韓國IT서비스學會誌
第9卷 第3號
2010年 9月

백트래킹 기법을 이용한 불확정성 하에서의 역방향추론 방법에 대한 연구

송용욱* · 신현식**

Development of a Backward Chaining Inference Methodology Considering Unknown Facts Based on Backtrack Technique

Yong Uk Song* · Hyunsik Shin**

■ Abstract ■

As knowledge becomes a critical success factor of companies nowadays, lots of rule-based systems have been and are being developed to support their activities. Large number of rule-based systems serve as Web sites to advise, or recommend their customers. They usually use a backward chaining inference algorithm based on backtrack to implement those interactive Web-enabled rule-based systems. However, when the users like customers are using these systems interactively, it happens frequently where the users do not know some of the answers for the questions from the rule-based systems. We are going to design a backward chaining inference methodology considering unknown facts based on backtrack technique. Firstly, we review exact and inexact reasoning. After that, we develop a backward chaining inference algorithm for exact reasoning based on backtrack, and then, extend the algorithm so that it can consider unknown facts and reduce its search space. The algorithm speeded-up inference and decreased interaction time with users by eliminating unnecessary questions and answers. We expect that the Web-enabled rule-based systems implemented by our methodology would improve users' satisfaction and make companies' competitiveness.

Keyword : Backtrack, Backward chaining, Inference, Rule, Uncertainty

논문투고일 : 2010년 08월 08일 논문수정완료일 : 2010년 08월 20일 논문게재확정일 : 2010년 09월 10일
* 연세대학교 원주캠퍼스 경영학부
** CJ오쇼핑 경영지원실, 교신저자

1. 서 론

기업 및 조직에서의 정보기술 활용 단계는 단위 업무 전산화, 전사 차원의 종합 정보화단계를 넘어 지식화 및 지능화의 단계로 들어서고 있다. 컴퓨터 기술 도입 초기의 단위 업무 전산화에 만족하던 기업들은 1990년대 중반 이후 전사적 자원관리(ERP, Enterprise Resource Planning)의 보급에 따라 전사 차원에서의 최적화된 업무 프로세스 구축 및 이를 통한 종합적 정보화를 달성할 수 있었다. 이후 기업들은 다시 정보기술을 활용한 새로운 경쟁우위 강화 가능성을 추구하기 시작했는데, 그 중 유력한 분야의 하나로 대두된 것이 지식경쟁력 강화이다. 보유하고 있는 수많은 자료와 정보로부터 지식을 추출하여 활용할 수 있는 기업은 그렇지 못한 기업보다 강한 경쟁력을 가질 수 있고, 이것은 기업의 생존 및 이윤 창출에 중대한 영향을 미치게 되었는데, 이것을 가능하게 해 주는 중요한 정보기술의 하나가 데이터마이닝(data mining)이다. 데이터마이닝은 대용량 데이터베이스가 구축된 거의 모든 분야에 응용될 수 있는데, 대표적인 응용 분야로는 카드 도용 방지(fraud detection), 위험 관리(risk management), 고객 불만 예방(claim prevention), 고객 유치 및 유지(customer acquisition and retention), 고객 세분화 및 프로파일링(customer segmentation and profiling), 마케팅 효과분석(campaign effect analysis), 목표 마케팅(target marketing), 교차 판매(cross-selling/up selling) 등을 들 수 있다[5, 7].

데이터마이닝의 결과로 만들어진 지식은 여러 가지 형태로 표현될 수 있는데, 그 중에서도 'IF A THEN B' 형태의 규칙이 전형적 지식표현 방법 중 하나로 많이 사용되고 있다. 이러한 규칙은 기계학습(machine learning) 기법 중에서는 C4.5, C5.0, CART, CHAID, QUEST 등 의사결정나무 (decision tree) 분석[20, 21]의 결과로 얻을 수 있고, 의미는 약간 다르지만 연관 규칙(association rule) 분석의 결과로도 얻어진다. 또한 현장 전문가(domain expert)로부터 직접 얻어지는 지식은 대부분 규칙의 형태를 가지고 있다. 이렇게 획득된 규칙들은 규칙베이스(rule-base)에 저장되어 규칙기반 지능형 시스템 또는 지능형 웹 사이트[1-3, 8, 9, 17] 등을 통하여 활용된다. 이 중 자문, 상담, 추천 등의 서비스, 특히 웹 사이트 등을 통하여 제공되는 서비스는 사용자와의 대화 형식을 통하여 제공되는데, 이때 규칙베이스로부터 사용자에 대한 질문을 생성한 후 사용자의 답변을 이용하여 새로운 사실을 얻어내는 지식시스템의 핵심 모듈이 추론기관(inference engine)이다. 이러한 대화형 추론기관은 대체로 역방향추론 (backward chaining inference) 방법으로 구현하며, 그 추론 알고리즘은 탐색 기법 중 하나인 백트래킹(backtracking) 기법을 바탕으로 한다.

그런데, 기업의 고객과 같이 일반 사용자를 대상으로 대화형 지식시스템을 구축할 경우 부딪치는 문제 중의 하나는 추론기관이 질문한 문제에 대하여 사용자가 그 답을 모르는 경우가 발생한다는 것이다. 기업 내부 직원용 대화형 지식시스템의 경우에는 그 사용자, 즉 직원이 질문에 대한 답을 모르는 경우도 별로 없고, 혹시 모를 경우에도 다른 직원 또는 전담부서의 도움을 받아 답을 구할 수 있다. 그러나 기업 외부 고객용 대화형 지식시스템의 경우에는 그렇지 못한 경우가 종종 발생한다. 이러한 문제를 본 논문에서는 불확실성 중 특히 불확정성이라고 부르며, 이러한 불확정성이 존재하는 상황에서 백트래킹 기법을 이용한 역방향 추론 방법론을 고안하는 것이 본 연구의 목적이다.

이를 위한 본 논문의 구성은 다음과 같다. 먼저 제 2장에서는 추론의 개념, 역방향 및 정방향 추론, 백트래킹 기법, 불확실성 하에서의 추론 등에 대한 기존 연구들을 살펴본다. 이를 바탕으로 제 3장에서는 확실성 하에서의 역방향추론 알고리즘을 제시한 후 이를 확장하여 불확정성 하에서의 역방향추론 알고리즘을 제시한다. 이어서, 역방향추론에서 사용하는 백트래킹 기법의 탐색 공간을 줄이는 방법을 토의한 후 다시 이를 반영하여 확장된

역방향추론 알고리즘을 제시한다. 제 4장에서는 제 3장에서 제시된 역방향추론 알고리즘의 장점 및 단점에 대하여 토의한 후, 제 5장에서 결론을 맺도록 하겠다.

2. 기존 연구

2.1 추론

추론이란 이미 알고 있는 사실과 지식으로부터 새로운 사실을 유추해 내는 것을 말한다[7]. 넓은 의미에서 추론은 주어진 지식표현 상에서 원하는 결론이나 해답을 도출하기 위한 모든 작업으로 볼 수 있으나, 본 연구는 규칙(rule) 형태의 지식표현 상에서의 추론에 초점을 맞추도록 한다. 규칙은 'IF A THEN B'의 형태 또는 'A => B'의 형태로 표현되며, IF 구문의 조건이 만족되거나 발생한 상태이면 THEN 구문이 수행되거나 논리적으로 참(true)이 되는 형태이다. 대부분의 현존하는 지식들이 상황(또는 조건)과 이에 따른 행동(또는 결론)의 형태를 띠고 있기 때문에 규칙이 지식표현의 수단으로 널리 사용되고 있다.

규칙 형태의 지식표현에서 기본적으로 사용되는 추론 기법은 긍정식(modus ponens)과 삼단논법이다. 긍정식은 'A => B'라는 규칙이 있고 A가 사실일 때 B라고 결론 내리는 것을 말한다. 예를 들어 "어떤 동물이 새이다 => 그 동물은 알을 낳는다"라는 규칙이 있고 "어떤 동물이 새"라면 "그 동물이 알을 낳는다"라고 결론 내릴 수 있으며, 이로써 "그 동물이 알을 낳는다"라는 새로운 사실을 유추해 냈다고 할 수 있다. 삼단논법은 연쇄규칙 (chain rule)이라고도 하며, 한 규칙의 결론부분과 다른 규칙의 조건부분이 같을 경우 이 두 규칙을 연쇄시켜 새로운 규칙을 도출해 내는 것을 말한다. 즉, 'A => B'와 'B => C'와 같이 첫 번째 규칙의 결론부분과 두 번째 규칙의 조건부분이 'B'로서 동일할 때 'A => C'라는 새로운 규칙을 만들어 낼 수 있다. 예를 들어 "그는 소크라테스이다 => 그는 사람이다"와 "그는 사람이다 => 그는 죽는다"의 두 규칙으로부터 "그는 소크라테스이다 => 그는 죽는다"라는 새로운 규칙을 도출해 낼 수 있다.

규칙기반 시스템(rule-based systems)은 주어진 규칙과 이미 알고 있는 사실들로부터 긍정식과 삼단논법을 적용하여 새로운 사실을 찾아주는 정보시스템을 말한다. 규칙기반 시스템에서 규칙을 저장한 곳을 규칙베이스(rule-base)라고 하고, 사실을 저장한 곳을 사실베이스(fact-base)라고 한다. 규칙베이스로부터 규칙을, 사실베이스로부터 사실을 검색(retrieve)한 후 추론 기법을 적용하여 새로운 사실을 유추하고 그 새로운 사실을 다시 사실베이스에 저장(store)하는 규칙기반 시스템의 한 부분을 추론기관(inference engine)이라고 부른다. 즉, 추론기관은 규칙베이스와 사실베이스의 정보(규칙과 사실) 및 사용자와의 인터페이스(interface)를 통하여 사용자가 알고자 하는 새로운 사실을 도출한 후 사실베이스에 저장하는, 규칙기반 시스템의 핵심 모듈이다.

2.2 정방향추론과 역방향추론

추론기관을 구현하기 위해서는 규칙베이스와 사실베이스의 여러 규칙과 사실들을 어떤 순서로 검색한 후 긍정식 및 삼단논법 등의 추론 기법을 어떻게 적용하여 사용자가 알고자 하는 새로운 사실을 도출할 것인가에 대한 방법, 즉 추론 방법을 고안하여야 한다. 추론 방법에는 크게 정방향추론(forward chaining inference)과 역방향추론(backward chaining inference)이 있다[7]. 규칙베이스의 규칙들은 삼단논법에 의하여 서로 연쇄(chain)되어 있으며, 연쇄된 규칙들에 대하여 긍정식을 연속적으로 적용하면 사용자가 알고자하는 최종 결론을 도출할 수 있게 된다. 여기서 추론기관 관점에서 중요한 것은 궁극적으로 사용자가 알고자하는 사실을 얻어내는 것이다. 즉, 규칙기반 시스템에서 추론이란 단순히 긍정식을 무조건적으로 적용하여 새로운 사실들을 방향성 없이 얻어내는 데

의미가 있는 것이 아니라, 사용자가 처한 특정한 상황에서 필요한 특정한 사실을 도출 또는 확인하는 데 의미가 있다.

정방향추론이란 기본적으로 사용자가 원하는 최종 결론을 얻어낼 때까지 긍정식을 연속적으로 적용하는 방법이다. 이 방법에서는 사용자가 추론하고자하는 문제와 관련하여 알고 있는 모든 사실들을 먼저 제공하여야 한다. 그러면 추론기관은 규칙베이스에 있는 규칙들 중에서 그 조건부분이 각각의 사실과 일치하는 것을 찾아 해당 규칙을 수행하여 그 규칙의 결론부분을 새로운 사실로서 사실베이스에 추가한다. 이러한 과정은 사용자가 원하는 최종 결론이 얻어질 때까지 또는 더 이상 새로운 사실을 알아낼 수 없을 때까지 진행된다. 역방향추론이란 사용자가 원하는 최종 결론을 가설로하여 이 가설을 지지하는 사실을 역으로 추적해 가는 방법이다. 이 추론의 시작점을 목표(goal)라고 하며, 추론의 방향은 이 목표를 지지하는 하위 목표 또는 사실들이 참인지를 알아보는 방향으로 진행된다. 예를 들어 'A => B'와 'B => C'의 규칙이 있고, 사용자가 C가 사실인지를 확인하고자 한다고 하자. 이 때 C가 목표가 되며, C가 사실이기 위해서는 긍정식에 의하여 B가 사실이어야 한다. 그러면 B가 새로운 목표 (하위 목표, subgoal)가 되고, 다시 긍정식에 의하여 A가 사실이어야 한다. 이 시점에서 만약 사용자가 A가 사실이라는 것을 제공하게 되면 C가 사실임이 확인되고, 만약 사용자가 A가 사실이라는 것을 제공하지 못하게 되면 C가 사실임이 확인되지 않게 된다.

2.3 백트래킹(backtracking)

실세계에서는 수많은 규칙과 사실들로부터 추론을 하게 되며, 추론기관이 이것을 적절히 지원하기 위해서는 효율적인 알고리즘이 필요하게 된다. 역방향추론을 구현하기 위하여 가장 대표적으로 사용되는 알고리즘이 백트래킹(backtracking) 방법이다. 백트래킹은 검색, 최적화, 의사결정 등의 문제에 광범위하게 적용할 수 있는 문제해결 방법이다. 검색, 최적화, 의사결정 등의 문제는 다음과 같이 정형화된다[11, 15].

n 개의 공간(selection spaces) X_1, X_2, …, X_n의 원소들의 순서쌍으로 이루어진 공간(product spaces) $X_1 \times X_2 \times \cdots \times X_n$에서 기준함수(criterion function) $\phi(x_1, x_2, \cdots, x_n)$를 최대화하는 벡터해 (sample vector) $(x_1, x_2, \cdots, x_n)$를 찾는다.

단, 여기서 x_1, x_2, …, x_n은 각각 X_1, X_2, …, X_n의 원소들이다. 다시 말하면, $X_1 \times X_2 \times \cdots \times X_n$의 순서쌍 중에서 $\phi(x_1, x_2, \cdots, x_n)$를 최대화하는 벡터 $(x_1, x_2, \cdots, x_n)$를 찾는 문제이다. 여기서 각 X_i의 원소들의 숫자는 유한하다고 가정한다.

X_i의 원소들의 숫자를 M_i라고 하고, $M = \Pi_{i=1}^{n} M_i$라고 하자. 그러면, 이 문제의 가장 단순한 접근방법은 M개의 가능한 모든 벡터(순서쌍)들에 대하여 기준함수 ϕ를 계산한 후 가장 큰 값을 갖는 벡터를 찾는 방법이다. 그러나 항상 그렇듯이, n이나 M_i들이 커질 경우 M이 매우 커지게 되어 위 방법에 의하여 문제를 해결하는 것은 매우 많은 시간을 필요로 하게 된다. 백트래킹은 M보다는 훨씬 적은 시도로 동일한 답을 얻기 위하여 고안된 알고리즘이다. 백트래킹의 기본 아이디어는 변형된 기준함수를 이용하여 벡터해의 각 원소를 한 개씩 추가해 나가는 것이다. 이때 변형된 기준함수는 현재까지 만들어진 부분 벡터해(예를 들어, $(x_1, x_2, -, \cdots, -)$)가 최적해가 될 가능성이 있는지를 알려줄 수 있도록 고안된다. 예를 들어, 만약 부분 벡터해$(x_1, x_2, -, \cdots, -)$가 최적해가 될 가능성이 없다면$(x_1, x_2, ?, \cdots, ?)$ 형태의 다른 벡터들에 대해서는 기준함수를 계산해 볼 필요가 없다는 뜻이고, 따라서 $\Pi_{i=3}^{n} M_i = M/M_1M_2$개의 계산을 해보지 않아도 되므로 그만큼의 계산 시간을 절약할 수 있게 된다. 잘 고안된 백트래킹 방법은 전체 순서쌍 집합 중 상당히 많은 부분을 계산에서 제외할 수 있는 방법이라 할 수 있다.

백트래킹을 구체적으로 설명하기 위하여 n개의 공간(selection spaces) X_1, X_2, …, X_n이 부분 벡터해를 만들어 나가는 순서대로 정렬되어 있다고 가정하자. 실제에서는 이 순서가 미리 정해질 수도 있고, 또는 백트래킹 중에 차츰차츰 정해져 나갈 수도 있다. 먼저 X_1의 첫 번째 원소 x_{11}을 선택한 후 변형된 기준함수 $\phi_1(x_{11})$의 값을 구한다. 기준함수의 결과값에 의하여 부분 벡터해(x_{11}, -, -, …, -)가 최적해가 될 가능성이 없다면 두 번째 원소 x_{12}를 선택한 후 다시 변형된 기준함수 $\phi_1(x_{12})$의 값을 구한다. 이러한 과정을 거쳐 처음으로 부분 벡터해(x_{1j}, -, -, …, -)가 최적해가 될 가능성이 있다고 밝혀졌다고 하자. 그러면, 다시 X_2의 첫 번째 원소 x_{21}을 선택한 후 변형된 기준함수 $\phi_2(x_{1j}, x_{21})$의 값을 구한다. 기준함수의 결과값에 의하여 부분 벡터해(x_{1j}, x_{21}, -, …, -)가 최적해가 될 가능성이 없다면 두 번째 원소 x_{22}를 선택한 후 다시 변형된 기준함수 $\phi_2(x_{1j}, x_{22})$의 값을 구한다. 이러한 과정을 거쳐 부분 벡터해(x_{1j}, x_{2k}, x_{3l}, -, …, -)를 만들어 간다. 그런데, 만약 X_4의 모든 원소 x_{4m}에 대하여 $\phi_4(x_{1j}, x_{2k}, x_{3l}, x_{4m})$ 값을 구한 결과 모두가 최적해가 될 가능성이 없다고 나왔다고 하자. 그러면, 부분 벡터해(x_{1j}, x_{2k}, x_{3l}, -, …, -)으로 돌아가서(backtrack), X_3의 원소 중 x_{3l} 다음의 원소들에서 최적해가 될 가능성이 있는 원소를 찾는다. 만약 있다면, 그 원소를 3번째 원소로 하는 부분 최적해를 만들고, 다시 X_4의 원소에 대하여 부분 벡터해를 찾는 작업을 계속한다. 만약 없다면, 부분 벡터해 (x_{1j}, x_{2k}, -, …, -)으로 돌아가서, X_2의 원소 중 x_{2k} 다음의 원소들에서 최적해가 될 가능성이 있는 원소를 찾는 작업을 반복한다. 이러한 과정을 거쳐 $\phi_n(x_1, x_2, \cdots, x_n)$을 만족하는 최적 벡터해($x_1$, x_2, …, x_n)를 찾게 된다. 하나의 최적 벡터해를 찾은 후 전체 최적 벡터해들을 전부 찾을 때까지 이 과정을 반복할 수도 있고, 아니면 하나의 해만 찾으면 되는 상황이라면 거기서 작업을 멈출 수도 있다.

2.4 역방향 추론과 백트래킹

백트래킹 방법은 역방향추론을 효과적으로 구현할 수 있는 방법이다. 예를 들어 참(true) 또는 거짓(false) 값을 갖는 n개의 조건문 X_i로 이루어진 규칙베이스가 있다고 하자. 예를 들어, 다음과 같은 4개의 규칙이 있다고 하자.

(r_1) A => C
(r_2) D => E
(r_3) B and C => F
(r_4) E or F => G

여기서 결론문 C, E, F, G를 제외한 A, B, D들이 X_i에 해당한다. 각 X_i들의 값은 t(참) 또는 f(거짓)으로 유한하다. 여기서 G가 참인지 거짓인지를 아는 것이 문제라고 할 때, 가장 단순한 접근은 (t, t, t), (t, t, f), …, (f, f, f) 등 $2^3 = 8$개의 원소로 이루어진 순서쌍 집합의 각 원소들을 위 4개의 규칙에 맞추어 보는 것이다. 이미 말했듯이 일반적으로 규칙베이스에는 수많은 규칙과 조건문이 있기 때문에 위와 같은 접근법은 매우 많은 계산과, 따라서, 긴 시간을 필요로 할 수 있다. 이 경우 백트래킹 방법이 많은 계산과 시간을 절약해 줄 수 있다.

먼저 부분 최적해를 만들어 나가는 X_i의 순서를 정렬하는 방법을 생각해 보자. 이것은 목표(goal)를 중심으로 규칙 연쇄(chain)를 따라 정해나가면 된다. 위의 예에서 목표는 G가 참인지를 아는 것이다. G가 참이기 위해서는 규칙 r_4에 의해 E 또는 F가 참이어야 한다. 여기서 다시 하위목표(subgoal) E가 참이기 위해서는 D가 참이어야 한다. 따라서 D가 부분 최적해를 이룰 첫 번째 대상이 된다. 다음으로 F가 참이기 위해서는 B가 참이어야 하므로 B가 두 번째 대상이 되고, 마찬가지로 A가 세 번째 대상이 된다. 따라서 세 공간 X_i들은 D, B, A의 순서대로 정렬된다. 다음으로 변형된 기준함수를 생각해보자. 각 공간 X_i의 원소는 t 또는 f이다. 따라서 첫 번째 공간 D에 대해서는 $\phi_D(x_D)$가 변형된

기준함수가 되며, x_D는 t 또는 f이다. 마찬가지로 두 번째 공간 B에 대해서는 $\phi_B(x_D, x_B)$가 변형된 기준함수가 된다. 역방향추론의 경우 변형된 기준함수는 X_i의 값과 X_i까지 연쇄된 규칙들에 의하여 평가된다. 예를 들어, $\phi_B(f, f)$의 값은 "최적해를 구할 수 없음", 즉 "G는 참이 아님"이 된다. 왜냐하면, D = f이므로 E가 참이 아니고, B = f이므로 F도 참이 아니며, 따라서 r_4에 의해 G가 참이라고 결론을 내릴 수 없기 때문이다. 이와 달리, E 또는 F가 참일 때 G가 참이므로 $\phi_D(t)$와 $\phi_D(f)$의 값은 모두 "최적해를 구할 수 있음"이 되며, 특히 D = t일 때 B와 A의 값에 상관없이 G는 참이 되게 된다. 다시 말하면, 변형된 기준함수는 연쇄된 규칙들을 바탕으로 목표가 참이 될 가능성이 있으면 "최적해를 구할 수 있음"이 결과값이 되고, 아니면 "최적해를 구할 수 없음"이 결과값이 된다. 위 부분 최적해 구성 순서와 변형된 기준함수를 바탕으로 앞 4개의 규칙으로 된 예제에 대하여 백트래킹 방법을 적용하면, G가 참이기 위한 해로(t, -, -)와 (f, t, t)를 얻게 된다. 만약, 1개의 해만 구하는 것이 목적이었다면 첫 번째 해인(t, -, -)을 해로 얻었을 것이다.

2.5 불확실성 하의 추론

지식에는 거의 필연적으로 불확실성이 내포되어 있다. 따라서 수학적 논리와 같이 오직 참이냐 거짓이냐 만을 다루는 결정적(deterministic)인 방법으로는 실제로 발생하는 불확실성을 제대로 나타낼 수 없는 경우가 많다. 불확실성(uncertainty)이란 "판단이나 의사결정에 필요한 적절한 정보의 부족"이라고 할 수 있다[7]. 불확실성의 이유로는 지식의 불완전성(incompleteness), 지식표현의 애매성(vagueness) 또는 모호함(ambiguity), 지식의 오류(incorrectness), 측정 오류(measurement error) 등이 지목된다[7]. 앞 절에서는 모든 지식에 불확실성이 없는 경우를 가정한 추론 방법에 대하여 설명했는데, 이러한 추론을 확실 추론(exact reasoning)이라고 한다. 이와 반대로 불확실성을 바탕으로 한 추론을 불확실 추론(inexact reasoning)이라고 한다.

규칙의 불확실성은 크게 세 가지로 분류된다. 첫째는 개별 규칙(individual rules)의 불확실성, 둘째는 규칙 적용 순서(conflict resolution)의 불확실성, 셋째는 규칙 간의 모순(incompatibility) 등이다[14]. 개별 규칙의 불확실성은 조건절 또는 결론절의 오류(errors; 모호함, 불완전함, 부정확함, 측정오류 등), 값이 올바를 가능성(likelihood of evidence) 등으로부터 야기된다. 규칙 적용 순서의 불확실성은 규칙 간 우선순위의 불명확성으로부터 야기된다. 규칙의 적용 순서가 바뀜에 따라 다른 결론을 도출할 수 있기 때문이다. 이것은 특히 1개의 해만을 찾는 경우 문제가 된다. 규칙 간의 모순은 포함(subsumption), 중복(redundancy), 부재(missing), 상호 모순(contradiction) 등으로부터 야기된다. 포함이란 'A => C'라는 규칙과 'A and B => C'라는 규칙이 동시에 존재하는 경우를 말한다. 중복은 동일한 규칙이 2개 이상 존재하는 경우, 부재는 입력하여야할 규칙을 규칙베이스에 입력하지 않은 경우, 그리고, 상호 모순은 'A => B'라는 규칙과 'A => not B'라는 규칙이 동시에 존재하는 경우를 말한다. 이와 같이 불확실성의 원인은 여러 가지가 있는데, 오류를 제외한 나머지 불확실성들은 불가피한 것으로, 지식 관리자가 규칙 입력 시 미리 처리해 줄 수는 없고, 추론기관이 추론 수행 중 처리해 주어야 하는 문제들이다. 그러나 오류 이외의 불확실성도 원인이 워낙 다양하므로 이러한 모든 불확실성들을 한꺼번에 해결하여 줄 수 있는 방법은 없고 일부 불확실성에 대하여 각각 몇 가지 처리 방법들이 제시되어 있다. 이와 같이 불확실성을 표현하고 처리하는 대표적인 방법에는 확률적 접근법, 확신도(certainty factor)를 이용한 방법, 퍼지추론(fuzzy reasoning) 등이 있다.

2.5.1 확률적 접근법

어떤 규칙의 조건부를 증거(evidence), 결론부를

가설(hypothesis)이라고 해석할 경우, 그 규칙은 "IF <증거> THEN <가설> WITH 확률 P"라는 형태로 나타낼 수 있다. 이는 규칙의 조건부에 있는 증거 E가 참으로 관찰되면 가설 H도 참일 확률은 P라고 할 수 있다는 것이다[7]. 이러한 확률의 개념을 이용한 불확실성의 처리방법으로 대표적인 것이 베이지안 (Bayesian) 확률론이다[12]. 베이스 정리에 따르면, 임의의 증거 E와 상호배타적인 가설 H_1, H_2, …, H_n이 주어졌을 때 i번째 가설 H_i의 조건부확률 $P(H_i \mid E)$는 다음과 같다.

$$P(H_i \mid E) = \frac{P(E \mid H_i)P(H_i)}{\Sigma_{j=1}^{n} P(E \mid H_j)P(H_j)}$$

베이지안 확률론에서는 $P(H_i)$를 사전확률(prior probability), $P(H_i \mid E)$를 사후확률(posterior probability)이라 한다. 사전확률은 증거 E를 관찰하기 전의 가설 H_i가 참일 확률이고, 사후확률은 증거 E를 관찰한 후에 H_i가 참일 확률이다. 따라서 위 식을 이용하면 규칙의 조건부 즉 증거 E가 참임이 알려졌을 때, 규칙의 결론부 즉 가설 H_i를 참이라고 할 수 있는 확률, 다시 말하면 사후확률 $P(H_i \mid E)$를 구할 수 있다.

이것을 유전 탐사 문제를 예로 들어 설명하도록 하겠다[7]. 특정 지역에 유전이 있을 확률을 $P(H_1)$, 유전이 없을 확률을 $P(H_2)$라고 하고 이들의 확률(사전 확률)이 각각 $P(H_1) = 0.6$, $P(H_2) = 0.4$라고 하자. 또한 유전발견을 위하여 폭발음에 의한 지진탐사가 이루어졌다고 하자. 그런데, 지진탐사는 100% 정확한 것이 아니고, 다만 과거 여러 번의 지진탐사 결과에 의하여 다음과 같은 조건부확률만 알려졌다고 하자.

$$P(E \mid H_1) = 0.8$$
$$P(E \mid H_2) = 0.1$$

여기서 E는 "지진탐사 결과 유전이 있다고 반응을 보인" 사건(event)을 나타낸다. 그러면, $P(E \mid H_1)$는 과거 유전이 있는 지역에서 지진탐사를 한 결과 유전이 있다고 반응을 보인 경우의 수를 확률로 표현한 것으로 앞으로도 그러할 것이라는 조건부확률이며, $P(E \mid H_2)$는 과거 유전이 없는 지역에서 지진탐사를 한 결과 유전이 있다고 반응을 보인 경우의 수로부터 구한 조건부확률이다. 이때, $P(H_1 \mid E)$는 다음과 같이 계산된다.

$$P(H_1|E) = \frac{P(E|H_1)P(H_1)}{P(E|H_1)P(H_1)+P(E|H_2)P(H_2)}$$
$$= \frac{0.8*0.6}{0.8*0.6+0.1*0.4}$$
$$= 0.923$$

다시 말하면 "IF '지진탐사 결과 유전이 있다고 반응을 보였다' THEN '유전이 있다'"라는 규칙의 확률 P가 0.923이라는 뜻으로, 즉, 지진탐사 결과 유전이 있다는 반응이 나왔을 때 유전이 있다는 가설이 참일 확률(즉, 실제로 유전이 있을 확률)은 0.923이라는 말이다.

이와 같이 베이지안 접근법은 이론적으로는 잘 정리되어 있으나 실제로 적용할 경우 사전확률, 조건부확률들을 모두 미리 알고 있어야 하고, 각 증거들이 조건부 독립이 아닌 경우에는 각 증거들 사이의 조건부 확률도 알아야 한다. 따라서 이 접근법은 확률적 추정이 용이하거나 복잡하지 않은 영역의 문제해결에서만 사용할 수 있다[7, 14].

2.5.2 확신도

확신도(certainty factor)는 확률적 접근법의 문제를 해결하기 위한 대안으로서 MYCIN이라는 규칙기반 시스템을 개발하는 과정에서 Shortliffe and Buchanan[22, 23]이 고안한 방법이다[7, 10, 13]. 확신도는 확신의 정도(degree of confirmation)를 나타내며, 다음과 같이 확신(belief)과 불확신(disbelief)의 차로써 표현한다.

$$CF(H, E) = MB(H, E) - MD(H, E)$$

여기서, CF는 증거 E가 주어졌을 때 가설 H에 대한 확신도, MB는 E로 인해 H에 대해 증가된 확신의 정도(measure of increased belief), MD는 E로 인해 H에 대해 증가된 불확신의 정도(measure of increased disbelief)를 말한다. 여기서 MB와 MD는 다음 공식에 의하여 구할 수 있다.

$$MB(H, E) = \begin{cases} 1, \text{ if } P(H) = 1 \\ \dfrac{\max[P(H|E), P(H)] - P(H)}{\max[1, 0] - P(H)}, \text{ o/w} \end{cases}$$

$$MD(H, E) = \begin{cases} 1, \text{ if } P(H) = 0 \\ \dfrac{\min[P(H|E), P(H)] - P(H)}{\min[1, 0] - P(H)}, \text{ o/w} \end{cases}$$

확신도에서 중요한 것은 개별 증거들에 대한 확신도로부터 결합된 확신도를 구하는 것이다. 예를 들어, 두 개의 규칙이 동일한 가설을 결론으로 갖고 있다고 하자. 이 두 규칙의 확신도를 각각 CF_1과 CF_2라고 하면 같은 가설을 결론으로 하는 두 규칙의 결합된 확신도는 다음과 같다.

$$\text{결합}CF(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) \\ \quad CF_1, CF_2 \text{ 모두 } > 0 \\ \dfrac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)} \\ \quad \text{두 } CF_1, CF_2\text{의 기호가 다를 때} \\ CF_1 + CF_2(1 + CF_1) \\ \quad CF_1, CF_2 \text{ 모두 } < 0 \end{cases}$$

예를 들어 다음과 같은 두 개의 규칙이 있다고 하자.

(r_1) IF 우물에서 기름성분이 발견되었다
THEN 유전이 있다 WITH CF = 0.21

(r_2) IF 지진탐사 결과 유전 반응이 있다
THEN 유전이 있다 WITH CF = 0.5

만약 우물에서 기름성분이 발견되었고, 지진탐사 결과 유전 반응이 있었다면, 유전이 있을 결합 확신도는 다음과 같이 구해진다.

$$\text{결합}CF(0.21, 0.5) = 0.21 + 0.5(1 - 0.21) = 0.605$$

세 개 이상의 확신도를 결합할 때는 먼저 두 개를 결합한 후, 그 결과와 세 번째 확신도를 결합하면 된다. 예를 들어 다음과 같은 규칙이 하나 더 있다고 하자.

(r_3) IF 시험시추 결과 기름성분이 발견되지 않았다
THEN 유전이 있다 WITH CF = -0.4

만약 우물에서 기름성분이 발견되었고, 지진탐사 결과 유전 반응이 있었고, 시험시추 결과 기름성분이 발견되지 않았으면, 유전이 있을 결합 확신도는 다음과 같이 구해진다.

$$\text{결합}CF(0.605, -0.4) = \frac{(0.605 - 0.4)}{(1 - \min(|0.605|, |-0.4|))} = 0.205/(1 - 0.4) = 0.34$$

확신도는 계산이 간단하고 쉽게 접근할 수 있어 여러 규칙기반 시스템에서 채택된 바 있다. 그렇지만, 어떤 경우에는 사후확률이 높은 쪽이 오히려 확신도는 낮아짐으로써 확신도가 사후확률과는 상치되는 현상을 야기할 수 있으며, 확신도를 성공적으로 처음 사용한 MYCYN의 경우 짧은 규칙 연쇄와 단순한 가설로만 이루어진 규칙베이스를 사용했기 때문에 규칙 연쇄가 길고 복잡한 가설이 필요한 분야(domain)에서는 확신도가 잘 작동하지

않을 수도 있다[7, 14].

2.5.3 퍼지추론

퍼지추론은 퍼지논리를 이용하여 몇 개의 퍼지 명제로부터 하나의 다른 근사적인 퍼지 명제를 유도하는 근사 추론(approximate reasoning)이다. 퍼지논리에서 명제는 진리값이 0(거짓) 또는 1(참)이 아닌 0과 1 사이의 값을 갖는다[7]. 고전논리에 바탕을 둔 긍정식에서는 'IF A THEN B'라는 규칙이 있으면 이를 수행하기 위하여 반드시 이 규칙의 조건부에 있는 명제 A와 완전히 동일한 명제가 사실베이스에 존재하여야 하며 또한 A나 B의 진리값이 0(거짓) 또는 1(참)이 되어야 한다. 이에 반하여 Zadeh[30]가 제시한 일반화된 긍정식(generalized modus ponens)에 의하면 전제, 조건부, 결론부 등이 완전 일치가 안 되어도 추론이 가능하다. 예를 들어 다음과 같은 퍼지추론을 살펴보자[7, 18, 24, 31].

규칙 : 키가 큰 사람은 몸무게가 무겁다
사실 : 철수가 키가 약간 크다

새로운 사실 : 철수는 몸무게가 약간 무겁다

여기서는 "키가 크다"라는 퍼지 술어를 다루고 있는데, 규칙의 조건부와 완전히 동일하지 않은 사실이 주어지더라도 자연스럽게 추론이 가능하다. 이를 설명하기 위하여 다음과 같이 위 예제의 전제, 조건부, 결론부 문장을 정형화 하자.

"키가 큰 사람은 몸무게가 무겁다"
--> IF X is F then Y is G
where X = 키(사람)
F = 크다
Y = 몸무게(사람)
G = 무겁다

"철수는 키가 약간 크다"
--> X is F^*
where X = 키(철수)
F^* = 약간 크다

"철수는 몸무게가 약간 무겁다"
--> Y is G^*
where Y = 몸무게(철수)
G^* = 약간 무겁다

정형화된 문장들을 이용하여 퍼지추론 문장을 다시 쓰면 다음과 같이 된다.

규칙 : IF X is F then Y is G
사실 : X is F^*

새로운 사실 : Y is G^*

여기서 X가 퍼지집합 F에 속하는 정도인 소속함수 $\mu_F(x)$를 X가 퍼지집합 F를 만족하는 가능성의 정도를 나타내는 가능성 분포(possibility distribution) Π_X로 이용한다. 즉,

$$\Pi_X(u) = \text{Poss}\{X = u\} = \mu_F(u)$$

이다. 예를 들어, 키가 180cm인 사람이 "크다"라는 퍼지집합에 속할 소속함수 값이 1이라면, 이것은 어떤 사람의 키 X가 180cm 일 때 그 사람을 "크다"라고 할 가능성이 1이라는 것이다. 이제 우리의 예제에서 F와 G의 소속함수로부터 얻어진 (사실상은 같은) F와 G의 가능성 분포가 다음과 같다고 하자.

$\Pi_X(u) = \mu_F(u)$
= {0.1/150cm, 0.3/160cm, 0.7/170cm,
0.9/180cm, 1.0/190cm}
$\Pi_Y(v) = \mu_G(v)$
= {0.1/40kg, 0.3/50kg, 0.7/60kg,
0.9/70kg, 1.0/80kg}

또한 "IF X is F THEN Y is G"라는 규칙의 조건부 가능성분포 $\Pi_{Y|X}(u, v)$를 구해보자. 퍼지추론에서 조건부 가능성분포를 구하는 공식은 다음과 같다.

$$\Pi_{Y|X}(u, v) = 1 \wedge (1-\mu_F(u)+\mu_G(v))$$

위 공식에 의하여 구해진 조건부 가능성분포는 <표 1>과 같다.

여기서 F^*(키가 약간 크다)의 가능성분포는 퍼지연산자 중 팽창(dilation) 연산자를 적용하여 구한다. 팽창연산자는 소속함수의 값들에 대하여 제곱근을 취하는 것이므로 F의 가능성분포로부터 다음과 같이 산출된다.

F^* : $\mu_{F^*}(u)$ = {0.316/150cm, 0.547/160cm, 0.836/170cm, 0.948/180cm, 1.0/190cm}

다음으로 $\mu_{G^*}(v) = \sup_u(\mu_{F^*}(u) \wedge (1-\mu_F(u)+\mu_G(v)))$를 구하기 위하여 먼저 $\mu_{F^*}(u) \wedge (1-\mu_F(u)+\mu_G(v))$를 구한다. 퍼지연산에서 $\mu(A \wedge B) = \min(\mu(A), \mu(B))$이므로, <표 1>과 $\mu_{F^*}(u)$에 대하여 $\wedge$ 연산을 수행함으로써 <표 2>를 얻게 된다.

<표 2>에서 $\sup_u (\mu_{F^*}(u) \wedge (1-\mu_F(u)+\mu_G(v)))$를 구하면 다음과 같다.

G^* : $\mu_{G^*}(v)$ = {0.547/40kg, 0.6/50kg, 0.836/60kg, 0.948/70kg, 1.0/80kg}

이제 이렇게 얻어진 가능성분포, 즉 소속함수를 자연어로 바꾸어 보자. "몸무게가 무겁다"의 소속함수는 앞에서 주어진 것처럼 $\mu_G(v)$ = {0.1/40kg, 0.3/50kg, 0.7/60kg, 0.9/70kg, 1.0/80kg}이다. 이 소속함수와 위의 G^*의 소속함수를 비교하여 보면 같은 몸무게일 때 G^*에 소속될 가능성이 더 높으므로 G^*에 대해 적절한 자연어는 "몸무게가 약간 무겁다"가 될 것이다. 요약하면, "키가 큰 사람은 몸무게가 무겁다"라는 퍼지규칙과 "철수가 키가 약간 크다"라는 퍼지명제로부터 일반화된 긍정식에 의하여 "철수는 몸무게가 약간 무겁다"라는 퍼지명제

〈표 1〉 조건부 가능성분포 예제

F \ G	(40kg) 0.1	(50kg) 0.3	(60kg) 0.7	(70kg) 0.9	(80kg) 1.0
(150cm) 0.1	1.0	1.0	1.0	1.0	1.0
(160cm) 0.3	0.8	1.0	1.0	1.0	1.0
(170cm) 0.7	0.4	0.6	1.0	1.0	1.0
(180cm) 0.9	0.2	0.4	0.8	1.0	1.0
(190cm) 1.0	0.1	0.3	0.7	0.9	1.0

〈표 2〉 $\mu_{F^*}(u) \wedge (1-\mu_F(u)+\mu_G(v))$ 계산 예제

F \ G	(40kg) 0.1	(50kg) 0.3	(60kg) 0.7	(70kg) 0.9	(80kg) 1.0
(150cm) 0.1	0.316	0.316	0.316	0.316	0.316
(160cm) 0.3	0.547	0.547	0.547	0.547	0.547
(170cm) 0.7	0.4	0.6	0.836	0.836	0.836
(180cm) 0.9	0.2	0.4	0.8	0.948	0.948
(190cm) 1.0	0.1	0.3	0.7	0.9	1.0

를 도출해 낼 수 있었다.

3. 불확정성 하의 추론

본 논문에서는 불확실성의 또 다른 형태인 '모름(unknown)'을 역방향추론에서 처리하는 방법론을 제시하고자 한다. 백트래킹 방법을 써서 부분최적해를 만들어 가면서 변형된 기준함수 ϕ_n의 값을 구하고자 할 때 추론기관은 먼저 X_i의 값을 알아야 한다. X_i의 값은 사실베이스에서 얻어올 때도 있지만 주로 사용자와의 인터페이스를 통하여 얻어진다. 즉, 추론기관이 문장(즉, 명제) X_i가 참인지 거짓인지를 사용자에게 질문하면 사용자는 그것에 대하여 답하게 된다. 그런데, 이때 사용자가 해당 명제가 참인지 거짓인지 모르는 경우 문제가 발생하게 된다. 이것은 지식관리자에 의한 사전해결이 불가능한 불확실성의 또 다른 형태로, 본 논문에서는 불확정성이라고 부르도록 하겠다. 다시 이야기하면, 본 논문의 목적은 불확정성 하에서의 역방향추론 방법론을 고안, 제시하는 것이다. 이를 위해, 먼저 확실 추론(exact reasoning)에서 백트래킹 방법을 사용하는 역방향추론 알고리즘을 설명한 후, 이것을 확장하여 불확정성 하에서 백트래킹 방법을 사용하는 역방향추론 알고리즘을 제시하도록 하겠다.

3.1 확실 추론에서의 역방향추론

확실 추론 하에서 백트래킹 방법을 규칙베이스와 사실베이스가 존재하는 역방향추론기관이라는 상황에 맞추어 알고리즘으로 표현하면 다음과 같다. 알고리즘을 단순화하기 위하여 두 개의 모듈 Main과 Evaluate를 이용하여 표현했으며, 특히 Evaluate는 자기 자신을 다시 부르는(recurse) 점에 유의하기 바란다.

3.1.1 모듈 Main

Step 1 : g = "목표(goal)"로 놓는다.

Step 2 : Evaluate(g) 서브모듈을 부른다. Evaluate의 결과값(return value)이 참이면 목표 g는 참이고, 결과값이 거짓이면 목표 g는 거짓이다.

Step 3 : 사용자에게 목표의 달성 여부(참, 거짓)를 보고한다.

3.1.2 서브모듈 Evaluate(g)

- 설명 : g는 단위명제들이 AND 또는 OR로 연결된 복합명제이다. 따라서 g의 단위명제별로 다시 Evaluate한 후 AND 또는 OR의 진리표에 따라 g의 참, 거짓을 구하여 리턴(return)하여야 한다.

Step 1 : G = "g의 단위명제들의 집합"으로 놓는다. T = ∅으로 놓는다.

Step 2 : G의 각 원소 p들에 대하여 다음 과정을 반복한다.

Step 2.1 : R = "p를 결론절에 갖는 규칙들의 집합"으로 놓는다.

Step 2.2 : R = ∅이면 p의 참, 거짓 여부를 사용자에게 묻는다. 그 결과를 집합 T에 추가한다.

Step 2.3 : R ≠ ∅이면 R의 각 원소(= 규칙)의 조건절 c_j에 대하여 Evaluate(c_j) 서브모듈을 부른다. (j = 1, …, n; n = R 원소의 개수) 재귀(recursion)를 함에 유의하기 바란다. Evaluate(c_j)의 결과 중 하나라도 그 결과값이 참이면, 참을 집합 T에 추가한다. 모든 결과가 거짓이면 거짓을 집합 T에 추가한다.

Step 3 : T는 g의 단위명제들의 참, 거짓 결과의 집합이다. g의 단위명제들은 AND 또는 OR로 연결되어 있으므로, <표 3>에 나타난 AND 또는 OR 진리표에 따라 g의 참, 거짓을 구하여 리턴한다. 예를 들어, g = A AND B이고, T = {참, 거짓}, 즉 단위 명제 A의 값은 참, 단위명제 B의

값은 거짓이라면, g의 값은 거짓이 된다. g의 단위명제의 개수가 1개라면, 단위명제의 결과값이 g의 결과값이 되며, g의 단위명제의 개수가 3개 이상이라면 AND 또는 OR 연산을 반복하여 g의 결과값을 얻는다.

〈표 3〉 AND 및 OR의 진리표

a	b	a AND b	a OR b
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

주) T = 참, F = 거짓.

서브모듈 Evaluate(g)을 이해하는 데 있어 유의하여야 할 사항이 하나 있다. Step 2.2는 Evaluate(c_j)의 결과 중 하나라도 그 결과값이 참이면, 참을 집합 T에 추가하고, 모든 결과가 거짓이면 거짓을 집합 T에 추가하고 있다. 이것은 일반적인 논리학 측면에서만 이해하면 잘못된 말이다. 예를 들어 다음 예제를 보자.

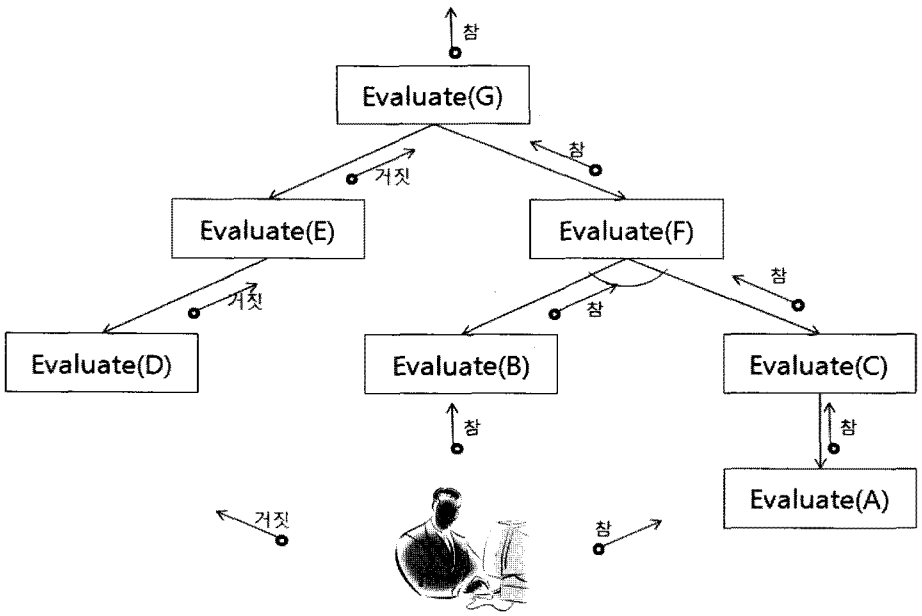
a => p
b => p

일반적으로 x => y라는 논리식에서 x가 참이면 y도 참이지만, x가 거짓일 때는 y값을 결정할 수 없다. 따라서 위 예제에서 a 또는 b가 참이라고 하면 p는 참이라고 할 수 있으나, a와 b 모두가 거짓일 경우 p의 값은 결정할 수 없게 된다. 그러나 역방향추론의 목적은 목표값이 참인지 아닌지를 판별하는 것이고, 참이 아닌 것에는 "거짓"과 "결정이 안 된 것"도 포함되는 것으로 생각하고, 그것을 단순히 "거짓"이라고 표현하기로 하면, 위 알고리즘과 진리표는 수정 없이 사용할 수 있다.

이제 지금까지 설명한 알고리즘의 이해를 돕기 위하여 다음과 같은 4개의 예제 규칙에 대하여 위 알고리즘을 적용해 보자.

(r_1) A => C
(r_2) D => E
(r_3) B and C => F
(r_4) E or F => G

목표는 G라고 하자. 먼저 Evaluate(G)를 부른다. Evaluate(G) 서브모듈은 G를 결론절에 갖는 규칙들을 찾는다. 이것은 {r_4}이다. 다음으로 r_4의 조건절을 이용하여 Evaluate(E)와 Evaluate(F)를 다시 부른다. 먼저 Evaluate(E)는 E를 결론절에 갖는 규칙을 찾는다. 이것은 {r_2}이다. 이제 r_2의 조건절을 이용하여 Evaluate(D)를 부른다. Evaluate(D)는 D의 참, 거짓 여부를 판별하기 위하여 D를 결론절에 갖는 규칙을 찾는다. 그러한 규칙이 없으므로 D의 참, 거짓 여부를 사용자에게 묻는다. 사용자가 참이라고 대답하면 Evaluate(D)는 참을, 거짓이라고 대답하면 거짓을 리턴한다. 본 예제에서는 사용자가 거짓이라고 답하였다고 하자. 그러면 Evaluate(D)는 거짓을 리턴하고, 따라서 Evaluate(E)에서의 T = {거짓}이 된다. T로부터 Evaluate(E)는 거짓(앞의 유의사항 설명에 따라 실제로는 "결정이 안 된 것")을 리턴한다. 다음으로는 Evaluate(F)는 F를 결론절에 갖는 규칙을 찾는다. 이것은 {r_3}이다. 이제 r_3의 조건절을 이용하여 Evaluate(B)와 Evaluate(C)를 부른다. 먼저 Evaluate(B)는 B의 참, 거짓 여부를 판별하기 위하여 B를 결론절에 갖는 규칙을 찾는다. 그러한 규칙이 없으므로 B의 참, 거짓 여부를 사용자에게 묻는다. 본 예제에서는 사용자가 참이라고 답하였다고 하자. 그러면 Evaluate(B)는 참을 리턴하고, 따라서 Evaluate(F)에서 현재의 T = {참}이 된다. 한편, Evaluate(C)는 C를 결론절에 갖는 규칙을 찾는다. 이것은 {r_1}이다. 이제 r_1의 조건절을 이용하여 Evaluate(A)를 부른다. Evaluate(A)는 A의 참, 거짓 여부를 판별하기 위하여 A를 결론절에 갖는 규칙을 찾는다. 그러한 규칙이 없으므로 A의 참, 거짓 여부를 사



[그림 1] 확실 추론에서의 역방향추론 알고리즘 적용 예제

용자에게 묻는다. 본 예제에서는 사용자가 참이라고 답하였다고 하자. 그러면 Evaluate(A)는 참을 리턴하고, 따라서 Evaluate(C)에서의 T = {참}이 된다. T로부터 Evaluate(C)는 참을 리턴한다. 따라서 이제 Evaluate(F)에서의 T = {참, 참}이 된다. T로부터, 즉 "참 AND 참"으로부터, Evaluate(F)는 참을 리턴한다. Evaluate(E)의 값이 거짓, Evaluate(F)의 값이 참이므로 Evaluate(G)의 T = {거짓, 참}이 된다. T로부터, 즉 "거짓 OR 참"으로부터, Evaluate(G)는 참을 리턴하고, Main 모듈을 사용자에게 목표 G가 참임을 보고한다. 지금까지 설명한 것을 그림으로 표현하면 [그림 1]과 같다.

3.2 불확정성 하에서의 역방향추론

사용자가 해당 명제가 참인지 거짓인지 모르는 경우, 즉 불확정성을 처리하기 위해서는 우선 명제의 값으로 "참"과 "거짓" 외에 "모름"을 추가로 다룰 수 있어야 한다. 즉, 사용자가 어떤 명제에 대하여 모르겠다고 답하였을 경우, 그 명제의 값은 "모름"이 되고, 이것을 AND 및 OR 연산의 피연산자로서 다룰 수 있어야 한다. 모름을 포함한 AND, OR 연산을 위한 진리표가 <표 4>에 나타나 있다.

〈표 4〉 "모름"을 고려한 AND 및 OR의 진리표

a	b	a AND b	a OR b
T	T	T	T
T	F	F	T
T	U	U	T
F	T	F	T
F	F	F	F
F	U	F	U
U	T	U	T
U	F	F	U
U	U	U	U

주) T = 참, F = 거짓, U = 모름.

3.1절 서브모듈 Evaluate(g)에서 유의사항으로 설명한 문제, 즉 논리식 x => y에서 x가 거짓인 경우의 처리문제도 불확정성 하의 추론에서는 명확히 하여야 한다. 여기서 연산자 "=>"는 x와 y 값

이 주어져 있을 때 수식 "x => y"의 값을 계산해 주는 이항연산자 (binary operator)가 아니라, x의 값이 주어져 있을 때 y의 값을 계산해 주는 단항 연산자 (unary operator)이다. 단항연산자 "=>"의 진리표는 <표 5>에 나타나 있다.

〈표 5〉 단항연산자 "=〉"의 진리표

a	a =>
T	T
F	U
U	U

주) T = 참, F = 거짓, U = 모름.

불확정성 하에서 백트래킹 방법을 규칙베이스와 사실베이스가 존재하는 역방향추론기관이라는 상황에 맞추어 알고리즘으로 표현하면 다음과 같다. 이 알고리즘은 앞 절의 확실 추론 하에서의 알고리즘을 확장한 것으로 앞의 것과 유사하게 두 개의 모듈 Main과 Evaluate를 이용하여 표현하였다.

3.2.1 모듈 Main

Step 1 : g = "목표(goal)"로 놓는다.

Step 2 : Evaluate(g) 서브모듈을 부른다. Evaluate의 결과값(return value)이 참이면 목표 g는 참이고, 결과값이 거짓이면 목표 g는 거짓, 결과값이 모름이면 목표 g는 모름이다.

Step 3 : 사용자에게 목표의 달성 여부(참, 거짓 또는 모름)를 보고한다.

3.2.2 서브모듈 Evaluate(g)

- 설명 : g는 단위명제들이 AND 또는 OR로 연결된 복합명제이다. 따라서 g의 단위명제별로 다시 Evaluate한 후 AND 또는 OR의 진리표에 따라 g의 참, 거짓, 모름을 구하여 리턴(return)하여야 한다.

Step 1 : G = "g의 단위명제들의 집합"으로 놓는다. T = ∅으로 놓는다.

Step 2 : G의 각 원소 p들에 대하여 다음 과정을 반복한다.

Step 2.1 : R = "p를 결론절에 갖는 규칙들의 집합"으로 놓는다.

Step 2.2 : R = ∅이면 p의 참, 거짓 또는 모름 여부를 사용자에게 묻는다. 그 결과를 집합 T에 추가한다.

Step 2.3 : R ≠ ∅이면 R의 각 원소(= 규칙)의 조건절 c_j에 대하여 Evaluate(c_j) 서브모듈을 부른다(j = 1, …, n; n = R 원소의 개수). 재귀(recursion)를 함에 유의하기 바란다. Evaluate(c_j)의 결과 중 하나라도 그 결과값이 참이면, 참을 집합 T에 추가한다. 모든 결과가 거짓 또는 모름이면 모름을 집합 T에 추가한다. 이것은 <표 5>의 진리표에 의한 것이다.

Step 3 : T는 g의 단위명제들의 참, 거짓 또는 모름 결과의 집합이다. g의 단위명제들은 AND 또는 OR로 연결되어 있으므로, <표 4>에 나타난 AND 또는 OR 진리표에 따라 g의 참, 거짓 또는 모름을 구하여 리턴한다. 예를 들어, g = A AND B이고, T = {모름, 거짓}, 즉 단위 명제 A의 값은 모름, 단위명제 B의 값은 거짓이라면 g의 값은 거짓이 된다. g의 단위명제 개수가 1개라면, 단위명제의 결과값이 g의 결과값이 되며, g의 단위명제 개수가 3개 이상이라면 AND 또는 OR 연산을 반복하여 g의 결과값을 얻는다.

위 알고리즘에서 모듈 Main이 "거짓"을 보고하는 경우는 목표 g를 결론절에 갖는 규칙이 하나도 없어서 사용자에게 직접 g의 값을 질문하여 거짓이라는 답을 얻었을 경우뿐이다. 만약 목표 g를 결론절에 갖는 규칙이 있다면, <표 5>의 진리표에 의하여 거짓이라는 결과는 결코 얻을 수가 없기

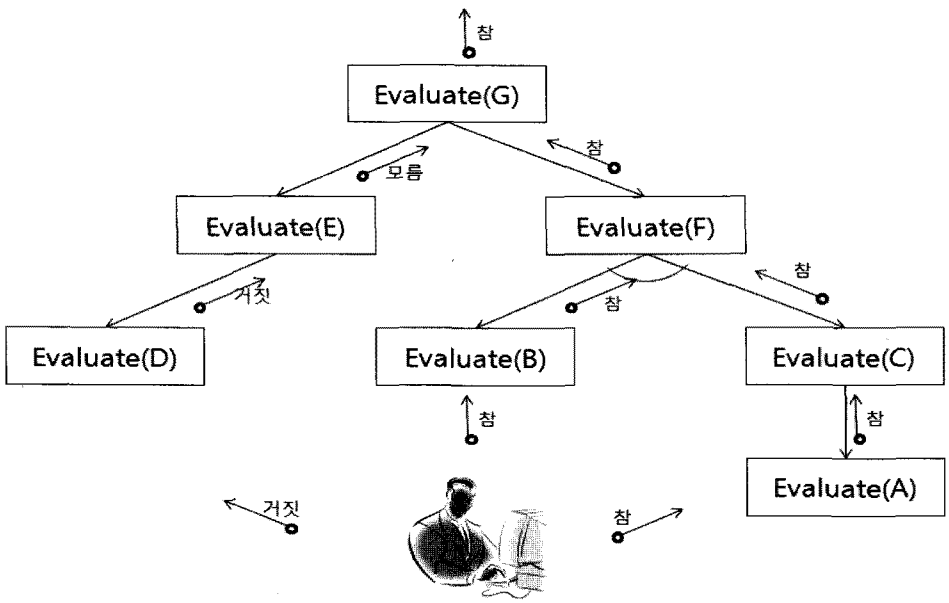
때문이다.

이제 지금까지 설명한 알고리즘의 이해를 돕기 위하여 앞 절에서와 같은 4개의 예제 규칙에 대하여 위 알고리즘을 적용해 보자.

(r_1) A => C
(r_2) D => E
(r_3) B and C => F
(r_4) E or F => G

목표는 G라고 하자. 먼저 Evaluate(G)를 부른다. Evaluate(G) 서브모듈은 G를 결론절에 갖는 규칙들을 찾는다. 이것은 {r_4}이다. 다음으로 r_4의 조건절을 이용하여 Evaluate(E)와 Evaluate(F)를 다시 부른다. 먼저 Evaluate(E)는 E를 결론절에 갖는 규칙을 찾는다. 이것은 {r_2}이다. 이제 r_2의 조건절을 이용하여 Evaluate(D)를 부른다. Evaluate(D)는 D의 참, 거짓, 모름 여부를 판별하기 위하여 D를 결론절에 갖는 규칙을 찾는다. 그러한 규칙이 없으므로 D의 참, 거짓, 모름 여부를 사용자에게 묻는다. 사용자가 참이라고 대답하면 Evaluate(D)는 참을, 거짓이라고 대답하면 거짓을, 모름이라고 대답하면 모름을 리턴한다. 본 예제에서는 사용자가 거짓이라고 답하였다고 하자. 그러면 Evaluate(D)는 거짓을 리턴하고, 알고리즘 Step 2.3에 의하여 Evaluate(E)에서의 T = {모름}이 된다. T로부터 Evaluate(E)는 모름을 리턴한다.

다음으로는 Evaluate(F)는 F를 결론절에 갖는 규칙을 찾는다. 이것은 {r_3}이다. 이제 r_3의 조건절을 이용하여 Evaluate(B)와 Evaluate(C)를 부른다. 먼저 Evaluate(B)는 B의 참, 거짓, 모름 여부를 판별하기 위하여 B를 결론절에 갖는 규칙을 찾는다. 그러한 규칙이 없으므로 B의 참, 거짓, 모름 여부를 사용자에게 묻는다. 본 예제에서는 사용자가 참이라고 답하였다고 하자. 그러면 Evaluate(B)는 참을 리턴하고, 따라서 Evaluate(F)에서 현재의 T = {참}이 된다. 한편, Evaluate(C)는 C를 결론절에 갖는 규칙을 찾는다. 이것은 {r_1}이다. 이제 r_1의 조건절을 이용하여 Evaluate(A)를 부른다. Evaluate A)는 A의 참, 거짓, 모름 여부를 판별하기 위하여 A를 결론절에 갖는 규칙을 찾는다. 그러한 규칙이 없으므로 A의 참, 거짓, 모름 여부를 사용자에게



[그림 2] 불확정성 하 추론에서의 역방향추론 알고리즘 적용 예제

묻는다. 본 예제에서는 사용자가 참이라고 답하였다고 하자. 그러면 Evaluate(A)는 참을 리턴하고, 따라서 Evaluate(C)에서의 T = {참}이 된다. T로부터 Evaluate(C)는 참을 리턴한다. 따라서 이제 Evaluate(F)에서의 T = {참, 참}이 된다. T로부터, 즉 '참 AND 참'으로부터, Evaluate(F)는 참을 리턴한다.

Evaluate(E)의 값이 모름, Evaluate(F)의 값이 참이므로 Evaluate(G)의 T = {모름, 참}이 된다. T로부터, 즉 '모름 OR 참'으로부터, Evaluate(G)는 참을 리턴하고, Main 모듈을 사용자에가 목표 G가 참임을 보고한다. 지금까지 설명한 것을 그림으로 표현하면 [그림 2]와 같다.

3.3 탐색 공간을 고려한 확장

앞 절에서는 확실 추론에서의 역방향추론 알고리즘을 제시한 후, 이를 확장하여 불확정성 하에서의 역방향추론 알고리즘을 고안하였다. 이 알고리즘들은 규칙 연쇄를 이용하여 부분 벡터해를 만들어나간다는 점에서는 백트래킹 방법을 그대로 따르고 있다. 그러나 모든 명제의 참, 거짓, 또는 모름을 판별하도록 되어 있기 때문에, 변형된 기준함수 이용 시 탐색 공간을 줄임으로써 탐색 시간을 줄이고자 하는 백트래킹의 고유의 장점은 살리고 있지 못하고 있다. 제안한 역방향추론 알고리즘에 탐색공간을 줄이는 기능을 추가하기 위하여 프로그래밍 언어 분야에서 Short-circuit evaluation (또는 minimal evaluation, McCarthy evaluation)이라고 알려진 연산을 도입하고자 한다. Short-circuit evaluation은 첫 번째 피연산자의 값이 전체 수식의 결과값을 결정하는데 충분하다면 두 번째 이하의 피연산자의 값은 계산하지 않는 것을 말한다. 예를 들어 'a AND b'라는 수식을 계산할 때 a의 값이 거짓이라면 b의 값과 상관없이 전체 수식 'a AND b'의 값은 거짓이 된다. 우리의 역방향추론 알고리즘에서 Step 2.3 및 Step 3에서 위 연산을 적용할 수 있다. 먼저, Step 2.3의 경우 R의 각 원소(= 규칙)의 조건절 c_j에 대하여 Evaluate(c_j) 중의 하나만이라도 참이면 그 결과는 참이므로, 모든 조건절 c_j에 대하여 Evaluate(c_j)를 할 필요 없이 참이 나오는 조건절 c_j를 만날 때까지만 Evaluate(c_j)를 하면 된다. 마찬가지로 Step 3에서 g를 계산할 때 T를 완성한 후 계산할 것이 아니라, g의 값이 결정될 수 있을 때까지만 T를 생성한 후 g의 값을 계산하면 된다. 이것을 우리의 알고리즘에 적용하여 확장하면 다음과 같이 된다.

3.3.1 모듈 Main

Step 1 : g = "목표(goal)"로 놓는다.

Step 2 : Evaluate(g) 서브모듈을 부른다. Evaluate의 결과값(return value)이 참이면 목표 g는 참이고, 결과값이 거짓이면 목표 g는 거짓, 결과값이 모름이면 목표 g는 모름이다.

Step 3 : 사용자에게 목표의 달성 여부(참, 거짓 또는 모름)를 보고한다.

3.3.2 서브모듈 Evaluate(g)

- 설명 : g는 단위명제들이 AND 또는 OR로 연결된 복합명제이다. 따라서, g의 단위명제별로 다시 Evaluate한 후 AND 또는 OR의 진리표에 따라 g의 참, 거짓, 모름을 구하여 리턴(return)하여야 한다.

Step 1 : G = "g의 단위명제 p_i들의 집합"으로 놓는다(i = 1, …, m; m = G의 원소의 개수).

Step 2 : g를 계산하기 위하여 p_i들을 다음과 같이 차례로 계산한다.

Step 2.1 : R = "p_i를 결론절에 갖는 규칙들의 집합"으로 놓는다.

Step 2.2 : R = ∅이면 p의 참, 거짓 또는 모름 여부를 사용자에게 묻는다. 그 결과를 p_i의 값으로 한다.

Step 2.3 : R ≠ ∅이면 R의 각 원소(= 규칙)의 조건절 c_j에 대하여 Evaluate(c_j) 서브

모듈을 차례로 부른다(j = 1, …, n; n = R 원소의 개수). 재귀(recursion)를 함에 유의하기 바란다. Evaluate(c_j)를 차례로 부르는 중에 그 결과값이 참인 것이 나오면 참을 p_i의 값으로 한다. Evaluate(c_n)까지 모두 그 결과값이 거짓 또는 모름이면 모름을 p_i의 값으로 한다. 이것은 <표 5>의 진리표에 의한 것이다.

Step 3 : p_i가 얻어질 때마다 <표 6>을 이용하여 g의 값을 결정할 수 있는지 본다. <표 6>에서 "'a' only"의 값이 Yes인 경우는 첫 번째 피연산자의 값만으로 연산값을 알 수 있는 경우이고, No이면 알 수 없는 경우이다. g의 값을 결정할 수 없으면 다음 p_i에 대하여 Step 2를 반복한다. g의 값을 결정할 수 있다면 <표 6> 진리표에 따라 g의 참, 거짓 또는 모름을 구하여 리턴한다. 예를 들어, g = A AND B이고, A의 값이 거짓이라면, g의 값은 거짓이 된다. g의 단위명제의 개수가 1개라면, 단위명제의 결과값이 g의 결과값이 되며, g의 단위명제의 개수가 3개 이상이라면 AND 또는 OR 연산을 반복하여 g의 결과값을 얻는다.

4. 토 의

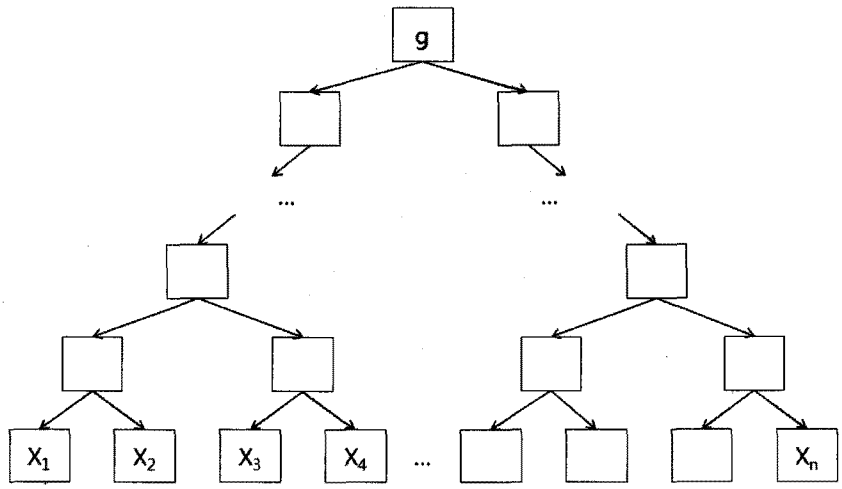
본 논문에서 제시한 확장된 백트래킹 기반 역방향추론 알고리즘의 장점에 대하여 살펴보겠다. 먼저, 이 알고리즘이 탐색 공간을 얼마나 줄일 수 있는지 생각해 보자[19], 일반적으로는 규칙 연쇄의 깊이, AND 또는 OR 연산자의 종류, 조건절에 나타난 단위명제의 개수 등에 따라 탐색 공간의 축소 정도가 달라진다. 그러나 본 절에서는 논의를 단순화하기 위하여 모든 규칙의 조건절에 나타난 단위명제의 개수가 동일하고, 규칙 연쇄에 의하여 나타나는 목표 명제(g)에서부터 각 질문 명제(사용자에게 그 값을 물어보아야 하는 명제)까지의 깊이가 동일하다고 가정하자. 그리고 AND와 OR는 Short-circuit evaluation 관점에서 서로 대칭이므로, 즉 첫 번째 피연산자가 거짓이면 AND 결과는 거짓, 첫 번째 피연산자가 참이면 OR 결과는 참이므로, AND 및 OR의 개수는 탐색 공간을 줄이는 정도와는 상관이 없다. 예를 들어, 각 규칙의 조건절에 나타난 단위명제의 수가 2개인 경우 일반화된 탐색 공간의 모습은 [그림 3]과 같이 된다.

[그림 3]과 같이 일반화된 탐색공간에서 목표명제 g가 있는 뿌리 노드와 질문명제 X_i(i = 1, …, n)들이 있는 말단 노드까지의 깊이(depth)를 d라고 하고, 각 조건절에 나타난 단위명제의 개수가 c

〈표 6〉 Short-circuit evaluation을 고려한 AND 및 OR의 진리표

a	b	a AND b	'a' only	a OR b	'a' only
T	T	T	No	T	Yes
T	F	F	No	T	Yes
T	U	U	No	T	Yes
F	T	F	Yes	T	No
F	F	F	Yes	F	No
F	U	F	Yes	U	No
U	T	U	No	T	No
U	F	F	No	U	No
U	U	U	No	U	No

주) T = 참, F = 거짓, U = 모름.



[그림 3] 일반화된 탐색 공간 예제

라고 하면 전체 노드의 개수는 $\Sigma_{j=0}^{d} c_j$이 된다. 만약 모든 조건절들의 각 단위명제들이 AND로 연결되었을 때 X_1이 거짓이면, 우리의 알고리즘은 d 번의 탐색만으로 g의 값(거짓)을 알 수 있게 된다. 대칭적으로, 만약 모든 조건절들의 각 단위명제들이 OR로 연결되었을 때 X_1이 참이면, 우리의 알고리즘은 d번의 탐색만으로 g의 값(참)을 알 수 있게 된다. 그러나 만약 모든 조건절들의 각 단위명제들이 AND로 연결되었을 때 모든 X_i가 참이면, 우리의 알고리즘은 $\Sigma_{j=0}^{d} c_j$번의 탐색으로 g의 값(참)을 알 수 있게 된다.

평균적으로 X_i 중 반은 참이고, 반은 거짓일 것으로 기대되며, 이러한 경우에 탐색공간이 얼마나 줄어들 수 있는지 알아보자. 이때, 우리의 알고리즘이 처음으로 거짓 값을 갖는 X_i를 만나게 되는 것은 최선의 경우 i = 1일 때, 최악의 경우 i = n/2 일 때 이다(n/2 이후는 모두 거짓 값을 가져야 하기 때문이다). 따라서 최악의 경우 $((\Sigma_{j=0}^{d} c_j)/2)+d$ 번의 탐색으로 g의 값을 알 수 있게 된다. 여기서 $((\Sigma_{j=0}^{d} c_j)/2)$는 전체 탐색 공간(tree)의 왼쪽 반에 있는 노드의 개수이고, d는 나머지 오른쪽 반 중에서 처음으로 거짓 값을 갖는 노드를 만날 때까지의 탐색 횟수이다. 따라서 최선의 경우 d, 최악의 경우$((\Sigma_{j=0}^{d} c_j)/2)+d$번의 탐색이 필요하므로, 평균 탐색횟수는 다음과 같이 된다.

$$\frac{d+\frac{\sum_{j=0}^{d} c^j}{2}+d}{2} = \frac{2d+\frac{\sum_{j=0}^{d} c^j}{2}}{2} = d+\frac{\sum_{j=0}^{d} c^j}{4}$$

다시 말하면, 평균적으로 탐색공간을 약 1/4 정도 줄일 수 있을 것으로 기대된다. 따라서 우리의 알고리즘이 P-시간(P-time, polynomial time) 알고리즘이 되지는 못한다. 그렇지만, 현실에서 우리의 알고리즘을 구현한 추론기관을 사용하게 될 때 다음 두 가지 측면에서 장점을 생각해 볼 수 있다. 첫째, 현실세계에서는 상당히 큰 탐색 공간을 갖는 규칙베이스를 다루게 되고, 이 경우 위와 같이 탐색공간을 줄일 경우 많은 수행시간을 줄일 수 있게 된다. 예를 들어 <표 7>에서 규칙들의 조건절 단위명제 수가 4이고, 규칙 연쇄의 깊이가 10일 경우 탐색 공간의 총 노드 수는 1,398,101이 되지만, 평균 탐색 횟수는 약 349,535로 줄어들게 된다. 특히, 수많은 사용자를 대상으로 웹을 통하여 추론 서비스를 제공할 경우 이것은 컴퓨팅 시간 측면에서뿐만 아니라 웹 서버의 부하 절감 측면에서도 많은 도움이 된다. 둘째, 목표 g의 값이 알려졌을 경우 더 이상 질문명제 X_i의 값을 질문하지 않는

〈표 7〉 단위명제 개수(c) 및 깊이(d) 별 탐색 횟수

깊이(d)	총노드수 (c = 2)	평균 탐색 횟수	총노드수 (c = 3)	평균 탐색 횟수	총노드수 (c = 4)	평균 탐색 횟수
1	3	1.75	4	2.00	5	2.25
2	7	3.75	13	5.25	21	7.25
3	15	6.75	40	13.00	85	24.25
4	31	11.75	121	34.25	341	89.25
5	63	20.75	364	96.00	1,365	346.25
6	127	37.75	1,093	279.25	5,461	1,371.25
7	255	70.75	3,280	827.00	21,845	5,468.25
8	511	135.75	9,841	2,468.25	87,381	21,853.25
9	1,023	264.75	29,524	7,390.00	349,525	87,390.25
10	2,047	521.75	88,573	22,153.25	1,398,101	349,535.25

다는 것은, 사용자에게 불필요한 질문을 하지 않는다는 뜻이 된다. 추론 서비스 중에 이미 결론을 알게 되면 더 이상 다른 질문을 하지 않는다는 뜻은 사용자와의 대화 시간을 줄임으로써 고객 상담 시간을 절감할 수 있을 뿐 아니라, 지루하고 불필요한 질문을 하지 않음으로써 사용자의 만족도를 높일 수 있게 된다. 일반적으로 실제 기업에서는 두 번째 장점이 더 큰 의미를 가질 것으로 생각된다.

5. 결 론

본 논문에서는 불확실성을 고려한 역방향추론 알고리즘을 제시하고자 하였다. 불확실성은 개별 규칙(individual rules)의 불확실성, 규칙 적용 순서(conflict resolution)의 불확실성, 규칙 간의 모순(incompatibility) 등에 기인한다. 지금까지 불확실성들을 다루기 위한 여러 가지 방법들이 개발되었으나 불확실성의 형태가 워낙 다양하여 아직까지 모든 불확실성 문제가 해결된 것은 아닌 상태이다. 본 논문에서는 불확실성 중 불확정성 문제에 초점을 맞추어 이를 해결하는 방법론을 제시하고자 하였다. 불확정성은 사용자가 어떤 사실의 진위여부를 모르는 것을 의미한다. 지금까지 역방향추론에서 사용자가 어떤 사실의 진위여부를 모를 경우 추론기관의 질문에 대답을 할 수 없었고, 그러면 더 이상의 추론이 불가능하였다. 이러한 불확정성 문제를 해결하면 사용자가 일부 사실의 진위여부만 알고 있는 경우에도 추론기관이 최대한 가능한 답을 찾아서 제시해 줄 수 있게 된다.

이를 위하여 본 논문에서는 먼저 추론의 정의, 추론의 종류 등을 살펴보고, 역방향추론의 구현을 위한 방법론으로 백트래킹 기법을 알아보았다. 그리고 지금까지 불확실성을 처리하기 위하여 개발된 방법 중 확률적 접근법, 확신도(certainty factor)를 이용한 방법, 퍼지추론(fuzzy reasoning) 등의 내용을 알아보고 그 장단점을 살펴보았다. 이러한 기존 연구에 대한 이해를 바탕으로 본 연구에서는 먼저 확실 추론 하에서 백트래킹 기법을 이용한 역방향추론 알고리즘을 제시한 후, 이 알고리즘을 불확정성, 즉 모름의 답을 처리할 수 있도록 확장하였다. 그리고 백트래킹 기법에서 탐색 공간을 줄이기 위하여 Short-circuit evaluation 연산 개념을 도입하여 위 알고리즘을 다시 확장하였다. 이러한 과정을 통하여 불확정성을 고려한 백트래킹 기법 기반 역방향추론 알고리즘을 최종 개발하였다. 일반 사용자를 대상으로 한 규칙 기반 자문, 상담, 추천 등 웹 기반 서비스의 필요성이 증대되고 있고 또 수많은 규칙 기반 웹 사이트가 개설되

고 있는 상황에서, 사용자가 완전한 정보를 제공해 주지 못할 때에도 최대한 결론을 이끌어내 줄 수 있고, 또 최소한의 질문을 통하여 웹 사이트 사용 시간을 경감시킬 수 있는 위 방법론은 기업의 지능형 웹 사이트 개발 및 운영에 많은 도움이 될 수 있을 것으로 기대된다.

참 고 문 헌

[1] 송용욱, 김우주, 홍준석, "지식분석도를 이용한 지식기반 웹 사이트 자동 생성 도구의 개발", 「경영정보학연구」, 제13권, 제1호(2003), pp.213-230.

[2] 송용욱, 이재규, "웹 기반 전문가시스템의 자동생성체계", 「한국지능정보시스템학회논문지」, 제6권, 제1호(2000), pp.1-16.

[3] 송용욱, 홍준석, 김우주, 이성규, 윤숙희, "차세대 웹을 위한 SWRL 기반 역방향 추론엔진 SMART-B의 개발", 「한국지능정보시스템학회논문지」, 제12권, 제2호(2006), pp.67-81.

[4] 이광형, 오길록, 「퍼지이론 및 응용(I, II)」, 홍릉과학출판사, 1991.

[5] 이재규, 권순범, 김우주, 김민용, 송용욱, 최형림, 「전자상거래 원론」, 제3판, 법영사, 2002.

[6] 이재규, 송용욱, 권순범, 김우주, 김민용, 「UNIK을 이용한 전문가시스템의 개발」, 법영사, 1996.

[7] 이재규, 최형림, 김현수 편저, 「인터넷 환경의 지식시스템」, 법영사, 2006.

[8] 정균범, 송용욱, 홍준석, 김우주, 이명진, 박지형, "차세대 웹 환경에서의 Rete Algorithm을 이용한 정방향 추론엔진 SMART-F 개발", 「한국지능정보시스템학회논문지」, 제13권, 제3호(2007), pp.17-29.

[9] Song, Y. U., Y. M. Chae, S. H. Ho, and K. W. Cho, "Web-enabled Healthcare System for Hypertension : Hyperlink-based Inference Approach", 「한국지능정보시스템학회논문지」, 제9권, 제1호(2003), pp.91-107.

[10] Adams, J. B., "Probabilistic Reasoning and Certainty Factors", in Buchanan, B. G. and Shortliffe, E. H. (Ed.), *Rule Based Expert Systems*, Addison-Wesley, 1984.

[11] Bitner, J. R. and E. M. Reingold, "Backtrack programming techniques", *Communications of the ACM*, Vol.18, No.11(1975), pp.651-656.

[12] Duda, R. O., P. E. Hart, and N. J. Nilsson, "Subjective bayesian methods for rule-based inference systems", AFIPS Joint Computer Conferences, *Proceedings of the, National computer conference and exposition*, (1976), pp.1075-1082.

[13] Friederich, S. and M. Gargano, *Expert Systems Design and Development Using VP-Expert*, John Wiley and Sons, 1989.

[14] Giarratano, J. C. and G. D. Riley, *Expert Systems : Principles and Programming, 4th Ed.*, Thompson Learning, 2004.

[15] Golomb, S. W. and L. D. Baumert, "Backtrack Programming", *Journal of the ACM*, Vol.12, No.4(1965), pp.516-524.

[16] Harrison, P. R. and J. G. Kovalchik, "Expert Systems and Uncertainty," in J. Liebowitz (Ed.), *The Handbook of Applied Expert Systems*, CRC Press, 1998.

[17] Kim, W., Y. Song, and J. Hong, "Web enabled expert systems using hyperlink-based inference", *Expert Systems with Applications*, Vol.28, No.1(2005), pp.79-91.

[18] Klir, G. J. and T. A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice Hall, 1998.

[19] Knuth, D. E., "Estimating the efficiency of backtrack programs", *Mathematics of Computation*, Vol.29, No.129(1975), pp.121-136.

[20] Quinlan, J. R., "Introduction to Decision Tre-

es", *Machine Learning*, Vol.1, No.1(1986), pp. 81-106.

[21] Quinlan, J. R., *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, 1993.

[22] Shortliffe, E. H. and B. G. Buchanan, "A Method of Inexact Reasoning", *Mathematical Biosciences*, Vol.23(1975), pp.351-379.

[23] Shortliffe, E. H. and B. G. Buchanan, "A Model of Inexact Reasoning in Medicine", in Buchanan, B. G. and Shortliffe, E. H. (Ed.), *Rule Based Expert Systems*, Addison-Wesley, 1984.

[24] Siler, W. and J. J. Buckley, *Fuzzy Expert Systems and Fuzzy Reasoning*, John Wiley and Sons, 2004.

[25] Suwa, M., A. C. Scott, and E. H. Shortliffe, "An approach to verifying completeness and consistency in a rule-based expert system", *AI Magazine*, Vol.3, No.4(1982), pp.16-21.

[26] Turban, E., D. King, J. K. Lee, and T. Liang, *Electronic Commerce 2010 : A Managerial Perspective*, Prentice Hall, 2010.

[27] Turban, E., J. E. Aronson, and T. Liang, *Decision Support Systems and Intelligent Systems*, 7th ed., Prentice Hall, 2005.

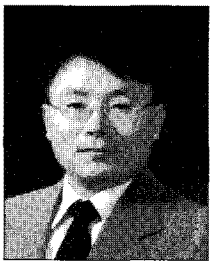
[28] Waterman, D. A., *A Guide to Expert Systems*, Addison-Wesley, 1986.

[29] Witten, I. H. and E. Frank, *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, 2000.

[30] Zadeh, L. A., "Fuzzy Sets", *Information and Control*, Vol.8(1965), pp.338-353.

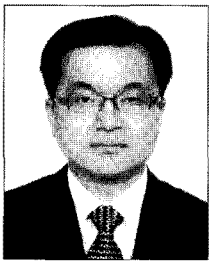
[31] Zimmerman, H. J., *Fuzzy Set Theory and Its Applications*, Kluwer-Nijhoff Publishing, 1985.

◈ 저 자 소 개 ◈



송 용 욱 (yusong@yonsei.ac.kr)

현재 연세대학교 원주캠퍼스 경영학부 부교수로 재직 중이다. 서울대학교 국제경제학과를 졸업하고, 한국과학기술원(KAIST) 경영과학과에서 석사학위를 산업경영학과에서 박사학위를 취득하였다. Management Science, Annals of Operations Research, Expert Systems with Applications 등에 논문을 게재한 바 있다. 주요 관심분야는 정보시스템 개발, 경영분야 문제에의 전문가시스템 응용, 전자상거래, 전문가시스템 및 수리계획법과 전자상거래의 통합 등이다.



신 현 식 (hshin@cj.net)

현재 CJ오쇼핑에 재직 중이며, 연세대학교 경영학과를 졸업하고 한국과학기술원(KAIST) 경영과학과에서 석사학위를, 경영정책학과에서 박사학위를 취득하였다. Journal of Systems and Software, Journal of Information Technology Management, 한국IT서비스학회지, 한국전자거래학회지, 정보시스템연구 등에 논문을 게재한 바 있다. 주요 관심분야는 정보화 전략, 전사적 자원관리(ERP), IT 거버넌스, 전자상거래, 지식경영, 유통 정보화 등이다.