Asynchronous operations in Stak Scheme

@raviqqe

May 10, 2025

Contents

- Stak Scheme
- Progress
  - Rust error handling in Scheme
  - The self-embedding compiler
  - Asynchronous operations in virtual machines
- Future work

Stak Scheme

- A bytecode compiler and virtual machine (VM) for Scheme
  - The compiler is written in Scheme.
  - The VM is written in Rust.
- It aims to support the R7RS-small standard.
- Forked from Ribbit Scheme

References

- GitHub
- Website

Progress

- Rust error handling in Scheme
- The self-embedding compiler
- Asynchronous operations in virtual machines

Rust error handling in Scheme

- Stak Scheme could handle errors from Rust in a limited way.
  - Rust primitives **return** error values.
  - If we check returned values and they are error values, we throw the errors in Scheme.
- In the new implementation of error handling, Rust primitives return `Result<V, E>` where `E` is an arbitrary error type.
  - The virtual machine captures such errors if error handlers are assigned in the Scheme side.
  - Then, it continues execution from the points of the error handlers.

Rust error handling in Scheme

- VM implementation in Rust
- Error handler implementation in Scheme

The self-embedding compiler

- The Stak Scheme compiler compiles itself to embed it into the `(scheme eval)` library while compiling given source codes.
- The article about it

Asynchronous operations in virtual machines

- The Stak Scheme virtual machine now handles asynchronous operations.
- Functions asynchronous potentially are marked with the `winter-maybe-async` crate.
- The current limitation is that it cannot make asynchronous and synchronous APIs coexist...
    - Feature unification | The Cargo book

Demo

- The REPL on browser thing

Future work

- Synchronous and asynchronous APIs in the same crate
- Unicode support
- Tree shaking
- `case-lambda` syntax
- `define-library` syntax in the command line interpreter
- `include` syntax

Summary

- Building Scheme is fun! 🥳