□ 기술해설 □

제한 논리 프로그래밍언어†

한국전자통신연구소 신동하*
숙명여자대학교 창병모*

● 목 차 ●

1. 서 론

제한 논리(constraint logic) 프로그래밍은 두 가지의 선언적 패러다임인 논리 프로그래밍과 제한 풀이(constraint solving)의 결합이라고 할 수 있는데 기존 논리 언어의 다음과 같은 문제점을 인식하면서 시작되었다. 첫째, 논리 프로그램에서 다루어지는 객체는 해석되지 않는 구조를 표현한다는 점이다. 따라서 모든 의미를 가지는 객체의 의미는 반드시 명시적으로 표현되어야 한다. 이러한 제약은 논리 프로그래밍의 추론이 원시적인 수준의 추론으로 머물게 하였다. 반면에 제한 언어에서는 의미 있는 객체 사이의 관계가 암시적으로 기술될 수 있다. 둘째, 논리 프로그램의 실행 과정이 깊이 우선(depth-first) 탐색으로 인하여 결국은 "생성 후 검사(generate and test)" 알고리즘이 되었다는 점이다. 잘 알려진 바와 같이 이 방법은 큰 탐색 문제에 대해서 성능의 문제점을 가지고 있다. 70년대 후반부터 80년대 초에 걸쳐서 인공 지능 분야에서 탐색을 보다 지능적으로 하기 위해서 제한 조작과 전파에 관한 연구가 활발하게 진행되었다[16,18,19]. 이러한 연구를 바탕으로 Gallair는 논리 프로그래밍의 탐색을 향상시키기 위하여 제한을 사용하는 방법을 제시하였으며[8], 기존의 "생성 후 검사(generate and test)" 알고리즘의 수행력을 향상시키기 위하여 제한을 도입하여 "제한 후 생성(constraint and generate)" 알고리즘을 제시하였다. 그 후로 다양한 추론 방법이 제시되었으며, 여기서 핵심적인 것은 탐색 과정과 제한 풀이 과정의 밀접한 결합이라고 할 수 있다. 이로 인해서 제한 논리 프로그램은 기존의 논리 프로그램보다 더 풍부한 표현력을 가지게 되었고 또한 보다 효율적으로 수행될 수 있게 되었다.

이 글은 제한 논리 언어에 대한 기초 이론, 언어의 종류 및 응용에 대해서 소개한다. 2절에서는 제한 논리 언어의 안전성(soundness) 및 완전성(completeness)에 대하여 설명하고, 제한 논리 언어의 수행 과정 및 중요한 도메인에 대한 제한 풀이 방법을 간략하게 소개한다. 3절에서는 제한 프로그래밍의 간단한 역사와 대표적인 제한 논리 언어인 CHIP, CLP(R), Prolog III에 대하여 소개한다. 제한 논리 언어 CHIP[6]은 유한 트리, 유한 정수, 불리언 및 유리수 도메인, CLP(R)[11]은 유한 트리 및

† 본 연구는 정보통신부 국책 과제 6MC1600 중 "논리 언어 컴파일러에 관한 연구"의 부분 지원을 받았습니다.
*정회원

실수 도메인, Prolog III[5]는 무한 트리, 유리수, 불리언 및 리스트 도메인에서의 제한 표현을 제공한다. 4절에서는 제한 논리 언어 응용의 소개로 옵션 거래 시스템 및 모델 기반 전문가 시스템을 소개한다. 5절에서는 이 글에 대한 결론을 맺는다.

2. 기본 이론

본 절에서는 제한 논리 언어를 정의하고 구현하는 데 필요한 기본 이론에 대하여 소개한다. 먼저 제한 논리 언어에 대한 안전하고 완전한 증명 절차(proof procedure)의 존재에 대하여 소개하고, 제한 논리 언어가 수행되는 과정을 추상 기계의 상태 변화(state transition)를 통하여 설명한다. 그리고 제한 논리 언어에서 널리 사용되는 실수 도메인 및 불리언 도메인에서의 제한 풀이를 소개한다.

2.1 안전성과 완전성

논리 프로그램은 "$p_0 :- p_1, ..., p_n$" 모양을 가지는 클로즈(clause)의 집합으로 정의된다. 여기서 p_i는 아톰(atom)이며, 클로즈의 논리적 의미는 "$p_1 \wedge ... \wedge p_n$이 참이면 p_0는 참이다." 이다. 그리고 목표(goal) 클로즈는 "$g_0, ..., g_m$" 모양을 가지며, 여기서도 g_i는 아톰이다. 논리 언어의 중요한 이론적 결과는 논리적 해석(interpretation)에 바탕을 둔 결론을 찾는 기계적 증명 절차가 존재한다는 것이다. 이를 좀 더 정확하게 표현하면 다음과 같으며, 이 사실은 증명 절차의 안전성과 완전성을 의미한다[15].

> "목표 클로즈 G가 논리 프로그램 P의 논리적 결론(logical consequence)이면 SLD 증명 절차는 논리 프로그램 P와 목표 클로즈 G에 대하여 yes 대답을 하고 그 역도 성립한다."

제한 논리 언어는 논리 언어의 클로즈를 구성하는 아톰에 제한의 표현이 가능한 언어이다. 제한은 두 가지 구성 요소인 도메인(domain) D와 원리(theory) τ에 의하여 정의된다. 여기서 도메인 D는 제한을 구성하는 기본 요소의 집합이고, 원리 τ는 제한을 표현하는 연산(function) 및 관계(relation)를 정의하는 공리(axiom)의 집합이다. 예를 들어 실수 집합에서 연산 '+' 및 관계 '='를 가지는 원리를 생각할 수 있다.

제한 논리 언어에서 우리가 특히 관심을 가지는 제한은 해답 구성성(solution-compactness) 조건과 만족 완전성(satisfaction-completeness) 조건을 만족하는 도메인 및 원리이다. 제한 논리 언어가 이 두 조건을 만족하면, 논리 언어에서와 같이, 안전하고 완전한 증명 절차가 존재하기 때문이다[10]. 어떤 도메인이 해답 구성 가능하다는 말은 그 도메인의 모든 요소가 유한 혹은 무한 집합의 제한으로 표현 가능하다는 의미이고, 어떤 원리가 만족 완전하다는 의미는 모든 제한의 진위가 증명 가능하다는 의미이다.

예를 들어 실수의 집합에서 연산 '+' 및 관계 '='를 가지는 원리는 해답 구성성 조건을 만족하지 않는다. 왜냐하면 실수의 구성 요소인 π를 제한의 집합으로 표현하기 불가능하기 때문이다. 그러나 실수의 집합에서 연산 '+', '*' 와 관계 '=', '≤', '<', '≥', '>'를 가지는 원리는 해답 구성 가능하다. 왜냐하면 모든 실수를 제한의 집합으로 표현 가능하기 때문이다. π의 경우 아래와 같이 무한의 제한 집합으로 표현 가능하다.

$3 < \pi < 4$
$3.1 < \pi < 3.2$
$3.14 < \pi < 3.15$
...

또한 정수의 집합에서 연산 '+', '*' 와 관계 '='를 가지는 원리는 만족 완전하지 못하다. 예를 들어 제한 "$X^n + Y^n = Z^n$"를 만족하는 정수 X, Y, Z가 존재하는 지 알려져 있지 않기 때문이다. 그러나 실수 집합에서 연산 '+', '*' 와 관계 '=', '≠', '<', '≤', '>', '≥'를 가지는 원리는 만족 완전하다. Tarski는 실수 집합에서 다항(polynomial) 등식 및 다항 부등식의 만족성을 결정하는 알고리즘이 존재한다는 것을 증명하였다.

제한 논리 언어는 완전하지 않을 수도 있지만 사용되는 도메인에 따라 여러 가지로 구분될 수 있다. 예를 들어 CLP(유리수), CLP(불

리언, 실수), 및 CLP(실수) 등이 있을 수 있다. 제한이 없는 Prolog 언어[3]도 CLP(트리)라는 일종의 제한 논리 언어라고 말할 수 있다. 여기서 트리는 텀을 의미한다.

2.2 프로그램의 수행

제한 논리 프로그램은 논리 프로그램과 비슷하게 "p_0 :- p_1, ..., p_n, C" 모양의 클로즈의 집합으로 정의된다. 여기서 C는 주어진 도메인에서 표현된 제한의 집합이다. 제한 논리 프로그램의 수행은 추상 기계의 상태 변화로 표현 가능하다[4]. 프로그램 수행의 상태는 σ=(V, G, C)로 표현되는 데, 여기서 V는 질의에 존재하는 변수의 집합이고, G는 목표(goal) 클로즈, C는 제한의 집합이다. 이 추상 기계의 최초의 상태는 σ_0= (V, G_0, C_0)로 표현되는데, 여기서 G_0 및 C_0는 각각 목표 클로즈에 있는 아톰과 제한의 집합이다. 추상 기계는 프로그램 P가 있을 때, 다음 조건을 만족하면 상태 σ_i에서 상태 σ_{i+1}로 상태 변화(state transition)가 가능하다.

(1) p_0 :- p_1, ..., p_n, C ∈ P
(2) σ_i = (V, g_0...g_m, C_i)
(3) σ_{i+1} = (V, p_1...$p_n g_1$...g_m, $C_i \cup C \cup (g_0 = p_0)$)
(4) $C_i \cup C \cup (g_0 = p_0)$가 만족 가능하다.

이 상태 변화에는 조건 (1)의 적용 과정에서 비결정성(nondeterminism)이 존재하므로 스택을 사용한 백트랙(backtrack)의 구현이 필요하며, 조건 (4)를 계산하기 위한 제한 풀이 과정이 필요하다. 추상 기계는 앞에서 정의된 상태 변화를 거쳐서 최종 상태 σ_f=(V, ∅, C_f)에 도달하면 수행을 멈추며, 주어진 목표 클로즈에 대한 답으로 제한 집합 C_f가 간략화 되어 변수 집합 V의 범위에서 출력된다.

2.3 제한 풀이[7]

실수 도메인에서 제한 풀이는 연산 '−', '+', '*', '/'를 사용하여 텀을 구성하고, 텀들 사이의 관계는 '=', '≠', '<', '≤', '>', '≥'를 사용한다. 예를 들어 "I+J+K≤10, I>0, J>0, K>0"은 제한이다. 실수 도메인에서의 제한은 크게 선형(linear)과 비선형(nonlinear)으로 나누어진다. 선형 제한인 경우 연산 '*'의 인수로 최대 하나의 변수가 사용가능하고, 연산 '/'의 분모 인수는 숫자이어야 한다. 선형 제한의 제한 풀이는 가우스 제거(Gaussian elimination) 알고리즘 혹은 확장된 심플렉스(Simplex) 알고리즘을 사용하며, 비선형 제한의 경우 Gröbner base 알고리즘이 사용된다. 비선형인 경우 풀이의 복잡도가 매우 크기 때문에 제한 논리 언어 CLP(R)에서는 비선형 제한이 풀이 과정에서 선형이 될 때까지 실제 풀이를 지연시키는 방법을 사용하기도 한다. 예를 들어 CLP(R)에서는 다음과 같은 비 선형 프로그램이 있을 때 :

```
zmult(R1,I1,R2.I2,R3,I3) :-
  R3 = R1*R2-I1*I2,
  I3 = R1*I2+R2*I1.
```

질의로 "zmult(1, 2, 3, 4, R3, I3)"를 주면 수행시간 시 비선형이 선형으로 변하여 "R3=-5, I3=10"이라는 답이 나온다.

불리언 도메인에서의 제한은 연산 '¬', '∧', '∨' 등을 사용하여 텀을 표현하고, 텀들 사이의 관계는 '=' 만을 사용한다. 불리언 도메인에서의 제한 풀이로 이용되는 방법은 불리언 일치화(Boolean unification) 알고리즘, 변형된 Gröbner base 알고리즘 등이 사용된다. 기본적으로 불리언 도메인에서의 풀이는 NP-complete이므로 이를 사용할 때는 주의가 필요하다. 다음은 전 가산기(full adder)를 표현한 불리언 도메인 프로그램의 보기이다. 여기서 연산 '⊕'는 xor를 의미한다.

```
full_adder(I1,I2,I3,O1,O2) :-
  X1 = I1⊕I2. A1 = I1∧I2,
  O1 = X1⊕I3, A2 = I3∧X1,
  O2 = A1∨A2.
```

3. 제한 논리 프로그래밍언어

제한 논리 언어가 등장하기 전에 이미 제한을 이용한 시스템이 등장하였는데 예를 들면 대화식 그래픽 시스템 THINGLAB[2], 회로 해석을 위한 CONSTRAINTS[19], 대수 문제 해결을 위한 MACSYMA[17] 등을 들 수 있

다. 그러나 이들은 제한 프로그래밍언어보다는 제한 프로그래밍언어의 요소를 가지는 시스템이라고 할 수 있다. 이와는 별도로 제한 만족 문제(constraint satisfaction problem)를 해결하기 위한 활발한 연구가 인공 지능, OR(Operations Research) 등의 분야에서 진행되어 왔다 [20].

제한 논리 프로그래밍[12]은 그 이전에 논의되었던 제한 풀이(constraint solving) 메커니즘을 포함하도록 Prolog 같은 논리 언어를 확장하면서 시작되었다. 논리 언어에 보다 풍부한 자료 구조를 도입하고 의미 객체(예를 들면 수식)가 직접적으로 표현되고 조작될 수 있도록 하였다. 그 기본 아이디어는 논리 언어의 핵심인 일치화(unification)를 제한으로 확장하고 일치화 알고리즘은 도메인 상에서 제한의 해를 구하는 제한 풀이기(constraint solver)로 대치하며 제한 조작과 전파 등에 관한 연구의 영향을 받아서 제한을 적극적으로 사용하여 "제한 후 생성(constraint and generate)"하도록 함으로써 탐색 트리를 줄이자는 것이다. 특정 제한 논리 언어를 설계하기 위해서는 계산 도메인과 제한 풀이기를 선택해야 한다. 이 과정에서 계산 도메인의 표현력, 효율적인 제한 풀이 알고리즘의 존재 여부, 가능한 응용에 대한 관심도 등과 같은 기준을 고려해야 한다. 이들 중 특히 제한 언어의 구현을 위해서 가장 중요한 점이 주어진 계산 도메인에서 제한 시스템이 만족되는지 검사하는 제한 풀이 알고리즘의 선택이다. 본 절에서는 지금까지 소개된 대표적인 제한 논리 언어인 CHIP, CLP(R), Prolog III에 대하여 살펴본다.

3.1 CHIP

CHIP(Constraint Handling in Prolog)[6]은 독일의 ECRC에서 개발된 제한 논리 언어로 가장 중요한 특징은 정수의 특정 구간에 해당하는 유한 도메인 상에서의 산술 제한의 도입과 풍부한 기호 제한의 제공이다. 또한 심볼 값을 포함하는 확장된 불리언 도메인에 대한 제한, 유리수에 대한 선형 제한 등을 제공한다. CHIP은 사용자로 하여금 자신의 제한을 정의하고 그 수행을 제어할 수 있도록 한다. CHIP에서 제공되는 도메인 및 도메인 상에서 가능한 연산 및 관계를 정리하면 다음과 같다.

도메인	연산 및 관계
유한 트리	{=}
유한 정수	{+, −, *, =, ≠, <, ≤, >, ≥} 및 심볼 변환
불리언	{¬, ∧, ∨, *xor*, *nand*, *nor*, =}
유리수	{+, −, *, =, ≠, <, ≤, >, ≥}

유한 도메인에 대한 산술 제한은 일관성 기법을 이용하고, 불리언 도메인에 대한 제한은 불리언 일치화 알고리즘을 이용하며, 선형 유리수에 대한 제한은 확장된 심플랙스 알고리즘을 이용하여 해를 구한다.

CHIP을 기반으로 한 상용 시스템이 개발되었는데 예를 들면 Bull 사는 CHARME 시스템에 유한 도메인 기술을 도입하였으며, ICL 사는 CHIP/SEPIA 컴파일러를 기초로 하여 DECISION POWER를 개발했으며, Siemens 사는 SNI-Prolog의 새로운 버전을 개발하였으며 프랑스 COSYTEC 사에서는 CHIP 해석기를 상품화하였다.

3.2 CLP(R)

제한 논리 언어 CLP(R)[10,11]은 호주의 Monash 대학, 미국의 IBM Watson 연구소, Carnegie Mellon 대학 등에서 개발하였다. 기호 R은 실수와 해석되지 않는 함수 이름으로 구성된 대수 구조(algebraic structure)를 나타낸다. 이 언어는 실수에서의 선형 제한을 허용하고, 비선형 제한은 지연 계산을 통해서 해결한다. 유한 트리에 대해서는 $t_1 = t_2$ 형태의 제한만이 가능하다. CLP(R)에서 허용하는 규칙(rule)은 $A_0 : \alpha_1, \alpha_2, \ldots, \alpha_k$ 형태이고 여기서 α_i는 기초 제한이거나 아톰이다. 이 언어에서 제공되는 계산 도메인과 가능한 연산 및 관계는 다음과 같다.

도메인	연산 및 관계
유한 트리	{=}
실수	{+, −, *, =, ≠, <, ≤, >, ≥}

CLP(R)의 제한 풀이기는 확장된 심플렉스 알고리즘을 사용하며 비선형 제한이 허용되나 제한 풀이 과정에서 비선형 제한은 선형 제한이 될 때까지 지연되며 끝까지 선형 제한이 되지 않는 경우에 해답은 "maybe"이고 해당 비선형 제한을 함께 출력한다.

3.3 Prolog III

Prolog III[5]는 Prolog가 개발된 프랑스의 Marseille 대학에서 개발되었다. 이 언어는 Prolog에 불리언 도메인에 대한 제한, 유리수에서의 선형 제한, 스트링 도메인에서의 제한 등을 도입하여 확장하였다. 확장된 도메인과 가능한 연산 및 관계는 다음 표와 같다.

도메인	연산 및 관계
무한 트리	{=, ≠}
유리수	{+, −, *, =, ≠, <, ≤, >, ≥}
불리언	{¬, ∧, ∨, ⊃, =, ≠}
리스트	{·, 리스트와 트리 간의 변화, =, ≠}

Prolog III는 선형 유리수에 대한 제한은 확장된 심플렉스 알고리즘을 이용하고, 불리언 제한은 포화(saturation) 방법을 이용하며, 리스트에 대한 제한은 제약된 스트링 일치화 알고리즘을 이용하여 해를 구한다.

4. 응용 분야 소개

제한 논리 프로그래밍은 매우 선언적이고[1], 심볼 계산 능력뿐만 아니라 수치 계산 능력이 뛰어나고, 논리적 변수(logical variable)의 사용에 따른 표현력 때문에 그 응용 분야가 매우 광범위하다. 주된 응용 분야는 심볼 계산 분야, 수치 해석 및 OR 분야, 순열 조합 연구(combinatorics) 분야, 인공 지능 분야, 공학 응용 분야 등이다. 본 절에서는 옵션 거래 시스템 및 전문가 시스템에서의 응용 보기에 대하여 간단하게 설명한다.

4.1 옵션 거래 시스템[14]

옵션(option)이란 주어진 자산의 가치에 따라 그 가치가 변하는 증서이다. 가장 일반적인 종류의 옵션은 주식 자산에 대한 옵션이다. 콜(call) 옵션의 소유주는 주어진 주식을, 주어진 날짜 이전에, 주어진 가격에 살 수 있는 권리를 가지며, 풋(put) 옵션의 소유주는 반대로 팔 수 있는 권리를 가진다. 옵션은 그 자체가 가치를 가지기 때문에 매매가 가능하다. 옵션 거래에는 다양한 전략이 요구되고, 복잡한 수치 계산이 필요하기 때문에 제한 논리 언어의 흥미로운 응용 분야이다. 미국의 IBM Watson 연구소와 호주의 Monash 대학에서는 CLP(R)[9]을 이용하여 이를 구현하였다.

예를 들어 옵션 거래의 이익을 계산하는 제한 논리 프로그램은 아래와 같다. 이 프로그램에서 변수 T, S, C, P. I, R, K는 각각 옵션의 종류, 현재 주가, 콜 옵션의 가격, 풋 옵션의 가격, 이자율, 옵션에 지정된 주가를 의미한다.

```
payoff(T,Buy_Sell,S,C,P,I,K,Value) :-
  get_sign(Buy_Sell,Sign),
  data(T,S,C,P,I,K,B1,B2,H1,H2,R1,R2),
  h(B1,S,T1), h(B2,S,T2).
  r(B1,S,T3), r(B2,S,T4),
  Value=Sign*(H1*T1+H2*T2+R1*T3+R2*T4).

get_sign(buy,-1).
get_sign(sell,1).

h(X,Y,Z) :- Y < X, Z = 0.
h(X,Y,Z) :- Y >= X, Z = 1.
r(X,Y,Z) :- Y < Z, Z = 0.
r(X,Y,Z) :- Y >= X. Z = Y-1.

data(call,S,C,P,I,K,0,K,C*(1+I),0,0,-1).
data(put,S,C,P,I,K,0,K,P*(1+I),0,1,-1).
```

위 프로그램에 다음 질의는 콜 옵션의 가격이 5, 콜 옵션의 지정 주가가 50, 이자율이 0.05, 현 주가가 60일 때, 콜 옵션을 사용하여 수식을 매입하면 얼마의 이득이 있는 가를 알아보는 질의이다. 이 질의에 대한 대답은 "Value = -4.75"이다.

```
?- C = 5, K = 50, I = 0.05, S = 60,
   payoff(call,sell,S,C,_,I,K,Value).
```

좀더 복잡한 질문으로 가격 4의 콜 옵션과 가격 3의 풋 옵션을 사는 경우 이익이 3보다 크기 위한 주가를 알아보는 질의는 다음과 같다. 이 질의의 답은 "Value = 42.65-S, 0 < S <

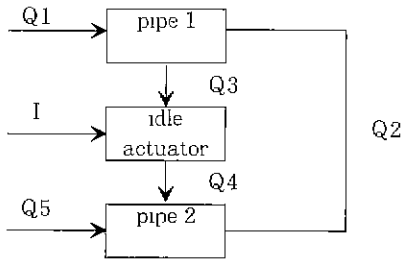
= 39.65" 혹은 "Value = S-57.35, S >= 60. 35"이다.

```
?- C = 4, P = 3, I = 0.05, K = 50,
   Value = V1+V2, Value > 3,
   payoff(call,buy,S,C,_,I,K,V1),
   payoff(put,buy,S,_,P,I,K,V2),
```

4.2 전문가 시스템[13]

제한 논리 언어는 모델 기반(model-based) 진단 전문가 시스템 개발에 탁월한 표현력을 발휘한다. 독일의 다임러 벤츠 회사가 Prolog III[5]를 사용하여 개발한 자동차 고장 진단 전문가가 시스템이 그 예이다. 이 시스템은 자동차 엔진을 계층적으로 분해한(decompose) 후 분해된 각 요소에 대한 입출력 관계를 제한을 사용하여 표현한다. 예를 들어 다음 그림과 같은 입출력 특성을 가진 작은 부 시스템을 가정한다.

이 시스템은 다음과 같은 프로그램으로 표현



된다.

```
system([S1,S2,S3],[Q1,Q2,Q3,Q4,Q5,I]) :-
   pipe(S1,Q3,Q2,Q1),
   idle_actuator(S2,I,Q3,Q4),
   pipe(S3,Q4,Q2,Q5).
pipe(1,Q1,Q2,Q3) :- Q3 = Q1+Q2.
pipe(0,Q1,Q2,Q3) :- Q3 =\= Q1+Q2.
idle_actuator(1,I,Q3,Q4) :-
   정상일 조건 나열.
idle_actuator(0,I,Q3,Q4) :-
   고장일 조건 나열.
```

위 프로그램에서 각 술어의 첫 인수에 사용되는 변수 S1, S2, S3는 각 구성 요소의 상태(1이면 정상, 0이면 고장)를 표현한 불리언 변수이다. 이 프로그램에 실제 시스템 각 요소의 입출력 값을 측정하여 얻은 값을 입력하면 어느 부분이 고장인 지를 알아낼 수 있다. 예를 들어 질의 "system([S1, S2, S3], [40, 37, 3, 3, 40, 450])"를 주면 답으로 "S1=1, S2=1, S3=1"이 나온다. 또한 질의 "system([1, 1, 1], [40, Q2, Q3, 18, Q5, 800])"은 모든 구성 요소가 정상 동작을 할 때 Q2, Q3, Q5의 값을 알아보는 질의이다.

6. 결 론

제한 논리 프로그래밍은 논리 프로그래밍과 제한 풀이가 결합된 새로운 프로그래밍으로 기반 이론이 잘 정립되어 있고 표현력과 수행성이 우수하여 그 미래가 밝은 언어이다. 특히 이 분야의 연구는 컴퓨터 과학의 여러 다른 분야인 프로그래밍언어, 컴파일러, 데이타 베이스, 소프트웨어 공학, 인공 지능 등의 분야에 미치는 영향이 크기 때문에 그 중요성이 더욱 강조되고 있다. 최근 유럽 및 미국 등의 선진국에서는 그 중요성으로 인하여 이 분야에 관한 연구가 활발하게 진행되고 있으며 이미 제한 논리 시스템이 상품화되었으며 이를 이용한 응용 소프트웨어가 개발되고 있다. 본 글을 통하여 국내에서도 제한 논리 프로그래밍에 관한 관심과 연구가 활성화되어 학교 및 관련 연구소간의 활발한 연구가 진행되기를 기대한다.

참고문헌

[1] A. L. Ambler, M. M. Burnett, and B. A. Zimmerman, Operational Versus Definitional : A Perspective on Programming Paradigms, 28-43, *Computer*, September 1992.

[2] A. Borning, The Programming Language Aspects of ThingLab, A Constraint-Oriented Simulation Laboratory, *ACM Trans. on Prog. Lang. and Syst.* Vol. 3, No. 4, 252-387, 1983.

[3] W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, Fourth Edition, Springer-Verlag, 1994.

[4] J. Cohen, Constraint Logic Programming Languages, *Communications of the ACM*, Vol. 33, No. 7, 52-68, 1990.

[5] A. Colmerauer, An Introduction to Prolog III, *Communications of the ACM*, Vol. 33, No. 7, 69-90, 1990.

[6] M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier, The Constraint Logic Programming Language CHIP, *Proceedings of International Conferenc on FGCS*, 693-702, 1988.

[7] T. Fr hwirth, A. Herold, V. K chenhoff, T. L. Provost, P. Lim, E. Monfroy, and M. Wallace, Constraint Logic Programming -An Informal Introduction, Technical Report ECRC-93-5, ECRC, 1993.

[8] H. Gallaire, Logic Programming : Further Developments, *Proceedings of IEEE Symposium on Logic Programming*, 88-99, IEEE, 1985.

[9] N. C. Heintze, J. Jaffar, S. Michaylov, P. J. Stuckey, and R. Yap, The CLP (*R*) Programmer's Manual Version 1.2, IBM J. Watson Research Center, 1992.

[10] J. Jaffar and J.-L. Lassez, Constraint Logic Programming, *Proceedings of 14th ACM Principles of Programming Languages*, 111-119, Jan. 1987.

[11] J. Jaffar, S. Michaylov, P. Stuckey, and R. Yap, The CLP(R) Language and System, *ACM Trans. on Prog. Lang. and Syst*. Vol. 14, No. 3, 339-395, 1992.

[12] J. Jaffar and M. J. Maher, Constraint Logic Programming : A Survey, *The Journal of Logic Programming*, Vol. 19/20, 503-581, 1994.

[13] K. Krautter and M. Steinert, A Knowledge Representation for Model-Based Reasoning using Prolog III, *The Proceedings of the 5th ESPRIT Conference : Puting the Technology to Use*, 814-825, Amsterdam, 1988.

[14] C. Lassez, K. McAloon, and R. Yap, Constraint Logic Programming and Option Trading, *IEEE Expert*, 42-50, Fall 1987.

[15] J. W. Lloyd, *Foundations of Logic Programming*, Second Extended Edition, Springer-Verlag, 1987.

[16] A. K. Mackworth. Constraint Satisfaction, *Encyclopedia of Arificial Intelligence*, 1986.

[17] MathLab, MACSYMA Reference Manual, The MathLab Group. Laboratory for Computer Science. MIT, 1983.

[18] U. Montanari, Networks of Constraints : Fundamental Properties and Applications to Pricture Processing, *Information Science*, Vol. 7, No. 2, 95-132, 1974.

[19] G. L. Steele. The Definition and Implementation of a Computer Programming Language based on Constraints. Technical Report MIT-AI TR 595, M.I.T., 1980

[20] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, 1993.

신 동 하

1980 경북대학교 전자공학과 전산묘돌 졸업, 학사
1982 서울대학교 전자계산기공학과 졸업, 석사
1982. 3~현재 한국전자통신연구소 컴퓨터 연구단, 책임연구원
1994 미국 Univ. of South Carolina, 컴퓨터과학과 졸업, 박사

관심분야 : (논리) 프로그래밍 언어, 컴파일러, 계산 이론, 인공 지능

창 병 모

1988 서울대학교 컴퓨터공학과 졸업, 공학사
1990 한국과학기술원 전산학과 졸업, 공학석사
1994 한국과학기술원 전산학과 졸업, 공학박사
1994.3 ~ 1995.2 한국전자통신연구소, 박사후 연수 연구원
1995.3 ~ 현재 숙명여자대학교 전산학과, 조교수

관심분야 : 정적 분석, 프로그래밍 언어, 연역 데이터베이스