コンピュータ・サイエンス入門
ラムダ計算第1回目資料

勝股 審也

2011年5月26日

1 ラムダ計算とは

ラムダ計算は1932-3年にAlonzo Churchによって発表された論理体系[1][2]の中に、関数を形式的に取り扱う表記法として登場した。Church自身はラムダ計算の考えを1928年頃には持っていたと言われている。後に表記法の部分が論理体系から取り出され、ラムダ計算として研究されるに至った。当初は関数を扱う理論が面白い研究対象になるとは認識されていなかったが、その後、数理論理学者 (Stephen Kleene, Haskell Curry) や計算機科学者 (Alan Turing) らの研究により、実際には豊かな研究対象であることが明らかとなった。ラムダ計算の発明は計算可能性の理論、プログラミング言語、そして論理学の発展に大きく貢献し、今もその価値は不動のものである。

この講義では純粋なラムダ計算を学ぶことにする。この資料ではラムダ項を定義し、ラムダ項の操作であるβ-簡約を紹介する。

参考文献

- 高橋 正子, 計算論 - 計算可能性とラムダ計算, コンピュータサイエンス大学講座 24, 近代科学社, 1991. ISBN: 4764901846
- H. P. Barendregt, The lambda calculus - Its Syntax and Semantics (2nd Edition). Volume 103 of Studies in logic and the foundations of mathematics, North Holland, 1984. ISBN: 444875085

授業の情報は `http://www.kurims.kyoto-u.ac.jp/~cs/lecturesj.html` に置いてある。

2 ラムダ項

まず可算無限個の**変数**と呼ばれるものの存在を仮定する。以降小文字のx, y, zが出てきたらそれらは変数を指しているものとする。

定義 2.1 以下の規則を考える。

1. 変数xはラムダ項である。
2. MとNがラムダ項ならば(MN)はラムダ項である。この形のラムダ項を**適用**と呼ぶ。

[1] Alonzo Church, A set of postulates for the foundation of logic 1. The Annals of Mathematics, 2nd Ser., Vol. 33, No. 2. (Apr. 1932), pp. 346-666

[2] Alonzo Church, A set of postulates for the foundation of logic 2. The Annals of Mathematics, 2nd Ser., Vol. 34, No. 4. (Apr. 1933), pp. 839-864

3. M がラムダ項ならば $(\lambda x.M)$ はラムダ項である。この形のラムダ項を**ラムダ抽象**と呼ぶ。

上の 3 つの規則のみを有限回使って「... はラムダ項である」と結論された文字列 "..." こそが**ラムダ項**である。

ラムダ項の集合を Λ で表すことにする。以降大文字 M, N, L が出てきたら、それらはあるラムダ項を指しているものとする。

記法 2.2 1. $M_1 \cdots M_n$ は $(\cdots((M_1\ M_2)\ M_3)\cdots M_n)$ のこととする。

2. $\lambda x_1 \cdots x_n.M$ は $(\lambda x_1.(\lambda x_2.\cdots(\lambda x_n.M)\cdots))$ のこととする。

3. $\lambda x.M_1 \cdots M_n$ は $\lambda x.(M_1 \cdots M_n)$ のこととする。

4. 取り除いてもあいまいさが失われない括弧は取り除く。

ラムダ項は今の段階では規則に従って作られた有限の長さの文字列に過ぎない。しかし、それらは数学的な操作や概念を表すために設計されている。

- ラムダ項 $(\lambda x.M)$ は「x **というラムダ項を** M **に写す関数**」を表している。例えば $(\lambda x.x)$ というラムダ項は「ラムダ項 x を x に写す関数」、つまりラムダ項上の恒等関数を表している。
- ラムダ項 $(M\ N)$ は「M **というラムダ項が表す関数に** N **を渡している式**」を表している。ここで、適用とは関数に具体的な値を渡すことである。例えば $(\lambda x.x)\ y$ というラムダ項は「恒等関数に y を渡している式」を表している。

我々は、恒等関数を y に適用した結果が y であることを知っているが、現段階ではラムダ項 $(\lambda x.x)\ y$ と y の間に関係は確立されていない。

3 自由変数と束縛変数

束縛変数の概念は数学、論理学、そしてプログラミング言語において普遍的に現われる構文上の概念である。以下の式を見てみよう。

$$\exists x.(x = x^2) \qquad \int sin(x)dx \qquad f(x) = x^2 + 1 \qquad \{\texttt{int x;}\cdots\texttt{x = x + 1;}\cdots\}$$

(最後は C あるいは Java のプログラムの一部だと思ってほしい) これらの式に共通なのは、変数 x を仮の値として導入して作られた命題や数式、プログラムとなっている点である。

前の節で定義した $\lambda x.M$ という形のラムダ項も上述の表記法と同様に、x を仮の値として導入して作られたラムダ項である。この形のラムダ項において、M の中の x は**束縛される**と言う。

定義 3.1 変数 x がラムダ項の中で λ の直後以外の位置に現われているとする[3]:

$$\cdots x \cdots$$

この位置の x が**束縛されている**とは、x が $(\lambda x.\cdots)$ によって囲まれている時、つまり

$$\cdots(\lambda x.\cdots x\cdots)\cdots$$

[3] $(\lambda x.M)$ という形をした項の λ の直後の変数については束縛あるいは自由ということを定義しない。

となっている時である。

ラムダ項 M の中のある位置の変数が**自由である**とは、それが束縛されていない時である。ラムダ項 M の中の自由な変数の集合を $FV(M)$ で現わすことにする。

自由変数を持たないラムダ項 M(つまり $FV(M) = \emptyset$ であるようなラムダ項) を**閉じている**と言う。閉じたラムダ項の集合を Λ_0 で書くことにする。

問題 3.2　1. 次のラムダ項にて、下線の位置の変数は束縛されているか？

$$\lambda xy.\underline{x}, \qquad \lambda x.\underline{y}\ x, \qquad (\lambda x.\underline{x})\ \underline{x}$$

2. 次の集合を計算せよ。

$$FV(x), \qquad FV(\lambda y.x), \qquad FV(\lambda xy.x), \qquad FV((\lambda xy.x)\ z)$$

問題 3.3　M, N をラムダ項とする。

1. $FV(M)$ と $FV(N)$ を用いて $FV(MN)$ を与えよ。
2. $FV(M)$ を用いて $FV(\lambda x.M)$ を与えよ。

4　α 同値

この節では α 同値の概念を導入する。今、$\lambda x\,.\,x$ と $\lambda y\,.\,y$ という二つの (異なる字面の) ラムダ項を考えよう。どちらもラムダ項 M を M へ写す関数 (恒等関数) を表現しているため、これらのラムダ項を同一視するのは自然な事である。似た例をさらに見てみよう。

$$\begin{array}{rl} \lambda x\,.\,xy & \lambda z\,.\,zy \\ (\lambda x\,.\,xy)y & (\lambda z\,.\,zy)y \\ \lambda y\,.\,(\lambda x\,.\,xy)y & \lambda v\,.\,(\lambda z\,.\,zv)v \end{array}$$

各行の左と右にあるラムダ項は字面は異なるものの、同一の式を表現しているものと考えられる。まず、最初の行のラムダ項はどちらもラムダ項 M を My というラムダ項に写す関数を表現している。二行目のラムダ項はどちらも一行目のラムダ項が表現している関数を y に適用したものを表現している。そして三行目の左のラムダ項はラムダ項 N を $(\lambda x\,.\,x\,N)\,N$ というラムダ項に写す関数を表現している。このラムダ項は $(\lambda z\,.\,z\,N)\,N$ と同一視できるため、三行目のラムダ項はどちらも同じ関数を表現していると考えられる。

このようなラムダ項の同一視を定式化したのが α **同値** (関係) である。例から見て推測できるように、二つのラムダ項が α 同値であるのは一方のラムダ項の中のラムダ抽象により束縛されている変数を別の変数につけかえた (別の変数を代入した) のが他方になっている時のように見える。この考え方はおおざっぱには良いが、厳密にはつけかえる変数の選び方に条件がある。α 同値の正確な定義を以下で与える。まずラムダ項中の自由変数に変数を代入する操作を定義する。

定義 4.1 ラムダ項 M の自由変数 x に変数 y を代入したラムダ項 $M[y/x]$ を M に関して再帰的に定義する。

$$\begin{aligned} x[y/x] &= y \\ z[y/x] &= z \qquad (z \neq x) \\ (MN)[y/x] &= (M[y/x]N[y/x]) \\ (\lambda x\,.\,M)[y/x] &= (\lambda x\,.\,M) \\ (\lambda z\,.\,M)[y/x] &= (\lambda z\,.\,M[y/x]) \qquad (z \neq x) \end{aligned}$$

続いて、あるラムダ項の中のラムダ抽象 $(\lambda x\,.\,M)$ の束縛変数をつけかえる操作を定義する。

定義 4.2 ラムダ項 L 中のある位置 p にラムダ抽象が現われていたとする。

$$L = \cdots \underbrace{(\lambda x\,.\,M)}_{p} \cdots$$

この時、M に (自由、束縛問わず) 現われない変数 y を取り、位置 p のラムダ抽象を $\lambda y\,.\,M[y/x]$ で置き換えたラムダ項 L' を考える。

$$L' = \cdots \underbrace{(\lambda y\,.\,M[y/x])}_{p} \cdots$$

この時 $L \to_\alpha L'$ と書き、「L **を** *1* **ステップ** α **変換して** L' **を得た**」と言うことにする。そして L に 0 回以上ワンテップ α 変換を施して L' を得た時、「L **と** L' **は** α **同値である**」と言い、$L =_\alpha L'$ と書くことにする。

以下はワンステップ α 変換の例である:

$$\begin{aligned} &\lambda x\,.\,xy \to_\alpha \lambda z\,.\,zy \\ &(\lambda x\,.\,xy)y \to_\alpha (\lambda z\,.\,zy)y \\ &\lambda y\,.\,(\lambda x\,.\,xy)y \to_\alpha \lambda y\,.\,(\lambda z\,.\,zy)y \to_\alpha \lambda v\,.\,(\lambda z\,.\,zv)v \end{aligned}$$

よって $\to_\alpha$ でつながれた項は互いに α 同値となる。

α 変換の定義 (定義 4.2) における変数の取り方の条件について説明する。今 $L = \lambda x\,.\,xv$ というラムダ項を考えよう。定義 4.2 の条件より、つけかえに用いることのできる変数は x, v 以外の変数である。この条件を無視した場合、例えば v を用いて α 変換を行なうと、

$$L' = \lambda v\,.\,((xv)[v/x]) = \lambda v\,.\,vv$$

という、L とは異なる関数を表現するラムダ項を得てしまい、不都合が生じる。よってつけかえには M に現われないの変数を取るという条件がついているのである。

以降、α 同値なラムダ項同士を同一視する。つまり $M =_\alpha M'$ ならば M と M' は同じラムダ項であると考えることにする (このことは形式的にはラムダ項を α 同値類として扱うことと同等である)。

α 同値の概念はラムダ計算に限らず、変数の束縛の概念がある言語全般に対して考えることができる。例えば以下に並べたの左右の式は互いに α 同値であると考えられる。

$\exists x\,.\,x = x$	$\exists y\,.\,y = y$
$\int \sin x dx$	$\int \sin y dy$
$f(x) = x^2 + 1$	$f(y) = y^2 + 1$
$\{\texttt{int x;} \cdots \texttt{x = x + 1;} \cdots\}$	$\{\texttt{int y;} \cdots \texttt{y = y + 1;} \cdots\}$

5 代入

次にラムダ項 M の自由変数 x にラムダ項 N を代入する操作を定義する。代入は単に自由変数 x を N に置き換える事ではない事を説明する。例えばラムダ項

$$(\lambda y\,.\,y\;x)$$

の中の自由な x に $y\;y$ を代入する事を考える。このラムダ項は「ラムダ項 M を $M\;x$ に写す関数」を表現している。よって、このラムダ項の x に $y\;y$ を代入して得られるラムダ項は「ラムダ項 M を $M\;(y\;y)$ に写す関数」を表現するもの、例えば $\lambda z\,.\,z\;(y\;y)$、になると考えるのが自然である。ところが、もしこのラムダ項の自由な x を単純に $y\;y$ で置き換えると

$$(\lambda y\,.\,y\;(y\;y))$$

というラムダ項を得る。これは「ラムダ項 M を $M\;(M\;M)$ に写す関数」を表現しているが、期待するものとは異っている。この問題は代入されるラムダ項の自由変数が置き換えにより束縛されてしまう事により発生している。上の例だと x の位置に置かれるラムダ項 $y\;y$ は自由な y を含んでおり、これが外側の λy によって束縛されてしまっているため、問題が生じたのである。

この問題を避けるため、$(\lambda y\,.\,M)$ 中の自由変数 x にラムダ項 N を代入する際、$y \in FV(N)$ ならば一度 $y' \notin FV(N)$ であるような変数で $(\lambda y\,.\,M)$ に α 変換を施してラムダ項 $(\lambda y'\,.\,M[y'/y])$ を得た後、その中の自由変数 x に N を代入する。この事を形式化したのが次の定義である。

定義 5.1 ラムダ項 M の自由変数 x にラムダ項 N を代入した項 $M[x := N]$ を M に関して再帰的に定義する。

$$\begin{aligned}
x[x := N] &= N \\
y[x := N] &= y \qquad (x \neq y) \\
(ML)[x := N] &= (M[x := N]L[x := N]) \\
(\lambda x\,.\,M)[x := N] &= (\lambda x\,.\,M) \\
(\lambda y\,.\,M)[x := N] &= (\lambda z\,.\,M[z/y][x := N]) \qquad (z \notin FV(N))
\end{aligned}$$

代入は最後の行の z の取り方に依存せずに一意な α 同値類 $M[x := N]$ を定める。また、$M =_\alpha M'$ かつ $N =_\alpha N'$ ならば $M[x := N] =_\alpha M'[x := N']$ である。

問題 5.2 $M[x := N]$ を計算せよ。

	$N = \lambda y.x$	$N = \lambda x.y\;x$
$M = x\;y$		
$M = \lambda y.x$		
$M = (\lambda y.x)\;x$		

補題 5.3 (代入補題) 任意のラムダ項 M, L, N に関して $x \notin FV(L)$ の時

$$M[x := N][y := L] = M[y := L][x := N[y := L]]$$

が成立する。

6 β 簡約

定義 6.1 $(\lambda x.M)N$ という形のラムダ項を β-*redex* と言う。

β-redex は「$x \mapsto M$ という関数に N を渡している」ことを表現しているラムダ項である。
あるラムダ項 L の中に β-redex が現われているとしよう。

$$L = \cdots (\lambda x.M)N \cdots$$

この位置の β-redex を

$$M[x := N]$$

というラムダ項におきかえることにより、我々は

$$L' = \cdots M[x := N] \cdots$$

というラムダ項を得る。L' は L よりも一段計算が進んでおり、より詳しい意味を持つと考えることができるので、この事を

$$L \to_\beta L'$$

と書き、「L **を** *1* **ステップ** β **簡約して** L' **を得た**」と言うことにする。1 ステップ β 簡約を正確に定義すると以下のようになる。

定義 6.2 以下の規則を考える。

- $(\lambda x.M)N \to_\beta M[x := N]$
- $M \to_\beta M'$ ならば $MN \to_\beta M'N$
- $M \to_\beta M'$ ならば $NM \to_\beta NM'$
- $M \to_\beta M'$ ならば $\lambda x.M \to_\beta \lambda x.M'$

以上の規則のみを有限回用いて $M \to_\beta N$ が導かれた時、「M **を** *1* **ステップ** β **簡約して** N **を得た**」と言い、$M \to_\beta N$ と書くことにする。

定義 6.3 ラムダ項 M に 1 ステップ β 簡約を *0* **回以上**施してラムダ項 N を得た時、「M **を** β **簡約して** N **を得た**」と言い、$M \to_\beta^* N$ と書くことにする。

これから β 簡約の例を見るが、その前に便利なラムダ項に名前をつける。

定義 6.4 以下のラムダ項を定義する (β-正規形はどれか?)。

$$\mathbf{I} = \lambda x.x \qquad \mathbf{S} = \lambda xyz.x\ z\ (y\ z) \qquad \mathbf{K} = \lambda xy.x$$

$$\omega = \lambda x.x\ x \qquad \mathbf{\Omega} = \omega\ \omega$$

$$\mathbf{Y} = \lambda f.(\lambda x.f\ (x\ x))(\lambda x.f\ (x\ x)) \qquad \mathbf{\Theta} = (\lambda xf.f\ (x\ x\ f))(\lambda xf.f\ (x\ x\ f))$$

例 6.5 ラムダ項の β 簡約の例を見る。

1. 一般に、任意のラムダ項 M に対して以下が成り立つ。

$$\underline{\mathbf{I}\ M} \to_\beta M$$

下線はその部分が β-redex であり、続く 1 ステップ β 簡約で代入に置き換わることを示している。$M \to_\beta M'$ ならばもちろん以下も成立する。

$$\mathbf{I}\ M \to_\beta \mathbf{I}\ M'$$

2. **K** は 2 つラムダ項を取り、2 つ目を捨て、1 つ目を返す。

$$\underline{\mathbf{K}\ M}\ N \to_\beta \underline{(\lambda y.M)\ N} \to_\beta M$$

3. **S** は 3 つラムダ項を取り、3 つ目を 1 つ目と 2 つ目に分配する。

$$\underline{\mathbf{S}\ M}\ N\ L \to_\beta \underline{(\lambda yz.M\ z\ (y\ z))\ N}\ L \to_\beta \underline{(\lambda z.M\ z\ (N\ z))\ L} \to_\beta M\ L\ (N\ L)$$

4. 異なる β-redex の 1 ステップ β 簡約が同じラムダ項を作る例。

$$\mathbf{I}\ \underline{(\mathbf{I}\ \mathbf{I})} \to_\beta \mathbf{I}\ \mathbf{I}, \qquad \underline{\mathbf{I}\ (\mathbf{I}\ \mathbf{I})} \to_\beta \mathbf{I}\ \mathbf{I}$$

5. 無限に続く β 簡約の例。

$$\mathbf{\Omega} \to_\beta \mathbf{\Omega} \to_\beta \mathbf{\Omega} \to_\beta \cdots$$

6. $\mathbf{S}\ \mathbf{K}\ \mathbf{K} \to_\beta^* \mathbf{I}$ が成り立つ。

$$\underline{\mathbf{S}\ \mathbf{K}}\ \mathbf{K} \to_\beta (\lambda yz.\underline{\mathbf{K}\ z}\ (y\ z))\ \mathbf{K} \to_\beta (\lambda yz.\underline{(\lambda y.z)\ (y\ z)})\ \mathbf{K} \to_\beta \underline{(\lambda yz.z)\ \mathbf{K}} \to_\beta \lambda z.z = \mathbf{I}$$

7. $\mathbf{Y}\ \mathbf{I} \to_\beta^* \mathbf{\Omega}$ が成り立つ。

$$\begin{aligned}
\mathbf{Y}\ \mathbf{I} &= \underline{(\lambda f.(\lambda x.f\ (x\ x))(\lambda x.f\ (x\ x)))\ \mathbf{I}} \\
&\to_\beta (\lambda x.\underline{\mathbf{I}\ (x\ x)})(\lambda x.\mathbf{I}\ (x\ x))) \\
&\to_\beta \omega\ (\lambda x.\underline{\mathbf{I}\ (x\ x)}) \\
&\to_\beta \omega\ \omega = \mathbf{\Omega}
\end{aligned}$$

問題 6.6 1. $\mathbf{\Theta}\ f \to_\beta^* f\ (\mathbf{\Theta}\ f)$ を示せ。

2. $\mathbf{X} = \lambda x.x\ \mathbf{K}\ \mathbf{S}\ \mathbf{K}$ とせよ。次を示せ。

$$\mathbf{X}\ \mathbf{X}\ \mathbf{X} \to_\beta^* \mathbf{K}, \qquad \mathbf{X}\ (\mathbf{X}\ \mathbf{X}) \to_\beta^* \mathbf{S}$$

ラムダ項の中には 1 ステップ β 簡約のしようが無いものもある。

定義 6.7 β-redex を含まないラムダ項を β-**正規形**と呼ぶ。言い換えると、$M \to_\beta N$ となるようなラムダ項 N が存在しないラムダ項 M を β-正規形と呼ぶ。ラムダ項 M が β-**正規形を持つ**とは、ある β-正規形 N が存在して $M \to_\beta^* N$ となるときである。

β-正規形は計算の進めようがないラムダ項、つまり最も簡単化されたラムダ項と考えられる。

一般にラムダ項は β-redex を複数含む場合がある。従って β 簡約をする際、どの β-redex を代入で置き換えるかにより異るラムダ項を得る。ここで心配なのは、ラムダ項 M を 2 通りの方法で β 簡約して正規形にたどりついた時:

$$M \to_\beta^* N_1 \qquad M \to_\beta^* N_2$$

N_1 と N_2 が異る正規形になってしまわないかということである。もしそうなったなら、直感的には一つの数式を異なる順序で変形したら異なる値が出てきてしまうことを意味するのでおかしい。

しかしながら、β 簡約に関して知られている代表的な結果に以下の Church-Rosser の定理があり、これにより上にあげた心配は無用であることが示せる。

定理 6.8 (Church-Rosser) 任意のラムダ項に対して $M \to_\beta^* N_1$ かつ $M \to_\beta^* N_2$ ならばある L が存在して $N_1 \to_\beta^* L$ かつ $N_2 \to_\beta^* L$ である。

問題 6.9 実際に Church-Rosser の定理から上で述べた心配は無用であることを示せ。つまり、ラムダ項 M と β-正規形 N_1, N_2 に対し、

$$M \to_\beta^* N_1 \qquad M \to_\beta^* N_2$$

ならば $N_1 = N_2$ であることを示せ。

このことはラムダ項 M は β-正規形を高々1 つしか持たないことを意味する (β-正規形を持たないラムダ項もある; 例えば $\mathbf{\Omega}$)。

問題 6.10 「$M \to_\beta M \to_\beta \cdots$ という 1 ステップ β 簡約が可能なラムダ項 M は β 正規形を持たない。」これは本当か？