development of efficient and reliable expert system. In the expert system the inference is executed, based on the knowledges stored in the knowledge base of specific domain. To acquire the reliable results of inference, the expert system requires the facilities which can access the various kinds of knowledge and maintain the consistency and accuracy of knowledge. In this context this paper implemented the knowledge management system which maintains the consistency and accuracy of knowledge, adding selectively the knowledges without error to the knowledge base by verifying their error before the knowledges are added to the knowledge base. At the same time this paper made the expert system call and use the procedural knowledge and the declarative knowledge in the data base so that it might use the various kinds of knowledge in the process of inference.

Key words : 전문가시스템, 지식관리 시스템

* Mokwon University, Dept. of Computer Engineering