*Journal of Statistical*
*Theory & Methods*
*1995, Vol. 6, pp 13~21*

# 확률적 명제 논리 프로그래밍[1,2]

신 양 규[3]

**요약** 논리 언어로 불확실한 정보의 표현과 처리가 가능하도록 논리 프로그램을 확장하였다. 이러한 확장을 위해 의미론이 명확한 확률 논리를 응용하였고, 확률적 연역 추론을 위해 추론 규칙들을 공리화하여 기본 지식과 함께 처리될 수 있게 하였다. 여기서는 기존 논리 프로그램의 명제 부분만을 대상으로 하였으며, 확장된 논리 프로그래밍 언어는 기존 언어에서 간단한 인터프리터를 사용하여 쉽게 구현하여 이용할 수 있다.

*주제어:* 확률 논리, 확률 논리 프로그래밍, 연역 추론, 인공지능

## 1. 서론

확률론 및 통계학의 응용은 인접 학문 분야들 중 하나인 전산학에서도 오랫동안 다양하게 이루어져 왔다. 특히 인공지능 분야에서는 초창기부터 불확실한 지식의 표현과 처리에 확률 및 통계적 이론들을 응용하고 있다. 예를 들어, 전문가 시스템들 중에서 초기 시스템에 해당되는 PROSPECTOR와 MYCIN은 각각 베이즈 확률론과 Stanford Certainty Theory라는 독자적인 방법을 이용하였으며, 그 외에도 많은 시스템들이 Zadeh의 퍼지 이론이나 Dempster와 Shafer의 evidence 이론 등을 사용하여 지식의 불확실성 문제를 해결하고 있다[9].

한편 인공지능에서 지식의 표현과 추론을 위해 사용하는 가장 보편적인 방법은 논리 언어를 이용한 연역 추론이다. 이유는 논리 언어가 간결한 문법과 풍부하면서도 명확한 의미론을 제공할 수 있기 때문이며, 또한 연역 추론은 (논리 언어의 특정 부분 집합에서) 기계적으로 처리될 수 있기 때문이다. 의미론적 입장에서 볼 때 논리 언어는 표현된 문장의 진리값이 참 혹은 거짓 중 하나라는 입장을 취한다. 그러나 논리 언어가 갖는 이러한 의미론적 특징은 인공지능에서 표현하고 처리하고자 하는 지식이 "불확실"한 경우 논리 언어를 그대로 이용할 수 없다는 문제점이 발생한다. 비록 퍼지 논리가 이에 대한 한 가지 대안이라고 볼 수 있으나 명확한 의미론을 제공하기 힘든다는 문제점이 제기된다[16].

---

1 A Probabilistic Propositional Logic Programming

2 본 연구는 1994년도 경산대학교 학술연구 조성비의 지원을 받았음.

3 경북 경산시 점촌동 경산대학교 통계학과

지금까지 논리 언어의 명확한 의미론을 그대로 유지하면서 불확실한 지식을 표현하고 처리하고자 하는 다양한 연구가 계속되어 왔다[1,2,3,5]. 이러한 연구는 지식 표현의 측면에서 논리적 방법론과 확률론적 방법론 사이의 연결을 위한 시도라고 볼 수 있다. 그 중 하나가 Nilsson에 의해 개발된 "확률 논리(probabilistic logic)"인 데[6,11,13], 이 이론은 비 고전 논리 중 하나인 modal 논리의 "상상의 세상의 의미론" (possible worlds semantics)을 확률론의 입장에서 재 구성하였으며, 논리적 entailment의 대체 개념으로 확률적 entailment의 개념과 이를 계산할 수 있는 방법을 제시하였다. 그러나 Nilsson 자신도 언급한 바와 같이 확률 논리는 의미론적 입장에 충실한 반면 확률적 연역 추론을 위한 증명론적 방법이 빈약하다는 단점을 갖고있다[14]. 이에 대해 Frisch와 Haddawy는 조건부 확률을 이용하여 확률 논리에서의 연역 시스템을 제안하였는데[5], 각 문장의 진리값에 해당하는 확률 구간을 이용하여 " 자연 연역 추론(natural deduction)" 방식으로 처리하도록 다양한 추론 규칙들을 구성하였다. 그러나 이 방법은 기계적으로 자동화하는데 많은 어려움이 있고, 또 효율성도 좋지않다는 단점이 있다.

본 논문에서는 확률 논리를 기존의 논리 프로그래밍 언어에 적용시켜 확률적 연역 추론이 가능하게 하며, 동시에 자연 연역 추론 보다 효율성이 우수하며 구현이 매우 간단한 새로운 증명론적 방법을 제시하는데 목적이 있다. 이를 위해 논리 프로그래밍 언어는 명제 논리로 한정하였으며, 실제 구현은 간단한 인터 프리터를 통해 가능성을 보일 것이다. 본 논문의 구성은 다음과 같다. 제 2절에서는 논리 프로그래밍 언어에 대해 기술하고, 3절에서는 확률 논리를 설명한다. 그리고 4절에서는 확률 논리를 적용한 논리 프로그래밍 언어와 이의 시험 및 구현 방법을 제시한다.

## 2. 논리 프로그래밍

여기서는 일차 논리와 그의 부분 집합인 Horn절 논리로 구성된 논리 프로그래밍의 내용을 간단히 요약한다. 다음에 소개되는 개념들에 대한 보다 자세한 내용은 논리 혹은 논리 프로그래밍에 관한 문헌들을 참고할 수 있다[4,6,7,8,9].

### 2.1 일차 술어 논리 언어

**일차 술어 논리 언어**란 주어진 알파벳에서 정의될 수 있는 모든 식들의 집합이다. 알파벳과 식 및 그에 관련된 개념들에 관한 정의는 다음과 같다. 하나의 **알파벳**은 변수(x,y,z,u,v ... 등으로 표현), 상수(a,b,c,d ... 등으로 표현), 함수 기호 (f,g,h ... 등으로 표현), 관계 기호 (p,q,r ... 등으로 표현), 명제 상수(참, 거짓), 논리 연산자 (¬(not), ∨(or), ∧(and), →(imply), ↔(if and only if)), 한정사 (∀, ), 그리고 괄호와 쉼표 같은 구둣점 등 모두 8종류의 기호들의 집합이다.

이때 명제 상수, 논리 연산자, 한정사, 그리고 구둣점들은 고정된 집합들이다. 또한 변수들의 집합은 무한 집합으로 가정하고 역시 고정된 집합이라 한다. 이러한 다섯 종류의 기호들을 **논리적 기호**들이라 부른다. 나머지 기호들, 즉 상수, 관계기호,

함수 기호들은 가변적이고 경우에 따라서는 공집합일 수도 있다. 이러한 세 종류의 기호들은 **비논리적 기호**(nonlogical symbol)들이라 한다. 각 함수와 관계 기호들은 일정한 갯수의 항을 가진다. 특히 0항 함수들은 상수들로 볼수 있고, 0항 관계들은 **명제기호**(혹은 **명제**)들이라 볼 수 있다. 따라서 하나의 알파벳은 상수, 함수, 그리고 관계 기호들에 의해 정의된다.

**항**(term)은 변수, 상수, 혹은 f가 *n*항 함수이고 $t_1$, ..., $t_n$이 항들일 때 $f(t_1, ..., t_n)$와 같은 형태를 말한다. 여기서 항은 s, t, u로 나타낸다. p가 만약 n항 관계이고 $t_1$, ..., $t_n$이 항들이면 $p(t_1, ..., t_n)$은 **식**이며 (이러한 식을 술어(predicate) 또는 원자식(atomic formula)이라 한다), 참과 거짓 또한 **식**이다. 만약 F와 G가 식이면 $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$ 그리고 $(F \leftrightarrow G)$은 **식**이다. 또한 F가 식이고 x가 변수이면, $(\forall x F)$와 $(\exists x F)$ 역시 **식**이다.

새로운 논리 기호 $\leftarrow$을 소개하는데, $(G \leftarrow F)$은 $(F \rightarrow G)$와 논리적으로 합동인 식이다. 이와 같이 식의 문법적 구성은 엄격하게 정의되므로 괄호를 많이 사용하여야 하는 불편함이 뒤따르지만, 괄호를 가능한 제거할 수 있는 방법은 연결어와 한정사들에게 우선 순위를 부여하는 것이다. 여기서는 $\neg$, $\exists$, $\forall$이 가장 우선 순위가 높고 그 다음에 $\wedge$, 그리고 $\vee$, 마지막으로 $\rightarrow$와 $\leftrightarrow$의 순으로 한다. 또한 $\wedge$, $\vee$,$\rightarrow$와 $\leftrightarrow$은 우결합 법칙(right association)을 만족한다고 가정한다.

## 2.2 논리 프로그램

하나의 원자식이나 혹은 그의 부정을 literal이라고 부른다. 경우에 따라서는 원자식을 양의 literal, 그의 부정을 음의 literal이라고 부르기도 한다. 절(clause)은 다음의 형태를 갖는 식이다:

$$\forall x_1, ..., \forall x_n(L_1 \vee ... \vee L_m)$$

이때 변수들 $x_1$, ..., $x_n$은 literal들 $L_1$, ..., $L_m$에 나타나는 것이어야 한다. 이와 같은 형식은 양과 음의 literal들을 따로 구분하여 다음과 같이 다시 배열할 수 있다:

$$\forall x_1, ..., \forall x_n(A_1 \vee ... \vee A_i \vee \neg B_1 \vee ... \vee \neg B_k)$$

그리고 이 식은 다음의 식과 논리적으로 합동이다:

$$\forall x_1, ..., \forall x_n(A_1 \vee ... \vee A_i \quad B_1 \wedge ... \wedge B_k)$$

이때 i=1인 경우를 Horn 절 혹은 프로그램 절이라 하고, 프로그램 절에서 k=0인 경우를 단위 절이라고 한다. 또한 i=0인 경우를 goal이라 한다. 여기서 $A_j$나 $B_l$는 모두 양의 literal들이다.

논리 프로그램은 공집합이 아닌 프로그램 절의 유한 집합이다. 일반 논리 프로그램은 공집합이 아닌 다음과 같은 일반 프로그램 절의 유한 집합이다:

$$\forall x_1, ..., \forall x_n(A \quad L_1 \wedge ... \wedge L_k)$$

여기서 $L_i$ $(1 \le i \le k)$는 양이나 혹은 음의 literal이다. 따라서 일반 논리 프로그램 절은 기호 오른쪽에 부정이 허용되고 있으며 실제 Prolog 언어에서는 Negation-As-

Failure(NAF)로 구현된다. 그러나 본 논문에서 이용할 명제적 논리 프로그램이란 위 식에서 A 와 $L_i$ 모두가 양의 literal인 명제의 경우를 가리킨다.

## 3. 확률 논리

확률 논리는 확률론과 일차 논리의 결합으로서[13], 일차 논리 문장의 진리값을 0 과 1 사이의 확률 값으로 대응시켜 일차 논리의 의미론을 일반화한 것이다. 전통적인 일차 논리에서 어떤 문장 $\phi$의 진리값은 참 혹은 거짓, 둘 중 하나이다. 만약 문장 $\phi$하나만을 갖고 있다면 두 가지 가능한 세상을 가정할 수 있다. 즉, 하나의 세상 $W_1$에서는 $\phi$가 참인 경우이고, 또 다른 하나의 세상 $W_2$에서는 $\phi$가 거짓인 경우이다. 우리가 살고있는 현실 세상이란 이 두 세상들 중 어느 하나에 속하며 구체적으로 어디에 속하는지는 알수 없다. 대신 문장 $\phi$의 확률이 $\rho$라는 것은 현실 세상이 $W_1$에 속할 확률이 $\rho$라는 것을 의미한다. 이와 같은 해석은 modal 논리의 "상상의 세상의 의미론"에 기초를 두고 있다[12,15].

하나 이상 여러개의 문장들이 있다면 더 많은 가능한 세상들을 고려해야 한다. 각각의 문장은 어떤 세상에서는 참이 되고, 또 다른 어떤 세상에서는 거짓이 될 수 있다. 그러나 세상들의 집합에서 문장들의 진리값들은 항상 일관성을 유지해야 한다. 만약 N개의 문장이 있다면 $2^N$개의 가능한 세상들이 있을 수 있으나, 논리적으로 일관성이 없는 경우들을 제외하면 $2^N$개 보다는 적은 수의 세상들만을 고려하면 된다. 예를 들어, 문장 P가 참이고, Q가 참이면서 $P \rightarrow Q$가 거짓인 세상은 존재하지 않기 때문이다. 사실 이 세 개의 문장들이 처할 수 있는 가능한 세상들의 수는 네 개이다. 문장들의 집합에서 각 문장들의 진리값의 조합 중 일관성이 있는 것들의 집합이 가능한 하나의 세상이 되며, 이것이 확률 분포를 정의하는 모 집단을 형성한다. 이러한 확률 분포는 가능한 세상 $W_i$ 각각에 대해 현실 세상이 $W_i$에 속할 확률 $\rho_i$를 정의한다. 가능한 세상들의 집합들은 상호 배반적이므로 개개의 확률 $\rho_i$의 합은 1이다. 현실 세상에서 문장 $\phi$의 진리값을 모른다고 가정하기 때문에 참과 거짓 사이의 중간 값을 가지는 확률 논리에서는 $\phi$의 확률을 그것의 진리값으로 정의하게 된다. 불확실한 지식을 가정할 때 의 확률과 확률 논리에서 의 진리값은 결과적으로 같은 개념이 된다.

이상의 내용은 벡터를 이용하여 재 구성할 수 있다. $\Gamma$를 임의의 순서로 배열된 N개의 일차 논리 문장들의 집합이라고 가정한다. 즉, $\Gamma = \{\phi_1, \phi_2, ..., \phi_N\}$. $\Gamma$의 N개 문장 각각에 대해 진리값을 부여한 하나의 열 벡터 $v$를 다음과 같이 구성한다:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}$$

(단, 여기서 $\phi_i$가 참이면 $v_i = 1$이고, 거짓이면 $v_i = 0$이다)

이 열 벡터는 대응되는 Γ의 문장들이 일관성이 있으면 v 역시 일관성이 있다고 한다. Γ에 대해 일관성이 있는 가능한 모든 열 벡터들의 집합을 V라고 두며, V의 원소의 갯수를 K로 둔다. 따라서 $V = (v_1, v_2, ..., v_K)$로 표현할 수 있으며, 이것은 N×K차원 행렬이 된다. 이때 $K \leq 2^N$ 이며, V의 각 원소들은 Γ의 문장들이 일관성 있게 참 혹은 거짓의 진리값을 가질 수 있는 가능한 세상들에 대응된다. 이를 이용한 간단한 예를 다음에 소개한다.

**예3.1** $\Gamma = \{P, Q \leftarrow P, Q\}$일 때

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

이다: 즉, 가능한 세상의 수는 네 개이고, 각각은 Γ의 세 문장들에 대한 일관성 있는 진리값의 부여를 나타낸다.■

각 문장의 진리값이 불확실할 때 가능한 세상들의 집합 상에서 확률 분포를 정의할 수 있다. K-차원 열 벡터 P가 가능한 세상들의 집합 각각에 대한 확률을 나타낸다고 가정한다. 이때 i번째 원소 $p_i$는 i번째 가능한 세상 $W_i$, 즉 i가 현실 세상일 확률이 된다. 열 벡터 $P = (p_1, ..., p_K)^t$ 에서 $0 \leq p_i \leq 1$ 이며, $\sum_{i=1}^{N} p_i = 1$ 인 조건을 만족해야 한다. 문장들의 확률 값들은 행렬 방정식

$$\Pi = VP$$

로 계산된다. 이때 Π의 i-번째 원소 $\pi_i$는 문장 $\phi_i$가 참일 확률이다.

위의 예제 3.1에서의 경우 비록 두 문장 P와 $Q \leftarrow P$의 일관성 있는 진리값이 결정된다고 해도 결론 Q의 확률은 유일하게 결정되지 않는다. 다만 특별한 조건 하에서 다음과 같은 경계치만을 얻을 수 있을 뿐이다:

$$pr(Q \leftarrow P) + pr(P) - 1 \leq pr(Q) \leq pr(Q \leftarrow P)$$

이러한 경계치로서는 원하는 결론 Q의 확률을 알 수 없으므로 확률 추론에는 도움이 되지 않는다. 따라서 더욱 더 정확(근사)한 확률 값을 계산하기 위한 방법이 제공되어야 한다. 다음에서 Nilsson이 제시한 확률적 entailment 계산 방법을 요약한다:

제 1단계: 먼저 확률적 entailment의 문제를 구성하기 위한 표준형을 만든다. 이미 언급한 바와 같이, 열 벡터 $v_i$로 일관성 있는 진리값을 나타낼 수 있도록 Γ 내의 문장들을 임의의 순서로 나열한다. 그 다음 행렬 V의 첫 번째 행에다 모두 1의 값을 갖는 행 하나를 추가하여 $p_i$의 합이 1이 되도록 제약조건을 만든다. 이 새로운 행은 Γ의 첫 번째 원소로서 T(참)라는 문장을 추가시킨 것으로 볼 수 있다 (T는 가능한 모든 세상에서 참이다). 그리고 유도될 문장 φ를 Γ의 마지막에다 다시 추가시킨다. 따라서 V의 마지막 행은 여러 가능한 세상들에서 일관성 있는 Γ의 진리값을 의미하게 된다. V의 나머지 행들은 기본 지식의 집합 Δ의 문장들에 대한 일관성 있는 진리값을 나타낸다.

제 2단계: 위의 처리를 마치면 Γ의 마지막 문장을 제외하고는 모든 문장들이 일관성 있는 진리값을 가지게 된다. 다음에 N×K의 크기를 갖는 행렬 V를 계산하며, 그 다음 행렬 방정식

$$\Pi = VP$$

를 계산한다.

제 3단계: Π의 마지막 원소가 미지수이므로 K차원 열 벡터 P 역시 미지수다. P를 계산하기 위해서는 먼저 V의 마지막 열을 제거한 (N-1)×K 차원 행렬 V′를 구하고, 그 다음 Π의 마지막 원소를 제거한 (N-1)차원 열 벡터 Π′를 구한다. 이 두개의 값을 이용해 행렬 방정식

$$\Pi' = V'P$$

를 계산한 후 pr(ϕ) = ϕP를 계산한다. 일반적으로 방정식 Π′ = V′P는 미결정 상태이며, 따라서 P에 대한 많은 해를 가지게 된다. 이때 V를 계산 가능하도록 충분히 작은 것으로 가정하여 pr(ϕ)를 한정짓는 답을 찾아야 한다. 답을 구하는 방법은 투사법과 최대 엔트로피법 두 가지가 제시되었으나 여기에 대한 설명은 생략하며 대신 간단한 예를 들어 본다.

**예3.2** Γ = {P, Q ← P, Q}일 때

$$V' = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \qquad \Pi' = \begin{bmatrix} 1 \\ pr(P) \\ pr(Q \leftarrow P) \end{bmatrix}$$

에 대해

$$pr(Q) = -0.5 + \frac{pr(P)}{2} + pr(Q \leftarrow P)$$

이 가능하고, 또한 열 벡터

$$P = \left[pr(P) + pr(Q \leftarrow P) - 1, 1 - pr(Q \leftarrow P), \frac{1 - pr(P)}{2}, \frac{1 - pr(P)}{2}\right]^t$$

으로 구해질 수 도 있다.■

이상에서 본 확률 논리는 의미론의 명확성에도 불구하고 일관성있는 모든 가능한 진리값들을 계산해야 하기 때문에 명제 논리의 경우만 해도 $2^N$ 만큼의 시간이 걸리므로 아주 비 효율적이다. 그리고 더욱 더 중요한 단점은 보편적인 형태의 증명 방법이 존재하지 않는다는 점이다. 다음에서 이 두 가지 단점을 논리 프로그래밍의 범주에서 보완할 수 있는 방법을 제시한다.

## 4. 명제 논리 프로그래밍을 위한 확률 논리의 확장

논리 프로그래밍에서의 증명론은 resolution 원리를 이용한 후향 추론인 반면, 일차 논리에서의 자연 연역 추론은 다양한 추론 규칙들을 이용한 전향 추론이다. 논리

프로그래밍은 하나의 추론 규칙만을 이용하지만 처리 속도가 빠르며 프로그래밍 언어로 구현되어 있어서 범용성이 뛰어나다.

논리 프로그래밍에서 불확실한 지식의 표현이 가능해 지기 위해서는 Horn절에다 확률값을 대응시켜야 한다. 그러나 한 가지 문제점은 확률은 문장을 인수로하는 함수라는 점이다. 즉, 일반적으로 문장 $\phi$의 확률 pr($\phi$)는 함수이지만, 논리 언어의 입장에서 보면 함수 이름인 pr이 인수로서 식의 형태인 문장을 가질 수 없다는 점이다. 이 문제는 메타 술어를 이용하여 해결할 수 있다. 따라서 본 논문에서는 "문장 $\phi$의 확률이 $\pi$이다"라는 사실을 나타내는 메타 술어로 pr($\phi$, $\pi$)을 사용한다.

Nilsson은 확률 논리에서 확률 값을 $\pi \in [0, 1]$인 한 점으로 정의한 반면, 여기서는 Frisch와 Haddawy의 시도 처럼 $\pi \subseteq [0, 1]$인 구간으로 정의하고자 한다. 따라서 불확실한 지식을 나타낼 때는 문장의 진리값을 0과 1 사이의 최소와 최대 확률값으로 표현하기로 한다. 이러한 표현은 한 점을 사용한 표현법보다 더 일반적이다. 예를 들어, 문장 Q ← P의 확률 (혹은 진리값)이 0.5인 경우 pr(Q ← P, [.5, .5])와 같이 나타낼 수 있기 때문이다. 또한 구간 확률을 대응시킨 이 방법은 확률 논리에서의 의미론을 그대로 유지한다는 사실이다[5].

구간 확률을 이용하여 문장의 불확실성을 표현한 후 다음으로 필요한 것은 어떤 결론의 근사한 확률 값을 계산해 낼 수 있는 추론 규칙들이다. 여기서는 논리 프로그래밍에서 기본적으로 필요한 추론 규칙들을 구성한다:

$$\text{(R1)}\ \frac{}{pr(true,[1,1])} \qquad \text{(R2)}\ \frac{}{pr(false,[0,0])}$$

$$\text{(R3)}\ \frac{\phi \text{ is an ordinary Horn clause}}{pr(\phi,[1,1])} \qquad \text{(R4)}\ \frac{pr(\phi,[x,y]),[x,y]\subseteq[u,v]}{pr(\phi,[u,v])}$$

$$\text{(R5)}\ \frac{pr(\phi,[x,y]),\ pr(\varphi \leftarrow \phi,[u,v])}{pr(\varphi,[\max(0,x+u-1),v])} \qquad \text{(R6)}\ \frac{pr(\phi,[x,y]),\ pr(\varphi,[u,v])}{pr(\phi \wedge \varphi,[\max(0,x+u-1),\min(y,v)])}$$

위의 추론 규칙 표현에서 $\frac{\beta}{\alpha}$는 "$\alpha$에서 $\beta$를 결론으로 추론한다"로 읽는다. 이때 만약 추론 규칙 R1과 R2 같이 $\beta$ 부분이 없으면 결론 $\alpha$는 항상 유도 가능한 것이 된다. 따라서 R1과 R2 전제 조건과 무관하게 참(true)의 진리값은 1이며, 거짓(false)의 진리값이 0이라는 기본적인 사실을 나타낸 것이다. R3는 불확실하지 않은 모든 문장(확률로 표현되지 않은 문장)은 확률이 1임을 나타낸 것이고, R4는 확률 구간의 확장 규칙이다. 마지막으로 R5와 R6은 modus ponens와 conjunction 규칙을 확률 구간의 관점에서 재 구성한 것이다. 이를 이용한 간단한 예를 다음에서 본다.

**예 4.1** 두 개의 문장으로 구성된 논리 프로그램 $\Gamma$ = {p ← q, pr(q, [.2, .2])}을 가정한다. 첫번째 문장 p ← q는 일반 명제 논리 Horn절 이므로 규칙 R3에 의해 pr(p ← q, [1,1])과 같은 결론을 얻는다. 그 다음 규칙 R5에 의해 pr(p, [.2, 1])을 얻는다.■

한편 이와 같은 추론 규칙들을 논리 프로그래밍에서 직접 활용하기 위해서는 이들을 공리화 하여야 한다. 다시 말해서 이들 추론 규칙이 논리 식으로 표현되어 문장들의 집합 Γ내에 함께 포함되어 있어야 한다. 논리 프로그래밍으로 표현한 추론 규칙들의 공리화는 다음과 같이 이루어질 수 있다:

(*A*1) $pr(true,[1,1]) \leftarrow$

(*A*2) $pr(false,[0,0]) \leftarrow$

(*A*3) $pr(P,[1,1]) \leftarrow ordinary(P)$

(A4) $pr(P, [X,Y]) \leftarrow pr(P, [U,V]) \wedge [U,V] \subseteq [X,Y]$

(A5) $pr(P, [Z,V]) \leftarrow pr(P \leftarrow Q, [U,V]) \wedge pr(Q, [X,Y]) \wedge Z = \max(0, X+U-1)$

(A6) $pr(P \wedge Q, [W,Z]) \leftarrow pr(P,[X,Y]) \wedge pr(Q,[U,V]) \wedge W = \max(0, X+U-1) \wedge Z = \min(Y,V)$

A1 ~ A6까지 여섯개의 문장들은 R1 ~ R6까지의 규칙들에 대응되며, 이것은 일반적인 논리 프로그래밍으로 표현된 문장들에다 불확실성을 나타내는 확률 값이 대응되면 확률적 연역 추론으로 결론을 얻어낼 수 있게해 준다. 다음의 예를 살펴 본다.

**예 4.2** 네 개의 문장으로 구성된 논리 프로그램

Γ = { pr(cough, [.8, .9]), pr(fever, [.9,1]), pr(flu ← cough ∧ fever, [.6, .7]),
pr(cold ← cough ∧ fever, [.8, .9])}

을 가정한다. 마지막 두 문장은 기침과 고열이날 때 독감일 확률과 감기일 확률을 각각 가정한 것이다. 여기서 기침과 고열이 나는 현상이 참일 확률은 A6에 의해 pr(cough∧fever, [.7, .9])으로 구해진다. 따라서 두 가지 결론 pr(flu, [.3, .7])과 pr(cold, [.5, .9])을 얻을 수 있다.■

그러나 이러한 공리화가 확률론의 기본적인 성질들을 모두 갖춘것은 아니다. 예를 들어, 여기서는 두 가지 사실 P와 Q가 배반적일 때 P∧Q의 확률에 대해 공리화 하지 않았다. 뿐만 아니라 다른 논리식, 예를 들어 P∨Q에 대한 확률도 정의하지 않았다. 확률적인 자세한 성질들은 필요에 따라 공리화(추론 규칙화)하여 추가할 수 있을 것이고, 논리적 연결어들의 다양한 결합에 따른 논리식들은 명제 논리 프로그래밍의 범주 내에서는 특별히 필요치 않다.

## 5. 결론

Nilsson의 확률 논리는 명확한 의미론을 갖는 반면에 확률적 연역 추론이 힘들며, 확률 논리를 구간 확률로 확장한 Frisch와 Haddawy의 자연 추론 시스템은 추론의 효율성과 구현이 어렵다는 문제점이 있다. 따라서 본 논문에서는 확률 논리를 논리 프로그래밍 언어에 적용시켜 확률적 연역 추론이 가능하게 하며, 동시에 자연 연역 추론 보다 효율성이 우수하며 구현이 매우 간단한 증명론적 방법을 제시하였다. 논리 프로그래밍은 이미 프로그래밍 언어로서 그 범용성과 효율성이 입증되었다. 따라

서 확률론적 방법과 프로그래밍 언어로서의 논리적 방법 사이의 결합은 비록 명제 논리로 한정된 특별한 경우이지만 구체적으로 가능하고, 그 효용성 또한 높을 것으로 예상할 수 있다. 명제 논리로 제한한 것은 문제의 단순화를 위해 본 논문에서 채택한 제한이므로 앞으로 계속 연구되어야 할 부분이다. 또한 확률론적 기본 성질들, 예를 들어 배반성과 종속성, 조건부 확률 등을 포함시킬 필요도 있다.

## 참 고 문 헌


[1] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities*,The MIT Press,1990.

[2] F. Bacchus, A. Grove, J. Halpern, D. Koller, "From Statistics to Beliefs", AAAI-92, 602-608, 1992.

[3] A. Balke, J. Pearl, "Probabilistic Evaluation of Counterfactual Queries", AAAI-94, 230-237, 1994.

[4] C. Chang, R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press,1973.

[5] A. Frish, P. Haddawy, Anytime Deduction for Probabilistic Logic, *Artificial Intelligence* 69, 93 - 122, 1994.

[6] M. Genesereth, N. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publishers, Inc.,1987.

[7] R. Kowalski, *Logic for Problem Solving*, North-Holland, 1979.

[8] J. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1984.

[9] G. Luger, W. Stubblefield, *Artificial Intelligence*, 2nd ed, The Benjamin /Cummings Publishing, 1993.

[10] E. Mendelson, *Introduction to Mathematical Logic*, 3rd ed, Wadsworth & Brooks/Cole Advanced Books & Software, 1987.

[11] M. McLeish, "Probabilistic Logic: Some Comments and Possible Use for Nonmonotonic Reasoning", *Uncertainty in Artificial Intelligence* 2, 55 - 62, 1988.

[12] R. Moore, "A *Formal Theory of Knowledge and Belief", Formal Theories of the Commonsense World* (J. Hobbs and R. Moore, editors), Ablex Pub. Corp., 319-358, 1985.

[13] N. Nilsson, "Probabilistic Logic", *Artificial Intelligence* 28, pp. 71 - 87, 1986.

[14] N. Nilsson, "Probabilistic Logic Revisited", *Artificial Intelligence* 59, 39 - 42, 1993.

[15] C. Smorynski, *Modal Logic and Self-Reference, Handbook of Philosophical Logic, II: Extensions of Classical Logic*, D. Reidel Publishing Company, 441-495, 1984.

[16] R. Turner, *Logics for Artificial Intelligence*, Ellis Horwood Ltd.,1984.