論文93-30B-1-2

CFGL 연결성 결정에 대한 병렬 알고리듬

(Parallel Algorithm for Determining Connectedness of Context Free Graph Languages)

方惠子*, 李哲熙**

(Hye Ja Bang and Chul Hee Lee)

要 約

여러가지 제약 조건하의 문맥 자유 그래프 언어상에서 본 논문은 연결성 문제들의 간결한 그래프 표현과 그에 대한 복잡도를 분석한다. 문맥 자유 그래프 문법인 SNLC 문법을 정의하고 그래프 생성시 확장 과정에서 필요한 많은 공간을 줄일 수 있는 축약방법을 소개한다. 그래프상의 연결성 문제의 효과적 해결책의 방법을 예증하고 축약 병렬 알고리듬을 구현하고 복잡도를 분석한다. 본 논문의 결과는 NETWORK, CAD, VLSI등의 공학 설계에 많은 도움이 될것이다.

Abstract

This paper analyzes succinct graph descriptions and its complexity of connectivity problems on context free graph languages under various restrictions. It defines SNLC (Simple Context Free Node Label Controlled) grammar and presents reduction method that solves graph problems without expanding the hierarchical description. It exemplifies the method by giving efficient solutions to connectivity problems on graphs and presents parallel algorithm for reduction and analyzes the complexity. Its results will help application of design for NETWORK, CAD, VLSI and other engineering problems.

Ⅰ. 서 론

그래프 문법들은 공학의 설계들을 묘사하는데 중요한 도구들이며 기계, 전자공학, 컴퓨터 설계등에 많이 응용되어 사용되고 있다.

기술 진보에 따라 처리할 그래프의 크기가 점점 증가되고 있는데 이런 발전의 대표적인 예는 VLSI 설계분야에서 볼 수 있다. 오늘날의 집적 회로 설계는 백만개의 정점에 달하는 그래프의 내부적 표현을 요구한다. 공학설계는 항상 규칙적 구조를 가지고 확장시켜 나가며 완전한 설계표현은 자료확장단계에서 컴퓨터에 의해 이루어지고 있다. 그래프 문법의 확장은 공간적, 시간적 복잡상을 초래하며 비능률적이다.

본 논문에서는 이런 문제를 극복 할 수 있는 방법을 제시한다. 그래프를 확장시키기 전에 간결하게 그래프를 처리하는 방법을 보인다. 자료확장 단계는 처리의 마지막 출력을 만들어내는데 필요하다. 결과적으로 설계처리에서 작업기억공간의 크기는 줄어들고,

*正會員, 서울 産業大學 電子計算學科
(Dept. of Com. Eng., Seoul Nat'l Univ. Polytechnic)
**正會員, 崇實大學校 電子計算學科
(Dept. of Com. Eng.,SoongSil Univ.)
接受日字: 1992年 9月 22日

보조기억장치 경영이나 페이지 교체가 필요없게 된다. 그러나 이것도 간결한 그래프 표현을 위해 적당한 모델을 선택했을 경우이다.

본 논문 2장에서는 형식언어의 문법적 특징과 연결성 문제를 반영한 SNLC 그래프 문법을 정의한다[1,2,4,6].

3장에서는 대체규칙에서 그래프를 무한정 확장시키지 않고 그래프 언어를 축약할 수 있는 방법과 예를 보인다. 4장에서는 축약 병렬 알고리듬을 구현하여 복잡도를 계산하고 5장은 결론과 향후 연구 동향에 대하여 논의한다.

Ⅱ. 문맥 자유 그래프상의 정의들

2장에서는 그래프 문법을 지원하기 위한 SNLC (Simple Context Free Node Label Controlled) 문법을 정의한다. 이것은 [11, 12] 와 같이 개략적이고 추상적인 표기법과는 달리 그래프 문법을 처리하는 복잡도 분석의 과정들을 지원하기 때문이다. Σ는 유한 알파벱이라 하자. 그때, $\Delta \subseteq \Sigma$는 단말 라벨들의 집합이며 $\Sigma-\Delta$는 Σ의 비단말 라벨들의 집합이라 하자.

(정의 1) Σ상에서 노드 라벨된 그래프 $G = (V,E, P,lab, \Sigma)$는 $lab : V \rightarrow \Sigma$에 의해 라벨된 (V,E)의 점들의 그래프이다. P는 연결정점의 집합이다.

Σ상에서 노드 라벨된 그래프 G에서, 단말 라벨된 정점을 단말 정점, 비단말 라벨된 정점을 비단말 정점이라 부른다. $P \subseteq V$의 점들은 연결점이라 하며 이 연결점에 부 그래프들을 구별하고 연결관계를 명확하게 표시하기위해 편의상 핀을 붙인다.

(정의 2) 노드 라벨 제어된(NLC) 그래프 문법은 Σ상에서 K 개의 라벨된 그래프들 $G_1,\cdots, G_K$들인 $\rho=(G_1,\cdots,G_K,C, R, \Sigma, \Delta)$인 하나의 시스템이다. 여기서
(i) 연결관계 $C \subseteq \Sigma \times \Sigma$이고 핀으로 나타내며 p로 표시한다.
(ii) 대체 규칙들의 집합 $R\ (\Sigma-\Delta) \{1,\cdots,k\}$ 이다.
노드 라벨된 그래프 G_k 는 ρ의 출발 그래프이다.

단순 문맥자유 NLC(SNLC) 그래프 문법은 연결관계 $C = \{(e,e)\ e \in \Delta\}$을 갖는 NLC 그래프 문법이다.

(정의 3) $N(u)=\{v \in V \mid \{u,v\} \in E\}$은 그래프 $G=(V,E)$ 에서 한 정점 u의 이웃하는 정점들의 집합이라고 하자.

(정의 4) NLC 그래프 문법 $\rho=(G_1,\cdots,G_K, C, R, \Sigma, \Delta)$에서, 한 노드 라벨된 그래프 $G = (V, E, P, lab, \Sigma)$ 는 대체 규칙 $(e,t) \in R$에 의해 노드 라벨된 그래프 J 로 직접 유도되는 것을 $G \Rightarrow_{(e,t)} J$로 표시한다.

만약 e 로 라벨된 G의 비단말 정점 u가 존재하고 J는 다음과 같이 구성되어진다:
(i) u는 G로부터 제거되며 노드 라벨 그래프 J가 G-{u} 와 G_t의 합집합이지만 교집합 결과가 공집합이 되기 위한 G_t의 하나의 그래프 $G'_t=(V_t,E_t,P_t,lab_t, \Sigma)$ 에 의해 u는 또한 대체되어진다.
(ii) G'_t의 내포 함수는 정점 $v \in N(u)$와 정점 $w \in P_t$ 사이 간선들의 합에 수행되어지는 것의 필요 충분 조건은 $(lab(v),labt(w)) \in C$이다.

"$\rightarrow$"의 이행적 닫힘은 "$\Rightarrow$"로 표시된다. 만약 $G = (V, E, P, lab, \Sigma)$이 상에서 하나의 노드 라벨된 그래프이라면, $unlab(G) = (V, \Sigma)$에 의해서 G의 라벨 되어지지 않는 그래프를 표시 한다.

(정의 5) 언어 $L(\rho)$는 ρ에서 출발그래프 G_k로부터 유도 될 수 있는 상에서 노드 라벨된 그래프들의 라벨 되어지지 않는 그래프이다.
다시 말해서,
$L(\rho) = \{ unlab(G)\ G_k \Rightarrow G,\ G$는 비단말 정점들을 갖지 않음 $\}$.

(정의 6) 부그래프 G_i의 언어를 $L(\rho_i)$라 한다.

단지 그래프를 유도하는데 라벨링을 사용하기 때문에 라벨되지 않은것으로서 $L(\rho)$에서 그래프들을 고려한다. SNLC 그래프 문법에서, 비단말 정점들의 간선들은 중복된다. ρ의 크기는 $|\rho|$로 나타내며 표현의 크기를 말한다.

(그림 1)은 위에서 정의한 그래프 언어의 예를 든 것이다.

$\rho = (G_1, G_2, G_3, G_4, \{(a,c), (c,c), (b,b)\}, \{(A,1), (B,2), (C,3),\}, \{A, B, C, a, b, c,$

d, e}, {a, b, c, d, e})

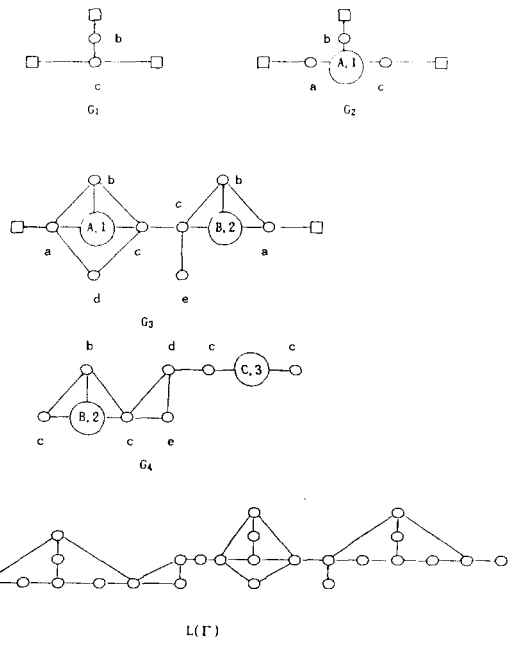


그림 1. SNLC 그래프 언어의 예
Fig. 1. SNLC Graph Language Example.

(그림1)은 정의한 SNLC그래프 문법을 바탕으로 그래프 언어를 구성하였다.

부그래프는 G_1, G_2, G_3, G_4 네개의 그래프로 구성되어 있으며 연결점은(a,c),(c,c),(b,b)이고 이 요소들만이 핀을 형성할 수 있다. 비단말정점은 A,B,C이고 (A,1)에서 A는 비단말정점의 이름이고 1은 부그래프 G_1의 형태를 말한다.

{A,B,C,a,b,c,d,e}은 유한 알파벳이고, 이중 {a,b,c,d,e}는 단말정점을 나타낸다.

언어의 확장은 한개의 비단말정점에서 이루어지며 그 단말정점에 대체규칙에 의해서 부그래프를 대체함으로 이루어진다. 즉,(A,1)의 비단말정점에 G_1의 부그래프를 대체한다. (B,2),(C,2)도 동일한 방법으로 한다.

Ⅲ. 그래프의 효과적 축약방법

일반 형식 언어에서 언어의 확장은 유도에 의해 이루어지며[7] 그래프 문법에서도 대체규칙에 의해 확장된다. 그러나 대체규칙만을 사용하는 방법은 공간적 시간적 소모로 인하여 매우 비능률적이며 비생산적이어서 본 논문에서는 대체규칙에 의한 확장 대신에 효과적 축약방법을 소개한다. 그래프 문제에서 부그래프에 대한 해는 부그래프가 놓여 있는 환경에 독립적일 수 있다. 이런 문제는 상향식 방법으로 해결될 수 있으며[9,10] 상향식 방법은 대체 규칙에 기본을 두고 있다.

Q를 그래프에 대한 질문이라고 하자

> Q : L(ρ)는 연결 되었는가 ?

이러한 질문을 풀기위하여 상향식 방법은 그래프의 테이블을 만들어 나가는데 이 테이블을 BU(Buttom Up)테이블이라 한다. BU 테이블은 세열을 가진다.

첫번째 열은 G_1, ……, G_k를 포함한다. 모든 다른 열들은 초기에 비어(Empty)있다. 이방법은 그래프 G_i를 위로부터 아래까지 $L(\rho_i)$보다 작은 G_i^b를 $L(\rho_i)$에 대체시키면서 두번째, 세번째 열을 채운다. G_i'(2열)는 G_i^b를 G_i의 형태 G_j의 모든 비단말 정점에 대해 $L(\rho_i)$대신 대체 시킴으로써 얻어진다. G_i^b와 $L(\rho_j)$가 정의에 의해 대체되어지기 때문에 G_i'과 $L(\rho_i)$또한 대체 가능하다.

G_i'는 과정을 축약시킴으로써 크기를 줄일 수 있다. 이것을 축약과정이라 하며 G_i^b는 3열에 놓이고 축약 그래프라 한다. 이 방법의 정확성은 축약과정이 대체 그래프를 산출하느냐에 달려있다. 대체 가능한 그래프를 같은 환경에 존재하게 한다면 다시 대체 그래프를 얻어낼수 있기 때문이다.

상향식 방법의 효율성은 2가지 조건에 달려있다.

첫째, 축약과정의 처리이고 두번째로 축약과정에서 작은 그래프를 산출해야한다.

G_i'을 G_i^b로 축약하기 위해 G_i'의 연결요소를 찾는다. 핀을 포함하는 각 연결요소에 대해 정점 C(지도자 노드)를 유지를 한다. 핀이 없는 연결요소가 있다면 그것모두를 나타내는 부가적 정점 또한 유지한다. G_i'의 이런 정점은 G_i^b의 정점이다.

G_i^b에서 간선은 각 핀을 이 핀을 포함하고 있는 연결요소를 나타내는 정점과 연결한다. 즉, 연결된 정점들은 다시 하나의 정점으로 나타냄으로써 축약과정을 산출 할 수 있다. 이는 문맥자유그래프 문법의 계층적 대체 규칙에서 얻을 수 있는 방법이다.

다음에 나오는 (그림2)는 축약 방법을 사용하여 BU 테이블을 만든 것이다. 첫단계에서 G의 열에는 주어진 $G_1,\cdots,G_k$를 그려넣고 첫행 G_i'과 G_i^b는 초기 단계이므로 G_i의 그래프를 그려넣는다. 두번째 단계에서는 G_i'의 그래프는 G_j^b $(j < i)$의 그래프를 비단말

정점에 대체시킨다. 즉, G_2'은 G_2^b의 그래프를 (A, 1)의 비단말 정점에 대체 시킴으로 얻을 수 있다. G_2^b는 연결요소들을 하나의 노드로 만들고 이 노드를 핀과 연결하여 얻을 수 있다. G_3'과 G_3^b도 위와 동일한 방법으로 할 수 있다. G_4'의 그래프에서 연결되었으므로 하나의 노드로 표현한 것이 G_4^b의 그래프가 되어 이 그래프가 연결되었음을 나타낸다. 그러므로 (그림 1)의 그래프 언어는 연결성을 갖는다. 다음은 위 축약방법을 사용하여 (그림 1)의 예제가 질문 P에 대해 BU테이블을 만들어서 결과를 보인다.

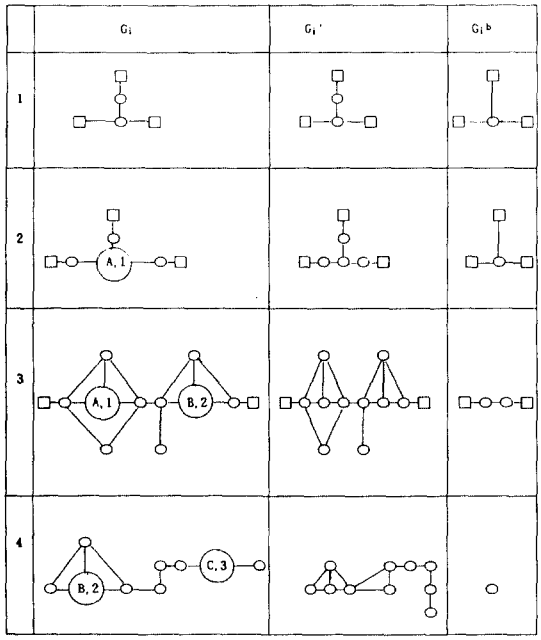


그림 2. BU 테이블
Fig. 2. BU Table.

(그림3)에서는 또 다른 하나의 그래프 문법을 정의한다. 부 그래프는 G_1, G_2, G_3, G_4 네개의 그래프이고 연결관계 (a, a), (b, b), (c, c)를 가지고 비단말정점은 (A, 1), (A, 2), (C, 3)의 형태를 가지고 단말정점은 a, b, c 이다. 그림 1과 동일하게 비단말 정점에 해당하는 부 그래프를 대체시킴으로 확장된 그래프 $L(\rho)$을 얻을 수 있다.

ρ= (G_1, G_2, G_3, G_4, {(a,a), (b,b), (c,c)}, {(A,1), (B,2), (C,3)}, {A, B, C, a, b, c, d, e, f}, {a, b, c, d, e, f})

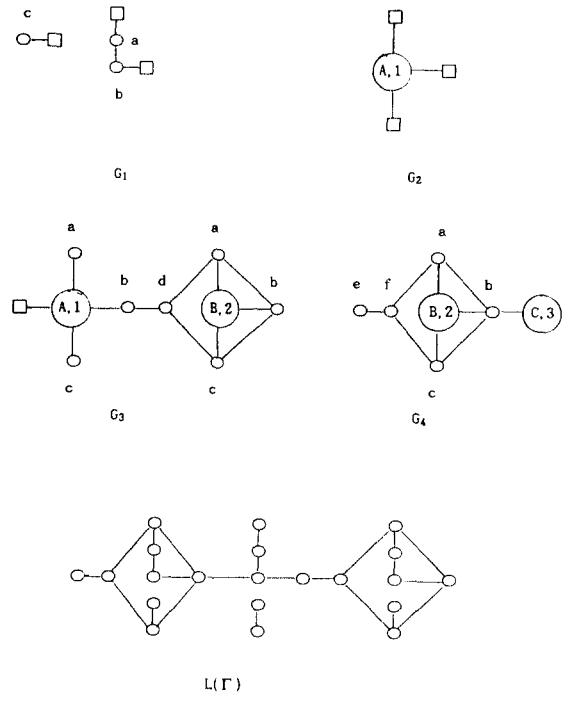


그림 3. 비연결성 그래프 언어
Fig. 3. Disconnected Graph Language.

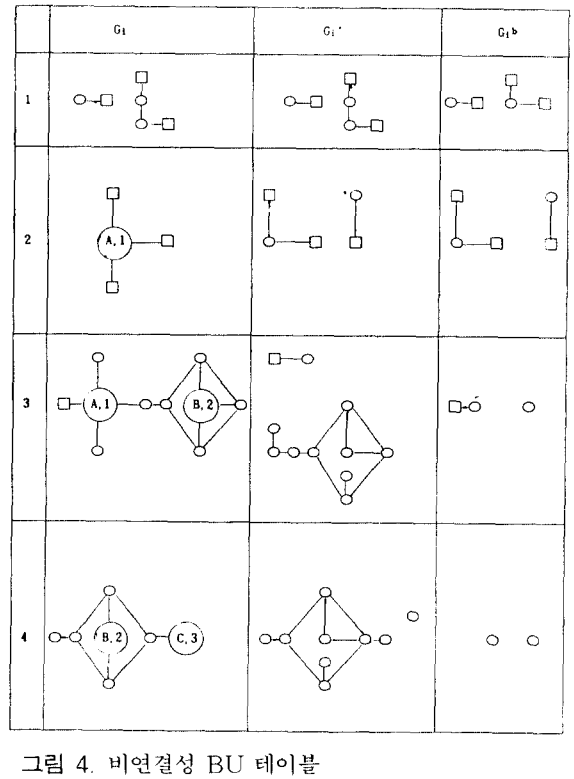


그림 4. 비연결성 BU 테이블
Fig. 4. Disconnected BU Table.

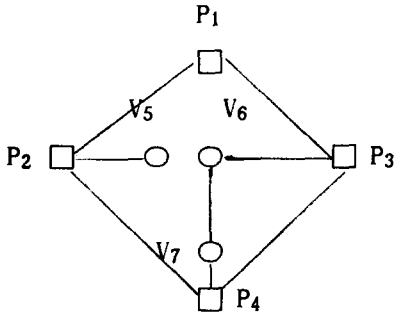
(그림 4)는 (그림 2)와 마찬가지로 (그림 3)의 그래프 문법 BU 테이블로 구성하여 연결성 문제를 풀어 보았다. G_i'의 그래프 만들고 여기에 G_j^b(j≠i)의 그래프를 대체시킴으로 G_i' 을 만들고 이 그래프에서 연결요소를 하나의 정점으로 만들어 핀을 연결한 그래프 G_i^b를 생성한다.

이렇게 만든 BU 테이블의 4행의 G_i^b의 그래프에서 2개의 정점으로 그려졌기 때문에 이 그래프는 비연결 그래프가 된다.

결국 (그림 3)의 그래프 언어는 비연결성을 갖는다.

결과로써 (그림 1)의 그래프는 연결성 그래프임을 알수있다. 또 다른 예로서 (그림 3)과 같은 언어를 소개하여 BU 테이블을 만들어 보면 (그림 4)의 결과로 비연결성 그래프임을 알 수 있다.

위 예들에서 일반적으로 모든 대체 유도 방법을 사용하면 그 크기에 따라 공간과 시간을 증가하게 되는 결과를 갖지만 BU 테이블 축약과정 방법을 사용하면 훨씬 뛰어난 공간적, 시간적 효율성을 얻을 수 있음을 알 수 있다.



$$A = \begin{bmatrix} 0&1&1&0&0&0&0 \\ 1&0&0&1&1&0&0 \\ 1&0&0&1&0&0&1 \\ 0&1&1&0&0&0&1 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&1 \\ 0&0&0&1&0&1&0 \end{bmatrix}$$

그림 5. 그래프 G와 인접행렬 A
Fig. 5. Graph A and Matrix A.

Ⅳ 축약과정의 병렬 알고리듬과 복잡도 분석

4장에서는 축약과정에서 정의된 방법을 구체적으로 병렬 컴퓨터에 구현하고 축약과정의 복잡도를 분석한다. 많은 학자들이 그래프 문법의 복잡도를 분석하고 있는데 대부분의 결과 범주는 DEXPTIME 이라고 밝히고 있다. [3, 5, 13, 14]

이 장에서 제안한 알고리듬으로 프로세서 갯수가 $4n^3-3n^2$인 3차원 그물망 트리를 사용하여 DEXPTIME을 DLOGTIME으로 줄일 수 있다. 이 알고리듬은 연결요소를 이행닫힘을 이용해 찾고 연결요소들을 하나의 정점으로 나타낸다. 이 하나의 정점을 핀 정점과 연결해 주는 알고리듬이다.

G_i' 그래프에서 G_i^b를 찾아내기 위한 알고리듬은 다음과 같다.

핀의 집합 |Sb| = k 라고 하자. 그리고 핀을 포함한 정점의 수를 n이라고 하자. 핀의 라벨은 P1 ,…, P_k 로 되어있고, 정점은 V_{k+1},…., V_n으로 라벨이 되어 있다고 하자. 이행 닫힘의 성질 [8] 을 이용하여 연결 요소를 찾는다.

연결요소는 3차원 그물망트리에서 구현될 수 있으며 이것은 인접행렬을 n-1번 곱함으로 찾아질 수 있다. 주 연산인 행렬 곱셈은 A = (a_{ij}), B = (b_{ij})일때, a_{ij}의 값은 3 차원 트리의 i, j 의 루트속에 저장되고 a_{jk}는 1차원 트리의 j, k의 루트속에 저장된다

(1≤j, k≤n). log n 단계동안 A 와 B값이 트리에서 내려오고 log n+1 단계에서 잎 노드(i, j, k)가 aij 와 bij 값을 받자마자 A, B값이 곱해지게 된다. 2 차원 트리의 루트에 그 값이 저장된다. 그래서 2logn + 1시간이 걸리게 되고, 이것을 logn 번 반복하게 됨으로 총시간은 $O(\log^2 n)$이 된다.

이렇게 해서 구한 연결요소에서 한 노드를 택해 그 노드로 통합하는 과정은 핀을 제외한 정점에서만 이루어지며 이중 포인터 방법을 사용한다. 같은 연결요소에 있는 정점중에서 가장 높게 라벨된 노드를 택하고 자기 보다 높게 라벨된 정점으로 포인트한다. 즉,

$$V_m \leftarrow P(V_j) \quad (j < m < n) \qquad (1)$$

(여기서 P는 포인터를 말한다.)

이것은 j < m < n 에서 최소값인 Vm을 선택하여 Vj의 포인터값에 할당한다.

가장 높게 라벨된 정점을 Vn이라고 한다면

$$V_n \leftarrow P(V_n) \qquad (2)$$

에 의해 자기 자신으로 포인터를 옮긴다. 나머지

정점에 대해서는

$$P(V_j) \leftarrow P(P(V_j)) \quad (k < j < N) \qquad (3)$$

으로 포인터를 이동시킨다.

다음은 이 과정에 대한 그림이다.

핀을 제외한 V_5, V_6, V_7의 정점에서 V_5의 포인터 값에 V_6을 할당하고 V_6은 V_7을 포인트하고 제일 높은 정점 V_7은 자신에게 포인트한다. 그리고나서 V_5는 V_7을, V_6도 V_7을 포인트하게 되어 V_7의 한 정점으로 통합한다.

이 과정을 한 정점으로 모든 정점이 통합될때까지 반복한다. 여기서 한 정점으로 통합한 정점을 지도자 정점이라고 한다.

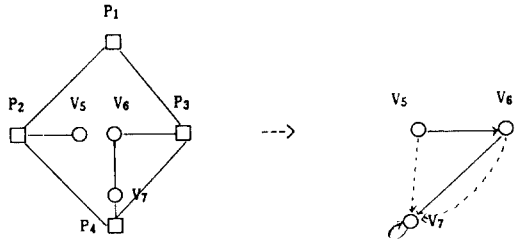


그림 6. 통합과정
Fig. 6. Coalescing Process.

방정식 (3)을 다시쓰면

$$Z(j) \leftarrow X(Y(j)) \qquad (4)$$

방정식 (4)에서 X(i)가 트리의 루트에서 잎 정점까지 내려오는 동시에 Y(j)가 내려오는데 log(n-k) 시간이 걸리고 잎 선택 알고리듬 [7]에 의해 Z(j)가 구해지고 이 값을 루트까지 보내는데 log(n-k)의 시간이 소요된다. 이것을 log(n-k)번 반복함으로 총 통합과정의 시간은 2log2(n-k)이 된다.

다음단계로 지도자 정점과 핀 정점을 연결한다. 지도자 정점을 $v_k(1 \le k \le n)$라고 하고, 핀의 수를 i개 $(1 \le i \le 4)$라 하면 다음과 같이 간선이 연결된다.

$$G = \{(V,E) \mid V=\{p_i, v_k\}$$
$$E = \{(p_i, p_k), (v_k, p_i)\}\} \qquad (5)$$

이것을 인접행렬로 나타내면 지도자 정점외의 핀이 아닌 모든 정점은 간선이 없음으로 0 이되고 지도자 정점과 핀 사이의 간선만이 존재하게 된다.

(그림 7)은 지도자 정점인 V_7만이 남게되고 V_7을 4 개의 핀과 연결시킴으로 축약과정을 하게된다.

위의 과정을 서술하면 아래와 같은 병렬 알고리듬을 기술할 수 있다.

(1) 그래프에서 이행 닫힘의 성질을 이용하여 연결요소를 찾아낸다.

(2) (1)의 단계에서 찾은 연결요소들을 하나의 정점으로 통합하게 된다. 이 통합과정은 이중포인터 방법을 통해서 이루어진다. 이 과정에서 정점의 수는 연결요소수를 나타낸다.

(3) 통합된 정점과 핀에 간선으로 연결시키게 된다. 이방법은 통합된 정점과 핀정점을 나타내는 인접행렬의 행과 열에 1이 되도록 해준다.

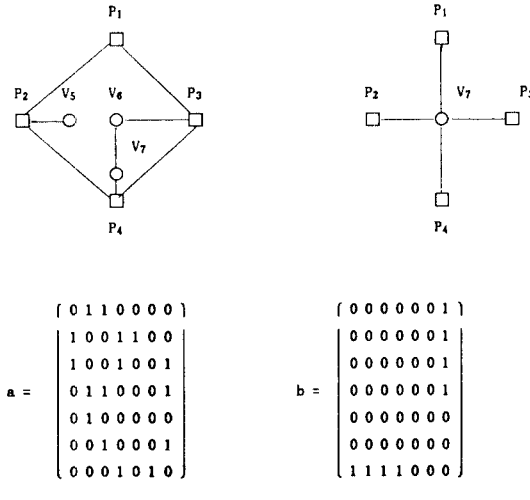


그림 7 축약과정 전과 후의 그래프와 인접행렬
Fig. 7. Graph and Matrix before and after the Reduction Process.

통합하는 과정에서 지도자 정점를 찾는데 핀을 제외한 정점에 한함으로 log(n-k) 번 반복이 필요하며, 3차원 그물망 트리 상에서 구현함으로 2log(n-k)시간이 걸리게 된다. 그래서 총 시간 복잡도는 연결요소를 찾는데 $O(\log^2 n)$이 걸리고 통합하는데 $O(\log^2 n)$이 걸리게 되어서 $O(k'(\log^2 n + \log^2(n-k)))$이 된다. 여기서 k'는 부 그래프의 수를 나타낸다.

V 결론

그래프 문법을 공학설계에 적용하려는 시도들은 많

은 학자들에 의해 논의 되었으나 형식화된 문법은 아직 미비한 상태에서 본 논문은 SNLC 그래프 문법으로 정형화하고자 하였다. 이러한 문법에서 많은 그래프들을 표현 할 수 있으며 그래프들의 연결성 문제는 설계에 막대한 영향을 주고 있는데 보다 더 큰 문제는 유도과정에서 생기는 공간적 시간적 효율성 문제이다. 간결한 BU 테이블을 사용하여 효율성을 증가시킬 수 있었으며 3차원 그물망트리구조에서 복잡도 $O(\log^2 n)$을 얻었다.

그래프 문법은 일반 스트링 문법과 달리 복잡하고 다양한 스타일을 가질 수 있는데 표현방법이나 구현은 아직도 미흡한 상태이다. 특히 그래프 문법 자체로만 그치지 않고 일반 컴퓨터 언어로 표현 방법과 그에 대한 복잡도 분석을 연구하는 것이 필요하다. 앞으로 연결성 문제에 있어서 프로세서의 갯수를 줄일수 있는 새로운 연구가 필요하다. 예를 들면 $O(n^3)$ 개의 프로세서수에서 $O(n)$으로 줄이는 방법이 연구중에 있다.

參考文獻

[1] B. Courcelle. An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theoretical Computer Science*, 55:141-181, 1987.

[2] B. Courcelle. On context-free sets of graphs and their monadic second-order theory. In H. Ehring, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, Proceedings of Graph-Grammars and *Their Application to Computer Science* '86, Pages 133-146, LNCS no. 291, Springer Verlag, Berlin/New Youk, 1987.

[3] J.Engelfriet, G Leih, and G Rozenberg. Apex graph grammars. In H. Ehring, M Nagl, A. Rosenfeld, and G. Rozenberg, editors, *Proceedings of Graph-Grammars and Their Application to Computer Science '86*, pages 167-185. LNCS no.291, Springer Verlag, Berlin/New York, 1987.

[4] A. Habel. Graph-theoretic properties compatible with graph derivations. In J. van Leeuwen, editor, *Proceedings of Graph-Theoretic Concepts in Computer Science, WG '88*, pages 11-29, LNCS no. 344, Springer Verlag, Berlin/New York, 1989.

[5] A. Habel and H.J. Kreowski. May we introduce to you: hyperedge replacement. In H.Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, Proceedings of Graph Grammars and Their Application to Computer Science '86, pages 15-26, LNCS No. 291, Springer Veriag, Berlin/New York, 1987.

[6] A. Habel, H.J. Kreowski, and W. Vogler. Compatible graph properties are decidable for hyperedge replacement graph languages. *In Bulletin of the European Association for Theoretical Computer Science* no. SS, pages 55-62, EATCS, October 1987. To appear in Acta Informatica.

[7] J.E.Hopcroft and J.D. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company, Massachusetts, 1979.

[8] F. Thomson Leighton. Introduction Parallel Algorithms and Architectures . Morgan Kaufman Publishers.Inc. 1992

[9] T. Lengauer and E. Wanke. Efficient analysis of graph properties on context-free graph languages. In T. Lepisto and A. Salomaa, editors, *Proceeding of ICALP '88*, pages 379-393, LNCS no. 317, Springer Verlag, Berlin/New York, 1988.

[10] T. Lengauer and E. Wanke. Efficient solution of connectivity problems on hierarchically defined graphs. *SIAM Journal of Computing*, 17(6):1063-1080, 1988.

[11] G. Rozenberg and E. Welzl. Boundary NLC graph grammars - basic definitions, normal forms, and complexity. Information and Control, 69(1-3):136-167, April/May/June 1986.

[12] G. Rozenberg and E. Welzl. Graph theoretic closure properties of the family of boundary NLC graph languages. Acta Informatica, 23:289-309, 1986.

[13] G. Rozenberg and E. Welzl.

Combinatorial properties of boundary NLC graph languages. Discrete Applied Mathematics, 16:59-73, 1987.

[14] E. Wanke. Algorithms for graph problems on BNLC structured graphs. Technical Report 58, Universitat-Gesamthochschule Paderborn, West-Germany, 1989.

著者紹介

方 惠 子(正會員)

1977年 숭실 대학교 전산과 졸업. 1977年~1980年 산업 연구원 연구원. 1983年 미국 North Texas State University 대학원 전산과 졸업. (석사) 1984年 유한 공전 전임 강사. 1992年 숭실 대학교 대학원 전산과 박사과정 수료. 1985年~현재 국립 서울 산업 대학 전산과 조교수. 주관심 분야는 계산이론, 그래프 이론, 컴파일러, 알고리듬 등임.

李 哲 熙(正會員)

1958年 6月 육군사관학교(.이학사) 1962年 8月 미 Purdue 대학교 대학원 전기공학과. (공학석사) 1988年 2月 중앙대학교 대학원 전자계산학과. (이학박사) 1962年 9月~1973年 2月 육군사관학교 전자공학과 교수. 1973年 3月~현재 숭실대학교 전자계산학과 교수. 주관심분야는 네트워크 이론, 통신망, 분산 시스템 등임.