조합논리 소개

정 계 섭 (덕성여대)

【요약문】 조합논리는 기본적으로 정해진 해석이 없는 순수한 형태만을 가지고 추상적으로 연산하는 관점에 관한 논리로서, 논리학을 기호학적 관점에서 볼 수 있는 토대를 제공해 준다. 조합논리의 특징은 연산자가 피연산자도 될 수 있다는 사실에 있으며 그래서 동일한 연산자가 그 자신의 피연산자도 될 수 있다. 이 논문에서 우리는 기본연산자들의 직관적 개념과 형식적 개념을 소개하고 연산자 대수에 대해 검토하고 나서, 조합논리와 λ-연산의 번역가능성에 대해 알아보겠다. 조합논리에 유형의 개념을 추가하면 자연언어 분석에서 아주 효율적인데, 기본유형인 대상자 명제 이외의 어떤 요소라도 함수자로 나타낼 수 있는데 이들은 조합자의 특수한 경우로서 파생유형들이다.

【주제어】 (피)연산자, 적용 (application), curryfication, 환원, 유형(type)

Ⅰ. 들어가면서

조합논리는 독일의 Schönfinkel (1920)과 미국의 논리학자 Curry (1930)에 의해 개발되었다. 그것은 변수[1]에 관련된 어려움과 그리고 모순을 가져오는 표현들에 대한 문제의식으로부터 비롯되었다. 여기에서는 인문과학 분야에서 조합논리를 활용하고자 하는 연구자들을 위해 형식언어의 관점에서 소개하고자 한다.

조합논리는 연산자 x가 피연산자 y에 작용하여 xy를 결과로 갖는 연산을 다루는 응용체계이다.

$$\frac{X \qquad\qquad Y}{X\,Y}$$

1) (p ⊃ q) = df(~ p ∨ q)는 Cpq = dfANpq로 나타낼 수 있으므로 결국 변수없이 C = dfAN으로 표현될 수 있다.

조합자들 (combinators)은 특정 분야의 해석과는 독립적이라는 의미에서 추상적 연산자들인데, 이 연산자들은 서로간에 결합하여 새로운 연산자를 만들 수 있으며, 또한 기본적인 연산자들을 가지고 보다 복잡한 연산자를 만들 수도 있다.

Ⅱ. 함수언어

형식언어로서 함수언어는 다음과 같이 부여한다.

① 알파벳 : 임의의 대상들의 집합 E
a, b, c, d, a1, b1, …
② 2항연산자 : *
③ 괄호 : (,)

그리고 다음과 같이 형성규칙 (formation rule)을 부여한다.

FR1 : 집합 E의 모든 원소는 대상이다.
FR2 : x와 y가 대상이면, (x*y)는 또 하나의 대상이다.
FR3 : 이 밖의 다른 것은 대상이 아니다.

하나의 대상 a는 b에 적용될 수 있고, 그 결과는 (a * b)로 쓴다. 이 새로운 대상은 다시 c에 적용하면 ((a * b) * c)가 되고, 만일 c를 c (a * b)에 적용하면 (c * (a * b)가 된다.

표현을 간결하게 하기 위해 우리는 "*"를 생략할 수 있다. 그래서 (a * b)는 (ab)가 되고 ((a * b) * c)는 ((ab) c)가 된다. 나아가서 하나의 대상 왼쪽에 있는 괄호는 생략할 수 있다.

(ab) = ab
((ab) c) = (ab)c

여기에서 왼쪽 결합규칙 (Left Association)을 도입하도록 하자.

LA : 여는 괄호가 하나의 표현의 왼쪽에 있을 때는 이 쌍의 괄호를 생략할 수 있으며, 여는 괄호가 생략할 수 없는 괄호에 의해 한정된 하위표현이 왼쪽에 있을 때 역시 이 쌍의 괄호도 생략할 수 있다.

첫 번째 경우가 (ab) c → abc 에 해당하고, 두 번째 경우는

(a ((bc) d) → a (bcd) 에 해당한다. (((ab) c) d) → abcd로 환원된다. 그러나 a (bc), a (b (cd) 에는 LA가 적용되지 않는다.2)

Schönfinkel의 전통적인 함수개념에 두 가지 주요한 수정을 가하였다.

첫째, 그는 하나의 함수가 그 값으로서 다른 함수를 취할 수 있도록 하여 함수의 개념을 확장하였다.

둘째, 여러 개의 논항을 가진 함수를 하나의 논항을 가진 함수로 환원하였다. 이에 대해 알아보도록 하자.

이러한 절차를 보통 Curryfication (Currying 또는 Curryage)라고 부른다.

f′(x1, … xn) 이 있다고 하자. 핵심은 f′를 단항함수 f로 취급하는 데에 있는데 x1에 적용되어 그 값으로서 단항함수 fx1을 갖고 fx1은 다시 x2에 적용되어 그 값으로서 다시 fx1x2라는 단항함수를 갖고 이런 식으로 계속 진행하여 단항함수 fx1 … xn-1을 가질 때까지 나아가서 끝으로 xn에 적용되어 값으로서 f′(x1, … , xn) 을 갖게 된다. 커리화된 표현 fx1x2 · · · xn는 그래서 ((… ((f x1) x2 … xn) 을 LA에 의해

2) 여기에서 기호열과 집합은 분명히 구분할 필요가 있다. abc는 3개의 대상 a, b, c가 나열된 기호열이고, a(bc)는 a라는 대상과 (bc)라는 대상의 기호열이다. { a, b, c }는 주어진 대상들의 집합이다.

괄호를 생략한 표현이다.

Ⅲ. 조합연산자의 직관적 개념

앞에서 만든 함수언어의 알파벳에 기본 조합연산자 I, K, W, C, B를 추가하면,

두 개의 이항관계를 도입하자 : → (환원가능성),
= (등식관계)

그리고 새로운 형성규칙을 추가한다.

FR4 : x와 y가 대상이면, x → y, x = y 는 적형식이다.

aIb 또는 (a * (I * b) 는 하나의 대상이므로 (a * (I * b)) → a 와 (a * (I * b)) = a 는 FR4에 의해 적형식이다. 하나의 표현이 적형식이라는 것은 그 표현의 진리치와는 별개의 문제이다.

위에서 도입한 다섯 개의 기본 조합연산자에 다시쓰기규칙 (또는 환원규칙)을 부여하는데, 이 규칙들은 어떤 기호열에 변형을 가져온다.

I	Ix	→	x	확인연산자
K	Kxy	→	x	제거연산자
W	Wxy	→	xyy	중복연산자
C	Cxyz	→	xzy	치환연산자
B	Bxyz	→	x (yz)	합성연산자

이 다섯 개의 원자연산자들이 결합하여 분자연산자들을 만들게 된다.

Ⅳ. 환원기술

이제부터 가장 중요한 변형기술 몇 가지를 소개한다.

1. x → y → z → r · · · 은 x → y, y → z, z → r, · · · 을 의미한다.

   사례 : CWab → Wbc → baa

2. 생략될 수 없는 괄호로 묶인 분자 대상 (용어)는 FR2에 의해 하나의 대상으로 간주된다.

   사례 : Ka (Wab) → a

3. 어떤 변형에서 생략될 수 없었던 괄호는 그 효력을 상실하고 LA에 의해 제거되어야 한다.

   사례 : B (WK) ab → WK (ab)
   ↛ (WK) (ab)

   WK (ab) → K (ab) (ab) → ab
   ↛ (ab)

4. 충분한 대상 (논항)의 부족으로 Ka, K(ab), C(ab)처럼 더 이상 환원이 불가능한 경우가 있다.

5. 동일한 기호가 위치에 따라 연산자도 되고 피연산자 (opérande)도 될 수 있다.

   사례 : W1 W2 a → W2 aa
   연산자 피연산자 연산자

6. a(Wbc)나 Wabc 등은 이제까지의 규칙을 가지고는 환원이 불가능하다. 이런 경우를 위해 두 가지 단조(monotony)규칙을 도입한다.

u (우단조) : x → y 이면 zx → zy이다.

v (좌단조) : x → y 이면 xz → yz이다.

그래서 x → y가 Wbc → bcc일 때 (u)에 의해 a(Wbc) → a(bcc)가 되고, x → y가 Wab → abb일 때 (v)에 의해 Wabc → abbc가 된다.

V. 조합연산자의 형식적 개념

다음과 같은 X를 조합연산자라고 한다.

$$Xx1x2 \cdot\cdot\cdot xn \underset{df}{\rightarrow} y1y2 \cdot\cdot\cdot yn$$

여기에서 X는 원자조합연산자이고, x1x2 · · · xn은 n개 대상의 기호열이며 y1y2 · · · yn은 x1x2 · · · xn의 어떤 일정한 방식에 의한 조합이다.

조합연산자의 특성은 귀납적으로 정의된다.

① I, K, W, C, B는 조합언어의 연산자들이다.
② x와 y가 조합연산자이면 (x * y)도 조합연산자이다.
③ 이 밖에 다른 것은 조합연산자가 아니다.

②에 의해 K(WB), CIK, WWW들이 분자연산자들임을 쉽게 알 수 있

다. 그리고 원자 조합연산자에 준 위의 정의에 의해 우리는 다른 원자 조합연산자들을 도입할 수 있다. 기본적 연산자가 아닌 고전적 연산자들로서 다음을 들 수 있다.

Sxyz	→	xz(yz)
Φxyzw	→	x(yw)(zw)
Ψxyzw	→	x(yz)(yw)

이렇게 원자 조합연산자의 정의를 만족시키는 연산자들을 고유연산자라고 하는데, 분자 조합연산자들 중 WW는 고유연산자이지만 WWW는 고유연산자가 아니다.

X의 첫 번째 논항이 불변일 경우, 즉

Xx1x2 · · · xn → x1y1 · · · ym이 되는 경우 이런 X를 정규 조합연산자라 한다.

그리고 조합연산자를 가지지 않는 a나 a(bc) 등을 정상형태 (Normal Form)라고 한다.

Ⅵ. 조합연산자의 대수

이제부터 2항연산자 "·" (적)을 도입하자.

(X · Y) x1x2 · · · xn 라는 표현의 의미는 X를 먼저 x1x2 · · · xn에 적용시키고 Y를 Xx1x2 · · · xn에 적용시키라는 것이다.

X가 정규연산자인 경우 정의에 의해 (X · Y) = BXY이다. 왜냐하면,

Xx1x2 ⋯ xn → x1y1 ⋯ ym

Yx1y1 ⋯ ym → z1z2 ⋯ zp 이고,

BXYx1x2 ⋯ xn → X(Yx1)x2 ⋯ xn

→ Yx1y1 … ym
→ z1z2 … zp 가 되기 때문이다.

"·" 와 "*"를 혼동해서는 안된다.
(C · K)abc는 먼저 Cabc → acb에서 Kacb → ab인데 반하여,
CKabc → Kbac → bc이기 때문이다.

하나의 연산자의 멱을 고려해야 할 때가 종종있다.
정의상 x1 = x 이므로 Xn+1 = (xn · x) 가 된다.
그리고 X가 정규적이면,

x2 = Bxx
x3 = B(Bxx)x 가 된다.[3)]

X가 기본연산자인 경우 두 가지 경우가 있는데 첫째는 Xn이 X의 논항보다 더 많은 수의 논항을 필요로 하지 않는 경우이다.

①		Cxyz → xzy	②		Wxy → xyy
	C^2	Cxzy → xyz		W^2	Wxyy → xyyy
	C^3	Cxyz → xzy		W^3	Wxyyy → xyyyy

두 번째 경우는 xn이 x의 논항보다 많은 수의 논항이 필요한 경우인데 논항의 수를 축소하는 K와 B가 이 경우에 속한다.

①		$Kx_1x_2x_3 \rightarrow x_1x_3$	②		$Kx_1x_2x_3x_4 \rightarrow x_1x_3x_4$
	K^2	$Kx_1x_3 \rightarrow x_1$		K^2	$Kx_1x_3x_4 \rightarrow x_1x_4$
				K^3	$Kx_1x_4 \rightarrow x_1$

3) (C · K)abc = BCKabc 가 되므로.

B의 경우를 보자.

① Bx1x2x3x4 → x1(x2x3)x4
B2 Bx1(x2x3)x4 → x1(x2x3x4)

② Bx1x2x3x4x5 → x1(x2x3)x4x5
B2 Bx1(x2x3)x4x5 → x1(x2x3x4)x5
B3 Bx1(x2x3x4)x5 → x1(x2x3x4x5)

이런 식으로 각 연산자의 몍을 점검하면 일정한 패턴이 나온다는 사실로부터 결과를 일반화할 수 있다.

1. Inx1 → x1 즉 In = I

2. Cnx1x2x3 → x1x2x3, 즉 n이 짝수일 때 Cn = I
Cnx1x2x3 → x1x3x2, n이 홀수일 때 Cn = C

3. Wnx1x2 → $\underbrace{x1x2 \cdots\cdots x2}_{n+1\text{개}}$

4. Knx1x2 ··· xn+1 → x1
Kn은 (n+1) 개의 논항을 필요로 하고 첫 번째 논항을 제외하고는 모두 삭제된다.

5. Bnx1x2 ··· xn+2 → x1 (x2 ···xn+2)
Bn은 (n+2) 개의 논항을 필요로 하고 첫 번째를 제외하고 나머지 모든 논항은 괄호로 묶인다.

몇가지 사례를 보도록 하자.

① WB2abcd → B2aabcd → a (abc)d
② BW3abcd → W3 (ab)cd → abccccd
③ B2W3abcd → W3 (abc)d → abcdddd

마지막 두 예는 B가 다른 연산자의 효력을 늦추는 작용을 한다는 것을 보여준다. BX일 때 X는 세 번째 논항에, B2일 때는 네 번째 논항에 작용하므로, Bkx에서 x는 (k+2)번째 논항부터 작용한다고 말할 수 있다.

이상의 여러 절차들을 사용하면 다양한 문제들을 풀 수 있다. 예컨대 앞에서 도입한 S와 동일한 X, 즉 Xabc → ac(bc)가 되는 그런 연산자를 구해보도록 하자.

① 먼저 기호열을 확인한다. Iabc → abc
② c를 반복하려면 BWabc → abcc
③ b와 c를 치환하려면 Cabcc → acbc
④ b와 c를 연결하려면 BBacbc → ac(bc)

그래서 X = I · (BW) · C · (BB)
= B (I · (BW) · C)(BB)
df

= B (B (I · (BW)) C)(BB)
df

= B (B (BI (BW) C)(BB)가 된다.
df

이 연산자가 기호열 abc에 작용하여 ac(bc)가 되므로

X = S . CQFD.

Ⅶ. 조합논리와 λ-연산

Curry의 조합논리는 속박변수가 있는 Church의 λ-연산으로 번역될 수 있다. 각각의 조합자들은 정의상 다음과 같이 번역된다.

I	$\lambda x.x$
K	$\lambda x \lambda y.x$
C	$\lambda xyz.xzy$
W	$\lambda xy.xyy$
B	$\lambda xyz.x(yz)$
S	$\lambda xyz.xz(yz)$
Φ	$\lambda xyzw.x(yw)(zw)$
Ψ	$\lambda xyzw.x(yz)(yw)$

하나의 예시를 통해 살펴보면 이 정의의 의미가 분명히 드러날 것이다. 이를 위해 β-환원(β-reduction)을 도입해야 하는데 이는 다음과 같이 정의된다.

$$(\lambda x.\ Y)X \xrightarrow[\beta]{} [x := X]Y$$

이제 Babc가 a(bc)가 됨을 보이도록 하자.

$$\mathrm{Babc} \quad = \quad (\lambda xyz.x(yz))abc$$[4]

4) 여기에서 x, y, z는 a, b, c에서 자유변수로 나타날 수 없도록 선정된 변수들이다.

$$= \quad (\lambda x(\lambda y(\lambda z.x(yz))))abc$$

$$\underset{\beta}{\rightarrow} \quad (\lambda y(\lambda z.a(yz)))bc)$$

$$\underset{\beta}{\rightarrow} \quad (\lambda z.a(bz))c$$

$$\underset{\beta}{\rightarrow} \quad a(bc) \qquad \text{CQFD.}$$

λ-연산자에 의해 술어(predicate)의 불포화성(non-saturation)을 명시적으로 나타낼 수 있다. 여기에는 발화자가 주어 또는 목적어를 테마로 설정하는지에 따라 두 가지 방식이 있다.

"like" ┌ (λx.λy[x likes y])
　　　└ (λy.λx[x likes y])

여기에 상항을 부여하면,

① (λx.λy[x likes y]) Jim Mary

$\underset{\beta}{\rightarrow}$ ([x:=Jim](λy.[x likes y])) Mary

= (λy.[Jim likes y]) Mary

$\underset{\beta}{\rightarrow}$ [y:=Mary]([Jim likes y])

= Jim likes Mary

② (λy.λx.[x likes y]) Mary Jim

$\underset{\beta}{\rightarrow}$ ([y:=mary](λx.[x likes y])) Jim

= (λx.[x likes Mary]) Jim

$\xrightarrow[\beta]{}$ [x:=Jim]([x likes Mary])

= Jim likes Mary

가 각각 도출된다.

Ⅷ. 유형 (Types) 도입

이제까지 우리는 여러 대상들을 다루어오면서 정작 그 대상들이 유형 (또는 범주)에 대해서는 언급하지 않았는데 유형의 개념은 범주문법의 핵심이 되므로 결론을 대신해 유형이론의 기초적인 의미를 소개하고자 한다.

x가 α라는 범주에 속할 때 우리는 $x \in \alpha$로 쓸 것이다. 연산자 F (→)를 도입하여 유형을 정의하자.

1) 기본유형은 w(대상)과 π(명제)이다.
2) α와 β가 범주이면 $F\alpha\beta$도 범주이다.
2) ab $\in \alpha$이고 b $\in \beta$이면 a $\in F\beta\alpha$이다.

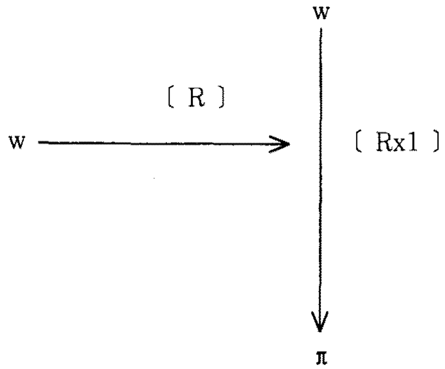
예컨대 대상 x, 대상범주 w, 일항술어 f, 명제범주 π가 주어졌을 때 fx는 명제이다.

$$fx \in \pi \quad \left[\begin{array}{l} x \in w \\ f \in Fw\pi \end{array} \right.$$

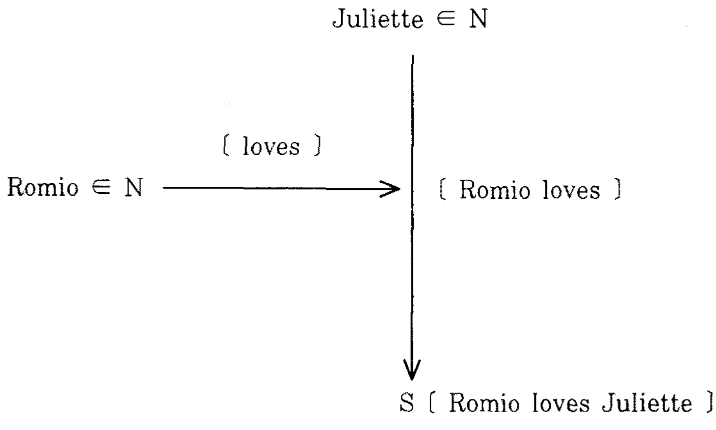
[x]를 x가 속하는 범주를 나타낸다고 할 때 다음과 같이 표현할 수도 있다.

$$w \xrightarrow{\quad [f] \quad} \pi$$

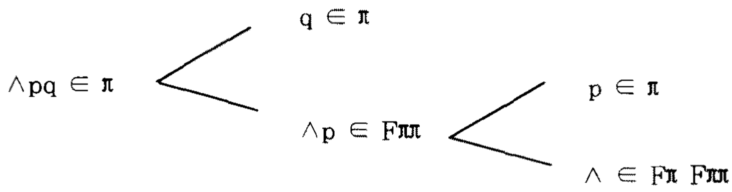
x1, x2가 두 개의 대상일 때 2항술어인 관계 R은 어떤 유형에 속할까? Rx1x2가 명제이고 x2가 대상이므로 Rx1은 하나의 Fwπ이다. 계속해서 x1이 대상이므로 R의 범주는 FπFwπ가 되겠다.



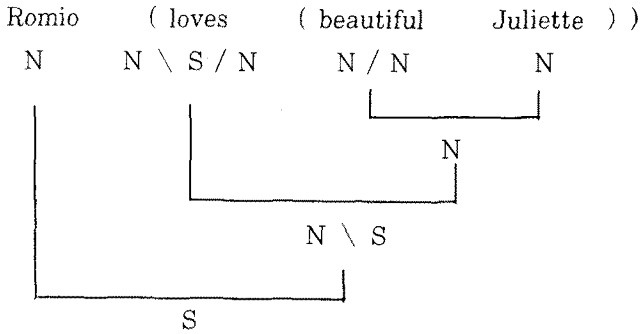
이는 즉각 자연언어 분석에 적용될 수 있다. "Romio loves Juliette"는 다음과 같이 분석할 수 있겠다.



"∧" (and)와 같은 2항 명제연산자는 p에 적용되어 1항 연산자 ∧p가 되고 다시 q에 적용되어 명제가 된다.



형용사가 어떤 유형인지 알아보기 위해 "Romio loves beautiful Juliette"라는 문장을 분석해 보자.

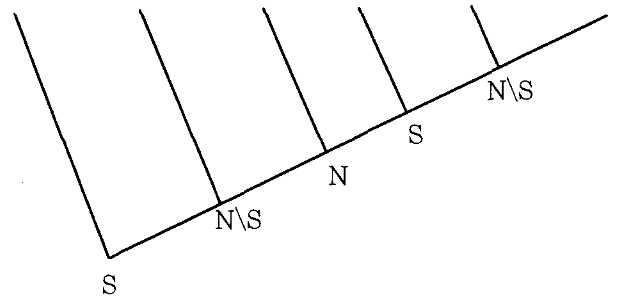


형용사는 명사를 취해 다시 명사가 되는 유형임을 즉각 알 수 있다. X / Y는 피연사자 Y가 오른 쪽에 있는 것을 나타내고, Y \ X는 피연산자 Y가 왼 쪽에 있음을 표시한다. 1차 방정식을 풀듯이 이렇게 축차적으로 진행해 나가면 문장의 어떤 요소라도 빠짐없이 그 유형을 알 수 있을 것이다.

부사와 접속사의 유형을 알아보기 위해 하나의 문장을 분석해 보자.

Georges believes that Mary sleeps soundly.

N (N\S)/N N/S N N\S (N\S)\(N\S)



이렇게 해서 문장의 그 어떤 성분이라도 유형을 알 수 있고, 유형을 알게 되면 문장에서 그 성분의 기능과 역할을 뚜렷하게 파악할 수 있는 것이다.

끝으로, λ-연산자를 도입하면 이른바 '일반화된 양화사'를 적절히 표현하는 일이 가능하다.

$$\text{' everybody '} =_{df} \lambda P(\forall x)(P(x))$$

$$\text{' someone '} =_{df} \lambda P(\exists x)(P(x))$$

$$\text{' every '} =_{df} \lambda P \lambda Q(\forall x)(P(x) \rightarrow Q(x))$$

$$\text{' some '} =_{df} \lambda P \lambda Q(\exists x)(P(x) \wedge Q(x))$$

이제 동일한 문장을 통사론적으로 분석한 것과 함수적으로 해석한 것을 비교해보자.

Everybody is pretty.

① 통사적 분석

Everybody　　is　　pretty.
S/(S\N)　　　　S\N
S

② 함수적 해석

π : (FFw$\pi\pi$: everybody) (Fwπ : is - pretty)
' everybody ' = λ P . $\forall$ x (P (x))
df
(FFw$\pi\pi$: λ P ($\forall$ x) (P (x)) (Fwπ : is - pretty)
($\forall$ x) (is - pretty (x))

이렇게 범주문법의 틀 내에서 수행되는 통사적 분석은 λ-연산의 함수적 표현으로 나타낼 수 있는데, 그것은 통사적 유형을 논리적 유형으로 전환함으로써 가능해지는 것이다.

나아가서 보편추론양화사 (illative universal quantifiers)를 도입하면, 그들의 유형은 다음과 같다.[5)]

FFw$\pi\pi$: Ⅱ1　　　FFwπFFw$\pi\pi$: Ⅱ2

Fwπ 유형의 두 술어 f와 g를 취하면, 고려되는 명제는 다음과 같이 얻어진다.

FFw$\pi\pi$: π1　　　Fwπ : g
π1 : Ⅱ1g

5) Ⅱ1 : (w → π) → π, Ⅱ2 : (w → π) → ((w → π) → π)

$$FFw\pi FFw\pi\pi : \pi 2 \qquad Fw\pi : f$$
$$FFw\pi\pi : \pi 2f \qquad Fw\pi : g$$
$$\pi 2 : \Pi 2fg$$

실상, $\Pi 1 \underset{df}{=} \lambda P.(\forall x)(P(x))$ 이고,

$\Pi 2 \underset{df}{=} \lambda P.\lambda Q.(\forall x)(P(x) \rightarrow Q(x))$ 이므로

$\pi 1$과 $\pi 2$는 자명하다.

$\pi 1$: everybody is pretty

$\pi 2$: every girl is pretty

존재추론양화사 (illative existential quantifiers)의 경우에도 앞에 나온 정의를 사용하면 쉽게 다음 두 문장을 도출할 수 있겠다.

$\pi 3$: someone is pretty

$\pi 4$: some girl is pretty

짐작할 수 있듯이 이 분석의 이점은 논란의 여지가 없지 않은 속박변수를 도입하지 않았다는 데에 있다.

Ⅷ. 나오면서

조합논리는 형식적 사고와 인문학에서 실질적 이점을 제공한다. 왜냐하면 그것은 추상적 개념이나 '사고의 대상'을 표현할 수 있기 때문이다. 정상형태(NF)를 갖지 못하는 WWW와 같은 조합표현도 엄연히 사고의 대상이며 하나의 개념인 것이다. 이런 표현과 의미와의 관계 그리고 지시체와의 관계는 또다른 문제이다. 끝으로 자연언어의 경우 명사와 문장 유형을 제외한 나머지 모든 유형은 파생유형으로서 이들은 함수자 (functor) 역할을 하며, 이들 덕분에 문장의 각각의 성분은 완벽하게 분석될 수 있음을 확인하였다.

조합논리로부터 λ - 계산으로의 이행은 조합자들의 정의에서 보듯이 거의

즉각적이다. 사실 조합논리와 λ - 계산은 외연적으로 동치이다.

참고문헌

Bach, E., "Categorial Grammars as theories of language" dans Oehrle et alii, 1988, pp. 17-34

Desclés, Jean-Pierre, *Langages applicatifs, langue naturelle et cognition*, Paris, Hermès, 1990

Ginist, Jean-Pierre, *La logique combinatoire*, PUF, Que sais-je? 3205, 1997,

Girard Y., Lafont P., Taylor, Proofs and Types, Cambridge University press, 1989

Gross M., Lentin A., *Notions de grammaires formelles*, Gauthier - villars

Hindley J. R., Seldin J. P., I*ntroduction to Combinators and Lambda-Calculus*, Cambridge univ. Press

Lambek J., *"The Mathematics of Sentence structure"*, American Mathematical Monthly, 65, 1958, pp. 154-165

Lesniewski S. T., *Sur les fondements de la mathématique*, Traduit du polonais par G. Kalinowski, Hermès. 1989

Miéville D., Vernant D. (éd.), *Stanislaw Lesniewski Aujourd'hui*, Recherches sur la philosophie et le langage, N0 16, 1995

Oehrle R. T., BACH E., Wheeler D. eds., *Categorial grammars and Naturel langages Structures*, D. Reidel, 1988

Schonfinkel, "Uber die Bausteine der mathematischen Logik", *Mathematischen Annalen*, 92, 1924, traduit en français in *Mathématiques et informatique appliquées aux sciences humaines*, MISH, 112, 1990, pp. 5~26.