분리가능 시스템의 지수 추이성과 변환

(Index Transitivity and Transformation of Separable Systems)

변 석 우 †

(Sugwoo Byun)

요 약 분리가능 시스템은 람다 계산법의 분리가능성을 따라서 정의된 항 개서 시스템이다. 본 연구에서는 기존 구성자 시스템에 한정되어 연구되었던 분리가능 시스템의 정의를 일반 항 개서 시스템으로 확장하고, 이 시스템의 특징을 전진 분지성 및 지수 추이성과 연관하여 설명한다.

분리가능성은 전진 분지성과 동일하며, 지수 추이성을 만족하는 강 순차성은 분리가능성을 갖는다. 이러한 특성에 따라 분리가능 시스템의 패턴 매칭 과정은 지수 트리 형태의 오토마타로 표현될 수 있고, 단순한 패턴을 갖고 있는 구성자 시스템으로 변환될 수 있는 장점을 갖고 있다. 특히, 분리가능 시스템은 람다 계산법으로 번역될 수 있다. ML이나 Haskell과 같은 함수형 언어들은 분리가능 시스템의 한 부류이므로, 본 연구는 함수형 언어의 의미를 람다 계산법으로 설명할 수 있도록 하는 이론적 기반으로도 이용될 수 있다.

키워드 : 람다 계산법, 항 개서 시스템, 순차성, 분리성

Abstract Separable systems are defined in term rewriting systems, respecting the notion of separability in the λ-calculus. In this research, we generalize separable systems of term rewriting systems, which was studied in restrictive systems such as constructive systems. We also associate separability with index-transitivity and with forward branching.

Separability is identified with forward branching, and strong sequentiality with index-transitivity satisfies separability. These are such good properties that enable us to describe the procedure of pattern-matching as an index tree, which is a sort of automata, and to transform separable systems into a constructor system with a simple pattern. Separable systems, in particular, can be translated into the λ-calculus. This research can serve a theoretical basis which allows functional languages to be explained by the λ-calculus, since functional languages such as ML and Haskell belong to a subclass of separable systems.

Key words : the lambda calculus, term rewriting systems, sequentiality, separability

1. 서 론

분리가능(separable) 시스템은 람다 계산법(the lambda calculus)의 분리가능성 (separability)을 따라서 정의된 항 개서 시스템(TRS, term rewriting systems)이다. 본 연구에서는 기존 *구성자(constructor)* 시스템에 한정되어 연구되었던 분리가능 시스템[1,2]의 정의를 일반 항 개서 시스템으로 확장하고, 이 시스템의 특징을 *전진 분지성(forward branching)* 및 *지수 추이성(index transitivity)*과 연관하여 설명한다.

람다 계산법의 분리가능성은 다음과 같다. 람다 텀들의 집합 $T = \{M_1, \cdots, M_p\}$가 주어졌을 때, 임의의 람다 텀 N_i $(1 \le i \le p)$에 대해서, $FM_1 = N_1 \wedge \cdots \wedge FM_p = N_p$, (여기서 $\wedge$은 논리적 and를 의미함) 을 만족하는 람다 텀 F가 존재하면 T는 분리가능하다고 한다. 분리가능성 이론에서는 "T 원소들의 '값(value)'이 서로 '다름(distinct)' ⇔ T가 분리 가능함"의 원리가 성립한다. 특히, T가 다른 값들로서 구성되었을 경우, 위의 등식을 만족시키는 람다 텀 F는 Böhm-out 변환 이론에 근거한 알고리즘에 의해서 찾아 질 수 있다[3].

람다 계산법 분리가능성 등식의 구조를 살펴보면, F는 *연산자(operator)*의 역할을 하며 M_i들은 F에 대한

· 이 논문은 2001학년도 경성대학교 학술지원 연구비에 의하여 연구되었습니다.

† 정 회 원 : 경성대학교 컴퓨터과학과 교수
swbyun@ks.ac.kr

논문접수 : 2003년 8월 4일
심사완료 : 2004년 2월 5일

입력 인수의 역할을 한다. 따라서 분리가능성을 갖는 항 개서 시스템은 연산자의 인수들이 '다른 값'을 갖는 조건을 만족해야 한다. 항 개서 시스템의 값에 대해서는 정형적인 논의가 이루어져야 하지만, 최소한 구성자들로서 형성된 항(term)들은 값으로 생각할 수 있다. 이런 이유 때문에 앞선 연구에서는 구성자 시스템을 중심으로 연구를 진행하였다.

본 연구에서는 분리가능 시스템에 대한 정의를 구성자 시스템이 아닌 일반적인 경우로 확대한다. 또한, 전진 분지성 이론[4,5]을 적용함으로써 분리가능성 및 변환 기술을 좀 더 체계적이고 구체적으로 설명한다. 분리가능성은 전진 분지성과 동일하며, 지수 추이성을 만족하는 강 순차성은 분리가능성을 갖는다. 이러한 특성에 따라 분리가능 시스템의 패턴 매칭 과정은 *지수 트리(index tree)* 형태의 오토마타로 표현될 수 있고, 단순한 형태의 구성자 시스템으로 변환될 수 있으며, 람다 계산법으로 번역될 수 있다. 본 연구에서 소개되는 변환 기법은 기존의 단순한 기법[6]보다는 좀 더 정형적이고 진보된 특성을 갖는다. 분리가능 시스템은 람다 계산법으로 번역 가능한 시스템을 탐구하기 위한 목적으로 정의되었지만, 람다 계산법으로의 번역 외에, 지수 추이성, 전진 분지성, 변환 등의 좋은 장점들을 가지고 있으므로 중요한 항 개서 시스템의 역할을 할 수 있다.

흔히 ML이나 Haskell과 같은 "함수형 언어는 람다 계산법의 구문적 치장"이라고 일컬을 정도로 함수형 언어와 람다 계산법이 밀접한 관계를 강조하고 있으나[7], 아직까지 함수형 언어의 람다 계산법으로의 번역을 정형적으로 제시한 적은 없다. 함수형 언어는 구성자 시스템의 한 부류이므로, 본 연구는 함수형 언어의 의미를 람다 계산법으로 설명하는 이론적 기반으로 이용될 수 있다.

본 논문에서는 독자들이 항 개서 시스템에 대한 기본적인 지식을 갖고 있음을 전제로 한다(자세한 관련 내용은 [8][9]을 참조할 것). 이미 소개된 내용들에 대해서는 요점만 간략히 설명하고, 본 논문의 핵심 내용에 대해서는 정형적인 방법으로 서술한다. 2절에서는 항 개서 시스템의 기초적인 사항을 간략히 소개하며, 3절에서는 call-by-need 계산의 원리에 대해서 설명한다. 4절에서는 분리가능성과 강 순차성 사이의 연관성을, 5절에서는 분리가능성과 전진 분지성 사이의 연관성을 논의한다. 6절에서는 분리가능 시스템의 변환에 대해서 논의하고, 7절에서 결론을 제시한다.

2. 항 개서 시스템 및 기초

TRS의 기호는 x, y, z 등으로 표현되는 변수들의 집합과 F, G 등으로 기술되는 함수기호들의 집합으로 구성된다. TRS의 항은 변수들과 함수기호를 이용하여 재귀적으로 정의된다. 모든 변수는 항이 될 수 있다. n개의 인수를 갖는 함수기호 F와 항 $t_1, \cdots, t_n$이 주어질 때 $F(t_1, ..., t_n)$ 또한 항이 된다. 개서 룰(rewrite rule)들은 두 항의 쌍 (l, r) 로 구성되는데, $l \rightarrow r$의 형태로 표기한다. 이때, l은 변수가 될 수 없으며, r에 나타나는 모든 변수들은 반드시 l에도 나타난다. 항 개서 시스템은 기호, 항, 룰의 세 가지로서 정의되지만, 대부분의 경우 같은 기호들을 사용하므로, 이들에 대해서는 언급을 생략한 채 항 개서 시스템을 룰들의 집합으로서 표현하는 경우가 많다. 또한 항 개서 시스템 대신 간략히 *시스템*이라고 부른다.

TRS의 함수 기호는 연산자(operators, 혹은 defined symbols)와 구성자(constructors)로 구분된다. 룰 왼쪽 항의 맨 왼쪽에 나타나는 기호는 연산자이며, 그렇지 않는 함수 기호들은 모두 구성자이다. 구성자 TRS는 연산자가 룰 왼쪽 항의 맨 왼쪽 외에는 나타나지 않는 시스템이다(즉, 연산자가 왼쪽 항의 인수로서 나타나지 않도록 정의된 시스템이다).

TRS의 항은 흔히 트리로서 표현된다. 어떤 한 항 $F(t_1, ..., t_n)$이 주어졌을 때 이 항에 대한 트리는 뿌리(root)가 F이고 각 서브텀(subterm, 혹은 부분식) $t_1, ..., t_n$에 대한 트리가 F의 서브트리(subtree)가 되는 재귀적인 방법으로 구성된다. 항의 '주소(position)'는 그 항의 트리 구조에 따라 쉽게 설명될 수 있다. 항 $F(t_1, ..., t_n)$에 대해서 주소 $<>$는 항의 뿌리인 F가 위치한 곳을 의미하며, $i(1 \le i \le n)$는 i 번째 노드를 의미한다. 또한, u가 항 t_i의 주소이면 $i \cdot u$는 항 $F(t_1, ..., t_n)$의 주소이다. u가 t의 주소일 때 $t|_u$ 는 u에 위치한 t의 서브텀을 의미한다. 즉, $t|_{<>} = t$ 이며, $F(t_1, \cdots, t_n)|_{i \cdot u} = t_i|_u$.

한 룰 $l \rightarrow r$에서 l에 어떤 변수도 두 번 이상 중복되어 나타나지 않으면 *좌선형(left-linear)*이라 하며, 시스템 R의 모든 룰들이 좌선형인 경우 R 또한 좌선형이라 부른다. 두 룰의 왼쪽에 위치한 항 s와 t가 주어졌을 때 (여기서 s와 t는 같은 항이 될 수 도 있다), 이 중 한 항이 다른 항의 서브텀과 단일화(unification)가 될 수 있으면 이 룰들은 *중복(overlap)*되었다고 한다. 한 시스템이 모든 임의 두 룰에 대해서 중복되지 않으면 그 시스템은 *비중복(non-overlapping)*되었다고 한다. 한 시스템이 좌선형이고 비중복일 때 그 시스템은 *직교적(orthogonal)*이라고 한다.

룰의 왼쪽 항이 실례화된(instantiation) 것을 *레덱스(redex)*라고 하며, 레덱스를 포함하지 않는 항은 *정규형(normal form)*이라 부른다. OTRS(orthogonal term rewriting systems)인 경우 한 레덱스를 결정하는 룰은 반드시 유일하게 존재한다.

두 식 t와 t'이 주어졌을 때, t에 유일하게 나타나는 어떤 변수에 대해서 t'이 t의 실례화(instance)인 경우 t는 t'의 *상부(prefix)*이라고 일컫는다. t와 t'의 관계는 이들을 트리로 표현하였을 때 t와 t'이 트리의 루트를 포함한 윗부분을 공유하는 모습을 갖는다. t'이 변수인 경우 t'에 대한 상부는 empty이며, 룰 왼쪽 항에 대한 상부는 *레덱스 상부(redex prefix)*라고 부른다. 한 식 t의 *구성요소(components)*는 t의 부분식의 상부다. *구성자 스키마(constructor schema)*는 룰 왼쪽 항의 변수가 아닌 *진 부분식(proper subterm)*이거나, 룰의 왼쪽 항에는 나타나지 않는 기호 C에 대해서 $C(x_1, ..., x_n)$인 경우이다. *구성자 정규형(constructor normal forms)*은 구성자 스키마에서 변수들을 구성자 정규형으로 치환한 것이다. 예를 들어, 왼쪽부분이 $F(F(A(B(x))))$일 때, 레덱스 상부는 $F(F(A(x)))$, $F(F(x))$, $F(x)$이고, 구성자 스키마는 $F(A(B(x))$, $A(B(x))$, $B(x)$이다. $F(F(A(B(x))))$의 구성요소(components)는 레덱스 상부, 구성자 스키마 외에 $F(A(x))$와 $A(x)$가 있다. $F(x)$는 $F(F(A(B(x))))$의 구성요소로서 두 번 나타난다.

3. Call-by-need 계산

항 t의 여러 레덱스들 중에서, 어떤 한 레덱스 d를 축약하기 않고는 t의 정규형을 계산하지 못할 경우 d는 *요구되는 레덱스(needed redex)*라고 부른다. call-by-need 계산은 오직 요구되는 레덱스들만을 축약하면서 계산함을 의미한다. call-by-need 계산의 중요성은 이 계산 방법이 *일 단계 정규화 전략(one step normalizing strategy)*의 존재를 의미하기 때문이다. 한 식의 계산은 *축약(reduction)*의 각 단계별로 요구 레덱스를 선택하여 계산함으로써 정규형을 얻을 수 있다.

Call-by-need 계산은 중요한 특성이지만 이 원리를 적용할 수 있는 시스템은 제한적이다. 일반적으로 OTRS에서는 call-by-need 계산을 적용할 수 없다. 예를 들어, 다음과 같이 Berry가 정의한 함수를 고려하자. $F(A, B, x) \rightarrow 1$, $F(B, x, A) \rightarrow 2$, $F(x, A, B) \rightarrow 3$. 이 시스템은 OTRS이다. 이때, R_1, R_2, R_3 세 개의 레덱스를 포함하는 항 $F(R_1, R_2, R_3)$에 대해서 어떤 레덱스가 요구되는 지를 결정할 수 없다. 왜냐 하면, 첫 번째 룰에 대해서 R_1이 A이고 R_2가 B일 때 R_3는 계산하지 않아도 되며, 두 번째 룰과 세 번째 룰을 고려하면 같은 이류로 R_1과 R_2를 축약하지 않고도 정규형에 도달할 수 있으므로, 결과적으로 세 개의 레덱스 중에서 요구되는 레덱스는 존재하지 않기 때문이다.

레덱스의 요구성(neededness) 판단은 룰 구조 (왼쪽 및 오른쪽)에 의해서 결정된다. 이와 같은 방법으로 레덱스의 요구성을 판단할 수 있는 시스템을 *순차(sequential)* 시스템이라고 한다. 그러나 순차 시스템에서 레덱스의 요구성을 결정하는 알고리즘은 룰의 왼쪽과 오른쪽을 모두 고려해야 하므로 높은 복잡도(complexity)를 갖는 문제를 갖고 있다. 이를 개선하는 한 방법으로서, 룰의 왼쪽만을 고려하여 요구성을 판단하는 방법이 제시되었는데, 이런 특성을 갖는 시스템을 *강 순차(strong sequential)* 시스템이라고 한다. 강 순차성에 대한 자세한 논의는 [10]과 [11]을 참조하기 바라며, 여기에서는 분리가능성과 연관된 사항들에 대해서만 기술한다.

정의 3.1. (1) 새로운 두개의 함수 기호 Ω와 $\bullet$를 추가 한다. 한 항 t의 *Ω-실례화*는 t의 Ω를 임의의 항으로 치환한 것이다.

(2) 항 t에서 $t|_u$는 Ω가 아니며 $t|_u$의 Ω-실례화가 (일반 룰에 대한) 레덱스인 경우 다음과 같은 Ω-축약을 정의한다.

$$t \rightarrow_\Omega t', \quad \text{여기서 } t' = t[u := \Omega].$$

(3) NF_s는 최소한 하나 이상의 Ω를 포함하지만, 레덱스는 포함하지 않으며, Ω-축약에 의해서 Ω로 축약될 수 있는 항들의 집합이다.

(4) NF_s에 속하는 t의 *지수(index)*는 t의 한 주소 u로서 다음 조건을 만족한다.

$$t|_u = \Omega, \quad t[u := \bullet] \not\rightarrow^*_\Omega \Omega.$$

(5) t의 지수 u는 각 $t' \in NF_s$의 모든 지수 v에 대해서 $u \cdot v$가 $t[u := t']$의 지수인 조건을 만족할 때 *추이적(transitive)*이라고 한다. NF_s의 각 원소들의 모든 지수가 추이적이면 한 OTRS는 추이적이라고 한다.

두 개의 항 t과 t'가 각각 지수 u와 v를 가지고 있다고 가정하자. 이때 $u \cdot v$가 항 $t[u:=t']$에서 지수가 되는 경우 지수의 추이성이 성립한다고 한다. 그러나 강 순차 시스템에서 일반적으로 지수 추이성은 성립하지 않는다. 예를 들어, $L = \{F(G(A, x), B), F(G(x, A), C), G(D, D)\}$을 고려하자. 이때, 1은 $F(\Omega, \Omega)$의 지수이고, 1은 $G(\Omega, \Omega)$의 지수이지만, $F(G(\Omega, \Omega), \Omega)$에서는 $1 \cdot 1$이 지수가 아니므로 지수 추이성을 만족시키지 못한다. 그러나 모든 Ω-레덱스들은 지수를 갖고 있으므로 이 시스템은 강 순차성을 만족한다. $F(G(\Omega, \Omega), \Omega)$은 주소 2에 지수를 갖고 있다. 또한, $F(G(\Omega, \Omega), B)$는 주소 $1 \cdot 1$에, $F(G(\Omega, \Omega), C)$는 주소 $1 \cdot 2$가 지수이다.

정리 3.2. *한 OTRS의 강순차성에 대한 필요충분조건은 NF_s의 모든 원소들이 지수를 갖는 것이다 [11].*

강 순차시스템에서 일반적으로 지수 추이성은 성립하지 않지만, 그 역은 성립한다. 즉, 한 항에서 주소 $u \cdot v$가 지수라면 이 항을 둘로 분리한 항 t와 t'에서 u와 v는 각각 t와 t'의 지수이다.

정리 3.3. *$u \cdot v$가 t의 지수이면 u는 $t[u := \Omega]$의 지수이*

고 v는 $t|_u$의 지수이다[11].

이 상황은 다음 그림 1로서 표현된다.

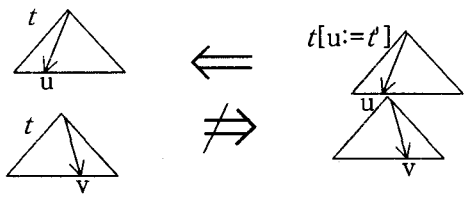


그림 1 지수 추이성의 불성립

4. 분리가능 시스템

정의 4.1. 선형 항들로 구성된 집합 L과, 그의 한 요소에 대한 *진 상부(proper prefix)* t, t의 주소 u에 위치한 변수 x를 가정하자. 모든 v 위치에 구성요소로 포함하는 L의 모든 원소 s에 대해서 $s|_{v\cdot u}$가 변수가 아니라면 u는 t의 *확장위치(extension site)*라고 부른다. 만약 L의 모든 진 상부가 확장위치를 갖는다면 L은 *분리가능*하다고 부른다. 중복된 항을 내포하지 않는 튜플들(tuples)의 분리가능성은 그들을 집합으로 간주하였을 때 분리가능인 지의 여부에 따른다. 항 개서 시스템의 분리가능성은 시스템의 왼쪽부분으로 구성된 튜플들의 분리가능 여부로 결정된다.

위에 정의한 내용을 다음의 예로서 설명한다. L = *{F(A, x), G(F(C, D))}*라고 하자. *F(x, y)*는 L의 한 구성원 *F(A, x)*의 진 상부다. 또한, *F(x, y)*는 *F(A, x)*의 <>에서, *G(F(C, D))*의 *1*에서 구성요소가 된다. 이 두 항에 대해서 진 상부 *F(x, y)*의 주소 *1*을 점검하면, $F(A, x)\,|_{<>\cdot 1}$과 $G(F(C, D))\,|_{1\cdot 1}$은 모두 변수가 아니므로 *1*은 확장위치이다. 그러나 *F(x, y)*의 주소 *2*는 $F(A, x)\,|_{<>\cdot 2}$가 변수이므로 확장위치가 될 수 없다. L의 다른 진 상부 *G(x)*, *G(F(x, y))*, *G(F(C, y))*, *G(F(x, D))* 모두 확장위치를 가지므로 L은 분리가능하다.

다른 예로서 (이 예는 [5]에서 소개되었음), 집합 *{H(G(A, A, x), A), H(G(A, x, A), B), G(B, B, B)}*을 고려하자. 이 집합의 진 상부 *G(x, y, z)*는 주소 *1*에 대해서 확장위치를 가지고 있으며, *G(B, x, y)*는 주소 *2*와 *3*에 대해서 확장위치를 가지고 있다. *G(A, x, y)*는 확장위치를 갖지 못하지만 이것은 L의 진 상부가 아니므로 고려의 대상이 아니다. 결과적으로 이 집합은 분리가능하다. 그러나 집합 *{H(G(A, x), A), H(G(A, x), B), G(B, B)}*는 분리가능하지 않다. *G(x, y)*는 이 집합 한 구성원의 진 상부이지만 주소 *1*과 *2*는 모두 확장위치가 아니다.

강 순차 시스템의 지수 추이성에 대해서 좀 더 자세히 살펴본다. 앞서 다루었던 예 *{F(G(A, x), B), F(G(x, A), C), G(D, D)}*는 강순차성을 만족하지만 지수 추이성을 만족하지 못하고 있다. *F(Ω, Ω)*에서 주소 *1*과 *2*가 지수이고, *G(Ω, Ω)*에서는 주소 *1*과 *2*가 지수이지만, *F(G(Ω, Ω), Ω)*에서 *1·1*나 *1·2* 중의 어느 것도 지수가 아니다. *F(G(Ω, Ω), Ω)*에서 주소 *2*에 있는 함수 기호가 결정되는 경우, 주소 *1·1*나 *1·2*가 지수가 될 가능성도 있다. 일반적으로, 강 순차 시스템에서 두 항 t와 t'이 NF_s에 속하며, u는 t의 지수이고, t'이 $v_1, \ldots, v_n$의 지수를 가지고 있을 때, 이 중에서 최소한 하나의 지수 v_i가 $t[u:=t']$의 지수 $u\cdot v_i$가 된다. 이때 어느 것이 선정되는 지는 t와 u의 영향을 받으면서 결정된다. 그러나 지수 추이성을 갖는 경우, t와 u에 상관없이 모든 $u\cdot v_i$는 지수가 된다.

분리가능성의 확장위치와 강 순차성의 지수사의 연관성에 대해서 논의해 보자. 분리가능성은 강 순차성과 유사하지만 추가적인 조건이 필요하다. 위에서 논의한 지수 추이성을 갖지 못하는 강 순차 시스템 *{F(G(A, x), B), F(G(x, A), C), G(D, D)}*을 다시 고려해 보자. 여기서 진 레덱스 상부 *G(x, y)*는 *G(x, y)*를 구성요소로 포함하는 *F(G(A, x), B)*와 *F(G(x, A), C)*에 대해서 *1*과 *2* 어느 것도 확장위치를 가질 수 없다. 그러나 앞서 논의한 예 *{H(G(A, A, x), A), H(G(A, x, A), B), G(B, B, B)}*는 분리가능하다. *G(Ω, Ω, Ω)*를 고려할 때, *1, 2, 3*은 모두 강 순차성의 지수이지만, 분리가능성의 확장위치를 만족하는 것은 이 중에서 오직 *1*뿐이다. 그런데, *1*은 강순차성에서 볼 때 지수 추이성의 조건을 만족하고 있다. *H(Ω, Ω)*는 지수 *1*과 *2*를 갖는데, *H(Ω, Ω)*의 주소 *2* 값에 상관없이, *H(G(Ω, Ω, Ω), Ω)*에서 *1·1*은 지수가 된다. 이와 같이 확장위치는 추이성을 만족시키는 지수이다.

강 순차성과 분리가능성 사이의 연관성과 관련하여, 모든 확장위치는 지수가 될 수 있으나, 지수는 지수 추이성을 만족할 때만 확장위치가 된다. 이와 같은 상황을 종합하여 볼 때, 우리는 다음과 같은 두개의 정리를 얻을 수 있다.

정리 4.2. *한 시스템이 분리가능하면, 그 시스템은 강 순차성을 만족한다. 그러나 그 역은 성립하지 않는다.*

증명. 분리가능성을 판단하는 데 이용되는 진 레덱스 상부의 변수들을 Ω로 대치하면 쉽게 Ω-항을 만들 수 있다. 한 시스템의 강 순차성을 판단하는 데 필요한 모든 Ω-항들은 이와 같은 방법을 적용하여 진 레덱스 상부로부터 생성될 수 있다. 한 진 상부의 확장위치는 그에 해당하는 Ω-항의 지수가 되므로, 모든 Ω-항들은 지수를 가지며, 따라서 그 시스템은 강 순차성을 만족한다.

위의 예에서 논의한 대로 레덱스의 왼쪽 부분 *{F(G(A, x), B), F(G(x, A), C), G(D, D)}*은 강 순차성을 만족하지만, 진 레덱스 상부 *G(x, y)*는 확장위치를 갖지 못하므로 분리가능성을 만족시키지 못한다. □

정리 4.3. *한 강 순차시스템이 지수 추이성을 만족하면, 그 시스템은 분리가능하다. 그러나 그 역은 성립하지 않는다.*

증명. 앞서 설명한 대로, 추이성을 만족하는 지수는 확장위치가 된다. 그러나 분리가능성을 만족하는 시스템이 반드시 지수 추이성을 만족하지는 않는다. 예를 들어, 룰의 왼쪽 항들이 *{F(A, B), G(F(C, x))}*인 시스템을 고려해 보자. 이 시스템의 진 레덱스 상부는 *F(x, y), F(A, x), F(x, B), G(x), G(F(x, y))*이다. 이 모든 진 레덱스 상부에서 *x*로 표시된 부분은 확장위치이므로, 이 시스템은 분리가능성을 만족한다. 그러나 이 룰에 대해서, *G(Ω), F(Ω, Ω), G(F(Ω, Ω))*를 고려할 때, 처음 둘은 각각 *1*과 *2*에서 지수를 갖지만, *1·2*는 세 번째의 지수가 아니므로 지수 추이성을 만족하지 못한다. □

일반적인 TRS 대신 구성자 시스템인 경우 강 순차성과 분리가능성은 상호 일치한다.

정리 4.4. *구성자 시스템 R을 가정하자. R이 강 순차성을 만족하는 경우 R은 지수 추이성을 갖는다[11].*

정리 4.5. *구성자 시스템 R을 가정하자. R이 분리가능인 것은 R이 지수 추이적 강순차성인 것의 필요충분조건이다.*

증명. *R*이 분리가능 시스템인 경우, 정리 4.2에 따라 *R*은 강 순차성을 만족한다. *R*이 구성자 시스템이므로, 정리 4.4에 따라 *R*은 지수 추이성을 만족한다. 역으로, *R*이 지수 추이적 강 순차성을 갖는 경우, 정리 4.3에 따라 *R*은 분리가능하다. □

5. 전진 분지성

전진 분지성은 OTRS의 지수 트리로서 설명된다[4,5](아래 그림 2 참조). OTRS의 지수 트리는 *유한 상태 머신(finite state automaton)*이다. *넌터미널(non-terminal)* 상태는 진 레덱스 상부 *t*와 *t*의 지수 *u*의 쌍 *(t, u)*으로 구성되어 있으며, 그의 터미널 상태는 시스템 룰의 왼쪽 항이다. 이것의 시작 상태(initial state)는 *(Ω, <>)*이며, *전이함수(transition function)*는 전이가 실패하는 경우와 성공하는 경우 두 가지가 발생할 수 있다. 성공하는 경우 함수기호 A에 대해서 쌍 *((t, u), A)*를 지수 트리의 상태로 사상한다(map). *((t, u), A)*가 상태 s로 사상되는 모습은 상태 *(t, u)*로부터 s로의 라벨(label) *A*에 따라 전이되는 그림으로 표현된다. 여기서 *s*는 $t' = t[u:=A(x_1, ..., x_n)]$인 쌍 (t', u') 혹은 터미널인 경우 t'의 형태를 갖는다.

지수 트리는 TRS를 위한 축약 알고리즘을 표현한다. 트리의 어떤 한 상태 *(t, u)*는 상부가 *t*이고 다음 점검할 주소가 *u*인 한 식의 계산과정을 의미한다. 전이함수는 이 한 레덱스 상부의 한 함수기호를 점검한 후, 이 레덱스 상부의 기호들을 모두 유지하면서 다음에 새로 점검해야 할 함수기호를 점검하는 상태로 이동하는 의미를 갖고 있다. 실패전이는 원하는 기호가 아닌 다른 기호가 발견되었을 경우 어떻게 처리해야 하는 지를 기술하고 있다. 이 경우에는 *t* 보다는 *t*의 서브텀이 먼저 계산되어야 하는데, 실패전이는 그 서브텀에 해당되는 상태로 이동하는 상황을 표현한다. 그림 2에서, 점선에 의한 전이는 실패전이를 의미한다. *H(G(Ω, Ω, Ω),A)*에서 주소 *1·1*에 함수기호 *A*를 찾지 못하면 성공적인 전이를 할 수 없으며, 이 경우 *H(G(Ω, Ω, Ω),A)*의 서브텀 *G(Ω, Ω, Ω)*에서 전이를 시도한다. 즉, 계산하려는 항의 주소 *1·1*에 함수기호가 *A*가 아니라면 이 항은 *H(G(A ...))* 형태의 레덱스를 갖지 못하므로 그 서브텀에 *G(B, B, B)* 형태의 레덱스로 존재하는 지를 검토해 보아야 한다. 이때, 서브텀 *G(Ω, Ω, Ω)*의 레덱스 여부는 *H*를 포함한 다른 문맥에 영향을 받지 않고 독립적으로 결정된다.

어떤 한 지수 트리에서 지수 트리의 모든 상태가 시작 상태로부터 실패하는 경우 없이 전이되어 도달할 수 있는 경우 *전진 분지*라고 정의한다. 어떤 한 시스템이 터미널 상태가 룰의 왼쪽 항들로 구성되는 전진분지 지수 트리로서 표현될 수 있으면 그 시스템은 전진 분지성을 갖는다고 정의한다.

예를 들어, 다음과 같은 룰을 갖는 시스템을 고려하자 - *{H(G(A, A, x), A) → A, H(G(A, x, A) → B, G(B, B, B) → C}*[5]. 이 룰은 그림 2의 전진분지 지수트리를 가질 수 있다.

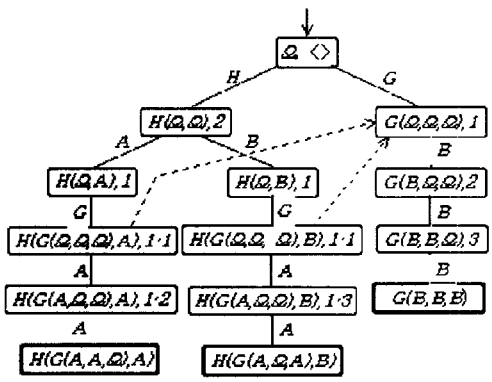


그림 2 전진 분지성에 의한 지수 트리

정리 5.1. *한 OTRS의 분리가능성은 전진 분지성의 필요충분조건이다.*

증명. 첫째, 분리가능하면 전진 분지임을 보인다. 한 시스템 R이 분리가능이면, R에 대한 전진 분지 지수 트리를 구성할 수 있음을 보여야 한다. 분리가능성의 원리에 따라 전진분지 지수트리를 구성해 나간다. 지수 트리의 구성은 상태 (레덱스 상부, 지수)와 각 상태에서 다음 상태로 전이되는 상황을 기술해야 한다. 한 넌터미널 상태의 (t, u)를 고려하자. 만약 $<> < v < u$이고 $t|_v$가 레덱스 상부라면, $t|_v$는 이미 지수 트리에 첨가된 상태이고, $t|_v$의 지수가 w 일 때, $u = v \cdot w$인 관계가 성립한다. 초기상태 $(x, <>)$는 당연히 성립한다. 각 단계별로 전 단계에 비해서 최소한의 깊이(depth)를 추가적으로 갖는 진 레덱스 상부 t를 선택한다. 분리가능의 조건에 따라 t는 최소한 하나의 확장위치를 갖는다. u를 이러한 확장위치라고 하자. 만약 $<> < v < u$이고 $t|_v$가 레덱스 상부인 v가 존재하지 않는다면, 이에 대한 상태는 (t, u)가 된다. 만약 그러한 v가 존재한다면, 위에서 언급한 대로 가장 위쪽에 위치한 v를 선택한다. w가 $t|_v$에 해당하는 지수일 때, t에 대한 지수는 $v \cdot w$가 된다.

상태 전이함수 $T(t, A)$는 레덱스 상부 t인 상태에서 함수기호 A와 연관된 다음 상태를 결정하는 기능을 한다. 각 함수기호 A에 대해서, 상태 (t, u)에 대해서 $t' = t[u := A(x_1, ..., x_n)]$가 레덱스 상부이면, $T(t, A) = t'$이다. 이와 같은 방법으로 룰의 왼쪽부분인 터미널 상태에 도달할 때까지 지수 트리를 구성해 나가면, 이 지수 트리는 전진 분지성을 갖게 된다.

둘째, 전진 분지이면 분리가능임을 보인다. 한 시스템의 각 진 레덱스 상부가 확장위치를 가짐을 보여야 한다. 한 진 레덱스 상부 t에 대해서, t의 모습을 (루트에서 시작하여) 가장 많이 포함하면서 지수 트리의 한 상태에 있는 항 t'과 그의 지수 u를 가정한다. $t|_u$ 항의 루트가 A인 경우, $t'[u := A(x_1, ..., x_n)]$은 t보다 큰 상부로서 지수 트리의 한 상태로 나타날 수 있다. 그러므로 $t|_u = \Omega$이다. u는 t'의 확장위치이므로 이것은 또한 t의 확장위치이다. □

정리 5.1과 정리 4.5에 따라 다음과 같은 보조정리를 얻을 수 있다.

보조 정리 5.2. *구성자 시스템 R을 가정하자. R이 강 순차성인 것은 R이 전진 분지성인 것의 필요충분조건이며, 또한 R이 분리가능성인 것의 필요충분조건이다.*

6. 분리가능 시스템의 변환

이 절에서는 분리가능 시스템은 매우 간단한 형태의 TRS인 단일 구성자 시스템으로 변환하는 기법을 소개한다. 이 과정은 *평탄화(flattening)*라고 부른다. 단일 구성자 시스템은 룰의 왼쪽 부분이 매우 제약된 형태로 구성되어 있다. 연산자는 루트 외에는 나타나지 않으며, 구성자는 나타나지 않거나 혹은 오직 하나만 나타날 수 있다.

일반적으로 분리가능 시스템에서는 한 레덱스 상부가 다수의 확장지수를 가질 수 있으므로 축약의 과정이 비결정적(nondeterministic)으로 진행될 수 있다. 분리가능 시스템에 평탄화를 적용함으로써 변환된 단일 구성자 시스템에서는 축약 과정이 특정 순서로 진행되도록 강제하게 되어, 결과적으로 비결정성을 제거하는 효과를 갖는다. 아래에서 표현 $\overline{w}$은 변수들로 구성된 벡터 $w_1, w_2, ..., w_n$을 의미한다.

알고리즘 6.1. (평탄화) OTRS와 연산자 F, 정수 i, 구성자 C를 다음과 같이 선택한다.

(1) i는 $F(\overline{w})$의 확장지수이다.

(2) $F(\overline{x}, C(\overline{y}), \overline{z})$는 레덱스 구성요소이다 (그러나 레덱스 그 자체는 아니다) (서브텀 $C(\overline{y})$는 F의 i번째 인수이다).

(3) 새로운 함수 기호 F_C와 다음과 같은 새로운 룰을 추가한다.

$$F(\overline{x}, C(\overline{y}), \overline{z}) \rightarrow F_C(\overline{x}, \overline{y}, \overline{z})$$

이 룰을 F_C에 대한 소개(introduction) 룰이라고 부른다.

(4) 룰의 왼쪽부분에서 구성요소 $F(\overline{x}, C(\overline{y}), \overline{z})$에 대한 실례화를 $F_C(\overline{x}, \overline{y}, \overline{z})$의 실례화로 대치한다.

예를 들어, 다음과 같은 분리가능 시스템을 고려하자.

$H(G(A, A, x), A) \rightarrow t_0$

$H(G(A, x, A), B) \rightarrow t_1$

$G(B, B, B) \rightarrow t_2$

위의 룰에서 진 레덱스 상부 $G(x, y, z)$는 확장지수 *1*을 가지고 있으며, $G(A, x, y)$는 레덱스가 아닌 레덱스 구성요소이고, A는 구성자이다. 따라서 새로운 구성자 G_A를 사용하는 새 룰 $G(A, x, y) \rightarrow G_A(x, y)$이 추가되고, 레덱스 구성요소 중에서 $G(A, x, y)$의 실례화를 $G_A(x, y)$의 실례화로 대치한다. 따라서 $G(A, A, x)$와 $G(A, x, A)$ 각 $G_A(A, x)$와 $G_A(x, A)$로 대치되어, 결과적으로 다음과 같은 룰로 변환된다.

$H(G_A(A, x), A) \rightarrow t_0$

$H(G_A(x, A), B) \rightarrow t_1$

$G(B, B, B) \rightarrow t_2$

$G(A, x, y) \rightarrow G_A(x, y)$

다음 진 레덱스 상부 $G(x, y, z)$는 확장지수 *1*을 갖고 있으며, $G(B, x, y)$는 레덱스 구성요소이다. 비슷한 방법으로 $G(B, x, y) \rightarrow G_B(x, y)$와 $G_B(B, B) \rightarrow t_2$의 룰이 생성된다.

$H(G_A(A, x), A) \rightarrow t_0$

$H(G_A(x, A), B) \rightarrow t_1$

$G_B(B, B) \rightarrow t_2$
$G(A, x, y) \rightarrow G_A(x, y)$
$G(B, x, y) \rightarrow G_B(x, y)$

이 과정을 반복하면 다음의 두 과정을 거쳐 최종적으로 단일 구성자 시스템을 얻을 수 있다.

$H(G_A(x, y), A) \rightarrow H_{GA}(x, y)$
$H(G_A(x, y), B) \rightarrow H_{GB}(x, y)$
$H_{GA}(A, x) \rightarrow t_0$
$H_{GA}(x, A) \rightarrow t_1$
$G_B(B, B) \rightarrow t_2$
$G(A, x, y) \rightarrow G_A(x, y)$
$G(B, x, y) \rightarrow G_B(x, y)$

$H(G_A(x, y), z) \rightarrow H_{AGA}(x, y, z)$
$H_{AGA}(x, y, A) \rightarrow H_{GA}(x, y)$
$H_{AGA}(x, y, B) \rightarrow H_{GB}(x, y)$
$H_{GA}(A, x) \rightarrow t_0$
$H_{GB}(x, A) \rightarrow t_1$
$G_B(B, x) \rightarrow G_{BB}(x)$
$G_{BB}(B) \rightarrow t_2$
$G(A, x, y) \rightarrow G_A(x, y)$
$G(B, x, y) \rightarrow G_B(x, y)$

원래의 분리가능 시스템에서 한 단계의 축약 $H(G(A, A, A), A) \rightarrow t_0$은 변환된 단일 구성자 시스템에서 다음과 같이 여러 단계를 거쳐 축약된다. $H(G(A, A, A), A) \rightarrow H(G_A(A, A), A) \rightarrow H_{AGA}(A, A, A) \rightarrow H_{GA}(A, A) \rightarrow t_0$.

정리 6.2. *평탄화는 종료된다.*

증명. 평탄화를 적용하는 경우, 각 룰 왼쪽에 나타나는 함수 기호들의 수를 측정하여 이 수가 감소함을 보임으로써 평탄화의 종료를 보장할 수 있다. 평탄화를 한 번 적용하는 경우, 왼쪽에 두 개의 함수 기호를 갖는 새로운 룰이 추가되고, 기존에 있던 룰의 왼쪽에서는 최소 한 개 이상의 함수기호가 감소된다. 이때, 평탄화가 적용된 룰의 왼쪽은 3개 이상의 함수 기호를 갖는다. 한 번의 평탄화는 3보다 큰 n 개의 함수기호를 $n-1$만큼씩 감소하는 효과를 가지며, 평탄화의 반복 적용에 따라 각 룰의 함수 기호는 감소하여 결국 종료되게 된다. □

정리 6.3. *평탄화에 의해 변환된 단일 구성자 시스템은 OTRS이다.*

증명. 평탄화의 한번 적용한 결과가 직교적이라는 것을 보이면 충분한 증명이 될 수 있다. 함수 F와 C가 관련된 기호들이고, i가 레덱스 상부 $F(\overline{x})$의 확장위치라고 하자. 변환된 시스템은 좌선형임을 쉽게 알 수 있다. 변환된 룰의 모호성 여부를 점검해 보자. $F(\overline{x})$가 두 룰의 공통적인 레덱스 상부일 경우, 평탄화 변환은 하나의 룰 $F(\overline{x}, \overline{y}, \overline{z}) \rightarrow F_C(\overline{x}, \overline{y}, \overline{z})$로 대치되고, $F_C(\overline{x}, A_1(\overline{y}), \overline{z})$와 $F_C(\overline{x}, A_2(\overline{y}), \overline{z})$의 실례화된 항이 새로운 두 룰의 왼쪽이 된다. 이때, A_1과 A_2는 다른 기호이며, 따라서 변환된 룰은 모호하지 않다. 만약 A_1과 A_2가 같은 기호라면 변환되기 전의 원래의 룰 또한 직교성을 갖지 못한다. 결과적으로 한 단계 변환된 룰을 직교적이며, 이것을 반복 적용한 것 또한 마찬가지이다. □

정리 6.4. *평탄화는 분리성을 유지하고(preserves) 반영(reflects)한다.*

증명. 평탄화의 한 단계를 적용한 결과에 대해서 이것이 성립함을 보임으로써 충분한 증명이 될 수 있다. F, i 및 C에 평탄화 한 단계를 적용하여 시스템 R을 R'으로 변환하였다고 가정하다.

첫째, R이 분리가능이라고 가정하자. l'은 R'에 속한 한 룰의 왼쪽 항이고 t'은 l'의 진 레덱스 상부라고 하자. l'이 F_C의 소개 룰인 경우, t'은 $F(\overline{x})$이고 i는 R'에서 t'의 확장위치이다. 이것이 성립하지 않는 경우라면 l'은 $F(\overline{t_1}, C(\overline{t_2}), \overline{t_3})$를 $F_C(\overline{t_1}, \overline{t_2}, \overline{t_3})$으로 대치함으로써 생성된 것으로서, l'의 연산자는 F_C이다. 이 새로운 F_C 룰에 대해서 R의 원래의 룰 l과 진 레덱스 상부 t를 고려한다. 그러면 t는 i 외에 다른 확장지수 u를 포함하고 있으며, 이 u는 t'에서 역시 확장지수가 된다. 따라서 R' 또한 분리가능이다.

둘째, R'이 분리가능이라고 가정하자. t를 R의 한 왼쪽 항 l의 진 상부라고 하자. t의 모든 $F(\overline{t_1}, C(\overline{t_2}), \overline{t_3})$를 $F_C(\overline{t_1}, \overline{t_2}, \overline{t_3})$로 치환하고, 이때 얻어지는 항을 t'이라고 하자. 만약 t'은 R'의 한 룰에 대한 진 레덱스 상부이면, 위에서 논의한 대로, t의 모든 확장지수는 t'의 확장지수가 된다. 그러나 t'이 그러한 상부가 아니라면 t는 어떤 주소 u에 $F(\overline{x})$ 형태를 서브텀으로 내포하고 있다. 그러나 이 경우, i가 $F(\overline{x})$ 의 확장지수이므로, $u \cdot i$ 또한 t의 확장지수가 된다. 그러므로 R 또한 분리가능이다. □

정리 6.5. *모든 직교성을 갖는 단일 구성자 시스템은 분리가능이다.*

증명. 단일 구성자 시스템에서 모든 진 레덱스 상부는 $F(\overline{w})$의 형태를 갖는다. 직교성에 따라 이 레덱스 상부와 연관된 룰의 왼쪽은 유일한 $F(\overline{x}, C(\overline{y}), \overline{z})$ 형태의 구성자를 갖는다. 이때, C가 주소가 곧 $F(\overline{w})$의 확장위치가 된다. □

정리 6.6. *R이 OTRS라고 하자. R이 분리가능인 것은 R의 평탄화 결과가 단일구성자 시스템인 것의 필요*

충분조건이다.

증명. 첫째, 변환된 시스템이 단일 구성자임을 가정하자. 정리 6.3에 의해서 이 시스템은 직교적이며, 정리 6.5에 의해서 분리가능이다.

둘째, 분리가능 시스템 R을 가정하자. 정리 6.2에 따라 평탄화는 종료되며 모든 룰의 왼쪽은 최대한 2개의 함수기호를 갖는 시스템으로 변환된다. □

이 절에서 연구된 평탄화 변환 방법은 Thatte 등에 의해서 소개된 단순 변환(*simple transformation*) [12] 기법보다 훨씬 향상된 기법이다. 단순 변환은 룰의 왼쪽에 연산자를 루트로 갖는 진 서브텀에 대한 새로운 룰을 첨가하는 방법을 이용한다. 예를 들어, 위의 룰은 단순 변환시키면 다음과 같이 변환된다 (여기서 C_1과 C_2는 새로운 구성자이다).

$H(C_1(x), A) \rightarrow t_0$

$H(C_2(x), B) \rightarrow t_1$

$G(B, B, B) \rightarrow t_2$

$G(A, A, x) \rightarrow C_1(x)$

$G(A, x, A) \rightarrow C_2(x)$

이 방법에 의해 변환된 시스템은 OTRS가 되지 못하는 문제점을 갖고 있다.

Durand 등 또한 전진 분지 시스템에 대한 변환 기법을 제시하고 있다[4,5]. 전진분지 지수트리를 이용하는 변환의 기본 원리를 본 연구에서 제시하는 변환 원리와 유사하지만, 본 연구에서는 분리성의 특성에 따라 새로운 변환기법을 제시하였다.

7. 관련 시스템 및 결론

지금까지 OTRS 및 그의 서브클래스인 강 순차 시스템에 대한 많은 관심과 연구가 진행되었다. 이들의 관계는 다음 그림 3으로서 설명될 수 있다.

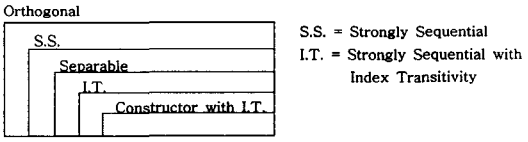


그림 3 OTRS의 서브클래스들

강 순차 시스템은 OTRS의 한 서브클래스이고, 지수 추이성을 갖는 강 순차 시스템은 분리가능하다. 그러나 분리가능 시스템 중의 일부는 지수 추이성을 갖지 않는다. 분리가능성은 전진 분지성과 일치한다. 분리가능 시스템은 모두 람다 계산법으로 번역될 수 있다[13]. 구성자 시스템에서는 분리성, 전진 분지성, 지수추이적 강 순차성은 모두 동일하다. Haskell 및 Clean[14] 등의 함수형 언어는 구성자 시스템의 문법적 구조를 가지고 있다. 이 언어들은 완전한 선언적(declarative) 특성을 갖지 못하고 "위에서 아래로, 왼쪽에서 오른쪽" 순서로 패턴 매칭을 수행하므로 TRS와 직접적으로 연관시키기는 어렵다. 그러나 [15]의 방법을 적용할 때 이들 함수형 언어들은 강 순차성을 갖는 구성자 시스템(즉, 분리가능 구성자 시스템)에 해당된다. 모든 분리가능 구성자 시스템들이 람다 계산법으로 번역 가능함을 보여주는 결과 [13]에 따라, *지연(lazy)* 함수형 언어들과 람다 계산법사이의 연관성을 정형적으로 설명할 수 있다.

분리 시스템은 강 순차 시스템보다는 약간 제한된 구문적 표현력을 갖지만, 대신 람다 계산법으로 변환될 수 있고, 단일 구성자 시스템으로 변환될 수 있으며, 지수 추이성을 만족시키는 등의 중요한 장점을 가지고 있다. 이러한 우수한 특징에 따라 향후 강 순차 시스템에 대한 관심이 높아질 것으로 예상한다.

참 고 문 헌

[1] S. Byun. Some properties of separable systems. *J. of Korean Information Science Society (A)*, vol. 25, No. 4, April 1989.

[2] S. Byun, J.R. Kennaway, and R. Sleep. Transformation of orthogonal term rewriting systems, *Lecture Notes in Computer Science* 1023, pp. 73-87, Springer-Verlag, 1995.

[3] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, North-Holland, 1984.

[4] I. Durand. Bounded, strongly sequential, and forward branching term rewriting systems. *J. of Symbolic Computation*, 18:319-352, Springer-Verlag, 1994.

[5] B. Saliner and R. Strandh. Simulating forward-branching systems with constructor systems. *TAPSOFT '97*, M. Bidoit and M. Dauchet (eds.), *Lecture Notes in Computer Science* No. 1214, 153-164, 1997.

[6] S. Thatte. On the correspondence between two classes of reduction systems. *Information Processing Letters*, 20:83-85, 1985.

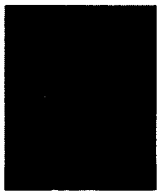
[7] S. Peyton-Jones. The Implementation of Functional Programming Languages. *Prentice-Hall*, 1987.

[8] J.W. Klop. Term rewriting systems, In Abramsky et al., editors, *Handbook of Logic in Computer Science*, volume II. Oxford University Press, 1992.

[9] N. Dershowitz and J.-P. Jounnaud. Rewrite Systems, in van Leeuwen (eds.), *Handbook of Theoretical Computer Science, Vol. B*, Elsevier, 1990.

[10] G. Huet and J.-J. Lévy. Computations in Orthogonal Rewrite Systems {I} and {II}, In Lassez and Plotkin, eds., *Computational Logic : Essay in Honor of Alan Robinson*, MIT Press 1991. (Originally appeared as Technical Report 359,

INRIA, 1979.)
[11] J.W. Klop and A. Middeldorp. Sequentiality in orthogonal term rewriting systems, *Journal of Symbolic Computation*, 12:161-195, 1991.
[12] Satish Thatte. On the correspondence between two classes of reduction systems. *Information Processing Letters*, 20:83-85. 1985.
[13] S. Byun, J.R. Kennaway, and R. Sleep. Lambda-definable Term Rewriting Systems, *Lecture Notes in Computer Science 1179*, pp 106-115, Springer-Verlag, 1996.
[14] R. Plasmeijer and M. Eekelen. *Functional programming and parallel Graph rewriting*, Addison-Wesley, 1993.
[15] J.R. Kennaway. The Specificity Rule for Lazy Pattern-Matching in Ambiguous Term Rewriting Systems, *ESOP '90, Lecture Notes in Computer Science* No. 432, Springer-Verlag, 1990.



변 석 우

1976년~1980년 숭실대학교 전자계산(학사). 1980년~1982년 숭실대학교 전자계산(석사). 1982년~1999년 ETRI 책임연구원. 1988년~1994년 영국 University of East Anglia 전산학(박사). 1998년~현재 경성대학교 컴퓨터과학과 부교수 관심분야는 rewriting system, 함수형 프로그래밍, 의미론, 정형 시스템 등