電子計算機 프로그래밍에 있어서의 歸納的 手法에 關한 研究

논 문
24~1~5

(A Study on the Recursive Technique in Digital Computer Programming)

林 濟 鐸*
(Lim, Chae Tak)

Abstract

A model computer is designed and using this model computer it is demonstrated that computable functions are equivalent to recursive functions. It is also shown that iteratively defined computable functions can be transformed to recursively defined computable functions and that recursive programming is possible in principle, when iterative programming is possible.

1. 序 論

Turing 機械의 意味에서 計算可能한 函數는 모두 原理的으로는 電子計算機로 計算할 수 있다는 것은 잘 알려진 事實이다[1]. 그러나 우리는 現在 電子計算機가 가지는 能力을 充分히 驅使하지 못하고 있는데 그 理由는 프로그래머로서의 우리의 能力不足에 起因하는 境遇가 많다. 이 能力을 向上시키기 爲하여 計算科學의 數學的研究가 많이 行해지고 있다[2]~[6].

그러나 지금까지의 理論은 主로 否定的인 問題의 發見或은 보다 一般的인 알고리듬의 摸索等 抽象的인 것이어서 그의 應用面에서 具體性을 缺하고 있다. 本論文은 歸納的函數의 應用的側面을 다룬 것으로서 하나의 模型計算機 MC를 設定함으로서 歸納的函數와 電子計算機 프로그램과의 關係를 論하고 反復的으로 定義된 函數와 歸納的으로 定義된 函數는 相互變換에 依해서 反復的 프로그램과 歸納的 프로그램은 原理的으로 相互變換이 可能함을 보였다.

2. 歸納的函數와 電子計算機프로그램

*正會員：漢陽工大敎授.

2—1. 計算可能函數

自然數 $\{0, 1, 2, \cdots\}$ 위에 定義되고 自然數의 값을 取하는 函數를 數論的函數라 한다. 數論的函數 $\varphi(x_1, x_2, \cdots, x_n)$가 任意의 變數의 組$(x_1, x_2, \cdots, x_n)$에 對하여 그의 函數値를 有限回의 操作으로 求할 수 있는 一般的인 節次가 存在할때 $\varphi(x_1, x_2, \cdots, x_n)$은 計算可能하다 하고 그 節次를 $\varphi(x_1, x_2, \cdots, x_n)$의 알고리듬(algorithm)이라 한다.

어떤 集合 M 위의 任意의 元의 組$(x_1, x_2, \cdots, x_n)$에 關한 命題 $P(x_1, x_2, \cdots, x_n)$의 成立如否를 有限回의 操作으로 判定할 수 있는 一般的인 節次가 存在할 때 그 節次를 命題 $P(x_1, x_2, \cdots, x_n)$의 알고리듬이라 하며 그 주어진 命題 P에 對한 알고리듬의 存否를 묻는 問題를 그 命題의 決定問題(dicision problem)라 한다. 命題 P에 對한 알고리듬이 存在하면 그 決定問題는 可解(solvable)라 하고 存在하지 않으면 非可解(non-solvable)라 한다.

有限的으로 生成된 集合 M 위의 命題는 素因數分解의 一意性에 依하여 自然數集合 N 위에 定義되는 命題로 寫像되며 N 위의 하나의 命題 $P(x_1, x_2, \cdots, x_n)$가 주어졌을때 다음과 같은 數論的函數 $\varphi(x_1, x_2, \cdots, x_n)$를 그의 特徵函數(representing function)라 한다.

$$\varphi(x_1, x_2, \cdots, x_n)=\begin{cases}0, & P(x_1, x_2, \cdots, x_n)\text{가 眞일 때}\\ 1, & P(x_1, x_2, \cdots, x_n)\text{가 僞일 때}\end{cases}$$

여기에서 P 의 決定問題가 可解이면 그의 眞僞를 決定하는 알고리듬이 存在하며 따라서 φ 의 값이 0인가 1인가를 決定하는 알고리듬이 存在하므로 φ 는 計算可能하다.

φ 가 計算可能하면 그 값을 有限回의 操作으로 計算하는 알고리듬이 存在한다. 지금 그에 따라 計算한 結果 0이 되면 P 는 眞이고 1이면 僞이다. 이는 命題 P 의 알고리듬이 存在함을, 따라서 그의 決定問題가 可解임을 나타낸다.

따라서 P 의 決定問題는 結局 어떤 하나의 數論的函數의 計算可能性을 判定하는 問題로 歸着된다.

計算可能函數의 간단한 例로는 各自然數의 後者(successor; $S(x)=x+1=x'$)를 만드는 操作을 自明한 것이라 한다면 두 自然數의 合 x_1+x_2 는 計算可能函數이다. 即 알고리듬이 存在한다. 왜냐하면

$$\varphi^+(x_1, 0)=x_1$$

$$\varphi^+(x_1, x_2')=(\varphi^+(x_1, x_2))' \tag{1}$$

에 依해서 任意의 自然數의 組(a, b)에 對해서 다음과 같은 節次에 依하여 $a+b=\varphi^+(a, b)$를 求할 수 있기 때문이다.

即 $a=6$, $b=3$일 때의 $6+3=\varphi^+(6, 3)$의 計算은

$$\begin{aligned}\varphi^+(6,3)&=(\varphi^+(6,2))'=((\varphi^+(6,1))')'\\ &=(((\varphi^+(6,0))')')'=(((6)')')'=((7)')'\\ &=(8)'=9\end{aligned}$$

即 a 의 後者를 만드는 操作을 繼續해서 b 回 施行함으로서 $a+b$ 를 만들 수 있다.

2—2 歸納的函數

다음 三種의 函數(基礎函數)

(1) $\varphi(x)=x+1=x'$

(2) $\varphi(x_1, x_2, \cdots, x_n)=x_i \ (1\leq i\leq n)$

(3) $\varphi(x_1, x_2, \cdots, x_n)=c$ (c 는 定數)

에서 出發하여 주어진 函數 $\psi, \chi, \chi_1, \cdots, \chi_m$ 으로 부터 새로운 函數를 만드는 操作으로서

(Ⅰ) $\psi(y_1, y_2, \cdots, y_m)$, $\chi_1(x_1, x_2, \cdots, x_n), \cdots, \chi_m(x_1, x_2, \cdots, x_n)$에서

$$\varphi(x_1, x_2, \cdots, x_n)=\psi(\chi_1(x_1, x_2, \cdots, x_n), \cdots, \chi_m(x_1, x_2, \cdots, x_n))$$

(Ⅱ) 定數 c 와 $\psi(x, y)$에서

$$\begin{cases}\varphi(0)=c\\ \varphi(x')=\psi(x, \varphi(x))\end{cases}$$

(Ⅲ) $\psi(z_1, z_2, \cdots, z_n)$, $\chi(x, y, z_1, z_2, \cdots, z_n)$에서

$$\varphi(0, z_1, z_2, \cdots, z_n)=\psi(z_1, z_2, \cdots, z_n)$$

$$\varphi(x', z_1, z_2, \cdots, z_n)=\chi(x, \varphi(x, z_1, z_2, \cdots, z_n), z_1, z_2, \cdots, z_n)$$

(Ⅳ) $\psi(x_1, x_2, \cdots, x_n, y)$가 주어졌을 때 $\forall x_1 \cdots \forall x_n \exists y(\psi(x_1, x_2, \cdots, x_n, y)=0)$가 成立할 때

$$\varphi(x_1, x_2, \cdots, x_n)=\min_y(\psi(x_1, x_2, \cdots, x_n, y)=0)$$

一般으로 (1)(2)(3)의 基礎函數를 出發點으로 하여 (Ⅰ)(Ⅱ)(Ⅲ)(Ⅳ)를 有限回 使用하여 만들어지는 函數를 歸納的函數(recursive function), 特히 (Ⅳ)를 한번도 使用하지 않고 만들어지는 歸納的函數를 原始歸納的函數(primitive recursive function)라 한다. 以下 前者를 간단히 rf, 後者를 pf 로 略記하기로 한다.

N 위의 命題 $P(x_1, x_2, \cdots, x_n)$의 特徵函數가 rf 또는 pf 일때 그 命題는 歸納的(recursive) 또는 原始歸納的(primitive recursive)이라 한다. 이들 命題를 간단히 rp 및 pp 로 略記하기로 한다.

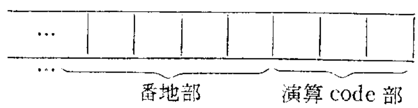
例를 들면 式 (1)은 (Ⅲ)을 使用한 pf 이다.

2—3 電子計算機 프로그램

計算可能函數를 電子計算機로 計算할 수 있는 函數로서 把握하기 위하여 우선 模型計算機 MC를 設定한다.

(1) MC는 二進機械로서 記憶裝置는 無限(countably infinite)히 많은 cell 을 가지며 各 cell 에는 自然數의 番地가 붙어 있다.

(2) 各 cell 의 構造는 다음과 같으며 i 番地의 cell 을 c_i 라 쓰기로 한다.



(3) MC 는 cell 과 同一한 構造의 累算器(Acc)를 하나 가지며 그 外의 置數器는 없다.

(4) MC 의 cell 或은 Acc에 記憶되는 語는 有限個의 1로 이루어지며 이를 數値語로 읽을때는 2進法으로 表現된 自然數로 看做한다.

(5) MC 는 다음과 같은 8個의 命令을 갖는다.

命令 code	命令 記號	意　味	命令 型式	演 算 內 容
000	C	clear	nC	[Acc]→Cn, 0→Acc
001	W	write	nW	[Cn]→output
010	A	Add	nA	[Acc]+[Cn]→Acc
011	S	subtract	nS	[Acc]−[Cn]→Acc
100	B	branch	nB	go to n
101	T	test	nT	[Acc]≒0면 go to n
110	E	end	nE	stop
111	R	read	nR	input→Cn

하나의 數論的函數 $\varphi(x_1, x_2, \cdots, x_n)$은 이 模型計算機 MC에 對하여 다음과 같은 프로그램이 存在할때 計算可能函數(cf)라 한다. 即 프로그램이 指定하는 cell

$ci_1, ci_2, \cdots, ci_n$ 에 $x_1, x_2, \cdots, x_n$ 을 記憶시키고 MC_j 를 始動하면 MC는 그 프로그램이 指定하는 cell cj 에 값 $\varphi(x_1, x_2, \cdots, x_n)$을 記憶시키고 停止한다.

모든 gf가 cf임을 證明하기 爲해서는 우선 基礎函數 (1), (2), (3)이 cf임을 보이고 다음에 이 cf에서 (Ⅰ), (Ⅱ), (Ⅲ), (Ⅳ)의 操作으로 만들어진 것이 또한 cf임을 보이면 된다[1].

여기에서는 그의 逆을 證明한다.

(1^c) 函數 $\varphi(x_1, x_2, \cdots, x_n)$이 cf라 하고 그것을 計算하는 프로그램 및 $x_1, x_2, \cdots, x_n$ 을 記憶裝置에 記憶시켰다 한다. 이때 다음과 같은 假定을 해도 一般性을 잃지 않는다.

(a) 프로그램은 c_0 에서 c_α 까지에 記憶되어 있다.

(b) $x_1, x_2, \cdots, x_n$ 은 $c_{\alpha+1}$ 에서 $c_{\alpha+n}$ 까지에 記憶되어 있다.

(c) 프로그램은 0番地부터 實行된다.

(d) 計算된 값 $\varphi(x_1, x_2, \cdots, x_n)$은 c_0 에 記憶시킨다.

(2^c) 지금 어느 時刻 t 에 있어서 0이 아닌 cell의 番地의 最大値를 ρ 라 하고 MC가 다음 時刻에 實行할 命令의 所在番地를 σ 라 한다. 各 $[c_i]$ 및 [Acc]는 各各 하나의 自然數이므로 時刻 t 에 있어서의 MC의 記憶裝置, Acc 및 制御의 狀態는 n 번째의 素數를 $P(n)$ 으로 表示하면

$$P(1)^{(Acc)}P(2)^{\sigma}P(3)^{(c_0)}P(4)^{(c_1)}\cdots P(\rho+3)^{(c_\rho)}$$

인 하나의 自然數로 表現된다. 이것을 時刻 t 에 있어서의 MC의 狀態라 한다.

(3^c) 어느 時刻에 있어서의 MC의 狀態 q 가 決定되면 다음 時刻의 狀態는 存在하는 限 唯一하게 決定되며 이것을 q 의 다음 狀態라 하고 $\bar{q}$ 로 쓴다. 여기에서

$$\psi(x)=\begin{cases}\bar{x}, & x\text{가 하나의 狀態이고 다음 狀態가 存在할때}\\ 0, & \text{그렇지 않을때}\end{cases}$$

를 定義하면 $\psi(x)$는 前述한 pf 및 경우에 따른 定義를 適用하여 하나의 pf임이 證明된다[6].

(4^c) 처음에 cell $c_0, c_1, \cdots, c_\alpha$ 에 記憶시킨 프로그램은 自然數의 列 $n_0, n_1, \cdots, n_\alpha$ 이며 이것은 또 하나의 自然數

$$c=P(1)^{n_0}P(2)^{n_1}\cdots P(\alpha+1)^{n_\alpha}$$

로 表現할 수 있다.

$n_i=(e)_i (i=0, 1, \cdots, \alpha)$로 表示하면 MC의 初期狀態는 $P(1)^0P(2)^0P(3)^{(e)0}P(4)^{(e)1}\cdots P(\alpha+3)^{(e)\alpha}$

$$P(\alpha+4)^{x_1}\cdots P(\alpha+n+3)^{x_n}$$

이것은 $e, x_1, x_2, \cdots, x_n$ 의 pf이며 이것을

$$q(e, x_1, x_2, \cdots, x_n)$$

이라 쓴다.

(5^c) 各狀態

$$x=P(1)^{(Acc)}P(2)^{\sigma}P(3)^{(c_0)}\cdots P(\rho+3)^{(c_\rho)}$$

에 $[c_0]$를 對應시키는 函數 $\xi(x)$는 pf이다.

(6^c) 차례로 다음과 같이 놓으면 이들은 모두 gf이다.

(a) $\begin{cases}\varphi_1(0, e, x_1, \cdots, x_n)=q(e, x_1, \cdots, x_n)\\ \varphi_1(x', e, x_1, \cdots, x_n)=\psi(\varphi_1(x, e, x_1, \cdots, x_n))\end{cases}$

$\varphi_1(x, e, x_1, \cdots, x_n)$은 MC의 x 時間後의 狀態를 表示한다.

(b) $\varphi_2(e, x_1, \cdots, x_n)=\min_x(\varphi_1(x', e, x_1, \cdots, x_n)=0)$

$\varphi_2(e, x_1, \cdots, x_n)$은 MC의 動作時間이다.

(c) $\varphi_3(e, x_1, \cdots, x_n)=\varphi_1(\varphi_2(e, x_1, \cdots, x_n), e, x_1, \cdots, x_n)$

이것은 MC가 停止한 時刻의 狀態를 表示한다.

(d) $\varphi_4(e, x_1, \cdots, x_n)=\xi(\varphi_3(e, x_1, \cdots, x_n))$

$$\varphi_4(e, x_1, \cdots, x_n)=\varphi(x_1, x_2, \cdots, x_n)$$

따라서 $\varphi(x_1, x_2, \cdots, x_n)$은 gf이다.

이로서 cf와 gf는 同一槪念임을 밝혔다. 그런데 gf는 前述한 바와 같이 基礎函數 (1), (2), (3)에 操作 (Ⅰ), (Ⅱ), (Ⅲ), (Ⅳ)를 有限回 適用함으로서 만들어지는 것이므로 이 定義式을 모두 整列시키면 函數를 計算하는 하나의 一般的인 節次가 된다. 따라서 電子計算機로 프로그램 할 수 있는 函數는 歸納的으로 定義되는 函數이며 그 函數를 計算하는 알고리듬을 記述하는 電子計算機 프로그램은 歸納的定義의 表現式을 나열한 것이다.

3. 歸納的 프로그램

反復的으로 定義된 函數는 反復的手法으로 프로그램할 수 있다[4]. 따라서 그 函數의 歸納的定義 및 歸納的 프로그램이 可能하다.

먼저 階乘函數의 두가지 定義를 생각한다.

$$\mathrm{Fact1}(n)=((n=0)\to 1,\ P(n,\ \mathrm{Fact1}(\delta(n)))) \quad (2)$$

$$\mathrm{Fact2}(n)=\phi(n, 1)$$

$$\phi(n, m)=((n=0)\to m,\ \phi(\delta(n),\ P(n, m))) \quad (3)$$

여기에서 δ 는 先者(predecessor), P 는 乘算函數이다. (2)와 (3)은 變形하면

$$\mathrm{Fact\,1}(n)=P(n, P(\delta(n), P(\delta^2(n), \cdots, P(\delta^{\lambda-1}(n), 1))\cdots)) \quad (4)$$

$$\mathrm{Fact\,2}(n)=P(\delta^{\lambda-1}(n), \cdots, P(\delta^2(n), P(\delta(n), P(n, 1))\cdots))) \quad (5)$$

여기에서 $\delta^2(n)$은 $\delta(\delta(n))$이고 λ 는 $\delta^\lambda(n)=0$이 되는 整數이다. 그런데 乘算函數 P 는 $P(\alpha, P(\beta, \gamma))=P(\beta, P(\alpha, \gamma))$의 性質이 있으므로 이 關係를 反復適用함으로서

$$\begin{aligned}
\text{Fact }2(n) &= P(\delta^{\lambda-1}(n), P(\delta^{\lambda-2}(n), \cdots, \\
&\quad P(\delta^2(n), P(\delta(n), P(n,1))\cdots)) \\
&= P(\delta^{\lambda-2}(n), P(\delta^{\lambda-1}(n), \cdots, \\
&\quad P(\delta^2(n), P(\delta(n), P(n,1))\cdots)) \\
&= P(\delta^{\lambda-2}(n), P(\delta^{\lambda-3}(n), \cdots, \\
&\quad P(\delta(n), P(n, P(\delta^{\lambda-1}(n),1))\cdots)) \\
&= P(\delta^{\lambda-3}(n), P(\delta^{\lambda-4}(n), \cdots, \\
&\quad P(n, P(\delta^{\lambda-2}(n), P(\delta^{\lambda-1}(n),1))\cdots)) \\
&= P(\delta(n), P(n, P(\delta^2(n), \cdots, \\
&\quad P(\delta^{\lambda-2}(n), P(\delta^{\lambda-1}(n),1))\cdots)) \\
&= P(n, P(\delta(n), P(\delta^2(n), \cdots, \\
&\quad P(\delta^{\lambda-2}(n), P(\delta^{\lambda-1}(n),1))\cdots)) \\
&= \text{Fact } 1
\end{aligned}$$

卽 두 定義는 同値이다.

다음에 이를 一般化하기 爲하여 (2), (3)을 다음과 같이 쓴다.

$$F1(n) = ((n=L) \rightarrow B,\ H((n), F1(\delta(n)))) \quad (6)$$

$$F2(n, A) = ((n=L) \rightarrow A,\ G(\delta(n),\ E(n, A))) \quad (7)$$

이것은 0, 1 대신 L, B로 P대신 (6)에서는 H로 (7)에서는 E로 置換한 것이다. 定義 (6), (7)은

$$H(\alpha, \beta) = E(\alpha, \beta) \quad (8)$$

및 $H(\alpha, E(\beta, \gamma)) = E(\beta, H(\alpha, \gamma))$ (9)

가 成立하면 同値이다[6].

지금 N을 list라 생각하면(LISP 言語로 表現해서)

$\delta(N) = \text{cdr}(n)$

$H(N, A) = \text{append}(\text{car}(n)\ ;\ A)$

$B = L = NIL$

따라서

$F1(N) = (\text{null}(N) \rightarrow NIL\ ;\ \text{append}(\text{car}(N)\ ;\ F1(\text{cdr}(N)))$

$F2(N) = G(N\ ;\ NIL)$

$G(N\ ;\ A) = (\text{null}(N) \rightarrow A\ ;\ G(\text{cdr}(N)\ ;\ E(n\ ;\ A)))$

그런데

$H(N\ ;\ A) = \text{append}(\text{car}(N)\ ;\ A)$

이므로

$H(a\ ;\ E(b\ ;\ c)) = \text{append}(\text{car}(a)\ ;\ E(b\ ;\ c))$

따라서 list는 $(E(b, c), \text{car}(a))$가 된다. 만일 $E(b\ ;\ c) = \text{cons}(\text{car}(b)\ ;\ c)$면 $H(a\ ;\ E(b\ ;\ c))$의 list는 $(\text{car}(b), c, \text{car}(a))$가 된다. 그리고

$$\begin{aligned}
E(b\ ;\ H(a\ ;\ c)) &= \text{cons}(\text{car}(b)\ ;\ H(a\ ;\ c)) \\
&= \text{cons}(\text{car}(b)\ ; \\
&\quad \text{append}(\text{car}(a)\ ;\ c))
\end{aligned}$$

의 list도 역시 $(\text{car}(b), c, \text{car}(a))$이다. 또

$H(N, NIL) = \text{append}(\text{car}(N)\ ;\ NIL)$,

list는 $(\text{car}(N))$

$E(N, NIL) = \text{cons}(\text{car}(N)\ ;\ NIL)$,

list는 $(\text{car}(N))$

卽 (8), (9)가 成立하고 두 定義는 同値이다.

따라서 歸納的手續과 反復的手續은 相互 變換할 수 있으며 다시 말하면 計算可能函數는 反復的 프로그램으로 計算할 수 있을뿐만 아니라 歸納的 프로그램으로 計算할 수도 있다.

例로서 Algol에 依한 階乘函數의 프로그램을 들면 反復的 프로그램은

```
INTEGER PROCEDURE Fact(n);
VALUE n; INTEGER n;
BEGIN REAL f; INTEGER i; f:=1;
  FOR i:=1 STEP 1 UNTIL n DO
f:=f×i; Fact:=f
END;
```

歸納的 프로그램은

```
INTEGER PROCEDURE Fact(n);
VALUE n; INTEGER n; Fact:=IF n=0
THEN 1 ELSE n×Fact(n-1);
```

프로그램의 容易性, 明快性等에 있어 歸納的 프로그램이 有利하며 이것은 定義가 本質的으로 歸納的이기 때문이다.

4. 結 論

計算可能函數는 歸納的으로 定義되는 函數이고 그것을 計算하는 알고리듬을 記述하는 電子計算機 프로그램은 歸納的定義의 表現式을 羅列한 것이다. 原理的定로는 反復的으로 處理할 수 있는 것은 歸納的 프로그램으로 處理할 수 있으며 歸納的 프로그램은 歸納的定義式를 그대로 프로그램 言語로 옮겨 쓰면 되므로 定義가 本質的으로 歸納的인 函數는 歸納的 프로그램이 容易하다.

實際의 計算機에서 歸納的 프로그램을 實現하는 方法에 關해서는 다음 機會에 다루기로 한다.

參 考 文 獻

(1) Minsky, M.; Computation Finite and Infinite Machines, Prentice-Hall, Englewood Cliffs, N.J., pp.208~210, 1967.

(2) Wang, Hao; A Variant to Turings Theory of Computing Machines, Journal of ACM, Vol. 4, No.1, p.63, 1957.

(3) Dijkstra, E.W.; Recursive Programming, Numerische Mathematik, Vol. 2, No.5, p.312, 1960.

(4) McCarthy, J.; Recursive Functions of Symbolic Expressions and their Computation by Machine, Part I, Communications of ACM. Vol. 3, p.184, 1960.

(5) ——; A Basis for a Mathematical Theory of Computation, Computer Programming and Formal Systems, Braffort and Hirschberg (eds.), North Holland, Amsterdam, p.33, 1967.

(6) Cooper, D.C.; The Equivalence of Certain Computation, Computer J., Vol. 9, No.1, p. 45, 1966.