1 source language $\mathcal{T}$

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

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 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

$$S = \langle Q, \texttt{pc}, U, R \rangle$$

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

2.3 values

$$v ::= \quad \texttt{null} \mid () \mid \texttt{i} \in \mathcal{N} \mid \texttt{true} \mid \texttt{false} \mid \texttt{clos}\,(Q, U)$$

2.4 Instructions

- `Load(a, kx)`

  set `C[kx]` to `R[a]`

- `SetBool(a, x, p)`

  set boolean `x > 0` to `R[a]`; if `p = 1` then `pc++`

- `Unit(a)`

  set `()` to `R[a]`

- `Clos(a, cx, p)`

  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]` if `p=1` then the closure is recursive

- `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)`

  exit closure evaluation and return `R[a]`

- `TailCall(a, b)`

  call closure `R[a]` with argument `R[b]` and exit closure evaluation and return `R[a]`

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

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

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

3.1 auxiliary funcion

We define an auxiliary function to get last used register and rest instructions by pattern match.

$$\begin{array}{lcr} mtch\,(I; \texttt{Return}\,(\texttt{a})) & = & (\texttt{a}, I) \\ mtch\,(I; \texttt{TailCall}\,(\texttt{a}, \texttt{b})) & = & (\texttt{a}, I; \texttt{Call}\,(\texttt{a}, \texttt{b})) \\ mtch\,(\_) & = & \text{undefined} \end{array}$$

3.2 compilation rules

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

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

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

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

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

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

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

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

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

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

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

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

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

$(\textsc{Fun})$

$$\frac{\begin{array}{c}Q=\langle I,C,F,Um\rangle\qquad [\![\texttt{e1},\Sigma,Q]\!]=\langle I_{\texttt{e1}},C',F',Um'\rangle,\Sigma'\qquad mtch\,(I_{\texttt{e1}})=(\texttt{r}_{\texttt{e1}},I')\\ [\![\texttt{e2},\Sigma',\langle I',C',F',Um'\rangle]\!]=\langle I_{\texttt{e2}},C'',F'',Um''\rangle,\Sigma''\qquad mtch\,(I_{\texttt{e2}})=(\texttt{r}_{\texttt{e2}},I'')\end{array}}{[\![\texttt{e1 e2},\Sigma,Q]\!]=\langle I'';\texttt{TailCall}(\texttt{r}_{\texttt{e1}},\texttt{r}_{\texttt{e2}}),C'',F'',Um''\rangle,\Sigma''}\quad(\textsc{Call})$$

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

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

$$\frac{\begin{array}{c}\texttt{a}\text{ and }\texttt{b}\text{ are fresh}\qquad Q=\langle I,C,F,Um\rangle\\ [\![\texttt{e}_1,\Sigma,Q]\!]=\left\langle I_{\texttt{e1}},C',F',Um'\right\rangle,\Sigma'\qquad mtch\,(I_{\texttt{e1}})=(\texttt{r}_{\texttt{e1}},I')\\ [\![\texttt{e}_2,\Sigma';\texttt{x}=\texttt{a},\left\langle I';\texttt{Move}(\texttt{b},\texttt{a}),C',F',Um'\right\rangle]\!]=Q',\Sigma''\end{array}}{[\![\texttt{let x}=\texttt{e}_1\texttt{ in e}_2,\Sigma,Q]\!]=Q',\Sigma''}\quad(\textsc{Let})$$

$$\frac{\begin{array}{c}\texttt{a}\text{ and }\texttt{cx}\text{ are fresh} \qquad Q = \langle I, C, F, Um\rangle \\ fv = (FV\,(\texttt{e}_1) \setminus \{\texttt{x}\} \cap FV\,(\texttt{e}_2)) \setminus \{f\} \qquad \Sigma_0 = \{(\texttt{x} = \texttt{i}_\texttt{r}) \mid (\texttt{x} = \texttt{i}_\texttt{r}) \in \Sigma \wedge \texttt{x} \in fv\} \cap \{\texttt{f} = \texttt{a}\} \\ \Sigma' = \{(\texttt{x} = -(\texttt{i}+\texttt{1})) \mid (\texttt{x} = \texttt{i}_\texttt{r})\ \text{ as } xi \in \Sigma_0 \wedge xi \text{ is } i\text{th element of } \Sigma_0\} \\ Um' = \{i \mid (\_ = \texttt{i}) \in \Sigma'\} \qquad Q_{init}(\texttt{e}_1) = [\![\texttt{e1}, \Sigma_{\texttt{e1}}, \langle I_{\texttt{e1}}, C_{\texttt{e1}}, F_{\texttt{e1}}, \_\rangle]\!] \qquad \Sigma'_{\texttt{e1}} = \Sigma_{\texttt{e1}}; \texttt{x} = \texttt{0} \\ [\![\texttt{e1}, \Sigma'_{\texttt{e1}}, \left\langle I_{\texttt{e1}}, C_{\texttt{e1}}, F_{\texttt{e1}}, Um'\right\rangle]\!] = \langle I'_{\texttt{e1}}, C'_{\texttt{e1}}, F'_{\texttt{e1}}, \_\rangle, \Sigma''_{\texttt{e1}} \qquad Clos = \langle I_{\texttt{e1}},', C'_{\texttt{e1}}, F'_{\texttt{e1}}, Um@Um'\rangle \\ I' = I; \texttt{Clos}(\texttt{a}, \texttt{cx}, \texttt{1}) \qquad F' = F; \texttt{cx} = Clos \qquad \Sigma' = Sigma; \texttt{f} = \texttt{a} \qquad [\![\texttt{e2}, \Sigma', \langle I', C, F', Um\rangle]\!] = Q', \Sigma''\end{array}}{[\![\texttt{let rec f x} = \texttt{e1 in e2}, \Sigma, Q]\!] = Q', \Sigma''}$$

(LetRecFun)

$FV\,(e)$ means a set of free variables in e

4 Evaluation: VM state transition rules