페트리네트를 이용한 규칙베이스의 검증

조 상 엽†

요 약

생성규칙(production rule)은 전문가시스템에서 전문가의 전문지식(expertise)을 표현하는데 가장 많이 사용하는 지식표현 방법이다. 본 논문에서는 페트리네트(Petri-net)를 이용하여 규칙베이스를 모형화하고, 규칙베이스 검증을 위해 페트리네트가 가지고 있는 체계적이고 구조적인 속성을 이용하여 규칙베이스(rule base)의 무결성(integrity)을 검증하는 방법을 제안한다. 이프러시져는 규칙베이스를 지역적 및 전역적 내부검증을 수행한다.

Verification of Rule Bases Using Petri-net

Sang Yeop Cho†

ABSTRACT

The knowledge representation technique by production rule has been popular method to represent the experts' expertise in expert systems. In this paper, we propose a method to verify the integrity of rule base. Proposed method models rule base as a Petri net and utilizes the systematic structural properties of the Petri net for verification. We describe the procedure to check rule base at both local and global level internal verification.

1. 서 론

전문지식을 규칙베이스에 포함하고 있는 전문가시스템은 인공지능에서 가장 성공적인 분야 중의 하나이다. 규칙베이스를 구성하는 생성규칙은 여러가지 지식표현방법 중에서 프로그램하고 개발하기가 가장 편리하다. 생성규칙은 지식을 조건-행동 쌍(condition-action pair)으로 표현하므로 규칙의 의미해석과 유지보수가 편리하다[5].

전문가시스템은 전문분야의 전문가와 지식공학자 간의 상호작용을 통해 프로토타입핑 방법으로 개발된다. 즉, 지식공학자는 전문가로부터 얻은 전문지식을 생성규칙의 형태로 규칙베이스에 점진적으로 추가한다. 이러한 방법은 피할 수 없는 문제점을 갖는다. 조각(piece-meal)으로 규칙베이스에 추가된 생성규칙은 몇가지 오류를 발생시킬 수 있고, 규칙기반 개발환경이 규칙베이스가 변화할 때마다 무결성(integrity)을 검증하는 방법을 어렵게 만든다.

전문가시스템의 구성 요소 중에 하나인 규칙베이스를 검증한다는 것은 다음과 같이 두가지로 구분할 수가 있다[1]. 첫째는 외부검증(external verification)이고 둘째는 내부검증(internal verification)이다. 외부검증은 전문가의 조언과 전문가시스템이 제공하는 충고가 어느정도 일치하는가를 검증하는 것이다. 내부검증은 규칙베이스 내에 있는 생성규칙의 논리적 정확성을 검증하는 것이다. Nazareth, D. L.은 이 두가지가 서로 밀접한 관계를 가지고 있다는 것을 설명하고 있다

※본 논문의 연구는 충남산업대학교의 교내 연구비를 일부 지원받아 수행됨.

† 종신회원:충남산업대학교 전산학과

논문접수:1996년 10월 28일, 심사완료:1996년 12월 30일

[9]. 내부검증은 검증하는 생성규칙의 대상에 따라 지역적 내부검증(local level internal verification)과 전역적 내부검증(global level internal verification)으로 구분할 수 있다. 지역적 검증은 생성규칙과 생성규칙간의 논리적관계를 검증하는 것이고 전역적 검증은 생성규칙의 집합과 생성규칙간의 논리적관계를 검증하는 것이다.

규칙베이스의 검증에 관한 연구 중 Geissman, J. R. et al., Liebowitz, J. 그리고 O'keefe, R. M., et al. 등은 외부검증에 관한 연구이고[3][6][11], 그외의 대부분의 연구는 내부검증에 관한 연구이다. 내부검증에 관한 연구를 살펴보면 다음과 같다. Suwa, M., et al.이 개발한 규칙 검증기(rule checker)는 ONCOCIN에서 발생하는 중복규칙(redundancy), 포함규칙(subsumption), 모순규칙(inconsistency) 등을 지역적 검증을 하고 결장규칙(missing rule)을 찾아낸다[14]. Nguyen, T. A.는 지역적 검증으로 중복규칙(redundancy), 포함규칙(subsumption) 그리고 모순규칙(inconsistency)을, 전역적 검증으로는 순환규칙(circular chain rule)과 완전성(completeness)을 조사하는 CHECK이라는 규칙베이스 검증 프로그램을 개발하였다[10]. Cragun, B. J., et al.는 중복규칙, 모순규칙, 완전성 그리고 결장규칙을 지역적 검증을 하는 ESC(expert system checker)를 개발하였다[2]. Ginsberg, A.는 진리값관리시스템(truth maintenance system)의 개념을 이용하여 원리적으로는 모든 중복규칙, 모순규칙 그리고 잠재적 모순규칙(potential contradiction)을 검출하는 KB-reduction이라는 프러시져를 발표하였다[4]. Murry, T. J., et al.은 망을 이용한 접근법(network-based approach)을 이용하여 중복규칙, 모순규칙 그리고 결장규칙을 찾아낸다[8]. Agarwal, R., et al.은 페트리네트의 구조적인 속성을 이용하여 중복규칙, 모순규칙 그리고 포함규칙을 지역 및 전역적으로 검증하는 방법을 제안하였다[1]. 그리고 불확실성을 포함한 규칙베이스의 검증하는 연구[16][17]와 혼합지식베이스를 검증하는 방법[18]을 제안한 연구도 있다.

2 장에서는 지식베이스의 검증에 대해서 기술한다. 3 장에서는 페트리네트를 간략하게 설명한다. 4 장에서는 페트리네트를 이용한 규칙베이스의 검증에 관한 새로운 방법을 제안한다. 끝으로, 5 장에서는 결론을 내린다.

2. 지식베이스의 검증

전문가시스템의 규칙베이스에는 전문가가 제공해준 영역지식을 기술한 많은 생성규칙을 포함하고 있다. 각각의 생성규칙은 전제부(antecedent)와 결론부(consequent)로 구성된다. 전제부는 조건절(condition element)이 결론부는 행동절(action element)이 각각 논리곱 형태(conjunctive form)를 이루고 있다[15]. 전제부는 결론부를 논리적으로 유도할 수 있는 사실(fact) 또는 실세계의 상태를 표현한다. 결론부는 전제부가 만족되어 실행될 때 생성규칙이 해야할 논리적으로 관계가 있는 행동을 표현한다[5].

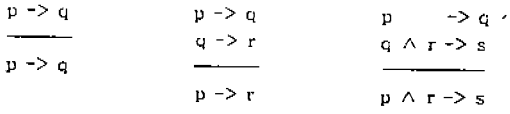
다음에 기술할 분류에서는 모든 생성규칙이 단일한 형식을 갖는 것으로 가정한다. 즉, 전제부는 조건절들의 논리곱형태로 구성되어 있고 결론부는 한 개의 행동절을 갖는 것으로 가정한다. 이와같은 생성규칙으로 구성된 규칙베이스를 잘 구성되었다(well-structured)고 한다. 잘 구성된 규칙베이스는 유지보수와 디버깅이 보다 편리하다[12].

2.1 중복규칙(redundency)

중복규칙은 규칙베이스에 이미 존재하는 생성규칙이 다시 규칙베이스에 추가될 때 발생한다. 지역적 중복규칙은 전문가시스템이 실행될 때 항상 문제를 발생시키지는 않는다. 그러나 규칙베이스에 존재하는 중복규칙을 갱신할 때, 한 개의 생성규칙만을 갱신할 경우 예상하지 않았던 문제가 발생할 수도 있다.

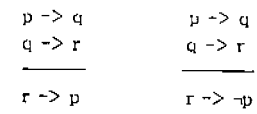
전역적 중복규칙은 규칙베이스내에 불필요한 함축(implication)이 존재할 때 발생한다. 즉, 전제부에서 직접 도달하거나, 추론사슬(inference chain)을 통해 도달하는 결론부가 같을 때 발생한다. 전역적 중복규칙은 조심스럽게 해석해야한다. 추론사슬에 존재하는 절이 중복규칙관계가 없는 다른 생성규칙에서 사용되거나, 사용자에게 설명을 제공할 필요가 있을 수도 있고, 단지 추론 사슬상에서 만 존재할 수도 있다. 각각의 경우에 따라 전역적 중복규칙의 관계에 있는 생성규칙 중에서 제거되어야할 생성규칙을 적절하게 선택하여야 한다.

중복규칙은 다음과 같은 경우에 발생한다.



(a) 지역적 중복규칙 (b) 전역적 중복규칙 (c) 전역적 중복규칙

(그림 1) 중복규칙의 예
(Fig. 1) An example of redundency



(d) 긍정형 순환사슬 (e) 부정형 순환사슬

(그림 2) 모순규칙의 예
(Fig. 2) An example of inconsistency

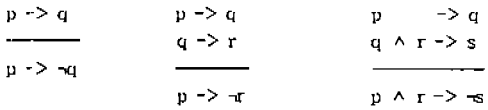
2.2 모순규칙(inconsistency)

전문가시스템은 똑같은 상태(fact)를 가지고 추론을 하면 일관된 추론 결과를 유도해야만 한다. 즉, 같은 입력에 대해서는 같은 추론을 해야 한다. 그러나 서로 다른 결과를 유도하는 같은 전제부를 갖는 생성규칙이 존재할 수 있는 상황이 발생할 수가 있다. 이러한 상황은 전제부는 같지만 결론부가 서로 다른 생성규칙이 규칙베이스내에 함께 존재할 때 발생한다. 이러한 것을 모순규칙(inconsistency), 논리적 충돌(logical conflict) 또는 모호성(ambiguity)이라고 부른다.

전제부가 같고 결론부가 다른 생성규칙들을 반드시 논리적 부정관계(logical negation)라고 볼 필요는 없다. 만일 결론부가 논리적 부정관계가 아닌 타당한 복수 개의 값을 가질 수 있다면, 이러한 상황에서는 같은 전제부에 대해서 서로 다른 결론을 내리는 것이 가능하다. 그러므로 모순규칙이 발생하는 상황은 같은 전제부에 대해서, 결론이 논리적 부정관계라기 보다는 상호배타적(mutually exclusive)인 관계일 때 발생한다고 보는 것이 보다 타당하다. 다음 기술에서 p와 ¬p는 논리적 부정관계가 아니고 상호배타적 관계를 의미한다.

전역적 모순규칙(global level inconsistency)은 두 개의 추론사슬이 동시에 실행 가능하고, 상호배타적인 결과를 출력할 때 발생한다. 그리고 순환사슬(circular chain)이나 규칙베이스가 순환을 가질 경우에도 변형된 형태의 모순규칙이 발생한다.

모순규칙은 다음과 같은 경우에 발생한다.



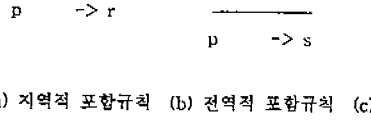
(a) 지역적 모순규칙 (b) 전역적 모순규칙 (c) 전역적 모순규칙

2.3 포함규칙(subsumption)

포함규칙은 하나의 생성규칙이 다른 생성규칙 보다 더 자세하게 정의되어 있을 때 발생한다. 즉, 한 규칙의 전제부가 다른 규칙의 부분집합이 될 때 발생한다. 포함규칙도 중복규칙과 같이 전문가시스템의 추론결과에 항상 영향을 주지는 않는다. 그러나 만일 지역적 포함규칙 중에서 한 개의 생성규칙만을 갱신하게 되면, 규칙베이스 내에 또 다른 오류를 발생시킬 수 있다. 전역적 포함규칙의 갱신을 주의깊게 하지 않으면 규칙베이스에 틈(gap)이 발생하거나, 특정한 추론을 유도할 수 없게 된다. 즉, 포함규칙은 그 자체가 문제라기 보다는 규칙베이스를 개발하고 유지보수하는 지식공학자를 괴롭히는 문제라고 할 수 있다.

규칙베이스내에 존재하는 포함규칙을 해결하는 방법이 몇 가지가 있다. 첫째는 보다 더 자세하게 정의되어 있는 생성규칙을 제거하는 것이다. 둘째는 보다 더 일반적으로 정의되어 있는 생성규칙을 제거하는 것이다. 셋째는 전문가시스템이 사용하고 있는 충돌해결(conflict resolution) 전략을 이용하여 자세하게 또는 일반적으로 정의된 생성규칙을 추론시에 선택하여 실행하는 방법이다. 마지막으로, 생성규칙에 우선순위를 부여하여 추론시에 실행할 생성규칙을 제어하는 방법이다. 규칙베이스 내에 있는 모든 포함규칙을 해결해야하는 것은 아니지만, 이것을 찾아내는 기능은 검증 프러시져가 반드시 가지고 있어야한다.

포함규칙은 다음과 같은 경우에 발생한다.



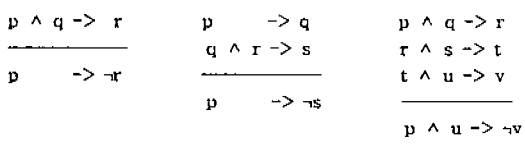
(a) 지역적 포함규칙 (b) 전역적 포함규칙 (c) 전역적 포함규칙

(그림 3) 포함규칙의 예
(Fig. 3) An example of subsumption

2.4 포함모순규칙(subsumed inconsistency)

포함모순규칙은 생성규칙의 전제부는 포함규칙이고 결론부는 모순규칙일 때 발생한다. 포함모순규칙은 전문가시스템이 사용하고 있는 충돌해결전략에 의해 지식공학자나 전문가가 원하는 생성규칙이 실행될 수 있다. 그러나 생성규칙의 결론부가 상호배타적인 관계이므로 경우에 따라 원하지 않는 결과를 추론할 수도 있다. 이와같은 문제를 해결하기 위해서 포함모순규칙도 규칙베이스를 검증할 때 찾아내야만 한다.

포함모순규칙은 다음과 같은 경우에 발생한다.



(a) 지역적 포함모순규칙 (b) 전역적 포함모순규칙 (c) 전역적 포함모순규칙

(그림 4) 포함모순규칙의 예
(Fig. 4) An example of subsumed inconsistency

3. 페트리네트

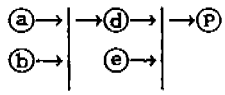
페트리네트는 정보의 흐름을 표현하기 위해 사용하는 방법으로 객체간의 입출력관계를 그래프를 이용하여 보여줄 수 있고, 병렬성, 비동기성 그리고 분산성을 갖는 정보처리시스템을 기술하거나 연구하는 유용한 도구이다[7][13].

페트리네트의 네가지 구성요소는 플레이스의 집합, 트랜지션의 집합, 입력함수 그리고 출력함수이다. 플레이스는 입력과 출력을 나타내는 객체를 표현할 때 사용하고, 입출력함수는 트랜지션에 대한 플레이스의 입출력관계를 나타낸다. 이러한 페트리네트를 그래프로 나타낼 수도 있다. 그래프로 그릴 경우, 플레이스는 원으로 그리고 트랜지션은 수직선으로 그린다. 트랜지션에 대한 플레이스들의 입출력관계는 아크로 표시한다.

페트리네트는 규칙베이스의 생성규칙을 모형화할 수 있는 자연스럽고 강력한 방법을 제공한다. 즉, 생성규칙의 전제부와 결론부에 있는 조건절과 행동절은 각각 입력과 출력 플레이스에 대응되고, 생성규칙은 트랜지션에 대응된다. (그림 5)는 규칙베이스에 있는 생성규칙을 나타내고 있고, (그림 6)은 (그림 5)의 생성규칙을 페트리네트를 이용하여 표현한 것이다.

rule1: a∧b -> d
rule2: d∧e -> p

(그림 5) 규칙베이스내의 생성규칙
(Fig. 5) Production rules in rule base



(그림 6) (그림 5)의 페트리네트 표현
(Fig. 6) Petri-net representation of (Fig. 5)

페트리네트로 표현된 입력과 출력 플레이스의 관계는 입력 인시던스 행렬(incidence matrix), 출력 인시던스 행렬 그리고 입출력 인시던스 행렬 등으로 표현할 수 있다. (그림 7), (그림 8) 그리고 (그림 9)는 이러한 인시던스 행렬의 예를 보여준다.

인시던스 행렬에서 생성규칙의 전제부, 즉 입력 플레이스에 해당하는 곳은 −1로 표현하고, 생성규칙의 결론부, 즉 출력 플레이스에 해당하는 곳은 1로 표현한다.

input	a	b	d	e	p
rule1	-1	-1	0	0	0
rule2	0	0	-1	-1	0

(그림 7) (그림 6)의 입력 인시던스 행렬
(Fig. 7) Input incidence matrix of (Fig. 6)

output	a	b	d	e	p
rule1	0	0	1	0	0
rule2	0	0	0	0	1

(그림 8) (그림 6)의 출력 인시던스 행렬
(Fig. 8) Output incidence matrix of (Fig. 6)

i/o	a	b	d	e	p
rule1	-1	-1	1	0	0
rule2	0	0	-1	-1	1

(그림 9) (그림 6)의 입출력 인시던스 행렬
(Fig. 9) Input/output incidence matrix of (Fig. 6)

4. 규칙베이스의 검증

4.1 규칙베이스 검증 알고리즘

본 논문에서 제안하는 규칙베이스를 검증하는 알고리즘은 페트리네트가 가지고 있는 체계적인 속성 중에서 인시던스 행렬을 이용한다. 검증 알고리즘은 규칙베이스의 생성규칙의 입출력관계를 나타낸 인시던스 행렬과 규칙베이스에 새롭게 추가되는 생성규칙을 표현한 인시던스 행렬을 이용하여 검증한다.

검증 알고리즘에서 사용하는 기호는 다음과 같다. IM^-, IM^+, IM^0, 그리고 IM은 각각 입력, 출력, 입출력 그리고 세가지 인시던스 행렬을 표현한다. $IPRIM^-$, $IPRIM^+$, $IPRIM^0$ 그리고 IPRIM은 각각 입력, 출력, 입출력 그리고 세가지 삽입 생성규칙(insert production rule) 인시던스 행렬을 표현한다. RM^-, RM^+, RM^0, 그리고 RM은 각각 입력, 출력, 입출력 그리고 세가지의 관계행렬(relevant matrix)을 나타낸다. Solution List는 입력 생성규칙과 중복규칙 또는 모순규칙의 관계에 있는 해규칙(Solution Rule)을 저장하는 리스트이다.

검증 알고리즘

입력 : 인시던스 행렬, 입력 생성규칙
출력 : 중복규칙 또는 포함규칙

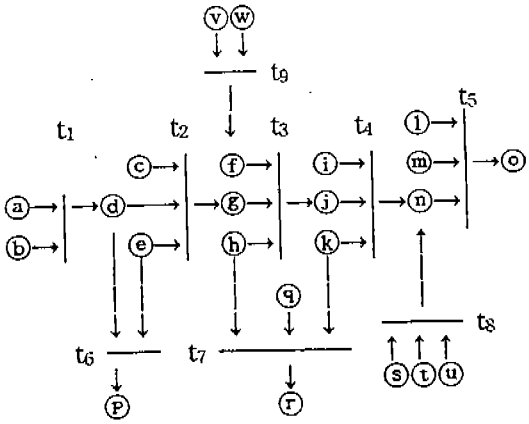
1. Incidence Matrix IM에서 Insert Production Rule IPR의 조건절을 조건절로 포함하는 Relevant Rule Set RRS을 구한다.
   1.1 $RRS = IM^- \cdot IPRIM^{-t}$
   1.2 RRS을 이용하여 Relevant Matrix RM을 구한다.
2. Relevant Matrix RM에서 Insert Production Rule IPR의 행동절을 행동절로하는 Solution Rule SR을 구한다.
   2.1 $SR = RM^+ \cdot IPRIM^{+t}$
   2.2 RRS = RRS − SR
3. RRS에 있는 각각의 Relevant Rule RR의 전제부 Ant(RR)와 결론부 Con(RR)를 구한다.
4. IPR의 전제부 Ant(IPR)와 결론부 Con(IPR)를 구한다.
5. Solution List = Solution List ∪ SR
6. $Ant(SR) \cap Con(RR) \neq \phi$ 동안 다음을 반복한다.
   6.1 $Ant(SR) \cap Con(RR) \neq \phi$로 하는 RR을 구한다.
   6.2 SR ← RR
   6.3 Solution List = Solution List ∪ SR
   6.4 RRS = RRS − SR
7. 만일 Ant(IPR) = Ant(Solution List)이면 중복규칙 그렇지않으면 포함규칙이다.

4.2 규칙베이스 검증의 예

(그림 10)은 현재 아홉 개의 생성규칙이 있는 규칙베이스를 나타낸 것이다. (그림 11)은 규칙베이스에 있는 아홉 개의 생성규칙을 페트네트를 이용하여 표현한 것이다. 트랜지션 t_1에서 t_9까지는 각각 rule1에서 rule9에 대응되고, 플레이스 ⓐ에서 ⓦ까지는 각 생성규칙의 전제부와 결론부에 있는 조건절과 행동절에 대응된다. (그림 11)에 대응하는 입력 인시던스 행렬, 출력 인시던스 행렬 그리고 입출력 인시던스 행렬은 각각 (그림 12), (그림 13) 그리고 (그림 14)에 나타나있다. 인시던스 행렬에서 입력 플레이스에 해당하는 곳은 −1을, 출력 플레이스에 해당하는 곳은 1을 표시한다.

rule1: a∧b -> d
rule2: c∧d∧e -> g
rule3: f∧g∧h -> j
rule4: i∧j∧k -> n
rule5: l∧m∧n -> o
rule6: d∧e -> p
rule7: h∧k∧q -> r
rule8: s∧t∧u -> n
rule9: v∧w -> f

(그림 10) 규칙베이스내의 생성규칙
(Fig. 10) Production rules in rule base



(그림 11) (그림 10)의 페트리네트 표현
(Fig. 11) Petri-net representation of (Fig. 10)

4.2.1 중복규칙의 검증

(그림 15)는 중복규칙을 검사하기 위한 생성규칙이다. (그림 16)은 (그림 15)의 페트리네트 표현이다. (그림 17), (그림 18) 그리고 (그림19)는 각각 (그림 16)의 입력, 출력 그리고 입출력 인시던스 행렬이다.

검증 알고리즘의 단계 1에서 $IM^- \cdot IPRIM^{-t}$을 계산하면 결과행렬 [0 3 2 2 0 2 2 0 0]^{t}을 얻는다. 결과행렬의 값이 1 이상인 규칙베이스내의 생성규칙을 RRS={rule2, rule3, rule4, rule6, rule7}로 하고, 이것을 이용하여 RM을 구한다. 결과행렬에서 1 이상의 값의 의미는 입력하는 생성규칙의 조건절을 조건절로 포함하고 있는 규칙베이스내의 생성규칙을 의미한다. RRS를 이용하여 구한 입력, 출력 그리고 입출력 관계행렬은 각각 (그림 20), (그림 21) 그리고 (그림 22)와 같다.

input	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule2	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule3	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule4	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0
rule5	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0
rule6	0	0	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule7	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	-1	0	0	0	0	0	0
rule8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	0	0
rule9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1

(그림 12) (그림 11)의 입력 인시던스행렬 M^-
(Fig. 12) Input incidence matrix M^- of (Fig. 11)

output	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
rule4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
rule5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
rule6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
rule7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
rule8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
rule9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

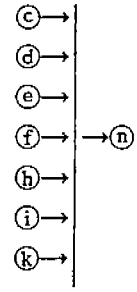
(그림 13) (그림 11)의 출력 인시던스행렬 M^+
(Fig. 13) Output incidence matrix M^+ of (Fig. 11)

i/o	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule1	-1	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule2	0	0	-1	-1	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule3	0	0	0	0	0	-1	-1	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
rule4	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	1	0	0	0	0	0	0	0	0	0
rule5	0	0	0	0	0	0	0	0	0	0	0	-1	-1	-1	1	0	0	0	0	0	0	0	0
rule6	0	0	0	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
rule7	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	-1	1	0	0	0	0	0
rule8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	-1	-1	-1	0	0
rule9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1

(그림 14) (그림 11)의 입출력 인시던스행렬 M°
(Fig. 14) Input/output incidence matrix M° of (Fig. 11)

rule10: c^d^e^f^h^i^k -> n

(그림 15) 중복규칙 검사를 위한 생성규칙
(Fig. 15) Production rule for testing redundancy



(그림 16) (그림 15)의 페트리네트 표현
(Fig. 16) Petri-net representation of (Fig. 15)

input	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule10	0	0	-1	-1	-1	-1	0	-1	-1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0

(그림 17) (그림 16)의 입력 인시던스 행렬 $IPRIM^{-}$
(Fig. 17) Input incidence matrix $IPRIM^{-}$ of (Fig. 16)

output	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

(그림 18) (그림 16)의 출력 인시던스 행렬 $IPRIM^{+}$
(Fig. 18) Output incidence matrix $IPRIM^{+}$ of (Fig. 16)

i/o	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule10	0	0	-1	-1	-1	-1	0	-1	-1	0	-1	0	0	1	0	0	0	0	0	0	0	0	0

(그림 19) (그림 16)의 입출력 인시던스 행렬 $IPRIM^{\circ}$
(Fig. 19) Input/output incidence matrix $IPRIM^{\circ}$ of (Fig. 16)

input	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule2	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule3	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule4	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0
rule6	0	0	0	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule7	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	-1	0	0	0	0	0	0

(그림 20) 입력 관계행렬 RM^-
(Fig. 20) Input relevant matrix RM^-

output	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
rule4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
rule6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
rule7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

(그림 21) 출력 관계행렬 RM^+
(Fig. 21) Output relevant matrix RM^+

i/o	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule2	0	0	-1	-1	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rule3	0	0	0	0	0	-1	-1	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
rule4	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	1	0	0	0	0	0	0	0	0	0
rule6	0	0	0	-1	-1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
rule7	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	-1	1	0	0	0	0	0

(그림 22) 입출력 관계행렬 RM^o
(Fig. 22) Input/output relevant matrix RM^o

알고리즘의 단계 2에서는 $RM^+ \cdot IPRIM^{+t}$를 계산하면 [0 0 1 0 0]^{t}을 얻는다. 이 행렬을 이용하여 SR={rule4}을 구한다. 그리고 이 SR을 RRS에서 제외한다. 그러므로 RRS={rule2, rule3, rule6, rule7}가 된다. 알고리즘의 단계 3과 단계 4를 실행한 결과는 각각 〈표 1〉과 〈표 2〉와 같다.

〈표 1〉 RRS의 전제부와 결론부
〈Table 1〉 Antecedent and consequent of RRS

rule	antecedent	consequent
rule2	c d e	g
rule3	f g h	j
rule6	d e	p
rule7	h k q	r

〈표 2〉 SR의 전제부와 결론부
〈Table 2〉 Antecedent and consequent of SR

rule	antecedent	consequent
rule4	i j k	n

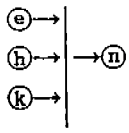
알고리즘의 단계 5에서는 Solution List={rule4}가 된다. 알고리즘의 단계 6을 실행하고 나면 Solution List={rule4, rule3, rule2}가 된다. 그리고 마지막으로 단계 7을 실행한 결과는 다음과 같다. Ant(IPR)={c, d, e, f, h, i, k}이고 Ant(Solution List)={c, d, e, f, h, i, k}이다. 즉, Ant(IPR)=Ant(Solution List)이므로 새롭게 추가되는 rule10은 현재 규칙베이스에 있는 rule4, rule3 그리고 rule2와 중복규칙이 된다.

4.2.2 포함규칙의 검증

(그림 23)은 포함규칙을 검사하기 위한 생성규칙이다. (그림 24)는 (그림 23)의 페트리네트 표현이다. (그림 25), (그림 26) 그리고 (그림 27)은 각각 (그림 24)의 입력, 출력 그리고 입출력 인시던스 행렬이다. rule11의 입력 생성규칙을 이용하여 검증 알고리즘을 실행하면 Ant(IPR)={e, h, k}이고 Ant(Solution List)={c, d, e, f, h, i, k}이다. 즉, Ant(IPR)≠Ant(Solution List)이므로 rule11과 rule4, rule3 그리고 rule2와는 포함규칙이 된다.

rule11: e∧h∧k -> n

(그림 23) 포함규칙 검사를 위한 생성규칙
(Fig. 23) Production rule for testing subsumption



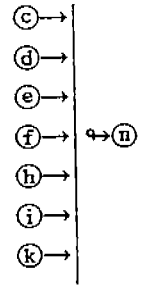
(그림 24) (그림 23)의 페트리네트 표현
(Fig. 24) Petri-net representation of (Fig. 23)

4.2.3 모순규칙의 검증

(그림 28)은 모순규칙을 검사하기 위한 생성규칙이다. (그림 29)는 (그림 28)의 페트리 네트 표현이다. →ⓝ는 ¬n을 페트리 네트로 표현한 것이다. 모순규칙의 검사는 rule12를 (그림 15)의 rule10으로 변경하여 알고리즘을 실행한다. rule10을 이용하여 알고리즘을 실행한 결과가 중복규칙이되면 원래의 생성규칙인 rule12와는 모순규칙이 된다.

rule12: c∧d∧e∧f∧h∧i∧k -> ¬n

(그림 28) 모순규칙 검사를 위한 생성규칙
(Fig. 28) Production rule for testing inconsistency



(그림 29) (그림 28)의 페트리네트 표현
(Fig. 29) Petri-net representation of (Fig. 28)

input	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule11	0	0	0	0	-1	0	0	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0

(그림 25) (그림 24)의 입력 인시던스 행렬 $IPRIM^-$
(Fig. 25) Input incidence matric $IPRIM^-$ of (Fig. 24)

output	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

(그림 26) (그림 24)의 출력 인시던스 행렬 $IPRIM^+$
(Fig. 26) Output incidence matrix $IPRIM^+$ of (Fig. 24)

i/o	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
rule11	0	0	0	0	-1	0	0	-1	0	0	-1	0	0	1	0	0	0	0	0	0	0	0	0

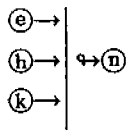
(그림 27) (그림 24)의 입출력 인시던스 행렬 $IPRIM^0$
(Fig. 27) Input/output incidence matrix $IPRIM^0$ of (Fig. 24)

4.2.4 포함모순규칙의 검증

(그림 30)은 포함모순규칙을 검사하기 위한 생성규칙이다. (그림 31)은 (그림 30)의 페트리 네트표현이다. 포함모순규칙의 검사도 rule13을 (그림 23)의 rule11로 변경하여 알고리즘을 실행한다. rule11을 이용하여 알고리즘을 실행한 결과가 포함규칙이면 원래의 생성규칙 rule13과는 포함모순규칙이 된다.

rule13: e^h^k -> ¬n

(그림 30) 포함모순규칙 검사를 위한 생성규칙
(Fig. 30) Production rule for testing subsumed inconsistency



(그림 31) (그림 30)의 페트리 네트 표현
(Fig. 31) Petri-net representation of (Fig. 30)

5. 결 론

본 논문에서는 페트리네트가 가지고 있는 체계적인 속성을 이용하여 규칙베이스를 검증하는 프러시져를 제안하였다. 이 프러시져는 전문가 시스템의 구성요소인 규칙베이스를 확장할 때, 중복규칙, 포함규칙, 모순규칙 그리고 포함모순규칙 등을 검사할 수 있다. 그리고 규칙베이스를 지역적으로 뿐만 아니라 전역적으로 검증한다. 앞으로의 연구과제로는 결장규칙이나 순환규칙 등도 검증할 수 있도록 프러시져를 확장하는 것과 행동절이 두 개 이상인 경우에도 적용할 수 있는 프러시져로 확장 개발하는 것이다.

참 고 문 헌

[1] Agarwal, R., and Tanniru, M., "A Petri-net based Approach for Verifying the Integrity of Production Systems", Int'l J. of Man-Machine Studies, 36, pp. 447-468, 1992.

[2] Cragun, B. J. and Steudel, H. J., "A Decision-table-based Processor for Checking Completeness and Consistency in Rule-based Expert Systems", Int'l J. of Man-Machine Studies, 26, pp. 633-648, 1987.

[3] Geissman, J. R. and Schultz, R. D., "Verification and Validation of Expert Systems", AI Expert, 3, pp. 26-33, 1988.

[4] Ginsberg, A., "Knowledge-base Reduction:A New Approach to Checking Knowledge Bases for Inconsistency and Redundancy", Proceedings of the National Conference on Artificial Intelligence, pp. 595-589, 1988.

[5] Jackson, P., "Introduction to Expert systems", Addison-Wesley, 1990.

[6] Liebowitz, J., "Useful Approach for Evaluating Expert Systems", Expert Systems, 3, pp. 86-96, 1986.

[7] Murata, T., "Petri Nets:Properties, Analysis and Applications", Proceedings of the IEEE, 77(4), pp. 541-580, 1989.

[8] Murry, T. J. and Tanniru, M. R., "Control of Inconsistency and Redundancy in PROLO-type KnowledgeBases", 20th Hawaii Int'l Conf. on System Science, 1987.

[9] Nazareth, D. L., "Issues in the Verification of Knowledge in Rule-based Systems", Int'l J. of Man-Machine Studies, 30, pp. 255-271, 1989.

[10] Nguyen, T. A., Perkins, W. A., Laffey, T. J., and Pecore, D., "Knowledge Base Verification", AI Magazine, pp. 69-75, 1987.

[11] O'keefe, R. M., Balci, O., and Smith, E. P., "Validating Expert System Performance", IEEE Expert, Winter, pp. 81-90, 1987.

[12] Pederson, K. "Well-structured Knowledge Bases", AI Expert, 4, pp. 44-55.

[13] Peterson, J. L., "Petri Net Theory and The Modelling of Systems", Prentice-Hall, 1981.

[14] Suwa, M., Scott, A. C., and Shortiffe, E. H., "Completeness and Consistency in a Rule-based Systems", In Buchanan, B. G., and Shortliffe, E.

H., Eds. Rule-based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project, pp. 159-170, MA: Addison-Wesley, 1984.

[15] Giarratano, J. and Riley, G., "Expert systems: Principles and Programming," PWS-KENT Publishing Co., 1989.

[16] Castro, J. K. and Trillas, E., "The Management of The Inconsistency in Expert Systems," Fuzzy Sets and Systems, Vol. 58, pp. 51-57, 1993.

[17] Hall, L. O., Friedman, M., and Kandel, A., "On The Validation and Testing of Fuzzy Expert Systems," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 18, No. 6, Nov./Dec., pp. 1023-1027, 1988.

[18] Lee, S. and O'keefe, R. M., "Subsumption Anomalies in Hybrid Knowledge Bases," Int'l Jl. of Expert Systems, Vol. 6, No. 3, pp. 299-320, 1993.

조 상 엽

1986년 한남대학교 전자계산학과 졸업(공학사)
1988년 중앙대학교 대학원 전자계산학과(이학석사)
1993년 중앙대학교 대학원 전자계산학과(공학박사)
1993년~1995년 중앙대학교 컴퓨터소프트웨어 연구소 객원 연구원
1995년~현재 충남산업대학교 전자계산학과 전임강사
관심분야: 인공지능, 전문가시스템, 퍼지이론