束縛と自由変数の基本

1 導入

この講義の核心は、「変数の名前は本質でなく、どこに束縛されているかが本質である」という点にある。$\lambda x.x$ と $\lambda y.y$ は名前が違うが「同じ関数」である。この見方をできるようにするために、束縛・自由変数・スコープの概念を正確に定義する。

2 中心課題

$\lambda x.(\lambda y.x\ y)$ において、最外側の x と内側の y はどう違うか。また $\lambda x.(x\ y)$ の y はなぜ「外から来る何か」なのか。

3 用語

- λ 項: 変数・λ 抽象 ・適用のいずれかから再帰的に構成される式
- 束縛変数: λ 抽象 のパラメータとして宣言され、そのスコープ内で参照される変数
  bound variable
- 自由変数: 束縛されていない出現を持つ変数
  free variable
- スコープ: λ 抽象 $\lambda x.M$ において x が束縛する領域 M
  scope

4 方針

まず λ 項の構文 (BNF) を確認する。つぎに自由変数の集合 $\mathrm{FV}(t)$ を構造帰納法で定義する。最後に、束縛の重なり（ネスト）がスコープに何をもたらすかを見る。

5 直感的な説明

プログラミングの経験から「関数の引数名はその関数の中だけで有効だ」という感覚はあるはずである。

$\lambda x.M$ の x はちょうどそれで、M の外では x という名前は意味を持たない。

逆に $\lambda x.(x\ y)$ の y は「この関数が呼ばれる文脈から来る何か」であり、定義の中にスコープが存在しない。これが自由変数である。

Display

λ x. (λ y. x y)

↑ ↑ ↑

y は内側の λ y で束縛

x は外側の λx で束縛
外側のスコープは λy. x y 全体である

6 厳密な説明

6.1 1. λ 項の構文

$$t ::= x \mid \lambda x.t \mid t_1\ t_2$$

- x: 変数（Variable）
- $\lambda x.t$: λ 抽象 （Abstraction）— 「x を受け取って t を返す関数」
- $t_1\ t_2$: 適用（Application）— 「t_1 に t_2 を渡す」

6.2 2. 自由変数集合 $\mathrm{FV}(t)$

$$\mathrm{FV}(x) = \{x\}$$

$$\mathrm{FV}(\lambda x.t) = \mathrm{FV}(t) \setminus \{x\}$$

$$\mathrm{FV}(t_1\ t_2) = \mathrm{FV}(t_1) \cup \mathrm{FV}(t_2)$$

λ 抽象 の場合、x が束縛されるので $\mathrm{FV}(t)$ から除かれる。

6.3 3. 束縛変数集合 $\mathrm{BV}(t)$

$$\mathrm{BV}(x) = \emptyset$$

$$\mathrm{BV}(\lambda x.t) = \{x\} \cup \mathrm{BV}(t)$$

$$\mathrm{BV}(t_1\ t_2) = \mathrm{BV}(t_1) \cup \mathrm{BV}(t_2)$$

6.4 4. 閉項

$\mathrm{FV}(t) = \emptyset$ である項を閉項（closed term）（またはコンビネータ（combinator））という。閉項は外部の文脈に依存しない。

7 最小の具体例

7.1 例 1

$t = \lambda x.(x\ y)$

$\mathrm{FV}(t) = \mathrm{FV}(x\ y) \setminus \{x\} = (\{x\} \cup \{y\}) \setminus \{x\} = \{y\}$

y が自由変数である。

7.2 例 2

$t = \lambda x.\lambda y.(x\ y)$

$\mathrm{FV}(t) = \mathrm{FV}(\lambda y.(x\ y)) \setminus \{x\} = (\{x, y\} \setminus \{y\}) \setminus \{x\} = \emptyset$
閉項である。

7.3 例 3: スコープの重なり

$t = (\lambda x.x)\ (\lambda x.x\ x)$
左側と右側で別々に x が束縛されており、2 つの x は別物である。どちらも外から見えない。$\mathrm{FV}(t) = \emptyset$。

8 別の見方

λ 抽象 を「新 しいスコープを開くブロック」とみると、スコープの入れ子はファイルシステムの階層に似ている。変数名の解決は「最 も内側のスコープから外側へ探す」ルールで 行 われる。自由変数はどのスコープにも見つからなかった変数名である。

9 見分け方

- 変数 x のある出現が束縛か自由かは、「そこから外側へ向かって λx が 現 れるか」を確認する
- $\mathrm{FV}(t)$ を計算するには、BNF の場合分けに 従 って内側から再帰的に求める
- $\mathrm{FV}(t) = \emptyset$ なら閉項であり、単独で評価できる

10 一言でいうと

λ 項における変数の意味は名前でなくスコープで決まる。自由変数とは「この項の外から 供給 される 値 のプレースホルダ」であり、束縛変数とは「この項の内部で宣言・参照されるもの」である。

11 関連リンク

→ 講義 **置換と α 同値の基本** lecture information programming-languages
https://study.bem130.com/lecture/information/programming-languages/foundation/置換と α 同値の基本-講義/

→ 講義 **ラムダ計算の基本** lecture information programming-languages
https://study.bem130.com/lecture/information/programming-languages/lambda-calculus/ラムダ計算の基本-講義/