

An Extended AND-OR Graph-based Simulation and Electronic Commerce

Kun Chang Lee¹ & Hyung Rae Cho²

¹School of Business Administration, Sung Kyun Kwan University, Seoul 110-745, Korea
leekc@yurim.skku.ac.kr

²Department of Industrial Engineering, Kyung Sang National University, Chinjoo 660-701, Korea

Abstract

The objective of this paper is to propose an Extended AND-OR Graph (EAOG)-driven inferential simulation mechanism with which decision makers engaged in electronic commerce (EC) can effectively deal with complicated decision making problems. In the field of traditional expert systems research, AND-OR Graph approach has been suggested as a useful tool for representing the logic flowchart of the forward and/or backward chaining inference methods. However, the AND-OR Graph approach cannot be effectively used in the EC problems in which real-time problem-solving property should be highly required. In this sense, we propose the EAOG inference mechanism for EC problem-solving in which heuristic knowledge necessary for intelligent EC problem-solving can be represented in a form of matrix. 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.

We have proved the validity of our approach with several propositions and an illustrative EC example.

1. 서론

본 논문에서는 전자상거래에 관한 새로운 추론 시물레이션 메카니즘으로서 확장된 AND-OR 그래프 (EAOG: Extended AND-OR Graph) 에 의한 추론메카니즘을 제안한다. 지금까지 전자상거래 관련 연구는 소비자와 생산자간의 효과적인 연결이나 소비자의 만족 극대화를 위한 쇼핑몰 설계, 소비자들의 소비행동론적 연구 등에 치중을 해왔다 (Appelgate et al. 1996; Bakos 1999; Strader & Shaw 1997; Whinston 1997). 그러나, 본 연구에서는 추론 시물레이션 메카니즘을 제안하여 전자상거래 관련 당사자들이 가지고 있는 비구조적인 거래조건을 추론할 수 있도록 한다. 특히, 본 연구에서 제안하는 추론 시물레이션 메카니즘은 기존의 전문가시스템이 가지고 있는 패턴매칭의 방법을 지양하고, 전자상거래 상황에 걸맞는 실시간이면서 수치적인 추론이 가능하도록 하여 보다 효과적인 전자상거래 추론이 가능하도록 한다. 특히 본 연구에서 제안하는 추론 시물레이션 메카니즘은 전통적으로 전문가시스템에서 사용되어 왔던 방법과는 달리 (Giarratano & Riley 1994), 행렬에 기초하기 때문에 추론속도가 빨라 전자상거래 상황에 맞는다.

2. 확장된 AND-OR 그래프

본 논문에서 다루는 규칙의 기본 형태는 다음과 같다.

$$\text{IF } \alpha_1 \text{ and } \alpha_2 \text{ and } \dots \text{ THEN } \beta \quad (\text{식 1})$$

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.

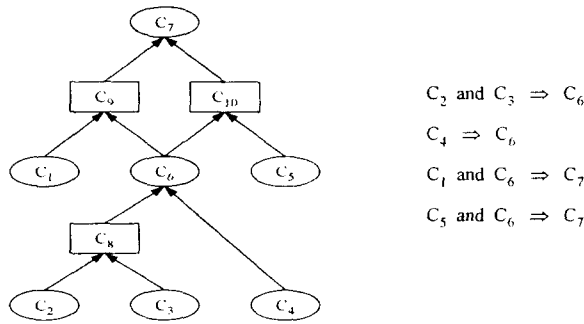
그런데 주어진 영역에서 문제를 해결하기 위한 규칙을 수집하면 항상 (식 1)과 같은 기본 형태를 띠고 있는 것은 아니다. 기본 형태가 아닌 규칙의 유형은 다음과 같다.

- Type 1 : $\alpha \Rightarrow (\beta \text{ and } \gamma)$,
- Type 2 : $(\alpha \text{ or } \beta) \Rightarrow \gamma$,
- Type 3 : $\alpha \text{ and } (\beta \text{ or } \gamma) \Rightarrow \delta$
- Type 4 : $\alpha \Rightarrow (\beta \text{ or } \gamma)$.

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 두 개의 기본 형태 규칙으로 나눌 수 있고, Type 2 는

$\alpha \Rightarrow \gamma$ 와 $\beta \Rightarrow \gamma$ 로 나눌 수 있다. 또한 Type 3 는 $(\alpha \text{ and } \beta) \Rightarrow \delta$ 와 $(\alpha \text{ and } \gamma) \Rightarrow \delta$ 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. 단순노드라 불리는 각 원은 하나의 조건을 나타내고 복합노드라 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. 예를들어 <그림 1>에서 데이터노드는 $C_1, C_2, C_3, C_4,$ 및 C_5 이다.

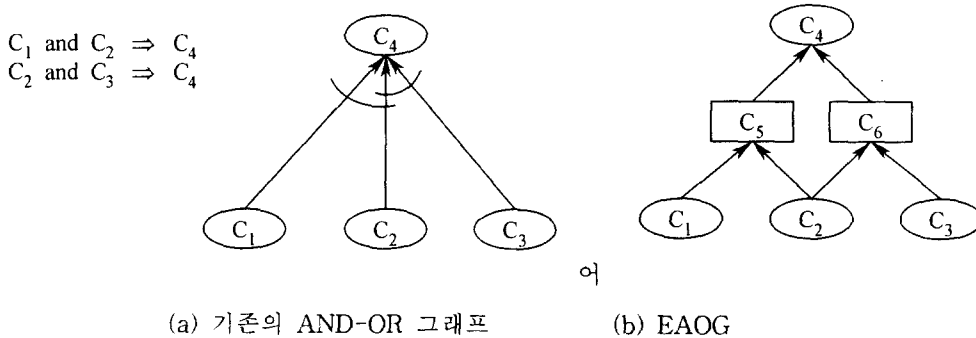
최종노드 : 들어오는 호만 있고 나가는 호는 없는 노드로서 주로 최종 관심의 대상이 되며 의사결정에 직접적인 영향을 미치는 사실을 나타낸다. 예를들어 <그림 1>에서 최종노드는 C_7 이다.

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 이루어



(a) 기존의 AND-OR 그래프

(b) EAOG

<그림 2> 기존의 AND-OR 그래프와 EAOG의 비교

제시된 추론방식의 핵심은 각 노드의 진위값을 해당 전제노드 중 참이라고 밝혀진 노드의 개수를 바탕

으로 판단한다는 것이다. 예를들어 <그림 2>에서 노드 C_4 가 참이 되기 위해서는 전제노드 중 2개 (C_1, C_2 또는 C_2, C_3)가 참이 되면 된다. 따라서 추론시 C_4 의 진위값은 전제노드중 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 이에따라 C_4 도 거짓이라는 추론이 정수행렬 연산만으로도 쉽게 이루어 질 수 있다.

3. EAOG를 이용한 추론 시물레이션 메카니즘

EAOG를 이용한 전방향 추론이란 주어진 데이터노드에 대한 진위값에 따라 EAOG에 나타난 노드들간의 연관관계를 통해 중간 또는 최종노드들의 진위값이 어떻게 변환되는가를 규명하는 것이다. 본 장에서는 이러한 추론과정이 정수행렬 연산으로만 이루어지는 간편하고 효율적인 추론방식을 제시하고자 한다. 우선 EAOG를 이용한 전방향 추론방식을 제시하기 위해 필요한 몇가지 개념에 대해 정의하면 다음과 같다.

[정의 1] 규칙행렬 $R : C_i, i = 1, \dots, n$ 을 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 단순노드일 경우 $C_i \Rightarrow C_j$ 를, C_j 가 복합노드일 경우 $C_i \in C_j$ 를 의미하게 된다. 이제 n 개의 노드로 이루어진 EAOG에 대한 $n \times n$ 규칙행렬 R 은 다음과 같이 정의된다 :

$$R_{ij} = \begin{cases} 1, & \text{if } C_i \rightarrow C_j \text{ (즉 } C_i \Rightarrow C_j \text{ 또는 } C_i \in C_j) \\ 0, & \text{otherwise} \end{cases}, \quad i, j = 1, \dots, n \blacksquare$$

이렇게 정의된 규칙행렬 R 은 주대각선이 모두 1인 특징을 갖는다 ($[C_i \Rightarrow C_j] \equiv [\sim C_i \vee C_j]$ 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 $S : C_i, i = 1, \dots, n$ 을 주어진 EAOG에 나타난 노드들이라 하자. 이 경우 $1 \times n$ 충분조건 벡터 $S = [S_1, \dots, S_n]$ 은 다음과 같이 정의된다 :

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

여기서 $NP(C_i)$ 는 노드 C_i 로 들어오는 호의 수를 의미 \blacksquare

이상과 같이 정의된 충분조건 벡터 $S = [S_1, \dots, S_n]$ 의 특성을 살펴보면 다음과 같다. 우선 데이터노드는 항상 단순노드이므로 임의의 데이터노드 C_i 에 대한 S_i 는 항상 1이라고 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 1이 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 T 라 하면, $T = (T_1, \dots, T_n)$ 으로 나타낼 수 있다. 이때, T 의 각 성분이 가지는 진리상태는 다음과 같이 정의된다 :

$$T_i = \begin{cases} 1, & \text{if } C_i \text{ is true} \\ 0, & \text{if } C_i \text{ is false or unknown} \end{cases}, \quad i = 1, \dots, n \blacksquare$$

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

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

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

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

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

이라고 정의하면 T^{k+1} 은 다음과 같은 성질을 갖는다.

[명제 1] 노드 C_i 가 어떤 단순노드 C_j 의 전제노드이고, $C_h, h = i^1, \dots, i^p$ 이 어떤 복합노드 C_w 의 모든 전제노드의 집합인 EAOG가 있다고 하자. 그러면 $T_i^k = 1$ 이고, $T_h^k = 1, h = i^1, \dots, i^p$ 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_i 가 단순노드 C_j 의 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

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

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

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

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 나타낸다는 것이다. 따라서 EAOG를 바탕으로 전방향 추론을 하기 위해서는 실험 또는 관측을 통해 주어진 데이터노드의 진위값으로 이루어진 초기 진리상태벡터 T^0 부터 시작하여 $T^k, k = 1, 2, \dots$ 를 진리상태가 더 이상 변하지 않을 때까지 계속 구해 나가면 된다. 예를들어 <그림 1>에서 실험 또는 관측을 통해 데이터노드중 C_1, C_2 및 C_3 이 참이라고 주어졌다고 하자. 이 경우 최초 진리상태벡터

$$T^0 = [1110000000]$$

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

$$(\text{식 2}) \quad U^0 = T^0 \cdot R = [1110000210]$$

$$(\text{식 3}) \quad T^1 = [1110000100]$$

$$(\text{식 2}) \quad U^1 = T^1 \cdot R = [1110010210]$$

$$(\text{식 3}) \quad T^2 = [1110010100]$$

$$(\text{식 2}) \quad U^2 = T^2 \cdot R = [1110010221]$$

$$(\text{식 3}) \quad T^3 = [1110010110]$$

$$(\text{식 2}) \quad U^3 = T^3 \cdot R = [1110011221]$$

$$(\text{식 3}) \quad T^4 = [1110011110]$$

$$(\text{식 2}) \quad U^4 = T^4 \cdot R = [1110011221]$$

$$(\text{식 3}) \quad T^5 = [1110011110] = T^4$$

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

4. 전자상거래 응용사례

본 장에서는 지금까지 서술한 EAOG 기법을 활용하여 전자상거래에서 구매기업과 공급기업의 실시간 계약체결을 위한 응용사례를 고려하여 보자. 전자상거래에서는 구매기업과 공급기업이 지역적으로 떨어져 있으므로 거래조건을 문서, 팩스, 전자우편 등을 사용하여 확인함으로 거래체결까지 많은 시간이 소요된다. 또한 거래 당사자들이 거래 협상에 있어 관련된 많은 조건들이 연관되어 있기 때문에 신속한 거래 체결이 어렵다. 따라서, 이러한 다양한 조건들을 유기적으로 반영하여 거래체결 시간을 단축시키기 위해서는 거래 당사자들간의 조건을 신속하게 반영하여 거래체결 여부를 확인시켜주는 메커니즘이 필요하다. 이에 EAOG 알고리즘은 이러한 전자상거래의 거래 협상에서도 뛰어난 기능을 보여줄 수 있다. 즉, EAOG를 이용함으로써 공급자와 구매자는 거래의 부결 시 그 원인이 어디에 있는지 실시간으로 파악하여 자신의 조건을 계속적으로 변경하면서 보다 빠른 시간 안에 거래를 체결할 수 있다. 다음 사례는 EAOG를 이용한 전자상거래의 거래 협상과정을 소개하고 있다.

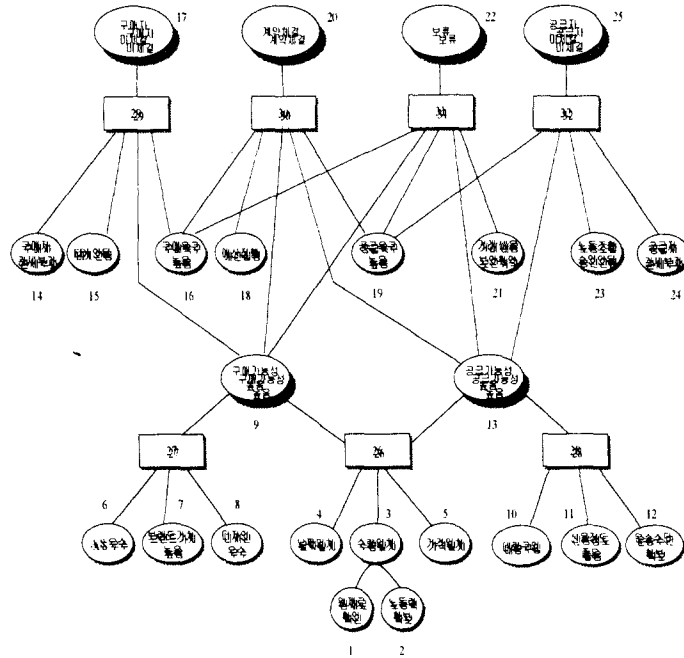
본 사례에서 소개하는 전자상거래의 계약 체결과정은 공급기업과 구매기업이 서로의 거래조건을 EAOG의 메커니즘에 입력하면서부터 시작된다. 이때 문제를 단순화 시키기 위하여 공급자와 구매자가 동시에 자신의 거래 조건을 입력한다고 가정한다. 이렇게 입력된 거래조건을 바탕으로 인터넷 상에서 EAOG에 의한 실시간 협상이 이루어지고 계약체결 여부는 계약의 체결, 미체결, 보류 중에서 하나의 값으로 결정된다. 이 때, 계약이 미체결 되거나 보류될 경우에는 그 원인이 구매기업에 있는지 또는 공급기업에 있는지 나타나기 때문에 거래 당사자들은 거래 조건을 신속히 변경하여 다시 협상에 임할 수 있다. 이와 같은 전자상거래의 계약체결에 있어서 고려해볼 수 있는 조건들은 다음 [표 1]과 같이 고려될 수 있다.

[표 1] 구매기업과 공급기업의 거래조건 자료

공급기업		구매기업	
노드이름	설명	노드이름	설명
가격	가격	가격	가격
수량	공급수량	수량	주문수량
원재료	원재료 확인	브랜드가치	브랜드가치의 선호
노동력	노동력 확보여부	디자인우수	디자인우수
운송수단	운송수단의 확보여부	A/S	A/S정도
거래날짜	납기일	전자지불	전자지불시스템 유무여부
대량구입	대량구입정도	예산집행	구매자의 예산집행여부
신용	구매자의 신용정도	관세	구매자 관세부담여부
관세	공급자 관세부담여부	보안	거래내용의 보안여부
공급욕구	공급자의 공급욕구정도	구매욕구	구매자의 구매욕구정도

[표 1]을 살펴보면 공급기업은 계약의 체결을 위하여 원재료, 노동력확보, 수량, 가격, 거래날짜, 대량구입정도, 신용정도, 운송수단확보, 노동조합, 공급자 관세, 공급욕구 등의 조건을 통합적으로 고려하여야 한다. 구매기업 역시 공급기업에서 제시한 수량, 가격, 거래날짜, 대량구입정도, 신용정도의 조건에다 관세, 전자지불, 구매욕구, 예산집행 등의 조건을 고려하여야 한다. 이와 같은 전자상거래에서 구매기업과 공급기업의 거래조건을 확장된 AND-OR 그래프로 나타내면 [그림 4]와 같다.

[그림 4] EAOG를 활용한 전자상거래의 사례



이러한 전자상거래에 있어서 구매기업과 공급기업간의 거래체결과정을 나타낸 확장된 AND-OR 그래프를 바탕으로 규칙행렬을 작성하면 [그림 5]와 같다.

[그림 5] 전자상거래에서의 거래체결을 위한 규칙 행렬

R	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30	C31	C32
C1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
C4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
C5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
C6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
C7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
C8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
C9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
C10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
C11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
C12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
C13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
C14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
C15	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	0	0	0	1	0	0	
C16	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	0	1	1	1	0	
C17	0	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	0	0	0	0	
C18	0	0	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	
C19	0	0	0	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	1	1	1	
C20	0	0	0	0	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	0	
C21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
C22	0	0	0	0	0	0	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	
C23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
C24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	
C25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
C26	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
C27	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
C28	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
C29	0	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	
C30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	
C31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	
C32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	

각각의 C값이 의미하는 바는 [표 2]와 같다.

