2022年度未踏IT人材発掘・育成事業

直和型の代わりにユニオン型を持つ静的型付け関数型プログラミング言語の開発
－ Cotton －

伊藤 謙太朗（電気通信大学）
福間 遼太郎（慶應義塾大学）

動機

ユニオン型を静的型付け関数型言語に取り入れる良さを言語設計者を含めた多くの人に知ってもらい、ユニオン型を持つ静的型付け関数型言語を増やしたい。

開発したもの

Rustのenumや、HaskellやOCamlのデータ型、Scalaのシールドクラスのような機能の代わりにユニオン型を持ち、網羅性チェック付きのパターンマッチや型推論などの機能を実装したシンプルな静的型付け関数型言語Cottonのコンパイラを開発した。またシンタックスハイライトと型推論結果の表示の機能を持つランゲージサーバーを開発した。

実装した機能

- ユニオン型
- 再帰的な型エイリアス
- 網羅性チェック付きのパターンマッチ
- 型推論
- 演算子定義
- 高階多相
- オーバーロード
- インターフェース
- モジュールシステム

など

Cottonのコード例

Fibonacci

```
fib : I64 -> I64 =
   // 網羅性チェックつきのパターンマッチ
   | 0 => 0
   | 1 => 1
   // .は左辺に右辺の関数を適用する演算子
   | n => (n - 1).fib + (n - 2).fib

main : () -> () =
   | () => 10.fib.println
```

データ型・型エイリアス

```
   // 数字の0を表すデータ型
   data O
   // +1を表すデータ型
   // `forall`は型引数の宣言
   data S(A) forall { A }
   // 型エイリアス
   type Even = O | S[Odd]
   type Odd = S[Even]

   div2 : Even -> I64 =
      | O => 0
      | S(S(n)) => 1 + n.div2
```

二分木

```
data E
data T {
   value: A,
   left: Tree[A],
   right: Tree[A],
} forall { A }

type Tree = | A => E | T[A]

insert : Tree[I64] -> I64 -> Tree[I64] =
   | E, x => T { value: x, left: E, right: E }
   // `/\`はタプルのコンストラクタ
   | T { value: v, left: l, right: r }, x => (x < v /\ v < x).
      | True /\ _ => T { value: v, left: l.insert(x), right: r }
      | _ /\ True => T { value: v, left: l, right: r.insert(x) }
      | _ => T { value: v, left: l, right: r }

contains : Tree[I64] -> I64 -> Bool =
   | E, _ => False
   | T { value: v, left: l, right: r }, x => (x < v /\ v < x).
      | True /\ _ => l.contains(x)
      | _ /\ True => r.contains(x)
      | _ => True
```

リポジトリ: https://github.com/nanikamado/cotton
紹介記事: https://zenn.dev/nanikamado/articles/e352e024a17b3f
Playground: https://gitpod.io/#https://github.com/nanikamado/cotton-playground