Embedding Stak Scheme in Rust

@raviqqe

January 14, 2024

Contents

Embedding Scheme in Rust

- Stak Scheme can be embeddable into Rust codes.
- Rust codes can simply "import" codes written in Stak Scheme.
- Currently, they can talk to each other only by I/O.
    - Object interoperability (maybe) in the future (probably!)

Examples

```rust
use stak_device::FixedBufferDevice;
use stak_macro::compile_r7rs;
use stak_primitive::SmallPrimitiveSet;
use stak_vm::Vm;

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

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

const PROGRAM: &[u8] = 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!");
```

Examples

Macros

- `compile_r7rs!` compiles R7RS Scheme codes into bytecodes.
- `include_r7rs!` includes and compiles R7RS Scheme codes from a file path.
- They run at a compile time.

Related crates

- A `stak-macro` crate contains `compile_r7rs!` and `include_r7rs!`.
- A `stak-compiler` crate contains a `compile_r7rs` function which runs compiler bytecodes and a VM for it to compile another Scheme program.
    - The crate is not dependent on another Scheme interpreter.

Release process

- A (binary) bytecode file is bundled with a crate of `stak-compiler` on release.
  - On development, they are built by `build.rs`.
  - On `cargo publish` of the crate, the bytecode file is bundled as an asset.
- `cargo install stak` or the other crates do not require another Scheme interpreter anymore!

Future work

- Scheme embedded in Rust
  - Scheme/Rust object interoperability
    - VTable?
    - Serde?
- Library system
- `eval` procedure

Summary

- Embedding Scheme in Rust is fun! 😄