ラムダ計算の話

宇佐見公輔

第3回　関西日曜数学　友の会

自己紹介

- 宇佐見公輔（@usamik26）
- 大学時代は数学専攻（代数、応用数理）
- 現在はプログラマ（フェンリル株式会社、iOS アプリ開発）

ラムダ計算（λ-calculus）とは

- 計算機科学で出てくる体系のひとつ
- プログラミング言語の型システムとの関連
- 関数型プログラミングを支える理論

ラムダ記法

- 数学で使われる関数の表記法

$$f(x) = x^2 - 1$$

$$x \mapsto x^2 - 1$$

- ラムダ記法による関数の表記法

$$\lambda x.\, x^2 - 1$$

ラムダ記号の意味合い

- ラムダ記法におけるλは、$\exists$や$\forall$のような意味合いの記号
- そう思って並べてみるとなんとなく似ている気がする

$$\lambda x.\, x^2 - 1$$

$$\exists x.\, x^2 - 1 = 0$$

$$\forall x.\, x^2 - 1 = (x + 1)(x - 1)$$

ラムダ記法：関数の適用

- ラムダ記法で関数に値を代入する

$$(\lambda x.\, x^2 - 1)3 = 3^2 - 1 = 8$$

- 記法の比較：関数 $f(x) = x^2 - 1$ に値を代入する

$$f(3) = 3^2 - 1 = 8$$

ラムダ記法：多変数関数

- 多変数関数のラムダ記法

$$\lambda xy.\, x + y$$

- これは1変数関数の組み合わせと同一視できる（カリー化）

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

ラムダ記法：高階関数

- x に関数 f を2回適用する関数の例

$$\lambda f x.\, f(f(x))$$

- カリー化して書けば

$$\lambda f.\, \lambda x.\, f(f(x))$$

ラムダ式（λ-term）の導入

- ラムダ記法そのものを、より抽象的に取り扱う
- そのために、記号列としてのラムダ式を導入する

ラムダ式の定義

- 可算無限濃度の集合 V が与えられているとする
- 以下で再帰的に定義される記号列の集合 Λ の要素をラムダ式と呼ぶ（なお、括弧は一定のルールで取り外すことができる）
- (1) $x \in V$ のとき $x \in \Lambda$
- (2) $x \in V, M \in \Lambda$ のとき $(\lambda x.\, M) \in \Lambda$
- (3) $M, N \in \Lambda$ のとき $(MN) \in \Lambda$

ラムダ式とラムダ記法

- V は変数の集合を意味する
- (1) $x \in \Lambda$ は、変数はラムダ式であることを意味する
- (2) $(\lambda x.\, M) \in \Lambda$ は、関数のラムダ記法に対応する
- (3) $(MN) \in \Lambda$ は、関数適用のラムダ記法に対応する

ラムダ計算

- こうして定義されたラムダ式の性質を見ることで、ラムダ記法で記述された関数のなす世界を研究するのが、ラムダ計算の理論である
- 特に、個々の関数（$f(x) = x^2 - 1$ とか $f(x, y) = x + y$ とか）の性質によらず、関数の世界に内在する性質を考察する

ラムダ式についての補足

- 記号列として定義されたラムダ式が、常にラムダ記法に対応して意味を持つとは限らない（ラムダ記法としてありえない式ができてしまう）
- しかし、まずはそれを許容して議論する方が見通しが良くなる
- その後、型（関数の定義域や値域に相当する概念）を導入することで、ラムダ記法との対応の議論ができる

ベータ変換

- 以下で再帰的に定義されるラムダ式の変換をベータ変換と呼ぶ
- （1） $(\lambda x.\, M)N \to_\beta M[x := N]$
- （2） $M \to_\beta N$ ならば $\lambda x.\, M \to_\beta \lambda x.\, N$ および $PM \to_\beta PN$
- ここで $M[x := N]$ は、ラムダ式 M の中の変数 x をラムダ式 N に置き換えたラムダ式のことである（代入）

代入についての補足

- ラムダ式 M の中の変数 x について、$(\lambda x.\cdots)$ という部分式があるとき x を束縛変数、そうでないとき自由変数と呼ぶ
- 代入 $M[x := N]$ を考えるとき、N の自由変数は M の束縛変数ではないものとする
- また N の束縛変数は x 以外の変数に置き換えて考える（一般に、ラムダ式の束縛変数をその式に現れない別の変数に置き換えた式を、元の式と同一視する）

ベータ変換の例

各ラムダ式の自由変数は他の式の束縛変数ではないものとする

$$(\lambda x.\, x)M \to_\beta M$$

$$(\lambda xy.\, x)MN \to_\beta (\lambda y.\, M)N \to_\beta M$$

$$(\lambda xyz.\, xz(yz))PQR \to_\beta (\lambda yz.\, Pz(yz))QR \to_\beta (\lambda z.\, Pz(Qz))R \to_\beta PR(QR)$$

Church-Rosser の定理

- ベータ変換のしかたは一通りではない

$$(\lambda x.\, xx)((\lambda y.\, y)z) \to_\beta (\lambda x.\, xx)(z)$$

$$(\lambda x.\, xx)((\lambda y.\, y)z) \to_\beta ((\lambda y.\, y)z)((\lambda y.\, y)z)$$

- Church-Rosser の定理：ひとつのラムダ式からベータ変換で得られたふたつのラムダ式は、何度かベータ変換を行うことで、同じラムダ式にできる

正規形

- Church-Rosser の定理により、ラムダ式 M から始めて有限のステップで止まる（式中に λ がなくなる）ベータ変換列があるとき、最終結果は一致する
- この最終結果を M の正規形と呼ぶ
- 与えられた関数（ラムダ記法）を機械的に処理して正規形を得るのが、関数型プログラミング言語における計算である（計算機科学の用語では評価と呼ぶ）

さらなる話題・・・

- ここまでのラムダ計算は、型なしラムダ計算と呼ばれる
- 型を導入した、型つきラムダ計算という体系がある
- 型つきラムダ計算は、ラムダ記法との対応ができる
- 型つきラムダ計算は、型を対象、ラムダ式を射として、圏をなす
- これにより、圏論の議論が応用できる