3　ラムダ計算

- プログラムの意味論を展開する際には，本質的な部分だけを残した言語として，ラムダ計算の理論体系を用いると便利
- ラムダ計算は，ML，Miranda，Haskell など関数型プログラミング言語の基礎

3.1　λ 抽象と関数適用

3.1.1　λ 抽象

例：加算 + が定義されているとして，新しく 1 を加算する関数：

直観的　：$\mathrm{add1}(x) = x + 1$
λ **抽象**　：関数名は計算にとって本質的でない $\Rightarrow \lambda x.x + 1$

- **仮引数**（formal parameter）を指定して，式から直接関数を定義する操作を λ **抽象**（λ-abstraction）あるいは λ **束縛**（λ-binding）と呼ぶ．

3.1.2　関数適用

定義した関数を呼び出す機構

例： **実引数**（actual parameter）を与えて関数 add1 を呼び出す：

直観的　：$\mathrm{add1}(2)$
関数適用　：$\lambda x.x + 1$ を 2 に適用する $\Rightarrow (\lambda x.x + 1)(2)$

- 実引数を与えて関数を呼び出す操作を**関数適用**（function application）と呼ぶ．
- 関数適用の実行は，$x + 1$ の仮引数 x が実引数 2 に置き換わり，$2 + 1$ が生成されて加算が実行され，答え 3 が得られる．仮引数 x を実引数 2 で置き換える操作を β **変換**（β-conversion）と呼ぶ．

$$(\lambda x.x + 1)(2) = 2 + 1 = 3$$

2 引数以上の関数　：

直観的　：$f(x, y) = 2 * x + y$
λ **抽象**　：$(\lambda(x, y).2 * x + y)(3, 4)) \overset{\text{カリー化}}{\Longrightarrow} (\lambda x.(\lambda y.2 * x + y))(3)(4)$

- 2 引数以上の関数を 1 引数の関数で表す操作を**カリー化**（currying）と呼ぶ．
- 関数を引数に受け取る関数や関数を値とする関数を**高階関数**（higher order function）と呼ぶ．

3.2　λ 式と β 変換

ラムダ計算の正確な定義と用語の解説

定義 3.1. ラムダ計算が扱う式を λ 式（*λ-term*）と呼び，次のように再帰的に定義する．

1. 変数 v，v', v''，... は λ 式である．
2. M が λ 式で x が変数のとき $(\lambda x.M)$ は λ 式である．
3. M と N が λ 式のとき $(M\ N)$ は λ 式である．

BNF で表すと次のとおり．

$$< \text{変数} > ::= v \mid\ < \text{変数} >'$$
$$< \lambda\text{式} > ::= < \text{変数} > \mid (< \lambda\ \text{式} > < \lambda\text{式} >) \mid (\lambda\ < \text{変数} > .\ < \lambda\text{式} >)$$

- 小文字 x，x_1，x_2，...，y，... は任意の変数

- 大文字 $M,\ M_1,\ M_2,\ ...,\ N,\ ...$ は任意の λ 式
- λ 式 M と N が構文上等しいとき，$M \equiv N$ と書く．
- λ 式 M に含まれる λ 式を，M の**部分式**（subterm）と呼ぶ．

3.2.1 略記法

1. $M_1\ M_2\ M_3\ ...\ M_n$ は $((\ ...\ ((M_1\ M_2)\ M_3)\ ...\ )\ M_n)$ の略記
2. $\lambda x_1 x_2\ ...\ x_n.M$ は $(\lambda x_1.(\lambda x_2.(\ ...\ (\lambda x_n.M)\ ...\ )))$ の略記

例： $\lambda xyz.xz(yz)$ は $(\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))$ の略記

3.2.2 束縛変数と自由変数

λ 式 M 中の変数 x が，$\lambda x. -\ \ x\ -$ のように出現するとき，“x は M 中で**束縛変数**（bound variable）として現れる” といい，そうでない x の出現があるとき，“x は M 中で**自由変数**（free variable）として現れるという．例： $\lambda x.y(\lambda y.xy)$
最初の y は自由変数．他の $x,\ y$ は束縛変数．

- 束縛変数だけが異なる λ 式は，意味的に同一視できる．

定義 3.2. λ 式 M 中の自由変数全体を $FV(M)$ と表す．厳密には，M の構造に関する帰納法で定義する．

$$\begin{aligned}
\mathrm{FV}(x) &= \{x\}, \\
\mathrm{FV}(M\ N) &= \mathrm{FV}(M) \cup \mathrm{FV}(N), \\
\mathrm{FV}(\lambda x.M) &= \mathrm{FV}(M) - \{x\}
\end{aligned}$$

- λ 式 M が自由変数を含まないとき，M を**閉じた λ 式**（closed λ-term）あるいは**コンビネータ**（combinator）と呼ぶ．

3.2.3 β 変換

λ 式 M 中の変数 x を λ 式 N によって置き換える操作を定義する．
例：$(\lambda xy.xy)(\lambda z.z)\ =\ \lambda y.(\lambda z.z)y$

- N が自由変数を含む場合に注意
  例：$(\lambda xy.xy)(\lambda z.zy)$
  $\Downarrow$ 束縛変数の名前付替え
  $(\lambda xy.xy)(\lambda z.zy)\ =\ \lambda v.(\lambda z.zy)v$

定義 3.3. λ 式 M 中の変数 x を λ 式 N に置き換えた λ 式 $M[x := N]$ を，次のように M の構造に関する帰納法で定義する．

1. $M \equiv z$ （変数）のとき，次の *2* つに場合分けして定義する．
  (*a*) $z \equiv x$ ならば $M[x := N] \equiv N$,
  (*b*) $z \not\equiv x$ ならば $M[x := N] \equiv z$.
2. $M \equiv M_1\ M_2$ のとき，$(M_1\ M_2)[x := N] \equiv M_1[x := N]\ M_2[x := N]$
3. $M \equiv \lambda y.M'$ のとき，次の *3* つに場合分けして定義する．
  (*a*) $x \equiv y$ ならば $M[x := N] \equiv \lambda y.M'$
  (*b*) $x \not\equiv y$ で，$y \notin \mathrm{FV}(N)$ か $x \notin \mathrm{FV}(M')$ ならば,
  $M[x := N] \equiv \lambda y.M'[x := N]$,
  (*c*) $x \not\equiv y,\ y \in \mathrm{FV}(N),\ x \in \mathrm{FV}(M')$ ならば,
  $M[x := N] \equiv \lambda y'.M'[y := y'][x := N]$,
  ここで，$y' \notin \mathrm{FV}(M') \cup \mathrm{FV}(N)$.
  例：

$$(\lambda y.xy)[x := \lambda z.z] \equiv \lambda y.(\lambda z.z)y,$$

$$(\lambda y.xy)[x := \lambda z.zy] \equiv \lambda y'.(\lambda z.zy)y'$$

- 複数置き換えへの拡張：$M[x_1, ..., x_n := N_1, ..., N_n]$ （$\neq M[x_1 := N_1]...[x_n := N_n]$）
  例： $M \equiv x_1 x_2$，$N_1 \equiv x_2 y$ とするとき，$M[x_1, x_2 := N_1, N_2] \equiv x_2 y N_2$
- 束縛変数名前の置換えを，α **変換**（α-conversion）と呼ぶ．$\lambda x.M \equiv \lambda y.M[x := y]$

3.2.4 λ 式間の等式

例 ：

- β 変換によって成り立つ等式： $(\lambda x.M)N \ = \ M[x := N]$
- 部分式を考慮した等式： $\lambda z.z((\lambda x.M)N) \ = \ \lambda z.z(M[x := N]))$

定義 3.4. λ 式間の等式 $M = N$ を導く形式体系 λ を次の公理と推論規則で定義する．

$$(\beta)(\lambda x.M)N = M[x := N]$$

$$(\xi)\frac{M = N}{\lambda x.M = \lambda x.N}$$

$$\frac{M = N}{ML = NL} \qquad \frac{M = N}{LM = LN}$$

$$M = M \qquad \frac{L = M \quad M = N}{L = N} \qquad \frac{M = N}{N = M}$$

- $M = N$ が上の公理と推論規則から導けるとき，$\lambda \vdash M = N$ あるいは略して，$M = N$ と書き，M と N は β 変換可能（β-*convertible*）であるという．

例 ： β 変換可能な λ 式：

$$(\lambda xy.x)(\lambda x.x)(\lambda x.x) = (\lambda yx.x)(\lambda x.x) = \lambda x.x$$
$$(\lambda xy.(\lambda x.xy)x)(\lambda x.x) = \lambda y.(\lambda z.zy)(\lambda x.x) = \lambda y.(\lambda x.x)y = \lambda y.y$$

3.3 様々な λ 式

λ 抽象，関数適用および β 変換だけをもつ単純な体系によって，様々な演算を表現できることを示す． $\Rightarrow$ 表現能力は?

3.3.1 数値の表現

定義 3.5. チャーチ数 (Church numeral)：
自然数（0 を含む）$n \in \mathbf{N} = \{0, 1, 2, ...\}$ について，λ 式 $\mathbf{c}_n$ を次のように定義する．

$$\mathbf{c}_n \equiv \lambda f x.f^n(x)$$

- $M^n(N) \equiv M(M(...(M\ N)...))$
- $\mathbf{c}_0 \equiv \lambda f x.x$

3.3.2 演算

命題 3.1. λ 式 $\mathbf{A}_+$，$\mathbf{A}_*$，$\mathbf{A}_{exp}$ を次のように定義する．

$$\mathbf{A}_+ \equiv \lambda xypq.xp(ypq)$$
$$\mathbf{A}_* \equiv \lambda xyz.x(yz)$$
$$\mathbf{A}_{exp} \equiv \lambda xy.yx$$

このとき，自然数 m と n について，次の β 変換が成り立つ．

1. $\mathbf{A}_+ \mathbf{c}_m \mathbf{c}_n \ = \ \mathbf{c}_{m+n}$

2. $\mathbf{A}_*\mathbf{c}_m\mathbf{c}_n \;=\; \mathbf{c}_{m*n}$

3. $\mathbf{A}_{\exp}\mathbf{c}_m\mathbf{c}_n \;=\; \mathbf{c}_{m^n}$（ただし，$n \geq 1$）

- 自然数をチャーチ数で表すと，加算，乗算，巾乗は，それぞれ $\mathbf{A}_+$，$\mathbf{A}_*$，$\mathbf{A}_{\exp}$ で表される．

証明：

1. 加算：

$$\begin{aligned}\mathbf{A}_+\mathbf{c}_m\mathbf{c}_n &= \lambda pq.\mathbf{c}_m p(\mathbf{c}_n pq)\\ &= \lambda pq.p^m(p^n(q))\\ &= \lambda pq.p^{m+n}(q)\\ &\equiv \mathbf{c}_{m+n}\end{aligned}$$

2. 乗算：

$$\begin{aligned}\mathbf{A}_*\mathbf{c}_m\mathbf{c}_n &= \lambda z.\mathbf{c}_m(\mathbf{c}_n z)\\ &= \lambda zx.(\mathbf{c}_n z)^m(x)\end{aligned}$$

なので，$(\mathbf{c}_n z)^m(x) = z^{m*n}(x)$ が証明されれば，$\mathbf{A}_*\mathbf{c}_m\mathbf{c}_n \;=\; \mathbf{c}_{m*n}$ が導かれる．これを m に関する帰納法で証明する．$m = 0$ のとき，明らか．$m = k$ のとき，$(\mathbf{c}_n z)^k(x) = z^{k*n}(x)$ が成り立つと仮定すると，

$$\begin{aligned}(\mathbf{c}_n z)^{k+1}(x) &= \mathbf{c}_n z((\mathbf{c}_n z)^k(x))\\ &= \mathbf{c}_n z(z^{k*n}(x)) &&\text{（帰納法の仮定）}\\ &= z^{n+k*n}(x)\\ &= z^{(k+1)*n}(x)\end{aligned}$$

したがって，$m = k+1$ のときも成り立つ．

3. 巾乗：

$$\begin{aligned}\mathbf{A}_{exp}\mathbf{c}_m\mathbf{c}_n &= \mathbf{c}_n\mathbf{c}_m\\ &= \lambda x.(\mathbf{c}_m)^n(x)\end{aligned}$$

なので，$(\mathbf{c}_m)^n(x) = \mathbf{c}_{m^n}(x)$ $(n \geq 1)$ が証明されれば，$\lambda x.\mathbf{c}_{(m^n)}x = \mathbf{c}_{m^n}$ から $\mathbf{A}_{\exp}\mathbf{c}_m\mathbf{c}_n \;=\; \mathbf{c}_{m^n}$ $(n \geq 1)$が導かれる．これを n に関する帰納法で証明する．$n = 1$ のとき，明らか．$n = k$ のとき，$(\mathbf{c}_m)^k(x) = \mathbf{c}_{m^k}(x)$ が成り立つと仮定すると，

$$\begin{aligned}(\mathbf{c}_m)^{k+1}(x) &= \mathbf{c}_m(\mathbf{c}_m^k(x))\\ &= \mathbf{c}_m(\mathbf{c}_{(m^k)}(x)) &&\text{（帰納法の仮定）}\\ &= \lambda y.(\mathbf{c}_{(m^k)}(x))^m(y)\\ &= \lambda y.x^{m^k*m}(y) &&\text{（2. の証明から）}\\ &= \mathbf{c}_{(m^{k+1})}(x)\end{aligned}$$

したがって，$n = k+1$ のときも成り立つ．

3.3.3 その他の演算

命題 3.2. λ 式 **true** と **false** を

$$\mathbf{true} \equiv \lambda xy.x, \qquad \mathbf{false} \equiv \lambda xy.y$$

と定義し，λ 式 L，M，N について，

$$\mathbf{if}\ L\ \mathbf{then}\ M\ \mathbf{else}\ N \equiv LMN$$

と表すと，

$$\begin{aligned}\mathbf{if\ true\ then}\ M\ \mathbf{else}\ N \;&=\; M\\ \mathbf{if\ false\ then}\ M\ \mathbf{else}\ N \;&=\; N\end{aligned}$$