비공유 병렬구조를 이용한 선형적 재귀규칙의 병렬평가

조 우 현[†] · 김 항 준[††]

요 약

이 논문에서는 비공유 병렬구조에서 이행적 종속성을 갖는 선형적 재귀규칙의 병렬평가에 대한 패러다임을 제안한다. 병렬평가를 위해 우리는 모든 노드가 메시지 교환을 위해 연결망만을 공유하는 비공유 병렬구조를 고려한다. 여기서 정규화된 규칙의 평가는 그 규칙의 증명-이론적 의미의 계산이다. 이행적 종속성을 갖는 정규화된 선형적 재귀규칙을 정의하고, 그 규칙이 등가의 표현식으로 변환될 수 있음을 보이고, 등가의 표현식을 근거로 결합, 분할, 이행성폐포 연산을 이용하여 정규화된 규칙에 대한 병렬평가를 위한 패러다임을 제안하고 시간 복잡도를 분석하였다.

Parallel Evaluation of Linearly Recursive Rules using a Shared-Nothing Parallel Architecture

Woo-Hyun Cho[†] · Hang-Joon Kim[††]

ABSTRACT

This paper is concerned with a new paradigm for parallel evaluation of linear recursion rules which contain transitive dependency in a shared-nothing parallel architecture. For parallel evaluation of rules, we consider a shared-nothing parallel architecture that consists of a set of nodes and a message passing network to these nodes. An evaluation of normalized rules is a computation of the proof-theoretic meaning of a collection of rules. We shall here define normalized recursion rules which contain transitive dependency, present an equivalent expression for the rule, propose a paradigm for parallel evaluation of normalized rule based on the equivalent expression using join, partition, and transitive closure operations, and analyze response-time complexity.

1. 서 론

연역 데이터베이스(deductive database) 또는 논리언어의 규칙에 내재하는 병렬성 규칙들을 병렬로 처리하기 위한 많은 연구가 있었으나 비용/효율 측면에서 만족스런 성과를 얻지 못하였다[1]. 재귀규칙(recursive rule)의 효율적 처리는 연역 데이터베이스에서 중요한 관심사이다. 처리하기 어렵고 복잡한 비선형적 재귀규칙을 쉽고 단순한 선형적 재귀규칙으로 변환 또는 정규화하기 위한 방법들이 연구되었다[2, 3]. 이 논문에서는 변수들 사이에 이행적 종속성(transitive dependency)을 가지도록 정규화된 선형적 재귀규칙은 결합 연산, 합집합 연산, 이행성폐포(transitive closure) 연산 등을 사용하여 등가의 표현식으로 변환될 수 있음을 보이고, 그 표현식에 근거하여 재귀규칙을 병렬로 평가하기 위한 새로운 패러다임을 연구한다.

† 종신회원:부경대학교 컴퓨터공학과
†† 종신회원:경북대학교 컴퓨터공학과
논문접수:1997년 4월 2일, 심사완료:1997년 10월 2일

병렬평가(parallel evaluation)를 위한 병렬구조에는 각각의 노드가 주메모리를 공유하는 공유 병렬구조와 각각의 노드가 메모리를 공유하지 않고 노드간의 메시지교환을 위한 연결망만을 공유하는 비공유 병렬구조가 있다. 비공유 병렬구조에서 노드들을 연결하는 연결망에는 스위칭 구조와 메쉬 구조가 있다[4]. 이 논문에서는 확장성을 위해 스위칭 구조의 연결망을 가지는 비공유 병렬구조를 고려한다. 각각의 노드는 자신의 프로세서, 주메모리, 디스크 등을 갖는다. 또, 노드들은 연결망을 통하여 메시지를 서로 교환할 수 있다. 정규화된 재귀규칙을 병렬평가하기 위해서 데이터를 각각의 노드에 분할 적재할 수 있다고 가정한다.

정규화된 재귀규칙 즉, 이행적 종속성을 갖는 선형적인 재귀규칙을 병렬평가하기 위해서는 먼저, 규칙의 몸체 부분의 술어들에 대한 결합 연산과 교집합 연산을 이용하여 이행성폐포를 계산할 릴레이션을 구한 다음에, 이행성폐포가 계산될 릴레이션을 적당한 분할 알고리즘을 사용하여 여러 노드에 분할 적재한다. 그리고, 각 노드에 분할 적재된 릴레이션으로부터 이행성폐포 릴레이션을 병렬로 계산하고 계산된 이행성폐포 릴레이션과 외연(extensional) 술어들에 대하여 결합 연산과 합집합 연산을 이용하여 재귀규칙의 증명-이론적 의미(proof-theoretic meaning)를 구한다.

단일 노드에서 비재귀규칙 또는 재귀규칙의 증명-이론적 의미를 계산하는 알고리즘은 참고문헌[5]에서 제시되었으며, 비공유 병렬구조에서 규칙의 증명-이론적 의미를 계산하는 알고리즘은 참고문헌[6]에서 제시되었다. 이 알고리즘들은 단조증가성을 이용하여 규칙의 최소 고정점을 구하는 간단한 알고리즘이다. 비공유 병렬구조에서 릴레이션의 이행성폐포를 병렬로 평가하기 위한 알고리즘은 참고문헌[7]에서 제안되었으며, 이 알고리즘은 이행적 종속성을 갖는 선형적인 재귀규칙을 병렬평가하기 위해서 수정되어 이용된다.

이 논문의 구성은 다음과 같다. 2절에서 재귀규칙의 변수들의 관계로 부터 이행적 종속성을 정의하고 이행적 종속성을 갖는 선형적 재귀규칙을 등가 표현식으로 나타낼 수 있음을 보인다. 3절에서 정규화된 재귀규칙을 평가하기 위한 병렬 처리 모델에 관한 여러 가지 가정을 한다. 4절에서 정규화된 재귀규칙을 병렬평가하기 위한 패러다임과 병렬 이행성폐포 알고리즘을 제시한다. 5절에서 병렬 이행성폐포 알고리즘의 성능 평가를 위한 측도에 관한 가정을 하고 응답 시간 복잡도를 분석한다.

2. 정규화된 선형적 재귀규칙의 등가 표현식

연역 데이터베이스는 기본 릴레이션(술어)들을 포함하는 외연적 데이터베이스와 연역 규칙들을 포함하는 내포적 데이터베이스로 구성된다[5]. 연역 데이터베이스의 모든 술어들은 외연적 데이터베이스와 내포적 데이터베이스로 분할된다. 규칙은 $p \leftarrow q_1, q_2, ..., q_k$의 형태이다. 여기서 p는 규칙의 머리부분이며, q_i들의 논리곱 형태는 규칙의 몸체부분이다. p와 q_i들은 각각 인수들의 리스트를 포함하는 술어들이다. 인수는 변수 또는 상수이다. 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[2].

어떤 술어에 포함된 인수들의 갯수를 그 술어의 차수라고 하자. 어떤 선형적 재귀규칙에서 머리부분 술어의 차수가 2이고 예 1의 규칙과 같이 술어들의 변수들사이에 추이적인 관계가 존재하면, 그 규칙은 이행적 종속성을 갖는다고 하자. 종료규칙과 이행적 종속성을 갖는 선형적 재귀규칙을 병렬적으로 평가하기 위해 다른 표현 방법을 고려한다.

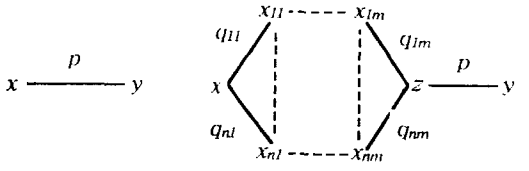
변수 x, y, z 사이에 이행적 종속성이 존재하는 일반적인 선형적 재귀규칙을 예 1에서 보인다.

예 1

$p(x, y) \leftarrow q_0(x, y).$

$p(x, y) \leftarrow q_{11}(x, x_{11}), ..., q_{1m}(x_{1m}, z), ..., q_{n1}(x, x_{n1}), ...,$
$q_{nm}(x_{nm}, z), p(z, y).$

예 1의 규칙에서 변수들의 종속성 관계 그래프를 나타내면 (그림 1)과 같다.



a) 머리부분 b) 몸체부분

(그림 1) 변수들의 종속성 관계
(Fig. 1) Dependency relationship of variables

각각의 술어에 대응되는 릴레이션을 해당 술어 기호의 대문자로 표기하자. 그러면 이행적 종속성을 갖는 선형적인 재귀규칙과 종료규칙은 대응되는 릴레이션들의 연산식으로 표현될 수 있다. 두 릴레이션에 대한 결합 연산을 다음과 같이 정의하자.

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

예 1의 규칙에 대응되는 릴레이션들의 연산식은 다음과 같다.

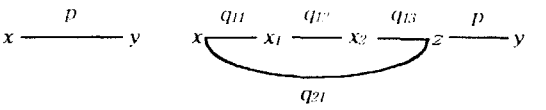
$$P = Q_0 \cup ((Q_1 \cap Q_2 \ldots \cap Q_n)^+ \cdot Q_0)$$

여기서 $Q_i = (Q_{i1} \cdot Q_{i2} \ldots Q_{im})$이며, Q^+는 릴레이션 Q에 대한 이행적으로 닫혀있는 릴레이션이다. 즉, Q^i를 관계 Q의 i차 결합 관계라고 하자. $Q^1 = Q$이고 $Q^i = Q^{i-1} \cdot Q$이다. 그러면, $Q^+ = \bigcup_{i>0} Q^i$이다. □

이행적 종속성을 갖는 선형적 재귀규칙의 간단한 예를 들자.

예 2
$p(x, y) \leftarrow q_0(x, y)$.
$p(x, y) \leftarrow q_{11}(x, x_1), q_{12}(x_1, x_2), q_{13}(x_2, z), q_{21}(x, z), p(z, y)$.

예 2의 선형적 재귀규칙에서도 변수 x, z, y 사이에 이행적 종속성이 존재하며, 재귀규칙에 대한 변수들의 종속성 관계를 그래프로 나타내면 (그림 2)와 같다.



a) 머리부분 b) 몸체부분

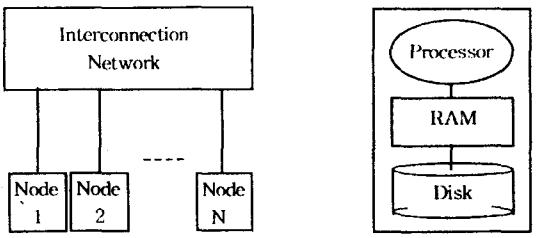
(그림 2) 변수들의 종속성 관계
(Fig. 2) Dependency relationship of variables

예 2의 규칙에 대응되는 릴레이션들의 연산식은 다음과 같다.

$$P = Q_0 \cup (((Q_{11} \cdot Q_{12} \cdot Q_{13}) \cap Q_{21})^+ \cdot Q_0)$$ □

3. 병렬 처리 모델

이 절에서는 이행적 종속성을 갖는 선형적 재귀규칙을 병렬평가하기 위한 병렬 처리 모델에 관하여 논한다. 처리 환경과 데이터베이스에 관한 많은 요인들은 병렬평가 알고리즘의 성능에 영향을 미친다. 우리는 병렬구조, 릴레이션, 알고리즘, 통신 프리미티브 등에 관한 여러가지 가정을 한다.



a) 연결망 b) 노드

(그림 3) 비공유 병렬구조
(Fig. 3) Shared-nothing parallel architecture

일반적인 비공유 병렬구조를 (그림 3)에 보인다. 각 노드는 프로세서, 지역 주메모리(RAM), 지역 데이터베이스가 적재되는 디스크 장치 등을 포함한다. 디스크 장치를 가지지 않은 노드들이 다른 기계와 인터페이스로써 사용되거나 일시적 릴레이션 데이타를 병렬로 처리하기 위해 사용될 수 있다. 비공유의 의미는 노드들 사이에 주메모리나 디스크 장치를 공유하

지 않음을 뜻한다. 노드들 사이에 유일하게 공유되는 것은 메세지를 교환하기 위한 연결망 뿐이다. 비공유 병렬구조는 작업관리와 통신을 지원하는 분산 운영체제에 의해 동작된다.

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 효률적으로 수행하기 위해 해쉬를 기반으로 하는 알고리즘이 사용되어 왔다. 분산 적재의 주요 특징은 릴레이션의 각 튜플에 대한 병렬 접근이다.

해쉬기반 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.

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

분할은 결합 속성에 적용하는 해쉬함수에 근거한다. 각각의 결합연산 $R_i \cdot S_i$와 S_i에 있는 각 튜플에 대하여 중첩된 반복 절차에 의해 계산된다. 이 알고리즘은 다중노드 환경에서 동작하도록 쉽게 확장될 수 있다. 다중노드 환경에서는 각 결합 $R_i \cdot S_i$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 데이타의 전송을 위해 두 가지의 기본적인 통신 프리미티브를 고려한다. *send*(메세지, 목적지 노드(들))는 메세지를 목적지 노드로 메세지를 전송하는 프리미티브이다. 상세한 구현을 피하기 위해 노드의 커널은 메세지를 검사하여 해당 작업에게 제공할 수 있다고 가정한다. R:=*receive*는 도착한 메세지의 내용을 받아야 할 작업이 R에 저장하는 프리미티브이다.

4. 선형적인 재귀규칙의 병렬평가

4.1 병렬평가를 위한 패러다임

이행적 종속성을 갖는 선형적인 재귀규칙의 병렬평가를 위한 패러다임은 다음과 같이 네 단계로 이루어 진다. 첫째, 규칙의 몸체 부분의 술어들에 대한 결합 연산과 교집합 연산을 이용하여 이행성폐포를 계산할 릴레이션을 구한다. 둘째, 이행성폐포가 계산될 릴레이션을 적당한 분할 알고리즘을 사용하여 여러 노드에 분할 적재한다. 셋째, 각 노드에 분할 적재된 릴레이션으로부터 이행성폐포 릴레이션을 병렬로 계산한다. 넷째, 셋째 단계에서 계산된 이행성폐포 릴레이션과 외연 술어들에 대하여 결합 연산과 합집합 연산을 이용하여 목표 릴레이션을 계산한다.

제 2절에서 보인 예 2의 규칙의 병렬평가를 위한 절차를 고려하자. 먼저 이행성폐포가 계산될 릴레이션 $Q=(Q_{11} \cdot Q_{12} \cdot Q_{13}) \cap Q_{21}$을 구한다. $Q_{11} \cdot Q_{12} \cdot Q_{13}$을 계산하기 위해 결합 연산을 병렬로 수행하고 그 결과와 Q_{21}의 교집합을 구한다. 비공유 병렬구조에서 결합 연산을 병렬로 처리하기 위해 중첩-블럭 결합(nested-block join) 방법, 정렬-합병 결합(sort-merge join) 방법, 해쉬-기반 결합(hash-based join) 방법 등이 연구되어 왔다[8].

단일 노드에 의해 두 릴레이션 R과 S의 중첩-블럭 결합 방법은 다음 단계들로 구성된다. 이 단계들은 릴레이션 R의 모든 튜플들이 읽혀지고 처리될 때까지 반복된다. 먼저, 주메모리의 용량에 적당하게 릴레이션 R의 여러 페이지를 읽는다. 이때 릴레이션 S를 위해 주메모리의 한 페이지를 남겨 둔다. 다음에, 릴레이션 S로부터 한번에 한 페이지씩 읽고 주메모리에 있는 릴레이션 R의 각 튜플과 주메모리에 있는 릴레이션 S의 각 튜플을 비교한다. 만약 결합 튜플이 존재하면 출력한다. 이 내부 반복은 릴레이션 S의 모든 튜플들이 읽혀지고 처리될 때까지 수행된다. 다중 노드에 의한 중첩-블럭 결합 방법을 위해 R과 S가 여러 노드에 분산 적재되어 있는 어떤 비공유 시스템에서 단순한 방법을 고려하자. 각각의 노드가 자신이 가지고 있는 릴레이션 R의 일부분을 모든 노드들에게 보낸다. 그러면 각각의 노드는 릴레이션 R의 모든 튜플을 가지게 된다. 각각의 노드는 릴레이션 R과 자신이 가지고 있는 릴레이션 S의 일부분을 중첩-블럭 알고

리즘으로써 처리한다.

정렬-합병 결합 방법은 중첩-블럭 결합 방법의 자연스런 대안으로써 결합에 이용되는 값으로 튜플들을 정렬하는 방법이 있다. 이 방법에서는 먼저 결합될 두 릴레이션 R과 S을 결합 속성으로 정렬한 다음에 정렬된 릴레이션 R과 S의 각각의 튜플을 결합한다. 일반적으로 이 방법은 R의 모든 튜플과 S의 모든 튜플이 비교될 필요가 없기 때문에 중첩-결합 방법보다 우수하다. 정렬-합병 결합 방법의 첫째 단계는 R과 S의 정렬을 포함하므로 병렬 정렬 알고리즘을 사용하여 쉽게 병렬화될 수 있다[9]. 정렬-합병 결합 방법의 둘째 단계는 합병 단계이며 동일한 범위 조건을 사용하여 정렬된 릴레이션을 노드들에 분할 적재할 수 있으면 병렬성의 정도를 높힐 수 있다.

해쉬-기반 결합 방법은 해쉬 함수를 이용하여 결합을 수행하는 방법이다. 해쉬-기반 결합 알고리즘들[10, 11]은 릴레이션 R과 S를 서로 소인 부분 집합들 즉 버켓이라고 하는 $R_1, R_2,..., R_n$과 $S_1, S_2,..., S_n$으로 분할한다. 동일한 결합 값을 가지는 튜플들은 동일한 버켓을 공유하게 된다. 두 릴레이션에 대해 같은 분할 스킴이 적용되기 때문에 버켓 R_i에 있는 튜플은 버켓 S_i에 있는 튜플과 결합될 것이다. 그러므로 두개의 큰 릴레이션의 결합은 각 릴레이션의 서로 소인 많고 작은 부분 집합들의 분리 결합으로 감축된다. 이 방법에는 단순 해쉬-결합, 그래이스 해쉬-결합, 하이브리드 해쉬-결합 등이 있다[8].

릴레이션 Q의 이행성폐포를 계산하기 위해 먼저 릴레이션 Q를 병렬구조 시스템에 적당한 방법으로 적재하여야 한다. 데이타 적재 또는 명백히 말해 데이타 분할은 병렬 데이터베이스 시스템에서 중요한 문제이다. 한 릴레이션을 분할하여 여러 노드에 적재하여 병렬로 각각의 노드에서 데이타를 접근케함으로써 시스템의 I/O 대역을 증대시킬 수 있다. 분할하기 위한 세가지의 기본적 방법에는 라운드로빈, 해쉬, 범위분할 등이 있다[12]. 라운드로빈 분할 스킴은 가장 단순한 방법이다. 릴레이션의 튜플들은 돌아 가면서 노드들에 적재된다. 이 스킴에서는 릴레이션의 전체 튜플의 갯수가 노드 수의 배수라면 각 노드에 적재되는 릴레이션의 튜플들의 갯수가 같게 될 것이다. 해쉬 분할 스킴은 해쉬 함수를 이용하여 노드에 튜플을 적재한다. 해쉬 분할 스킴에서는 튜플 데이타 값들의 일부분이 해쉬 함수의 인자로 사용된다. 범위 분할 스킴에서는 튜플들이 범위 함수에 의해 노드들에 적재된다. 범위 함수는 튜플 데이타 값의 범위에 의해 적재될 노드를 결정한다.

이제, 이행성폐포를 계산하는 방법을 고려하자. 릴레이션 Q의 이행성폐포를 단일 처리기에서 계산하기 위한 간단한 알고리즘을 (그림 4)에서 보였다. (그림 4)의 알고리즘은 차등 릴레이션 D를 이용하여 Q의 이행성폐포를 계산한다. (그림 4)의 알고리즘에서 Q의 이행성폐포 T는 $Q \cup D^2 \cup D^3 \cup D^4 \cup ...$가 된다. 이 논문에서는 참고문헌[7]에서 제안된 병렬 알고리즘을 수정하여 4. 2절에서 상세하게 제시한다.

알고리즘 1. 단일 노드에 의한 이행성폐포
입력 : 릴레이션 Q
출력 : 추이닫힘 릴레이션 T
처리
1 $T := Q$;
2 $D := Q$;
3 **repeat**
4 $D' := D$;
5 $D := D \cdot Q$;
6 $T := T \cup D$;
7 **until** $(D - D') = \Phi$;

(그림 4) 단일 노드에 의하여 릴레이션의 이행성폐포를 구하는 알고리즘
(Fig. 4) The algorithm for computing transitive closure of a relation using single node

예 2의 규칙을 평가하기 위한 이 패러다임의 마지막 단계에서 이행성폐포 릴레이션 Q^+ 즉, $((Q_{11} \cdot Q_{12} \cdot Q_{13}) \cap Q_{21})^+$와 외연 술어($Q_0$)들에 대하여 결합 연산과 합집합 연산을 이용하여 목표 릴레이션을 계산한다.

4.2 병렬 이행성폐포 알고리즘

릴레이션의 이행성폐포를 구하기 위하여 먼저 그 릴레이션을 분할한 후에 비공유 병렬구조의 각 노드에 적재하여야 한다. 이행성폐포를 구하고자 하는 릴레이션을 d개 노드로 분할 및 적재하였다고 가정하자. 각각의 노드로 분할 및 적재된 릴레이션을 R_i,

$R_2,..., R_d$라고 두자. 그리고, 간단히 하기위해 d를 2^k이라고 하자. 알고리즘 2는 여러 패스들로 구성된다.

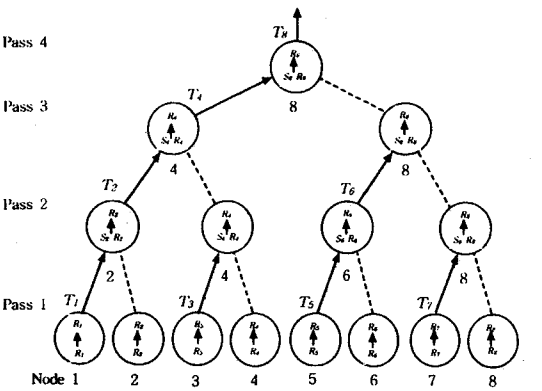
알고리즘 2. 비공유 병렬구조에서 병렬 이행성폐포
입력 : 각 노드에 적재된 R_i
출력 : 노드 d에서 계산된 T_d
처리

```
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 [log_2 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;
```

(그림 5) 비공유 병렬구조에서 릴레이션의 이행성폐포를 구하는 병렬 알고리즘

(Fig. 5) The algorithm for computing transitive closure of a relation using a shared-nothing parallel architecture

첫째 패스는 단계 1에서 단계 12까지이다. 첫째 패스에서 각 노드는 알고리즘 1을 이용하여 자신에게 할당된 릴레이션 R_i의 이행성폐포 T_i를 계산한다. 첫째 패스에 참여한 d개 노드들 중에서 노드 $2n-1$은 단계 11에서 계산된 R_{2n-1}의 이행성폐포 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 이루어 진다. 둘째 패스에서 d/2개의 수신 노드들(2, 4,.., d)은 자신이 만든 이행성폐포 릴레이션 R_i와 수신된 이행성폐포 릴레이션 S_i를 가지고 $R_i \cup S_i$에 대한 이행성폐포 릴레이션 T_i를 계산한다. 각각 이행적으로 닫혀 있는 두 릴레이션의 합집합 $R_i \cup S_i$에 대한 이행성폐포를 구하는 부분은 단계 16에서 단계 29까지이다. $R_i \cup S_i$에 대한 이행성폐포 릴레이션 $T_i \cup S_i$ 즉, $(R_i \cup S_i) \cup (R_i \cup S_i)^2 \cup (R_i \cup S_i)^3 \cup ...$을 구하기 위해 알고리즘 1을 적용한다면 R_i^+와 S_i^+를 다시 계산할 것이다. 이 계산은 바로 앞의 패스에서 이미 R_i와 S_i가 각각 이행적으로 닫혀 있는 릴레이션으로 계산되었으므로 중복계산이다. 이 중복계산을 피하기 위해 단계 17-19에서 부울변수 *Flip*와 $D1_i$, $D2_i$를 정의하여 단계 24-25와 같이 교번 결합 순서를 생성한다. 이 교번 결합 순서는 $R_i \cdot R_i$ 혹은 $S_i \cdot S_i$와 같은 중복 계산을 포함하지



(그림 6) 알고리즘 2의 패스들(d = 8일 경우)
(Fig. 6) Passes of the algorithm2 (in case of d = 8)

않기 때문에 알고리즘 2는 중복 계산을 하지 않는다.

$j-1$번째 패스에서 $d/2^{j-2}$개 노드들이 생성한 추이적으로 닫혀 있는 릴레이션을 바로 오른쪽으로 인접한 노드로 보내면, j번째 패스에서 $d/2^{j-1}$개 노드들은 자신이 만든 릴레이션과 수신된 릴레이션을 가지고 단계 16-29를 적용하여 새로운 이행성폐포 릴레이션을 계산한다. 결국 새로운 이행성폐포 릴레이션을 생성해야할 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. $d=8$ 일 때, 각 패스를 도식화하면 (그림 6)과 같다.

5. 복잡도 분석

패러다임에서 규칙의 몸체 부분의 술어들에 대한 결합 연산과 교집합 연산을 이용하여 추이 닫힘을 계산할 릴레이션을 구하기 위해 소요되는 시간을 t_r, 이행성폐포가 계산될 릴레이션을 적당한 분할 알고리즘을 사용하여 여러 노드에 분할 및 적재하는데 소요되는 시간을 t_p, 각 노드에 분할 적재된 릴레이션으로부터 이행성폐포 릴레이션을 병렬로 계산하기 위해 소요되는 시간을 t_{tc}, 계산된 이행성폐포 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. 여기서 알고리즘2의 응답시간 t_{tc}을 자세히 분석한다. t_{tc}는 노드들의 통신 시간과 각 노드에서 새로운 튜플들을 계산하는 시간의 합이다. 이행성폐포를 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 알고리즘2에 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 패스 1의 노드 1에서 n_1개, 패스 2의 노드 2에서 n_2개, 패스 3의 노드 4에서 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, 알고리즘2의 응답시간 t_{tc}는 다음과 같다.

$$t_{tc}=n_t*t_{tup}+n_c*t_{new}$$

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

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

6. 결 론

연역 데이터베이스의 비선형적 재귀규칙을 단순한 형태인 선형적 재귀규칙으로 정규화하기 위한 많은 연구가 있었다. 일반적으로 비선형적 재귀규칙의 처

리가 선형적 재귀규칙의 처리보다 복잡하다. 이 논문에서는 연역 데이터베이스에서 정규화된 규칙 즉, 이행적 종속성을 갖는 선형적 재귀규칙을 병렬로 평가하기 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 보이고, 선형적 재귀규칙에 대한 등가의 표현식에 근거하여 선형적 재귀규칙에 대한 병렬평가의 패러다임을 제안하였다. 제안된 패러다임에서는 이행성폐포 연산을 이용하여 이행적 종속성을 갖는 선형적 재귀규칙을 병렬로 평가할 수 있음을 보였다. 릴레이션의 이행성폐포를 비공유 병렬구조 환경에서 병렬로 평가하기 위한 참고문헌[7]의 알고리즘을 제안된 패러다임에 적용하기 위해 수정하여 제시하고, 이 수정된 알고리즘의 시간 복잡도를 분석하였다. 노드간의 통신시간을 무시한다면 이행성폐포 릴레이션을 구하기 위한 생성 시간의 합은 d(참여한 노드의 수)로 나누어짐을 확인하였다. 그러나 노드간의 통신시간은 응답시간에 결정적이므로, 통신의 오버헤드를 고려한 효율적 적재 방법, 시스톨릭어레이 구조를 이용한 병렬평가 방법, 재귀규칙의 정규화 등은 향후 연구 과제이다.

참 고 문 헌

[1] J. C. Kergommeaux and P. Codognet, "Parallel Logic Programming Systems," ACM Computing Surveys, Vol. 26, No. 3, pp 295-336, September 1994.

[2] J. Han, K. Zeng, and T. Lu, "Normalization of Linear Recursions in Deductive Databases," Proceedings of Ninth International Conference on Data Engineering, pp. 559-567, Aprill 1993.

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

[4] S. K. Das, "Deductive Databases and Logic Programming," Addison-Wesley Publishing Company, pp. 325-328, 1992.

[5] J. D. Ullman, "Principles of Database and Knowledge-Base Systems," Computer Science Press, Vol. 1, pp. 96-145, 1988.

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

[7] 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, 1988.

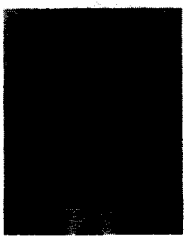
[8] D. A. Schneider and D. J. DeWitt, "A Performance Evaluation of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment," Proceedings of the ACM-SIGMOD International Conference on Management of Data, Vol. 18, pp. 110-121, June 1989.

[9] H. C. Young and A. N. Swami, "The Parameterized Round-Robin Partitioned Algorithm for Parallel External Sort," Proceedings on the 9th International Parallel Processing Symposium, pp. 213-219, Aprill 1995.

[10] M. Kitsuregawa, H. Tanaka and T. Motooka, "Application of Hash to Database Machine and its Architecture," New Generation Computing, Vol. 1, No 1, pp. 63-74, 1983.

[11] E. Omiecinski and E. Lin, "Hash-Based and Index-Based Join Algorithms for Cube and Ring Connected Multicomputers," IEEE trans. on Knowledge and Data Engineering, Vol 1, No. 3, pp. 329-343, 1989.

[12] D. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance System Database Systems," CACM, Vol. 35, No. 6, pp. 85-98, 1992.



조 우 현

1985년 경북대학교 전자공학과 전산공학전공 졸업(공학사)
1988년 경북대학교 대학원 전자공학과 전산공학전공(공학석사)
1988년~1989년 한국전자통신연구원 연구원
1989년~현재 부경대학교 컴퓨터공학과 부교수

김 황 준

1977년 서울대학교 전기공학과 졸업(공학사)
1979년 한국과학기술원 전기 및 전자공학과(공학석사)
1997년 시즈오까 대학 전자정보학부(공학박사)
1980년~현재 경북대학교 컴퓨터공학과 교수