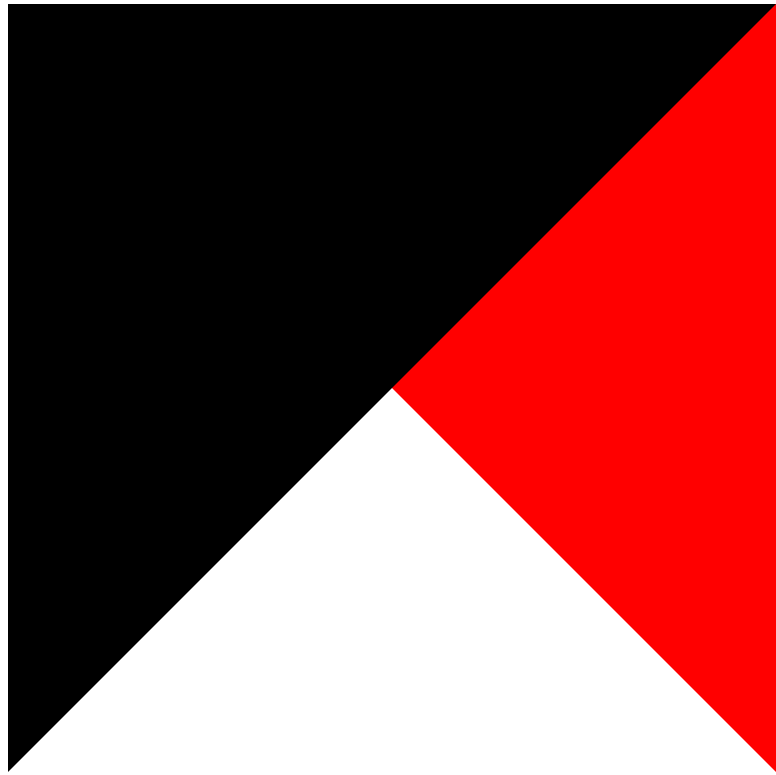
Stak Schemeの紹介

Shibuya.lisp #114

2024年3月28日

自己紹介

- raviqqe (GitHub, Bluesky)
- 普段はRustバックエンドエンジニア
- プログラミング言語が好き



概要

- Schemeとは
- Stak Schemeとは
  - Ribbit Schemeとは
  - Stak/Ribbit Schemeの比較
  - デモ
- 今後の予定

Schemeとは

- Lispの方言
- 単純な言語仕様
  - R7RSが最新
- 継続(`call/cc`)が有名

```
(display
  (call/cc
    (lambda (continue)
      (continue "Hello, world!"))))
```

Stak Schemeとは

- 自分が作っているScheme処理系
  - GitHub: https://github.com/raviqqe/stak
  - ドキュメンテーション: https://raviqqe.com/stak
- SchemeとRustで書かれている
- R7RS準拠が目標
- **標準ライブラリが無い環境でも動く**
- Ribbit Schemeが元

Stak Schemeとは

実装されている言語機能

- 標準ライブラリ `(scheme base)`, `(scheme read)`, `(scheme write)` の大体の手続き
- 継続 (`call/cc`)
- 例外 (`raise`, `guard`)
- マクロ (`define-syntax`, `syntax-rules`)
- ライブラリシステム (`define-library`, `import`, `export`)

Ribbit Schemeとは

- カナダのMontreal大学で作られたScheme処理系
- R4RS準拠
- **小さい**
  - 7KBメモリフットプリント
- 仮想マシン（VM）を色々な言語で実装
  - C, JavaScript, Python, x86アセンブリ等

Stak Schemeの実装

Ribbit Schemeと同じところ

- Schemeで書かれたバイトコードコンパイラ
- VM内部では全てがリスト
    - オブジェクトに加え、バイトコードや内部スタックも

Stak Schemeの実装

Ribbit Schemeと異なるところ

- R7RS準拠が目標
- Rustで書かれた仮想マシン
  - 命令セットはRibbit Schemeとほぼ同じ
- Rustとの相互運用性

Rustとの相互運用性

- RustからSchemeのコードを呼び出し
- 標準ライブラリが無い環境でも動く（e.g. ブラウザ上、組み込み）

```
const HEAP_SIZE: usize = 1 << 16;
const BUFFER_SIZE: usize = 1 << 10;

let mut heap = [Default::default(); HEAP_SIZE];
let device = stak_device::FixedBufferDevice::<BUFFER_SIZE, 0>::new(&[]);
let mut vm = stak_vm::Vm::new(&mut heap, stak_primitive::SmallPrimitiveSet::new(device)).unwrap();

const PROGRAM: &[u8] = stak_macro::compile_r7rs!(r#"
    (import (scheme write))

    (display "Hello, world!")
"#);

vm.initialize(PROGRAM.iter().copied()).unwrap();
vm.run().unwrap();

assert_eq!(vm.primitive_set().device().output(), b"Hello, world!");
```

デモ

Program

```
(import (scheme base) (scheme read) (scheme write))

(display "Hello, world!")
```

Compile　Interpret

stdin

stdout

https://raviqqe.com/stak/demo

今後の予定

- `eval`の実装
- ファイル操作
- コンパイラやVMの性能改善
- Rustオブジェクトの変換
    - RustとScheme間で値を共有

まとめ

- Stak Scheme処理系
  - Schemeで書かれたコンパイラ
  - Rustで書かれたVM
  - R7RS準拠が目標
  - Rustとの相互運用
- Scheme処理系作るのは楽しい