関数型プログラミングの入口

1 導入

この講義の核心は、「関数型プログラミングとはスタイルの好みではなく、計算を式の簡約として見る計算モデルである」という点にある。状態の書き換えを中心に据えた命令型モデルと、式の値への簡約を中心に据えた関数型モデルの違いを見ることで、型が重要になる理由の下地ができる。

2 中心課題

なぜ変数への代入（状態更新）をなくすと、プログラムの性質を述べやすくなるのか。

3 用語

- 副作用: 戻り値以外の観察可能な変化（変数の書き換え、I/O、例外など）
- 参照透過性: 同じ式はどこに現れても同じ値に評価される性質
- 第一級関数: 関数を値として変数に束縛したり、別の関数に渡したりできる性質

4 方針

まず命令型の計算モデルを「状態の列」として見る。つぎに関数型の計算モデルを「式の簡約」として見る。参照透過性が成り立つと何がうれしいかを確認し、型が計算の「形の宣言」として機能する下地を作る。

5 直感的な説明

命令型プログラムでは、`x = x + 1` という行は「x という箱の中身を書き換える」命令である。数学の等号とは違う意味を持っている。

関数型では、変数は「ある値への名前」であり、後から書き換えない。`f(x) = x + 1` という定義は数学的な関数と同じ意味を持つ。したがって `f(3)` は「3 + 1 を計算する」というだけで、副作用が発生しない。

Display

命令型: x := 0; x := x + 1; print x

関数型: let f x = x + 1 in f 0

命令型では「手順を追って状態が変わる」のに対し、関数型では「式が値へと変形される」。

6 厳密な説明

6.1 1. 状態と副作用

命令型の計算は、状態（変数と値の対応）を引数にとり新しい状態を返す関数としてモデル化できる。副作用はこの状態の変化である。副作用があると、「同じ関数を同じ引数で呼んでも結果が異なる」ことが起きうる。

6.2 2. 参照透過性

参照透過性が成り立つとき、e という式を v（e の値）で置き換えてもプログラムの意味が変わらない（等式推論 (equational reasoning) が成立する）。これにより、プログラムの部品を独立して理解・検証できる。

6.3 3. 第一級関数と高階関数

関数を値として扱えると、「関数を引数にとる関数」（高階関数 (higher-order function)）が自然に書ける。この発想の極限が λ 計算であり、すべての計算を「変数への束縛」と「関数適用」だけで表す。

6.4 4. 型の動機

参照透過性と第一級関数を組み合わせると、「この式が整数に評価されることは保証されているか」という問いが自然に出てくる。型はこの問いに答える仕組として導入される。

7 最小の具体例

7.1 例 1: 参照透過性の崩れ

```
// 命令型（副作用あり）
var counter = 0;
function inc() { counter += 1; return counter; }
inc() // => 1
inc() // => 2  同じ呼び出しで異なる値
```

```
// 関数型（副作用なし）
let inc n = n + 1
inc 0  (* => 1 *)
inc 0  (* => 1 *)
```

7.2 例 2: 高階関数

```
let apply f x = f x

let double n = n * 2

apply double 3  (* => 6 *)
```

`apply` は関数を引数にとっている。`double` が値として渡されている。

8 別の見方

命令型プログラミングを「時間と空間の中で状態が変化する機械のモデル」とすると、関数型プログラミングは「式の変形規則の集まり」である。前者はコンピュータの物理的動作に近く、後者は数学の証明操作に近い。

9 見分け方

- 「同じ式を 2 回評価して結果が変わりうるか」を考えると、副作用の有無が見える
- 「この関数の戻り値は入力だけで決まるか」が参照透過性の確認問いである
- 関数を変数に入れたり引数に渡したりしているなら、それは第一級関数の使用である

10 一言でいうと

関数型プログラミングとは、計算を「状態の書き換え」ではなく「式の値への簡約」として扱うことで、プログラムの部品を数学的に推論できるようにする計算モデルである。

11 関連リンク

→ 講義 **関数型プログラミングと型理論ポータル** lecture information programming-languages
https://study.bem130.com/lecture/information/programming-languages/foundation/関数型プログラミングと型理論ポータル-講義/

→ 講義 **束縛と自由変数の基本** lecture information programming-languages
https://study.bem130.com/lecture/information/programming-languages/foundation/束縛と自由変数の基本-講義/