2000년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

생성 규칙으로 표현된 지식의 일관성 유지 시스템

서의현
*목원대학교 컴퓨터공학과
e-mail : ehsuh@mokwon.ac.kr

A System for Maintaining Consistency of Knowledge Represented as Production Rule

Euy-Hyun Suh
Dept. of Computer Engineering, Mokwon University

요 약

지식기반시스템에서 지식의 일관성을 유지하는 것은 신뢰성 있는 추론 결과를 위해서 필수적인 요소이다. 추론은 지식을 기반으로 행해지기 때문이다. 따라서 본 논문에서는 새로운 지식이 지식베이스에 첨가되기 전에 기존의 지식베이스의 지식들과 첨가될 지식과의 일관성 여부를 검사한 후 일관성이 유지될 때만 지식이 첨가되는 시스템을 구축하였다.

1. 서론

지식 기반 시스템에서 가장 많이 사용되는 생성 규칙으로 표현된 지식은 독립적이며 지식의 양은 방대한 경우가 많기 때문에 지식 제공자(전문가)나 지식공학자가 내용 전체를 파악하기가 어렵고, 전문가는 때때로 직관적으로 생각하므로 추론에서의 많은 과정을 생략한 채 지식을 지식베이스에 삽입할 수 있다. 또한 지식베이스는 점진적인 방식으로 오랜 기간에 걸쳐 구축되고 여러 명의 전문가들이 지식을 함께 저장하기 때문에 지식의 중복, 순환과 모순의 문제가 야기될 수 있다. 이러한 문제들을 해결하기 위해서는 지식기반시스템이 지식베이스의 지식을 사용하기 전에 정확성을 검증하여 발견된 오류를 자체 수정하거나 자체 수정이 불가능할 경우 지식 제공자에게 문의하여 수정하는 시스템이 요구된다.

따라서 본 논문은 지식이 지식베이스에 첨가될 때 지식의 정확성 및 일관성을 보장하기 위한 일관성 유지 시스템을 제안하였다. 지식의 일관성 유지 시스템에서는 확실한 특성의 리스트와 우발적 특성의 리스트를 생성하고 검증 방법 및 단계를 개선하는 형태의 보완한 쌍 검증 방식(pairwise checking)[7][9]이 사용되었다. 특히, 본 시스템은 첨가하려는 지식의 중복, 모순, 순환의 일관성 오류는 물론 도달될 수 없는 규칙과 더 이상 연결되지 않는 규칙 등의 오류를 수정한 뒤 지식을 첨가한다.

일관성을 유지하기 위한 방법들을 2 장에서 설명한 후 3 장에서는 지식 베이스와 지식 컴파일러 등의 시스템 구성요소를 소개한다. 4 장에서는 제안한 일관성 유지 시스템에 대해 자세히 설명하고, 결론에서는 일관성 유지 시스템을 구축함으로써 확인된 결과를 요약한다.

2. 관련 연구

신뢰성 있는 시스템을 구축하기 위하여 지식 검증을 위한 많은 시도가 있었다.

전통적인 방법으로 의존 관계표를 이용하여 일관성 및 완결성을 점검하는 방법[7]이 있는데 이는 열거 방식으로 처리하기 때문에 시간이 많이 걸리는 단점이 있다.

결정표(Decision Table)를 사용하는 방식에서는 간혹 오류의 존재만을 발견하고 오류의 위치를 정확하게 찾아내지 못하는 단점이 있다. 또한 복잡한 관계를 가지는 규칙들은 추론 과정 중 연결된 규칙이 많아 지수 함수적인 성능을 나타낸다[6].

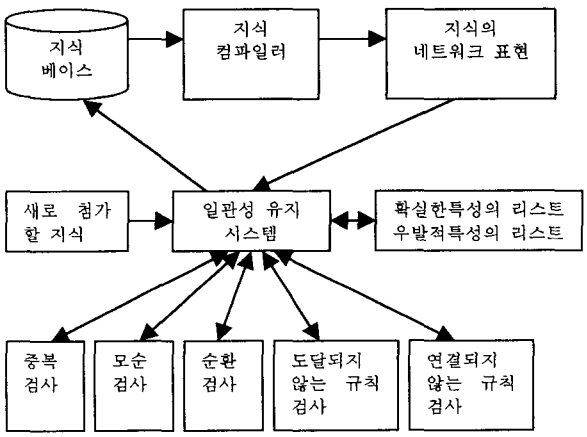
쌍 검증(Pairwise Checking) 방식에서는 생성된 절들

이 적용 가능한 규칙들과 비교된다[3,5]. 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[4].

그래프 표현 방식은 프레미스의 개념적인 의존 관계를 나타내기 쉬운 표현법으로서 방향 그래프(Directed Graph), 추론 그래프(Inference Graph), 하이퍼 그래프(Hypergraph)등의 방법이 있다. 그러나 이러한 방법들은 복합절과 단순절 사이의 의존 관계를 나타내기 어렵다[8]. 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)[8]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 구조로서 지식을 체계적으로 분류하여 계층적으로 저장하고 편집 방법을 통해 중복을 제거한다[1,2]. 그러나 이 방식은 표현의 다양성 때문에 극히 제한적인 일관성만을 검진할 뿐 여러 곳에 있는 비 일관성 오류를 점검하기 어렵다.

3. 시스템 개요

지식의 일관성 유지 시스템은 [그림 1]과 같이 구성된다.



[그림 1] 일관성 유지 시스템

3.1 지식 베이스

지식베이스의 기본 구조는 생성 규칙이다. 생성규칙은 규칙들을 모듈화하여 나타낼 수 있고, 비절차적 방식으로도 규칙들로서 저장된 지식을 사용할 수 있을 뿐 아니라 여러 종류의 지식을 쉽게 표현할 수 있기 때문이다. 지식은 조건 부분과 실행 부분으로 표현되며 일반 지식과 절차적 지식도 같은 방식으로 나타낼 수 있도록 하였다. 규칙의 조건 부분은 연언 연산자(and)를 사용하여 여러 개의 프레미스를 나타낼 수 있고 실행 부분은 단 하나의 프레미스로만 표현될 수 있다. 이러한 생성 규칙들의 실행 효율을 증가 시키기 위하여 내부 표현 구조는 네트워크 구조를 사용하였다.

생성 규칙 표현법의 장점들 중 하나는 다양한 지식을 표현할 수 있다는 것이다. 지식 기반시스템은 필요한 경우 함수를 호출하여 비결정적인 값을 계산할 수 있는 절차 부가의 개념을 사용할 수 있어야 한다. 예를 들면 다음과 같다.

```
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 0 개 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.

지식 기반 시스템에서 추론 엔진은 특정한 영역의 지식과 사실(fact)들을 가지고 추론을 행한다. 그러나 경우에 따라서는 사실의 양이 많아지고 데이터 베이스에 저장되어 있기 때문에 데이터 베이스 내에 저장된 사실들을 호출해야 할 경우가 있다.

따라서 본 시스템은 데이터 베이스 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 지식 컴파일러와 지식의 네트워크 표현

지식 컴파일러는 지식기반시스템이 실행되기 전에 지식을 네트워크의 내부 표현 방식으로 컴파일한다. 네트워크 표현 방식은 하나의 규칙 내부에 이 규칙과 연결되어지는 규칙의 정보를 포함하고 있다. 이 방식은 추론이 실행될 때 이러한 정보를 이용함으로써 모든 지식을 검색할 필요 없이 꼭 필요한 지식만 검색하여 추론의 속도를 증가시킬 수 있다.

또한 네트워크로 표현함으로써 도달될 수 없는 규칙, 더 이상 연결되지 않는 규칙들을 쉽게 찾아낼 수 있는 장점을 가지기 때문이다. 지식은 오류를 점검하는 편집 과정을 거쳐 오류가 없으면 네트워크 구조에 추가되며 최종적으로 지식베이스에 저장된다.

3.3 지식의 일관성 유지시스템

일관성 유지 시스템은 추론 결과의 신뢰도를 높이기 위해 새로운 지식을 지식베이스에 추가하고자 할 경우 지식이 추가되기 전에 정확성을 검증한다. 새로 첨가할 지식에 만약 오류가 존재한다면 일관성 유지시스템은 오류를 자체 수정하거나 자체 수정이 불가능할 경우 전문가에게 문의하여 오류를 수정하도록 한 후 지식이 첨가되도록 하는 시스템이다.

3.4 확실한 특성의 리스트와 우발적 특성의 리스트

확실한 특성이란 만약 프레미스 A 의 실행이 프레미스 B 의 실행을 내포하면 B 는 A 의 확실한 특성이라 정의한다.

우발적 특성은 일어날 수도 있고 그렇지 않을 수도 있는 것으로 만약 프레미스 A 의 실행이 프레미스 B 의 실행에 기여한다면, A 는 B 의 우발적 특성이라 정의한다.

4. 지식의 일관성 유지 시스템

일관성 유지시스템에서는 확실한 특성의 리스트와 우발적 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.

특히 확실한 특성의 리스트에는 일반적인 지식이나 상식 등도 포함될 수 있어서 의미적 오류도 찾아 낼 수 있다. 본 논문은 부정의 프레미스를 포함한 네트워크로 지식을 표현하고 보완된 쌍 검증 방식을 이용하여 구조적 측면에서 뿐만 아니라 의미적 측면에서도 중복, 모순, 순환의 일관성 오류는 물론 도달될 수 없는 규칙과 더 이상 연결되지 않는 규칙 등의 완결성 오류를 찾아내어 자체 수정하거나 자체 수정이 불가능할 경우 전문가에게 수정하도록 메시지를 전달하는 일관성 유지시스템을 제시, 구축하였다.

4.1 지식의 오류

(1) 중복
- 두개의 규칙이 같은 경우.
- 한 규칙의 조건 부분이 다른 규칙의 조건 부분의 부분 집합이고 실행부가 같을 경우.
- 한 규칙의 실행부가 다른 규칙의 조건부에 있으면서 두 규칙의 조건부에 같은 프레미스가 1 개 이상인 경우
- 하나의 규칙 안에 프레미스와 그 프레미스의 확실한 특성이 같이 있는 경우.
- 한 규칙의 조건부분이 다른 규칙의 조건부와 같거나 부분 집합이고 한 규칙의 실행부가 다른 규칙의 확실한 특성일 경우.

(2) 모순
- 두 규칙의 조건부가 같고 실행부가 모순인 경우. 모순이란, 한 실행부가 다른 실행부의 확실한 특성이나 우발적 특성과 반대인 경우,
- 한 규칙의 조건 부분이 다른 규칙의 조건 부분의 부분 집합이고 실행부가 모순일 경우.
- 하나의 규칙 안에 프레미스와 그 프레미스의 확실한 특성의 반대의 의미를 가진 프레미스가 같이 있는 경우.

(3) 순환
- 하나의 규칙의 조건부에 프레미스를 첨가하려고 하는데 실행부가 첨가하려는 프레미스의 우발적 특성일 경우.

(4) 더 이상 연결되지 않는 절
- 규칙의 실행부가 목표가 아니고 다른 규칙의 조건부분에 포함되어 있지 않는 경우.

(5) 도달될 수 없는 절
- 한 규칙의 조건부가 입력 변수가 아니고 다른 규칙의 실행부도 아닐 경우.

4.2 새로운 지식의 첨가 과정

새로운 지식이 첨가될 때 일관성 유지 시스템은 다음과 같은 절차로 실행된다.

1. 새로 첨가할 지식을 확실한 특성의 리스트와 우발적 특성의 리스트들을 이용하여 확장한다.
2. 확장된 지식의 조건 부분간에 혹은 조건부분과 실행부 간에 모순과 순환의 오류를 검사한다.
3. 확장된 규칙의 프레미스 중 불가피한 프레미스만 남겨두고 제거한다. 이 때 불가피한 프레미스란 새로 첨가할 지식에서 확실한 특성의 리스트에 없는 프레미스를 말한다.
4. 지식베이스 내의 규칙들 중 새로 첨가하려는 규칙의 조건 부분과 같은 조건 부분을 가진 규칙과 비교하여 모순이나 중복이 없는지 검사한다.
5. 실행부가 같은 규칙들과 비교하여 모순이나 중복을 검사한다.
6. 새로운 규칙이 첨가됨으로써 기존 규칙이 변화되어야 하는 경우를 검사하기 위해 각 지식에 대해 다음을 수행한다.
   - 검사할 지식의 프레미스를 확실한 특성의 리스트와 우발적 특성의 리스트들을 이용하여

확장한다.
- 확장된 지식의 조건부분간에 혹은 조건부분과 실행부 간에 모순과 순환의 오류를 검사한다.
- 확장된 규칙의 프레미스 중 불가피한 프레미스만 남겨두고 제거한다.

7. 새로 첨가할 규칙의 진입차수와 진출차수를 구한다.
8. 도달될 수 없는 절과 더 이상 연결되지 않는 절들을 찾아낸다.

5. 결론

본 논문은 지식 기반 시스템의 일반적 표현 방법인 생성규칙의 일관성을 유지시키기 위한 방법을 제시하고 시스템화 하였다. 이는 전적으로 지식제공자에게 의존하던 지식의 신뢰도 문제를 일관성 유지 시스템이 담당하게 한다. 즉 확실한 특성의 리스트와 우발적 특성의 리스트를 이용하여 기존의 지식과의 중복, 모순, 순환, 도달되지 않는 규칙과 더 이상 연결되지 않는 규칙 등의 오류를 판별한 뒤 오류가 없는 경우에만 새로운 지식이 지식베이스에 첨가되도록 함으로써 지식의 일관성을 유지하여 추론 결과의 신뢰도를 높일 뿐 아니라 오류판별이 자동적으로 행해지므로 맨-머쉰 인터페이스의 기능이 향상되었다.

참고문헌

[1]Walling R. Cyre, "Capture, integration, and analysis of digital system requirements with conceptual graphs," IEEE Trans. on Knowledge and data engineering, Vol.9, No.1, pp.8-23, jan-feb, 1997.

[2]Gerard Ellis, "Compiling Conceptual Graphs," IEEE trans. on Knowledge and data engineering, Vol.7, No.1, pp.68-81, feb, 1995.

[3]P. LeBeux, D. Fontaine, "Un systeme d'acquisition des connaissances pour systemes experts," Technique et Science Informatique, Vol.5, No.1, pp.7-20, 1986.

[4]N.K.Liu, "Formal Verification of Some Potential Contradictions in Knowledge Base Usion a High Level Net Approach," Applied Intelligence, Vol.6, No.4, Oct, 1996

[5]P.Morizet-Mahoudeaux, "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, pp.47-60, 1991.

[6]D.L.Nazareth, "Issues in the Verification of Knowledge in Rule-Based Systems," I.J.Man-Machine Studies, Vol.30, pp.255-271, 1989.

[7]T.A. Nguyen, N.A. Perkins, et al,"Checking an Expert system knowledge base for consistency and completeness," Proc. 9th IJCAI, LA, Califonia, pp.375-378, 1985.

[8]M. Ramaswamy, 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, march-april, pp.221-237, 1997.