Y コンビネータ

金谷一朗

2008 年 12 月 4 日

1　ラムダ記法

ふつう式 $y = f(x)$ と書くところを今後 $y = fx$ と書く．式 $f(x) = x^2$ と書くところを $f = \Lambda x.\, x^2$ と書く．

2　関数名を使わない再帰

関数 F があり Fn が正の整数 n の階乗を表すとする（すなわち $Fn = n!$ である）．関数 F を定義する．

$$F \equiv \Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times F(n-1) & \text{otherwise} \end{array} \right.$$

定義式の両辺に F があるので，右辺の F を消す努力をこれからする．まず関数 F を返す関数 F' を定義する．

$$F' \equiv \Lambda. \left(\Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times F(n-1) & \text{otherwise} \end{array} \right. \right)$$

関数 F が右辺に残っているので，これを引数とする．

$$F'' \equiv \Lambda \phi. \left(\Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times \phi(n-1) & \text{otherwise} \end{array} \right. \right)$$

これで右辺から F が消えた．関数 F は $F''F$ とすれば手に入る．しかし手元に F があるのであれば，最初から F を使えばよいので，これでは本末転倒である．では $F''F''$ ならどうだろうか．

$$\begin{aligned} F''F'' &= \left(\Lambda \phi. \left(\Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times \phi(n-1) & \text{otherwise} \end{array} \right. \right) \right) F'' \\ &= \Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times F''(n-1) & \text{otherwise} \end{array} \right. \end{aligned}$$

であって，$F''(n-1)$ という無意味な項が顔を出す．そこで関数 F'' を次のように書き換える．

$$F''' \equiv \Lambda \phi. \left(\Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times (\phi\phi)(n-1) & \text{otherwise} \end{array} \right. \right)$$

この関数 F''' を

$$(F'''F''')5$$

のように呼び出せば階乗を計算することができる．実際

$$(F'''F''') = \Lambda n. \left\{ \begin{array}{rl} 1 & \text{if } n = 0 \\ n \times (F'''F''')(n-1) & \text{otherwise} \end{array} \right.$$

であるから，F の定義式において $F \to F'''F'''$ と置き換えれば納得できる．このようにして，右辺から左辺に現れる項を消し去ることができた．

3　Z コンビネータ

関数 $F'''F'''$ を生成する関数を考える．一般に ϕn と $(\Lambda\nu.\,\phi\nu)n$ の評価値は同じであるから，関数 F''' の定義式中の

$$(\phi\phi)(n-1)$$

は

$$(\Lambda\nu.(\phi\phi)\nu)(n-1)$$

と書き直せる．よって F''' は

$$F''' \equiv \Lambda\phi.\left(\Lambda n.\left\{\begin{array}{rl} 1 & \text{if } n=0 \\ n\times(\Lambda\nu.(\phi\phi)\nu)(n-1) & \text{otherwise}\end{array}\right.\right)$$

と書ける．引数に $\Lambda\nu.\,(\phi\phi)\nu$ を期待できるのであれば，関数 F''' に代えて次の関数 F_0 を階乗関数を生成する関数として利用できる．

$$F_0 \equiv \Lambda f.\left(\Lambda n.\left\{\begin{array}{rl} 1 & \text{if } n=0 \\ n\times f(n-1) & \text{otherwise}\end{array}\right.\right)$$

（これを大元の F の定義式と比べてみると面白い．）

もともとの F''' と F_0 の関係は

$$F''' = \Lambda\phi.F_0(\Lambda\nu.(\phi\phi)\nu)$$

である．階乗関数 $F'''F'''$ は次のようにして得られる．

$$\begin{aligned} F'''F''' &= \big(\Lambda\phi.F_0(\Lambda\nu.(\phi\phi)\nu)\big)\big(\Lambda\phi.F_0(\Lambda\nu.(\phi\phi)\nu)\big) \\ &= \Big(\Lambda\Phi.\big(\Lambda\phi.\Phi(\Lambda\nu.(\phi\phi)\nu)\big)\big(\Lambda\phi.\Phi(\Lambda\nu.(\phi\phi)\nu)\big)\Big)F_0 \\ &= ZF_0 \end{aligned}$$

最後の行で

$$Z \equiv \Lambda\Phi.(\Lambda\phi.\Phi(\Lambda\nu.(\phi\phi)\nu))(\Lambda\phi.\Phi(\Lambda\nu.(\phi\phi)\nu))$$

とした．非再帰関数 F_0 を ZF_0 とすることで再帰関数を得られた．

関数 Z は関数 F_0 のディテールに関係なく，関数を返す関数を再帰させる．関数 Z を Z コンビネータと呼ぶ．

4　Y コンビネータ

Z コンビネータ Z は引数 ϕ を遅延評価している．遅延評価しない Z コンビネータを Y コンビネータと呼ぶ．（Z コンビネータを Y コンビネータに含む場合もある．）Y コンビネータは以下のかたちをしている．

$$Y \equiv \Lambda\Phi.(\Lambda\phi.\Phi(\phi\phi))(\Lambda\phi.\Phi(\phi\phi))$$

遅延評価を行わない言語（Scheme を含む）では，このオリジナルの Y コンビネータは役に立たない．かわりに Z コンビネータを使うべきである．

付録 A　チャーチ数

正の整数は次のようにラムダ式で表現できる．

$$
\begin{aligned}
0 &\equiv \Lambda fx.x \\
1 &\equiv \Lambda fx.fx \\
2 &\equiv \Lambda fx.ffx \\
3 &\equiv \ldots
\end{aligned}
$$

Successor 演算子 S を次のように定義しておけば，任意の自然数は 0 から生成できる．

$$S \equiv \Lambda nfx.f(nfx)$$

例として $S1 = 2$ を確かめる．

$$
\begin{aligned}
S1 &= (\Lambda nfx.f(nfx))(\Lambda fx.fx) \\
&= \Lambda fx.f((\Lambda fx.fx)fx) \\
&= \Lambda fx.f(\Lambda x.(fx)x) \\
&= \Lambda fx.ffx \\
&= 2
\end{aligned}
$$

このように，計算機で扱う全ての数はラムダ式だけで記述可能である．

付録 B　真偽値と条件分岐

真偽値は次のようにラムダ式で表現できる．

$$
\begin{aligned}
T &\equiv \Lambda xy.x \\
F &\equiv \Lambda xy.y
\end{aligned}
$$

If 文に相当する関数 P は次のように表現できる．関数 P は引数 p, x, y をとり，$p = T$ であれば x を，$p = F$ であれば y を返す．

$$P \equiv \Lambda pxy.pxy$$

関数 P が if 文であることの証明は次の通り．

$$
\begin{aligned}
PTXY &= (\Lambda pxy.pxy)(\Lambda xy.x)XY \\
&= (\Lambda xy.x)XY \\
&= X \\
PFXY &= (\Lambda pxy.pxy)(\Lambda xy.y)XY \\
&= (\Lambda xy.y)XY \\
&= Y
\end{aligned}
$$

上述のように正の整数，真偽値，条件分岐がラムダ記法だけで表現できる．また Y コンビネータがあれば再帰が手に入るため，繰り返しも表現できる．条件分岐と繰り返しがあれば任意のアルゴリズムを表現できる．事実，ラムダ式はチューリング完全である．

ラムダ記法は万能チューリング機械のもうひとつの表現とも言える．

付録 C　カリー化

これまでの説明で複数の引数をとる関数（ラムダ式）を紹介した．複数の引数をとるラムダ式は，ひとつの引数しかとらないラムダ式へ展開できる．例えば次の通り．

$$\Lambda xy.(x+y) = (\Lambda x.(\Lambda y.(x+y)))$$

複数の引数をとるラムダ式をひとつの引数しかとらないラムダ式へ展開することをカリー化と呼ぶ．

付録 D　なんでもラムダ

この節（説）は余興である．

物理学ではある変数 x がパラメータ t に依存することを伝統的に

$$x = x(t)$$

と書く．ラムダ記法を使えばより意図がはっきりする．

$$x = \Lambda t \ldots$$

運動方程式は物理法則であって，関数の定義ではない．従ってニュートンの運動方程式 $F = \dot{p}$ はラムダ記法でも $F = \dot{p}$ である．しかしラグランジアンを使った運動方程式には定義式が必要である．ラグランジアンの定義は以下の通り．

$$L \equiv \Lambda q\dot{q}.Tq\dot{q} - Uq\dot{q}$$

一様重力中（重力ポテンシァルを g とする）での質点（質量を m とする）の運動を考える場合は

$$T \equiv \Lambda q\dot{q}.\frac{1}{2} \times m \times \dot{q}^2;\ U \equiv \Lambda q\dot{q}.g \times m \times q$$

である．運動方程式は以下のようになる．

$$\frac{\partial}{\partial \dot{q}}\dot{L}q\dot{q} - \frac{\partial}{\partial q}Lq\dot{q} = 0$$

もちろん微分演算子もラムダ記法で定義できる．例えば $\partial/\partial q$ は次の通り．

$$\frac{\partial}{\partial q} \equiv \left(\Lambda hfq.\lim_{h\to 0}\frac{f(q+h)-fq}{h}\right)h$$

このようにラムダ記法で記述する方が，伝統的な $(\partial/\partial q)f \equiv \ldots$ という書き方よりも，微分演算子が「飢えて」いることを明示する．また，ラムダ記法は微分演算子もまた関数（あるいは写像）であることを容易に理解させるという点で伝統的記述法よりも優れている．