継続の効果、あるいは、継続と効果

yuki @lmdexpr

2025-10-17

自己紹介

- Yuki Tajiri (@lmdexpr)
- 2025-02 より M3 基盤チームでソフトウェアエンジニア
- Software Design で連載中の「つまみぐい関数型プログラミング」今月が最終回です

前回のあらすじ

継続は力なり

今回のあらすじ

- 前半: 継続がもたらすもの、継続との戦い
- 後半: 継続を考える最新機能

前半

皆さんは

Function Coloring 問題

をご存知でしょうか。

What color is your function?[1]

というブログポストにて提唱された問題です。

ブログポスト内の「色付き関数」ルール

1. 全ての関数は（赤か青の）色を持ちます
2. 色ごとに関数の呼び出し方法が異なります
3. 赤色の関数は赤色の関数からしか呼び出せません
4. 赤色の関数を呼び出すことはより面倒です
5. いくつかのコアライブラリは赤色です

このルールはあくまで例えですが、
特定の機能を想定したものです。
何かわかりますか？

正解は、

「赤の関数」は「非同期関数」です。

Q. どうしてこんなことを？

A. 非同期関数は結果を待ち合わせるために継続を考える必要があるから

- コールバック
- promise, future
- async, await
- ジェネレーター

-> どれも、継続と向き合う必要がある

実は、あなたも既に継続と戦っていた！

余談

- 例えば、Haskell の IO モナドも同じ問題を抱えています
  - 手法は違えど、関数に色がついてしまう問題を回避する方法が必要です
- 一方で、非同期に限定すれば Go の goroutine はうまく回避しています
  - 関数側で同期/非同期を分類せず、呼び出し側で並行性を選べるため、と理解しています

後半

Algebraic Effect

and Handlers

をご存知でしょうか。

Algebraic Effect

- Adequacy for Algebraic Effects[2] で提唱された
- 和訳は「代数的エフェクト」と呼ばれることが多い
- 計算効果 (computational effects) を代数的に扱おうという試み
- モナドに対して、計算効果の操作的意味論を考えてみようという感じ

Algebraic Effect and Handlers

- Handlers of Algebraic Effects[3] で提唱された
- Algebraic Effect にハンドラを加えたもの
- ハンドラも含めて「Algebraic Effect」と呼ばれることもある
- 計算効果の発生だけでなく、発生後の処理について言及

簡単に言うと、

「継続を取ってこれる例外」

```
// 色がない関数
function getUser() {
  // await fetch(...);
  return perform({ type: 'Fetch' });
}
// 呼び出す関数も着色されない
function f() {
  const user = getUser();
  console.log(user.name);
}
// 非同期処理は呼び出し側で handle する
try {
  f();
} handle (effect) {
  if (effect.type === 'Fetch') {
    // 実際の非同期処理
    const user = await Promise.resolve({ name: 'Alice' });
    // 中断した処理から再開する = 保存しておいた継続を呼び出す
    resume(user);
  }
}
```

※ JavaScript に寄せた嘘文法[5]

結局、何が嬉しいの？

- function coloring が起きない
  - 高階関数を同期/非同期で用意しなくて良い
- ユーザーレベルで色々実装できる
- DI もできる
- 純粋関数型とも相性がいい

問題点

- 意図しないエフェクトの伝播
  - 要は例外なので投げっぱなしになるとランタイムエラーとかになる
  - 型システムが整備されていればだいぶマシ
- 効率が悪い
  - 要は例外なので実装次第で遅い
  - 例外が速い言語なら問題ない

そして、残念なお知らせ。

Algebraic Effects and Handlers が実装されている言語は、
研究のための実験的なものばかりです。

でも、皆さんは**一つだけ** Production-ready な言語を知っています。

そうですね。

今回のオチ

OCaml ですね。

参考資料

1. What color is your function?
2. Adequacy for Algebraic Effects
3. Handlers of Algebraic Effects
4. Algebraic Effects for the rest of us