論文92-29B-1-3

네트워크를 이용한 지식베이스시스템 규칙들의 중복 및 모순검출에 관한 연구

(A Network Approach to Check Redundancies and Inconsistencies of Knowledge-Based System Rules)

崔 成 鎬*, 朴 忠 植**, 金 在 熹**, 申 東 弼***

(Sung Ho Choi, Choong Shik Park, Jaihie Kim, and Dong Pil Shin)

要 約

본 논문은 지식베이스내의 중복과 모순등을 검증하여 일관성있는 지식베이스를 구성하도록 도와주는 규칙검증기를 제안하였다. 제안된 방법은 규칙간의 연결상태를 네트워크로 나타내어 규칙들을 검증하는 것이다. 이 방법에서 채택한 규칙의 표준모델은 NOT을 포함한 Conjuntive Normal Form의 형태로서, 기존의 전문가시스템의 규칙은 별도의 변환기에 의하여 표준모델로 변환시킨 뒤 검증할 수 있다. 기존의 방법중에서 개념적으로 가장 유사한 Ginsberg의 KB-reducer와 비교실험한 결과, 규칙이 360개 정도일 때 검증속도가 약 3배정도 향상되었으며 규칙의 갯수가 늘어날수록 속도차이가 커지는 것으로 확인되었다. 메모리 소비는 제안된 방법이 규칙이 360개 정도일때 1.2배 정도가 더 소비되었다. 이밖에도 순환규칙의 검증, 중복과 모순이 발생한 경로를 쉽게 찾을 수 있다는 장점을 가지고 있다.

Abstract

In this paper, a rule checker which aids in composing a consistent knowledge base by checking redundancies and inconsistencies in a knowledge base is proposed. The proposed algorithm checks the rules by representing the rule connections as a network. The standard model of the rules adapted in this algorithm is in the Conjunctive Normal Form which includes NOT's, and rules of conventional expert system can be checked by converting them into the standard form by a rule form at converter. When compared with Ginsberg's KB-reducer which is conceptually most similar to the proposed algorithm among existing methods, it is shown by a computer simulation that with 360 rules, the checking time is three times faster and the rate increased as the number of rules increased, but the total memory requirement of the proposed agorithm is 1.2 times larger. The proposed algorithm has further advantages in that it can check circular rule chains and can find the paths of the redundant and inconsistent rules.

*正會員, 韓國通信
(Korea Telecom.)
**正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.)
***正會員, 韓國科學技術院 시스템工學센터
(Division of System Eng., KIST)
接受日字 : 1991年 10月 30日

I. 서 론

전문가 시스템은 여러 응용분야에서 그 가치를 인정받고 있으며 많은 전문가시스템이 개발되었다. 실제 전문가의 지식을 컴퓨터에 맞게 지식베이스로 구성하는 일은 시간과 비용이 많이 소비된다. 이를 지식획득(knowledge acquisition)이라 하고 전문가 시스

템을 개발하는데 있어서 어려운 문제로 대두된다. 한편 지식획득의 과정이나 이후의 또다른 문제점은 지식개선(knowledge refinement)이다. 지식베이스는 많은 지식이 삭제, 첨가 또는 수정되면서 구성된다. 이에 따라 이전에 있던 지식과 새로 수정된 지식은 서로 중복이나 모순이 생기게 될 수도 있다. 지식베이스의 규모가 커질수록 이러한 오류가 생길 가능성은 더욱 커지게 된다. 이러한 오류를 검증하는 방법으로써 CHECK[1], ARC[2], ONCOCIN 시스템의 규칙검증기[3], KB-reducer[4] 등 여러가지 방법이 제시되었다.

ONCOCIN시스템의 규칙검증기와 CHECK는 여러 규칙의 연결 즉, 규칙사슬(rule chain)에의 한 중복이나 모순의 검색이 용이하지 않고 지식베이스의 규모가 커질수록 복잡해진다. 이를 위하여 ARC와 KB-reducer가 제시되었다. ARC는 CHECK를 좀더 발전시킨 방법으로 규칙들간의 전제부와 결론부들을 비교한 여러가지 table을 사용하여 규칙들을 검증하였다. 그러나 이들 table은 전체 지식베이스에 대한 모든 규칙들의 전제부와 결론부를 서로 비교하여 구성되므로 불필요한 시간과 메모리가 많이 소비되어 검증이 비효율적이게 된다[2]

KB-reducer의 경우에는 ATMS[5]의 label에 대한 개념을 이용하여 검증하였다. 이는 각 규칙들을 만족시킬 수 있는 조건들을 조사한 뒤 이들에 대한 비교에 의한 것이다. 모순되는 결론을 내린 규칙들이 같은 조건을 가지고 있으면 모순으로 검증하고, 같은결론을 내린 규칙들이 같은 조건을 가지고 있으면 중복으로 검증한다. KB-reducer는 ARC에 비하여 보다 체계적으로 접근하였으나, 다음과 같은 문제점을 가지고 있다. 첫째, 규칙의 레벨(level)을 정하여 레벨(lebel)에 따라 규칙을 재정돈 해야한다. 둘째, 중복이나 모순을 검출할 수가 있으나 이들이발생한 경로를 찾기는 힘들다. 셋째, 중복과 모순을 동시에 검증하지 않고 중복검증이 완전히 끝난후에 모순검증을 한다. 네째, 실제 실용적 규칙의 형태를 고려한 것이 아니라 단순한 개념적 형태에서만 고려되었다[4]

본 논문은 지식베이스를 규칙간의 연결상태를 잘 나타내는 데이타구조인 네트워크로 변환하여 검증함으로써, KB-reducer의 문제점을 보완하고 검증속도를 향상시킬 수 있는 방법을 제시하였다. 중복과 모순검증은 KB-reducer에서와 마찬가지로 literal들을 만족시킬 수 있는 조건인 label을 결정해 나가면서 하위 literal 노드의 label은 상위 노드로부터 물려 받아 결정이 된다. 이때 하나의 literal에 대하여 여러 경로를 통해서 내려오는 label들 중 겹치는 경우에는 중복으로 검증하고, 서로 모순되는 literal의 label을 비교하여 겹치는 부분이 있으면 모순으로 검증한다.

한편, 본 논문에서는 특정한 전문가 시스템 개발도구를 설정하지 않고 생성시스템(production system)의 보편적인 형태인 O-A-V(object-attribute-value)를 literal로 하는 표준모델을 설정하였다. 이 표준모델에서는 불확실정도를 나타내는 확신율(certainty factor)은 배제하였으나, 제시된 규칙검증 방법은 확신율을 고려한 지식베이스에서도 적용 가능하다. 그리고 특정한 전문가 시스템 개발도구에 의해 구현된 지식베이스도 표준모델로 변환하여 검증할 수 있다.

Ⅱ. 규칙의 중복, 모순 및 순환

지식베이스에 있는 규칙들간에 존재할지 모르는 논리적인 오류를 검증하기 위해서 먼저 오류의 유형을 정의해야 한다. 오류의 유형은, 같은 지식을 중복으로 표현한 규칙(redundant rule), 지식이 서로 모순이 생기는 규칙(inconsistent rule), 연관된 지식이 무한루프를 이루는 규칙(cycle rule)등 세가지로 정의하였다.

두개 이상의 규칙이나 규칙사슬(rule chain)이 동일한 지식을 표현하여, 그중 하나를 제거하더라도 결론을 내리는데 영향을 주지 않으면, 이를 중복(redundancy)이라 정의한다. 예를들어 다음과 같은 규칙들의 집합에서

A & B −>P
A−>C, C−>P
A−>D, D−>P

첫번째 규칙과 두번째 규칙사슬을 제거하더라도, A가 참(true)이면 P라는 결론을 내릴 수 있으므로 같은 지식이 중복으로 표현되었다. 단, 이때 두번째 규칙사슬중에 얻어지는 중도결과인 C가 다른 용도가 있다면 이 규칙사슬은 제거할 수 없다. 따라서 이 연구에서 반드시 제거되어야함을 의미하는 것은 아니나, 지식제공자에게 재검토 필요성을 제시하여 준다.

한편, 두 규칙이나 규칙사슬이 서로 모순되는 지식을 표현하여, 어떤 타당한 조건에 대해서 상충되는 결론을 동시에 내리게 되면 이를 모순(inconsistency)으로 정의한다. 예를들어 다음과 같은 규칙들의 집합에서

A & B−>P
A−>C, C−>~P

A와 B가 모두 참이면 두 규칙이 모두 만족되어, 첫번째 규칙은 P라는 사실을 결론내리게 되고, 두번째 규칙사슬은 ~P라는 사실을 결론내리게 된다. 따라서 서로 모순되는 결론인 P와 ~P를 동시에 결론

을 내리게 되므로 지식이 서로 모순되게 표현되었다.

규칙들의 연결이 루프를 형성하는 경우를 순환(cycle)이라 정의한다. 예를들면,

P−>Q
Q−>R
R−>P

세 규칙이 순환을 형성하여 무한루프로 빠지게 되며 따라서 결론을 얻을 수가 없게 된다. 또한 아래와 같은 예에서 살펴보면, 두번째 규칙의 결론부인 P가 첫번째 규칙에서는 조건부의 일부이므로, 이러한 경우도 두 규칙이 일종의 순환을 이루고 있다.

P & Q−>R
R−>P

Ⅲ. 네트워크를 이용한 검증

1. 표준모델의 정의

본 논문에서는 O-A-V 형식을 literal 단위로 하는 규칙을 표준모델로 정의하였다. 그리고 규칙의 전제부는 literal들이 conjunctive normal 형식으로 구성되고, 결론부는 조건부가 만족되었을때 유도되는 literal들로 구성된다. 그림 1은 하나의 규칙에 대한 실제 표준모델의 예를 보여주고 있다. 이 표준모델에서는 불확실성을 나타내는 확신율(certainty factor)은 생략한다. 그러나 확신율을 고려하지 않더라도 중복이나 모순이 발생할 가능성이 있는 경우는 검증할 수 있으므로, 확신율을 고려한 지식베이스에서도 적용 가능하다.

```
(R1 ($ AND ($ OR (ANYONE HAS LIFE) ($ NOT ANTONE IS DIE))
           (ANYONE IS-A GOD))
 ->
    (ANYONE IS IMMORTAL)
    (NOT ANYONE IS-A MAN))
```

그림 1. 표준모델의 규칙예
Fig. 1. An example rule of standard model.

2. 규칙형태의 단순화

지식베이스를 검증할 때, 규칙의 literal들을 간단한 심볼로 변환하여 검증하는 것이 시간과 메모리 소비를 줄이게 된다. 예로써, 그림 2(a)와 같은 실제 규칙형태를 그림 2(b)와 단순한 형태로 검증한다.

주어진 literal에 상응하는 노드를 빨리 찾기 위해서 그림 3과 같이 literal내의 object, attri bute, value를 노드단위로 하는 트리를 이용한다. 이때, 변수

```
(R1 ($ OR (PERSON IS-A CSH) (PERSON IS-A WOO))
 ->(PERSON IS-A MAMMAL))
(R 2 ($ OR (PERSON IS-A JKS) (PERSON IS-A WOO))
 ->(PERSON IS-A MAN))
(R3 (PERSON IS-A MAMMAL)
 ->(PERSON HAS LIFE)
(R4 (PERSON IS-A MAN)
 ->(PERSON HAS LIFE)
(R5 ($ AND (PERSON HAS LIFE) (PERSON NOT DIE))
 ->(PERSON IS IMMORTAL))
(R6 (PERSON IS-A MAN)
 ->($ NOT PERSON IS IMMORTAL))
```

(a) 표준모델 규칙

```
(R1 (OR N1 N2) (N3))
(R2 (OR N4 N2) (N5))
(R3 N3 (N6))
(R4 N5 (N6))
(R5 (AND N6 N8) (N9))
(R6 N5 (NOT N9))
```

(b) 단순한 심볼형태의 규칙

그림 2. 규칙형태의 단순화
Fig. 2. Simplification of rule formats.

```
literal: (BIRD CAN FLY)          :N1
         (BIRD HAS WING)         :N2
         (BIRD HAS FEATHER)      :N3
         (MAN IS-A MAMMAL)       :N4
         (?X IS-A INSECT)        :N5
         (?Y IS-A INSECT)        :N6
```

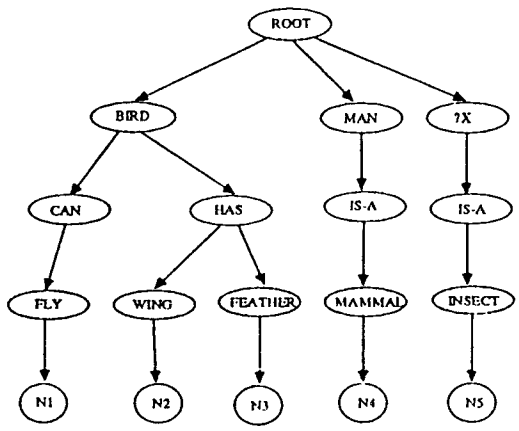


그림 3. 실제 literal들에 대한 트리구조
Fig. 3. Tree architecture for practical literal.

들은 다른 이름으로 쓰여졌더라도 동일하게 취급한다. 예를들어,(?X IS-A INSECT)과 (?Y IS-A INSECT)은 ?X와 ?Y가 모두 변수이므로 두 literal은 동일한 심볼 N5로 나타낸다.

한편 두 literal (KIM AGE 23)과 (?X AGE 23)에서 변수 ?X에 KIM이 binding될 수 있다면 (KIM AGE 23)은 (?X AGE 23)의 instance로 볼 수 있다.

따라서, 같은 조건에서 같은 두 literal이 동시에 유도될 수 있다면 중복으로 검증해야 하되 변수를 포함한 literal인 경우에는 이들의 instance에 해당되는 literal들까지 동시에 고려해야 한다. 그림 3에서 (BIRD HAS ?X)라는 literal이 새로 입력되면, 이는 tree에서 BIRD에 이어서 HAS가 매칭되지만, HAS의 하위노드인 WING과 FETHER는 ?X와 매칭되지 않고 새로운 심볼 N6로 변환되고, 변수 ?X는 WING과 FETHER를 포함할 수 있으므로, 이 literal N2와 N3가 N6의 instance로 찾아내어진다.

이와는 반대로(BEE IS-A INSECT)가 입력되는 경우 먼저 BEE에 매칭되는 노드가 없으므로 literal은 새로운 심볼인 N7으로 변환되지만, 변수인 ?X노드의 하위노드인 IS-A와 INSECT가 매칭되므로 N5의 instance로 찾아내어진다.

이와같이 변수를 포함한 literal인 경우에는 나머지 부분이 매칭되는지의 여부를 확인하여 instance를 찾아내게 된다. 이렇게 하여 literal의 심볼변환이 완전히 끝나게 되면 변수를 포함한 literal의 경우에는 이들의 instance에 해당되는 정보를 가지고 있게 된다.

3. 네트워크의 구조

본 논문에서 사용한 네트워크에는 다섯가지 종류의 노드가 있으며, 이들은 각각 다음과 같은 의미를 지니고 있다.

- ○literal노드 : 각각의 literal에 해당된다.
- ○OR노드 : 위로 묶여진 노드가 OR로 연결됨을나타낸다.
- ○AND노드 : 위로 묶여진 노드가 AND로 연결됨을 나타낸다.
- ○NOT노드 : 위로 묶여진 두개의 literal이 모순되는 관계임을 나타낸다.
- ○규칙이름노드 : 규칙의 이름을 나타낸다.

그림 4에서 아래 두 규칙이 네트워크로 변환된 형태를 보여주고 있다.

R1 : (A<B) & C-<G & H

R2 : (~A<D) & (E<F)->~G

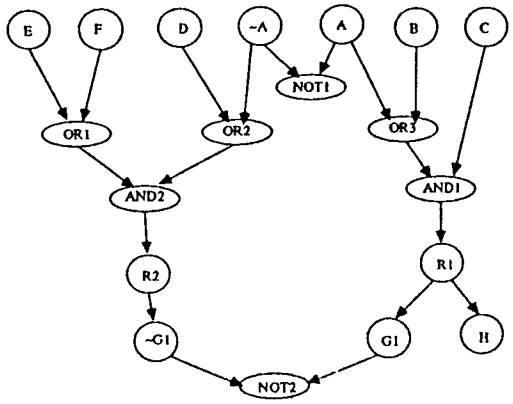


그림 4. NOT 노드를 가진 구조 네트워크
Fig. 4. A Network architecture with NOT node.

4. 지식베이스의 네트워크 구성과 순환검증

지식베이스 내의 규칙들을 검증하기 위하여, 외부화일로부터 규칙을 하나씩 읽어오면서 네트워크를 구성해 나간다. 그리고 네트워크의 구성과 동시에 순환(cycle) 검증도 행하게 된다. 순차적으로 규칙을 읽고 네트워크를 구성할 때, 순환검증을 필요로 하는 경우에는 아래의 (1)~(4) 과정을 반복하면서 네트워크의 구성과 순환검증을 병행하고, 필요로 하지 않는 경우에는 (1)~(2) 과정만 반복하여 네트워크의 구성만 행한다. 규칙들간에 순환이 있으면 시스템이 실행시 결론을 내지 못할 뿐더러, 중복과 모순 검증시 무한 루프에 빠질 수도 있다.

(1) 규칙을 읽고 2절에서 설명한 것처럼 트리를 이용하여 규칙을 단순한 형태로 바꾼다.

(2) 단순한 형태로 바꾸어진 규칙을 가지고 3절에서 논의한 바와 같은 네트워크로 구성한다.

(3) 이 때, 2규칙의 조건부를 이루고 있는 literal 노드중에서,

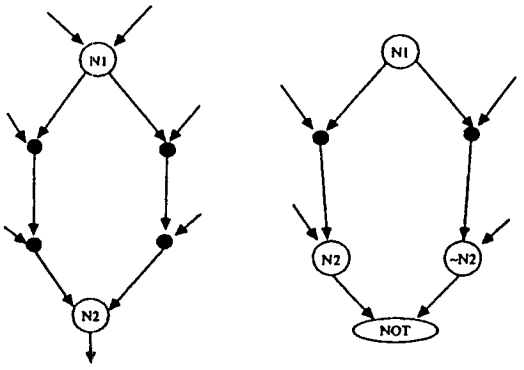
(i) 새로운 litteral 노드는 순환을 형성하지 않으므로 순환검증에서 제외하고,

(ii) 이미 구성되어 있는 literal 노드중에 속하는 literal 노드는 순환검증의 출발점으로 한다.

(4) 출발점으로부터 네트워크를 따라 옮겨 다니다가 다시 출발점을 만나게되면 지식베이스의 순환 이 검증된다.

5. 지식베이스의 중복과 모순검증

네트워크가 완전히 구성된 뒤(이는 이미 순환검증이 끝나 있음), 중복과 모순 검증이 시작된다. 네트워크에서 중복과 모순의 형태를 그림 5(a)를 살펴보



(a) 중복 (b) 모순

그림 5. 네트워크에서 중복과 모순
Fig. 5. Redundancy and Inconsistency in network.

면, N1이 참이면 서로 다른 경로를 통해서 N2가 유도될 수 있고, 그림 5(b)를 살펴보면, N1이 참이면 서로 모순되는 N2와~N2가 유도될 수 있다.

위와 같은 중복과 모순에 대한 검증을 위하여, 네트워크의 최상위 노드에서 최하위 노드까지 옮겨가면서(travelling) 각 literal들이 유도될 수 있는 조건을 조사하게 된다. 그림 6에서 살펴보면 A라는 사실은 입력 I1, &I2&I3 또는 14&15 또는 I6에 의하여 유도될 수 있음을 나타내고 있다. 이와 같이 어떤 사실을 유도할 수 있는 입력 literal들의 Disjunctive Nomral Form 형태를 ATMS에서는 label이라 정의하고 있다.[5]

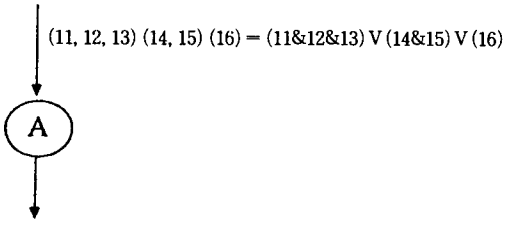
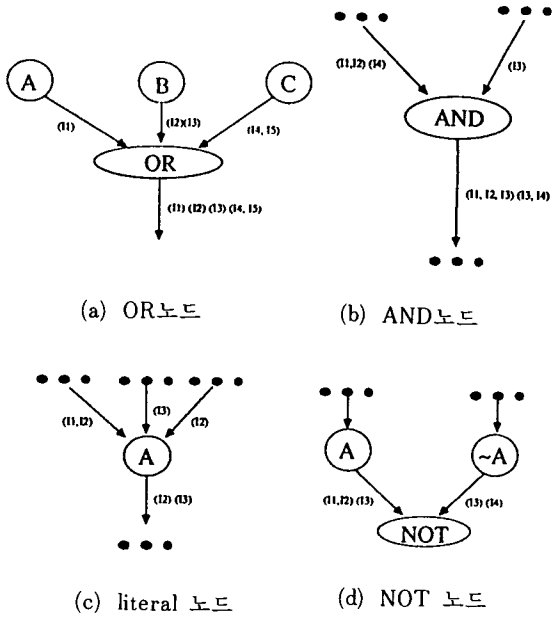


그림 6. 하나의 literal을 유도할 수 있는 입력
Fig. 6. Input inducung a liteal.

규칙의 검증을 위하여 네트워크의 각 노드에서 label 전달은 다음과 같이 행한다.

○OR노드 : 위로 묶여진 노드들이 OR로 연결되어 있음을 나타내므로, 각 경로를 통해서 내려온 label을 그대로 받아서 다음노드에 전달한다. 그림 7(a) 에서 살펴보면, A노드에서 물려받은 label은(I1), B노드에



(a) OR노드 (b) AND노드

(c) literal 노드 (d) NOT 노드

그림 7. 각 노드에서 Travelling
Fig. 7. Travelling in each node.

서 물려받은 label은 (I2) (I3) 그리고, C노드에서 물려받은 label은 (I4, I5)이다. 따라서 OR노드에서는 (I1), (I2), (I3), (I4, I5)를 물려받게 되고 이를 다음 노드에 그대로 전달한다.

○AND노드 : 각각의 경로를 통해서 내려온 label을 다시 DNF(dis juncitive normal from)의 형태로 정돈하여 내려보낸다. 그림 7(b)에서 살펴보면 AND 노드에서는 (I1, I2) (I4)와 (I3)를 각각 물려받았다. 이를 논리적으로 표현해보면 {(I1 &I2) < I4}&I3이다. 이를 DNF형태로 바꾸면서로 다른 (I1&I2&I3) < (I3& I4)으로 된다. 따라서 다음노드에는 (I1, I2, I3), (I3, I4)를 내려보낸다.

○literal노드 : 여러 경로를 통해서 내려온 label들을 서로 비교하여 만약, 겹치는 부분이 있으면 중복으로 검출해낸다. 왜냐하면 그 literal은 다른 두 경로를 통해서 유도될 수 있기 때문이다. 이때 물론 서로 다른 중도결과가 얻어지고, 이들이 필요한 것이라면 중복에 대한 검증이 끝나고 상위노드로부터 받은 label의 겹치는 부분을 정리하여 다음 노드로 전달한다. 그림 7(c)에서 살펴보면 A라는 literal노드는 상위노드로부터 (I1, I2), (I2), (I3)를 각각 물려받았다. 이때 (I1, I2)와 (I2)에서 I2는 I1&I2를 논리적으로 함축하고 있으며 I1&I2는 불필요할 수가 있다. 따라서 이를 중복으로 검출해낸다. 그리고 다음노드로 label

을 전달할 때는 겹치는 부분인 (I1, I2)는 제거하고 (I2), (I3)를 내려보낸다.

○NOT노드 : 이 노드는 모순을 검출해내기 위한 노드로서, 상위노드는 하나의 literal노드와 이의 부정노드로 되어있다. 두 경로를 통해서 내려온 label을 비교하여 겹치는 부분이 있으면 모순으로 검출해낸다. 왜냐하면 label이 겹친다는 것은 서로 모순되는 literal이 같은 조건에서 동시에 유도될 수 있기 때문이다. 그림 7(d)에서 살펴보면 NOT노드에서는 (I1, I2) (I3)와 (I3) (I4)를 각각 A노드와 ~A 노드로부터 물려받았다. 이때 (I3)가 겹치므로 모순으로 검증된다.

이처럼 네트워크를 옮겨가면서 중복과 모순을 검증하게 된다. 따라서 네트워크의 옮겨다님이 종료되면 검증도 종료된다. 그림 8에서는 실제 간단한 지식베이스와 그 네트워크의 예를 들어 중복과 모순을 검증하는 과정을 보여주고 있다. 여기에서 P, Q, R, S는 최상위노드 나타내고 이는 입력 literal에 해당되며, 노드옆에 쓰여진 번호는 노드가 open되는 순서를 나타낸다. 그리고 노드와 노드사이의 아크(arc)에는 전달되는 label을 보여주고 있다. D노드에서 살펴보면, R3노드와 R4노드에서 각각 (P) (Q)와 (Q) (R) 을 물려 받았다. 그런데 이는 (Q)가 겹치므로 중복이검출된다. 그리고 NOT1노드에서는 T노드와 ~T 에서 각각 (P, S) (Q, S) (R, S)와 (Q) (R)을 물려 받았다. 그런데 여기에서 (Q, S)는 (Q)와 (R, S)는 (R)과 겹치므로 모순이 검출된다. 여기에서 중복이나 모순으로 검증하는 것은 약간이라도 가능성이 있는 것들이며, 이것들에 대한 실제 제거 조치여부는 개발자가 결정하게 된다. 한편, 중복이나 모순으로 검증되지 않은 지식 베이스는 완전한 것으로 간주될 수 있다.

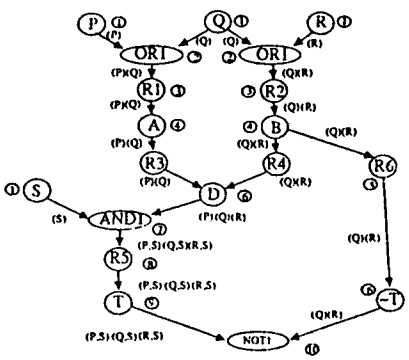


그림 8. 네트워크에서 중복과 모순 검증
Fig. 8. Check on redundancy and inconsistency in network.

Ⅳ. 실험 및 결과고찰

본 실험은 LISP으로 구현하였다. 실험에서 비교대상으로는 KB-reducer를 채택하였으며, 규칙의 갯수를 증가시키면서 수행한 결과를 표 1에서 표3까지 보여주고 있다. 표 1에서는 네트워크 구성을 포함한 전체검증 시간을 비교한 결과이고, 표 2는 순수하게 검증에만 소비되는 시간을 비교한 결과이다. 순수검증 시간은, 제안된 방법의 경우는 네트워크를 옮겨가면서 검증에 소비되는 시간과, KB-reducer의 경우에는 규칙들이 레벨화(leveling)된 다음부터 순수히 검증에 소비되는 시간을 각각 비교한 것이다. 그리고 표 3은 각각의 메모리 소비량을 비교한 결과이다.

표1과 표 2에서 보는 바와 같이 규칙의 갯수가 늘어남에 따라 제안된 방법이 KB-reducer보다 시간소비가 작아진다. 이는 다음과 같은 이유 때문이다. 첫째, KB-reducer의 경우에는 전체 지식베이스의 규칙을 각각 레벨을 정하여 다시 레벨에 따라 재정돈해야 하므로, 규칙을 읽어오면서 바로 네트워크를 구성하는 것보다 더 많은 시간이 소비되고 규칙의 갯수

표 1. 규칙갯수의 증가에 따른 전체 검증시간
Table 1. No of rules vs. total check time.

R-n:n은 규칙의 갯수, 단위 sec .

	R-60	R-120	R-240	R-360
제안된 방법	17	30	59	90
KB-reducer	22	56	143	301

표 2. 규칙갯수의 증가에 따른 순수 검증시간
Table 2. No of rules vs. Pure check time.

R-n:n은 규칙의 갯수, 단위 sec.

	R-60	R-120	R-240	R-360
제안된 방법	4	9	19	28
KB-reducer	9	23	64	133

표 3. 규칙갯수의 증가에 따른 전체 메모리 소비량
Table 3. No of rules vs. Total dnemdy size.

R-n:n은 규칙의 갯수, 단위 Kbyte.

	R-60	R-120	R-240	R-360
제안된 방법	39	61	110	155
KB-reducer	45	64	110	132

가 늘어날수록 시간차는 더욱 생기게 된다. 둘째, 검증에 소비되는 시간도 KB-reducer의 경우에는, 각각 규칙에 대한 label을 결정하기 위해 조건부 절들의 partial-lεbel을 찾아야 하나, 우리의 경우에는 바로 상위노드에서 하위노드로 전달하므로 시간이 적게 걸린다. 세째, KB-reducer 경우에는 중복검증 후에 모순검증을 따로 해야 하나, 우리 방법의 경우에는 동시에 하므로 시간소비가 작아진다.

표3에서 보는 바와 같이 규칙의 갯수가 늘어날수록, 제안된 방법의 메모리 소비가 더 많음을 알 수 있다. 이는 네트워크를 꾸미는데 필요한 메모리로 인해서 더 많이 소비된다. 그러나 KB-reducer의 경우에는 각 literal에 대한 label의 정보를 모두 가지고 있어야 하므로, 검증과정에서는 더 많은 메모리가 소비되어 많은 차이는 보이지 않고 규칙이 360개 정도일때 약 1.2배 정도 차이가 있었다.

검증시간 단축 이외에도 KB-reducer에 비해 우리의 방법은 다음과 같은 여러가지 장점을 지니고 있다. 첫째, 규칙들의 연결을 네트워크로 구성하였으므로, 중복이나 모순이 검출되었을 경우 그 경로 추적을 쉽게 할수 있다. 둘째, 규칙베이스를 네트워크로 변환하는 동시에, 순환규칙 검증이 가능하고, 또한 순환검증 시간 소비도 조금밖에 걸리지 않았다. 그러나 KB-reducer의 경우에는 순환규칙에 대한 검증은 따로 행해져야 한다. 세째, 전체 지식베이스에 새로운 규칙을 첨가하는 경우 KB-reducer 경우에는 새로운 규칙이 입력되면 규칙의 레벨이 달라지므로 전체 지식베이스에 대한 규칙의 레벨화를 다시 해야하지만 우리의 경우에는, 현재 형성된 네트워크에다 새로 입력된 규칙을 첨가하기만 하면 되므로 지식베이스를 만드는 사용자와 interactive하게 사용될 수 있다.

V. 결 론

전문가시스템을 개발하는데 있어서, 중요한 문제는 지식을 효율적이고 적합하게 표현하는 것이다. 본 논문은 이를 위하여 저장된 지식간의 중복, 모순들을검증하여 완전하고 일관성 있는 지식베이스를 구성하도록 도와준다. 이를 위하여, 지식베이스의 규칙들을 네트워크로 변환하여, 이전 방법보다 빠르고 보다 강력한 검증방법을 제시하였다. 그러나, 검증된 결과의 제거 여부는 지식 제공자가 판단하여 실시하게 된다.

KB-reducer와 비교하여 결과, 시간적인 소비면에서는 규칙이 갯수가 360개 정도일 때 약 3배 정도의 속도가 향상되었으며, 규칙의 갯수가 증가함에 따라 점점 더 향상될 수 있다. 그리고 메모리 소비면에서 제안된 방법은 네트워크를 구성하기 때문에 KB-reducer보다 많이 소비되지만, 중복과 모순을 검증하는 과정에서는 메모리 소비가 적어, 규칙이 360개 정도일 때 약 1.2배 정도 더 소비되었다. 이밖에도 4장에서 보인 것처럼 여러가지 장점을 살려 효과적인 검증을 할 수가 있었다. 또한 기존의 전문가 시스템 개발 도구들로 만들어진 지식베이스도 본 논문에서 채택한 표준모델의 형태로 변환하여 검증할 수 있다.

그러나 KB-reducer에서 지적되었던바와 같이 각 literal의 label이 굉장히 복잡한 경우에는 검증시간과 메모리 소비가 많아진다는 문제점을 여전히 가지고 있다.

향후 연구과제로, 네트워크의 특성을 살려 label 을 결정하는데 소비되는 시간과 메모리 소비를 줄이게 되면 보다 더 효과적인 검증을 할 수 있을 것으로 생각된다. 또한 변수를 포함한 literal에 대한 처리문제도 보완되어야 할 연구과제이다. 그러나 변수처리 문제는 규칙의 형태에 의한 분석(syntactic refinement) 뿐 아니라, 실제로 지식베이스의 의미 분석을 통한 완전성(completeness)와 일치성(consistency)을 검증하는 지식의 의미적 개선(semantic refinement)에 관한 연구가 요구된다.[6],[7],[8]

參 考 文 獻

[1] T.A. Nguyen, W.A. Perkins, T.J. Laffey, and D. Pecora "Checking and expert systems knowledge base for consistency and completeness," *Proceedings of the 9th International Joint Conference Artificial Intelegence,* Los Angeles, CA, August 18-24, 1985, pp. 374-378.

[2] T.A. Nguyen, "Verifying consistency of production systems," *Proceedings of the Third Conference on Artificial Intelligence Applications,* IEEE Computer Society Press, 1987, pp. 4-8.

[3] M. Suwa, A.C. Scott, and E.H. Shortliffe, "An approach to verifying completeness and consistency," *The AI Magazine,* Fall, 1982, pp. 16-21.

[4] A. Ginsberg, "A new approach to checking knowledge base for inconsistency and redundancy," *Proceedings of the Third Annual Expert System in Government Conference,* IEEE Computer Society Press, pp. 102-111, 1987.

[5] Johan de Kleer, "An assumption-based TMS," *Artificial Intelligence 28,* 1986, pp. 127-162.
[6] Peter Politakis and Sholom M. Weiss, "Using emprical analysis to refine expert system knowledge bases," *Artificial Intelligence 22,* pp. 22-48, 1984.
[7] Allen Ginsberg, Sholom Weiss, "SEEK2: A generalized approach to automatic knowledge base refinement," *Proceedings of the 9th International joint Conference Artificial Intelegence,* Los Angeles, CA, August 18-24, 1985, pp. 365-374.
[8] W. Marek, "Completeness and consistency in knowledge base systems," *Proceedings Of the First International Conference Expert Database System,* Bejamin Cumming Publishing Company Inc., 1987, pp 119-126.

著 者 紹 介

崔 成 鎬(正會員)
1988年 2月 연세대학교 전자공학과 졸업. 1990年 8月 연세대학교 대학원 전자공학과 석사학위 취득. 현재 한국통신 위성사업단 전임연구원. 주관심분야는 전문가시스템, 지식표현 등임.

●

金 在 熹(正會員)
1953年生. 1979年 2月 연세대학교 전자과 졸업. 1982年 8月 Case Western Reserve University 전자과 석사학위 취득. 1984年 5月 Case Western Reserve University 전자과 박사학위 취득. 현재 연세대학교 전자공학과 부교수. 주관심분야는 인공지능, 퍼지시스템, 패턴인식 등임.

朴 忠 植(正會員)
1962年生. 1985年 2月 한양대학교 전자공학과 졸업. 1985年 8月 연세대학교 전자공학과 석사학위 취득. 현재 연세대학교 대학원 전자공학과 박사과정 재학. 주관심분야는 인공지능 지식표현, 전문가시스템, 객체지향시스템, 지능적 교육시스템 등임.

●

申 東 弼(正會員)
1945年 5月 3日生. 1983年 5月 Utah State University Computer Science 석사. 1986年 5月 University of Oklahoma Computer Science 박사. 1972年～1992年 한국과학기술연구원(KIST) 시스템공학연구소 연구원. 현재 책임연구원 인공지능 연구부장. 주관심분야는 Neural Network, 문자인식, Expert System, Logic Machine 등임.