1 source language $\mathcal{T}$

$$
\begin{array}{rl}
\texttt{n} ::= & n \in \mathcal{N} \\
e ::= & \texttt{n} \mid \texttt{true} \mid \texttt{false} \mid () \mid \texttt{x} \\
& \mid \texttt{e} + \texttt{e} \mid \texttt{e} - \texttt{e} \mid \texttt{e} * \texttt{e} \mid \texttt{e/e} \\
& \mid \texttt{e} = \texttt{e} \mid \texttt{e} < \texttt{e} \mid \texttt{e} >= \texttt{e} \mid \texttt{not e} \\
& \mid \texttt{if e then e else e} \\
& \mid \texttt{let x} = \texttt{e in e} \mid \texttt{fun x} \rightarrow \texttt{e} \mid \texttt{e e}
\end{array}
$$

2 Virtual Machine specificatoin

2.1 closure representation

$$Q = \langle I, C, F, Um \rangle$$

I is instruction sequense, C is constant table, F is closure table and Um is just an information, to create upvalues, composed of index and *meta* index, which of the former represents current register value and the latter does the upvalue index.

2.2 machine state

$$S = \langle Q, pc, U, R \rangle$$

ps is program counter which points nth instruction of I, U is upvalues table and R is register list.

2.3 values

$$v ::= \quad \texttt{null} \mid () \mid \texttt{i} \in \mathcal{N} \mid \texttt{true} \mid \texttt{false} \mid \texttt{closure}$$

2.4 Instruction set

- `Load(a, kx)`
  set `C[kx]` to `R[a]`
- `SetBool(a, x)`
  set boolean `x > 0` to `R[a]`
- `Unit(a)`
  set `()` to `R[a]`
- `Clos(a, cx)`
  create $U' = U$ @ $\{u \mid i_u \in Um \wedge u = R[i_u]\}$ and set $\langle \texttt{F[cx]}, 0, U', R' \rangle$ with environment to `R[a]`
- `Upval(a, ux)`
  set `U[rx]` to `R[a]`
- `Add(a, b, c)`
  set `R[b] + R[c]` to `R[a]`
- `Sub(a, b, c)`
  set `R[b] - R[c]` to `R[a]`
- `Mul(a, b, c)`
  set `R[b] * R[c]` to `R[a]`
- `Div(a, b, c)`
  set `R[b] / R[c]` to `R[a]`
- `Eq(a, b)`
  if `R[a] == R[b]` then `pc++`
- `Lt(a, b)`
  if `R[a] < R[b]` then `pc++`
- `Ge(a, b)`
  if `R[a] >= R[b]` then `pc++`
- `Test(a, p)`

if `p > 0 && R[a] || p <= 0 && !R[a]` then `pc++`

- `Jump(x)`

  `pc += x`

- `Move(a, b)`

  set `R[b]` to `R[a]`

- `Call(a, b)`

  call closure `R[a]` with argument `R[b]` and set return value to `R[a]`

- `Return(a)`

  terminate VM and returns `R[a]`

3 compilation: translate source language program to VM initial state (input bytecode)

relation $[\![\mathcal{T}, \Sigma, Q]\!] = Q', \Sigma'$

Σ is an environment from source language variable to register index.
$Q_{init}\,(e) = [\![e, [], \langle [], [], [], [] \rangle]\!]$

$$\frac{\texttt{a} \text{ and } \texttt{kx} \text{ are fresh}}{[\![\texttt{n}, \Sigma, \langle I, C, F, Um \rangle]\!] = \langle I; \texttt{Load(a, kx)}; \texttt{Return(a)}, C; \texttt{kx} = \texttt{n}, F, Um \rangle, \Sigma'} \quad \text{(Int)}$$

$$[\![(), \Sigma, \langle I, C, F, Um \rangle]\!] = \langle I; \texttt{Unit(a)}; \texttt{Return(a)}, C, F, Um \rangle, \Sigma' \quad \text{(Unit)}$$

$$[\![\texttt{true}, \Sigma, \langle I, C, F, Um \rangle]\!] = \langle I; \texttt{SetBool(a, 1)}; \texttt{Return(a)}, C, F, Um \rangle, \Sigma' \quad \text{(True)}$$

$$[\![\texttt{false}, \Sigma, \langle I, C, F, Um \rangle]\!] = \langle I; \texttt{SetBool(a, 0)}; \texttt{Return(a)}, C, F, Um \rangle, \Sigma' \quad \text{(False)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um \rangle \qquad [\![\texttt{e1}, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um' \rangle, \Sigma' \\ [\![\texttt{e2}, \Sigma', \langle I', C', F', Um' \rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um'' \rangle, \Sigma''\end{array}}{[\![\texttt{e1} + \texttt{e2}, \Sigma, Q]\!] = \langle I''; \texttt{Add}(\texttt{a}, \texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{Return}(\texttt{a}), C'', F'', Um'' \rangle, \Sigma''} \quad \text{(Add)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um \rangle \qquad [\![\texttt{e1}, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um' \rangle, \Sigma' \\ [\![\texttt{e2}, \Sigma', \langle I', C', F', Um' \rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um'' \rangle, \Sigma''\end{array}}{[\![\texttt{e1} - \texttt{e2}, \Sigma, Q]\!] = \langle I''; \texttt{Sub}(\texttt{a}, \texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{Return}(\texttt{a}), C'', F'', Um'' \rangle, \Sigma''} \quad \text{(Sub)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um \rangle \qquad [\![\texttt{e1}, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um' \rangle, \Sigma' \\ [\![\texttt{e2}, \Sigma', \langle I', C', F', Um' \rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um'' \rangle, \Sigma''\end{array}}{[\![\texttt{e1} * \texttt{e2}, \Sigma, Q]\!] = \langle I''; \texttt{Mul}(\texttt{a}, \texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{Return}(\texttt{a}), C'', F'', Um'' \rangle, \Sigma''} \quad \text{(Mul)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um \rangle \qquad [\![\texttt{e1}, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um' \rangle, \Sigma' \\ [\![\texttt{e2}, \Sigma', \langle I', C', F', Um' \rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um'' \rangle, \Sigma''\end{array}}{[\![\texttt{e1}/\texttt{e2}, \Sigma, Q]\!] = \langle I''; \texttt{Div}(\texttt{a}, \texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{Return}(\texttt{a}), C'', F'', Um'' \rangle, \Sigma''} \quad \text{(Div)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um \rangle \qquad [\![e1, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um' \rangle, \Sigma' \\ [\![e2, \Sigma', \langle I', C', F', Um' \rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um'' \rangle, \Sigma''\end{array}}{\begin{array}{c}[\![\texttt{e1} = \texttt{e2}, \Sigma, Q]\!] \\ = \langle I''; \texttt{Eq}(\texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{SetBool(a, 1)}; \texttt{SetBool(a, 0)}; \texttt{Return(a)}, C'', F'', Um'' \rangle, \Sigma''\end{array}} \quad \text{(Eq)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um \rangle \qquad [\![e1, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um' \rangle, \Sigma' \\ [\![e2, \Sigma', \langle I', C', F', Um' \rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um'' \rangle, \Sigma''\end{array}}{\begin{array}{c}[\![\texttt{e1} = \texttt{e2}, \Sigma, Q]\!] \\ = \langle I''; \texttt{Lt}(\texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{SetBool(a, 1)}; \texttt{SetBool(a, 0)}; \texttt{Return(a)}, C'', F'', Um'' \rangle, \Sigma''\end{array}} \quad \text{(Lt)}$$

$$\frac{\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um\rangle \qquad [\![e1, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um'\rangle, \Sigma' \qquad [\![e2, \Sigma', \langle I', C', F', Um'\rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um''\rangle, \Sigma''}{\begin{array}{c}[\![\texttt{e1} = \texttt{e2}, \Sigma, Q]\!] \\ = \langle I''; \texttt{Ge}(\texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{SetBool}(\texttt{a}, 1); \texttt{SetBool}(\texttt{a}, 0); \texttt{Return}(\texttt{a}), C'', F'', Um''\rangle, \Sigma''\end{array}} \quad \text{(GE)}$$

$$\frac{\texttt{a} \text{ is fresh} \qquad Q = \langle I, C, F, Um\rangle \qquad [\![e, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e}}), C', F', Um'\rangle, \Sigma'}{\begin{array}{c}[\![\texttt{not e}, \Sigma, Q]\!] \\ = \langle I'; \texttt{Test}(\texttt{r}_{\texttt{e}}, 0); \texttt{SetBool}(\texttt{a}, 1); \texttt{SetBool}(\texttt{a}, 0); \texttt{Return}(\texttt{a}), C', F', Um'\rangle, \Sigma'\end{array}} \quad \text{(NOT)}$$

$$\frac{\begin{array}{c}\texttt{a} \text{ and } \texttt{cx} \text{ are fresh} \qquad Q = \langle I, C, F, Um\rangle \qquad fv = FV\,(\texttt{fun x} \to \texttt{e}) \qquad \Sigma_0 = \{(\texttt{x} = \texttt{i}_{\texttt{r}}) \mid (\texttt{x} = \texttt{i}_{\texttt{r}}) \in \Sigma \wedge \texttt{x} \in fv\} \\ \Sigma' = \{(\texttt{x} = -\,(\texttt{i} + 1)) \mid (\texttt{x} = \texttt{i}_{\texttt{r}}) \text{ as } xi \in \Sigma_0 \wedge xi \text{ is } i\text{th element of } \Sigma_0\} \qquad Um' = \{\texttt{i} \mid (\_, \texttt{i}) \in \Sigma'\} \\ Q_{init}\,(\texttt{e}) = [\![\texttt{e}, \Sigma_{\texttt{e}}, Q_{\texttt{e}}]\!] \qquad Q_{\texttt{e}}[Um] := Um' \qquad \Sigma'_{\texttt{e}} = \Sigma_{\texttt{e}}; \texttt{x} = 0 \qquad [\![\texttt{e}, \Sigma'_{\texttt{e}}, Q_{\texttt{e}}]\!] = \langle I_{\texttt{e}}, C_{\texttt{e}}, F_{\texttt{e}}, \_\rangle, \Sigma''_{\texttt{e}} \\ Clos = \langle I_{\texttt{e}}, C_{\texttt{e}}, F_{\texttt{e}}, Um@Um'\rangle\end{array}}{[\![\texttt{fun x} \to \texttt{e}, \Sigma, Q]\!] = \langle I; \texttt{Clos}(\texttt{a}, \texttt{cx}); \texttt{Return}(\texttt{a}), C, F; \texttt{cx} = Clos, Um\rangle, \Sigma} \quad \text{(FUN)}$$

$$\frac{Q = \langle I, C, F, Um\rangle \qquad [\![\texttt{e1}, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{r}_{\texttt{e1}}), C', F', Um'\rangle, \Sigma' \qquad [\![\texttt{e2}, \Sigma', \langle I', C', F', Um'\rangle]\!] = \langle I''; \texttt{Return}(\texttt{r}_{\texttt{e2}}), C'', F'', Um''\rangle, \Sigma''}{[\![\texttt{e1 e2}, \Sigma, Q]\!] = \langle I''; \texttt{Call}(\texttt{r}_{\texttt{e1}}, \texttt{r}_{\texttt{e2}}); \texttt{Return}(\texttt{a}), C'', F'', Um''\rangle, \Sigma''} \quad \text{(CALL)}$$

$$\frac{\Sigma[\texttt{x}] = \texttt{i} \wedge \texttt{i} >= 0}{[\![\texttt{x}, \Sigma, \langle I, C, F, Um\rangle]\!] = \langle I; \texttt{Move}(\texttt{a}, \texttt{i}); \texttt{Return}(\texttt{a}), C, F, Um\rangle, \Sigma'} \quad \text{(VARLOCAL)}$$

$$\frac{\Sigma[\texttt{x}] = \texttt{i} \wedge \texttt{i} < 0}{[\![\texttt{x}, \Sigma, \langle I, C, F, Um\rangle]\!] = \langle I; \texttt{Upval}(\texttt{a}, -\texttt{i} - 1); \texttt{Return}(\texttt{a}), C, F, Um\rangle, \Sigma'} \quad \text{(VARUPVAL)}$$

$$\frac{\begin{array}{c}\texttt{b} \text{ is fresh} \qquad Q = \langle I, C, F, Um\rangle \\ [\![e_1, \Sigma, Q]\!] = \langle I'; \texttt{Return}(\texttt{a}), C', F', Um'\rangle, \Sigma' \qquad [\![e_2, \Sigma'; \texttt{x} = \texttt{a}, \langle I'; \texttt{Move}(\texttt{b}, \texttt{a}), C', F', Um'\rangle]\!] = Q', \Sigma''\end{array}}{[\![\texttt{let x} = \texttt{e}_1 \texttt{ in } \texttt{e}_2, \Sigma, Q]\!] = Q', \Sigma''} \quad \text{(LET)}$$