自作言語 F--の紹介

自己紹介

- 名前:とやま ようた
- GitHub, Twitter: raviqqe
- HHKB が手に馴染んできた

F--

- 関数型コンパイラターゲット言語
- https://github.com/raviqqe/fmm

特徴

- C 言語に似ている
  - 弱い型付け
- 関数型
  - 再代入禁止
  - 後方へのジャンプが無い
  - 制御命令が `if`、`return`、`call` のみ
- LLVM と C のバックエンド
- **CPS 変換パスがある**

参考

- Haskell や OCaml の C--
- LLVM との類似･相違点
  - 全ての式が定数式
  - 全ての命令が文扱い
  - 関数やブロックがある
  - ブロック引数がある (MLIR に似ている)

CPS 変換とは

- プログラムを**継続呼び出しスタイル** (CPS) という形式に変換すること
  - 継続という関数の末尾呼び出しが唯一の制御構文
- 継続
  - プログラムのその後の動作を表現するオブジェクト
  - クロージャで実装

CPS 変換

目的

- スタックのガーベージコレクション
  - 末尾呼び出し最適化と組み合わせる
  - 関数型言語で書いたプログラムが意図通りに動く
    - 再帰によるループの実装
    - 使われていない、または、使われない変数はメモリに存在して欲しくない
- 非同期処理の実装
  - 全ての `return` 命令が継続呼び出しに置き換わる
  - つまり、普通の値を返す `return` がプログラムから無くなる
  - `return` を別の処理、例えば非同期処理での脱出 (yield)、に使える

CPS 変換

基本的なアルゴリズム

1. 全ての関数に継続を表す引数を追加
2. 全ての `return` 命令を渡された継続を呼び出すように変換
3. 全ての `call` 命令を奥から順に、
    i. その命令以降の処理を継続クロージャに包む
    ii. その継続を引数とする末尾呼び出しで置き換える

CPS 変換

注意

- 関数呼び出しのみを変換
  - `if` 命令等は変換しない
  - する必要がないため
- クロージャを全て独自スタックに置いている
  - malloc 重そう
- 参考：Compiling with continuations

まとめ

- F--という言語を作った
  - C に似たミニマルな関数型言語
  - より抽象度が高い言語のターゲット言語
  - CPS 変換ができる

今後の目標

- 非同期処理を高レイヤーの言語で実装したい