논리와 논리머신

申東弼,* 鄭泰忠**
(正 會 員)
韓國科學技術院 시스템工學센터,*
慶熙大學校 工科大學 電算工學科 助教授**

요 약

지식처리시스템은 지식의 표현과 그 추론을 필요로 한다. 그 표현방법과 추론방법은 여러가지가 있으나 논리(logic)는 매우 뛰어난 지식표현능력과 추론능력을 갖고 있어서 많은 인공지능 연구자들이 관심을 갖고 있다. 이번 원고에서는 일차술어 논리(first order predicate logic)과 prolog가 무엇인지 설명하고, 논리언어를 빠르게 추론해 주는 논리머신은 어떤 종류가 있는지 살펴 보기로 한다.

Ⅰ. 서 론

논리프로그래밍이란 논리에 근거를 두어 프로그래밍하는 유형을 말한다. 기존의 프로그램언어로 프로그램된 프로그램은 컴퓨터가 어떻게(how) 일을 처리할 것인가를 규정해 놓은 절차이다. 따라서 무엇(what)을 해야할지에 관한 논리(logic)와 어떻게(how) 해야할지의 제어(control) 들이 프로그램 내에 섞여 있다.[6] 그 결과 기존의 프로그램은 읽기가 어렵고, 유지 및 보수가 힘들다.

사람이 생각을 전개하는 대로 컴퓨터가 계산하도록 프로그램을 할 수 있다면, 이런 프로그램은 알아보기도 쉽고, 위치 및 보수도 쉬울 것이다. 사람이 생각을 전개하는 것을 이론적으로 정리한 것이 논리(logic)이다. 따라서 논리에 근거를 둔 프로그램언어가 있다면, 기계지향적(machine oriented)인 언어가 아닌 사람지향적(human-oriented)인 언어가 될 것이다. 현재 컴퓨터 상에서 사용가능한 논리언어는 모두 일차술어 논리(first order predicate logic)에 근거를 두고 있으며,[5,8] prolog라는 논리언어는[7] 기존의 폰노이만 머신에서 처리될 수 있도록 만든 순차적 논리언어이다.

논리언어는 병렬처리라는 측면에서도 매우 중요하다. 하드웨어의 가격하락으로 많은 수의 프로세서들을 이용한 컴퓨터의 구성이 쉬워졌다. 문제는 그 많은 프로세서들을 바쁘게 사용할 수 있는 병렬성(parallelism)을 어디서 얻는가 하는 것이다. 그런데 논리언어로 작성된 프로그램 내에는 AND/OR 병렬성 등이 풍부하다. 그러한 병렬성의 근본적인 근원은 "어떻게(how)" 부분을 나타내는 제어(control) 부분을 프로그램에 표현할 필요가 없으므로 문제를 푸는데 자유도가 많기 때문이다.

본 원고에서는 논리언어가 무엇인지, 논리언어 및 논리머신의 종류에 대한 고찰을 한다.

Ⅱ. 논리언어

1. 일차술어논리와 호온절(Horn Clause)

자연어를 논리식으로 표현하는데 일차술어논리(first order predicate logic)을 사용하면 편리하다. 참(true)과 거짓(false)의 값만 갖을 수 있는 predicate를 atom으로 하고 있는 논리로서, atom은 다음의 형태

Predicate-name(term 1, term 2, ⋯, term n)

로 표현되는데, term에 atom이 올 수 없기 때문에 일차논리이다. Term에는 상수, 변수 및 함수 (혹은 structure)가 있으며, 함수는 "funtor(term 1, term 2, ⋯, term n)"의 형식을 갖는다. 일차술어논리의 WFF (well-formed formula)는 atom과 연결자들(connectives)의 결합으로 이루어 진다. 연결자엔 NOT(~), AND(∧), OR(∨), IF-THEN(→) 등이 있다. 또한 변수에 대한 한정자(quantifier)인 ∀(for all)과 ∃(there exists)가 있어서 집합개념을 잘 표현해 주

고 있다.

일차술어논리는 자연어나 논리적 지식을 잘 표현할 수 있다. 예를 들어, "길동이의 모든 조상은 그의 부모이거나 그의 조상의 조상이다."를 일차술어논리로 표현하면,

$$(\forall X)\{조상(길동, X) \rightarrow 부모(길동, X) \vee (\exists Y)\{조상(길동, Y) \wedge 조상(Y, X)\}\}$$

이다. 이렇게 자연어나 지식을 잘 표현해주는 일차술어 논리도 그 표현된 지식을 이용하고 추론하는 방법이 있어야 의미가 있다. 그런데 일차술어 논리 그 자체로 추론하는 효율적 방법이 없으므로, 표현력과 의미는 유지하면서도 추론하기에 좋은 형태로 바꿀 필요가 있다. 그 형태가 conjunctive normal form 이다. 즉 모든 일차술어논리는 OR($\vee$)로 연결된 literal들을 AND($\wedge$)로 연결된 형태로 만들수 있다는 것이다. 즉

$$(L_{11} \vee L_{12} \vee \cdots \vee L_{1n}) \wedge (L_{21} \vee L_{22} \vee \cdots \vee L_{2m}) \wedge (\quad) \wedge \cdots$$

(여기서 literal은 positive atom과 negative atom을 말하며, 모든 변수는 univerally quntified 되어 있다.)

Conjunctive normal form의 "$\vee$"로만 연결된 것을 clause(절) 이라고 한다. 따라서 일차술어논리는 clause들의 집합으로 변환될 수 있는 것이다. 절들의 집합에 대한 추론방식으로 resolution이 있다. 매우 효율적이면서도 증명이 가능한 질문에 대하여는 꼭 답을 찾아주는 성질 즉 complete 한 성질이 있어서, 논리를 컴퓨터언어로 사용가능하게 해 주었다.

일차술어논리를 clausal form으로 바꾸는 과정은 참고문헌[8]에 잘 나와 있다. 하나의 clause는 "$L_1 \vee L_2 \vee \cdots \vee L_n$"으로 표현되는데 Li는 atom 혹은 $\sim$ atom 이라고 하였다. 한 clause의 양의 liferal과 음의 liferal을 나누어 놓으면

"$P_1 \vee P_2 \vee \cdots \vee P_n \vee \sim Q_1 \vee \sim Q_2 \vee \cdots \vee \sim Q_m$"

이 되며

"$P_1 \vee P_2 \vee \cdots \vee P_n \rightarrow Q_1 \wedge Q_2 \wedge \cdots \wedge Q_n$"

과 같은 형태이다. 여기서 P의 갯수가 1개 이하일 때 그를 horn clause 라고 부른다.

위의 한 clause는 여러개의 horn clause로 변형시킬 수 있다.

$$P_1 \leftarrow Q_1 \wedge Q_2 \wedge \cdots \wedge Q_m$$
$$P_2 \leftarrow Q_1 \wedge Q_2 \wedge \cdots \wedge Q_m$$
$$\vdots$$
$$P_n \leftarrow Q_1 \wedge Q_2 \wedge \cdots \wedge Q_n$$

이 모양은 prolog의 형태와 같음을 알아 차릴 수 있겠다. 이렇게 horn clause로 변형하는 이유는, clausal form에 대한 추론 방식엔 적용된 정책에 따라 여러가지 resolution 방법이 있는데 효율이 떨어지든지 아니면 완전성이 없다. 그런데 horn clause에 대해서는 효율도 좋으면서 완전성도 있는 resolution이 존재한다. 즉 SLD resolution[9] 으로 linear-input resolution[8] 과 같다. Prolog[7]는 horn clause에 약간의 제어기능을 포함시킨 언어이다.

2. 단일화(Unification) [8]

한정된 변수를 가진 공식의 정리를 증명하기 위해서는 대개 어떤 부속 표현들을 비교 선택하는 작업이 필요하다. 예를 들어 $(\forall X)[W1(X) = W2(X)]$ 와 W1(A) 로부터 W2(A)를 유도하기 위해 modus ponens를 이용할 때, W1(A) 와 W1(X)가 동일한 것으로 만들기 위해 "X에 A를 치환" 하면 된다. 그 치환을 구하는 과정을 단일화라고 한다.

3. 비교 흡수 부정(Resolution Refutation) 추론 방법

논리의 WFF(well formed formular) 의 집합 S에 어떤 WFF X가 논리적으로 뒤따르는 것을 증명하기 위해서는, 먼저 집합 S에 X 부정인 $\sim$X를 첨가하여 집합 P=SU{$\sim$X} 가 서로 모순이 되는가를 이끌어냄으로서 X가 S와 모순이 안됨을 아는 방식을 사용하는데 그 방법을 비교 흡수 부정 방식[8]이라고 한다.

이 방법의 설명에 앞서 비교 흡수절(resolvent)에 대하여 알아보자. 두개의 절 $P_1 \vee P_2 \vee \cdots \vee P_n$과 $\sim P_1 \vee Q_2 \vee \cdots \vee Q_m$이 있고 P_i 와 Q_j 를 모두 다른 것일때 그 두 절로부터 합법적인 절인 $P_1 \vee \sim P_1 \vee P_2 \vee \cdots \vee P_n \vee Q_2 \vee \cdots \vee Q_m$이 가능한데 보수쌍(complementary pair) 인 $P_1 \vee \sim P_1$ 은 항상 참(true)이므로 제거할 수 있다. 따라서 최종 비교 흡수절은 $P_2 \vee P_3 \vee \cdots \vee P_n \vee Q_2 \vee \cdots \vee Q_n$이 된다. P와 $\sim$P의 비교 흡수절은 $P_2 \vee P_3 \vee \cdots \vee P_n \vee Q_2 \vee \cdots \vee Q_n$이 된다. P 와 $\sim$P의 비교 흡수절은 'Nil' 로서 모순을 나타낸다. 즉 P도 맞고 $\sim$P도 맞는 상황은 모순이기 때문이다. P $\rightarrow$ Q는 $\sim$P$\vee$Q와 같은데 P와 P$\rightarrow$Q의 비교 흡수절은

Q가 됨을 알 수 있는데 이를 모더스 포넨스 (modus ponens) 라고 한다.

그러면 논리를 증명하는 방식인 비교 흡수 부정 방법을 예를 들어 알아보자.

WFF 집합 S가 다음과 같고

(1) ~R(X) ∨L(X)

(2) ~D(Y) ∨~L(Y)

(3) D(A)

(4) I(A)

증명해야할 목표가 I(Z)∧~R(Z) 이면, 목표를 부정한 ~(I(Z)∧~R(Z)) 과 S가 모순이 되는가를 봄으로써 목표가 S와 모순이 되지 않음을 증명한다. 목표의 부정을 ~I(Z) ∨R(Z) 이 되며 비교 흡수를 이용하여 Nil을 이끌어 내보자.

그림 1에서 Nil이 이끌어내진 과정이 목표를 증명한 과정이 되며, 변수의 값이 단일화 과정중 대치되어감을 유의하자.

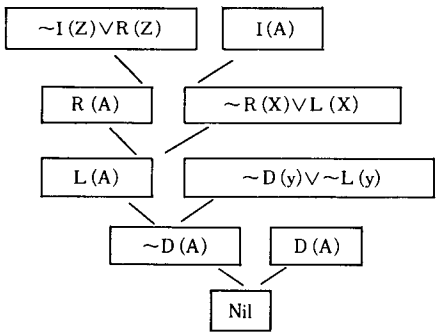

그림 1. 비교흡수 부정 트리

Ⅲ. 인공 지능형 머신과 논리 머신

인공 지능형 머신은 여러 가지 특성을 갖고 있다.

심볼을 주로 다루며, 계산 과정이 비결정적(non deterministic) 하고 병렬 및 분산 처리가 가능하며, 지식을 다루는데 알맞고, 항상 확장 가능한 시스템 (open system) 이어야 한다.

인공 지능형 머신에 관계된 과제를 여러 나라에서 각각 추진하고 있는데 일본의 FGCS, 미국의 Strategic Computing, 유럽의 Esprit, 영국의 Alvey 과제들이다.

AI 머신은 그림 2와 같이 여러 수준의 층으로 이

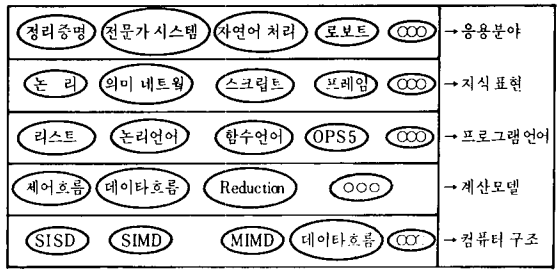

그림 2. 인공지능머신의 계층 및 기법

루어지며 각 층은 여러가지 기법들이 있다.[10]

논리머신은 인공지능머신의 일종으로, 지식 표현의 한 방법인 논리를 빠르게 추론해 주는 기계를 말한다. 논리언어로는 prolog나 병렬논리언어등이 있으며, 계산 모델과 컴퓨터 구조를 어떻게 할 것인가를 결정함은 매우 중요하다.

논리머신에는 추론을 순차적으로 하는 언어인 프롤로그를 빨리 수행하게 하는 프롤로그머신과, 병렬로 추론을 해주는 병렬논리머신이 있다.

논리언어에는 병렬성이 풍부하지만 순차적으로 수행하게 하는 언어인 프롤로그가 있다. 순차적으로 추론하게 하는 이유는 이 언어를 프로세서가 하나인 기존의 폰노이만형 컴퓨터에서 돌아가게 하기 위해서이다.

프롤로그를 전문적으로 빨리 돌리는 기계를 설계할 때 일반적인 특성은 다음과 같다.

(1) WAM (Warren abstract machine)[11] 에 바탕을 두고 있다.

(2) 스택을 이용한다.

(3) 주 프로세서로 사용하던가 보조-프로세서로 사용한다.

(4) 속도를 높이는데 다음의 방법을 이용한다.

- 마이크로 프로그래밍
- 메모리 캐싱
- 파이프 라이닝 기법
- 주문형 VLSI 사용

프롤로그 머신에는 SRI의 'Pipeline Prolog Machine', Berkeley 대학의 PLM, ICOT의 PSI, Hitachi사의 IPP, Kobe대학의 PEK, NEC의 CHI 등이 있으며, 100K LIPS ~500K LIPS 의 속도를 갖는다. 여기서 'LIPS란 lcgical inference per second'의 약자로써 초당 추론하는 횟수를 표시하는 단위이다.

Ⅳ. 논리의 병렬추론머신

병렬추론머신을 만들기 위해서는 결정해야 할 요소가 많다.

(1) 어떤 논리언어를 목표(target) 언어로 할 것인가.

(2) 어떤 병렬성을 추구할 것인가.

(3) 수행모델(execution model) 은 무엇인가.

(4) 컴퓨터 설계에 필요한 논쟁점들을 어떻게 선택할 것인가.

위의 각 문제들을 각각 살펴보자.

1. 목표언어

병렬추론머신을 만들기 위해서는 그 머신이 어떤 논리언어를 병렬추론하는가가 매우 중요하다. 왜냐하면 그 목표언어가 무엇이냐에 따라서 추론하는 방법과 그 복잡성이 매우 달라지기 때문이다. 대표적인 병렬추론을 위한 논리언어는 다음과 같다.

1) Horn clause

순수한 논리형태인 호온-절은 그대로 논리언어로 사용될 수 있다. 이 언어는 아무런 제약을 받지 않기에 표현력과 병렬성은 강하나, 수행모델을 만들기엔 방대하고 어렵다. 다음의 다른 언어는 이 언어를 변형하고 제약하여 수행방법을 구체화 및 효율화한 것들이다.

2) Parallel prolog[12]

논리언어의 AND 병렬성과 OR 병렬성에서, 프로그래머에게 그 병렬성을 표현할 수 있게 하는 언어이다. 즉 프로그램 작성시 AND 및 OR 병렬성을 이용하는 것이 문제풀이에 효율적이면 순차적으로 하도록 표시한다. 그 정보는 결국 제어 정보인데 그 제어정보는 각 목표(goal) 사이에 혹은 절(clause) 사이에 표시할 수 있다.

3) Parlog[13]

Parallel prolog 보다 더 많은 제어정보를 줄 수 있는 언어로 스트림-병렬성(stream parallelism)도 가능하게 제어해 줄 수 있다. 즉 어떤 고울의 어떤 변수는 값을 만들어 주는(generator) 한편 다른 고울의 그 변수는 그 값을 받아주는 변수 즉 consumer라는 것도 정해줄 수 있다. 또한 완전한 답을 generator에서 구한 다음 consumer로 넘기지 못하는 상황에서, 답을 구한 일부분만 consumer로 넘길 수 있게 함으로써 병렬성을 더 증가시킬 수 있도록 제어할 수 있는 언어이다.

4) Concurrent prolog와 flat concurrent prolog[18]

하나의 답만 필요로 할 경우 답이 구해져 있는데 다른 답들을 구하기 위해 OR 병렬가지를 추론할 필요는 없다. Concurrent prolog는 위의 다른 언어와는 다르게 하나의 답을 효율적으로 얻을 수 있게 만든 언어이다. 따라서 OR 병렬성을 commit할 때, 즉 하나의 답이 얻어질 길이 마련이 되는 때, 없앰으로써 효율을 증가시킨다. Flat concurrent prolog는 commit 되기 전까지도 좀더 효율적으로 수행할 수 있도록 concurrent prolog에 제약을 가한 언어로 매우 수행 효율이 좋다. 그러나 논리언어의 순수성을 파괴하고 제어정보가 너무 많이 들어 간다고 비난을 받기도 한다. Guarded horn clause 도 이 프롤로그와 비슷하다.

5) 확장된 논리언어

논리언어를 여러가지로 프로그램하기에 표현력이 강하도록 만들거나 기능을 다양하도록 확장 및 변형을 하는 언어들로서 여러가지가 있다. 좀더 제어정보를 많이 넣음으로써 효율을 높이는 언어들과, 함수언어와 같이 다른 종류의 언어특성을 논리 언어의 장점과 결합하도록 시도한 언어들도 여러가지 등장했다.

2. 논리언어의 병렬성

논리언어에는 병렬성이 풍부하다.[6,14] 기존의 언어는 병렬성이 약하여 컴퓨터 구조가 아무리 병렬성이 있어도 프로그램 언어가 병렬성이 없어 순차적으로 수행시킬 수 밖에 없었던 상황을 고려하면 논리언어의 병렬성은 매우 높이 평가해야 한다. 논리언어의 대표적인 병렬성은 AND 병렬성과 OR 병렬성이며, 부속적인 병렬성으로는 단일화(unification) 병렬성[3]과 스트림(stream) 병렬성이다.

3. 수행 모델

목표언어를 어떻게 수행하여 어떤 의미를 줄 것인가를 높은 수준에서 기술해 놓은 것이 수행모델이다. 수행모델은 어떤 병렬성을 구현할 것인지를 결정해 주므로 목표언어와 관련이 깊다. 대표적인 모델로 AND/OR 프로세스 모델,[14] goal rewriting 모델[15] 및 데이타 흐름모델[1,2,12,16,17] 등이 있다.

1) AND/OR 프로세스 모델

AND 병렬성과 OR 병렬성을 함께 추구하는 모델로서, AND/OR 트리의 각 프로세스들이 메시지의 교환을 통하여 논리언어의 답을 구하게 하는 방법이다. 가장 근본이 되었던 모델로서, 너무 많은 프로세스를 만들어 내는 단점이 있으나 병렬성이 무한하다는 장점이 있다.

2) Goal rewriting 모델

고울을 하나씩 해당되는 절(clause)들을 이용하여 부고울(subgoals)로 대치해 나가는 방식으로 OR 병렬성만 구현하는 모델이다. AND 병렬성을 취급하지 않으므로 부고울들 간의 공유변수 관리가 쉬워진다.

3) 데이타 흐름 모델

데이타흐름 컴퓨터에서 수행할 수 있도록 만든 모델로서, 논리프로그램을 수행할 수 있는 데이타흐름 그라프가 가능함을 보여준다. AND/OR 프로세스 모델을 데이타흐름 그라프를 변형한 모델이 있고, goal rewriting 모델을 데이타흐름 그라프로 바꾼 모델도 있다. 데이타흐름 모델은 병렬성이 많은 컴퓨터 구조와 병렬성이 많은 논리언어를 연결해 보고자 하는 하나의 시도이다.

4. 컴퓨터 구조 설계의 논쟁점

수행모델을 구현할 때 컴퓨터 구조상 결정해야할 변수가 많이 있다. 거기에는 논리 프로그램의 수행모델과 밀접한 관계가 있는 부분과, 컴퓨터 구조상 언제나 문제가 되는 안건들로 나뉘어진다. 다음에 나타나는 처음 3가지는 전자에 해당되는 문제이고 그 나머지는 후자에 해당된다.

1) 환경복사 혹은 공유문제(environment coding or sharing)

OR 병렬성을 구현할 때 변수의 결합(binding) 환경을 각자 따로 갖느냐 공유하느냐의 문제이다. 복사하는 경우는 서로 독립적이 되는 장점과 derefencing하는 시간이 작아진다는 장점이 있으나 기억장치가 많이들고 복사하는 시간이 든다는 단점이 있다.

2) Structure 복사 및 공유

논리 프로그램의 기본적인 데이타 구조인 스트럭춰를 공유하고 그 변수값만 따로 다루는 방법과 스트럭춰를 복사하여 쓰는 방법이 있다. 복사하면 메모리를 많이 차지하는 단점이 있지만 값이 산재되어 있지 않고 모여 있으므로 reference 하기가 쉽다.

3) 프로그램 복사 및 공유

많은 프로세서 메모리 장치(PU)가 연결망(interconection network)에 연결되어 있는 구조에서 프로그램을 모든 PU에 복사해 놓느냐 공통 메모리에 두어 공유하느냐의 문제이다.

4) 공통메모리의 존재유무

공통으로 사용할 메모리가 있느냐 없느냐의 문제로 공통메모리가 있으면 제어 및 통신이 간단한 반면 공통메모리의 과다한 접근이 성능을 떨어뜨릴 수도 있다.

5) 연결망(interconnection network)의 구조

어떤 모양의 연결망을 사용하느냐는 수행 모델의 성격 및 풀어야 할 문제의 분야에 따라 영향을 받는다.

5. 병렬논리머신의 예

병렬논리머신은 아직 실험단계의 제작수준에 머물러 있다. 근본적으로 이상적인 병렬프로세싱 시스템을 만들기란 어렵기 때문이다. 여러가지 실험제작 중 다음의 것들은 많이 알려져 있다.

1) PIM/R(parlled inference machine/reduction)

일본의 ICOT에서 제작하였다. Goal rewriting(혹은 reduction) 모델을 채택하였고 concurrent prolog가 목표언어다. 추론모듈과 단일화 장치 및 구조적 메모리 모듈로 이루어진다. 또한 2차원의 Mesh 구조의 연결망을 채용하고 있다.

2) PIM/D

데이타 흐름 모델을 수행 모델로 하며 PIM/R과 마찬가지로 ICOT 작품이다. GHC(guarded horn-clause)를 목표언어로 하고 있다.

3) PIE

도쿄대학 작품으로 goal rewriting 모델이며, GHC를 목표 언어로 한다. 부하 형평 문제도 다루었다.

4) BAGOF

스웨덴의 Haridi 작품으로 OR 병렬성만 추구하며 공통 메모리를 가정하고 설계한 종이 머신이다.

5) ZMOB

미국의 매리랜드 대학 작품으로 256개의 Z-80마이크로 프로세서를 링으로 연결한 구조로 공통 메모리는 없다. 동작 중이다.

6. 한국에서의 논리머신

우리나라에서는 1984년경부터 논리 프로그래밍과 논리머신에 관심을 갖어 왔는데 자료 수집과 이론을 이해하는데 시간이 필요했으며 현재는 약간의 결과가 나오게 되었다. 우선 인력이 과학기술원 전산과를 중심으로 양성되었고 연구팀이 만들어져 있다. 현재 양성된 작품은 없으나 함수 언어와 논리 언어를 결합한 언어를 설계하는 사람이 있고, 수행 모델을 세우는 사람도 있다.[1,2,3]

또한 프롤로그해석기와 프롤로그 머신도 실험제작되었다. 병렬수행때의 back tracking에 관한 문제와 프로세스들을 프로세서에 배당하고 부하를 조절하는 알고리즘도 개발되었다.[4] (여기에 언급되지 못한 많은 논문과 projects 도 있음.)

V. 결 론

논리언어는, 인공지능 시스템에서 필요한 기능인 지식의 표현과 그 지식을 처리하는 좋은 방법을 갖고 있으므로, 지식처리에 매우 중요한 수단이다. 논리언어가 어떻게 구성되고 어떻게 추론되는가를 알아 봤으며, 논리언어를 빨리 수행해 주는 기계에 대하여 살펴보았다. 순차적 논리머신인 prolog 머신은 이제 실용적인 수준에 와 있는 반면 병렬논리머신은 아직도 연구되어야 할 많은 문제들이있어 실험적 기계들만 나왔음을 알 수 있었다. 우리나라의 처리에서는 논문을 만드는 정도의 경력이 쌓여 있는 상태이며, 시스템을 구현할 정도로 되려면 더 많은 좋은 인력과 조직과 자본에 노력이 필요하다고 하겠다.

參 考 文 獻

[1] Tae-Choong Chung, "A Mode Driven Dataflow Model for AND/OR Parallelism of Logic Programs," Doctoral Dissertation, KAIST, August 1987.

[2] Tae-Choong Chung, Jung-wan Cho, "An AND-Parallel Dataflow Model for Logic Programs Based on the Mode Prediction," Australian Computer Journal, Feb. 1988.

[3] Tae-Choong Chung, Jung-wan Cho, "The effects of mode and lazy execution on unification in dataflow model for logic programs," *1987 IEEE Region 10 Conference*, Aug 1987, Seoul, Korea.

[4] 은성배, 정태충, 맹승렬, 조정완, "AND/OR 프로세스 모델에서 프로세스 할당정책 : LOPT," 정보과학회지, vol. 14, no. 3, 8 월 1987년.

[5] R.A. Kowalski, "Predicate Logic as a Programming Language," *Proc. IFIPS 74*, 1974.

[6] R.A. Kowalski, *Logic for Problem Solving*, Elsevier North-Holland Inc., 1979.

[7] W.F. Clocksin and C.S. Mellish, *Programming in Prolog*, Springer-Verlag, Berlin, New York, 1981.

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

[9] J.W. Lloyd, *Foundation of Logic Programming*, Springer-Verlag, 1984.

[10] Benjamin W. Wah, "Guest editor's introduction: New computers for artificial intelligence processing," *IEEE Computer*, vol. 20, no. 1, pp. 10-18, Jan. 1987.

[11] H.D. Warren, "Implementing Prolog: Compiling Predicate Logic Programs," DAI Research report no. 39, May 1977.

[12] N. Ito, H. Shimizu, M. Kishi, E. Kuno, and K. Rokusawa, "Dataflow based execution mechanisms of parallel and concurrent prolog," *New Generation Computing*, vol. 3, no. 1, pp. 15-42.

[13] K. Clark and S. Gregory, *PARLOG: A Parallel Logic Programming Language*, Research Report DOC83/5, Dept. of Computing, Imperial College of Science and Techology, May, 1983.

[14] J.S. Conery, *The AND/OR Model for Parallel Interpretation of Logic Program*, PhD thesis, Dept. of Information and Computer Science, Univ. of California, Irvine, 1983.

[15] S. Haridi, and A. Ciepielewski, "An OR-Parallel Token Machine," *Proc. Logic Programming Workshop'83*, pp. 537-552, Portugal, 1983.

[16] S. Umeyama, and K. Tamura, "A parallel execution model of logic programs," *10th Symp. on Computer Architecture, ACM*, pp. 349-355, 1983.

[17] Z. Halim, "A data-driven machine for OR-parallel evaluation of logic programs," *New Generation Computing*, vol. 4, no. 1, pp. 5-33, 1986.

[18] E.Y. Shapiro, *A Subset of Concurrent Prolog and Its Interpreter*, Technical Report TR-003, ICOT, January, 1983.

♣ 用 語 解 說 ♣

Fragment(단편)

문서, 프로그램, 루틴의 일부 혹은 문서의 내용을 작은 부분으로 나누는 작업