韓國IT서비스學會誌
第9卷 第4號
2010年 12月

효율적인 J 관계 계산을 위한 L 클래스 계산의 개선*

한재일** · 김영만***

Improved Computation of L-Classes for Efficient Computation of J Relations*

Jae-Il Han** · Young-Man Kim***

■ Abstract ■

The Green's equivalence relations have played a fundamental role in the development of semigroup theory. They are concerned with mutual divisibility of various kinds, and all of them reduce to the universal equivalence in a group. Boolean matrices have been successfully used in various areas, and many researches have been performed on them. Studying Green's relations on a monoid of boolean matrices will reveal important characteristics about boolean matrices, which may be useful in diverse applications. Although there are known algorithms that can compute Green relations, most of them are concerned with finding one equivalence class in a specific Green's relation and only a few algorithms have been appeared quite recently to deal with the problem of finding the whole D or J equivalence relations on the monoid of all nxn Boolean matrices. However, their results are far from satisfaction since their computational complexity is exponential-their computation requires multiplication of three Boolean matrices for each of all possible triples of nxn Boolean matrices and the size of the monoid of all nxn Boolean matrices grows exponentially as n increases. As an effort to reduce the execution time, this paper shows an isomorphism between the R relation and L relation on the monoid of all n x n Boolean matrices in terms of transposition, introduces theorems based on it, discusses an improved algorithm for the J relation computation whose design reflects those theorems and gives its execution results.

Keyword : Isomorphism, Boolean Matrix, Semigroup, Monoid, Ideal, Green's Relation, Algorithm, Computational Complexity

논문투고일 : 2010년 10월 24일 논문수정완료일 : 2010년 12월 14일 논문게재확정일 : 2010년 12월 16일
* 본 연구는 2010년도 국민대학교 교내연구비를 지원받아 수행되었음.
** 국민대학교 전자정보통신대학 컴퓨터공학부, 주저자
*** 국민대학교 전자정보통신대학 컴퓨터공학부, 교신저자

1. 서 론

불리언 행렬은 원소가 0(거짓)이나 1(참) 값을 갖는 행렬로써, 논리 최적화, 선형 우선순위 함수 계산, 구문분석 등의 다양한 분야에서 유용하게 사용되고 있다[9-13]. Green 관계(relation)는 반군에서의 분할성(divisibility)을 이해하는데 매우 유용한 개념으로써, 반군에 속한 원소의 특성을 보여주는 다섯 개의 동치관계(equivalence relation)인 R, L, H, D, J 관계로 구성되며[5], 변환반군(transformation semigroup), 추계적 행렬(stochastic matrix), 다항식(polynomials) 등 여러 분야에서 응용되고 있다[6-8]. 불리언 행렬의 모노이드에서의 Green 관계는 다양한 분야에 응용할 수 있는 가능성이 있음에도 불구하고 Green 관계 계산에 요구되는 기하급수적인 계산복잡도로 인해 그 특성을 규명하기 위한 연구에 큰 어려움이 있으며, 현재 5×5이하 크기의 불리언 행렬 모노이드에서의 J 관계에 대한 극히 제한된 결과만이 알려져 있다[1, 4, 14].

본 논문은 5×5보다 큰 불리언 행렬의 모노이드에서의 J 관계 특성을 밝히기 위해 벡터 기반의 불리언 행렬 곱셈 이론과 반군에서의 Green 관계에 대한 정리를 이용하여 불리언 행렬의 모노이드에서만 나타나는 특성인 R 클래스와 L 클래스 사이의 전치(transpose) 관계를 보이고 이를 이용하여 보다 개선된 J 관계 계산 알고리즘을 제시한다. 본 논문의 구성은 다음과 같다. 제 2장은 관련연구에 대하여 논하고, 제 3장은 본 논문에서 사용할 용어와 기호를 정의한다. 제 4장은 보다 효율적인 J 관계 계산을 위해 R 관계와 L 관계 사이의 전치 행렬 변환함수가 동형(isomorphism)임을 보이는 수학적 이론을 정립하고 이를 이용하여 설계한 J 관계 계산 알고리즘을 기술하며, 제 5장은 알고리즘의 계산복잡도와 실행결과에 대하여 논한다. 제 6장은 결론 및 향후 연구방향에 대하여 논한다.

2. 관련 연구

불리언 행렬에 대한 기존의 연구는 주로 두 불리언 행렬의 효율적인 곱셈에 초점을 두고 있으며[15-22], 모든 불리언 행렬 사이의 곱셈은 극히 소수의 연구에서 다루고 있다[2, 3]. Green 관계와 관련된 거의 모든 연구도 변환반군의 특성을 이용하는 알고리즘에 대한 것이며[21-27], 변환반군뿐 아니라 [27-31]과 같은 여러 종류의 반군을 다룰 수 있는 범용 알고리즘은 매우 소수로써 현재 [30]에서 제시한 알고리즘이 가장 효율적인 알고리즘으로 알려져 있다. 그러나 [30]의 알고리즘은 하나의 L 클래스나 R 클래스 계산을 위한 것으로, 불리언 행렬 반군에서의 J 클래스의 계산은 모든 불리언 행렬 사이의 이중 연속곱셈이 요구되어 J 클래스 계산에 적합하지 않으며 J 관계 계산 즉 모든 J 클래스의 계산에 최적화되어 있지 않다[4].

최근 불리언 행렬의 모노이드에서 모든 J 클래스를 계산하기 위한 알고리즘이 [4]에 제시되었으나, 5×5보다 큰 불리언 행렬 모노이드에서의 J 관계를 계산할 수 있을 만큼 효율적이지 못하며 아직 알고리즘 개선에 대한 많은 연구가 필요하다.

3. 용어 및 기호 정의

본 논문은 다음과 같이 용어와 기호를 정의한다. 이항연산 $\cdot$이 하나 주어진 집합 S가 다음 특성을 만족하면 반군(semigroup)으로 부르며 $(S, \cdot)$ 또는 간략히 S로 표기한다.

$$x \cdot y \in S \text{ for any } x, y \in S$$
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \text{ for any } x, y, z \in S$$

$x \cdot y$는 간략한 표기를 위해 보통 $\cdot$을 생략하고 xy로 표기한다. 모든 $a \in S$에 대해서 $ae = ea = a$인 원소 e를 단위 원소(identity element)라고 하며, $ee = e$인 원소 $e \in S$를 멱등원(idempotent element)이라 부른다. 단위원소를 가지는 반군 M은

모노이드(monoid)로 정의된다.

$I \subseteq S$일 때, 어떤 $s \in S$와 모든 $x \in I$에 대해 $xs \in I$ 이면 I를 우 아이디얼(right ideal)이라 부르고 $sx \in I$이면 I를 좌 아이디얼(left ideal)이라 부른다. 주 아이디얼(principal ideal)은 하나의 원소 $a \in S$에 의해 생성되는 아이디얼을 의미하며, a에 대해 S의 모든 원소를 연산한 결과의 집합으로써 구체적으로 다음과 같이 정의된다. 어떤 원소 $a \in S$가 주어졌을 때 집합 $Sa = \{sa : s \in S\}$, $aS = \{as : s \in S\}$, $SaS = \{s_1 a s_2 : s_1, s_2 \in S\}$는 각각 a에 의해 생성되는 좌 주 아이디얼(left principal ideal), 우 주 아이디얼(right principal ideal), 양측 주 아이디얼(two-sided principal ideal)을 정의하며, a는 자신이 생성하는 각 아이디얼의 생성자(generator)로 부른다.

M을 유한개의 원소를 갖는 모노이드, a, b를 M에 속한 임의의 두 원소라고 할 때 Green 관계는 다음 다섯 개의 R, L, J, H, D 관계로 정의된다.

aRb if and only if $aM = bM$
aLb if and only if $Ma = Mb$
aJb if and only if $MaM = MbM$
aHb if and only if aRb and *aLb*
aDb if and only if there exists a c in S such that *aRc* and *cLb*

임의의 $n \times m$ 불리언 행렬 A가 주어지고 $F = \{0, 1\}$이라 할 때, $M_n^m(F)$는 모든 $n \times m$ 불리언 행렬의 집합을 정의하며, A_i와 A^i는 각각 A행렬의 i행과 i열을 의미한다. A^T는 A의 전치(transpose) 행렬이며, m차원 벡터 v는

$$v = (b_0 b_1 \cdots b_{m-1}),\ b_i \in F,\ 0 \le i \le m-1$$

로 정의하고 $n \times m$ 불리언 행렬의 행에 대응하여 행벡터로 부른다. v^T는 v의 열과 행 번호를 바꾼 $m \times 1$ 불리언 행렬로서

$$v^T = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{pmatrix} \quad where\ v = (b_0 b_1 \cdots b_{m-1})$$

으로 정의되며 $m \times n$ 불리언 행렬의 열에 대응하여 열벡터로 부른다. $V(m)$은 모든 m차원 행벡터의 집합이며 $(V(m))_k$는 m차원 행벡터를 k개 조합하여 생성한 모든 $k \times m$ 불리언 행렬의 집합이다. $V^T(m)$은 모든 m차원 열벡터의 집합이며 $(V^T(m))^k$는 m차원 열벡터를 k개 조합하여 생성한 모든 $m \times k$ 불리언 행렬의 집합이다.

$$V(m) = \{ (b_0 b_1 \cdots b_{m-1}) \mid b_i \in F \text{ for } 0 \le i \le m-1 \}$$

$$(V(m))_k = \left\{ \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{k-1} \end{pmatrix} \mid v_i \in V(m) \text{ for } 0 \le i \le k-1 \right\}$$

$$V^T(m) = \{ v^T \mid v \in V(m) \}$$

$$(V^T(m))^k = \{ (v_0^T v_1^T \cdots v_{k-1}^T) \mid v_i \in V(m) \text{ for } 0 \le i \le k-1 \}$$

$n \times m$ 불리언 행렬 A와 m 차원 열벡터 v^T의 곱셈은 n 차원 열벡터를, m 차원 행벡터 v와 $m \times n$ 불리언 행렬 B의 곱셈은 n 차원 행벡터를 생성한다.

$$vB = (vB^0 vB^1 \cdots vB^{n-1})\ where\ v \in V(m)$$

$$Av^T = \begin{pmatrix} A_0 v^T \\ A_1 v^T \\ \vdots \\ A_{n-1} v^T \end{pmatrix}\ where\ v = V(m)$$

4. J 관계 계산 알고리즘의 개선

[4]는 수학적 이론을 바탕으로 모든 J 클래스를 보다 효율적으로 계산하기 위한 J 관계 계산 알고리즘을 제시하였으나 위에 언급하였듯이 5×5보다 큰 불리언 행렬 모노이드에서의 J 관계를 계산할 수 있을 만큼 효율적이지 못하다. 본 장은 불리

언 행렬의 모노이드에 대한 여러 정리를 기반으로, 불리언 행렬의 모노이드에서의 R 관계와 L 관계가 주어졌을 때 R 클래스(또는 L 클래스)의 불리언 행렬을 전치행렬 변환으로 매핑하는 함수를 정의하면 이 함수가 동형(isomorphism)이라는 특성을 수학적으로 증명하고, 이를 이용하여 실행시간을 개선한 알고리즘을 기술한다. 본 논문은 논문의 완결성을 위해 알고리즘 설계에 필요한 기존의 여러 정리와 증명 그리고 간단한 설명을 포함한다. 기존 정리에 대한 자세한 설명은 [2, 3, 4, 5]를 참고하라.

$n\times n$ 불리언 행렬의 집합인 $M_n^n(F)$에 불리언 연산에 의한 행렬 곱셈을 이항연산으로 정의하면 [정리 1]에서 보는 것처럼 $M_n^n(F)$가 모노이드임을 쉽게 알 수 있다.

[정리 1] $M_n^n(F)$의 두 불리언 행렬의 연산을 불리언 연산에 의한 행렬 곱셈으로 정의하면 $M_n^n(F)$는 모노이드이다.

(증명) A, B, C가 $M_n^n(F)$에 속한 임의의 $n\times n$ 불리언 행렬이면, 행렬곱셈 정의에 의해 $AB\in M_n^n(F)$이고, $A(BC)=(AB)C$이며, 단위행렬(identity matrix)이 $M_n^n(F)$에 속하므로 모노이드이다. ■

[정리 2] $M_n^m(F)$에 속한 임의의 $n\times m$ 불리언 행렬 A에 대해 R_A, C_A를

$$R_A=\{AB \mid B\in M_m^k(F)\},$$
$$C_A=\{(A_0v^TA_1v^T\cdots A_{n-1}v^T)^T \mid v\in V(m)\}$$

로 정의할 때 $R_A=(C_A)^k$이다.

(증명) H를 $(C_A)^k$에 속한 임의의 $n\times k$ 불리언 행렬이라 하고 $H\notin R_A$라고 가정하자. $H\notin R_A$이므로 R_A에 속한 모든 $n\times k$ 불리언 행렬 G에 대하여 $H\neq G$이다. 따라서 H의 어떤 i열이 R_A에 속한 모든 $n\times k$ 불리언 행렬 G의 i열과 달라야 한다. $M_n^m(F)$는 모든 $n\times m$ 불리언 행렬의 집합이므로 V가 m 차원 불리언 벡터의 전체 집합일 때

$$M_m^k(F)=\{(v_0^Tv_1^T\cdots v_{k-1}^T) \mid v_i\in V,\ 0\le i\le k-1\}$$

이고

$$R_A=\{((A_0u_0^TA_1u_0^T\cdots A_{n-1}u_0^T)^T\ (A_0u_1^TA_1u_1^T\cdots A_{n-1}u_1^T)^T\ \cdots\ (A_0u_{k-1}^TA_1u_{k-1}^T\cdots A_{n-1}u_{k-1}^T)^T) \mid u_i\in V,\ 0\le i\le k-1\}$$

이 된다. H는 $(C_A)^k$에 속한 $n\times k$ 불리언 행렬이므로 H의 각 열 H^i는 V에 속한 어떤 m 차원 벡터 u에 대하여 $H^i=(A_0u^TA_1u^T\cdots A_{n-1}u^T)$이 되므로 H가 R_A에 속하게 되어 모순이다.

D를 R_A에 속한 임의의 $n\times k$ 불리언 행렬이라 하자. C_A는 $n\times m$ 불리언 행렬 A의 각 행에 V^T에 속한 각 m 차원 불리언 벡터 v^T를 곱하여 얻은 n차원 벡터의 전체 집합이다. 따라서 0과 $k-1$ 사이의 모든 i에 대해 D가 C_A에 속하며, $D\in(C_A)^k$이다. ■

[정리 2]는 주어진 $n\times m$ 불리언 행렬에 모든 $m\times k$ 불리언 행렬을 곱하여 얻는 불리언 행렬의 집합이 $n\times m$ 불리언 행렬에 모든 m 차원 열벡터를 곱하여 얻는 열벡터들을 모든 가능한 k번의 조합으로 얻는 불리언 행렬 집합과 같다는 것을 보인다 [2]. [정리 3]도 [정리 2]와 유사하게 증명되며, 모든 $l\times n$ 불리언 행렬을 하나의 $n\times m$ 불리언 행렬에 곱하여 얻는 불리언 행렬의 집합이 모든 n차원 행벡터를 $n\times m$ 불리언 행렬에 곱하여 얻는 행벡터들을 모든 가능한 k번의 조합으로 얻는 불리언 행렬 집합과 같다는 것을 보인다.

[정리 3] $M_n^m(F)$에 속한 임의의 $n\times m$ 불리언 행렬 A에 대해 L_A, T_A를

$$L_A=\{BA \mid B\in M_l^n(F)\},$$
$$T_A=\{(vA^0vA^1\cdots vA^{m-1}) \mid v\in V(n)\}$$

로 정의할 때 $L_A=(T_A)_l$이다.

[정리 4] M이 주기적 반군(periodic semigroup)이라면 $D=J$이다.
(증명) 임의의 두 원소 $a, b\in M$에 대해 aJb라고 하자. J 관계 정의에 의해 $xay=b$이고 $ubv=a$인 x, y, u, v가 M에 존재한다. $ubv=a$에서 b를 xay로 반복 치환하면

$$a=(ux)a(yv)=(ux)^2a(yv)^2=(ux)^3a(yv)^3=\cdots$$

을 얻으며 같은 방법으로

$$b=(xu)a(vy)=(xu)^2a(vy)^2=(xu)^3a(vy)^3=\cdots$$

를 얻는다. M이 주기적이므로 $(ux)^m$이 멱등원인 m이 존재한다([5]의 정리 1. 2. 3). $c=xa$라고 하면,

$$a=(ux)^ma(yv)^m=(ux)^m(ux)^ma(yv)^m$$
$$=(ux)^ma=(ux)^{m-1}uc$$

이므로 aLc이다. $cy=xay=b$이므로 $(vy)^n$이 멱등원인 n에 대해

$$c=xa=x(ux)^{n+1}a(yv)^{n+1}$$
$$=(xu)^{n+1}xay(vy)^nv=(xu)^{n+1}b(vy)^{2n}v$$
$$=(xu)^{n+1}b(vy)^{n+1}(vy)^{n-1}v=b(vy)^{n-1}v$$

이고, cRb이다. 따라서 aLc이고 cRb인 c가 항상 존재하므로 $D=J$이다. ■

[정리 1]은 $M_n^n(F)$가 유한 모노이드임을 보이고 있으며, [정리 4]는 유한 모노이드에서 J관계와 D관계가 같음을 보이고 있다[5]. 따라서 $M_n^n(F)$에서 $J=D$ 이다.

[정리 5] $a, b\in M$일 때 R_a와 L_b를 다음과 같이 정의하자.

$$R_a=\{c\in M \mid aM=cM\}$$
$$L_b=\{c\in M \mid Mb=Mc\}$$

M이 유한 모노이드라면 임의의 두 원소 $a, b\in M$에 대해서

$$aJb \text{ if and only if } R_a\cap L_b\neq\varnothing .$$

이다.
(증명) M이 유한 모노이드라면 임의의 두 원소 $a, b\in M$에 대해서 $aJb=aDb$이다. aDb라면 정의로부터 aRc이고 cLb인 c가 존재하므로 $R_a\cap L_b\neq\varnothing$이다. ■

[정리 5]는 공통원소를 가지고 있는 모든 좌 주 아이디얼과 우 주 아이디얼의 생성자 집합을 찾아 합집합을 계산하면 J 관계를 얻을 수 있음을 보인다[4].

[정리 6] R_A가 임의의 $A\in M_n^n(F)$에 대한 R 클래스일 때 $(R_A)^T$는 L 클래스이다.
(증명) 임의의 $A\in M_n^n(F)$에 대해 R_A, C_A, L_A, T_A를 다음과 같이 정의하자.

$$R_A=\{AB \mid B\in M_n^n(F)\},$$
$$C_A=\{Av^T \mid v\in V(n)\}$$
$$L_A=\{BA \mid B\in M_n^n(F)\},$$

$$T_A = \{vA \mid v \in V(n)\}$$

[정리 2]에 의해 $R_A = (C_A)^n$이고 [정리 3]에 의해 $L_A = (T_A)_n$이다. 임의의 $B \in M_n^n(F)$에 대해 $B^T \in M_n^n(F)$이므로,

$$\begin{aligned}(R_A)^T &= \{(AB)^T \mid B \in M_n^n(F)\} \\ &= \{B^T A^T \mid B \in M_n^n(F)\} \\ &= \{DA^T \mid D \in M_n^n(F)\} = L_{A^T} = (T_{A^T})_n\end{aligned}$$

이다. 따라서 $(R_A)^T \in L$이다. ■

[정리 6]은 $n \times n$ 불리언 행렬의 모노이드에서의 R 클래스에 속한 불리언 행렬을 전치(transpose)하면 L 클래스를 얻을 수 있음을 의미한다. [정리 7]은 L 클래스에 속한 불리언 행렬을 전치(trans- pose)하면 R 클래스를 얻을 수 있음을 보이며 [정리 6]과 유사하게 증명된다.

[정리 7] L_A가 임의의 $A \in M_n^n(F)$에 대한 L 클래스일 때 $(L_A)^T$는 R 클래스이다.

[정리 8] $n \times n$ 불리언 행렬의 모노이드에서의 R 관계와 L 관계가 주어졌을 때 다음과 같이 전치행렬 변환에 의해 R 클래스를 L 클래스로 매핑하는 함수 $\tau : R \rightarrow L$를 정의하자.

$$\tau(r) = r^T \; where \; r \in R$$

이 때 함수 τ는 동형(isomorphism)이다.

(증명) L 관계에 속한 서로 다른 두 개의 L 클래스를 택하고, 각 L 클래스에 속한 임의의 불리언 행렬 A, B를 택하여, 두 L 클래스의 이름을 L_A, L_B라 하면, [정리 6]에 의해 $L_A = (R_{A^T})^T = \tau(R_{A^T})$이고 $L_B = (R_{B^T})^T = \tau(R_{B^T})$인 R_{A^T}와 R_{B^T}가 R 관계에 존재하며, $R_{A^T} = R_{B^T}$이면 $L_A = L_B$가 되어 모순이 되므로 $R_{A^T} \neq R_{B^T}$이다. 즉, τ는 1대1 함수이다.

L 관계에 속한 임의의 L 클래스를 택하고, 이 L 클래스에 속한 임의의 불리언 행렬 A를 택하여, L 클래스의 이름을 L_A라 하면, [정리 6]에 의해 $L_A = (R_{A^T})^T = \tau(R_{A^T})$인 R_{A^T}가 R 관계에 존재하므로 $L_A \in \tau(R)$이며, 따라서 $L \subseteq \tau(R)$ 즉 τ가 전사함수(onto function)이다. 따라서 τ는 1대1 함수이면서 전사함수이므로 동형이다. ■

[정리 8]은 $n \times n$ 불리언 행렬의 모노이드에서 R 관계와 L 관계가 주어졌을 때, R 관계의 R 클래스를 L 관계의 L 클래스로 전치행렬 변환에 의해 매핑하는 함수가 동형(isomorphism)임을 보인다. [정리 9]도 [정리 8]과 유사하게 증명되며, L 관계의 L 클래스를 R 관계의 R 클래스로 전치행렬 변환에 의해 매핑하는 함수가 동형(isomorphism)임을 보인다.

[정리 9] $n \times n$ 불리언 행렬의 모노이드에서의 R 관계와 L 관계가 주어졌을 때 다음과 같이 전치행렬 변환에 의해 L 클래스를 R 클래스로 매핑하는 함수 $\mu : L \rightarrow R$를 정의하자.

$$\mu(l) = l^T \; where \; l \in L$$

이 때 함수 μ는 동형(isomorphism)이다.

[정리 8]과 [정리 9]는 R 관계와 L 관계가 서로 전치행렬 변환에 의해 얻어질 수 있음을 보여준다.

이 특성은 Green 관계에서 일반적이지 않으며, 불리언 행렬의 모노이드에서만 나타나는 특이한 특성이다.

[그림 1]은 $M_n^n(F)$의 모든 J 관계를 계산하기 위해 [정리 2]~[정리 5]를 적용하여 설계한 기존

RiList = ∅
RiGenList = ∅
for each boolean matrix A in $M_n^n(F)$
  compute a right principal ideal RI_A of A
  if ($RI_A \in RiList$)
    insert A into GEN(RI_A)
  else
    make a new generator set GEN(RI_A) for RI_A
    insert GEN(RI_A) into *RiGenList*
    insert A into GEN(RI_A)

LiList = ∅
LiGenList = ∅
for each boolean matrix A in $M_n^n(F)$
  compute a left principal ideal LI_A of A
  if ($LI_A \in LiList$)
    insert A into GEN(LI_A)
  else
    make a new generator set GEN(LI_A) for LI_A
    insert GEN(LI_A) into *LiGenList*
    insert A into GEN(LI_A)

JRel = ∅
for each GEN(RI_A) ∈ *RiGenList*
  J_A = GEN(RI_A)
  for each GEN(LI_A) ∈ *LiGenList*
    if GEN(RI_A) ∩ GEN(LI_A) ≠ ∅
      $J_A = J_A \cup$ GEN(LI_A)
  if $J_A \notin$ *JRel*
    insert J_A into *JRel*

- RI_A : a right principal ideal of A
- LI_A : a left principal ideal of A
- *RiList* : a list of right principal ideals
- *LiList* : a list of left principal ideals
- GEN(*PIDL*) : a set of generators of PIDL
- *RiGenList* : a list of generator sets for right principal ideals
- *LiGenList* : a list of generator sets for left principal ideals
- *JRel* : a set of J relations

[그림 1] J 관계 계산 알고리즘

의 알고리즘이다. 이 알고리즘은 먼저 [정리 2]를 적용하여 $M_n^n(F)$의 모든 우 주 아이디얼의 생성자 집합을 계산하고, 다음 [정리 3]을 적용하여 모든 좌 주 아이디얼의 생성자 집합을 계산한 후, [정리 5]를 적용하여 공통원소를 가지고 있는 모든 우 주 아이디얼과 좌 주 아이디얼의 생성자 집합의

RiList = ∅
RiGenList = ∅
for each boolean matrix A in $M_n^n(F)$
  compute a right principal ideal RI_A of A
  if ($RI_A \in RiList$)
    insert A into GEN(RI_A)
  else
    make a new generator set GEN(RI_A) for RI_A
    insert GEN(RI_A) into *RiGenList*
    insert A into GEN(RI_A)

LiGenList = ∅
for each GEN(RI_A) in *RiGenList*
  for each boolean matrix A in GEN(RI_A)
    insert A^T into GEN(LI_A)
  insert GEN(LI_A) into *LiGenList*

JRel = ∅
for each GEN(RI_A) ∈ *RiGenList*
  J_A = GEN(RI_A)
  for each GEN(LI_A) ∈ *LiGenList*
    if GEN(RI_A) ∩ GEN(LI_A) = ∅
      $J_A = J_A \cup$ GEN(LI_A)
  if $J_A \notin$ *JRel*
    insert J_A into *JRel*

- RI_A : a right principal ideal of A
- LI_A : a left principal ideal of A
- *RiList* : a list of right principal ideals
- GEN(*PIDL*) : a set of generators of PIDL
- *RiGenList* : a list of generator sets for right principal ideals
- *LiGenList* : a list of generator sets for left principal ideals
- *JRel* : a set of J relations

[그림 2] 개선된 J 관계 계산 알고리즘

합집합을 얻어 모든 J 클래스를 계산한다. [그림 2]는 [정리 3]대신 [정리 8]을 적용하여 L 관계를 R 관계의 전치행렬 변환에 의해 바로 얻도록 개선한 알고리즘을 보이고 있다.

5. 계산 복잡도 및 실행 결과

J 관계 계산 알고리즘은 Java 언어로 구현하였으며 Intel Core 2 CPU 2.66 GHz, 3GB RAM, 250 GB HDD, Micorsoft Windows XP Professional 환경에서 실행하였다. 불리언 행렬과 벡터의 곱셈

〈표 1〉 실행시간 비교

행렬 크기	J 관계 계산 알고리즘[4]	L 관계 계산 개선 알고리즘
2×2	0.029초	0.021초
3×3	0.359초	0.284초
4×4	76.714초	61.056초
5×5	195,947.344초	167,800.403초

〈표 2〉 공간 및 시간 복잡도 비교

구 분	행렬곱셈 알고리즘	정리적용 알고리즘
시간 복잡도	2^{n^2}	2^n
공간 복잡도	$n^2 2^n$	$n2^n$

은 [4]와 같이 행 OR-연산 기반 불리언 행렬 곱셈 [19]을 비트사이의 논리연산에 적합하도록 변형하여 수행하였다.

<표 1>은 [4]와 본 논문에서 제시한 J 관계 계산 알고리즘의 실행시간을 보이고 있다. 모든 J 관계 계산 알고리즘의 계산복잡도는 본질적으로 기하급수적이다. 따라서 [그림 2]의 알고리즘은 <표 2>와 같이 [4]의 알고리즘과 같은 계산 복잡도를 가진다. 그러나 <표 1>에서 보는 것처럼 [정리 8]을 적용하여 L 클래스 계산을 개선한 본 논문의 알고리즘은 전체 실행시간이 [4]의 알고리즘보다 약 20% 개선된 결과를 보이고 있으며, L 클래스 계산 시간만을 비교하면 $M_3^3(F)$에서 2.6배, $M_4^4(F)$에서 6.8배, $M_5^5(F)$에서 12.6배의 속도개선이 이루어져 불리언 행렬의 크기가 커질수록 훨씬 더 좋은 결과를 보이고 있다.

6. 결 론

Green 관계에 속한 J 관계 계산은 모든 불리언 행렬에 대한 이중 연속곱셈을 요구하여 근본적으로 기하급수적인 계산복잡도를 가진다. 따라서 J 관계 계산 알고리즘의 공간 및 시간 복잡도 개선은 근본적인 어려움을 내포하고 있다. 본 논문은

〈표 3〉 세부 실행시간 비교

행렬 크기		J 관계 계산 알고리즘 [+1]	L 관계 계산 최적화 알고리즘
2×2	R관계	0.008초	0.005초
	L관계	0.005초	0.005초
	J클래스	0.016초	0.011초
3×3	R관계	0.127초	0.119초
	L관계	0.123초	0.047초
	J클래스	0.109초	0.118초
4×4	R관계	18.886초	19.097초
	L관계	19.167초	2.839초
	J클래스	38.661초	39.12초
5×5	R관계	21,736.547초	21,807.719초
	L관계	15,902.807초	1,266.084초
	J클래스	158,307.99초	144,726.6초

$n \times n$ 불리언 행렬의 모노이드에서 R 관계와 L 관계가 서로 전치행렬 변환에 의해 얻어질 수 있음을 보이고, 이를 이용하여 L 관계 계산을 개선한 J 관계 계산 알고리즘을 제시하였다.

그러나 <표 3>에서 보는 것처럼 R 관계나 J 클래스 계산에 소요되는 시간은 개선되어 있지 않다. 따라서 아직 여러 부분에서 J 관계 계산 알고리즘의 최적화가 필요하며, 정규(regular) J-클래스를 신속히 계산할 수 있는 알고리즘, 분산 또는 병렬 알고리즘의 설계 및 구현 등에 대한 연구도 필요하다.

참 고 문 헌

[1] 김창범, 한재일, "Classifications of D-classes in the semigroup $M_n(F)$ of all nxn Boolean matrices over F = {0, 1}", 「퍼지 및 지능시스템학회논문지」, 제16권, 제3호(2006), pp. 338-348.

[2] 한재일, "모든 lxn, nxm, mxk 불리언 행렬 사이의 중첩곱셈에 대한 연구", 「한국IT서비스학회지」, 제5권, 제1호(2006), pp.191-198.

[3] 한재일, "효율적인 D-클래스 계산을 위한 알

고리즘", 한국IT서비스학회지, 제6권, 제1호 (2007), pp.151-158.

[4] 한재일, "불리언 행렬의 모노이드에서의 J 관계 계산 알고리즘", 한국IT서비스학회지, 제7권, 제4호(2008), pp.221-230.

[5] Howie, J. M., *An Introduction to semigroup theory*, Oxford University Press, 1995.

[6] Linton, S. A., G. Pfeiffer, E. F. Robertson, and N. Ruskuc, "Computing Transformation Semigroups", *Journal of Symbolic Computation*, Vol.33(2002), pp.145-162.

[7] Wall, J. R., "Green's relations for stochastic matrices", *Czechoslovak Mathematical Journal*, Vol.25, No.2(1975), pp.247-260.

[8] Heyworth, A., "One-Sided Noncommutative Grobner Bases with Applications to Computing Green's Relations", *Journal of Algebra*, Vol.242(2001), pp.401-416.

[9] Lee, L., "Fast context-free grammar parsing require fast Boolean matrix multiplication", *JACM*, Vol.49, No.1(2002), pp.1-15.

[10] Yi, X. et al., "Fast Encryption for Multimedia", *IEEE Transactions on Consumer Electronics*, Vol.47, No.1(2001), pp.101-107.

[11] Martin, D. F., "A Boolean matrix method for the computation of linear precedence functions", *CACM*, Vol.15, No.6(1972), pp.448-454.

[12] Nakamura, Y. and T. Yoshimura, "A partitioning-based logic optimization method for large scale circuits with Boolean matrx", *Proceedings of the 32nd ACM/IEEE conference on Design automation*, (1995), pp. 653-657.

[13] Pratt, V. R., "The Power of Negative Thinking in Multiplying Boolean matrices", *Proceedigs of the annual ACM symposium on Theory of computing*, (1974), pp.80-83.

[14] Rim, D. S. and J. B. Kim, "Tables of D-Classes in the semigroup B of the binary relations on a set X with n-elements", *Bull. Korea Math. Soc.*, Vol.20, No.1(1983), pp.9-13.

[15] Atkinson, D. M., N. Santoro, and J. Urrutia, "On the integer complexity of Boolean matrix multiplication", *ACM SIGACT News*, Vol.18 No.1(1986), p.53

[16] Yelowitz, L., "A Note on the Transitive Closure of a Boolean Matrix", *ACM SIGMOD Record*, Vol.25, No.2(1978), p.30.

[17] Comstock, D. R., "A note on multiplying Boolean matrices II", *CACM*, Vol.7 No.1 (1964), p.13.

[18] Macii, E., "A Discussion of Explicit Methods for Transitive Closure Computation Based on Matrix Multiplication", 29th Asilom Conference on Signals, *Systems and Computers*, Vol.2(1995), pp.799-801.

[19] Angluin, D., "The four Russians' algorithm for boolean matrix multiplication is optimal in its class", *ACM SIGACT News*, Vol.8, No.1(1976), pp.29-33.

[20] Booth, K. S., "Boolean matrix multiplication using only bit operations", *ACM SIGACT News*, Vol.9, No.3(1977), p.23.

[21] Cousineau, G., J. F. Perrot, and J. M. Rifflet, "APL programs for direct computation of a finite semigroup", *APL Congres*, Vol.73(1973), pp.67-74.

[22] Lallement, G., *Semigroups and Combinatorial Applications*, John Wiley and Sons, New York, 1979.

[23] Champarnaud, J. M. and G. Hansel, "AUTOMATE, a computing package for automata and finite semigroups", *Journal of Symbolic Computation*, Vol.12(1991), pp.197-220.

[24] Sutner, K., "Finite State Machines and Syntatic Semigroups", *The Mathematica Journal*, Vol.2(1991), pp.78-87.

[25] Lallement, G. and R. McFadden, "On the determination of Green's relations in finite transformation semigroups," *Journal of Symbolic Computation*, Vol.10(1990), pp.481-498.

[26] Konieczny, J., "Green's equivalences in finite semigroups of binary relations", *Semigroup Form*, Vol.48(1994), pp.235-252.

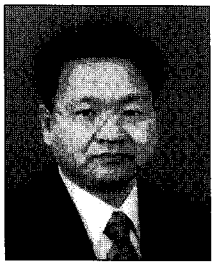
[27] Plemmons, R. J. and M. T. West, "On the semigroup of binary relations", *Pacific Journal of Mathematics*, Vol.35(1970), pp.743-753.

[28] Simon, I., "On semigroups of matrices over the tropical semiring", *Informatique Theorique et Applications*, Vol.28(1994), pp.277-294.

[29] Straubing, H., "The Burnside problem for semigroups of matrices", *in Combinatorics on Words, Progress and Perspectives, L. J. Cummings(ed.), Academic Press*, (1983), pp.279-295.

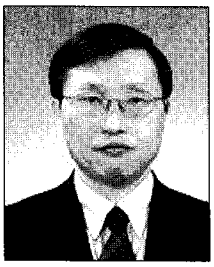
[30] Froidure, V. and J. Pin, "Algorithms for computing finite semigroups", *Foundations of Computational Mathematics*, (1997), pp. 112-126.

◈ 저 자 소 개 ◈



한 재 일 (jhan@kookmin.ac.kr)

연세대학교에서 이학사, 미국 Syracuse University에서 전산학 석사와 박사 학위를 취득하고, 국민대학교 컴퓨터학부 교수로 재직 중이다. 현재 분산처리, 객체지향 시스템과 RFID/USN 미들웨어를 연구 중이다. 관심분야는 서비스 사이언스, 서비스 컴퓨팅, 분산 시스템, 객체지향 시스템, 미들웨어, 컴퓨터 및 네트워크 보안, 지능형 시스템, 공개 소프트웨어 등이다.



김 영 만 (ymkim@kookmin.ac.kr)

서울대학교 기계공학과에서 학사, Ohio State University 전산과학과에서 석사와 박사학위를 취득하고, 국민대학교 컴퓨터공학과 교수로 재직 중이다. 현재 컴퓨터 네트워크와 미들웨어를 연구 중이다. 관심분야는 컴퓨터 및 네트워크 보안, SaaS 플랫폼, USN 등이다.