Statia: 式レベル型注釈による段階的型付き Lisp 系言語

梶塚 太智[a)]

概要：本稿では, 式レベルで型注釈を可能にする段階的型付けの Lisp 系言語「Statia」を提案する. 従来のプログラミング言語では, 型注釈は変数や関数にのみ付与されることが一般的であり, 式全体に対して細かく指定することは難しい. Statia は, プログラムを構成する S 式に対してリストやアトムごとに型注釈を付与できる点が特徴であり, 式レベルの型指定をサポートすることで, より柔軟かつ安全なプログラム設計を可能にする. 本稿では, Statia の型システムの設計, 実行時の挙動, および式レベルでの型注釈による利点を考察し, プログラムの静的保証と柔軟性を両立させるための新しいアプローチを示す.

1. はじめに

現在のソフトウェア需要は型安全性にあると言えよう. 例えば Rust や TypeScript など, 最近の言語はパフォーマンスと静的型付けを前面に押し出している.

型推論の技術が進歩したため, 静的型でもプログラマは以前ほど型へ気を使わずに安全なプログラムを書けるようになっている.

動的型付け言語である Lisp では, declare 関数を使って型注釈を付与することが可能だが, 前述したソフトウェア需要により, 直感的で本質的な型システムを持つ Lisp の必要性を痛感し, この研究に至ったのである.

2. 型システム

Statia の型システムは, 段階的型付けを採用している. 型注釈を省略すると, 動的に式の結果の型が決まり, 従来の Lisp のように使用できる.

いっぽう型注釈を付けると, 実行前に式の結果の型が決まり, 厳密に型チェックをして式の結果の型が期待されたのと違えばエラーを吐く.

これにより, 動的型の柔軟性を保ちながらも, 必要に応じて型注釈を付与し型安全性を高めることが出来る.

3. 挙動

以下のコードは, 受けとった値を 2 倍する単純な関数を Statia で実装したものだ.

```
(define '(x2 n) '(* n 2))
```

型注釈が無いため動的型になり, 数値はもちろんのこと, 文字列を与えて呼び出しても暗黙の型変換されて実行される. 以下の出力例は REPL で実行した物である.

```
> (x2 3)
6
> (x2 " 4")
8
```

これに完全に型注釈を付けると同じ式で以下のようなコードになる.

```
(define:function '(x2:symbol n:number):
    list '(*:function n:number 2:number)
    :list):function
```

文法に注目するとコロンで S 式のアトムやリストを区切り, 左辺が式で右辺が型注釈となっている. これは, Python や Rust など多くの言語に見られプログラマにとって馴染みがあり非常に直感的な構文である.

型注釈があるので厳密に型チェックされ, 文字列を与えると引数の型が数値型なのでエラーを返す.

```
> (x2 5)
10
> (x2 " 6")
Error! the passed argument value `" 6"`
     is different to expected type `
    number` of the function
```

4. 式レベル型注釈の是非

式レベルで型注釈を付けられることには多くのメリットがある.

a) kajizukataichi@outlook.jp

一般的な言語では, 型注釈は変数に付けるものだが, 一つの変数に纏めようと長い式でメソッドチェーンや関数のネストの入れ子が膨大になり, 特に動的型付け言語の場合はバグの温床になる.

また静的型付け言語でも, あまり長い式では型推論が適切に処理しにくいといった問題もある.

一方 Statia の式レベル型注釈は, S 式のリストだけではなく, 変数やリテラルまでの詳細に型注釈を付与することが出来て, ソフトウェアの保守性も向上することが期待される.