1 Concrete Semantics of Scheme CESK*

Syntax Domains:

$$
\begin{aligned}
e \in \mathsf{Exp} ::= \;& \text{æ} \\
\mid\;& (\texttt{if}\ e\ e\ e) \\
\mid\;& (\texttt{let}\ ([x\ e]\ ...)\ e) \\
\mid\;& (\texttt{call/cc}\ e) \\
\mid\;& (\texttt{set!}\ x\ e) \\
\mid\;& (\texttt{prim}\ op\ e\ ...) \\
\mid\;& (\texttt{apply-prim}\ op\ e) \\
\mid\;& (\texttt{apply}\ e\ e) \\
\mid\;& (e\ e\ ...)
\end{aligned}
$$

$$
\begin{aligned}
\text{æ} \in \mathsf{AExp} ::= \;& lam \mid \mathbb{Z} \mid \texttt{\#t} \mid \texttt{\#f} \\
\mid\;& (\texttt{quote}\ e)
\end{aligned}
$$

$$lam \in \mathsf{Lam} ::= (\lambda\ (x...)\ e) \mid (\lambda\ x\ e)$$

$x \in \mathsf{Var}$ A set of identifiers

$op \in \mathsf{Prim}$ A set of primitives

Atomic Evaluation:

Transition:

Injection:

Collection:

Tick/Alloc:

Semantic Domains:

$$
\begin{aligned}
\varsigma \in \Sigma &\triangleq \mathit{Eval} + \mathit{Apply} \\
\mathit{Eval} &\triangleq \mathsf{Exp} \times \mathit{Env} \\
&\quad \times \mathit{Addr} \times \mathit{Time} \\
\mathit{Apply} &\triangleq \mathit{Val} \times \mathit{Env} \\
&\quad \times \mathit{Addr} \times \mathit{Time} \\
\rho \in \mathit{Env} &\triangleq \mathsf{Var} \rightharpoonup \mathit{Addr} \\
\sigma \in \mathit{Store} &\triangleq \mathit{Addr} \rightharpoonup \mathit{Val} \\
v \in \mathit{Val} &\triangleq \mathit{Clo} + \mathit{Kont} + \mathbb{Z} \\
&\quad + \{\texttt{\#t}, \texttt{\#f}, \mathit{Null}, \mathit{Void}\} \\
&\quad + \mathbf{quote}(e) + \mathbf{cons}(v, v) \\
clo \in \mathit{Clo} &\triangleq \mathsf{Lam} \times \mathit{Env} \\
a, b, c \in \mathit{Addr} &\triangleq \mathbb{N} \\
t, u \in \mathit{Time} &\triangleq \mathbb{N} \\
\kappa \in \mathit{Kont} &::= \mathbf{mt} \\
&\mid \mathbf{ifk}(e, e, \rho, a) \\
&\mid \mathbf{callcck}(a) \mid \mathbf{setk}(x, a) \\
&\mid \mathbf{appappk}(val?, e, \rho, a) \\
&\mid \mathbf{appk}(done, todo, \rho, a) \\
&\mid \mathbf{appprimk}(op, a) \\
&\mid \mathbf{primk}(op, done, todo, \\
&\qquad\qquad \rho, a) \\
&\mid \mathbf{letk}(vars, done, todo \\
&\qquad\qquad e, \rho, a) \\
done &\triangleq \mathit{Val}^* \\
todo &\triangleq \mathsf{Exp}^*
\end{aligned}
$$

Eval Rules

Rules for when the control is an expression

$$E\langle æ, \rho, a, t\rangle \rightsquigarrow A\langle v, \rho, a, u\rangle$$
where $u = tick(st, 1)$
$v = \mathcal{A}(\varsigma, \sigma)$

$$E\langle(\texttt{if}\ e_c\ e_t\ e_f), \rho, a, t\rangle \rightsquigarrow E\langle e_c, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{ifk}(e_t, e_f, \rho, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{let}\ ()\ e), \rho, a, t\rangle \rightsquigarrow E\langle e, \rho, a, u\rangle$$
where $u = tick(\varsigma, 1)$

$$E\langle(\texttt{let}\ ([x_0\ e_0]\ [x_s\ e_s]\ ...)\ e_b), \rho, a, t\rangle \rightsquigarrow E\langle e_0, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{letk}(x_0 :: x_s, [\,], e_s, e_b, \rho, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{call/cc}\ e), \rho, a, t\rangle \rightsquigarrow E\langle e, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{callcck}(a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{set!}\ x\ e), \rho, a, t\rangle \rightsquigarrow E\langle e, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{setk}(x, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{prim}\ op), \rho, a, t\rangle \rightsquigarrow A\langle v, \rho, a, u\rangle$$
where $v = op$ applied to 0 arguments
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{prim}\ op\ e_0\ e_s\ ...), \rho, a, t\rangle \rightsquigarrow E\langle e_0, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{primk}(op, [\,], x_s, \rho, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{apply-prim}\ op\ e), \rho, a, t\rangle \rightsquigarrow E\langle e, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{appprimk}(op, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(\texttt{apply}\ e_f\ e_x), \rho, a, t\rangle \rightsquigarrow E\langle e_f, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{appappk}(\varnothing, e_x, \rho, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

$$E\langle(e_f\ e_s\ ...), \rho, a, t\rangle \rightsquigarrow E\langle e_f, \rho, b, u\rangle$$
where $\sigma[b \mapsto \textbf{appk}([\,], e_s, \rho, a)]$
$b = alloc(\varsigma, 0)$
$u = tick(\varsigma, 1)$

Apply Rules

Rules for when the control is a value

$$A\langle v, \rho, a, t\rangle \rightsquigarrow E\langle e_f, \rho, b, u\rangle$$
$$\begin{aligned} \text{where } \kappa &\triangleq \sigma(a) \\ \kappa &= \mathbf{ifk}(e_t, e_f, \rho, a) \\ u &= tick(\varsigma, 1) \\ v &= \texttt{\#f} \end{aligned}$$

$$A\langle v, \rho, a, t\rangle \rightsquigarrow E\langle e_t, \rho, b, u\rangle$$
$$\begin{aligned} \text{where } \kappa &\triangleq \sigma(a) \\ \kappa &= \mathbf{ifk}(e_t, e_f, \rho, a) \\ u &= tick(\varsigma, 1) \\ v &\neq \texttt{\#f} \end{aligned}$$

$$A\langle v, \rho, a, t\rangle \rightsquigarrow E\langle e_c, \rho, b, u\rangle$$
$$\begin{aligned} \text{where } \sigma[b \mapsto \mathbf{ifk}(e_t, e_f, \rho, a)] \\ b = alloc(\varsigma, 0) \\ u = tick(\varsigma, 1) \end{aligned}$$

Syntax:

$$
\begin{aligned}
e \in \mathsf{Exp} &::= æ \\
&\mid (\texttt{if}\ e\ e\ e) \\
&\mid (\texttt{let}\ ([x\ e]\ ...)\ e) \\
&\mid (\texttt{prim}\ op\ e\ ...)e \\
&\mid (\texttt{call/cc}\ e) \\
&\mid (e\ e\ ...) \\
æ \in \mathsf{AExp} &::= lam \mid \mathbb{Z} \mid \texttt{\#t} \mid \texttt{\#f} \\
lam \in \mathsf{Lam} &::= (\lambda\ (x...)\ e) \\
x \in \mathsf{Var} & \quad \text{A set of identifiers}
\end{aligned}
$$

Atomic Evaluation Function:

$$
\begin{aligned}
\mathcal{A} &: (\mathsf{AExp} \times Env \times Store) \to Val \\
\mathcal{A}(x, \rho, \sigma) &\triangleq \sigma(\rho(x)) \\
\mathcal{A}(lam, \rho, \sigma) &\triangleq (lam, \rho) \\
\mathcal{A}(æ, \rho, \sigma) &\triangleq æ
\end{aligned}
$$

Transition Function:

$$(\Sigma \times Store) \rightsquigarrow (\Sigma \times Store)$$

Semantics:

$$
\begin{aligned}
\varsigma \in \Sigma &\triangleq \mathsf{Exp} \times Env \times Addr \\
\rho \in Env &\triangleq \mathsf{Var} \rightharpoonup Addr \\
\sigma \in Store &\triangleq Addr \rightharpoonup Val \\
v \in Val &\triangleq Clo + Kont \\
&\quad + \mathbb{Z} + \{\texttt{\#t}, \texttt{\#f}\} \\
clo \in Clo &\triangleq \mathsf{Lam} \times Env \\
a, b, c \in Addr & \quad \text{A set of addreses} \\
\kappa \in Kont &\triangleq \mathbf{mt} \\
&\mid \mathbf{ifk}(e, e, \rho, a) \\
&\mid \mathbf{letk}(x, e, \rho, a) \\
&\mid \mathbf{appk}(done, todo, \\
&\qquad \rho, a) \\
&\mid \mathbf{primk}(op, done, todo, \\
&\qquad \rho, a) \\
&\mid \mathbf{callcck}(a) \\
op \in Prim & \quad \text{A set of primitives} \\
done &\triangleq Val^* \\
todo &\triangleq \mathsf{Exp}^*
\end{aligned}
$$

$(\varsigma \times \sigma) \rightsquigarrow (\varsigma' \times \sigma)$, where $\kappa = \sigma(a), b = alloc(\varsigma)$
proceed by matching on ς

$\langle(\texttt{if}\ e_c\ e_t\ e_f), \rho, a\rangle$	$\langle e_c, \rho, b\rangle$ $\sigma[b \mapsto \mathbf{ifk}(e_t, e_f, \rho, a)]$
$\langle(\texttt{let}\ ([x\ e_x]\ ...)\ e_b), \rho, a\rangle$	$\langle e_x, \rho, b\rangle$ $\sigma[b \mapsto \mathbf{letk}(x, e_b, \rho, a)]$
$\langle(\texttt{prim}\ op\ e_0\ ...)es, \rho, a\rangle$	$\langle e_0, \rho, b\rangle$ $\sigma[b \mapsto \mathbf{appk}([op], es, \rho, a)]$
$\langle(\texttt{call/cc}\ e), \rho, a\rangle$	$\langle e, \rho, b\rangle$ $\sigma[b \mapsto \mathbf{appk}([\mathbf{call/cc}], [\,], \rho, a)]$
$\langle(e_f\ es...), \rho, a\rangle$	$\langle e_f, \rho, b\rangle$ $\sigma[b \mapsto \mathbf{appk}([\,], e_s, \rho, a)]$
$\langle æ, \rho, a\rangle$ let $v = \mathcal{A}(æ, \rho, \sigma)$ match on κ below	
$\mathbf{mt}$	ς
$\mathbf{ifk}(e_t, e_f, \rho', c)$ when $v = \texttt{\#f}$	$\langle e_f, \rho', c\rangle$
$\mathbf{ifk}(e_t, e_f, \rho', c)$ when $v \neq \texttt{\#f}$	$\langle e_t, \rho', c\rangle$
$\mathbf{letk}(x, e_b, \rho', c)$	$\langle e_b, \rho'[x \mapsto b], c\rangle$ $\sigma[b \mapsto v]$
$\mathbf{appk}(done, e_h :: e_t, \rho', c)$	$\langle e_h, \rho', b\rangle$ $\sigma[b \mapsto \mathbf{appk}(done +\!\!+ [v], e_t, \rho', c)]$
$\mathbf{appk}(op :: v_s, [\,], \rho', c)$	$\langle v', \rho', c\rangle$ $v' = op$ applied to $(v_s +\!\!+ [v])$
$\mathbf{appk}(clo :: v_s, [\,], \rho', c)$ where $clo = ((\lambda\ (x_s...)\ e_b), \rho_\lambda)$	$\langle e_b, \rho_\lambda[x_{s0} \mapsto b_0\ ...\ x_{sn} \mapsto b_n], c\rangle$ $v'_s = v_s +\!\!+ [v]$ $\sigma[b_0 \mapsto v'_{s0}\ ...\ b_n \mapsto v'_{sn}]$
$\mathbf{appk}([\mathbf{call/cc}], [\,], \rho', c)$ where $v = ((\lambda\ (x)\ e), \rho_\lambda)$	$\langle e, \rho_\lambda[x \mapsto c], c\rangle$
$\mathbf{appk}([\kappa_v], [\,], \rho', c)$	$\langle v, \rho', b\rangle$ $\sigma[b \mapsto \kappa_v]$

Rules returning a value in control position are broken! Values are not syntax! What is the solution‽

2 Abstract Semantics of Scheme CESK*