비공유 병렬구조를 이용한 정규화된 재귀규칙에 대한 증명-이론적 의미의 효율적 계산

조 우 현[†] · 이 종 희[††]

요 약

연역데이터베이스는 외연적 데이터베이스인 사실들의 집합과 내포적 데이터베이스인 규칙들의 집합으로 구성된다. 이 규칙들을 계산하기 어렵기 때문에 규칙을 병렬로 계산하기 위한 많은 알고리즘들을 연구해 왔으나 만족스런 결과를 얻지 못하였다. 이 논문에서는 이행적 종속성을 가지는 선형적 재귀 규칙의 증명-이론적 의미를 비공유 병렬구조를 이용하여 효율적으로 계산하는 새로운 방법을 제안한다. 먼저 선형적 재귀규칙을 위한 등가의 표현식이 존재함을 증명하고, 이 표현식을 근거하여 선형적 재귀규칙을 계산하기 위한 알고리즘을 고안하며, 마지막으로 제안된 알고리즘의 성능을 분석한다.

An Efficient Evaluation of Proof-theoretic Meaning for Normalized Recursive Rules using Shared-Nothing Parallel Architecture

Woo-Hyun Cho[†] · Jong-Hee Lee[††]

ABSTRACT

A deductive database consists of facts being the extensional database and rules being the intensional database. Because of the difficulty of evaluating rules, many parallel evaluation algorithms for rules have been presented. But we have not gotten an acceptable result. This paper proposes a new methodology to evaluate proof-theoretic meaning of the linear recursion rule which contains transitive dependency by using a shared-nothing parallel architecture. At first, we prove that there exists the equivalent expression for a linear recursion rule, design the algorithm for evaluating the linear recursion rule based on the equivalent expression, and finally analyse performance of the proposed algorithm.

1. 서 론

연역 데이터베이스는 지식 데이터로 불리는 사실(fact)들의 유한 집합과 규칙(rule)들의 유한 집합으로 구성된다. 사실들의 유한 집합을 외연적 (extensional) 데이터베이스라고 하고, 규칙들의 유한 집합을 내포적 (intensional) 데이터베이스라고 한다[1]. 연역 데이터베이스의 내포적 데이터베이스를 구성하는 규칙들이 비결정적 (non-deterministic) 성질을 갖기 때문에, 규칙들을 단일 프로세서에 의한 von Neumann 방식으로 처리하면 실행시간에 오버헤드(overhead)가 매우 커진다. 따라서 이를 줄이기 위하여 고성능 병렬구조를 이용하여 규칙들을 병렬로 처리함으로써 규칙의 계산(evaluation) 시간을 줄이는 연구가 이루어져 왔으나 비용 및 효율 측면에서 만족스런 성과를 얻지 못하였다[2-7]. 그래서 재귀규칙(recursive rule)의 효율적 계산은 연역 데이터베이스에서 중요한 관심사이다. 또,

† 종신회원 : 부경대학교 컴퓨터멀티미디어공학부 교수
†† 정 회 원 : 신라대학교 컴퓨터정보공학부 교수
논문접수 : 1999년 4월 2일, 심사완료 : 1999년 10월 12일

처리하기 어렵고 복잡한 비선형적 재귀규칙을 쉽고 단순한 선형적 재귀규칙으로 변환 또는 정규화하기 위한 방법들이 연구되었다[8-11].

Ullman[1]은 단일 처리요소에서 비재귀규칙 또는 재귀규칙의 증명-이론적 의미를 계산하는 알고리즘을 제시하였으며, Wolfson 등[12]은 비공유 병렬구조에서 규칙의 증명-이론적 의미를 계산하는 알고리즘을 제안하였다. 이 알고리즘들은 단조 증가성을 이용하여 규칙 사례화(instantiation)를 반복 수행한 후에 연역 데이터베이스의 의미(meaning)를 구하는 간단한 알고리즘이다. Valduriez 등[13]은 비공유 병렬구조에서 릴레이션(relation)의 이행성 폐포(transitive closure)를 병렬로 계산하기 위한 알고리즘을 제안하고 분석하였다.

본 논문에서는 참고문헌[14]에서 제시하지 않은 표현식 변환에 관한 증명과정을 보인다. 또 본 논문은 참고문헌[14]을 수정 및 보완한 논문이다. 먼저, 변수들 사이에 이행적 종속성을 가지도록 정규화 된 선형적 재귀규칙을 정의하고, 그 규칙은 결합 연산, 합집합 연산, 및 이행성 폐포 연산 등을 사용하여 등가의 표현식으로 변환될 수 있음을 증명하였다. 그리고 그 표현식에 근거하여 비공유 병렬구조를 이용하여 재귀규칙의 증명-이론적 의미를 병렬로 계산하기 위한 새로운 방법을 고안하였다. 마지막으로 제안된 방법과 병렬 논리 프로그래밍 시스템과 성능비교 및 고찰을 하였다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어, 2장에서는 정규화된 선형적 재귀규칙을 등가 표현식으로 나타낼 수 있음을 증명하며, 3장에서는 등가표현식에 근거하여 비 공유 병렬구조를 이용한 새로운 병렬 계산방법을 제안하고 병렬 논리 프로그래밍 시스템과 비교 및 고찰을 한다. 그리고, 4장에서 연구 결과에 대한 결론 및 추후 연구 과제를 제시한다.

2. 정규화된 선형적 재귀규칙의 등가 표현식

이 장에서는 이행적 종속성을 가진 정규화된 선형적 재귀규칙을 정의하고, 그 규칙에 포함된 술어들에 대응되는 릴레이션들을 이용하여 그 재귀규칙의 사례화 결과와 동치인 표현식으로 나타낼 수 있음을 증명한다.

연역 데이터베이스의 내포적 데이터베이스를 구성하는 규칙은

$$p \leftarrow q_1, q_2, \cdots, q_k$$

의 형태이다. 여기서 p는 규칙의 머리부분이며, q_i들의 논리곱 형태는 규칙의 몸체부분이다. p와 q_i들은 각각 인수들의 리스트를 포함하는 술어들이다. 어떤 규칙의 머리부분 p가 그 규칙의 몸체부분에 나타나면 그 규칙을 직접적 재귀규칙이라고 한다. 우리는 간접적 재귀규칙을 생각할 수 있으나, 이 논문에서는 직접적 재귀규칙만을 고려한다. 단순 재귀규칙 프로그램은 비재귀적인 종료규칙과 직접적 재귀규칙으로만 구성된다. 어떤 규칙에 대한 재귀의 등급은 그 규칙의 머리부분 술어 p가 몸체부분에 나타나는 횟수이다. 재귀의 등급이 1인 규칙을 선형적 재귀규칙이라고 하고, 재귀의 등급이 1보다 큰 규칙을 비선형적 재귀규칙이라고 한다[11].

어떤 술어에 포함된 인수들의 개수를 그 술어의 차수라고 하자. 어떤 선형적 재귀규칙에서 머리부분 술어의 차수가 2이고, 술어들의 변수들 사이에 이행적인 관계가 존재하면, 그 규칙은 이행적 종속성을 갖는다고 하자. 종료규칙과 이행적 종속성을 갖는 선형적 재귀규칙을 병렬적으로 계산하기 위해 다른 표현 방법을 고려한다.

두 릴레이션에 대한 결합 연산을

$$Q \bowtie Q' = \{(a,b) \mid \exists c\ ((a,c) \in Q \wedge (c,b) \in Q')\}$$

와 같이 정의하자.

또, Q^+는 릴레이션 Q에 대한 이행적으로 닫혀있는 릴레이션이라고 정의하고, Q^i를 릴레이션 Q의 i차 결합 릴레이션이라고 정의하자. 즉,

$$Q^1 = Q$$

이고

$$Q^i = Q^{i-1} \bowtie Q$$

이다. 그러면, Q^+는

$$Q^+ = Q^1 \cup Q^2 \cup Q^3 \cup \cdots = \bigcup_{i>0} Q^i$$

와 같다.

[정리 1] 임의의 릴레이션 A, A', B에서 $\cup$ 연산에 대한 $\bowtie$ 연산의 분배법칙은 성립한다. 즉,

$$(A \bowtie B) \cup (A' \bowtie B) = (A \cup A') \bowtie B$$

이다.

[증명]

임의의 (a,b)에 대하여

$$\begin{aligned}
&(a,b) \in (A \bowtie B) \cup (A' \bowtie B)\\
&= \{(a,b) \in (A \bowtie B)\} \vee \{(a,b) \in (A' \bowtie B)\}\\
&= \exists c\ \{((a,c) \in A) \wedge ((c,b) \in B)\} \vee \exists c'\ \{((a,c') \in A') \wedge ((c',b) \in B)\}\\
&= \exists c\ \{((a,c) \in A) \wedge ((c,b) \in B)\} \vee \exists c\ \{((a,c) \in A') \wedge ((c,b) \in B)\}\\
&= \exists c\ [\{((a,c) \in A) \wedge ((c,b) \in B)\} \vee \{((a,c) \in A') \wedge ((c,b) \in B)\}]\\
&= \exists c\ [\{((a,c) \in A) \vee ((a,c) \in A')\} \wedge ((c,b) \in B)]\\
&= \exists c\ [\{(a,c) \in (A \cup A')\} \wedge ((c,b) \in B)]\\
&= (a,b) \in (A \cup A') \bowtie B
\end{aligned}$$

은 성립한다.

그러므로, $(A \bowtie B) \cup (A' \bowtie B) = (A \cup A') \bowtie B$는 성립한다.

[증명 끝]

다음과 같은 재귀규칙을 고려하자.

$$p(x,y) \leftarrow q(x,z),\ p(z,y).$$

여기서 p는 내포술어 기호이며, q는 외연술어 기호이다. 또, p의 차수는 2이다.이때, 이 규칙은 재귀등급이 1이므로 선형적 재귀규칙이라고 할 수 있다. 그리고 머리부분의 내포술어에서 사용된 두 변수(x와 y) 중에서, 한 변수(y)가 몸체부분의 재귀를 위한 술어에 한번 사용되고, 또 다른 변수(x)가 몸체부분의 외연술어에 한번 사용되며, 몸체부분의 술어들이 머리부분의 내포술어에서 사용되지 않은 변수들로 결합되면, 이행적 종속성이 존재한다고 한다. 외연술어 $q(x,z)$는 외연술어들의 논리곱의 형태 $q'(x,\cdots)$, $\cdots$, $q''(\cdots,z)$로 구성될 수 있다.

[정리 2] 다음의 ①과 같은 이행적 종속성을 가진 선형적 재귀규칙의 계산 결과는 규칙에 포함되어 있는 외연술어에 대응되는 외연릴레이션에 대한 연산식 ②의 계산 결과와 같다. 여기서 내포술어 p는 릴레이션 P와 대응된다.

① 재귀규칙

$$p(x,y) \leftarrow q_e(x,y).$$

$$p(x,y) \leftarrow q(x,z),\ p(z,y).$$

② 연산식

$$P = Q_e \cup (Q^+ \bowtie Q_e)$$

[증명]

규칙 ①의 내포술어 p를 위한 사례화는 다음과 같다.

$$\begin{aligned}
&(p(x,y) \leftarrow q_e(x,y)) \vee\\
&(p(x,y) \leftarrow q(x,z),\ q_e(z,y)) \vee\\
&(p(x,y) \leftarrow q(x,z),\ q(z,z'),\ q_e(z',y)) \vee\\
&(p(x,y) \leftarrow q(x,z),\ q(z,z'),\ q(z',z''),\ q_e(z'',y)) \vee\\
&\cdots
\end{aligned}$$

위에서 규칙의 몸체부분 $[q(x,z),\ q'(z,y)]$는 외연술어 q와 q'의 논리곱이며 대응되는 외연릴레이션 Q와 Q'의 결합을 의미한다. 그러므로 내포술어 p를 위한 사례화는 정리 1의 분배법칙과 정의에 의하면

$$\begin{aligned}
P &= Q_e \cup (Q \bowtie Q_e) \cup (Q \bowtie Q \bowtie Q_e) \cup (Q \bowtie Q \bowtie Q \bowtie Q \bowtie Q_e) \cup \cdots\\
&= Q_e \cup (Q \cup (Q \bowtie Q) \cup (Q \bowtie Q \bowtie Q) \cup \cdots) \bowtie Q_e\\
&= Q_e \cup (Q^1 \cup Q^2 \cup Q^3 \cup \cdots) \bowtie Q_e\\
&= Q_e \cup (Q^+ \bowtie Q_e)
\end{aligned}$$

와 같다.

결국, $P = Q_e \cup (Q^+ \bowtie Q_e)$를 유도하기 때문에, 이 결과는 정리 2의 ②와 같다. 그러므로, 정리 2는 성립한다.

[증명 끝]

변수 x, y, z사이에 이행적 종속성이 존재하는 일반적인 선형적 재귀규칙은 다음 예에서와 같다.

예 1) 다음과 같은 일반적인 선형적 재귀규칙을 고려하자.

$$p(x,y) \leftarrow q_e(x,y).$$

$$p(x,y) \leftarrow q_{1,1}(x,\ x_{1,1}),\ \cdots,\ q_{1,m}(x_{1,m},\ z),\ \cdots,\ q_{n,1}(x,\ x_{n,1}),\ \cdots,\ q_{n,m}(x_{n,m},\ z),\ p(z,y).$$

예 1)의 재귀규칙에서 변수들의 관계를 보면 이행적 종속성이 존재한다. 각각의 술어에 대응되는 릴레이션을 해당 술어 기호의 대문자로 표기하자. 그러면 이행적 종속성을 갖는 선형적인 재귀규칙과 종료규칙은 대응되는 릴레이션들의 연산식으로 표현될 수 있다. 예 1)

의 규칙에 대응되는 릴레이션들의 연산식은

$$P = Q_e \cup ((Q_1 \cap Q_2 \cdots \cap Q_n)^* \bowtie Q_e)$$

이다. 여기서 Q_i는

$$Q_i = (Q_{i,1} \bowtie Q_{i,2} \bowtie \cdots \bowtie Q_{i,m})$$

이다.

3. 등가 표현식에 근거한 정규화 된 재귀규칙의 효율적 병렬계산

이 장에서는 정규화 된 재귀규칙의 등가 표현식에 근거하여 효율적인 병렬계산을 위한 비공유 병렬구조 및 처리환경에 대한 가정과 병렬계산 알고리즘을 제시하고, 알고리즘의 성능 분석과 그 결과에 대하여 고찰한다.

3.1 비공유 병렬구조

일반적인 비 공유 병렬구조는 여러 개의 노드가 연결망으로 연결된 형태이다. 여기서 각 노드는 프로세서, 지역 주 메모리(RAM), 지역 데이터베이스가 적재되는 디스크장치를 포함한다. 디스크 장치를 가지지 않은 노드들이 다른 노드와 인터페이스로서 사용되거나 일시적인 릴레이션 데이터를 처리하기 위해 사용될 수 있다. 비 공유의 의미는 노드들 사이에 주 메모리나 디스크 장치를 공유하지 않음을 의미한다. 노드들 사이에 유일하게 공유되는 것은 메시지를 교환하기 위한 연결망 뿐이다.

데이터베이스는 여러 노드에 분산되어 적재되어 있는 릴레이션(술어)들로 구성된다. 릴레이션을 수평적으로 분할하여 여러 노드에 분산 적재할 수 있다. 분산 적재되는 노드의 갯수는 릴레이션의 크기, 접근 횟수 등과 관련이 있다. 데이터베이스 연산의 성능을 향상시키기 위해 릴레이션들을 여러 개의 노드로 분산 적재하는 방법은 많이 있다. 가장 간단한 방법은 여러 노드에 튜플을 라운드로빈(round-robin) 방법으로 적재하는 것이다. 결합 연산과 합집합(교집합) 연산을 효율적으로 수행하기 위해 해쉬를 기반으로 하는 알고리즘이 사용되어 왔다. 분산 적재의 주요 특징은 릴레이션의 각 튜플에 대한 병렬 접근 가능성이다.

해쉬 기반 알고리즘의 기본 개념은 결합될 각각의 릴레이션(R과 S라고 두자)를 다음 식을 만족하도록 상호 배타적 집합들($R_1,..,R_n$ 과 $S_1,..,S_n$)로 분할하는 것이다.

$$R \bowtie S = \bigcup_{0 \le i \le n} (R_i \bowtie S_i)$$

분할은 결합 속성에 적용하는 해쉬함수에 근거한다. 각각의 결합연산 $R_i \bowtie S_i$은 R_i와 S_i에 있는 각 튜플에 대하여 중첩된 반복 절차에 의해 계산된다. 이 알고리즘은 다중노드 환경에서 동작하도록 쉽게 확장될 수 있다. 다중노드 환경에서는 각 결합 $R_i \bowtie S_i$은 서로 다른 노드에 의해 병렬로 수행된다. 합집합(교집합) 연산도 해쉬을 이용하여 결합연산과 유사한 방법으로 구현될 수 있다.

데이터베이스 연산들의 병렬 수행을 위해서는 노드들 사이에 데이타를 전송해야 할 필요가 있다. 우리는 데이터의 전송을 위해 두 가지의 기본적인 통신 프리미티브를 고려한다. *send*(메세지, 목적지 노드(들))는 메세지를 목적지 노드로 메세지를 전송하는 프리미티브이다. 각 노드의 커널은 메세지를 검사하여 해당 작업에게 제공할 수 있다고 가정한다. R := *receive*는 도착한 메세지의 내용을 받아야 할 작업이 R에 저장하는 프리미티브이다.

3.2 병렬계산 방법

이행적 종속성을 갖는 선형적인 재귀규칙의 병렬계산을 위한 방법은 알고리즘 1과 같이 네 단계로 이루어진다.

알고리즘 1. 이행적 종속성을 갖는 선형적 재귀규칙의 병렬계산

입력: 재귀규칙과 술어들
출력: 재귀규칙의 증명-이론적 의미
처리

1) 재귀규칙의 몸체 부분의 술어들에 대한 결합 연산과 교집합 연산을 이용하여 이행성 폐포를 계산할 릴레이션을 구한다.
2) 이행성 폐포가 계산될 릴레이션을 적당한 분할 알고리즘을 사용하여 여러 노드에 분할 적재한다.
3) 각 노드에 분할 적재된 릴레이션으로부터 이행성 폐포 릴레이션을 병렬로 계산한다.
4) 셋째 단계에서 계산된 이행성 폐포 릴레이션과 외연 술어들에 대하여 결합 연산과 합집합 연산을 이용하여 목표 릴레이션을 계산한다.

[알고리즘 1의 끝]

알고리즘 1을 이용하여 예 1의 규칙에 대한 증명-이론적 의미를 계산하는 방법을 보자. 이행성 폐포를

계산할 릴레이션 Q를 구하기 위해 먼저 $Q_{i,1}\bowtie Q_{i,2}\bowtie\cdots\bowtie Q_{i,m}$의 계산으로 $Q_i(1\le i\le n)$를 구한 다음에 $Q_1\cap Q_2\cap\cdots\cap Q_n$의 계산으로 Q를 구한다. 릴레이션 Q를 적당한 분할 알고리즘을 이용하여 여러 노드에 분할적재한다. 본 논문에서는 분할 적재 알고리즘에 관해서는 언급하지 않는다. 각 노드들은 분할적재된 릴레이션의 이행성 폐포를 계산한 후에, 그 결과를 Q의 이행성 폐포를 계산하기 위해 다른 노드로 전송한다. 각 노드에서의 계산 알고리즘은 알고리즘 2에서와 같다. 마지막으로, 계산된 Q의 이행성 폐포 릴레이션 Q^+와 Q_e으로 $Q^+\bowtie Q_e$를 계산한 다음 이 결과와 Q_e의 합집합을 구함으로써 목표 릴레이션을 얻는다.

이제 여기서는 릴레이션 Q의 이행성 폐포를 계산하기 위한 알고리즘을 고려한다. 이행성 폐포를 계산할 릴레이션을 d개 노드로 분할 및 적재하였다고 가정하자. 각각의 노드로 분할적재된 릴레이션을 R_1, R_2,..., R_d라고 두자. 그리고, 간단히 하기위해 d를 2^k이라고 하자. 비공유 병렬구조를 이용하여 릴레이션 Q의 이행성 폐포를 계산하는 각 노드에서의 처리과정은 알고리즘 2에서와 같다.

알고리즘 2. 비공유 병렬구조에서 병렬 이행성 폐포 계산

입력 : 각 노드에 적재된 R_i
출력 : 노드 d 에서 계산된 T_d
처리

```
1)  at each node i (i=1,2,..,d) do   (* 첫째 패스 *)
2)    begin                          (* d = 2^k *)
3)      T_i := R_i ;
4)      D_i := R_i ;
5)      repeat
6)          D'_i := D_i ;
7)          D_i := D_i ⋈ R_i ;
8)          T_i := T_i ∪ D_i ;
9)      until (D_i - D'_i) = φ ;
10)     R_i := T_i ;
11)     if i mod 2 ≠ 0 then send(T_i, node(i+1)) ;
12)   end
13) for j := 1 to [log2 d] do   (* 둘째 이상의 패스들 *)
14)   at each node i (i=1*2^j, 2*2^j, 3*2^j,...,d) do
15)   begin
16)     S_i := receive ;
17)     Flip := true ;
18)     D1_i := R_i⋈S_i ;
19)     D2_i := S_i⋈R_i ;
20)     T_i := R_i ∪ S_i ∪ D1_i ∪ D2_i ;
21)     repeat
22)       D1'_i := D1_i ;
23)       D2'_i := D2_i ;
24)       if Flip then D1_i := D1_i⋈R_i else D1_i := D1_i⋈S_i ;
25)       if Flip then D2_i := D2_i⋈S_i else D2_i := D2_i⋈R_i ;
26)       T_i := T_i ∪ D1_i ∪ D2_i ;
27)       Flip := not Flip ;
28)     until (D1_i - D1'_i) = φ and (D2_i - D2'_i) = φ ;
29)     R_i := T_i ;
30)     if i mod 2^j ≠ 0 then send(T_i, node(i+2^(j-1)));
31)   end;
```

[알고리즘 2의 끝]

알고리즘 2는 여러 패스들로 구성된다. 첫째 패스는 단계 1에서 단계 12까지이다. 첫째 패스에서 각 노드는 알고리즘 1을 이용하여 자신에게 할당된 릴레이션 R_i의 이행성 폐포 T_i를 계산한다. 첫째 패스에 참여한 d개 노드들 중에서 노드 $2n$-1은 단계 11에서 계산된 R_{2n-1}의 이행성 폐포 릴레이션 T_{2n-1}를 바로 이웃 노드인 노드 $2n$으로 보낸다(여기서 $1\le n\le d/2$ 임). 둘째 이상의 패스들은 단계 13에서 단계 31까지의 반복으로 이루어진다. 둘째 패스에서 $d/2$개의 수신 노드들(2, 4, .., d)은 자신이 만든 이행성 폐포 릴레이션 R_i와 수신된 이행성 폐포 릴레이션 S_i를 가지고 $R_i\cup S_i$에 대한 이행성 폐포 릴레이션 T_i를 계산한다. 각각 이행적으로 닫혀 있는 두 릴레이션의 합집합 $R_i\cup S_i$에 대한 이행성 폐포를 구하는 부분은 단계 16에서 단계 29까지이다. $R_i\cup S_i$에 대한 이행성 폐포 릴레이션 T_i 즉, $(R_i\cup S_i)\cup(R_i\cup S_i)^2\cup(R_i\cup S_i)^3\cup\ldots$을 구하기 위해 병렬구조를 이용하지 않고 단일 노드를 사용하여 계산한다면 R_i^+와 S_i^+를 다시 계산할 것이다. 이 계산은 바로 앞의 패스에서 이미 R_i와 S_i가 각각 이행적으로 닫혀있는 릴레이션으로 계산되었으므로 중복 계산이다. 이 중복계산을 피하기 위해 단계 17-19에서 부울변수 $Flip$와 $D1_i$, $D2_i$를 정의하여 단계 24-25와 같이 교번결합 순서를 생성한다. 이 교번 결합 순서는 $R_i\bowtie R_i$ 혹은 $S_i\bowtie S_i$와 같은 중복 계산을 포함하지 않기 때문에 알고리즘 2는 이 중복 계산을 하지 않는다.

j-1번째 패스에서 $d/2^{j-1}$개 노드들이 생성한 이행적으로 닫혀 있는 릴레이션을 바로 오른쪽으로 인접한 노드로 보내면, j번째 패스에서 $d/2^j$개 노드들은 자신이 만든 릴레이션과 수신된 릴레이션을 가지고 단계 16-29를 적용하여 새로운 이행성 폐포 릴레이션을 계산한다. 결국 새로운 이행성 폐포 릴레이션을 생성해야 할 노드가 하나만 남으면, 이 노드가 마지막 패스를 수행한 후에 알고리즘 2는 종료된다. 알고리즘 2가 d개 노드에 적용될 경우에 첫째 패스를 포함한 전체 패스의 수는 $1+\log_2 d$이다.

3.3 성능 분석 및 고찰

패러다임에서 규칙의 몸체 부분의 술어들에 대한 결합 연산과 교집합 연산을 이용하여 이행성 폐포를 계산할 릴레이션을 구하기 위해 소요되는 시간을 t_r, 이행성 폐포가 계산될 릴레이션을 적당한 분할 알고리즘을 사용하여 여러 노드에 분할 및 적재하는데 소요되는 시간을 t_p, 각 노드에 분할적재된 릴레이션으로부터 이행성 폐포 릴레이션을 병렬로 계산하기 위해 소요되는 시간을 t_{tc}, 계산된 이행성 폐포 릴레이션과 외연 술어들에 대하여 결합 연산과 합집합 연산을 이용하여 목표 릴레이션을 계산하는데 걸리는 시간을 t_o라고 두자. 패러다임에 의한 응답시간은 $t_r + t_p + t_{tc} + t_o$이다. t_{tc}는 이 전체 응답시간에 결정적이므로 여기서는 알고리즘 2의 응답시간 t_{tc}을 자세히 분석한다. t_{tc}는 노드들의 통신 시간과 각 노드에서 새로운 튜플들을 계산하는 시간의 합이다. 이행성 폐포를 구하려는 릴레이션을 R이라고 두자. 단순성을 위해 각 패스의 각 노드에서 생성되는 튜플의 수가 일정하다고 가정하자. 여기서 $|R|$은 릴레이션 R의 튜플들의 수이고, $|R^+|-|R|$은 알고리즘 2에 의하여 생성되는 튜플의 수이며, d는 알고리즘 2에 참여하는 노드들의 수이다. 또, $2d-1$은 각 패스에 참여하는 노드들의 수의 합이다. 그러면 각 패스의 각 노드에서 생성되는 튜플들의 수는 $\frac{|R^+|-|R|}{2d-1}$이다. 릴레이션 R이 균등하게 분할되어 $\frac{|R|}{d}$개의 튜플들이 각 노드에 할당된다고 가정하자. 각 패스에서 각 노드가 전송하는 튜플의 개수를 패스1의 노드1에서 n_1개, 패스2의 노드2에서 n_2개, 패스3의 노드4에서 n_4개 등이라고 두면, n_1, n_2, n_4 등은 다음과 같다.

$$n_1 = \frac{|R|}{d} + \frac{|R^+|-|R|}{2d-1}$$

$$n_2 = 2(\frac{|R|}{d} + \frac{|R^+|-|R|}{2d-1}) + \frac{|R^+|-|R|}{2d-1}$$

$$n_4 = 2(2(\frac{|R|}{d} + \frac{|R^+|-|R|}{2d-1}) + \frac{|R^+|-|R|}{2d-1}) + \frac{|R^+|-|R|}{2d-1}$$

. . .

각 노드는 튜플들을 병렬로 전송하므로 응답 시간에 영향을 미치는 전송 튜플 수의 합 n_t는 $n_1 + n_2 + n_4 + \ldots$개이다.

$$n_t = \frac{(d-1)}{d}|R| + \frac{2d-\log_2 d-2}{2d-1}(|R^+|-|R|)$$

각 패스에서 각 노드가 $\frac{|R^+|-|R|}{2d-1}$개의 튜플을 병렬로 생성하므로 전체 패스중에 응답시간에 영향을 미치는 생성 튜플 수의 합 n_c는 다음과 같다.

$$n_c = (1+\log_2 d)\frac{|R^+|-|R|}{2d-1}$$

여기서 $1+\log_2 d$는 전체 패스의 수이다. 한 개의 튜플을 어떤 노드에서 다른 노드로 옮기는데 소요되는 평균 시간을 t_{tup}이라고 하고, 각 노드에서 한 개의 튜플을 생성하는데 소요되는 평균 시간을 t_{new}라고 하면, 알고리즘 2의 응답시간 t_{tc}는 다음과 같다.

$$t_{tc} = n_t * t_{tup} + n_c * t_{new}$$
$$= (\frac{(d-1)}{d}|R| + \frac{2d-\log_2 d-2}{2d-1}(|R^+|-|R|))t_{tup} +$$
$$(1+\log_2 d)\frac{|R^+|-|R|}{2d-1}t_{new}$$

이 식에서 통신시간을 무시하면 다음 식과 같다.

$$t_{tc} \approx (1+\log_2 d)\frac{|R^+|-|R|}{2d-1}t_{new}$$

위 식의 의미는 t_{tc}가 노드의 수에 반비례함을 의미한다.

언어론적 측면에서 논리 프로그램을 효율적으로 번역하고 실행하는 방법에 관하여 오래 전부터 연구가 이루어 왔다. 논리 프로그램을 단일 처리기에 의해 처리하는 방법은 논리 프로그램의 비결정성, 과도한 퇴각검색 등 때문에 비효율적이다. 따라서, 절(clause) 혹은 규칙의 집합로 구성되는 논리 프로그래밍 언어에 포함되어 있는 병렬성을 활용하기 위한 병렬 논리 프로그래밍 시스템에 관해서 많은 연구가 이루어졌으나, 비용 및 효율 측면에서 만족스런 결과를 얻지 못하였다. 논리 프로그래밍 언어에 내재하는 병렬성에는 OR 병렬성, 목적(goal) 절에 제약을 받는 종속적 AND 병렬성, 목적절에 제약을 받지 않는 독립적 AND 병렬성이 있다.

병렬 논리프로그래밍 시스템에는 OR 병렬성을 활용하는 ANL-WAM 시스템, 독립적 AND 병렬성을 활용하는 &-Prolog 시스템, 종속적 AND 병렬성을 활용하는 JAM 추상 기계 등이 있다[16-18]. 또, 독립적 AND 병렬성과 OR 병렬성의 조합을 활용하는 ROLOG 시스템, 종속적 AND 병렬성과 OR 병렬성의 조합을 활용하는 Andora 모델 등이 있다[18-19].

OR 병렬성과 독립적 AND 병렬성을 활용하는 시스템들은 OR 병렬성과 독립적 AND 병렬성만을 포함하는 논리 프로그램에 대해서는 선형적으로 속도가 개선되었으나, 정리 2에서의 재귀규칙과 같은 종속적 AND 병렬성이 포함된 프로그램에 대해서는 최악의 경우에 단일 처리기에서 처리하는 시간과 같은 성능을 보여주었다. 또, 종속적 AND 병렬성을 활용하는 시스템들은 선형적인 속도 개선을 이룰 수 없었고, 약간의 성능개선만 보였다[7, 15-19].

그러나, 본 논문에서 증명한 정리 2에 근거하여 제안한 방법을 논리 프로그램의 병렬처리에 활용한다면, 종속적 AND 병렬성이 포함된 이행적 종속성을 가진 선형적 재귀규칙에 대한 증명-이론적 의미 계산의 속도는 처리기의 수에 반비례함을 확인할 수 있다.

4. 결 론

본 논문에서는 연역 데이터베이스에서 정규화된 규칙 즉, 이행적 종속성을 갖는 선형적 재귀규칙을 병렬로 계산하기 위한 새로운 패러다임을 연구하였다. 먼저 선형적 재귀규칙에서 이행적 종속성을 정의하고, 이행적 종속성을 갖는 선형적 재귀규칙을 대응되는 릴레이션에 대한 결합 연산과 집합 연산을 이용한 등가의 표현식으로 나타낼 수 있음을 증명하고, 선형적 재귀규칙에 대한 등가의 표현식에 근거하여 선형적 재귀규칙에 대한 병렬계산의 패러다임을 제안하였다. 비공유 병렬구조를 이용하여 릴레이션의 이행성 폐포를 병렬로 계산함으로써 이행적 종속성을 갖는 선형적 재귀규칙을 병렬로 계산하기 위한 방법을 보였다. 본 논문에서 제안한 정규화 된 선형적 재귀규칙의 병렬계산 방법은 추론기계의 구조적 측면 또는 추론 방법론적 측면에서 활용될 수 있다. 그러나 재귀규칙의 정규화, 규칙처리를 위한 효율적 병렬 구조, 효율적 적재 방법 등은 향후 연구 과제이다. 또 여러 참고문헌[20, 21]에서 소개하는 병렬화에 관한 관심사들을 고려하여 본 논문에서 제안한 병렬계산 알고리즘은 확장되어야 한다.

참 고 문 헌

[1] J. D. Ullman, Principle of Database and Knowledge-Base Systems, Vol.1, Computer Science Press, Maryland, 1988.

[2] J. S. Conery and D. F. Kibler, "Parallel Interpretation of Logic Programs," Proceeding of Conference on Functional Programming Language on Computer Architecture, pp.163-170, 1981.

[3] J. H. Chang and D. DeGroot, "And-Parallelism of Logic Programs Based on A Static Data Depen- dency Analysis," Digest of Papers of COMCON Spring85, pp.218-225, 1985.

[4] S. Ganguly, A. Silberschatz, and S. Tsur, "A Framework for the Parallel Processing of Datalog Queries," Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.143-152, 1990.

[5] B. Ramkumar and L. V. Kale, "Machine Independent AND and OR Parallel Execution of Logic Programs : Part I - The Binding Environment," IEEE Transaction on Parallel and Distributed Systems, Vol.5, No.2, pp.170-180, Feb. 1994.

[6] B. Ramkumar and L. V. Kale, "Machine Independent AND and OR Parallel Execution of Logic Programs : Part II - Compiled Execution," IEEE Transaction on Parallel and Distributed Systems, Vol.5, No.2, pp.181-192, Feb. 1994.

[7] J. Chassin and P. Codognet, "Parallel Logic Programming Systems," ACM Computing Survey, Vol.26, No.3, pp.295-336, Sept. 1994.

[8] D. J. Troy, C. T. Yu, and W. Zhang, "Linearization of Nonlinear Recursive Rules," IEEE Transaction on Software Engineering, Vol.15, No.7, pp.1109-1119, Sept. 1989.

[9] D. Troy and C. T. Yu, "Necessary and Sufficient Conditions to Linearize Doubly Recursive Programs in Logic Databases," ACM Transaction on Database Systems, Vol.15, No.3, pp.459-482, Sept. 1990.

[10] A. Al-Sukari and L. J. Henschen, "Query Independent Compilation of Linear Recursions," Proceeding of International Conference Software Engineering and Knowledge Engineering, pp.177-182, 1990.

[11] J. Han, K. Zeng, and T. Lu, "Normalization of

Linear Recursions in Deductive Databases," Proceedings of 9th International Conference on Data Engineering, pp.559-567, 1993.

[12] O. Wolfson and A. Ozeri, "Parallel and Distributed Processing of Rules by Data-Reduction," IEEE Transaction on Knowledge and Data Engineering, Vol.5, No.3, pp.523-530, Jun. 1993.

[13] P. Valduriez and S. Khoshafian, "Parallel Evaluation of the Transitive Closure of a Database Relation," International Journal of Parallel Programming, Vol.17, No.1, pp.19-42, Sept. 1988.

[14] W. H. Cho and H. J. Kim, "Parallel Evaluation of Linearly Recursive Rules using a Shared-Nothing Parallel Architecture," Transactions of Korea Information Processing Society, Vol.4, No.12, pp.3096-3077, Dec. 1997.

[15] R. Butler, E. L. Lusk, R. Olson, and R. A. Overbeek, "ANLWAM : A parallel implementation of the Warren Abstract Machine," Techical Report, U. S. Department of Energy's Argonne National Laboratories, 1986.

[16] M. V. Hermenegildo and K. J. Greene, "&-Prolog and its Performance : Exploiting independent And-parallelism," Proceedings of the 7th International Conference on Logic Programming, pp. 253-268, 1990.

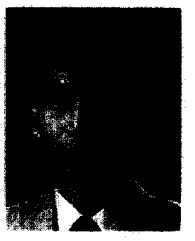
[17] J. Crammond, "Implementation of commited choice logic languages on shared memory multiprocessors," Ph.D. Thesis, Herroit-Watt University, Edinburgh, U.K., 1988.

[18] L. V. Kale and B. Ramkumar, The Reduce-OR process model for parallel logic programming on non-shared memory machines, John Wiley and sons, New York, 1992.

[19] R. Yang, T. Beaumont, I. Dutra, V. S. Costa, and D. Warren, "Performance of the compiler-based Andora-I system," Proceedings of the 10th International Conference on Logic Programming, pp.150-166, 1993.

[20] M. Mehta and D. J. DeWitt, "Managing Intra-operator Parallelism in Parallel Database Systems," Proceedings of the 21th International Conference on Very Large Data Bases, pp.382-394, 1995.

[21] E. Omiecinski and E. Lin, "The Adaptive-Hash Join Algorithm for Hypercube Multicomputer," IEEE Transactions on Parallel and Distributed Systems, Vol.3, No.3, pp.334-349, 1992.



조 우 현

e-mail : whcho@pine.pknu.ac.kr

1985년 경북대학교 전자공학과 전산공학전공 졸업(공학사)

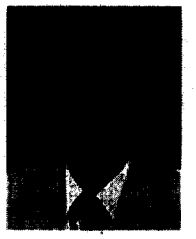
1988년 경북대학교 대학원 전자공학과 전산공학전공(공학석사)

1998년 경북대학교 대학원 전자공학과 전산공학전공(공학박사)

1988년~1989년 한국전자통신연구소 연구원

1989년~현재 부경대학교 컴퓨터멀티미디어공학부 부교수

관심분야 : 연역데이터베이스, 병렬처리, 멀티미디어데이터베이스 등



이 종 희

e-mail : jhlee@silla.ac.kr

1978년 경북대학교 전자공학과 전산공학전공 졸업(공학사)

1984년 경북대학교 대학원 전자공학과 전산공학전공(공학석사)

1990년 경북대학교 대학원 전자공학과 전산공학전공(공학박사)

1979년~1987년 경남정보대학 조교수

1988년~현재 신라대학교 컴퓨터정보공학부 부교수

관심분야 : 인터넷응용, 퍼지신경망, 컴퓨터교육 등