韓國通信學會論文誌'87-2 Vol. 12 No. 1

論 文

Content Addressable Memory를 이용한 Production System에서의 Rule 선택에 관한 연구

準會員 白 舜 哲* 正會員 金 在 熹**

A CAM Approach to the Selection of Rules in a Production System

Soon Cheol BAEG*, Jai Hie KIM** *Regular Members*

要 約 많은 production rule (혹은 간단히 production)로부터 현 상태에 만족되는 rule을 빨리 찾아내기 위하여 현재까지는 RAM (Ramdom Access Memory)에 바탕을 둔 필터(filter) 사용등의 여러 방법이 제시되었으나, 본 연구에서는 보다 효율적인 CAM (Content Addressable Memory)의 이용을 제시하고, 이를 위해 CAM의 각 bit에따라, 용도에 따른구분 및 데이타 구조를 설계하고, 이를 컴퓨터 시뮬레이션을 통해 기존 RAM을 사용했을 경우와 비교하였다.

ABSTRACT So far a variety of RAM-based approaches including the Filtering Method have been suggested to shorten the rule selection time in production systems, but this paper presents a somewhat different approach based on the use of CAM. This paper suggests a proper use of CAM bits respect to their characteristics and describes data stsuctures for basic Artificial Intelligence symbolic list processing, and finally compares the simulation results from the CAM-based approach to those from RAM-based approaches.

1. 서 론

Production System〈PS〉은 기본적으로는 지식을 나타내며 많은 rule들과 현 상태를 나타내는 데이타들로 구성된다. Production 시스템은 반복되는 주기(cycle)로써 운영되는데, 매 주기마다 현 상태에 만족되는 rule을 선택하여 이를 수행하고, 그 결과로 데이타의 변화가 생겨 새로운 상태로 가게된다. 이러한 과정을 반복하며 주어진 문제를 해결하게 된다. 이러한 production 시스템은 POST[1]에 의해 최초로 제시된 후에, 프로그램 구성, 지식획득, 지식공학, 자연언어 처리, Cognitive Modelling 등의 전문가 시스템을 포함한 인공지능(AI, Artificial Intelligence)에서 일반 혹은 전

*, **延世大學校電子工學科
Dept. of Electronic Engineering
Yonsei University, Seoul, 120 Korea.
論文番號 : 87-06 (接受 1986. 11. 27)

문적인 지식을 표현하기 위하여 사용되었다.

이 production 시스템은 단편적인 지식을 독립적으로 취급할 수가 있으므로 rule의 추가및 제거가 쉽고, 저장되는 지식이 대부분의 시스템에서 거의 균일한 구조를 가지며, 지식이 법칙의 형태로 자연스럽게 표현되고, 제시된 결론을 얻게 된 과정 또는 근거를 쉽게 설명할 수 있는 장점을 가진다. 반면에 문제해결을 위한 과정을 제어하는 흐름을 추적하기 어려운 점등의 단점이 있으나, 무엇보다 큰 문제점은 현상태에 만족되는 rule을 선택하는데 시간이 많이 걸린다는 점이다. 보통의 실용적인 규모를 가진 시스템의 경우에는 수백개의 production rule을 지니게 되어, 매 주기마다 이 많은 rule들을 하나씩 검토해나가면서 현상태에 적합한 rule을 선택해야 하므로 많은 시간이 소비된다. Rule 선택에 소비되는 시간은 그 시스템의 평가를 좌우한다. Rule선택 시간을 단축하기 위한 연구로는, rule을 종적으로 구성시킨 hierachies 혹은 횡적으로 연결시킨 transition network, patri nets, meta rule등을 이용하는 것과 같이 rule의 구성을 조직화하는 방법[2]과 필터(filter)등을 이용하여 관계된 rule을 쉽게 찾게 하는 방법이[3] 있다. 그러나 이것들은 근본적으로 RAM(Random Access Memory)에 근거를 둔 방식이고 RAM은 인공지능에서 자주 필요로 하는 pattern matching에는 부적합하므로 보다 pattern matching에 적합한 구조의 선택이 필요하게 되었다. 따라서 본 연구에서는 pattern matching에 적합한 CAM을 이용하여 architectures를 구성하고, 이를 이용하여 심볼(symbol)로 구성된 리스트(list)의 처리에 대한 기본방법을 제시하여 production 시스템에서 rule선택의 효율화를 시도하였다.

이를 위해 2장에서는 본 연구에서 사용될 PS의 모델을 설정하고, 3장에서는 RAM을 이용한 기존의 rule선택방법을 차후 CAM방식과의 비교자료로써 제시하며, 4장에서는 CAM을 이용해서 심볼로 구성된 리스트를 처리하기 위한 방법을 고안한다. 이어서 5장에서는 4장의 방법을 이용한 rule선택 방법을 설명하고 6장에서는 CAM방식과 RAM방식을 이용하여 2장의 PS 모델을 실험해본 후 비교한다.

2. Production System의 모델설정

Production 시스템은 일반적으로 다음의 세가지 요소로써 구성되어진다. Produection Memory〈PM〉 Working Memory〈WM〉, 그리고 Interpreter이다.

실제로는 이러한 기본적인 세가지 요소의 구성에 대하여 응용부문에 따라 다소의 변화가 있다[4],[5]. 이 장에서는 차후의 논의를 위하여 다소 특수이지만 보편성을 지닌 Production 시스템의 기본 모델을 제시한다.

[Production Memory]

Production rule로 표현된 지식을 저장하는 곳이 PM이다. 현재 제시되고 있는 모델의 PM은 다음과 같이 구성된다.

(1) Production memory는 production들의 리스트이다.

(2) Production은 condition statement와 action statement로 구성된 리스트로 이들은 심볼 '→'에 의해 구분된다.

(3) Condition statement와 action statement는 각각 condition element〈CE〉와 action element들의 리스트이다.

(4) Condition element는 심볼(변수 혹은 상수), 또는 심볼과 sublist 들로써 구성된 리스트이다.

(5) Action element는 심볼들의 리스트이다. Action element의 처음 심볼은 procedure 이름이며 이 뒤에 있는 심볼들은 이 procedure의 argument들이다.

(6) 변수 이름은 대문자로 시작한다. 그외 모든심볼은 상수로 간주한다.

다음은 이러한 방법으로 구성된 production rule의 예이다.

```
PRODUCTION # 1 : ((start) → ((delete start)
                             (insert continue))
PRODUCTION # 2 : ((apple (banana cabbage)) →
                 ((delete apple) (insert fish)
                 (insert egg)))
PRODUCTION # 3 : ((egg (banana cabbage)) →
                 ((delete cabbage) (insert
```

grape)))
PRODUCTION # 4 : ((grape ((doughunt egg) ×) ×) → ((delete doughnut) (insert end)))
PRODUCTION # 5 : ((continue end) → (stop))

[Working Memory]

현재 상황을 나타내는 데이타들을 저장하는 곳이 WM이다. 이 데이타들에 의해 만족되는 rule이 선택되며 선택된 rule에 의해 새로운 데이타가 첨가되거나 제거된다. 이러한 데이타의 구조는 여러형태로 나타낼 수 있는데, 여기서 제시되고 있는 모델의 WM은 다음과 같이 구성된다.

(1) Working memory는 memory element〈ME〉 들의 리스트로 구성된다.
(2) Memory element는 상수 심볼(constant symbol), 또는 상수 심볼과 함께 변수를 갖지않는 sub-list들로 구성된 리스트이다.

다음은 working memory의 한 예이다.

WM: (((doughnut egg) fish) (banana cabbage) apple start)

[Interpreter]

인터프리터(Interpreter)는 WM내의 데이타들을 이용해서 PM에 있는 rule들에 대한 'select-execute' 주기를 반복 수행하는데, 더이상 만족되는 rule이 없거나, 혹은 규정된 종료조건이 만족되면 수행을 끝낸다.

3. RAM을 이용한 rule의 선택방법및 문제점

여기서는 차후 CAM을 이용한 방식과의 비교를 위해, 기존의 RAM을 이용한 rule선택의 대표적인 방법 두가지를 기술한다.

3-1 Linear Searching에 의한 고전적인 방식 : Type 1

이 방식에서는 한번 'select' 주기를 수행할 때마다, WM내의 ME와 PM내에 있는 모든 production의 CE들을 순차적으로 하나씩 pattern matching해 본다. 그 다음 매칭(matching)되는 CE가 있으면 이것이 속해있는 production을 찾아내어, 차후 실행될 후보가 되는 production들을 모아두는 conflict set에 저장한다. 이 방식에서는 처리속도가 production의 수및memory element의 수에 비례하게 되므로 수백개의 rule을 가진 실용적 시스템에서는 Rule 선택시간이 너무 길게 된다.

3-2 필터를 이용한 Rule의 선택방법 : Type 2

만일WM내의특정한 ME와matching되는 CE가 어떤production에 속해있는가를 미리 알 수만 있다면 불필요한 production을 검토하지 않아도 되므로 'select'주기의 시간이 절약될 것이다. 이를 위한 정보를 시스템이 운영되기 전에 필터에 저장시킴으로써 각 'select' 주기에서 처리해야할 production의 수를 줄일 수 있다[3].

이 필터에서는 primary feature〈PF〉의 개념을 이용한다. PF는 CE가 하나의 리스트인 경우는, 그 리스트의 처음 심볼이되고, CE가 하나의 atom인 경우는 그 atom이 PF가 된다. ME에서의 PF도 비슷하게 정의된다. 이러한 PF는 Predicate Calculus 혹은 Logic에서의 predicate의 확장된 개념이다. 그림 1)은 이 방식에서의 필터와 production momory에 대한 데이타 구조를 나타내고 있다.

그림 1)에서 보이듯이 필터는 PM내에 있는 각각의 PF에 대하여 filter branch를 하나씩 배정, 이를 연결한다. 또 각 filter branch마다 여러 filter leaf가 있어서, 이것이 '어떤 production이 filter branch의 PF와 동일한 PF를 소유하는가'를 알려준다. 따라서 WM에 ME가 삽입되면 이의 PF에 관계된 filter branch를 찾은 다음 이것에 대한 filter leaf를 통해, 이 PF를 포함하는 production을 빨리 찾을 수 있다. 이 필터에 의해 선택된 production들은 p+set에 놓이게 되는데, 이 p+set 은 WM에 의해 현재 만족될 가능성이있는 production들을 포함하게 된다. 즉 p+set 에 있는 production은 conflict set에 포함될 후보가 되는 것이다.

이 방식은 고전적인 방식보다 rule선택이 빠른 장점이 있으나, 다음의 문제점이 있다.

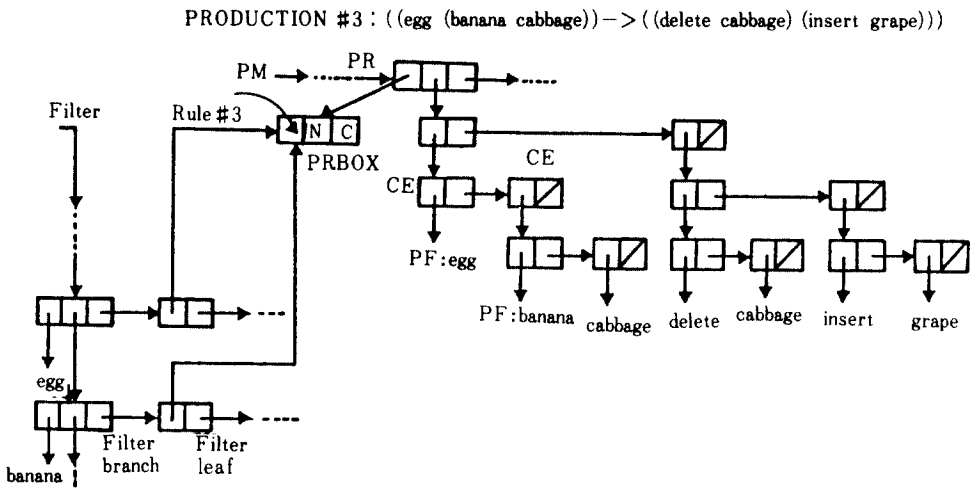


그림 1 Filtering 방식에서의 Production Memory와 Filter의 구조

(1) 필터구성을 위한 예비연산과 메모리 공간 확보가 필요하다.

(2) 시스템이 실행되는 동안에 PM의 변경이 어렵다.

(3) 필터는 오직 p+set만을 얻는데 사용된다. 따라서 p+set에 놓인 production은 고전적인 방식으로 수행된다.

이러한 문제점은 RAM이 본질적으로 pattern matching에 적합하지 않기 때문인데, CAM을 사용하므로써 효율적인 처리가 가능하게 된다.

4. CAM을 이용한 AI Symbolic List Processing

4-1 CAM의 구성

여기서 구상된, CAM에 기반을 둔 구조는 이미 널리 알려진[3], [6] 개념을 이용하여 부분적인 변화및 첨부를 한 것이다.

이 구조는 단순히 정보를 기억하기만 하는 메모리 역활만 하는 것이 아니라 정보를 처리할 수 있는 약간의 기본적인 능력이 있는 것이다. 이러한 CAM 프로세서(processor)의 구조는 그림 2)에 있다. 데이타 항들은 CAM array에 저장되며 다음의 4가지 access mode로 다루어 진다.

1) Multi-read : CAM의 주요 access 형태로 Match라고도 불리운다. 찾고자 하는 데이타 항은 search argument register〈SAR〉에 수록되어, multi-read동작이 수행되면, SAR에 있는 항과 매칭(matching)되는 데이타 항의 CAM에서의 위치가 response store〈RS〉에 기록된다. 여러곳에서 매칭이 일어나면, multiple response resolver〈MRR〉에 의하여 이 중 하나가 선택된다. 만일 데이타의 부분적인 매칭이 필요할 때는 mask register를 이용하여, SAR의 일부분을 mask-out 한다.

2) Multi-write : SAR에 수록된 데이타 항을, 이전의 multi-read 동작에 의해 매칭이 일어난 모든 장소에다 동시에 쓰기 위하여 사용된다. 이때 SAR의 일부분을 역시 mask register를 이용해 mask-out 시킴으로써, mask된 위치의 bit들을 변화가 없게 된다.

3) Single-read : 이 동작에 의해 (부분적인) 매칭이 일어난 곳의 내용을 word output port로 읽어낸다. CAM word(5-1절 참조)는 이런 방법으로 읽혀진다. Word의 address나 포인터(pointer) 부분은 permutation circuit에 의해 internal address port로 보내진다.

4) Single-write : Adderss를 사용하여 데이타 항

을 써 넣기 위해 필요하다.

CAM array는 전통적인 방법인 address selector와 decoder를 통해서도 access가 가능하다. 즉 single-read와 single-write가 바로 RAM을 사용한 때와 같은 형태의 access mode이다. Address selector는 RAM 프로세서로부터 오는 external address port와 CAM 프로세서내에서 오는 internal address port의 선택을 한다. 주지할 것은 CAM 프로세서에서는 아무런 변형없이 RAM 에서의 모든 기능을 할 수 있다는 것이다. Multiple response resolver의 속리속도가, 회로를 다소 많이 첨부함으로써 무시될 정도로 빠르다면[7], CAM의 속도와 RAM의 속도는 거의 같다[6] 따라서 차후 CAM과 RAM에서의 결과 비교는 각각의 access 속도가 거의 비슷하다는 가정하에서 이루어질 것이다.

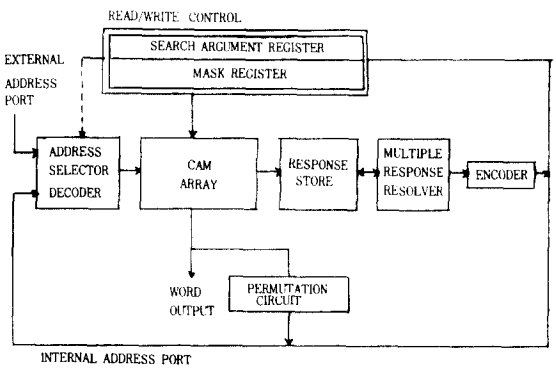


그림 2 CAM Processor의 전체적인 구조.

4-2 CAM을 이용한 기본적인 Symbolic List Processing의 고안

AI에서는 일반적으로 수치데이타 보다는 심볼로 된 리스트를 처리하게 된다. 다음의 5개의 요소를 지닌 리스트에서 X를 지닌 요소를 찾는 문제를 생각하여보자. 각 리스트 요소에는 밑줄이 그어서 있다.

리스트 1 : (Y (X Z) W X (V (W X)))

그림3)은 이러한 리스트를 저장하기 위한 전형적인 데이타 구조를 보여준다. 그림에서 보면, 만일 RAM으로써 이러한 리스트를 저장하였다면, top-down/left-to-right 방식으로 이동하며 처리한다. 이 경우 각 리스트의 모든 요소를 모두 방문해야하는 시간적 낭비를 하게된다. 반면에 CAM을 이용하였다면 bottom-up processing이 가능하여, 즉각적으로 X를 지닌 리스트요소를 찾아낼 수 있다. 즉 CAM 구조의 SAR에 X를 넣고 CAM array중 이와 매칭되는 것이 있는가를 multi-read로써 찾아내어, address A1이 X를 저장하고 있

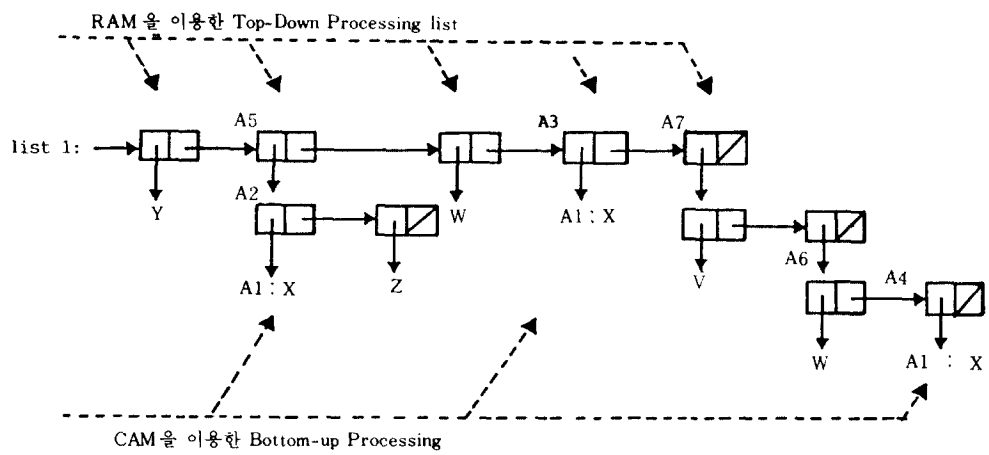


그림 3 기본적인 list traversing을 위한 방법

음을 알아낸다. 또 다시 A1을 SAR에 넣어 address A2가 A1을 가르키고 있음을 알아내고, A2를 SAR에 넣어 A5를 찾아낸다. 이 과정 중에서 mask register를 적절히 사용하여야 한다. 이러한 방식으로 요소 Y나 W를 거치지 않고 address A3과 A7도 빠르게 찾아낼 수 있다.

인공지능에서는 심볼로 구성된 리스트를 이러한 방식으로 빠르게 처리해야할 필요성이 자주 요구된다. 따라서 여기서 고안한 bottom-up processing방법을 잘 이용한다면, PS에서 rule 선택 시간이 단축될 뿐 아니라, rule들중 중복되거나 서로 모순되는 rule들을 빨리 찾아낼 수도 있고, logic, semantic net 그리고 frame 등에서 유용하게 이용될 수 있을 것이다.

5. CAM을 이용한 rule선택 방식 : Type 3

5-1 Production 운용을 위한 CAM-CELL의 배분

Production의 운용을 위한 CAM cell은 CAM array 1 word에 해당한다. CAM array는 한 word가 64bit로 이루어진 12K개의 word들을 포함할 수 있도록 시뮬레이션되었다. 1 word의 용도에 따른 구분은 그림 4)에 나타나 있다.

이 방식에서는 CAM array에 있는 momory cell을 크게 atom cell과 pointer cell로 'Cell Type' bit에 의하여 구분된다.

Atom cell은 atom을 저장하기 위한 것으로 'Atom Type' bit에 의해 상수 심볼(최대 8자로 구성)과 변수 심볼(최대 5자)로 구분된다. Variable cell은 이 변수의 값을 가르키는 포인터를 가진다. 한편 Pointer cell은 'Pointer Cell Type' bit에 의해 primary cell과 ordinary cell로 구분된다. 그중 primary cell은 3개의 포인터를 저장하는데, 2개는 ordinary cell과 마찬가지로 list structure의 구성을 위해 쓰여지게 되나, left-pointer는 특히 PF〈Primary Feature〉를 가르키게 된다. 이러한 PF가 현재의 WM에 의하여 만족되는지의 여부를 candidate bit〈CN〉가 나타내게 된다. Primary cell에만 있는 up-pointer는 production의 root-node를 가르키게 된다. WM에서도 PM에서와 같이 primary cell이 관계된 PF를 가르키게 되나, up-pointer는 관계된 ME의 root-node를 가르키게 된다. 이런 식으로 구성된 PM과 WM이 그림 5)에 주어져 있다. 그밖에 discrimination bit〈DB〉는 environment bit〈EB〉와 함께 리스트 처리를 위하여 쓰이고 garbage bit는 garbage collection을 쉽게 한다.

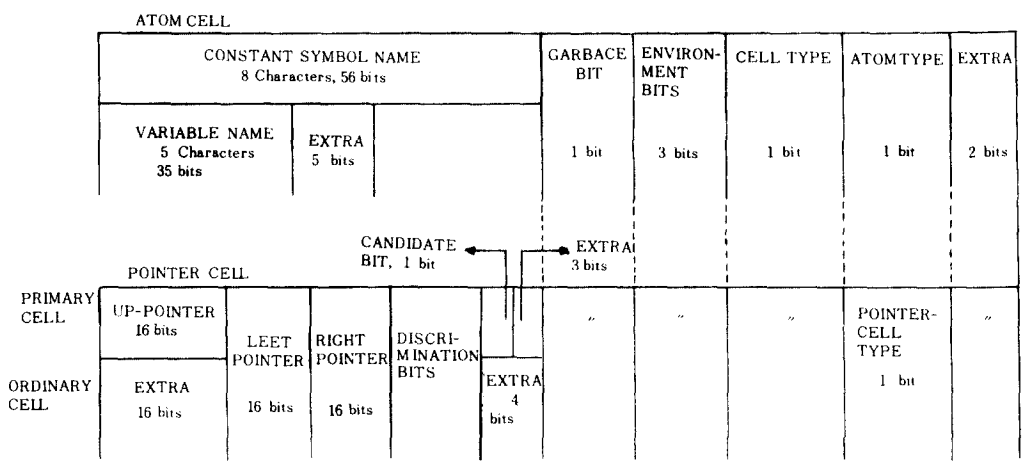


그림 4 Production System을 위한 CAM-cell의 분배

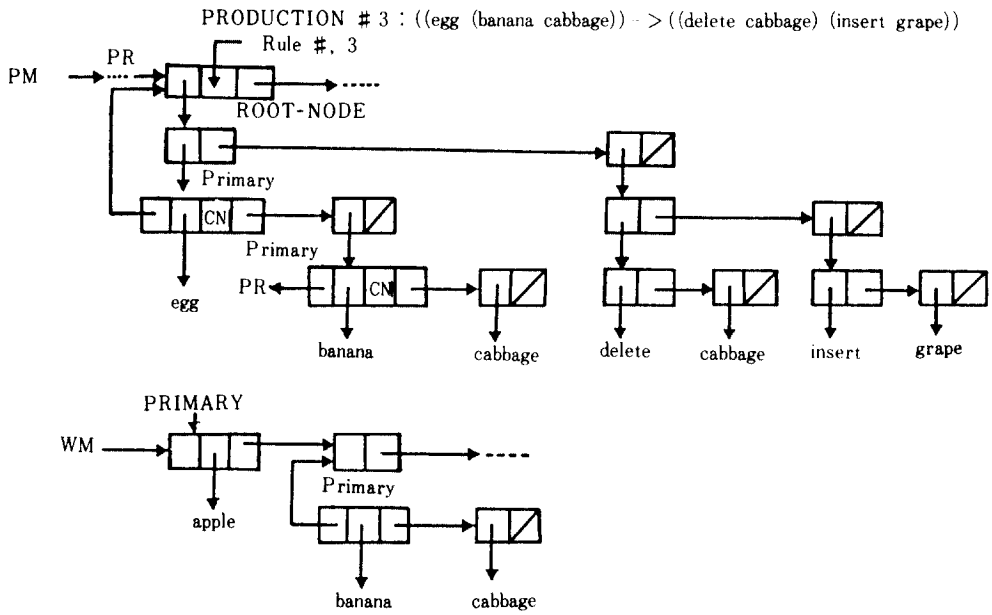


그림 5 CAM 방식에서 Production Memory와 Working Memory

5-2 CAM 방식에서의 Rule 선택의 과정

CAM 방식에서 interpreter에 의한 rule의 'select' cycle은 다음과 같은 순서로 진행된다.

(1) PM으로부터 p+set의 생성

(2) P+set으로부터 conflict set의 선택

(3) Conflict set으로 부터 수행할 rule선택

P+SET의 생성

여기서는 production중 변수로된 PF를 가졌거나, WM에 있는 ME의 PF와 동일한 상수 PF를 가진 production을 찾아서 p+set에 저장한다. p+set을 얻은 과정은 다음과 같다.

(1) CE가 구성될 때, CN bit을 저장한 cell의 left pointer가 가르키는 PF가 상수이면 CN은 'false'로 set되고, PF가 변수이면 (변수 PF는 WM에 의해 항상 만족되므로) CN은 'true'로 set된다. EB(environment bits)는 이 EB를 저장한 cell이 PM를 구성하면 0으로 set되고, WM를 구성하면 1로 set된다. 모든 DB(discrimination bits)는 0으로 set한다.

(2) 각 cycle 동안

1) 어떤 PF X가 working memory에 삽입되면

(a) 만일 이미 X가 WM에 PF로서 이미 존재하면, 아무것도 하지 않는다. (X의 존재여부는 CAM의 multi-read로써 1 clock에 확인할 수 있다.)

(b) 그렇지않으면,

(i) X를 가르키고 CN=false(PF가 상수)이며 EB=0(즉 PM에 있는 cell중에서 선택)인 cell을 찾는다. (이 역시 1 clock에 가능)

(ii) 찾아진(즉 (b)(i)을 만족하는) 모든 primary cell에 대해,

a. CN←true로 set.(즉 이 cell을 포함한 production의 PF(상수)가 WM의 PF와 같으므로, 이 production은 나중에 WM에 의해 완전히 만족될 일차적인 후보가 되는 셈이다.)

b. 이 primary cell을 포함하는 production내에서 CN=false인 primary cell이 더이상 없다면, CN이 true인 primary cell의 up-pointer가 가르키는 root-node의 DB

를 2로 set한다. 따라서 DB=2인 root-node를 갖는 production은 이의 모든 CE의 PF들이 WM에 의하여 만족된다는 것을 알려준다.

2) PF X가 WM에서 제거될 때는

(a) 하나 이상의 X가 아직도 WM에 있다면, 아무 일도 않는다.

(b) 아니면 1) (b)와 같은 작업을 하되, CN=true인 primary cell의 위치를 찾아서, 이러한 cell의 CN을 false로 한뒤, 관계된 production의 root node의 DB를 0으로 한다. (이로써 이 production은 WM에 의해 만족될 가능성은 없어진다.)

(3) 'Select' cycle의 초기에는,

DB=2인 모든 root-node를 찾아 p+set에 넣는다. 이것은 p+set에 저장될 production의수를 N이라면, N번의 multi-read만을 수행하면 된다. 따라서 모든 production을 방문하는 비효율을 제거할 수 있다.

Conflict set의 선택

P+set에 저장된 production은 단지 PF 만이 ME에 만족된다. 따라서 여기서는 p+set에 있는 production의 CE가 WM에 완전히 만족되는지 여부를 검토해야 하는데 이때 고전적인 방법의 비 효율을 피하기 위해 4장에서 제시한 bottom-up processing을 사용할 수 있다. 즉 오직 CE의 PF와 동일한 PF를 갖는 ME들만이 매칭될 자격을 갖게되므로 우선 이런 동일한 PF를 갖는모든 primary cell을 WM에서 찾은다음, 이 cell의 up-pointer를 따라감으로써 매칭해볼 ME들을 빨리 찾아낼 수 있다.

Conflict resolution

본 연구의 production 시스템 모델에서는 conflict set내의 모든 production중 가장 낮은 번호를 갖는 production이 가장 먼저 선택된다.

6. 실험 및 결과

여기에서는 4장에서 논의된 CAM processor를 Pascal programming language로 시뮬레이션을 하였다. CAM 방식에서는 PM과 WM, 중도및 최종 연산 결과는 CAM에 저장하고 인터프리터를 포함한 일반적인 control program은 RAM(host의역할을 담당)에 저장토록 하였으나, CAM의 구조가 일반화되면 control program 역시 CAM에 저장할 수 있다. RAM 방식에서는 CAM의 single-read/write만을 허용함으로써 실재 RAM을 사용하는 것과 같도록 하였다.

실험은 앞에서 제시한 3가지 방식(type 1,2,3)에 대해 여러 다른 test case들을 설정하여 진행하였다. 결과는 table 1과 그림 6)에 있다. 여기서 비교수치는 메모리의 access 횟수이다. 'select' cycle중 p+set를 얻을 때까지의 메모리 access 횟수가 필터 비용이고, conflict set를 얻을 때까

표 1 Production과 Momory Elements의 수에 따른 각 방식의 처리속도 비교

TEST CASE \ TYPE		(RAM) CLASSICAL 방 식 (TYPE 1)	(RAM) FILTER 방 식 (TYPE2)	CAM (TYPE3)
1. BASIC CASE 5PRODUCTIONS 4MEMORY ELEMENTS	FILTER CONDITION	-	126	81
	ACTION	364	94	137
	TOTAL	61	61	81
		425	281	301
2. 50 MORE ACTION PRODUCTIONS	FILTER CONDITION	-	716	81
	ACTION	3414	94	137
	TOTAL	61	61	81
		3475	871	301
3. 100 MORE PRODUCTIONS	FILTER CONDITION	-	1216	81
	ACTION	6464	94	137
	TOTAL	61	61	618
		6525	1371	301
4. 150 MORE PRODUCTION	FILTER CONDITION	-	1716	81
	ACTION	9515	94	137
	TOTAL	61	61	81
		9575	1871	301
5. 10 MORE MEMORY ELEMENT	FILTER CONDITION	-	126	81
	ACTION	968	260	137
	TOTAL	81	81	81
		1049	467	301
6. 20 MORE MEMORY ELEMENTS	FILTER CONDITION	-	126	81
	ACTION	1572	426	137
	TOTAL	101	101	81
		1673	653	301
7. 30 MORE MEMORY ELEMENTS	FILTER CONDITION	-	126	81
	ACTION	2176	592	137
	TOTAL	121	121	81
		2297	839	301

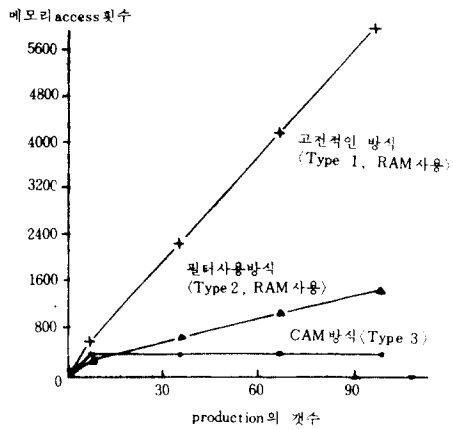


그림 6 여러 개의 Production에 대한 처리 비용(cost) 관련있는 Production은 5개이고 초기의 memory element 갯수는 4개임

지가 condition 비용이며 action 비용은 'execute' cycle 동안에 요구되는 메모리 access 횟수이다. Table 1에서 보듯이 기본 test case의 경우는 2장에서 예를 든 production 5개와 초기 memory element 4개로 구성된 경우다. 이 경우 모든 type의 실험에서 거의 같은 cost가 든다. 한편 WM와 동일한 PF를 갖지 않는 무관한 production을 PM에 첨부한 경우에는 typel 및 type 2 모두에서 비용이 증가했지만 CAM 방식에서는 기본 test와 변함없었다. 따라서 큰 시스템에서는 CAM이 효율적임을 알수 있다. 또한 production의 어느 CE도 만족하지 않는 무관한 memory element를 WM에 첨부한 경우에도 마찬가지 결과를 얻음을 알 수 있다.

7. 결 론

이 논문에서는 PS에서의 본질적인 문제-rule 선택에 시간이 결린다. -를 해결하기 위해 CAM을 이용하여 bottom-up list processing을 제안하고, 이를 이용하여 문제를 해결하였다.

실험에서 알수 있듯이 CAM 방식에서는 전반적인 처리속도는 첨부되는 무관한 production 및 memory element의 수에는 무관하므로 큰 production system의 응용에 적합함을 말해 준다. 작은 시스템의 경우에는 CAM 방식이 필터 방식보다 처리 속도면에 있어서 탁월하다고 말할 수는 없으나, (이는 bottom-up processing이 작은 크기의 리스트에서는 top-down processing보다 비 효율적이므로) 필터를 사용하지 않기 때문에 필터방식에서의 문제점이 해결된다. 즉, 시행중에 새로운 production의 첨가, 제거가 가능하고, filter 구성을 위한 메모리 공간의 확보가 필요없으며, 시스템 실행전의 예비연산이 필요없다. 한편, 기본적인 bottom-up processing에 대한 개념은 logie, semantic net등 여러 다른 지식 표현 방식을 사용한 경우에도 유용할 것이다.

여기서 제시된 PS의 모델은 PF를 근거로 한 것이다. PF는 논리(Logic)에서 사용되는 predicate 보다 일반적이므로 predicate를 사용하여 rule을 구성한 PS는 어떤것이나 CAM 방식에 의해 실현될 수 있다. 그밖에 predicate가 아닌 테이블이나 기타 다른 방법으로 rule이 구성된 PS은 PF에 해당하는 실마리(key word)를 선택하면 **필터**의 방법및 CAM의 방법을 적용시킬 수 있는 것으로 판단된다.

향후 계속될 연구점은 CAM processor를 host로 사용할 것과, 여기서 제시한 단순한 production 모델을 일반화 시켜서 CAM으로 처리하는 문제, 그리고 CAM의 단가를 저하시킬 것과 보다 조직적으로 EB〈Environment Bits〉와 DB〈Discriminant Bits〉를 사용하는 문제등이 남아있다.

본 논문은 한국과학재단의 연구 지원에 의하여 연구된 논문입니다.

參 考 文 獻

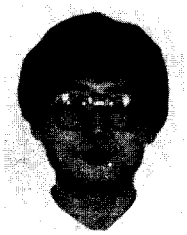
(1) E. POST, 'Formal Reductions of the General Problems', AM. Jnl. math. 65, 1943, p197-268

(2) D. A. Waterman and F. Hayes-Roth,' An Overview of Pattern-Directed Inference Systems', in Pattern-Directed Inference Systems (8).

(3) T. Kohonen, Content Addressable Memories, Springer-Verlag, Berlin, 1980.

(4) R. Lindsay, B. G. Buchanan, E. a. Feigenbaum, and J. Lederberg, DENDRAL, Mc Graw-hill, New York, 1980.

(5) N. J. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Co., Palo Alto, CA, 1980
(6) Intel Component Data Catalog 1978, Intel Corp., 3065 Bovers Avenue, Santa Clara , CA 95051
(7) J. McDermott, A. Newell, and J. Moore, 'The Efficiency of Certain Production System Implementation', in Pattern-Directed Inference Systems (8) p155 - 176



白 舜 哲(Soon Cheol BAEG) 準會員
1963年 2月17日生
1982年 3月～1986年 2月 : 延世大學校電子工學科卒業
1986年 3月～現在 : 延世大學院電子工學科 碩士過程 在學中

金 在 熹(Jai Hie KIM) 正會員
1953年 9月 1日生
1972～1979 : 延世大學校 電子工學科
1980～1982 : 美國Case Western Reserve University)電子工學碩士
1982～1984 : 美國Case Western Reserve University 電子工學博士
1984～現在 : 延世大學校電子工學科助教授
研究分野 : 人工知能