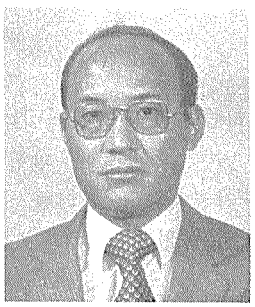

吳 吉 祿
韓國電子通信研究所
컴퓨터연구부장/工 博

인공지능 구조 동향

Ⅰ. 소개

응용 영역의 급진적인 변화로 인하여 순차적이고 결정적인 수치 응용을 위한 오늘날 많은 컴퓨터의 구조적인 특징들은 기호적 처리를 효율적으로 보조하기에 부적합하게 되었다. 인공지능 응용을 위한 컴퓨터의 구조는 오늘날 보통 사용되는 폰 노이만 컴퓨터로는 적절하지 않다. 이는 인공지능 응용들이 다음과 같은 특징들을 갖기 때문이다.

먼저, 인공지능 처리를 위한 문제의 크기가 대체적으로 매우 크다. 가끔 탐색공간은 지수적인 비율로 증가하게 되며 이는 거대한 양의 자료를 다루기 위하여 기억장소는 매우 크고 계산시에 높은 정도의 병렬성을 요구한다.

둘째, 기억장소 접근이 매우 빈번하다. 그러므로 자료 교통량을 감소하기 위하여 지역적으로 기억장소안에 있는 자료를 처리함으로써 자료의 불필요한 전달을 피할 수 있어야 한다. 결국 폰 노이만 컴퓨터의 중앙 제어는 처리기와 기억장소사이에 병목현상을 갖게 되므로 적절하지 않다. 이로써 인공지능 문제들은 병렬적이며 분산되는 처리가 요구되어진다.

세째, 많은 인공지능 알고리즘들은 비 결정적이다. 예를 들어 프로덕션 시스템은 단지 초기상태와 목적과 규칙들만을 두고 제어에서 벗어나게 되며 초기상태에서 목적상태로의 흐름 또한 유일하지 않다. 이로 인하여 동적인 자원할당과 균형적 부담이 주요한 요소가 된다.

네째, 큰 지식베이스를 가짐으로 인한 지식베이스의 관리에 있어 고도의 입출력 처리와 처리기/기억장소 사이의 효율적인 상호연관이 필요하게 되었다.

초기의 인공지능 응용을 보조하는 출발점은 선언어 접근으로써 사용하는 그의 언어이었다. 이는 크게 람다기반과 논리기반 언어로 나눌 수

註 : 본고는 한국전자통신연구소 인공지능 연구실에 근무하고 있는 강현구 연구원이 작성하여 전자통신연구소 내부소식('87. 8)에 실린 것임을 밝힌다.

있으며 최근 연구는 논리와 람다를 통합하려는 움직임을 보인다. 그러나 각 언어는 다른 언어에는 부적합한 구조를 갖는 단점을 갖는다.최근에는 좀더 진보된 언어를 보조하는 병렬 구조를 위한 지침들을 제공한다. 인공지능 응용을 보조하는 다른 인공지능 구조로는 그의 지식 표현과 연관을 갖는다.

이의 접근방법은 선지식 접근이라 불리우는데 지식의 표현방법에 따라 의미 네트워크 구조, 규칙지향 구조, 객체 지향 구조 등으로 나눌 수 있고 또한 최근에는 다중연결 구조를 제공하는 신경계 네트워크 구조 등이 있다. 또한, 여러가지 구조들은 복합지식표현을 보조하기 위하여 설계되어지곤 한다. 또 하나의 접근 방식은 지적 대화 기계로써 음성, 영상, 패턴 인식, 컴퓨터 비전 등을 보조할 수 있는 인공지능 구조가 많이 제공되고 있다.

본고의 Ⅱ장에서는 인공지능 구조의 요구를 논의할 것이며 Ⅲ장에서는 인공지능 구조의 구조층을 간단히 논의하고 언어 지향의 인공지능 구조와 지식 지향의 인공지능 구조를 각 기계를 포함한 비교를 통해 논의할 것이다. 그리고 Ⅳ장에서 결론을 맺는다.

Ⅱ. 인공지능 구조의 요구

1. 기호적 처리

인공지능 응용은 일반적으로 기호 형태의 자료를 처리한다. 하위단계의 인공지능 응용은 기호를 비교하고, 선택하고, 순서화하며, 형식을 부합하고, 합집합, 교집합이나 부정같은 논리집합을 하며, 지식베이스의 문맥처리와 분할처리를 하며, 패턴의 검색과 인식 등의 기본적인 기호적 연산을 한다. 좀더 윗단계의 인공지능 응용은 문, 음성, 그래픽 또는 영상위에서의 기호적 처리를 요구한다.

2. 병렬과 분산 처리

대부분의 인공지능 문제는 매우 복잡하다. 이를 해결하기 위한 방법으로 컴퓨터의 성능을 높이기 위한 방법의 하나가 병렬처리이다. 병렬처리를 위한 구조로써는 MIMD 연산을 보조하는 multiprocessor와 다중 SISD 처리를 보조하는 multicomputer와 SIMD 또는 다중-SIMD 또는 MIMD 형태로 동작하는 multipurpose 컴퓨터로 구분될 수 있다.

3. 비 결정적 계산

많은 인공지능 알고리즘들은 비결정적이다.즉, 가능한 정보를 가지고 수행과 종료에 관한 절차를 계획하는 것이 불가능하다. 이러한 특성은 지식의 부족과 문제의 불완전한 이해로부터 오게 된다. 이러한 결과로써 문제가 해결될때 모든 가능성을 완벽하게 나열하는 것이 필요하게 된다. 이러한 결과 비결정성은 프로덕션 시스템과 규칙기반 시스템을 요구하게 되며 OR-병렬성을 제공한다.

4. 동적 수행

완벽한 지식의 부족과 해결과정의 불확실성으로 인하여 존재하는 자료구조와 함수의 특징이 정의되고 문제가 해결될때 새로운 자료구조와 함수가 만들어진다. 주어진 구조에 대한 최대 크기는 매우 크기 때문에 실제 시간에 따른 필요한 기억장소 공간은 할당이 불가능하게 된다. 결과적으로 기억장소 공간은 동적으로 할당이 되며 문제가 해결될때 할당된다.

5. 지식 관리

지식은 문제해결의 복잡성을 감소시킬 수 있는 매우 중요한 요소이다. 유용한 지식은 탐색을 줄일 수 있게 한다. 그러나 많은 인공지능 문제들은 복잡성이 크기 때문에 유용한 지식의 양은 커진다. 게다가 지식은 모호와 경험에 의한 지식과 불확실한 특성으로 습득된다. 그러므로 지식의 관리, 표현, 취급과 학습은 매우 중요한 문제이다. 인공지능 응용에서 매우 많은 양의 정보가 저장되고 검색되기때문에 커다란 지식베이스는 피할 수가 없다. 보통기억장소를 사용한 구현은 접근하기가 부적당하다. 그러므로 분산지능을 갖는 분산된 기억장소를 사용한 구현은 접근하기가 부적당하다. 그러므로 분산

지능을 갖는 분산된 기억장소 시스템과 패턴 부합과 근사적 탐색의 능력이 요구된다.

6. 개방 시스템

많은 인공지능 응용에서 문제를 해결하는데 필요한 지식은 지식의 근원이해가 고안될때 알 수 없거나 환경이 변화되고 설계 시점에서 예측할 수 없기 때문에 불완전하다. 그러므로 새로운 지식은 쉽게 시스템에 더해질 수 있도록 하는 인공지능 시스템이 개방개념을 가지고 설계되어야 하고 새로운 지식의 계속적인 보완과 습득이 허용되어야 한다.

Ⅲ. 인공지능 구조의 설계

1. 인공지능 구조 계층

인공지능 구조에는 3가지의 계층으로 나눌 수 있다. 〈그림 1〉에서 보는 바와 같이 인공지능 응용에 가깝게는 인공지능 응용 프로그램이 가지는 그의 지식표현 방법에 따라 일본의 FGCS의 인공지능 구조를 이루는 논리, connection machine과 NETL의 구조가 기본적인 표현을 이루는 의미 네트워크, DADO가 그의 기본적인 표현을 갖는 프로덕션 시스템, apiary가 제공하는 프레임, 그리고 ZMOB의 인공지능 구조가 지식표현으로 갖는 순차의 지식표현을 갖는 윗단계를 갖는다.

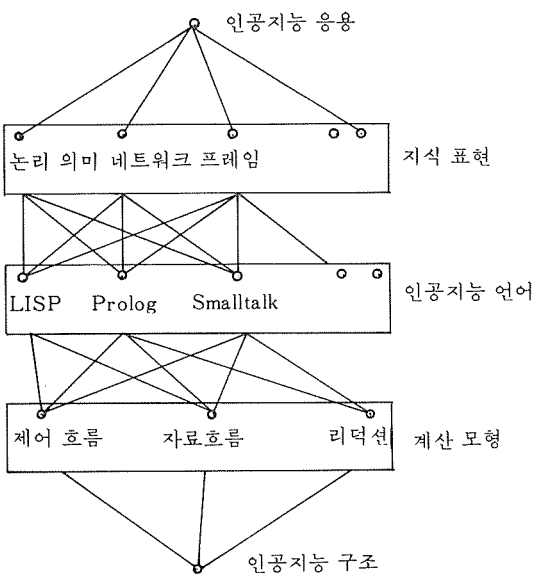

그림 1. 인공지능 구조 계층

그다음 두번째 계층으로써 사용하는 그의 인공지능 언어가 하나의 계층을 형성한다. 이에는 제어흐름의 구조를 갖는 재래식 언어, 대부분의 LISP머시인이 갖는 LISP-like 언어, FGCS의 구조가 갖는 KL0나 KL1 등의 PROLOG-like 언어, 리덕션의 구조에서 이용하는 함수 언어, 그리고 객체지향구조가 제공하는 객체지향 언어 등이 인공지능 계층의 중간 계층을 형성한다. 가장 하위단계에서는 인공지능의 마이크로 단계에서의 계산 모형의 계층으로써 제어흐름, 자료흐름, 리덕션의 계산모형이 하위계층을 형성한다.

가. 인공지능 소프트웨어 구조

1) 인공지능 소프트웨어

초기의 인공지능 소프트웨어들은 오로지 명백한 추론방법에 촛점을 맞추었다. 초기 인공지능 프로그램으로는 Logic Theorem Machine (LTM)과 General Problem Solver (GPS)를 들 수가 있다. 1960년대 중엽이래 인공지능의 능력은 추론방법 보다는 지식이라는 인식을 갖게 되었다. 그후 많은 인공지능 분야의 연구는 지식을 표현하고, 지식을 효율적으로 사용하고, 지식을 습득하는 일련의 연구들이 수행되는 지식공학의 연구가 활발히 이루어지고 있다. 대부분 성공적인 인공지능 프로그램은 지식기반 전문가 시스템이다. 많은 인공지능 시스템들은 전통적인 알고리즘한 방법을 사용하여 해결하기 어려운 많

표 1. 전통적인 프로그래밍과 인공지능 프로그래밍의 차이점

특 징	인공지능 프로그래밍	전통적인 프로그래밍
처리형태	기 호	숫 자
기 법	경험적지식 탐색	알고리즘
해결단계의 정의	명시되지 않음	명확함
답의 정도	만 족	최 적
제어와 자료의 분리도	분리됨	혼합적
지 식	비적확함	적확함
수정정도	잦은 수정	드문 수정

은 문제를 해결할 수 있는 새로운 능력을 제공하고 있다. 이는〈표 1〉에서 보는 바와 같이 전통적 프로그래밍과는 현저히 다른 인공지능 프로그래밍의 특성들로부터 기인한다.

인공지능 프로그램의 기본적인 구성요소는 영역에 대한 정보를 인코딩하는 지식표현, 문제에 연관된 지식을 검색하는 패턴부합, 많은 가능한 해답사이에 만족할 만한 해결을 위한 탐색으로 구성된다.

2) 인공지능 프로그램을 위한 개발환경

인공지능 프로그램에서 또한가지 중요한 의미를 갖는 것이 개발보조이다. 개발보조 기술의 변천은 〈그림2〉에서와 같이 5가지 단계로 분

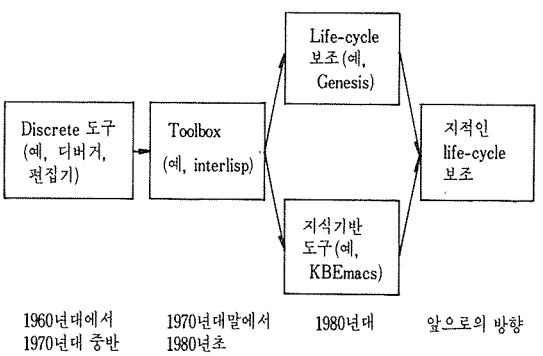

그림 2. 개발환경 변천도

류될 수 있다. 인공지능 프로그램 개발단계는 요구, 분석, 설계, 시제품에 의한 검사, 보수유지, 추적의 단계에 의하여 개발되는 것이 요즈음의 추세이다.

나. 인공지능 언어와 도구

1) 인공지능 언어

인공지능 프로그래밍에 있어서 Fortran, Pascal, Ada와 같은 명령적 언어는 소프트웨어의 복잡성에 적합하지 않고 병렬 작업들을 명시하는 능력이 없기 때문에 적절하지 않다. 그러므로 인공지능 프로그램에 맞는 문제 지향의 프로그램을 할 수 있고, 문제를 해결하는데 절차적이기 보다는 찾는 해결책의 종류를 명시할 수 있으며, 복잡한 문제들을 자동적으로 또 동시적으로 할 수 있는 언어가 필요하다. 바로 이러한 형태의 언어가 선언적 언어이다.

선언적 언어에는 크게 함수적 언어와 논리언어로 나눌 수 있다. 함수적 언어는 람다 계산에 기반을 두고 함수들을 정의한다. 이의 예로써 LISP, FP, HOPE 등을 들 수 있다. 논리언어는 일차논리에 기반을 두고 있다. 이 언어는 관계를 정의하는 절들의 집합으로 구성되는데, 이 언어의 예로서는 대표적으로 Prolog를 들 수 있다. 또한, Prolog와 LISP언어사이에 인터페이스를 제공하는 언어가 존재하는데 이에는 LOGLISP, QLOG, POPLOG, Qute, Lambda PROLOG 등이 있다.

그리고 함수기반과 논리기반이외에 인공지능 언어로써 각광을 받고 있는 언어로써 객체지향 언어가 있으며 대표적 언어가 Smalltalk 언어이다. 최근에는 병렬처리할 수 있는 언어의 개발이 인공지능 언어 분야에서 활발히 연구되고 있으며이의 대표적인 언어가 최초의 병렬 LISP으으로써 C-LISP과 PAR-LISP이 있고 병렬 논리 프로그래밍 언어로써 Concurrent Prolog와 PARLOG 등이 있다.

2) 도구

인공지능 프로그램 개발용 소프트웨어 도구는 주로 전문가 시스템 개발용 패키지들이 대부분이다. 그러나 이같은 패키지들은 거의 대부분의 영역에 적용될 수 있는 기능을 갖고 있기 때문에 그 중요성이 날로 증대하고 있다. 또한 이 같은 패키지는 그 환경이 아주 뛰어나기 때문에 프로그램 개발에 뛰어난 효율을 제공하고 있다.

소프트웨어중 패키지로는 첫째로 지식 공학의 실제 적용을 위한 소프트웨어 도구로 M.1과 보다 광범위하고 전문적인 지식 시스템 개발을 위한 복합적 소프트웨어 종합 패키지로 S.1이 있다. 두번째로는 KEE (Knowledge Engineering Environment) 와 KES (Knowledge Engineering System) 시스템으로 주로 전문가 시스템 개발을 위한 도구지만 그외에 다른 지식 시스템 개발을 위해서도 많은 환경을 제공하고 있다. KEE에서 제공되는 주요 특징으로는 객체 지향적 프로그래밍, 프레임에 입각한 표현, 상속, 프로덕션 규칙과 액티브 값들을 포함하고 있다. 또하나의 도

구는 서로 다른 지식 베이스와 함께 운용되는 추론 엔진으로 아주 많은 범위의 전문가 시스템을 구성할 수 있는 ART(Automated Reasoning Tool)이 있다. 네번째로는 LOOPS로써 개발용 복합적 지식 프로그래밍이다. 이는 객체 지향적 프로그래밍, 절차 프로그래밍, 규칙 지향적 프로그래밍, 데이타 지향적 프로그래밍의 기법을 통합한 시스템이다. 이외에도 KC(Knowledge Craft), Krypoton, MRS, OPS5, DUCK 시스템 등이 있다.

다. 계산 모형

인공지능 구조 계층의 가장 하위단계에서는 제어흐름, 자료흐름, 리덕션의 3가지 종류의 계산모형으로 구분된다. 제어흐름은 기본적으로 순차적 제어흐름을 하게 되며 병렬 제어흐름을 위해서는 FORK-JOIN으로써 수행된다.이는 자료흐름과 제어 흐름으로 분리되며 처리기와 기억장치 사이의 병목현상을 갖는다. 자료흐름은 프로그램의 방향을 갖는 그래프로 나타내며 공유 기억장소를 가지지 않는다. 기본적인 제어흐름은 자료의 흐름에 의존한다. 리덕션은 프로그램이 중첩된 표현에 의해서 작성된다. 값을 주는 참조의 투명성의 특성을 갖는다. 이에는 스트링 리덕션과 그래프 리덕션의 두가지로 구분된다. 이들 세가지 형태의 계산모형의 계산은 선택, 검토, 수행의 연속적인 반복에 의해 이루어지며, 〈표 2〉와 〈표 3〉에서는 이들의 장단점과 이들사이의 차이점을 보여준다.

이들 세가지 형태의 계산모형은 특별한 형태의 언어와 몇가지의 제한에 의하여 적절한 모형을 취하게 된다. 결국 이는 한 모형이 우수하다는 것은 명백하지 않으며 복합적 형태가 요구되고 있다.

표 2. 계산모형의 장단점

	제어흐름	자료흐름	리덕션
장점	• 제어에 의한 순차적 수행 • 쉽게 자료구조와 복잡한 제어를 다룬다.	• 병렬의 정도가 매우 높다 • 처리율이 높다 • 사이드 이펙트가 자유롭다	• 필요한 명령어만 수행한다 • 병렬성의 정도가 높다 • 자료구조를 다루기가 쉽다
단점	• 효율성이 떨어진다 • 프로그래밍하는데 부담이 크다 • 수행시간 에러를 예방하기가 어렵다	• 불필요한 인수를 기다리는 시간이 낭비된다 • 제어에 부담이 크다 • 자료구조를 다루기가 어렵다	• 지역상태를 변화시키는 객체의 대규모공유의 보조를 못한다 • 요구 토큰을 전달하는데 시간이 걸린다

표 3. 계산모형의 비교

제어흐름	자료흐름	리덕션
• 프로그램 계수기에 의하여 명령어를 선택한다	• 모든명령어를 위한 계산원소를 지역적으로 할당한다	• 다른선택을 한다: 가장 안쪽 또는 가장 바깥쪽
• 검사를 하지않는다	• 가용한 자료항목에 기반을 둔 규칙적용	• 다른 적용 규칙 : 충분한 인수들이 수행을 위해 가용할 때까지 인수의 평가를 요구한다
• 다음 수행할 명령어가 즉각 선택되고 공유기억장치의 임의 부분을 변화한다	• 지역 상태를 변화시킨다	• 상황에 따라 명령어를 재기록한다 지역상태를 변화한다

2. 언어 지향 구조

가. LISP 머신

LISP 머신 프로젝트는 1974년에 MIT인공지능 연구실에서 시작되었다. LISP 머신에 대한 상용화는 1980년도에 MIT 인공지능 연구실에서 두 그룹으로 분리되어 각 그룹이 회사를 설립함으로써 이루어졌는데 그 중 하나가 LMI (LISP Machine Inc.)이며 다른 하나가 Symbolics이다. 같은 모델로 TI(Texas Instruments)에서 개발한 EXPLORER가 있다. 또 1974년에 Xerox PARC의 연구원들은 Xerox의 개인용 컴퓨터인 Alto를 위한 INTERLISP의 개발을 시작했다. Alto LISP이라 불리우는 이 제품의 성능은 주기억장치의 부족으로 만족스럽지는 못했다.

1980년에 PARC는 2개의 더 적절한 개인용 컴퓨터를 개발했는데 이것이 Dorado와 Dolphin

이다. 이와는 별개로 새로운 스택구조와 가베지 콜렉션 시스템을 갖춘 고성능의 LISP머신이 일본에서 개발되었는데 이것을 ALPHA라 한다. ALPHA는 주 컴퓨터에 대한 back-end 처리기로써 대형 컴퓨터 보다 더 빠른 속도의 리스트 처리를 시분할로 사용자에게 제공한다. 또한, 현재 UC Berkeley 에서 SPUR라는 프로젝트를 진행중인데 이는 Berkeley RISKII구조에서 약간의 수정을 가한 LISP 마이크로 프로세서이다. 이것은 40-비트 태그 구조이며 LISP 프로그램의 성능 향상을 목표로 삼는다.

1) LISP 머신의 특징

비록 LISP 머신들이 서로 다른 회사에서 제공되어 각기 다른 기능을 지니고 있긴 하지만 제작자에 관계없이 공통으로 갖는 기능이 몇 가지 있다. 많은 인공지능 프로그램들은 특별한 소프트웨어와 하드웨어를 필요로 하는데 LISP머신에서 사용되는 소프트웨어로는 인공지능 프로그래밍언어 LISP과 PROLOG, 강력한 편집 기능을 갖는 Zmacs 편집기 오류 검출 및 수정을 위한 디버거, 전문가 시스템 개발도구로써 ART, KEE, KES, Knowledge craft, LOOPS 등이 있으며 다양한 기능을 갖는 스크린 윈도우 시스템 등이 있다.

LISP 머신에서 발견되는 하드웨어의 기능으로는 첫째, 고속의 프로세서를 제공한다는 점이다. LISP 머신은 기존의 시분할 컴퓨터에 비하여 혼자 사용함으로써 얻는 잇점 이외에도 기호처리의 효율을 극대화 할 수 있게 설계되었는데 이는 인공지능 프로그램은 알고리즘 프로그램에 비하여 되부름의 특성과 탐색작업 때문에 더욱 더 복잡하기 때문이다. 두번째 기능은 인공지능 프로그램은 보다빠른 프로세서 뿐만아니라 더 많은 기억장소를 요구하므로 단일사용자 임에도 불구하고 매우 큰 기억장치를 요구한다. 세번째 기능은 LISP 머신은 Bit-mapped 디스플레이를 사용하여 윈도우를 보다 효과적으로 나타낼 수 있다. 네번째 기능으로는 여러 종류의 키로 구성된 특수한 키보드를 제공하며 마우스를 사용하여 윈도우상에서 텍스트의 블럭을 표시하거나, 그래픽을 만들거나, icon이라고 불리우는 그래픽 표현을 지적함으로써 함수들을 지적하기에 편리하다. 다섯번째 기능으로는 LISP 머신이 단일 사용자로 설계되었지만 다른 컴퓨터 (LISP 머신도 가능)와의 통신도 가능하다.

위와 같은 기능 이외에도 LISP 머신은 다른 워크스테이션과는 달리 인간이 자신들이 소유하고 있는 지식을 이용하여 어떤 사실을 추론해 가는 방법과 유사하게 기호로 표시된 지식과 정보를 처리하여 주어진 문제를 해결 할 수 있는 환경을 제공하는 고성능의 기호처리 워크스테이션으로써 빠른 프로토 타이핑 능력과 높은 생산성을 갖기 때문에 해외의 유명한 인공지능 소프트웨어가 LISP 머신을 이용하여 대량 개발되었으며 현재 많은 인공지능 분야에서 그 진가를 발휘하고 있다. 즉, 기호처리 전용 워크스테이션인 LISP 머신은 기존의 컴퓨터와 비교해 볼 때 전용의 기호처리 프로세서인 LISP 프로세서를 갖고 있으며 많은 인공지능 소프트웨어 개발도구 및 환경을 제공하고 사용자가 사용하기 편리한 인터페이스를 제공한다는 큰 특징을 갖고 있다.

2) 구조적 기능

LISP 머신은 LISP 프로그램의 수행시간 보조와 효율적인 구현을 위하여 다음과 같은 여러가지 기능과 사용자의 요구나 기호에 따라 쉽게 환경을 바꿀 수 있는 많은 매개 변수를 제공한다.

가) 효율적인 리스트 표현

초기의 LISP들에 있어서는 리스트는 두 포인터 리스트 셀의 링크된 리스트로써 표현되었다. 그러나 이러한 표현 방법의 단점은 검색과 접근에 많은 비효율적인 면을 가지고 있다. 이를 하여 표현을 컴팩트하게 하고, 검색과 접근을 효율적으로 하기 위하여 선형 리스트를 갖는 벡터 코드 표현 방법을 씀으로써 시간과 공간적인 면에서 효율성을 가져오게 되었다. 그러나 이또한 크기가 유동적인 경우에 관리하기가 어렵게 되어 있으며 이의 대안으로써 구조화된 표현으로 리스트를 저장하고 검색, 접근한다. 이는 태그된 링크 벡터를 이용하여 리스트 접근 속도를 높이기 위하여 연관기억장치를 사용하기도 한다.

나) 기억장소 관리

가베지 콜렉션은 사용되는 구조에 무관하게 리스트 처리를 위해서 요구된다. 인공지능 구조에서 가베지 콜렉션은 만일 링크드 자료구조가 언어에 사용된다면 매우 중요한 요소의 하나이다. 대부분의 시스템은 증가식 가베지 콜렉션 기능을 가지고 있으며 가베지 콜렉터는 접근 불가능하고 더 이상 쓰이지 않는 저장 셀을 개선한다. 가베지 콜렉터는 다음 두가지 방법중의 하나로 쓰일 수 있다.

-자동 증가식 모드는 연산과 동시에 발생한다. 증가식 가베지 코렉터는 사용자가 설정한 사용 가능한 기억장치의 임계값이 초과 되었을 때 트리거 된다.

-정지후 콜렉트 모드는 사용자가 프로그램에서 명세된 지점 또는 사용자가 이 모드로 가베지 콜렉터를 옮겼을때 발생한다.

다) 마이크로 코드

많은 LISP언어 프리미티브와 시스템 함수가 시스템 성능을 향상시키기 위해서 마이크로 코드로 구현되며, 이들은 다음과 같다.

- 장치 핸들링
- 가상 기억장치 관리
- 기억장치 할당
- 가베지 콜렉션
- LISP 자료 객체 보조
- LISP 언어 커널
- 스크린 드로잉 프리미티브

라) 프로세스

LISP 머신은 사용자가 자신의 프로세스를 생성하고 제어할 수 있게 한다. 여러 연산이 분리된 프로세스에 각각 위치함으로써 동시에 수행될 수 있다. 각 프로세스는 자신의 고유한 프로그램 카운터, 함수 호출에 따른 스택 그리고 같은 특수 변수 바인딩 환경을 지니고 있다. 모든 프로세스는 LISP 객체 집합을 공유하면서,같은 가상 기억공간에서 수행된다. 그리고 특이한점은 공유된 LISP 객체를 통해 프로세스간의 유연한 통신을 허용한다는 점이다. 스케듈러는 활동 프로세스를 관리하고 동시에 수행하면서 서로 통신할 수 있게 해 준다. 프로세스는 플레이버를 정의할 수 있고 이들 프로세스에 많은 연산을 행할 수 있다.

나. Prolog 머신

Prolog 머신은 1982년 일본 ICOT의 제 5 세대 컴퓨터 시스템 프로젝트가 그의 동기가 되었다. 이는 이 프로젝트의 기본언어를 Prolog 로 채택함으로써 많은 연구가 이루어지게 되었다. 제 5 세대 컴퓨터 시스템 프로젝트의 주요 연구분야의 하나인 문제해결과 추론 머신 개발에 있어서 logic 프로그래밍 머신을 개발 목표로 두게 되었다. 주로 거의 자연어에 가까운 정도의 강력한 서술 논리에 기반을 둔 추론과 계산모형을 보조하는데 필요한 구조를 연구하고 개발하는 것이다.

기본적인 기술의 개발은 병렬 시스템의 연구와 특별목적의 머신의 개발에 있다.이 프로젝트에서는 추론 머신으로써 4 가지 기법의 머신 구조를 가지고 있다. 이는 리덕션 기법, 자료흐름 기법, 완벽한 복제 기법, 클로즈 단위 처리 기

표 4 상용화된 LISP machine의 특징 비교

생산자와 모델	소개된해	특 징
Xerox 1100 series	1981	바이트코드 이뮬레이션을 갖는 Interlisp환경을 보조하는 개인적 워크스테이션이다.
Symbolics 3600 series	1983	수행시간에서의 자료형태 검사와, 가베지 콜렉션, 선채취를 하는 명령어 파이프라인을 직접 하드웨어로 수행한다.
Lisp Machine Inc. (LMI), Lambda	1983	빠른 NuBus를 중심으로 하는 모듈화된 확장가능한 다중 처리기이고 : 거의 무한의 가상 기억장치를 제공한다.
Fujitsu ALPHA	1983	프리 변수의 스택주소를 저장하는 캐쉬에 의해 인수화된 가상 스택을 통한 복수 프로세스를 보조하는 하드웨어.
Tektronix 4400 series	1984	Motorola 68010/20 처리기를 탑재한 Lower end 워크스테이션.
Texas Instruments Explorer	1984	전용의 Lisp처리기와 확장가능한 다중처리기를 갖는 32-비트-NuBus 기반의 오픈구조

법이다. 현재 시제품으로써 설계하고 있는 병렬 추론 머신으로써 리덕션 기법에 기반을 두고 있는 PIM-R을 들 수 있으며, 자료흐름 기법에 기반을 두고 있는 PIM-D 병렬추론 머신이 있다. 또한 FGCS의 결과로써 개발된 High-speed Prolog Machine (HPM)이 있다. 이는 큰 기억장치 용량과 단일화, 스택 연산을 위한 특별한 하드웨어를 제공, 고급 스택 제어 명령어를 갖는 컴파일러 지향의 구조, 사이드 이펙트 연산, 다중처리 보조 등의 특징을 제공하고 있다.

1) Prolog 머신의 특징

가) AND/OR 트리 표현

논리 프로그램의 평가는 AND/OR 트리의 탐색으로 볼 수 있다. 여기서 루트는 질의되는 초기문제로 볼 수 있으며, OR-노드는 목적 (또는 부목적)으로 AND 노드는 클로즈로 볼 수 있다. 그리고 하나의 클로즈의 같은 바디에 있는 모든 목적들은 AND 노드의 자식으로 볼 수 있다. 목적 (또는 부목적, OR 노드)과 그의 자식은 같은 헤드를 갖는 클로즈의 비결정적 선택을 나타내며 말단 노드는 단지 그라운드 텀을 실체화할 수 있는 클로즈나 부목적으로써 나타내어진다. 이러한 논리 프로그램의 AND/OR 트리로써의 표현은 병렬성이 뛰어나다.

나) 논리 프로그램의 병렬성

논리기반 인공지능 머신들은 prolog-like 언어로 표현된 프로그램을 직접 수행하는 분해에 기반을 두고 있다. 자료흐름 모형에서 중간 자료흐름 언어는 prolog 프로그램과 하드웨어 사이에서 개입되어진다. 변환기는 수행전에 prolog 프로그램과 자료흐름 그래프사이에서 사상이 요구되어진다.

Prolog 기반 프로그램의 수행은 여러 종류의 병렬성을 이용함으로써 속도를 증진할 수 있다. 논리적으로 AND-ed 클로즈 들의 동시 수행을 AND-병렬성이라고 한다. 만일 두 클로즈가 공통의 변수를 가지고 있으면 두 클로즈가 같은 값을 변수에 성공적으로 바운드하는 단일화가 요구된다. 그러므로 두 클로즈의 단일화가 두 개의 동일한 병행처리를 통해 이루어지면 공통변수가 일치성 있게 바운드 되었는지를 검사해야 한다. 결국 이로 인한 부담이 생기게 된다. 하나의 해결책은 공통변수를 갖지않는 AND-ed 클로즈를 집단화함으로써 이들 집단을 병렬적으로 수행할 수 있다.

이와는 달리 스트림-병렬성이 있는데 이는 공통변수를 갖는 AND-ed 클로즈들은 순차적 AND평가를 증진하기 위하여 생산자-소비자 관계로 연결 지을 수 있다. 이는 필수적으로 거시적 파이프 라인을 포함하고 일치성 검사의 필요성을 제거한다. 대표적으로 PIM-R는 공통변수와 일시중단된 프로세스들의 리스트를 갖는 메시지 흑판의 수단에 의한 스트림-병렬성으로 구현되었다.

또 하나의 병렬성은 OR-병렬성으로써 여러가지의 해결 길을 병행 탐색하게 된다. OR-병렬성을 선호하는 대부분의 머신들은 비결정적이고 복수개의 해결책이 유도될 수 있다. 여기에서는 변수 바인딩의 일치성 검사에 대한 부담이 필요없게 된다. 그러나 제한을 가지지 않는 OR-병렬성은 실제적인 OR-병렬성에서 지수적 증가가 유도될 수 있고 무한 루프에 들어갈 수 있다. 목적 클로즈와 prolog 지식베이스에 있는 클로즈들 사이에 병렬 부합과 변수에 대한 상수값들이 병렬로 실제화되는 단일-병렬성이 있다. PIM과 PIE는 목적 클로즈와 지식베이스 사이에 병렬 부합을 위한 하드웨어가 제공되며 변수의 병렬 실체화를 제공한다.

2) 구조적 기능

Prolog 머신은 일반적으로 다음과 같은 구조적 기능을 가지고 있으며 기호적 처리를 위하여 요구 조건들을 만족하여야 한다.

○단일화

이는 적절한 클로즈를 탐색을 통하여 찾고, 호출자와 호출당하는자 프레디키드의 인수를 채취하며, 인수의 동등성을 검토하는 일련의 과정을 거치게 된다.

○백 트래킹

○변수 호출의 빠른 작성과 제거를 위한 동적 기억장치 할당

○인수를 채취하김 위한 빠른 기억장치 접근

○인수 형태의 빠른 검사를 위한 태그드 자료형

태

○큰 수행환경을 위한 커다란 기억장소 공간

표 5. 설계된 Prolog머신의 특징 비교

머신과 개발자	시스템구조	수행 패러다임: 사용되는 병렬성별	상 태
Parallel Inference Machine(PM). ICOT, 일본.	Loosely coupled 다중처리기	논리(PIM-R)/자료흐름(PIM-D); AND, OR, unification 병렬성	설계중
Parallel Inference Engine(PIE) University of Tokyo	Loosely coupled 다중처리기/다중컴퓨터	논리; OR와 unification 병렬성	설계중
Programmed Logic Machine(PIM). UC Berkely.	특별목적의 스택을 갖는 단일처리기	자료흐름; OR와 unification 병렬성	설계중

3. 지식 지향 구조

지식 지향 구조의 목적은 지식을 표현하고 조작하기 위한 강력한 모형의 효과적인 수행을 제공하기 위한 것이다. 〈표 6〉은 일반적인 지식 표현 방법들과 추론 기법, 그리고 그들과 관계된 대표적인 구조를 보여준다.

표 6 지식표현과 관련된 구조

지식표현	추론기법	머 신
서술 논리	분 해	FGCS
자료 인캡슐레이션	상 속	iAPX 432
프로덕션 시스템	순방향 추론	DADO
프레임	절차성 부착	Apiary
프로시듀어	프로그램 수행	Zmob
의미 네트워크	마커 전달	NETL
신경계 네트워크	에너지 minimization	Boltzmann

가. 의미 네트워크 구조

의미 네트워크에서 노드들은 객체나 개념들을 표현하고 객체나 개념사이의 관계는 두 노드 사이에 링크로 표현한다. 의미 네트워크로부터의 지식 검색은 전체 네트워크를 탐색함으로써 할 수 있다. 만일 원하는 사실이 명백하게 저장되어 있지 않으면 이는 다른 저장된 정보로부터 연역될 수 있다. 그러므로 지식 검색을 위한 프로그램은 집합사이에 논리적 동작, 정렬, 패턴 부합, 의미 상속 네트워크로부터 사실을 연역하는 기본적인 동작을 반복함으로써 수행된다. 속도 증진은 네트워크의 많은 노드들사이에 기본적인 동작들을 동시에 함으로써 달성할 수 있다. 이상적으로는 각 개념(노드)이 하나의 처리기에 할당되고 이들 처리기들 사이에 상호연관은 대응되는 개념들 사이에 관계를 표현하기 위하여 융통성이 있어야 한다. 이상적으로는 프로그램할 수 있는 논리적인 연결을 갖는 많은 처리기 소자를 요구한다. 그래서 토폴로지는 이 문제에 적절하게 구성될 수 있다. 만일 실리콘 영역이 제한되어 있으면 처리기는 단순해야 하고 작은 지역 기억장소를 가지며 상호연관 네트워크는 정규화 해야한다. 그러나 기억장소는 머신을 통해 분산되어 있고 기억장치가 지역 처리기들을 통하여 동시에 많은 장소에서 수정될 수 있으므로 지능적이다. 폰 노이만 기억장치의 기본적인 비효율성은 단지 어떤 주어진 시간에 소수의 위치만이 변화될 수 있다는 것이며 이는 회피되었다. 이들 고려사항들은 CM(Connection Machine) 같은 실제적인 의미 네트워크 처리기 뿐만아니라 NETL과 Thistle machine과 같은 대규모 병렬과 화인-그래인된 구조를 제안했다. SNAP은 의미 네트워크를 처리하기 위한 배열-구조화된 머신이고 이는 VLSI로 설계 되었다. CM의 능력은 셀을 재구성할 수 있는 가상 상호관련 패턴들을 저장하고 매우 많은 셀들을 동일하고 단순한 작동을 동시에 수행할 수 있다는 것이다. CM은 또한 의미 네트워크 프레임 월크로 표현되지는 않지만 객체 인식과 비구조화된 교재검색과 같은 응용에 효과적이다. 그렇지만 이는 많은 탐색과 정렬을 요구한다. 대량 병렬구조를 분류하는 하나의 방법은 소자들 사이에 신호형태를 전달하는 방법이다. 이에는 마커-패싱, 값-패싱, 메시지-패싱의 세가지 종류로 나누어진다. 메시지-패싱 시스템은 가장 강력하고 가장 복잡하다. 이는 임의 복잡성의 메시지를 전달하고 이들 메시지 위에서 복잡한 동작들을 수행한다. 이 일반화의 댓가는 개별 계산소자의 복잡성과 높은 의사소통 가격과 기반위에 경쟁과 교통 혼잡의 가능성에 따라 증가한다.

Connection machine은Thinking machine사가 6만 5,536개의 처리기를 연결한 컴퓨터이다. 이는 다른 대규모 병렬 처리기와는 달리 모든 처리기가 인접한 프로세서와 3차원으로 연결될 뿐만 아니라 모든 처리기는 중간에 있는 처리기를 경유하지 않고도 멀리 떨어져 있는 처리기와 초당 30억 비트의 속도로 직접 통신할 수 있는 기능(router)을 갖고 있다. 즉 처리기가 처음에는 그리드, 형태로 연결되어 있지만 처리기간의 연결을 트리형태로 재구성할 수 있다. CM은 모든 처리기가 동일한 프로그램을 수행하되 각각의 프로세서는 서로 다른 하나의 자료 항목을 동시에 처리하는 SIMD의 형태로 구성되어 있다. 이의 처리성능은 범용계산에 있어서는 1 BIPS(1000MIPS)이며 특수한 분야에서는 7 BIPS까지 된다. 종래의 수퍼컴퓨터인 X/MP (Cray research사)보다 유체역학의 시뮬레이션은 4배가 빠르고 10Giga byte의 데이타 베이스 질의는 5배가 빠르다. 그리고 IBM 본체가 6시간에 걸리는 일을 3분만에 해치운다. 영상처리는 VAX11/785(DEC사)보다 최고 1만배까지 빠르다.

NETL은 마커-패싱 시스템으로 이 시스템에서 전달 소자들 사이에 의사소통은 단일-비트 마커의 형태이다. 각 "노드" 소자는 소수의 별개의 마커비트(16)을 저장할 능력이 있으며 저장된 비트위에서 간단한 부울 연산을 수행하고 마커비트는 다른 소자로부터 도착한다. 이 노드들은 외부제어 계산으로부터 순서하에서 노드에서 노드로 마커를 전달하는 "link"라는 하드웨어에 의해 연결된다. 계산의도의 탐색과 연역은 네트워크의 링크를 통하여 노드에서 노드로 마커를 전달 함으로써 수행된다. 마커 패싱 시스템에서 중요한 점은 메시지 교통량으로 인하여 결코 임의의 경쟁이 없다는 것이다. 만일 한순간 한 노드에 같은 마커의 마크와 카피들이 도착하면 이는 단순히 OR'ed로 모이게 된다. 값-패싱 시스템은 연속적인 양의수들 주위에서 전달되고 이들 값위에서 단순한 동작을 수행한다. Thistle은 CMU에서 개발한 16단일 비트 마커들과 4개의 8비트 값을 갖는다. 또한 이는 지식을 위하여 지역 표현을 갖는다. 즉, 각 개념 또는 어써션은 특별한 처리소자와 연결에 상주한다. 만일 하드웨어 소자가 고장이면 대응되는 지식은 잃어버리게 된다. 〈표 7〉은 의미 네트워크를 처리하기 위한 대표적인 인공지능 머신을 요약한다.

나. 규칙 기반 구조

R1과 XSEL같은 전문가 시스템은 OPS5와 같은 특수 언어의 규칙기반 프로덕션 시스템으로써 쓰여졌다. 프로덕션 시스템의 수행을 위한 기본적인 알고리즘은 3단계의 규칙-적용 순환을 통해 이루어 진다. 먼저 모든 규칙의 조건절이 세계의 상태를 표현하는 작업 기억장치와 "부합"된다. 성공적으로 하나의 규칙이 부합되면 제어 전략에 따라서 "선택"된다. 마지막으로 "적용"단계에서 선택된 규칙은 수행되거나 적용되며 작업 기억장치가 갱신된다. 그러나 이는 직렬 형태로 수행되며 매우 느리다. DADO와 Production System Machine(PSM)은 수행을 빠르게 하기 위해 설계된 특별한 구조이다. DADO는 기본적인 인터프로세서 의사소통 네트워크로써 이진 트리를 사용한다. 내부 트리 노드는 프로덕션 시스템 규칙과 작업 기업장치의 원소를 저장하기 위해 사용되며 루트는 성공적인 규칙중 하나를 선택하고 변화되는 결과를 전체 트리에 전달하는데 사용한다. 이는 분산된 형태로 시스템의 상태를 저장하고 병렬로 머신을 수행하며 상태를 갱신한다. 이진트로 토폴로지는 VLSI 기술로 효과적으로 구현될 수 있고 또한 루트로부터 많은 수의 수령자들에게 전달을 쉽게 다룰 수 있기 때문이다. 프로덕션 시스템의 수행은 각 노드는 단지 그의 선조들과 교신하며 이진 트로 병목현상을 피할 수 있다. 프로덕션 시스템은 속도가 느리기 때문에 속도를 향상시키기 위해서는 그의 자료구조를 변화 시키거나 일반적인 경우를 효과적으로 다루기 위한 특별한 코드를 넣거나 LSH부분의 컴파일 방법을 모색함으로써 달성할 수 있다. OPS83의 경우에는 컴파일 코드를 간략화시킴으로써 속도를 향상시키고 있다. 병렬 머신에서는 프로덕션 해석기의 부합, 충돌 해결, 적용의 세 동

작을 파이프라인 되어지도록 설계하고 있다. 또한 병렬성을 위하여 프로덕션 단계에서는 그룹핑을 함으로써 프로세스들 사이에 의사소통이 요구되지 않도록 하며, 조건단계에서는 조건마다 처리기를 둠으로써 속도증진을 하며, 적용단계에서는 작업 기업장치의 변환이 병렬로 수행될 수 있게 한다. 그러나 이는 부가적으로 동기화의 필요성이 요구된다. 그러나 XSEL, PTRANS, DAA등의 전문가 시스템은 처리기의 수가 무한히 증가하더라도 그의 속도정진에는 제한이 있다. RISC는 CISC보다 2내지 4배의 속도를 증진시킬 수 있다. 그러므로 대량의 병렬 접근방법 보다는 배우 간단한 처리기의 적절성이 요구된다. DAD02는 SIMD/MIMD/다중-SIMD모드에서 동작되는 1,023개의 이진트리의 구조화된 다중처리 병렬 머신이다. 이는 중위 그레인의 병렬 머신이다. DADO 2가 제공하는 언어를 보면 C나 Lisp의 병렬 버전인 PL/M의 수퍼셑이다.

다. 객체 기반 구조

객체 지향 프로그래밍은 자료의 인캡슐레이션에 기반을 두며 모든 절차는 객체라는 일정한 형태로 이를 다룬다. 객체는 메시지를 보냄으로써 다른 객체와 상호작용을 한다. smalltalk 80같은 언어는 변수의 형태 명세를 요구하지 않으며 on the fly 서브루틴을 링크할 수 있으며 메시지 다이나믹하게 자료구조를 변화할 수 있다. 이 언어의 빠른 구현을 위해서는 고급 명령어 집합과 동적 자료 타이핑과 자주적이고 과중한 절차 호출을 위하여 그리고 객체지향의 기억장소관리를 위하여 효과적인 하드웨어 보조가 필요하다. 이는 LISP 머신의 요구와 비슷하며 자료 태그의 병렬 검사, 스택 지향의 수행, 빠른 트래핑, 절차 호출을 위한 중복된 "쉐도우" 레지스터 집합의 특성을 갖는다. 객체는 자주 만들어지고 수정되며 사라져야 된다. 그러므로 가상 기억장치에 상주하고 동적인 기억장소 할당과 효과적인 가베지 콜렉션을 갖는 큰 가상 기억장치 공간이 제공된다. 또한 객체 지향 기억장소는 객체 보호를 위한 설계가 요구되며 기억장치로부터 객체의 빠른 검색 기법이 요구된다.

처음 상품화된 객체 지향 구조는 Intel iAPX 432이며 이는 CISC 기반의 다중처리기이다. 대조적으로 SOAR 프로젝트는 객체를 다루고 복잡한 동작을 제공하는 객체 지향 프로그래밍 환경에서 RISC를 취한다. 또한 Dragon은 RISC의 접근방식이며 Xerox Palo Alto연구소에서 설계된 32비트 수퍼 워크스테이션이다. 이는 10개의 동일한 처리기를 사용하는 tightly coupled 시스템이며 일반 버스를 통하여 주기억장치를 접근한다. 버스 인터페이스는 전체 연관 명령어와 자료 캐쉬를 통하여 이루어진다. 그래픽과 시뮬레이션 응용을 위해 부동 소수점 가속기가

표 7 의미 네트워크를 위한 인공지능 머신

시스템과 개발자	구조적 특징	보조되는 언어	응용분야
Connection Machine, Thinking Machines, Inc., Cambridge, Mass. (marketed, 1986).	65,536개의 처리기와 hypercube와 같이 상호 연결된 기억장치 노드들을 갖는 SIMD 머신	Connection Machine (CmLisp)	의미 네트워크로 표현된 지식처리 : 탐색공간을 검색하는데 화인-그래인 병렬성을 이용할 수 있는 다른 응용
Thistle, Carnegie Mello University, Pittsburgh (conceptual machine)	값과 마크 전달방법을 사용하는 대규모 병렬 화인 그래인	Unkown	의미 네트워크 처리 : low-level vision을 위한 반복적 relaxation.
SNAP, University of Southern California, Los Angeles (conceptual machine)	content-address memory를 갖는 cell들의 2D mesh;associative와 cellular array 처리	microinstruction primitive들의 집합으로 구성되는 고급 명령어	의미네트워크 처리 : production 시스템 : vision을 위한 discrete relaxation

각 처리기에 통합되었다. FAIM 1은 원자적 객체가 논리적 또는 절차적 형태로 표현될 수 있는 OIL이라는 객체 지향언어를 보조하는 효과적인 하드웨어를 제공하도록 설계되었다. 이 프로젝트의 특별한 특징은 많은 함수들이 소프트웨어로부터 하드웨어로 전문화된 기억장치 부시스템에 포함되었다. 〈표 8〉은 객체 지향 인공지능 구조의 특징을 간략히 보여주고 있다.

Ⅳ. 결 론

인공지능 응용들은 기호적이고, 비결정적이며, 동적이고, 병렬과 분산처리를 요구하며, 지식을 관리하기 때문에 전형적인 폰 노이만 컴퓨터로는 처리하기에 적합하지 않다. 인공지능 문제를 해결하기 위해서는 지적인 경험적 지식을 이용하며, 상위 레벨의 메타지식을 보다 잘 이용하며, 기계학습의 기법을 활용한 효율적인 인공지능 알고리즘이 요구된다. 또한 인공지능 소프트웨어 관리 방법에 있어서 효과적인 지식표현방법과 조작, 병행성의 프로그래밍 언어와 컴파일러, 더좋은 소프트웨어도구와 사용자가 사용하기 쉽고 분산된 환경의 개발, 그리고 신뢰성있는 소프트웨어의 개발이 요구된다. 인공지능 구조의 입장에서는 화인-그레인 다중처리 또는 다중컴퓨터에 기반을 둔 고도의 병렬처리가 설계되어야 하며, 병렬 하드웨어 구조와 병렬 인공지능 알고리즘 사이에 더 좋은 부합이 이루어져야 한다. 또한 새로운 기술로써 신경 컴퓨터 등의 기술을 사용할 수 있는 인공지능 구조가 앞으로 이루어져야 할 것이다.

참 고 문 헌

[1] B.W.Wah and G.J.Li, ed., Computers for Artificial Inteligence Application, IEEE Computer Society Press, Washington DC, May 1986

[2] B.W.Wah and G.J.Li, "A Survey of Special Purpose Computer Architectures for AI," ACM SIGART Newsletter, Number 66, Apr. 1986, pp. 28-46.

[3] B.W.Wah, "Guest Editor's Introduction : New Computers for Artificial Intelligence Processing, Computer, Vol.20, No.1, Jan. 1987, pp. 10-15.

[4] K.Hwang, J.Ghosh and R.Chowkwanyun, "Computer Architectures for Artificial Intelligence Processing", Computer, Vol.20, No.1 Jan. 1987, pp. 19-27.

[5] M.F.Deering, "Hardware and Software Architectures for Efficient AI, " Proc. Nat'l Conf. Artificial Intelligence, AAAI, Aug. 1984, pp. 73-78.

표 8 객체 지향 인공지능 구조

머신과 개발자	구조와 보조되는 언어	수행되는 패러다임	상 태
iAPX 432, Intel Corp.	공용 버스를 사용하는 확장 가능한 다중처리기 : Ada와 "puter"객체지향 언어	스택과 3주소 명령어가 둘 다 수행된다.	1981년에 상품화.
SOAR, UC Berkeley.	Smalltalk를 위한 단일 처리기	태그를 사용한 동적 기반의 형검사를 갖는 레지스터 표현평가.	하나의 칩으로 1984년에 구현됨.
FAIM-1, Fairchild, Palo Alto, Calif.	hexagonal 매쉬 토폴리지 : OIL을 수행한다.	병렬 태그검사 하드웨어를 갖는 스택기반의 수행	설계중
Dragon, Xerox, Palo Alto, Calif.	공용 버스를 사용하는 tightly-coupled 다중처리기 : Ceder, Interlisp과 smalltalk를 사용.	스택과 memory-to-memory 명령어 둘다를 수행.	설계중
AI-32, Hitachi, 일본.	마이크로프로그램할 수 있는 단일처리기 : smalltalk 등을 보조한다.	태그된 오퍼란드를 갖는 스택기반.	단일 칩으로 1986년에 구현되었다.

[6] C.V.Ramamoorthy, S.Sheckhar and V.Garh, "Software Development Support for AI Programs," Computer, Vol.20, No. 1, Jan.1987, pp.30-40.

[7] P.C.Trleaven, "The New Generation of Computer Architecture," Proc. Annual Int'l Symp. Computer Architecture, IEEE / ACM, June 1983, pp. 402-409.

[8] A.R.Pleszkun and M.J.Thazhuthaveetil, "The Architecture of Lisp Machines," Computer, Vol.20, No.3, Mar. 1987, pp. 80-87.

[9] H.Hayashi, et al., "ALPHA : A High Performence LISP Machine Equipped with a New Stack Structure and Gabage Collection System," Proc. 10th Int'l Symp. on Computer Architecture, June 1983, pp. 342-348.

[10] G.S.Taylor et al. "Evaluation of the SPUR Lisp Architecture," Proc. 13th Annual Int'l Symp. Computer Architecture, IEEE /ACM, June 1986, pp. 444-45 2

[11] "Design Decisions in SPUR," Computer, Vol. 19, No.10, Nov. 1986, pp. 8 -22.

[12] D.A.Moon, "Architecture of the Symbolics 36 00," Proc. 12th Annual Int'l Symp. Computer Architecture, IEEE / ACM, June 1985,19 85, pp. 76-83.

[13] K.Murakami et al., "Research on Parallel Machine Architecture for Fifth-Generation Computer Systems," Computer, Vol.18, No.6, June 1985, pp. 76-92.

[14] H.Tanaka, "A Parallel Inference Machine," Computer Vol.19, No.5, May 1986, pp. 48-54

[15] R.Nakazaki et al., "Design of a High-speed Prolog Maschine (HPM), "Proc. Int'l Symp. Computer Architecture, IEEE-ACM, June 1985, pp. 191-197.

[16] W.D.Hillis, The Connection Machine, MIT Press, Cambridge, MA, 1985.

[17] S.E.Fahlman and G.EHinton, "Massively Parallel Architectures for AI : NETL. THISTLE and BOLTZMANN Machines," Proc. Nat'l Conf. Artificial Intelligence, AAAI 1983, pp. 109-113.

[18] S.E.Fahlman and G.E.Hinton, "Connectionist Architectures for Artificial Intelligence," Computer, Vol.20, No.1, Jan. 1987, pp. 100-109.

[19] S.J.Stolfo, "Initial Performance of the DAD-02 Prototype," Computer, Vol.20, No.1, Jan. 1987, pp. 75-83.

[20] D.Ungar et al., "Architecture of SOAR:Smalltalk on a RISC," Proc. 11th Int'l Symp. Computer Architecture, IEEE / ACM, June 1984, pp. 188-197.

[21] A.L.Davis and S.V.Robison, "The Architecture ofthe FIM-1 Symbolic Multiprocessing System," Proc. Ninth International Joint Conf. Artifitial Intelligence (IJCAI), Aug. 1985, pp. 32-38.

[22] D.I.Moldovan, MODERN PARALLEL PROCESSING, Dept.of Electrical Engineering-Systems., University of Southern California, Jan. 1986.

[23] 이 성환, 김 진형, "LISP Machine의 현황 및 개발동향," 정보과학회지, 제 4 권 제 3 호, Sep. 1986, pp. 3-14.