Rewrite System에서 다항식 순서의 자동생성

이 정 미† · 서 재 권†† · 위 규 범†††

요 약

우리는 등식으로 표현된 많은 정보들을 다룬다. 이러한 정보에 관한 가장 근본적인 문제중의 하나는 '어떤 주어진 등식이 한 가지 방법이다. Rewrite system은 주어진 항(term)에 rewrite 규칙(rule)들을 적용하여 단순화한다. 따라서 어떤 항이라도 단순화 과정이 무한히 계속되지 않아야 함은 rewrite system의 중요한 성질이다. Rewrite system의 이러한 종료(termination) 여부를 결정하는 방법들 중 하나가 다항식 순서(polynomial ordering)이다. 이 방법은 rewrite system의 함수기호에 적절한 다항식을 대응시켜주는 방법이다. 그러나, 주어진 rewrite system이 종료함을 보이는 다항식 순서를 자동적으로 생성하는 방법은 알려져 있지 않다. 본 논문에서는 유전자 알고리즘을 사용하여, 다항식을 자동으로 생성하는 방법을 제시한다.

Automatic generation of polynomial orderings in rewrite systems

Jung-Mi Lee† · Jae-Kwon Suh†† · Kyu-Bum Wee†††

ABSTRACT

Equations are widely used in representing information. One of the basic questions about equations is to determine whether a given equation follows logically from the set of equations. Rewrite systems are one of the method to answer many instances of this problem. A rewrite system simplifies a given term by applying rewrite rules successively. Hence it is important that the process of simplification does not go on indefinitely. One of the methods to check whether a rewrite system terminates (that is, the rewrite system does not simplify any term indefinitely) is polynomial orderings. A polynomial ordering assigns an appropriate polynomial to each function symbol. However, how to assign polynomials to function symbols is not known. We propose an automatic way of generating polynomial orderings using genetic algorithms.

1. 서 론

우리는 등식으로 표현된 많은 정보들을 다룬다. 그 중에서 가장 중요하고 근본적인 문제중의 하나는 '어떤 주어진 등식이 그 등식체계로부터 연역되어질 수 있는가를 결정하는 것이다. 이러한 문제의 많은 경우에 해답을 줄 수 있는 방법 중 하나가 rewrite system이다. Rewrite system은 등식체계에 적용되어 질 수 있는 rewrite 규칙(rule)들의 집합이다. 주어진 식을 rewrite 규칙들에 따라 전개해 나가는 과정을 유도(derivation)라 하며, Rewrite system이 무한(infinite)한 유도를 하지 않을 때 'rewrite system이 종료 (terminate)한다'고 정의한다. '주어진 rewrite system이 종료하는가'를 결정하는 것은 중요한 문제이다. 이 문제를 풀이하는 여러 방법들 중 하나가 다항식 순서(polynomial ordering)이다. 이 방법은 등식체계내의 함수 기호(function symbol)들에게 적절한 다항식들을 대응시킴으로써 '주어진 rewrite system이 종료하는가'에 대한 답을 준다. 그러나 적절한 다항식들을 찾기 위한 정형화된 방법이

† 정 회 원 : LG-Hitachi
†† 정 회 원 : 충북대학교 대학원 전자계산학과
††† 종신회원 : 아주대학교 정보 및 컴퓨터공학부 교수
논문접수 : 1999년 6월 9일, 심사완료 : 1999년 8월 23일

알려져 있지 않다. 본 논문에서는 다항식들의 생성과 대응을 유전자 알고리즘(genetic algorithm)을 이용하여 해결해 본다. 본 논문은 다음과 같이 구성된다. 제 2 장과 제 3 장에서는 rewrite system, 종료(termination), 다항식 순서와 다항식 순서의 구현에 관한 이전 연구를 소개한다. 제 4 장에서는 유전자 알고리즘을 소개한다. 제 5 장에서는 유전자 알고리즘을 이용하여 다항식 순서를 만족하는 다항식 해석기(polynomial interpreter)를 생성하기 위한 생성기의 설계와 구현에 관한 내용과 함께, 적용 결과에 대해 설명한다.

2. Rewrite systems

Rewrite system이란 rewrite 규칙들의 집합이다. Rewrite 규칙이란 두 개의 항(term)의 짝으로, $s \rightarrow t$으로 표시한다. 어떤 항 M이 rewrite 규칙 $s \rightarrow t$에 의하여 다른 항 N으로 단순화(rewrite, reduce, 또는 simplify)된다는 것은 s가 M의 어떤 부분항(subterm)과 match할 때 (다시 말해서, 어떤 치환 σ에 대하여 $\sigma(s)$가 그 부분항과 일치할 때) 그 부분항을 $\sigma(t)$로 치환한 것이 N이다. 이때 $M \Rightarrow N$으로 표기하며, $\Rightarrow$의 reflexivetransitive closure를 $\Rightarrow^*$로 표기한다. Rewrite system 안에 있는 어떤 rewrite 규칙도 항 t를 단순화할 수 없을 때 t를 irreducible이라 한다. $t \Rightarrow^* u$이고, u가 irreducible이면 u를 t의 정규형(normal form)이라 한다.

예를 들면, rewrite system $R = \{h(a) \rightarrow b, f(b, X) \rightarrow g(X, X)\}$에 의해서 다음의 유도(derivation)가 가능하다 :

$$\begin{aligned} & k(f(h(a), f(a, Y)), f(X, b)) \\ \Rightarrow\ & k(f(b, f(a, Y)), f(X, b)) \\ \Rightarrow\ & k(g(f(a, Y), f(a, Y)), f(X, b)). \end{aligned}$$

(그림 1) 유도의 예

Rewrite system이 종료한다는 것은 어떠한 항에 대해서도 무한 유도가 없다는 뜻이다. Rewrite system이 합류적(confluent)이라는 것은 $M \Rightarrow^* N_1$이고 $M \Rightarrow^* N_2$일 때 어떤 P가 존재하여 $N_1 \Rightarrow^* P$와 $N_2 \Rightarrow^* P$를 만족함을 뜻한다. 종료는 정규형이 존재함을 의미하고, 합류(confluence)는 정규형이 유일(unique)함을 의미한다. 종료하고 합류적인 rewrite system을 완전(complete) 하다고 부른다.

등식체계의 각 등식에 방향을 주어서 rewrite 규칙으로 변환하면 등식체계는 rewrite system이 된다. 만약 이 rewrite system이 완전하다면, word problem을 해결하는 도구로 사용할 수 있다. Word problem 이란 주어진 등식이 주어진 등식체계의 논리적 결과로서 도출될 수 있는가를 결정하는 문제이다. 주어진 등식의 양변의 항을 각각 정규형으로 단순화하여 그 두 개의 정규형이 동일하면 주어진 등식은 등식체계의 논리적 결과이고, 동일하지 않으면 논리적 결과가 아니다. 만약 이 rewrite system이 완전하지 않으면, rewrite 규칙들을 추가하여 완전하게 만드는 것이 가능한 경우도 있다. 이러한 완전화 절차의 대표적인 것이 Knuth-Bendix 완전화 절차(completion procedure)이다[1].

예를 들어서, 군(group)의 공리(axiom)들은 오른쪽과 같다 : 여기서 e는 좌항등원(left identity)이며, i는 좌역원(left inverse)를 나타낸다.

$$\begin{aligned} &(1)\ *(e, X) = X \\ &(2)\ *(i(X), X) = e \\ &(3)\ *(*(X, Y), Z) = *(X, *(Y, Z)) \end{aligned}$$

(그림 2) 군의 예

위의 등식체계를 아래와 같은 rewrite system으로 변환할 수 있다 :

$$\begin{aligned} &(1)\ *(e, X) \rightarrow X \\ &(2)\ *(i(X), X) \rightarrow e \\ &(3)\ *(*(X, Y), Z) \rightarrow *(X, *(Y, Z)) \end{aligned}$$

(그림 3) Rewrite system 의 예

위의 rewrite system은 완전하지 않다. 이것이 Knuth-Bendix 완전화 절차에 의해서 아래와 같은 완전한 체계로 변환된다:

$$\begin{aligned} & i(e) \rightarrow e \\ & *(e, X) \rightarrow X \\ & *(X, e) \rightarrow X \\ & i(i(X)) \rightarrow X \\ & *(i(X), X) \rightarrow e \\ & *(X, i(X)) \rightarrow e \\ & *(X, *(i(X), Y)) \rightarrow Y \\ & i(*(X, Y)) \rightarrow *(i(Y), i(X)) \\ & *(*(X, Y), Z) \rightarrow *(X, *(Y, Z)) \end{aligned}$$

(그림 4) (그림 3)과 동치인 완전한 rewrite system

위의 완전한 체계는 주어진 등식이 군의 공리로부터 논리적 결과로서 성립하는지 아닌지 판정하는 척자로서 사용될 수 있다. 예를 들어서, $i(*(X,i(Y)),Z) = *(*(i(Z),Y),i(X))$이 성립하는가 살펴보자. 위의 등식의 좌변 항과 우변 항은 각각 아래와 같이 단순화된다 :

$$i(*(X,i(Y)),Z) \Rightarrow *(i(Z),i(*(X,i(Y)))) \Rightarrow$$
$$*(i(Z),*(i(i(Y)),i(X))) \Rightarrow *(i(Z),*(Y,i(X)))$$

$$*(*(i(Z),Y),i(X)) \Rightarrow *(i(Z),*(Y,i(X)))$$

좌변 항과 우변 항이 같은 항으로 단순화되므로, 다시 말해서, 두 개의 항의 정규형이 같으므로, 주어진 등식은 성립하는 것이다.

Rewrite system은 word problem 이외에도 자동증명[2], 소프트웨어 검증[3], 프로그램 합성[4] 등에 응용된다. 이러한 여러 문제에서 이용되는 rewrite system은 종료하는 시스템이어야 의미가 있다. 따라서 주어진 rewrite system이 종료하는가를 검사하는 방법은 매우 중요하다.

Rewrite system이 종료하는가를 검사하기 위해서는 well-founded partial ordering을 사용하며, 널리 쓰이는 방법들은 다항식 순서[5], recursive path ordering[6], paths of subterms ordering[7], recursive decomposition ordering[8], KNS ordering[9], Knuth-Bendix ordering[1] 등이다.

3. 다항식 순서 (Polynomial orderings)

3.1 다항식 순서의 소개

다항식 순서는 rewrite system에 등장하는 각 함수기호 $f(v_1, v_2, \ldots, v_N)$에 적절한 다항식 $p(x_1, x_2, \ldots, x_N)$을 대응시킨다. 변수(argument)의 개수(arity)가 0인 함수기호(function symbol)를 상수(constant)라고 부르며, 상수에는 적절한 양의 상수(차수가 0인 다항식)을 대응시킨다. 다항식들을 합성(composition)함으로서 임의의 항에 다항식이 대응된다. 이때 대응하는 다항식을 그 항의 다항식 해석 (polynomial interpretation) 이라고 부른다. 각 rewrite 규칙의 좌변항의 다항식 해석이 우변항의 다항식 해석보다 크면 그 rewrite system은 종료함을 쉽게 알 수 있다[5].

아래의 예는 3 개의 규칙으로 이루어진 rewrite system의 다항식 순서를 보여준다. 여기서 τ는 항을 다항식에 매핑(mapping)하는 다항식 해석을 나타낸다.

$$(1)\ \alpha\nabla(\beta\circ\gamma) \rightarrow (\alpha\nabla\beta)\circ(\alpha\nabla\gamma)$$
$$(2)\ (\beta\circ\gamma)\nabla\alpha \rightarrow (\beta\nabla\alpha)\circ(\gamma\nabla\alpha)$$
$$(3)\ (\alpha\circ\beta)\circ\gamma \rightarrow \alpha\circ(\beta\circ\gamma)$$
$$\tau : \alpha\nabla\beta \mapsto f(x,y) = xy$$
$$\tau : \alpha\circ\beta \mapsto g(x,y) = 2x+y+1$$
$$\tau : constants \mapsto h() = 2$$

$$(1)\ \tau(\alpha\nabla(\beta\circ\gamma)) - \tau((\alpha\nabla\beta)\circ(\alpha\nabla\gamma))$$
$$= x(2y+z+1) - [2xy+xz+1]$$
$$= x-1 \geq 2-1 > 0$$

$$(2)\ \tau((\beta\circ\gamma)\nabla\alpha) - \tau((\beta\nabla\alpha)\circ(\gamma\nabla\alpha))$$
$$= (2y+z+1)x - [2yx+zx+1]$$
$$= x-1 \geq 2-1 > 0$$

$$(3)\ \tau((\alpha\circ\beta)\circ\gamma) - \tau(\alpha\circ(\beta\circ\gamma))$$
$$= 2(2x+y+1)+z+1 - [2x+(2y+z+1)+1]$$
$$= 2x+1 \geq 5 > 0$$

(그림 5) 다항식 순서에 의하여 종료하는 예

임의의 rewrite system이 종료하는가를 판정하는 것은 결정불가능(undecidable)한 문제이다. 또한 종료하는 모든 시스템이 다항식 순서를 가지는 것도 아니다. (그림 6)은 종료하는 시스템이지만, 다항식 순서에 의해서는 종료함을 증명할 수 없는 rewrite system의 예이다[10]. 그러나, 다항식 순서를 만족하는 rewrite system은 반드시 종료한다.

$$-(-(X)) \rightarrow X$$
$$-(+(X,Y)) \rightarrow *(-(X),-(Y))$$
$$-(*(X,Y)) \rightarrow +(-(X),-(Y))$$
$$*(X,+(Y,Z)) \rightarrow +(*(X,Y),*(X,Z))$$
$$*(+(Y,Z),X) \rightarrow +(*(Y,X),*(Z,X))$$

(그림 6) 다항식 순서에 의하여 종료함을 증명할 수 없는 예

3.2 다항식 순서 구현에 관한 이전 연구

REVE system에는 입력으로 다항식 변환기(polynomial interpreter)가 주어지면, 해당하는 rewrite system이 다항식 순서를 만족하는지 혹은 만족하지 않는 지의 여부를 판단하는 모듈이 포함되어 있다. REVE는 프랑스의 Centre de Recherche en Informatique de Nancy에서 개발한 rewrite system 생성기이다[11]. 그 모듈에서는 다항식의 계수를 반복적으로 변화시키는 방법을 사용하여 다항식이 양인가를 판단하였다[12]. REVE

시스템의 사용자에 의해서 다항식 순서가 주어지고, 이 모듈은 주어진 다항식에 의해서 TL rewrite system이 종료하는가를 검사한다.

그러나 다항식 순서 자체를 자동으로 생성하는 이전 연구는 알려져 있지 않다. 본 논문에서는 다항식 순서를 자동으로 생성하는 방법을 제안한다.

다항식이 양인지를 판단하는 방법으로 eventually positive의 개념을 사용할 수 있다. Eventually positive인 다항식 해석은 상수(변수가 0 개인 함수기호)에 충분히 큰 정수를 할당함으로써 positive인 다항식 해석으로 변환할 수 있다[10]. Eventually positive의 순환적 정의는 아래와 같다. p를 변수(x_1, x_2, ..., x_n)로 이루어진 다항식이라 하자.

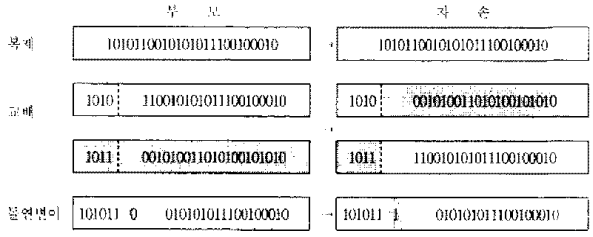
1) $p(x_1, x_2, ..., x_n)$이 positive 이면, eventually positive이다.
2) $\frac{\partial p}{\partial x_1}, \frac{\partial p}{\partial x_2}, ..., \frac{\partial p}{\partial x_n}$이 모두 eventually positive 이면,
   $p(x_1, x_2, ..., x_n)$도 eventually positive이다.

(그림 7) Eventually positive 의 순환적 정의

4. 유전자 알고리즘

유전자 알고리즘은 적자생존과 진화라는 생명 현상에서 나타나는 군집, 개체 (individual), 세대 (generation), 유전자 (gene), 염색체 (chromosome), 교배, 돌연변이 등의 개념을 이용하여 문제를 해결하는 방법이다. 주위의 환경에 잘 적응한 유전자로 구성된 염색체를 가지는 개체가 살아 남아 더 나은 개체로 진화해 가듯이, 주어진 조건에 잘 부합하는 답을 표현하는 문자열들로 구성되는 후보해들의 군집을 계속하여 진화시켜 우리가 원하는 답을 찾는다. 이러한 방법의 탐색 방법은 J. H. Holland에 의해 제안되었다[13]. 이 방법을 사용하기 위해서는 우선 주어진 문제의 답을 고정 길이의 문자열로 표현하여야 한다. 이렇게 인코딩 된 문자열을 염색체라 한다. 각각의 염색체의 문자열을 개체라 하고 개체의 집합을 군집이라 하며, 진화해 나감에 따라 변하게 되는 집합을 한 세대라고 한다. 초기 세대의 개체는 임의(random)로 생성한다. 적합도 평가 함수(fitness function)를 사용하여 개체가 얼마나 만족할 만한 지를 평가한 후, 유전 연산자(복제, 교배, 돌연변이)를 적용시켜 진화시켜 나간다. 단순 유전자 알고리즘 (simple genetic algorithm) 에서 사용되는 세 가지의 연산은 다음과 같다.

1) 복제 : 선택된 부모와 같은 형태의 자손을 생성한다.
2) 교배 : 선택된 부모로부터 두 자손을 생성한다. 임의로 교배 지점을 선택한 다음, 한 자손에게는 자신의 문자열의 앞부분을, 다른 자손에게는 뒷부분을 전달해준다.
3) 돌연변이 : 선택되어진 부모의 문자열 중 임의의 부분의 값을 변형시킨다. 대개의 경우 이 연산이 적용되는 비율은 0.001정도로 아주 작다.



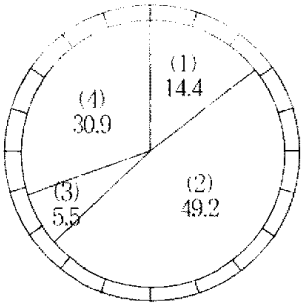
(그림 8) 단순 유전자 알고리즘에서 사용하는 세 가지 연산

세 가지의 연산은 모두 선택 (selection) 이라는 과정이 선행한다. 선택은 각 개체의 적합도 값에 근거해서 다음 세대로 전해질 유전자를 가지고 있는 개체를 선택하는 연산이다. 선택을 위한 방법으로 roulette wheel이라는 방법이 많이 쓰이는 데, roulette 게임처럼 구슬을 굴려 구슬이 들어간 칸에 있는 개체를 선택하는 방법이다. 각 개체는 roulette wheel상에서 적합도(fitness value)에 비례하는 면적을 차지한다. 적합도가 좋을수록 차지하는 면적이 넓어지고, 그만큼 선택될 확률이 높아지게 되는 방법이다.

〈표 1〉 문자열과 적합도의 예

번호	문 자 열	적 합도	전체에 대한 비율 (%)
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

네 개의 개체에 대해 적합도 평가 함수를 적용하여 <표 1>과 같은 결과를 얻었다면 <표 1>로부터 (그림 9)

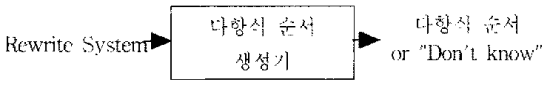


(그림 9) <표 1>의 내용으로 구성한 roulette wheel

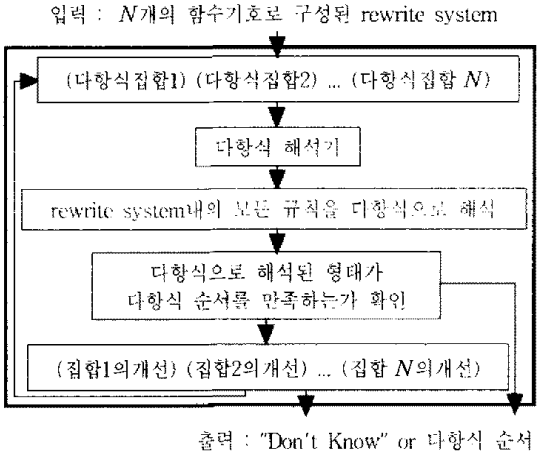
와 같은 roulette wheel을 구성할 수 있다. 이 외에도 roulette wheel을 개선한 stochastic universal sampling, rank selection, tournament selection 등이 있다 [14, 16]. 선택의 기본이 되는 적합도 값의 계산은 적합도 평가 함수에 의해 수행된다. 어떠한 개체가 주어진 조건을 완벽히 만족할 때, 'HIT'이라 하고 그 개체가 해가 된다. 적합도 평가 함수는 주어진 문제에서 요구하는 바가 인코딩 된 문자열에서 얼마나 잘 표현되는지를 평가할 수 있도록 정의되어야 한다.

5. 유전자 알고리즘을 이용한 다항식 순서의 생성

5.1 다항식 순서 생성기의 설계

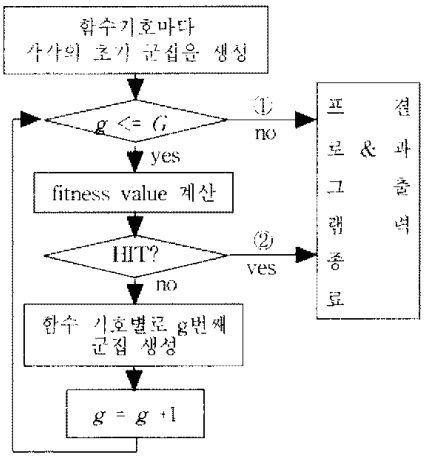


(그림 10) 설계



(그림 11) 생성기의 내부

생성기는 다음과 같이 동작한다. Rewrite system을 입력받으면, 함수기호의 수만큼 다항식의 군집을 생성한다. 하나의 군집은 하나의 함수기호에 대응하는 후보 다항식들의 집합이다.

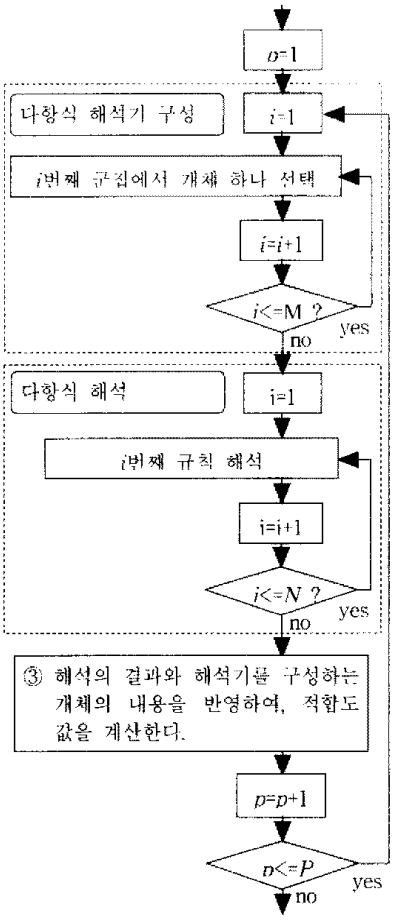


(그림 12) 생성기의 흐름도

각 군집으로부터 하나씩의 개체를 선택하여 다항식 해석기(polynomial interpreter)를 구성한 후, rewrite 규칙을 다항식의 형태로 해석한다. 해석한 결과가 다항식 순서의 정의를 만족하는 지 혹은 어느 정도 정의에 근접하는지 확인한다. 평가의 결과를 바탕으로 군집의 구성을 변화시킨다. 이러한 과정을 반복해서 수행하는 과정에서 다항식 순서를 만족하는 다항식 해석기를 찾는다.

(그림 11), (그림 12), (그림 13)에서 M은 rewrite system을 구성하는 함수 기호의 개수이고, G는 최대 세대수, N은 rewrite system의 규칙의 개수, P는 군집의 크기이다. 앞에서 설명하였듯이, 각 함수기호에 대하여 하나의 군집을 구성한다. 함수 기호에 관계없이 모든 군집의 크기는 P이다. 동일한 형태라 할지라도 서로 다른 군집에 속하면 다른 개체이다. 개체의 적합도 값은, 개체가 속한 군집이 어떤 군집인지, 개체가 어떠한 형태인지, 다항식 순서를 얼마나 만족하는지의 사항들을 반영하여야 한다. 각 함수 기호가 독립적으로 군집을 가지고 있으므로, rewrite 규칙을 다항식으로 해석하기 위해서는 다항식 해석기를 구성하는 일이 필요하다. 다항식 해석기는 각 군집으로부터 임의로 한 개의 개체를 선택하여 구성한다.

개체의 적합도 값을 계산하고 복수개의 군집이 공진화하는 과정을 그림으로 나타낸 것이 (그림 13)이다. 5.3



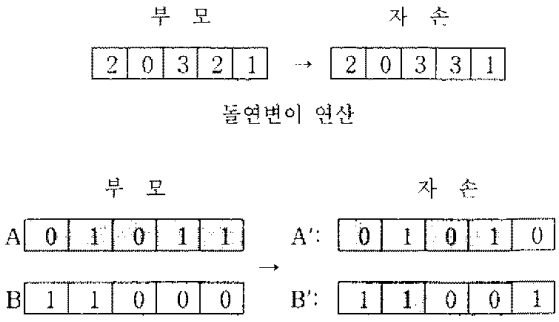
(그림 13) 적합도 값의 계산

절에서 실제로 적합도 값을 계산하는 함수(그림 13의 ③)에 대해 설명한다. 이러한 과정을 통해 M개의 군집은 공진화한다. 다항식 해석기 생성기를 구성하는 유전자 알고리즘에서 사용한 연산자는 복제 연산자와 돌연변이 연산자와 교배 연산자이다.

선택 방법으로는 roulette wheel 방법을 사용하였다. 복제 연산자는 부모 군집에서 선택된 개체와 동일한 형태의 개체를 자손 군집에 추가한다. 돌연변이 연산자는 유전자 하나의 값을 즉, 계수의 값을 1 증가시키거나 혹은 1 감소시키는 방법을 사용하였다.

교배 연산자는 uniform 교배 연산자를 사용하였다. Uniform 교배 연산은 부모 A, B의 유전자 위치 각각에 대하여 A의 자손으로 갈 것인지 B의 자손으로 갈 것인지를 정해진 확률에 의해서 결정하는 교배 방법이다. 여기서는 0.5의 확률을 사용했다. 각 위치에서 A의 유전자가 A의 자손으로 전해지면, 같은 위치의 B의 유전자는 B의 자손으로 전해지고, A의 유전자가 B의 자손으로 전해지면, 같은 위치의 B의 유전자는 A의 자손으로 전해진다.

교배연산이나 돌연변이 연산을 통하여 유효하지 않은 개체가 생겨난 경우에는, 그 개체를 무시하고 새로운 개체를 다시 생성하였다.



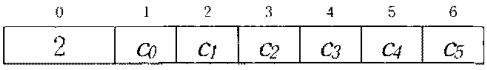
(그림 14) Uniform 교배연산

생성기가 종료되는 경우는 두 가지이다. 한 경우는 정의한 세대수만큼 진화가 이루어진 경우(그림 12의 ①)이고, 다른 하나는 다항식 순서를 만족하는 다항식 해석기를 찾아낸 경우(그림 12의 ②)이다.

5.2 개체의 정의

개체는 차수가 d인 x와 y의 다항식으로 제한하였다. d가 2인 경우 개체는 다음과 같이 표현된다.

$$c_5x^2 + c_4xy + c_3y^2 + c_2x + c_1y + c_0$$



(그림 15) 개체의 정의와 코드화

(그림 15)에서 볼 수 있는 바와 같이, 배열의 첫 번째 요소(인덱스가 0인 곳)는 다항식의 차수이다. 다항식의 계수 c_i 는 배열에서 인덱스가 $(i+1)$ 인 곳에 위치하게 된다. 차수가 2인 하나의 개체를 정의하기 위해서는 c_0에서 c_5까지 계수만 정해주면 된다. 일반적으로 c_0, ..., c_5는 모두 임의로 선택된 값을 가져야 하지만, 변수 두 개를 갖는 함수 기호의 군집을 구성할 때에는, x와 y의 차수가 모두 1 이상이어야 하고, 변수 하나를 갖는 함수 기호의 군집을 구성할 때는 x의 차수는 1 이상 y의 차수는 반드시 0 이어야 한다. 이는, 함수 기호의 변수 모두가 다항식 해석기에 반영되도록 다항식을 정의하기 위함이다.

개체 : | 2 | 0 | 3 | 2 | 0 | 0 | 1 |

개체가 표현하는 다항식 : $x^2+2x+3y$

(그림 16) 코드화한 개체의 예

이렇게 정의한 개체에 uniform 교배연산의 수행은 다음과 같이 이루어진다. 교배시킬 개체 A, B를 부모 군집에서 선택한다. 부모 개체의 유전자 중 계수 부분에 uniform 교배 연산을 적용하여 자손 개체 A', B'의 계수 부분을 결정한다. 자손 개체의 계수들로부터 다항식의 차수를 결정한다. 이때 A 와 B의 차수가 다른 경우에는 먼저 차수가 낮은 개체의 인코딩을 차수가 높은 인코딩으로 변경한 후에 교배연산을 시행하였다. 이 과정을 (그림 17)에 나타내었다.

A:

차수	다항식 : $xy+2x$					
2	0	0	2	0	1	0

B:

차수	다항식 : x^3+y^3+xy+1									
3	1	0	0	0	1	0	1	0	0	1

< 교배연산 전 >

A':

차수	다항식 : $xy+2x$									
	0	0	2	0	1	0	0	0	0	0

B':

차수	다항식 : x^3+y^3+xy+1									
	1	0	0	0	1	0	1	0	0	1

< 교배과정 중 >

A':

차수	다항식 : xy									
2	0	0	0	0	1	0	0	0	0	0

B':

차수	다항식 : $x^3+y^3+xy+2x+1$									
3	1	0	2	0	1	0	1	0	0	1

< 교배완료 >

(그림 17) uniform 교배연산의 예

5.2.1 초기 군집

초기 군집을 구성할 때 개체가 가질 수 있는 최대 차수(D)를 다항식 생성기의 변수로 입력받도록 하였다.

(1) $2x+3y$
(2) x^2+y^2+2
(3) $2x^2y$

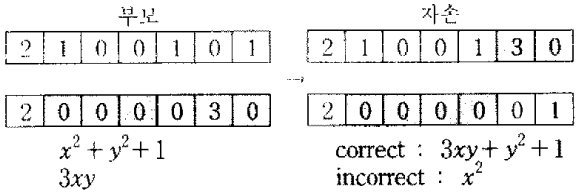
(그림 18) 초기군집에서 허용되는 다항식의 형태

초기 군집에서 개체의 차수는 1에서 D까지 골고루 나타나도록 하였다. 차수가 d인 개체의 계수를 정의할 때는 차수가 d가 아닌 항들의 계수는 0으로 하고, 차수가 d인 항들의 계수만을 0이 아닌 값으로 정의하였다. 계수의 최대값은 *MAXSUM*이다. 계수는 1에서 *MAXSUM*사이의 정수 중 하나를 임의로 취하는 방법으로 정의된다. 이렇게 생성한 각 개체에 0.5의 확률로 상수항을 추가시켜 주었다. (그림 18)은 초기 군집에서 허용되는 다항식의 형태의 예이다.

초기 군집에서는 모든 차수($1 \le d \le D$)가 동일한 비율로 나타나도록 구성하였다. 이는 초기 세대를 가장 단순한 형태의 다항식으로만 구성하기 위해서이다. 진화가 거듭되는 동안 서로 다른 형태의 다항식이 섞이게 되어, 다양하고 복잡한 형태의 다항식들이 등장하게 될 것이다.

5.2.2 자손 세대

초기 세대에서 정의된 형태의 다항식은 세대를 거듭하며, 다양한 형태로 변화해 나간다. 변화한 형태가 적합한 형태인가를 판단하는 작업이 필요하다. 변수가 두 개인 경우, 자손 세대를 구성할 때에는 형태의 구분에 관계없이 변수(x, y)가 모두 나타나면 올바른 개체로 인정하였다. 다항식 해석기에는 함수 기호의 변수가 모두 등장하여야 하기 때문에, 하나의 변수만 (x 또는 y)으로 구성되는 경우는 올바르지 않은 개체이다.



(그림 19) 적합하지 않은 개체의 예

올바르지 않은 개체라고 판단되면, 그 개체는 무시하고, 해당하는 개체를 재정의(redefine)하였다. 재정의 시에는 초기 군집을 구성하는 개체를 만드는 것과 동

일한 방법을 사용하였다. 적합하지 않은 개체를 버리고 다시 정의하는 과정을 통하여, 진화가 이루어지는 동안에도 부분적으로 새로운 개체들이 생겨나게 된다.

5.3 적합도 계산하기

각 함수 기호는 자신의 변수에 맞는 다항식으로 구성된 군집을 갖는다. 복수개의 군집으로부터 개체의 적합도 값을 계산하기 위해서는 먼저 다항식 해석기를 구성하여야 한다. 이 과정은 (그림 13)에서 보여지듯이, 각 군집으로부터 임의로 하나의 개체를 선택해 오는 작업으로 시작된다. 그런 후에, 각 rewrite 규칙을 다항식으로 해석한다. 해석된 다항식이 다항식 순서를 만족하는지 혹은 만족하지 않는지를 판단하고, 만족하지 않는 경우에는 어떠한 개체가 우월한지 평가한다. 다항식 순서를 만족하지 않는 경우에, 해당하는 개체가 다른 것에 비해 어느 정도 우월한가 평가하는 절대적인 방법은 존재하지 않는다. 본 논문에서는 아래와 같은 평가 척도를 사용하였다. 함수 기호2에서부터 함수 기호 M(M은 함수기호의 개수)까지의 각 군집에서 개체를 선택하여 완전하지 않은 다항식 해석기 I를 생성하자. 함수 기호 1의 군집에서 개체 p를 선택하고 I에 추가하여 생성된 다항식 해석기를 P, 함수 기호 1의 군집에서 개체 q를 선택하여 추가한 다항식 해석기를 Q라 하자. 동일한 군집에 속하는 개체 p, q와 그들이 속한 다항식 해석기 P, Q에 대해 p가 q보다 우월하려면 다음의 조건을 만족해야 한다.

1) P로 해석했을 때, 다항식 순서를 만족하는 규칙의 개수가 Q로 해석한 경우보다 많다.
2) 1)의 조건의 의해 p와 q가 같은 경우에는, P로 해석한 결과가 Q로 해석한 결과보다 최고차항의 계수가 양수인 개수의 비율이 크다. (항의 개수 비교)
3) 2)의 조건에 의해 p와 q가 같은 경우에는, P로 해석한 결과가 Q로 해석한 결과보다 최고차항의 계수가 양수인 항의 계수들의 합의 비율이 크다. (항의 계수 비교)
4) 3)의 조건의 의해 p와 q가 같은 경우에는, p의 차수가 q의 차수보다 작다.
5) 4)의 조건의 의해 p와 q가 같은 경우에는, p의 최고차항의 계수가 q의 최고차항의 계수보다 작다.

(그림 20) 비교 조건

이런 모두를 적합도 평가 함수에 반영하고자 하였다. 적합도 평가 함수에는 1), 2), 3), 4), 5)의 내용을 독립적으로 정의하여 포함시켰다. 적합도 계산 순서와 실제적인 함수의 정의에 앞서, 적합도 평가 함수 정의에 나타나는 상수와 함수들을 먼저 정리해 보았다.

M : 함수 기호의 수.
N : rewrite 규칙 수.
c_n : n번째 군집에서 선택된 개체 ($1 \le n \le N$).
C : c_n들의 집합으로 정의된 interpreter.
y_m : m번째 규칙을 C로 해석한 다항식 ($1 \le m \le M$).
Y : y_m들의 집합.
MAXORDER : 정의할 수 있는 최고 차수.
MAXSUM : 정의할 수 있는 최고 계수.
PI(C) : C로 정의된 다항식 해석이 다항식 순서를 만족하는 규칙의 수 ($0 \le PI(C) \le M$).
Order(p) : 다항식 p의 차수
CountTerm(p, d) : 다항식 p에서 차수가 d인 항의 개수
CountPTerm(p, d) : 다항식 p에서 차수가 d이고, 계수가
SumTerm(p, d) : 다항식 p에서 차수가 d인 항의 계수의 합
SumPTerm(p, d) : 다항식 p에서 차수가 d인 항 중 양수인 계수의 합
SumATerm(p, d) : 다항식 p에서 차수가 d인 항의 계수의 절대값의 합
LargestSum(p, d) : 다항식 p에서 차수가 d인 항들 중 최대 계수값

(그림 21) 적합도 평가 함수 정의에 사용되는 상수와 함수들

각 군집으로부터 한 개씩의 개체를 선택한다. C로 정의된 다항식을 사용하여 rewrite 규칙에 대응하는 다항식을 구하여 Y를 구성한다. Y로부터 c_n들의 적합도를 계산한다. 적합도를 계산하기 위한 식은 아래와 같다.

$$F(c_n) = w_1F_1(C) + w_2F_2(c_n) + w_3F_3(c_n) \quad (w_1 + w_2 + w_3 = 1)$$

$$F_1(C) = w_4F_4(C) + w_5F_5(C) + w_6F_6(C) \quad (w_4 + w_5 + w_6 = 1)$$

$$F_2(c_n) = \frac{MAXORDER - Order(c_n)}{MAXORDER}$$

$$F_3(c_n) = \frac{MAXSUM - LargestSum(c_n, Order(c_n))}{MAXSUM}$$

$$F_4(C) = \frac{PI(C)}{M}$$

$$F_5(C) = \frac{1}{M}\sum_{i=0}^{M}\frac{CountPTerm(y_i, Order(y_i))}{CountTerm(y_i, Order(y_i))}$$

$$F_6(C) = \frac{1}{M}\sum_{i=0}^{M}\frac{SumPTerm(y_i, Order(y_i))}{SumATerm(y_i, Order(y_i))}$$

(그림 22) 적합도 평가 함수의 정의

$PI(C)$는 다음과 같이 계산된다. Rewrite 규칙들을 C에서 정의하는 바에 따라 다항식으로 해석한다. 먼저 $PI(C)$의 값을 0으로 초기화한다. $Y=\{y_1, y_2, y_3, \ldots, y_M\}$의 y_m $(1\le m\le M)$이 다항식 순서를 만족하면, $PI(C)$값을 하나 증가시킨다. m값을 1에서부터 M까지 변화시켜가며, 같은 일을 반복한다. 따라서, $PI(C)$는 1에서 M사이의 값을 가지며, C가 다항식 순서를 만족시키는 다항식 해석기라면(모든 y_m이 다항식 순서를 만족하면) $PI(C)=M$이 된다. $F_1(C)$, $F_5(C)$, $F_6(C)$, $F_2(c_n)$, $F_3(c_n)$는 각각 (그림 19)에서 1), 2), 3), 4), 5)를 반영하고 있다. 적합도 함수의 정의에서, w_i $(1\le i\le 6)$ 값의 조정에 신중하여야 한다. $F_1(C)$, $F_5(C)$, $F_6(C)$가 모두 0 에서 1 사이의 값을 가지고 $w_1+w_2+w_3=1$로 정의하였으므로 $F(c_n)$가 갖는 값의 범위는 [0,1]이다.

함수기호에 다항식을 대응하고 나면, rewrite 규칙들은 다항식으로 해석하는 일이 필요하게 된다. 해석된 다항식들을 보고 종료 여부를 결정하게 된다. 여기서 rewrite 규칙의 좌변항에 대응하는 다항식이 좌변항에 대응하는 다항식보다 큰가를 확인해야 한다. 본 구현에서는 좌변 다항식에서 우변 다항식을 뺀 결과 다항식이 eventually positive 인가 심사하였다. 결과 다항식의 최고차항의 계수가 양수이면 eventually positive로 판정하였다.

5.4 결과

실험을 통해 여러 데이터에 대하여 비교적 좋은 진화를 이끄는 파라미터 값들을 찾아내었다. 그 결과를 아래 표에 나타내었다.

여기서 $MAXSUM$은 10이다. 입력 데이터로 rewrite system에 관한 문헌들에서 나타난 다양한 형태의 76

(표 2) 유전자 알고리즘의 파라미터

generations :	150
population size :	20
crossover rate :	0.8
reproduction rate :	0.2
mutation rate :	0.05
w_1:	0.92
w_2:	0.05
w_3:	0.03
w_4 :	0.75
w_5 :	0.15
w_6 :	0.10

개의 rewrite system을 사용하였다[1, 9, 16, 17, 18, 19]. 각 rewrite system은 독립적인 것으로, 서로 다른 interpreter를 사용하여 rewrite 규칙들을 다항식으로 해석한다.

본 논문에서 제안한 방법을 구현하여, 67개의 rewrite system에 대하여 다항식 순서를 자동으로 찾아내는데 성공하였다. 다른 9개의 rewrite system 중에는 종료하지 않는 rewrite system 하나와 다항식 순서로는 종료함을 증명할 수 없는 rewrite system이 2개 포함되어 있다. 개체의 최대 차수를 1에서 6까지 증가 시켜가며, 다항식 해석기를 생성하여 보았다. 대부분의 rewrite system에 대해서, 최대 차수가 1,2 차인 경우에 다항식 순서를 만족하는 다항식 해석기를 찾아 낼 수 있었다.

다음은 (그림 5)에서 소개한 rewrite system을 입력으로 하였을 때, 찾아낸 다항식 결과이다.

(1) $\alpha\nabla(\beta\circ\gamma) \rightarrow (\alpha\nabla\beta)\circ(\alpha\nabla\gamma)$
(2) $(\beta\circ\gamma)\nabla\alpha \rightarrow (\beta\nabla\alpha)\circ(\gamma\nabla\alpha)$
(3) $(\alpha\circ\beta)\circ\gamma \rightarrow \alpha\circ(\beta\circ\gamma)$

τ : $\alpha\nabla\beta \mapsto f(x,y)=2xy+x+y$
τ : $\alpha\circ\beta \mapsto g(x,y)=9x+y+9$

(1) $\tau(\alpha\nabla(\beta\circ\gamma))-\tau((\alpha\nabla\beta)\circ(\alpha\nabla\gamma)) = 9\alpha$
(2) $\tau((\beta\circ\gamma)\nabla\alpha)-\tau((\beta\nabla\alpha)\circ(\gamma\nabla\alpha)) = 9\alpha$
(3) $\tau((\alpha\circ\beta)\circ\gamma)-\tau(\alpha\circ(\beta\circ\gamma)) = 72\alpha+72$

(그림 23) 결과의 예 1

(1) $(X*Y)*Z \rightarrow X*(Y*Z)$
(2) $e*e \rightarrow e$
(3) $X*i(X) \rightarrow e$
(4) $f(e,X) \rightarrow X$
(5) $g((X*Y),Y) \rightarrow f((X*Y),X)$

$\tau : e \mapsto 1$
$\tau : \alpha * \beta \mapsto a(x,y) = 2x + y$
$\tau : i(\alpha) \mapsto b(x) = 4x$
$\tau : f(\alpha, \beta) \mapsto c(x,y) = x + 4y$
$\tau : g(\alpha, \beta) \mapsto d(x,y) = 3x + 3y$

(1) $\tau((X*Y)*Z) - \tau(X*(Y*Z)) = 2X$
(2) $\tau(e*e) - \tau(e) = 2$
(3) $\tau(X*i(X)) - \tau(e) = 6X-1$
(4) $\tau(f(e,X)) - \tau(X) = 3X+1$
(5) $\tau(g((X*Y),Y)) - \tau(f((X*Y),X)) = 5Y$

(그림 24) 결과의 예 2

6. 결론 및 향후과제

등식으로 표현된 많은 정보들을 다루기 위한 도구로서 rewrite system을 사용한다. Rewrite system에서의 중요한 문제 중 하나는 '주어진 rewrite system이 종료하는가'하는 것이다. 종료 여부를 결정하기 위한 여러 방법들이 있다. 그 중에서 다항식 순서라는 방법은 함수 기호에 적절한 다항식을 대응시켜 주는 방법이다. 유전자 알고리즘을 이용하여 각 함수 기호에 대응하는 다항식을 자동적으로 구하는 시도를 하였다. 본 논문에서 제안한 방법을 다양한 rewrite system에 적용하여 많은 경우에 다항식 순서를 성공적으로 찾아내었다.

향후과제로는 다음과 같은 것을 생각해 볼 수 있다. 본 논문에서 제안된 방법을 등식의 자동증명 시스템에서 rewrite system이 종료하는지 테스트하는 모듈로서 사용해 보는 것이다. 다른 하나는 생성기 구현의 변화이다. 함수 기호의 변수 개수에 제한이 없는 시스템으로 확장하는 것도 요구된다. 선택 방법과 유전 연산자의 변경이 성능에 개선을 가져올 수도 있으리라 생각된다.

참 고 문 헌

[1] D. E. Knuth and P. B. Bendix, Simple word problems in universal algebra, In J. Leech, editor. "Computational Problems in Abstract Algebra," pp.263-297, Pergamon Press, Oxford, 1970.

[2] J. Hsiang and N. Dershowitz, "Rewrite methods for clausal and non-clausal theorem proving," Proc. 10th Internet. Colloq. on Automata, Languages and Programming, pp.331-346, 1983.

[3] J. V. Guttag, E. Horowitz, and E. R. Musser, "Abstract data types and software validation", Comm. ACM, Vol.21, pp.1048-1064, 1978.

[4] N. Dershowitz, "Computing with rewrite systems, Information and control", Vol.65, pp.122-157, 1985.

[5] D. S. Lankford, "On proving term rewriting systems are Noetherian," Technical Report Memo MTP-3,8 Mathematics Department, Louisiana Tech, University, Ruston, LA, 1979.

[6] N. Dershowitz. "Orderings for term rewrite systems." Theoretical Computer Science, Vol.17, pp.279~301, 1982.

[7] D. A. Plaisted, "Recursively defined ordering for probing termination of term rewriting systems," Technical Report, R78-943, Dept. of Computer Science, Univ. of Illinois, Univ. IL, 1978.

[8] P. Lescanne, "On the recursive decomposition ordering with lexicographic status and other related orderings," Journal of Automated Reasoning, Vol.6, pp.39-49, 1990.

[9] D. Kapur, P. Narendren, and G. Sivakumar, "A path ordering for proving termination of term rewriting systems," Proc. 10th Colloq. Trees in Algebra and Programming, pp.173-185, 1985.

[10] N. Dershowitz, "Termination of Rewriting," Journal of Symbolic Logic 3, pp. 69-116, 1987.

[11] P. Lescanne, "Computer Experiments with the REVE Term Rewriting System Generator," Proceedings of the 10th ACM Symposium on Principles of Programming Languages, pp.99-108, Austin, TX.

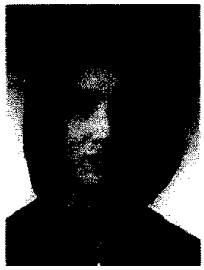
[12] A. B. Cherifa and P. Lescanne, "An actual implementation of a procedure that mechanically proves termination of rewriting systems based on inequalities between polynomial interpretations," Proceedings of the 8th Conference on Automated Deduction, Oxford, England, 1986.

[13] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[14] D. E. Goldberg, 'Genetic Algorithms in Search, Optimization and Machine Learning,' Addison-Wesley, 1989.

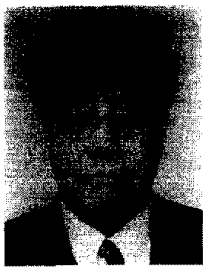
[15] M. Mitchell, 'An Introduction to Genetic Algorithms,'

MIT press, Cambridge, MA, 1996.
[16] G. Higman and B. Neumann, "Groups as groupoids with one law," Publ. Math. Debrecen., Vol.2, pp.215-227, 1952.
[17] D. Kapur and G. Sivakumar, "Experiments with RRL, a rewrite rule laboratory," Proceedings of NSF Workshop on Rewrite Rule Laboratory, pp.33-56, Schenectady, New York, 1984.
[18] A. MIddeldorp and Y. Toyama, "Completeness of combinations of constructor systems," Proceedings of the Fourth International Conference on Rewriting Techniques and Applications, pp.188-199, Como, Italy, 1991.
[19] M. Rusinowitch, "Path of subterms ordering and recursive decomposition ordering revisited," Journal of Symbolic Computation, Vol.3, pp.117-131, 1987.



이 정 미

e-mail : EOS@shinbiro.com
1996년 아주대학교 정보과학과 (학사)
1998년 아주대학교 대학원 컴퓨터공학과 (석사)
1999년~현재 LG-Hitachi
관심분야 : 컴퓨터이론, 유전자 알고리즘



서 재 권

e-mail : SUHJAEK@chollian.net
1979년 성균관대학교 수학과 (학사)
1983년 Indiana University 전자계산학과 (석사)
1997년~현재 충북대학교 전자계산학과 박사과정
관심분야 : logic database, fault-tolerant computing

위 규 범

e-mail : kbwee@madang.ajou.ac.kr
1978년 서울대학교 수학과(학사)
1985년 University of Wisconsin-Madison 전산학과(석사)
1992년 Indiana University 전산학과(박사)
1993년~현재 아주대학교 정보 및 컴퓨터공학부 교수
관심분야 : 컴퓨터 이론