IPA

2022 年度 未踏 IT 人材発掘・育成事業 採択案件評価書

1. 担当 PM

竹迫 良範（株式会社リクルート　データプロダクトユニット　ユニット長）

2. クリエータ氏名

伊藤 謙太朗（電気通信大学 情報理工学域 I 類コンピュータサイエンスプログラム）
福間 遼太郎（慶應義塾大学 理工学部情報工学科）

3. 委託金支払額

2,736,000 円

4. テーマ名

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

5. 関連 Web サイト

- コンパイラ、ランゲージサーバーのリポジトリ:
  https://github.com/nanikamado/cotton
- VSCode 向け拡張機能のリポジトリ:
  https://github.com/nanikamado/cotton-language-server
- Cotton の紹介記事:
  https://zenn.dev/nanikamado/articles/e352e024a17b3f
- Playground:
  https://gitpod.io/#https://github.com/nanikamado/cotton-playground

6. テーマ概要

本プロジェクトではシンプルな関数型プログラミング言語 Cotton を開発した。Cotton はパターンマッチの網羅性検査や高度な型推論などの、静的型付け関数型言語によくある機能を備えつつも、直和型（Rust の enum や、Haskell や OCaml のデータ型のような機能）を持っておらず、ユニオン型を持つことを特徴としている。本言語によって、静的型付け関数型言語にユニオン型を採用する有用性を実例によって示した。

7. 採択理由

enum は古くは C 言語から存在する列挙型で既存のプログラミング言語では当たり前のように使われてきたが、TypeScript v3.4 以降の const assertion を使うと、従来の列挙型の enum を安全にユニオン型として置き換えて書けるようになった。

TypeScript では enum と union の両方を使えるが、どのようなシーンで enum と union を使い分けるべきかといったコーディング標準のようなものはまだ整備されていないが、union は enum を内包する概念と表現力を持っているため、ユニオン型に統一してしまって記法をシンプルにしてしまうことに強力なメリットを感じている人も出始めている。

Rust の enum や Haskell や OCaml の data type、Swift の enumeration などで使われる直和型についても、もしもユニオン型の定義が使えれば、直和型を廃止した表現が可能になる。Scala や Kotlin で使われるシールドクラスもユニオン型に統一できる可能性がある。

最近は V 言語や Julia、Scala 3 などでもユニオン型が使えるので、最近はユニオン型に便利さを感じているプログラマも増えてきている。提案者達が直和型の代わりにユニオン型を持つ静的型付け関数型言語を新しく開発し、世の中に広く情報発信することによって、ユニオン型言語大統一理論を概念実証し、その後開発されるプログラミング言語に長期的な影響を与え続けることを期待したい。

8. 開発目標

静的型付け関数型言語の世界でユニオン型がまだ人気でない理由として、ユニオン型を採用する静的型付け関数型言語がまだ少ないことや、あったとしても、ユニオン型は直和型やシールドクラスなどの他の機能の補助的な役割を担うのみに留まっている点などがあると考えられる。本プロジェクトでは、網羅性チェックつきのパターンマッチや型推論、flat_map 関数に対するシンタックスシュガー（Haskell の do 記法や、Scala の for のような機能）などの、関数型言語によくある機能を取り入れつつも、直和型やシールドクラスなどではなくユニオン型を採用した型システムを持つシンプルな静的型付けプログラミング言語を開発することで、静的型付け関数型言語にユニオン型を採用する有用性を多くの人に知ってもらうことを目標とした。

9. 進捗概要

本プロジェクトでは静的型付けプログラミング言語 Cotton のコンパイラとランゲージサーバーを開発し、以下の機能を実装した。

- 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

ランゲージサーバーではシンタックスハイライトと型推論結果の表示の機能を実装し、VSCode で使えるようにした。コンパイラ、ランゲージサーバーは、MIT ライセンスで公開している。

Cotton で書かれたサンプルコードを図 1、図 2、図 3 に示す。

```
fib : I64 -> I64 =
    // 網羅性チェックつきのパターンマッチ
    | 0 => 0
    | 1 => 1    // '.'は左辺の値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
```

図 1：フィボナッチ数列の第 10 項目を出力する例

```
// 演算子定義
(..) : I64 -> I64 -> List[I64] =
    | a, b => (a < b).
        | True => a /\ (a + 1..b) // '/\'はリストのコンストラクタ
        | False => Nil

// 演算子の結合の方向・優先順位の宣言
infixl 4 ..

fizzbuzz : I64 -> String =
    // '/\'はタプルのコンストラクタでもある
    | n => (n % 3 /\ n % 5).
        | 0 /\ 0 => "FizzBuzz"
        | 0 /\ _ => "Fizz"
        | _ /\ 0 => "Buzz"
        | _ /\ _ => n.to_string

main : () -> () =
    | () => do
        (1..101).map(
            | i => i.fizzbuzz.println
        )
        ()
```

図 2：FizzBuzz を出力する例

```
// 数字の0を表すデータ型
data O
// +1を表すデータ型
data S(A) forall { A } // ‘forall‘は型引数の宣言

// 型エイリアス
type Nat = O | S[Nat]
type Even = O | S[Odd]
type Odd = S[Even]

// 偶数を引数にとって2で割った結果を言語組込みの整数として返す関数
div2 : Even -> I64 =
    | O => 0
    | S(S(n)) => 1 + n.div2
```

図 3：型エイリアスの例

10. プロジェクト評価

本プロジェクトでは、網羅性チェックつきのパターンマッチや高度な型推論、flat_map 関数に対するシンタックスシュガーなどの関数型言語によくある機能を持ちつつも直和型やシールドクラスなどではなくユニオン型を採用した型システムを持つシンプルな静的型付けプログラミング言語を開発したことで、ユニオン型の魅力や静的型付け関数型言語にユニオン型を採用する有用性を実際の動く実例によって示すことができた。

直和型と比較したユニオン型のメリットについての詳しい説明は、Web 上でプログラミング言語 Cotton の紹介記事（https://zenn.dev/nanikamado/articles/e352e024a17b3f）を通して公開したが、記事のコメント欄や Twitter などで「面白い」といった感想の他、言語設計に関するメリット・デメリットの議論が発生した。今後のプログラミング言語の設計に影響を与える場を作れたと評価できる。

11. 今後の課題

今回開発したプログラミング言語の特徴や設計ポリシーなどを記した解説用のドキュメントはブログ記事の形で公開はしたが、言語処理系としての公式ドキュメントを整備していく必要がある。現状では、実用的なプログラムを書くにはファイル操作や乱数の取得のような組み込み関数が不足しており、FFI の呼び出しインターフェースや、標準ライブラリを整備する必要がある。一部のエラーはわかりやすいエラーメッセージが出るが、開発者以外意味不明な内部表現そのままのエラーメッセージが出たりすることがあるため、エラーメッセージの改善も必要である。