1 Formalization of a CESK* machine with basic Scheme features.

But the Store is globalized, so CEK* with a global store.

$$
\begin{aligned}
\varsigma \in \Sigma &= \mathsf{Exp} \times \mathsf{Env} \times \mathsf{Kont} \\
op \in \mathsf{Primitive} & \quad \text{The set of primitves} \\
\psi \in \mathsf{PrimStore} &= Var \rightharpoonup \mathsf{Primitive} \\
v \in \mathsf{Val} &::= (\lambda\ (x\ ...)\ e) \mid op \mid\ \#\text{t} \mid \#\text{f} \mid 0 \mid 1\ ... \\
x \in \mathsf{Var} & \quad \text{A set of identifiers} \\
e \in \mathsf{Exp} &::= x \mid v \mid (\text{if}\ e\ e\ e) \mid (let\ (x\ e)\ e) \mid (e\ e\ ...) \\
\mathsf{a,b,c} \in \mathsf{Addr} & \quad \text{A set of addresses} \\
\rho \in \mathsf{Env} &= \mathsf{Var} \rightharpoonup \mathsf{Addr} \\
\sigma \in \mathsf{Store} &= \mathsf{Addr} \rightharpoonup (\mathsf{Val} \times \mathsf{Env}) \\
done &= \text{list of } \mathsf{Val} \\
todo &= \text{list of } \mathsf{Exp} \\
\kappa \in \mathsf{Kont} &= \mathbf{mt} \mid \mathbf{appf}(done, todo, \rho, a) \\
& \quad \mid \mathbf{iff}(e, e, \rho, a) \\
& \quad \mid \mathbf{letf}(x, e, \rho, a)
\end{aligned}
$$

$$alloc(\varsigma) = \text{a new address}$$

Our transition function is of type

$$(\Sigma \times \mathsf{Store}) \rightarrow (\Sigma \times \mathsf{Store})$$

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

$\langle x, \rho, a\rangle$	$\langle v, \rho', a\rangle$ where $(v, \rho') = \sigma(\rho(x))$
$\langle x, \rho, a\rangle$	$\langle op, \rho, a\rangle$ where $x \notin \rho$ and $op = \psi(x)$
$\langle (e_0\ e_z ...), \rho, a\rangle$	$\langle e_0, \rho, b\rangle$ $\sigma = \sigma[b \longmapsto \textbf{appf}([\,], e_z, \rho, a)]$
$\langle (\text{if } e_c\ e_t\ e_f), \rho, a\rangle$	$\langle e_c, \rho, b\rangle$ $\sigma = \sigma[b \longmapsto \textbf{iff}(e_t, e_f, \rho, a)]$
$\langle (\text{let } (x\ e_0)\ e_1), \rho, a\rangle$	$\langle e_0, \rho, b\rangle$ $\sigma = \sigma[b \longmapsto \textbf{letf}(x, e_1, \rho, a)]$
$\langle v, \rho, a\rangle$ match on κ below	
mt	ς
$\textbf{appf}((op :: (vs...), [\,], \rho', c)$	$\langle v', \rho', c\rangle$ $v' = op$ applied to $(vs +\!\!+ [v])$
$\textbf{appf}((\lambda(xs...)e) :: (vs...), [\,], \rho', c)$	$\langle e, \rho'[xs_0 \longmapsto b_0 ... xs_i \longmapsto b_i], c\rangle$ $vs = vs +\!\!+ [v]$ $\sigma = \sigma[b_0 \mapsto (vs_0, \rho) ... b_i \mapsto (vs_i, \rho)]$
$\textbf{appf}(dn, h :: (t...), \rho', c)$	$\langle h, \rho', b\rangle$ $\sigma = \sigma[b \longmapsto \textbf{appf}(dn +\!\!+ [v], t, \rho', c)]$
$\textbf{iff}(e_t, e_f, \rho', c)$ if $v = \#\text{f}$	$\langle e_f, \rho', c\rangle$
$\textbf{iff}(e_t, e_f, \rho', c)$ if $v \neq \#\text{f}$	$\langle e_t, \rho', c\rangle$
$\textbf{letf}(x, e, \rho', c)$	$\langle e, \rho'[x \longmapsto b], c\rangle$ $\sigma = \sigma[b \longmapsto (v, \rho)]$