Differences between Stak and Ribbit Scheme

@raviqqe

October 22, 2023

Contents

- Ribbit Scheme
- Stak Scheme
- Differences between them
- Progress
- Next tasks

Ribbit Scheme

- A small and portable R4RS implementation
- Bytecode compiler and VM
- Instructions and primitive functions
- Everything is on heap.
  - Only integers and cons's
  - Bytecodes are lists.
  - A stack is a list.
  - A set of symbols is a list.

Stak Scheme

- A fork of Ribbit Scheme
- Its overall framework is the same as Ribbit Scheme.
- Some details deviated from Ribbit Scheme.
- There are some missing features from Ribbit Scheme.
  - e.g. incremental compilation and variadic instruction encoding

Primitive functions

- In Ribbit Scheme, primitive functions push a result value in a new cons onto a stack.
- In Stak Scheme, it destructively updates a top of a stack.
- In the design of Ribbit Scheme, we have both temporary values and bound variables (ones in a function frame) in the same stack.

```
(foo (+ 1 2)) ; (+ 1 2) -> 3 is a temporary value.

(let ((x 42)) ; x -> 42 is a bound variable.
  (foo x))
```

- Ribbit Scheme doesn't distinguish them and Stak does.
  - This implies that we do not need to push new values onto a stack always to treat it as a persistent data structure for continuations.
- No `drop` and `skip` primitive functions.

`apply` procedure

```
(apply f xs)
```

- In Ribbit Scheme, this is a primitive function.
- In Stak Scheme, this is a part of calling convention.

Arguments	Parameters	Algorithm
Fixed	Fixed	Compare an argument count
Fixed	Variadic	Stuff overflown arguments into a list (1)
Variadic	Fixed	Fill missing parameters with elements in the last argument of a list (2)

Skip instruction encoding

- Ribbit Scheme uses a special `skip` instruction to merge continuations of `if` instructions on decoding bytecodes.
  - The Ribbit Scheme compiler does not mark codes as continuations.
  - This is an optional feature.
- Stak Scheme provides the same feature by marking continuations of `if` instructions.
- It's a dirtier way but faster.
  - `O(n^2)` v.s. `O(n)`
- The Ribbit Scheme compiler needs to search for continuations of `if` instructions during encoding from bytecodes in memory into binary format while they are known at compilation.

Progress

- Record type
- Dynamic wind
- Parameter object
- Exception
- Website at https://raviqqe.github.io/stak/

Next tasks...

- Self-hosting
- Library system