Matrix-Based Intelligent Inference Algorithm Based On the Extended AND-OR Graph

Kun Chang Lee
Sung Kyun Kwan University
(Tel) 760-0505, (Fax) 745-4566
leekc@yurim.skku.ac.kr

Hyung Rae Cho
Kyung Sang National University
Department of Industrial Engineering

Abstract

The objective of this paper is to apply Extended AND-OR Graph (EAOG)-related techniques to extract knowledge from a specific problem-domain and perform analysis in complicated decision making area. Expert systems use expertise about a specific domain as their primary source of solving problems belonging to that domain. However, such expertise is complicated as well as uncertain, because most knowledge is expressed in causal relationships between concepts or variables. Therefore, if expert systems can be used effectively to provide more intelligent support for decision making in complicated specific problems, it should be equipped with real-time inference mechanism.

We develop two kinds of EAOG-driven inference mechanisms-(1) EAOG-based forward chaining and (2) EAOG-based backward chaining. and The EAOG method possesses the following three characteristics.

1. Real-time inference: The EAOG inference mechanism is suitable for the real-time inference because its computational mechanism is based on matrix computation.

2. Matrix operation: All the subjective knowledge is delineated in a matrix form, so that inference process can proceed based on the matrix operation which is computationally efficient.

3. Bi-directional inference: Traditional inference method of expert systems is based on either forward chaining or backward chaining which is mutually exclusive in terms of logical process and computational efficiency. However, the proposed EAOG inference mechanism is generically bi-directional without loss of both speed and efficiency.

I 서론

현대의 경영이 복잡계 경영으로 나아가면서 의사결정시 단순한 형태의 문제보다는 심층적인 지식과 신속한 의사결정이 요구된다 (Konskynski, 1993). 이러한 경우에 기존의 전문가시스템에서는 술어논리(Predicate Logic)를 이용한 규칙기반 추론 메카니즘으로 활용하였으나 이러한 전문가시스템의 경우 그 추론의 결과가 의사결정에 적합하기는 하지만 복잡한 문제의 실시간 처리에 적용하기에는 부적절하다는 한계점이 있었다 (Looney & Alfize, 1987).

이러한 이유로, 실시간 추론을 위하여 술어논리 대신 명제논리(Propositional Logic)를 사용하는 방안을 생각해 볼 수 있다. 명제논리를 이용한 실시간 추론방법은 술어논리에 비해 기능적인 측면에서 상대적으로 약하긴 하지만, 단순하고 빠르며, 특히 고속의 처리를 요하는 제어시스템에 적용하기에 충분한 성능을 제공한다는 장점이 있다. 이런 명제논리를 이용한 제어명령 생성기법에 관한 연구는 Looney & Alfize (1987)에 의해 선행된 적이 있다. 그들은 연구에서 명제논리를 이용하여 표현된 문제영역에 관한 규칙을 리스트(list)로 처리하는 것이

아니라 부울 규칙행렬(Boolean rule matrix)을 이용하여 보다 빠르게 추론을 실행하는 방법에 관해 연구하였다. 그러나 그들의 연구내용은 획득된 규칙의 전제부분이 단일조건이 아닌 여러 조건들의 결합으로 구성될 경우 이 결합조건에 포함되는 단일조건들은 모두 추론이 아닌 실험 또는 관측을 통해 그 진위값을 알 수 있는 조건들이라는 가정에 바탕을 두고 있다. 이러한 가정은 그들이 제시하고 있는 방식을 실제 문제에 적용하는 것을 거의 불가능하게 만들고 있다 (왜냐하면 그러한 가정을 충족하는 실제 문제는 거의 없다고 해도 과언이 아니기 때문). 따라서 본 논문에서는 아무리 복잡한 규칙구조를 가지는 문제에도 적용가능한 새로운 실시간 추론기법으로 확장된 AND-OR그래프 (EAOG: Extended AND-OR Graph) 에 의한 추론메카니즘을 제안한다. 본 연구에서 제안하는 추론 메카니즘의 특징을 살펴보면 다음과 같다.

(1) 행렬을 기본으로 하는 지능적인 메카니즘으로 비구조적인 거래조건을 추론할 수 있도록 한다.
(2) 기존의 전문가시스템이 가지고 있는 패턴매칭의 방법을 지양하고, 실시간으로 수치적인 추론이 가능하도록 하여 보다 효과적인 추론이 가능하도록 한다.
(3) 전통적으로 전문가시스템에서 사용되어 왔던 방법과는 달리 (Giarratano & Riley 1994), 행렬에 기초하기 때문에 추론속도가 빨라 실시간으로 의사결정을 내릴 수 있다.

따라서 본 추론 메카니즘은 심층적인 지식이 내포되어있는 복잡한 의사결정 문제를 해결하거나, 인터넷을 통하여 원거리에서 실시간으로 의사결정을 내려야하는 전자상거래 상황에 적합한 메카니즘이라고 말할 수 있다. 특히 전자상거래의 경우에는 거래당사자들이 서로의 거래 조건을 추론 메카니즘을 활용하여 입력하고 그 추론 결과를 실시간으로 의사결정에 반영함으로써 협상을 통하여 서로의 거래를 성사시킬 수 있다 (Kalakota, & Winston, 1996). 본 논문의 구성으로는 먼저 2장에서는 확장된 AND-OR 그래프에 대하여 살펴보고, 3,4장에서는 본 연구의 추론 메카니즘의 하나인 전방향/역방향 추론 메카니즘에 대하여 살펴보도록 한다. 5장에서는 전문가시스템 사례를 통하여 본 연구에서 주장하는 추론 메카니즘의 효율성을 검증하고 마지막으로 6장에서는 결론 및 향후 연구 방향에 대하여 살펴보도록하자.

II. 확장된 AND-OR 그래프

기존의 AND/OR 그래프 추론 방식은 복잡한 추론방식보다는 소규모의 지식베이스를 갖고 추론하는 경우 널리 사용되는 트리구조의 추론 방식이다. 하지만 최근 표면지식(Surface Knowledge) 또는 피상적 지식(Shallow Knowledge) 보다는 심층지식(Deep Knowledge)에 대한 관심이 고조되고 있다 (Chismar & Meier, 1992). 그런데 일반적으로 심층지식의 경우 이를 표현하기 위한 규칙경로의 길이는 매우 복잡하다. 따라서 기존의 일반적인 AND/OR 그래프 추론 방식으로는 이를 효율적으로 추론하기가 불가능하다. 이에 본 논문에서 확장된 AND/OR 그래프를 통하여 복잡한 규칙경로를 효율적으로 추론하기로 한다. 본 논문에서 다루는 추론 메카니즘 규칙의 기본 형태는 다음과 같다.

IF α_1 and α_2 and THEN β (식 1)

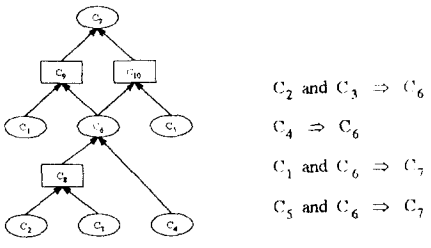
여기서 각 α_i 및 β 는 조건(condition or proposition)을 의미하며, 조건들의 결합(conjunction)인 α_1 and α_2 and ... 는 규칙의 전제부분(premise), β 는 규칙의 결론부분(conclusion or action)을 나타낸다. 그리고 규칙의 전제부분이 하나의 조건으로 구성되어 있을 경우 이를 **단순규칙**(simple rule)이라 하고, 규칙의 전제부분이 둘 이상의 조건으로 결합되어 있을 경우 이를 **복합규칙**(composite rule)이라 한다. (식 1)에 나타난 형태의 규칙은 간략히 α_1 and α_2 and ... $\Rightarrow \beta$ 로 표기된다.

그런데 주어진 영역에서 문제를 해결하기 위한 규칙을 수집하면 항상 (식 1)과 같은 기본 형태를 띠고 있는 것은 아니다. 기본 형태가 아닌 규칙의 유형은 다음과 같다.

Type 1 : $\alpha \Rightarrow (\beta$ and $\gamma)$,
Type 2 : $(\alpha$ or $\beta) \Rightarrow \gamma$,
Type 3 : α and $(\beta$ or $\gamma) \Rightarrow \delta$
Type 4 : $\alpha \Rightarrow (\beta$ or $\gamma)$.

하지만 이와같은 유형의 규칙은 다음과 같이 동등한(equivalent) 기본 형태의 규칙으로 변경할 수 있다. 즉 Type 1 은 $\alpha \Rightarrow \beta$ 와 $\alpha \Rightarrow \gamma$ 라는 두개의 기본 형태 규칙으로 나눌 수 있고, Type 2 는 $\alpha \Rightarrow \gamma$ 와 $\beta \Rightarrow \gamma$ 로 나눌 수 있다. 또한 Type 3 는 $(\alpha$ and $\beta) \Rightarrow \delta$ 와 $(\alpha$ and $\gamma) \Rightarrow \delta$ 로 나눌 수 있다. Type 4 는 추론에 있어 특정한 의미를 가지지 못하며, 특히 자동제어 분야에는 적용하기 어려운 유형이기 때문에 본 논문에서는 고려하지 않는다.

이제 주어진 문제영역에서 수집한 규칙들이 기본 형태의 단순규칙 및 복합규칙으로 표현되었다고 할 때 이를 그림으로 나타내는 확장된 AND-OR 그래프 (Extended AND-OR Graph : 이하 EAOG라 약함)에 대해 설명하면 다음과 같다. <그림 1>에서 보듯이 EAOG는 두가지 형태(원과 네모)의 노드와 이들을 연결하는 호로 구성된다. **단순노드**라 불리는 각 원은 하나의 조건을 나타내고 **복합노드**라 불리는 각 네모는 조건들의 결합(conjunction)을 나타낸다. 그리고 단순노드로 들어오는 호는 단순 또는 복합규칙을 의미하며 복합노드로 들어오는 호는 결합조건의 구성요소를 나타낸다.



<그림 1> 예제 EAOG

또한 EAOG를 구성하는 노드들은 그 역할에 따라 다음과 같이 여러 가지 종류로 나눌 수 있다.

데이터노드 : 데이터 노드란 들어오는 호는 없이 나가는 호만 있는 노드로서 진위값이 실험 또는 관측을 통해 정해질 수 있는 사실을 의미한다. 예를 들어 <그림 1>에서 데이터노드는 C_1, C_2, C_3, C_4 및 C_5 이다.

최종노드 : 들어오는 호만 있고 나가는 호는 없는 노드로서 주로 최종 관심의 대상이 되며 의사결정에 직접적인 영향을 미치는 사실을 나타낸다. 예를들어 <그림 1>에서 최종노드는 C_7 이다.

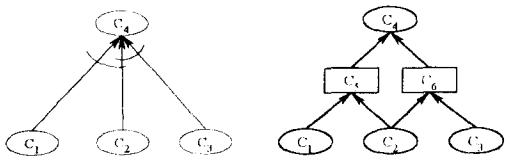
중간노드 : 들어오는 호 및 나가는 호가 동시에 존재하는 노드로서 데이터노드와 최종노드를 연결하는 매개체 역할을 한다. 특히 EAOG에서 복합노드는 항상 중간노드가 된다. 예를들어 <그림 1>에서 중간노드는 C_6, C_8, C_9 및 C_{10} 이다.

전제노드 : 임의의 노드 C_i 에 대한 전제노드란 노드 C_i 로 들어오는 호의 꼬리 부분에 연결되어 있는 노드를 의미한다. 예를들어 <그림 1>에서 노드 C_6 의 전제노드는 C_4 와 C_8 이다.

결과노드 : 임의의 노드 C_i 에 대한 결과노드란 노드 C_i 에서 나가는 호의 머리 부분에 연결되어 있는 노드를 의미한다. 예를들어 <그림 1>에서 노드 C_6 의 결과노드는 C_9 와 C_{10} 이다.

이상에서 설명한 EAOG와 기존의 AND-OR 그래프의 차이점은 복합규칙을 나타내기 위한 복합노드의 도입이라 할 수 있다(<그림 2> 참조). 이렇게 규칙의 구조를 표현할 때 기존의 AND-OR 그래프처럼 단순노드만을 사용하여 호로 연결시키는 것이 아니라 EAOG처럼 복합노드를 도입할 경우의 장점은 다음과 같다. 첫째, 복합규칙들을 구성하는 조건들이 상호 복잡하게 얽혀있을 경우 이를 보다 체계적으로 나타내 보임으로서 규칙의 구조에 대한 이해도를 높이고 잘못된 부분이나 누락된 부분의 발견을 용이하게 한다. 둘째, EAOG를 이용하면 본 논문의 III, IV절에서 제시된 것처럼 정수행렬 연산으로만 이루어진 간편하고 효율적인 추론이 가능하다는 것이다. 제시된 추론방식의 핵심은 각 노드의 진위값을 해당 전제노드 중 참이라고 밝혀진 노드의 개수를 바탕으로 판단한다는 것이다. 예를들어 <그림 2>에서 노드 C_4가 참이 되기 위해서는 전제노드 중 2개(C_1, C_2 또는 C_2, C_3)가 참이 되면 된다. 따라서 추론시 C_4의 진위값은 전제노드 중 2개 이상이 참이면 참이 된다고 추론하고 싶다는 것이다. 그런데 <그림 2(a)>와 같은 기존의 AND-OR 그래프를 바탕으로 이러한 추론방식을 이용하면 노드 C_1 및 C_3 만 참일 경우 노드 C_4 가 거짓이 되어야 함에도 불구하고 참이라고 추론된다는 문제점이 발생한다. 하지만 <그림 2(b)>와 같은 EAOG를 이용하면 C_1 및 C_3 만 참일 경우 C_5 및 C_6 가 거짓이 되고 이에 따라 C_4도 거짓이라는 추론이 정수행렬 연산만으로도 쉽게 이루어질 수 있다.

C_1 and C_2 ⇒ C_4
C_2 and C_3 ⇒ C_4



(a) 기존의 AND-OR 그래프 (b) EAOG

<그림 2> 기존의 AND-OR 그래프와 EAOG의 비교

III. EAOG를 이용한 전방향 추론

EAOG를 이용한 진방향 추론이란 주어진 데이터 노드에 대한 진위값에 따라 EAOG에 나타난 노드들 간의 연관관계를 통해 중간 또는 최종노드들의 진위값이 어떻게 변환되는 가를 규명하는 것이다. 본 장에서는 이러한 추론과정이 정수행렬 연산으로만 이루어지는 간편하고 효율적인 추론방식을 제시하고자 한다. 우선 EAOG를 이용한 진방향 추론방식을 제시하기 위해 필요한 몇가지 개념에 대해 정의하면 다음과 같다.

[정의 1] 규칙행렬 R : C_i , i = 1, ..., n을 주어진 EAOG에 나타난 노드들이라 하고, C_i 로부터 C_j 로 가는 호가 있을 경우 이를 $C_i \rightarrow C_j$ 로 나타낸다고 하자. 그러면 앞절에서도 설명하였듯이 $C_i \rightarrow C_j$ 는 C_j 가 단순노드인 경우 $C_i \Rightarrow C_j$ 를, C_j 가 복합노드일 경우 $C_i \in C_j$ 를 의미하게 된다. 이제 n개의 노드로 이루어진 EAOG에 대한 n × n 규칙행렬 R 은 다음과 같이 정의된다 :

$$R_{ij} = \begin{cases} 1, \text{ if } C_i \rightarrow C_j (\text{즉 } C_i \Rightarrow C_j \text{또는 } C_i \in C_j) \\ 0, otherwise \end{cases},$$

i, j = 1, ..., n ■

이렇게 정의된 규칙행렬 R은 주대각선이 모두 1인 특징을 갖는다 ([$C_i \Rightarrow C_i$] ≡ [$\sim C_i \vee C_i$] 로 항상 참이 되므로). 예로서 <그림 1>의 예제 EAOG에 대한 규칙행렬이 <그림 3>에 나타나 있다.

$$R = \begin{bmatrix} 1&0&0&0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&1&0&0 \\ 0&0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&1&0&0 \\ 0&0&0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&0&1 \end{bmatrix}$$

<그림 3> 예제 EAOG에 대한 규칙행렬 R

[정의 2] 충분조건 벡터 S : C_i , i = 1, ..., n 을 주어진 EAOG에 나타난 노드들이라 하자. 이 경우 1 × n 충분조건 벡터 $S = [S_1, \ldots, S_n]$ 은 다음과 같이 정의된다 :

$$S_i = \begin{cases} 1, & C_i\text{가 단순노드일 경우} \\ NP(C_i), & C_i\text{가 복합노드일 경우} \end{cases}, \ i = 1,..., n$$

여기서 NP(C_i) 는 노드 C_i로 들어오는 호의 수를 의미 ■

이상과 같이 정의된 충분조건 벡터 $S = [S_1, \ldots, S_n]$ 의 특성을 살펴보면 다음과 같다. 우선 데이터노드는 항상 단순노드이므로 임의의 데이터노드 C_i 에 대한 S_i 는 항상 1 이라고 정의되었음을 알 수 있다. 그리고 임의의 중간 또는 최종노드 C_i 에 대한 S_i 는 노드 C_i 가 참이 되기 위하여 선행적으로 참이 되어야 하는 전제노드의 수중 최소값을 의미한다. 중간 또는 최종노드 C_i 가 단순노드인 경우 C_i 의 전제노드 중 하나만 참이 되어도 C_i 는 참이 된다. 따라서 임의의 단순노드 C_i 에 대한 S_i 는 항상 1 이 된다. 반면에 노드 C_i 가 복합노드인 경우 C_i 의 전제노드 모두가 참이 되어야만 C_i 가 참이 될 수 있다. 따라서 임의의 복합노드 C_i 에 대한 S_i 는 C_i 로 들어오는 호의 수와 일치한다. 예를들어 <그림 1>의 예제 EAOG에 대한 S 는 다음과 같다 :

S = [1 1 1 1 1 1 1 2 2 2].

[정의 3] 진리상태 벡터 T : EAOG에 나타난 n 개의 노드가 갖는 진리상태(truth state)를 T라 하면, T = ($T_1, \ldots, T_n$) 으로 나타낼 수 있다. 이 때, T의 각 성분이 가지는 진리상태는 다음과 같이 정의된다 :

$$T_i = \begin{cases} 1, \text{ if } C_i \textit{ is true} \\ 0, \text{ if } C_i \textit{ is false or unknown} \end{cases}, \ i=1,..., n$$

앞서도 언급하였듯이 전방향 추론이란 주어진 데이터노드에 대한 진위값에 따라 EAOG에 나타난 노드들 간의 연관관계를 통해 중간 또는 최종노드들의 진위값이 어떻게 변환되는 가를 규명하는 것이다. 이러한 진리상태의 변환(transformation)은 규칙행렬 R과 진리상태벡터 T 및 최소 충분조건 벡터 S 를 이용하면 다음과 같이 간단한 정수 연산을 통해 처리할 수 있다.

T^k = ($T_1^k, \ldots, T_n^k$)를 현재의 진리상태라 하고, 벡터 U^k를 T^k 와 R의 곱 즉,

$$U^k = T^k \cdot R \qquad \text{(식 2)}$$

이라 하자. 그러면 U^k의 각 성분 U_i^k 는 노드 C_i 의 전제노드 중 참인 것의 수를 의미하게 된다. 그리고 T^{k+1} = ($T_1^{k+1}, \ldots, T_n^{k+1}$) 을

$$T_i^{k+1} = \begin{cases} 1, \text{ if } U_i^k \geq S_i \\ 0, \textit{otherwise} \end{cases}, \ i = 1,..., n \text{ (식 3)}$$

이라고 정의하면 T^{k+1} 은 다음과 같은 성질을 갖는다.

[명제 1] 노드 C_i 가 어떤 단순노드 C_j 의 전제노드이고, C_h, h = i^1, ..., i^p 이 어떤 복합노드 C_w의 모든 전제노드의 집합인 EAOG가 있다고 하자. 그러면 T_i^k = 1 이고, T_h^k = 1, h = i^1, ..., i^p 인 임의의 진리상태 벡터 T^k 에 대한 T^{k+1} 을 (식 2) 및 (식 3)을 이용하여 구하면 T_j^{k+1} 및 T_w^{k+1} 역시 1이 된다.

[증명] C_i가 단순노드 C_j의 전제노드이면 [정의 1] 및 [정의 2]에 의해 R_{ij} = 1, S_j = 1 이 된다. 여기서 T_i^k = 1 이면 (식 2)에 의해

$$U_j^k = \Sigma_{r=1}^n T_r^k \cdot R_{rj} \geq T_i^k \cdot R_{ij} (=1) \geq S_j (=1)$$

이 성립하여 (식 3)에 의해 T_j^{k+1} = 1 이 된다. 마찬가지로 C_h, h = i^1, ..., i^p 가 복합노드 C_w의 모든 전제노드의 집합이면 [정의 1] 및 [정의 2]에 의해 R_{hw} = 1, h = i^1, ..., i^p 이고 S_w = p 가 된다. 여기서 T_h^k = 1, h = i^1, ..., i^p 이면 (식 2)에 의해

$$U_w^k = \Sigma_{r=1}^n T_r^k \cdot R_{rw} \geq \Sigma_{h=i^1}^{i^p} T_h^k \cdot R_{hw} (=p) \geq S_w (=p)$$

가 성립하여 (식 3)에 의해 T_w^{k+1} = 1 이 된다. 즉 [명제 1]의 내용은 진리상태 T^k 에 의해 전제부분이 충족되는 노드의 진리상태는 T^{k+1}에서 참이라고 나타난다는 것이다. 따라서 EAOG를 바탕으로 전방향 추론을 하기 위해서는 실험 또는 관측을 통해 주어진 데이터노드의 진위값으로 이루어진 초기 진리상태벡터 T^0 부터 시작하여 T^k, k = 1, 2, ...를 진리상태가 더 이상 변하지 않을 때까지 계속 구해 나가면 된다. 예를들어 <그림 1>에

서 실험 또는 관측을 통해 데이터노드중 C_1, C_2 및 C_3 이 참이라고 주어졌다고 하자. 이 경우 최초 진리상태벡터

$$T^0 = [\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ]$$

이 된다. 이제 이를 바탕으로 (식 2) 및 (식 3)을 이용하여 T^k, k = 1, 2, ...를 차례로 구하면 다음과 같다.

(식 2) $U^0 = T^0 \cdot R = [\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 2\ 1\ 0\ ]$
(식 3) $T^1 = [\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ ]$
(식 2) $U^1 = T^1 \cdot R = [\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 2\ 1\ 0\ ]$
(식 3) $T^2 = [\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ ]$
(식 2) $U^2 = T^2 \cdot R = [\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 2\ 2\ 1\ ]$
(식 3) $T^3 = [\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ ]$
(식 2) $U^3 = T^3 \cdot R = [\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 2\ 2\ 1\ ]$
(식 3) $T^4 = [\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ ]$
(식 2) $U^4 = T^4 \cdot R = [\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 2\ 2\ 1\ ]$
(식 3) $T^5 = [\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ ] = T^4$

따라서 T^4 가 구하고자 하는 최종 진리상태벡터가 됨을 알 수 있다.

IV. EAOG를 이용한 역방향 추론 (Reverse Chainning)

앞절에서 설명한 전방향 추론방식은 추론과정이 정수연산으로만 이루어진다는 점에서 기존 방식에 비해 훨씬 효율적이라 할 수 있다. 하지만 주어진 EAOG의 규모가 증가하고 특히 EAOG에 나타난 규칙경로(modus ponens chain or implication path)가 길어짐에 따라 필요한 계산량도 비례적으로 증가한다. 본 장에서는 최종 진리상태벡터 T^F 를 규칙경로길이에 무관하게 보다 효율적으로 구하기 위한 새로운 기법인 역방향 추론방식을 제시하고자 한다.

1. 역방향추론의 기본 개념

역방향 추론의 기본 개념에 대해 설명하기 위하여 우선 최종 진리상태벡터 T^F 가 가지는 특성에 대해 살펴보면 다음과 같다.

[명제 2] 초기 진리상태 벡터를 T^0, 이를 바탕으로 (식 2) 및 (식 3)의 연속적인 적용을 통해 얻어진 최종 진리상태벡터를 T^F라 하고 S_j 및 S_k 를 각각 [정의 2]에 정의된 충분조건벡터 S 의 j번째 및 k번째 요소를 나타낸다고 하자. 그러면 임의의 진리상태 벡터 T가 T^F가 되기 위한 필요충분조건은 다음과 같다 :

(1) 모든 데이터노드 C_i 에 대해 $T_i \equiv T_i^0$ 이다. 즉 T에 나타난 데이터노드의 진리상태는 T^0와 일치 한다.

(2) T에 참이라고 나타난 모든 중간 또는 최종노드 C_j 에 대해 다음 사항이 성립한다. (C_j의 전제노드 중 T에 참이라고 나타난 것의 개수) $\geq S_j$.

(3) T에 거짓이라고 나타난 모든 중간 또는 최종노드 C_k 에 대해 다음 사항이 성립한다.(C_k의 전제노드 중 T에 참이라고 나타난 것의 개수) < S_k ■

전방향추론에서 사용한 초기 진리상태벡터 T^0 는 [명제 2]의 (조건 1)과 (조건 2)를 만족한다. 따라서 전방향추론은 (조건 1)과 (조건 2)를 만족하는 진리상태벡터로부터 출발하여 (조건 3)을 충족하지 않는 노드의 진위값을 단계적으로 바꾸어 줌으로써 세가지 조건 모두를 만족하는 최종 진리상태벡터 T^F를 구하는 방식이라 할 수 있다. 반면에 본절에서 제시하고자 하는 역방향추론은 전방향추론과는 반대로 [명제 2]의 (조건 1) 및 (조건 3)을 만족하는 초기 진리상태벡터 $\overline{T^0}$(전방향추론의 초기 진리상태벡터 T^0와 구분하기 위하여 $\overline{T^0}$ 로 표기함)를 구한 후 (조건 2)를 충족하지 않는 노드들의 진위값을 단계적으로 바꾸어 줌으로써 최종 진리상태벡터 T^F를 구하는 방식이다. 따라서 역방향추론은 크게 역방향추론을 위한 초기진리상태벡터 $\overline{T^0}$ 를 구하는 단계와 이를 바탕으로 최종 진리상태벡터 T^F를 구하는 단계로 나누어진다.

2. 단계 1 : 역방향추론을 위한 초기해 $\overline{T^0}$ 계산방법

역방향추론을 위한 초기 진리상태벡터, 즉 [명제 2]의 (조건 1)과 (조건 3)을 만족하는 초기해 $\overline{T^0}$ 를 구하는 방법은 여러 가지가 있을 수 있다. 가장 간단하게는 전방향추론을 위한 초기 진리상태벡터 T^0에서 중간노드 및 최종노드의 진위값을 모두 1로 바꾸어 주면 된다. 그런데 초기해 $\overline{T^0}$ 는 최종 진리상태벡터 T^F에 가까울수록 다시말해 [명제 2]의 (조건 1) 및 (조건 3)을 만족하면서 (조건 2)를 만족하지 않는 요소가 적으면 적을수록 효율성이 높다고 할 수 있다. 이런 관점에서 볼 때 앞서 언급한 간단한 방식에 의해 구해진 $\overline{T^0}$ 는 매우 비효율적이다. 본 논문에서 제시하는 역방향추론의 효율성은 초기해 $\overline{T^0}$의 효율성에 크게 좌우된다. 따라서 본절에서는 주어진 EAOG에 나타난 규칙경로(modus ponens chain or implication path)를 이용하여 효율적인 초기해 $\overline{T^0}$ 를 구하는 방식을 제시하고자 한다.

우선 1차 경로행렬 P를 다음과 같이 정의하자.

$$P_{ij} = \begin{cases} 1, \text{ if } C_i \rightarrow C_j \text{ and } i \neq j \\ 0, \textit{ otherwise} \end{cases}, \quad \text{i, j = 1, ..., n.}$$

이와같이 정의된 1차 경로행렬 P의 각 요소 P_{ij}는 주어진 EAOG에서 노드 C_i 로부터 노드 C_j로 연결되는 호가 존재하는가의 여부를 나타낸다. 즉 호가 존재하면 1이고 없으면 0이다. 즉 행렬 P는 [정의 1]에 정의된 규칙행렬 R의 대각선 요소를 모두 0 으로 바꾼 결과와 일치한다 (<그림 4 (a) 참조).

이제 1차 경로행렬 P의 곱 P^k의 의미에 대해 살펴보자. 우선 행렬 P의 제곱인 2차 경로행렬 P^2는 어떤 의미를 가지는가? P^2_{ij}를 행렬 P^2의 i번째 행과 j번째 열에 해당하는 요소라고 하면 이는 노드 C_i 로부터 노드 C_j로 2단계만에 연결되는 경로의 수를 의미한다. 참고로 <그림 1>의 예제 EAOG에 대한 P^2이 <그림 4 (b)>에 나타나 있다. <그림 4 (b)>에서 P^2_{67} 이 2 라는 것은 노드 C_6로부터 노드 C_7로 2단계만에 연결되는 경로가 두 개($C_6 \rightarrow C_9 \rightarrow C_7$ 및 $C_6 \rightarrow C_{10} \rightarrow C_7$)라는 것을 의미한다. 이러한 논의를 일반화하면 k차 경로행렬 P^k는 노드간에 k개의 호를 통해 연결되는 경로의 수를 표시한다는 것을 알 수 있다. 또한 L을 주어진 EAOG에 존재하는 최대경로라고 하면(<그림 1>의 예제 EAOG에 대한 L은 4가 됨), P^{L+1}은 모든 성분이 0인 행렬이 됨을 유추할 수 있다.

<그림 4> 예제 EAOG에 대한 1, 2차 경로행렬

$$A = \begin{bmatrix} 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \end{bmatrix} \quad A^2 = \begin{bmatrix} 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&2&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

(a) 1차 경로행렬 (b) 2차 경로행렬

이제 총경로행렬 Q를 다음과 같이 정의하자 :

$Q = \sum_{l=0}^{L} P^l$, 여기서 L은 최대경로의 수, P^0 는 I를 의미함 ■

이렇게 정의된 총경로행렬 Q의 모든 대각선 요소는 1이 되며, 대각선 요소가 아닌 $Q_{ij}\,(i \neq j)$는 노드 C_i 로부터 노드 C_j로 연결되는 모든 경로의 수를 의미한다. <그림 1>에 나타난 예제 EAOG에 대한 총경로행렬 Q가 <그림 5>에 예시되어 있다.

$$Q = \begin{bmatrix} 1&0&0&0&0&0&1&0&1&0 \\ 0&1&0&0&0&1&2&1&1&1 \\ 0&0&1&0&0&1&2&1&1&1 \\ 0&0&0&1&0&1&2&0&1&1 \\ 0&0&0&0&1&0&1&0&0&1 \\ 0&0&0&0&0&1&2&0&1&1 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&2&1&1&1 \\ 0&0&0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&0&1 \end{bmatrix}$$

<그림 5> 예제 EAOG에 대한 총경로행렬

$$Q = I + P + P^2 + P^3 + P^4$$

효율적인 초기해 $\overline{T^0}$ 를 구하기 위해서는 이상에서 정의된 총경로행렬 Q 외에 다음과 같이 최소경로벡터에 대한 정의가 필요하다.

[정의 4] 최소경로벡터 M : C_i , i = 1, ..., n 을 주어진 EAOG에 나타난 노드들이라 하자. 이 경우 1 × n 최소경로벡터 $M = [M_1, \ldots, M_n]$ 은 다음과 같이 정의된다. 우선 데이터노드 C_i 에 대한 최소경로 M_i 는 항상 1로 정의된다. 노드 C_i 가 중간 또는 최종노드일 경우 M_i 는 노드 C_i 가 추론을 통해 참이되기 위하여 반드시 참이 되어야 하는 데이터노드의 집합으로부터 노드 C_i 에 이르는 경로(path)의 수중 최소값을 나타낸다. 예를들어 <그림 1>에서 노드 C_6 가 참이 되기 위해 반드시 참이 되어야 하는 데이터노드의 집합은 { C_2, C_3} 또는 { C_4} 이다. 그리고 집합 { C_2, C_3}에 속하는 노드로부터 노드 C_6 에 이르는 경로는 두 개($C_2 \rightarrow C_8 \rightarrow C_6$ 및 $C_3 \rightarrow C_8 \rightarrow C_6$)이고, 집합 { C_4}에 속하는 노드로부터 노드 C_6 에 이르는 경로는 $C_4 \rightarrow C_6$ 하나이다. 따라서 노드 C_6 에 대한 최소경로 M_6의 값은 1 이 된다. 이러한 의미를 나타내는 최소경로벡터 M을 보다 엄밀하게 정의하면 다음과 같다 :

(1) C_i 가 데이터 노드인 경우: M_i = 1,

(2) C_i 가 데이터 노드가 아닌 경우:

C_j, j = $i^1, \ldots, i^j$ 를 노드 C_i 에 대한 전제노드들의 집합이라 하자. 그러면,

(i) C_i가 단순노드인 경우: $M_i = MIN_{j=i^1}^{i^j} \{ M_j \}$

(ii) C_i가 복합노드인 경우: $M_i = \Sigma_{j=i^1}^{i^j} M_j$

따라서 n 개의 노드 C_i, i = 1, ..., n 로 구성된 EAOG가 주어진 경우 M_i, i = 1, ..., n 는 위의 정의를 이용하여 EAOG의 하위 레벨에 속하는 노드부터 시작하여 차례로 구해 나가면 된다. 예를들어 <그림 1>의 예제 EAOG에 대한 최소경로벡터 M은 다음과 같이 구할 수 있다 :

데이터노드: $M_1 = M_2 = M_3 = M_4 = M_5$ = 1

M_8(복합노드) = M_2 + M_3 = 2

M_6(단순노드) = Min { M_4, M_8 } = 1

M_9(복합노드) = M_1 + M_6 = 2

M_{10}(복합노드) = M_6 + M_5 = 2

M_7(단순노드) = Min { M_9, M_{10} } = 2

=> M = [1 1 1 1 1 1 2 2 2 2].

이상에서 설명한 총경로행렬 Q 및 최소경로벡터

M이 주어졌을 경우 이를 바탕으로 초기해 $\overline{T^0}$를 구하는 절차는 다음과 같다. 추론을 위해 실험 또는 관측으로부터 얻어진 초기 진리상태를 T^0 라 하자. 그리고 벡터 W를 T^0 와 총경로행렬 Q의 곱, 즉

$$W = T^0 \cdot Q \qquad \text{(식 4)}$$

라 하자. 그러면 W의 각 요소 W_i, i = 1, ..., n 는 C_i 가 데이터노드일 경우에는 $W_i = T_i^0$ 가 되고, C_i 가 중간 또는 최종노드일 경우에는 T^0 에 참이라고 나타난 모든 데이터노드로부터 노드 C_i로 연결되는 모든 경로의 수를 의미한다. 이제 벡터 $\overline{T^0}$ 를 다음과 같이 정의하면,

$$\overline{T^0} = \begin{cases} 1, \text{ if } W_i \geq M_i \\ 0, \textit{otherwise} \end{cases}, \text{ i= 1, ..., n, (식 5)}$$

$\overline{T^0}$는 다음 명제에서 알 수 있듯이 역방향추론을 위한 초기 진리상태벡터가 되기 위한 조건, 즉 [명제 2]의 (조건 1) 및 (조건 3)을 만족한다.

[명제 3] (식 4) 및 (식 5)에 의해 구해진 $\overline{T^0}$는 [명제 2]의 (조건 1) 및 (조건 3)을 만족한다.

[증명] 우선 (조건 1) 즉 임의의 데이터노드 C_i 에 대해 $\overline{T_i^0} = T_i^0$ 가 성립함을 증명해 보자. (식 4)에 의해 $W_i = \Sigma_{h=1}^{n} T_i^0 \cdot Q_{hi}$ 이다. 그런데 C_i 가 데이터노드이므로 $h \neq i$ 일 경우 $Q_{hi} = 0$ 이다. 따라서 $W_i = T_i^0 \cdot Q_{ii} = T_i^0 \cdot 1 = T_i^0$ 이 성립한다. 또한 데이터노드 C_i 에 대한 $M_i = 1$ 이므로 (식 5)에 의해

$[\ T_i^0 = 1] \Rightarrow [\ W_i = 1] \Rightarrow [\ W_i] \Rightarrow [\ \overline{T_i^0} = 1]$ 및

$[\ T_i^0 = 0] \Rightarrow [\ W_i = 0] \Rightarrow [\ W_i] \Rightarrow [\ \overline{T_i^0} = 0]$

이 되어 항상 $\overline{T_i^0} = T_i^0$ 가 성립함을 알 수 있다. 이제 (조건 3)이 성립함을 증명해 보자. 노드 C_j를 임의의 중간 또는 최종노드라 하고 CNT_j를 C_j의 전제노드 중 $\overline{T^0}$ 에 참이라고 나타난 것의 개수라 할 때 (조건 3)의 내용은 $[\ \overline{T_j^0} = 0] \Rightarrow [\ CNT_j < S_j]$ 가 성립한다는 것이다. 이는 $[\ CNT_j \geq S_j] \Rightarrow [\ \overline{T_j^0} \neq 0]$ 와 동일하고 $[\ \overline{T_j^0} \neq 0] \equiv [\ W_j \geq M_j]$ 이므로 (조건 3)을 증명하기 위해서는 $[\ CNT_j \geq S_j] \Rightarrow [\ W_j \geq M_j]$ 가 성립함을 증명하면 된다.

임의의 데이터노드로부터 노드 C_j에 이르는 경로는 반드시 C_j의 전제노드를 거치게 된다. 따라서 C_j의 전제노드의 집합을 C_h, h = i^1, ..., i^j 라 하면 $W_j = \Sigma_{h=i^1}^{i^j} W_h$ 가 된다. 이를 바탕으로 C_j 가 단순노드일 경우와 복합노드일 경우로 나누어 $[\ CNT_j \geq S_j]$ 가 성립한다는 가정하에 $[\ W_j \geq M_j]$ 가 성립함을 증명하면 다음과 같다.

(i) C_j 가 단순노드일 경우

C_j가 단순노드일 경우 $S_j = 1$ ([정의 2]), $M_i = \text{Min}\{M_{i^1}, \ldots, M_{i^j}\}$ ([정의 4])이 된다. 여기서 $CNT_j \geq S_j$ 가 성립한다는 것은 W_h, h = i^1, ..., i^j 중 M_h, h = i^1, ..., i^j 보다 같거나 큰 것이 최소한 하나 이상 존재한다는 것을 의미한다. 따라서 이 경우

$W_j = \Sigma_{h=i^1}^{i^j} W_h \geq \text{Min}\{M_{i^1}, \ldots, M_{i^j}\} = M_i$ 가 성립한다.

(ii) C_j 가 복합노드일 경우

C_j가 복합노드일 경우 S_j 는 C_j의 모든 전제노드의 수가 되고([정의 2]), $M_i = \Sigma_{h=i^1}^{i^j} M_h$([정의 4])가 된다. 따라서 $CNT_j \geq S_j$ 가 성립한다는 것은 모든 W_h, h = i^1, ..., i^j 가 M_h, h = i^1, ..., i^j 보다 같거나 크다는 것을 의미한다. 따라서 이 경우

$W_j = \Sigma_{h=i^1}^{i^j} W_h \geq \Sigma_{h=i^1}^{i^j} M_h = M_i$ 가 성립한다 ■

이제 III절에서 사용한 예(<그림 1>의 예제 EAOG에서 초기 진리상태벡터 T^0 = [1 1 1 0 0 0 0 0 0 0]일 경우)에 대한 W 및 $\overline{T^0}$ 를 예시하면 다음과 같다 :

(식 4) W = $T^0 \cdot Q$ = [1 1 1 0 0 2 5 2 3 2]

(식 5) (W를 M과 비교하여) $\overline{T^0}$ = [1 1 1 0 0 1 1 1 1 1].

이상에서 구한 $\overline{T^0}$ 와 III절에서 구한 T^F = [1 1 1 0 0 1 1 1 1 0] 를 비교해 보면 $\overline{T^0}$ 가 T^F 와 일치하지는 않지만 매우 효율적임을 알 수 있다.

3. 단계 2 : 초기해 $\overline{T^0}$를 바탕으로한 역방향 추론 절차

앞절에서 구한 초기해 $\overline{T^0}$ 를 바탕으로 최종 진리상태벡터 T^F를 구하는 절차를 제시하기 위해서는 다음과 같이 역방향추론용 규칙행렬 $\overline{R}$ 에 대한 정의가 필요하다.

[정의 5] 역방향추론용 규칙행렬 $\overline{R}$: C_i , i

= 1, ..., n을 주어진 EAOG에 나타난 노드들이라 하고, C_i 로부터 C_j 로 가는 호가 있을 경우 이를 $C_i \rightarrow C_j$ 로 나타낸다고 하자. 그러면 n개의 노드로 이루어진 EAOG에 대한 n ×n 역방향추론용 규칙행렬 $\overline{R}$ 는 다음과 같이 정의된다 :

$$R_{ij}(i \neq j) = \begin{cases} 1, \text{ if } C_i \rightarrow C_j \\ 0, \textit{otherwise} \end{cases}, \ i, j = 1, ..., n,$$

$$R_{ii} = \begin{cases} 1, \text{ if } C_i\text{가 데이터노드} \\ 0, \textit{otherwise} \end{cases}, \ i = 1, .., n$$

이상과 같이 정의된 역방향추론용 규칙행렬 $\overline{R}$ 는 [정의 1]에 나타난 규칙행렬 R 의 대각선 요소 중 중간 또는 최종노드에 대한 값을 0으로 바꾸어 준 결과와 일치한다. 예를들어 <그림 1>의 예제 EAOG에 대한 역방향추론용 규칙행렬 $\overline{R}$ 가 <그림 6>에 나타나 있다.

$$\overline{R} = \begin{bmatrix} 1&0&0&0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&1&0&0 \\ 0&0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \end{bmatrix}$$

<그림 6> 예제 EAOG에 대한 $\overline{R}$

이제 이상에서 정의된 역방향추론용 규칙행렬 $\overline{R}$, 진리상태벡터 $\overline{T}$ (전방향추론과 구분하기 위해 T 대신 $\overline{T}$ 를 사용하였음) 및 [정의 2]에 정의된 충분조건벡터 S를 이용한 역방향 추론 절차를 설명하면 다음과 같다.

$\overline{T^k} = (\overline{T_1^k}, \ldots, \overline{T_n^k})$를 역방향 추론에 의한 현재의 진리상태라 하고, $\overline{U^k}$를 벡터 $\overline{T^k}$와 $\overline{R}$ 의 곱 즉,

$$\overline{U^k} = \overline{T^k} \cdot \overline{R} \qquad \text{(식 6)}$$

라 하자. 그러면 $\overline{U^k}$의 각 성분 $\overline{U_i^k}$ 는 노드 C_i 의 전제노드 중 $\overline{T^k}$에 참이라고 나타난 것의 수를 의미하게 된다. 그리고 $\overline{T^{k+1}} = (\overline{T_1^{k+1}}, \ldots, \overline{T_n^{k+1}})$ 을

$$\overline{T_i^{k+1}} = \begin{cases} 1, \text{ if } \overline{U_i}^k \geq S_i \\ 0, \textit{otherwise} \end{cases}, \ i = 1, .., n \text{ (식 7)}$$

이라고 정의하면, 진리상태 $\overline{T^k}$ 에 값이 0으로 나타난 노드 중 [명제 2]의 (조건 2)를 충족하지 않는 노드는 진리상태 $\overline{T^{k+1}}$에서 값이 1로 나타나게 된다. 따라서 역방향추론을 통해 최종 진리상태벡터 T^F 를 구하기 위해서는 초기해 $\overline{T^0}$ 을 바탕으로 (식 6) 및 (식 7)을 이용하여 $\overline{T^k}$, k = 1, 2, ...를 진리상태가 더 이상 변하지 않을 때까지 계속 구해 나가면 된다. 앞절에서 구한 초기해 $\overline{T^0}$ = [1 1 1 0 0 1 1 1 1 1]을 바탕으로 이상에서 제시한 역방향 추론과정을 예시하면 다음과 같다 :

(식 6) $\overline{U^0} = \overline{T^0} \cdot \overline{R}$ = [1 1 1 0 0 1 2 2 2 1]
(식 7) $\overline{T^1}$ = [1 1 1 0 0 1 1 1 1 0]
(식 6) $\overline{U^1} = \overline{T^1} \cdot \overline{R}$ = [1 1 1 0 0 1 1 2 2 1]
(식 7) $\overline{T^2}$ = [1 1 1 0 0 1 1 1 1 0] = $\overline{T^1}$

따라서 $\overline{T^1}$ 이 역방향 추론에 의해 구하고자 하는 최종 진리상태벡터가 되며 이는 전방향추론 결과와 일치함을 알 수 있다. 또한 진방향추론에 비해 최종진리상태에 아주 빨리 도달함을 알 수 있다.

V. 예제를 통한 역방향추론의 효율성 검증

본 장에서는 양도소득세 예제에 대한 EAOG를 바탕으로 역방향추론의 전방향추론에 대한 상대적 효율성을 비교해 보고자 한다. 본 연구에서 제시하는 양도소득세 사례는 이건창(1997)이 "세무자문을 위한 전문가 시스템 개발에 관한 연구"에서 제안한 세무자문 전문가시스템의 지식베이스를 활용하였다. 일반적으로 세무자문과 관련된 지식은 일반사용자들이 상용하기에는 복잡하며 전문성을 띄게 된다. 따라서 일반인이 쉽게 접근하기 어려운 세무분야는 이러한 전문가시스템의 활용이 절대적으로 요구된다. 그런 의미에서 본 연구에서는 이러한 세무자문과 관련된 지식을 일반 사용들이 쉽게 이용할 수 있도록 지식베이스화하고 이를 EAOG를 바탕으로 역방향추론의 전방향추론에 대한 상대적 효율성을 비교해 보고자 한다. 이를 위해 우선 역방향추론의 전방향추론에 대한 상대효율(Relative Efficiency of Reverse Chainning over Forward Chainning : 이하 RE(R/F)라 약함)을 다음과 같이 정의하자.

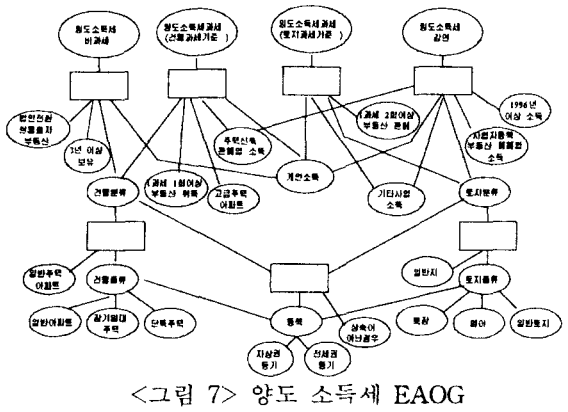
RE(R/F) = $\dfrac{\text{전방향추론으로 최종 진리상태벡터 } T^F\text{를 구하는 계산량}}{\text{역방향추론으로 최종 진리상태벡터 } T^F\text{를 구하는 계산량}}$

따라서 RE(R/F)의 값이 1보다 크면 클수록 전방향추론에 비해 역방향추론의 효율성이 상대적으로 높다고 할 수 있다. 그런데 역방향추론 과정에서 (식 6) 및 (식 7)을 이용하여 $\overline{T^k}$로부터 $\overline{T^{k+1}}$을 구하는데 필요한 계산량은 전방향추론 과정에서 (식 2) 및 (식 3)을 통해 T^k로부터 T^{k+1}을 구하는데 필요한 계산량과 같다. 다만 전방향추론에 비해 역방향추론은 (식 4) 및 (식 5)를

이용하여 초기해를 구하는 추가적인 계산이 필요하다. 그런데 (식 4) 및 (식 5)에 필요한 계산량은 각각 (식 2) 및 (식 3)에 필요한 계산량과 같다. 따라서 a를 전방향추론에서 $T^a = T^F$를 만족하는 가장 작은 자연수라 하고 b를 역방향추론에서 $\overline{T^b} = \overline{T^F}$를 만족하는 가장 작은 자연수라 하면 이 경우 RE(R/F)는 다음과 같이 계산된다 :

$$RE(R/F) = \frac{a}{b+1}.$$

일반적으로 양도소득세 지식베이스의 경우 그 추론 절차가 매우 복잡하다. 따라서 본 연구에서는 EAOG 추론 메카니즘의 효율성을 증명하기 위하여 지식베이스를 간단하게 수정하여 건물과 토지를 기준으로 일부분을 수정하여 다음 <그림 7>에 제시하였다.



<그림 7> 양도 소득세 EAOG

<그림 7>에 제시되어 있는 각 노드값은 의사결정자들이 입력할 수 있는 단순노드와 입력할 수 없는 복합 노드로 이루어져 있으며, 이 노드값을 통하여 EAOG 추론 메카니즘을 엑셀로 구현하였다. 다음 [표 2]에 제시되어 있는 결과는 건물에 대한 양도소득세 과세에 대한 추론 결과를 전방향 추론 결과와 역방향 추론 결과에 각각 제시한 것이다. 이때 초기값은 다음과 같다.

• T^0= [1 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

<그림 8> 양도소득세 과세에 대한 EAOG 추론 결과

(a) EAOG의 전방향 추론 결과

(b) EAOG 역방향 추론 결과

<그림 8(a)>의 결과에서 노드 C20, C24, C27, C30은 각각 양도소득세 비과세(건물), 양도소득세 과세(건물), 양도소득세 과세(토지), 양도소득세 감면(토지)를 나타내고 있다. 전방향 추론에 관하여 노드 C24의 최종진리값 벡터(T^6)가 1로 추론됨으로써, 해당 조건에 대해서는 건물에 대하여 양도소득세를 부과해야 한다는 추론 결과가 제시되어 있다. 또한 <그림 8(b)>에서 역방향 추론의 경우에도 노드 C24의 최종진리값 벡터($\overline{T^2}$)가 1로 추론된다. 이러한 데이터노드의 진위값의 결과를 통하여 역방향추론의 전방향추론에 대한 상대효율성을 설명하기 위하여 <그림 8>의 결과를 [표 2]에 제시하였다.

[표 2] <그림 7>의 양도소득세 EAOG에 대한 RE(R/F)

최종노드	최종 진리상태벡터 T^F		RE (R/F)
	전방향추론	역방향추론	
양도소득세 과세(건물)	T^6	$\overline{T^2}$	3
양도소득세 비과세	T^6	$\overline{T^2}$	3
양도소득세 과세(토지)	T^6	$\overline{T^2}$	3
양도소득세 감면	T^6	$\overline{T^2}$	3

[표 2]는 양도속득세에 대하여 EAOG를 추론한 결과를 요약하고 있는데, 이를 보면 역방향추론이 전방향추론에 비해 평균 3배의 효율성이 있음을 보여주고 있다. 하지만 주어진 EAOG의 규모가 클수록 특히 EAOG에 내재된 규칙경로가 길면 길수록 역방향추론의 효율성은 더욱 증대되리라 예상된다. 따라서 양도소득세 전체를 표현할 수 EAOG로 표현할 수 있으면 그 효율성은 매우 뛰어날 것이다. 최근 표면지식(Surface Knowledge) 또는 피상적 지식(Shallow Knowledge) 보다는 심층지식(Deep Knowledge)에 대한 관심이 고조되고 있다. 그런데 심층지식을 표현하는 EAOG에 내재된 규칙경로의 길이는 <그림 7>처럼 표면지식에 대한 EAOG에 비해 본질적으로 매우 길어지게 된다. 따라서 앞으로 심층지식에 대한 연구가 진행될수록 본 논문에서 제시된 역방향추론의 효율성은 더욱 증대되리라 판단된다.

VI. 결론 및 향후 연구방향

일반적으로 전자상거래 등에서 실시간으로 의사결정을 내려야 하는 경우 적절한 추론메카니즘이 필요하게 된다. 이에 본 연구에서는 기존의 AND-OR 그래프의 단점을 보완하고 복잡한 문제에 대하여 그 추론 과정이 더욱더 효율적이며 실시간으로 추론이 가능한 확장된 AND-OR 그래프 방식에 대하여 살펴보았다. 본 연구에서 제시한 추론 메카니즘은 크게 전방향 추론과 역방향 추론 두가지의 형태로 나눌 수가 있다. 전방향 추론이란 주어진 데이터노드에 대한 진위값에 따라 EAOG에 나타난 노드들 간의 연관관계를 통해 중간 또는 최종노드들의 진위값이 어떻게 변환되는 가를 규명하는 방식으로 추론과정이 정수행렬 연산으로만 이루어지는 간편하고 효율적인 추론방식이다. 반면에 역방향추론은 추론과정이 정수연산으로만 이루어진다는 점에서는 전방향 추론과 같으나, 주어진

EAOG의 규모가 증가하고 특히 EAOG에 나타난 규칙경로가 길어짐에 따라 필요한 계산량도 비례적으로 증가하는 전방향 추론의 단점을 보완하여 최종 진리상태벡터 T^F 를 제안함으로써 규칙경로길이에 무관하게 보다 효율적으로 구하기 위한 새로운 추론 방법이다. 또한 양도속득세 사례를 통하여 비전문가라도 EAOG 추론 메카니즘을 활용하여 전문적인 의사결정을 내릴 수 있다는 것을 증명하였다. 또한 본 사례를 전방향 추론과 역방향 추론에 모두 적용하여 추론 메카니즘의 효율성을 비교함으로써 역방향 추론이 그 처리 과정에서 더 효율적임을 증명하였다. 결론적으로 본 연구에서 제시한 추론방식은 추론과정이 정수연산으로만 이루어진다는 점에서 기존의 AND-OR 추론 방식에 비해 훨씬 효율적이라 할 수 있다. 향후 연구 방향으로는 보다 복잡하며 비구조적인 문제의 추론에 적합란 메카니즘의 확장이 필요하다고 하겠다.

참고문헌

이건창, *세무자문을 위한 전문가 시스템 개발에 관한 연구*, 한국조세연구원, 1997.

Chismar, W.G. and J. Meier, "A Model of Competing Interorganizational Systems and its Application to Airlines Reservation Systems", *Decision Support Systems*, 8(5), 1992, pp.447-458.

Giarratano, J. and G. Riley, *Expert Systems: Principles and Programming*, PWS Publishing Company, 1994.

Looney, C.G. andAlfize, A.A., "Logical Controls via Boolean Rule Matrix Transformations", *IEEE Transactions on Systems, man, and Cybernetics*, vol. SCM-17, no. 6, 1987, pp.1077-1082.

Kalakota, R. and A.B. Winston, *Frontiers of Electronic Commerce*, Reading, MA: Addison-Wesley, 1996.

Konskynski, B.R., "Strategic Control in the Extended Enterprise", *IBM Systems Journal*, 32(1), 1993, pp.111-142.