# 1 source language $\mathcal{T}$

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

# 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 tableand $Um$ 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 $U' = U \mathbin{@} \{u \mid i_u \in Um \wedge u = R[i_u]\}$ 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)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}Q=\langle I,C,F,Um\rangle\qquad [\![\mathtt{e1},\Sigma,Q]\!]=\langle I';\mathtt{Return}(\mathtt{r_{e1}}),C',F',Um'\rangle,\Sigma'\\ [\![\mathtt{e2},\Sigma',\langle I',C',F',Um'\rangle]\!]=\langle I'';\mathtt{Return}(\mathtt{r_{e2}}),C'',F'',Um''\rangle,\Sigma''\end{array}}{[\![\mathtt{e1\ e2},\Sigma,Q]\!]=\langle I'';\mathtt{Call}(\mathtt{r_{e1}},\mathtt{r_{e2}});\mathtt{Return}(\mathtt{a}),C'',F'',Um''\rangle,\Sigma''}\quad(\textsc{Call})$$

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

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

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