Single-Phase Model

November 4, 2015

$$
\begin{aligned}
\mathit{ast} &::= \mathit{var} \mid \textbf{App}(\mathit{ast}, \mathit{ast}, ...) \mid \mathit{val} \\
\mathit{var} &::= \textbf{Var}(\mathit{name}) \\
\mathit{val} &::= \textbf{Fun}(\mathit{var}, \mathit{ast}) \mid \mathit{atom} \mid \textbf{List}(\mathit{val}, ...) \mid \mathit{stx} \\
\mathit{stx} &::= \textbf{Stx}(\mathit{atom}, \mathit{ctx}) \mid \textbf{Stx}(\textbf{List}(\mathit{stx}, ...), \mathit{ctx}) \\
\mathit{id} &::= \textbf{Stx}(\mathit{sym}, \mathit{ctx}) \\
\mathit{ctx} &::= \overline{\mathit{scp}} \\
\overline{\mathit{scp}} &::= \{\mathit{scp}, ...\} \\
\mathit{atom} &::= \mathit{sym} \mid \mathit{prim} \mid .... \\
\mathit{sym} &::= \text{'}\mathit{name} \\
\mathit{prim} &::= \texttt{stx-e} \mid \texttt{mk-stx} \mid .... \\
\xi &::= \text{a mapping from } \mathit{name} \text{ to } \mathit{transform} \\
\mathit{transform} &::= \texttt{lambda} \mid \texttt{let-syntax} \mid \texttt{quote} \mid \texttt{syntax} \mid \textsc{Var}(\mathit{id}) \mid \mathit{val} \\
\Sigma &::= \text{binding store}, \mathit{name} \to (\overline{\mathit{scp}} \to \mathit{name}) \\
\mathit{name} &::= \text{a token such as } \texttt{x}, \texttt{egg}, \text{ or } \texttt{lambda} \\
\mathit{scp} &::= \text{a token that represents a scope}
\end{aligned}
$$

$\mathsf{eval} : \mathit{ast} \to \mathit{val}$

$$
\begin{aligned}
\mathsf{eval}[\![\textbf{App}(\mathit{ast}_{fun}, \mathit{ast}_{arg})]\!] &= \mathsf{eval}[\![\mathit{ast}_{body}[\mathit{var} \leftarrow \mathsf{eval}[\![\mathit{ast}_{arg}]\!]]]\!] \\
\text{subject to } \mathsf{eval}[\![\mathit{ast}_{fun}]\!] &= \textbf{Fun}(\mathit{var}, \mathit{ast}_{body}) \\
\mathsf{eval}[\![\textbf{App}(\mathit{prim}, \mathit{ast}_{arg}, ...)]\!] &= \delta(\mathit{prim}, \mathsf{eval}[\![\mathit{ast}_{arg}]\!], ...) \\
\mathsf{eval}[\![\mathit{val}]\!] &= \mathit{val}
\end{aligned}
$$

$$
\begin{aligned}
\delta(\texttt{stx-e}, \textbf{Stx}(\mathit{val}, \mathit{ctx})) &= \mathit{val} \\
\delta(\texttt{mk-stx}, \mathit{atom}, \textbf{Stx}(\mathit{val}, \mathit{ctx})) &= \textbf{Stx}(\mathit{atom}, \mathit{ctx}) \\
\delta(\texttt{mk-stx}, \textbf{List}(\mathit{stx}, ...), \textbf{Stx}(\mathit{val}, \mathit{ctx})) &= \textbf{Stx}(\textbf{List}(\mathit{stx}, ...), \mathit{ctx})
\end{aligned}
$$

$\text{parse} : stx\ \Sigma \rightarrow ast$

$$\text{parse}[\![\mathbf{STX}(\mathbf{LIST}(id_{lam}, id, stx_{body}), ctx), \Sigma]\!] = \mathbf{FUN}(\mathbf{VAR}(\text{resolve}[\![id, \Sigma]\!]), \text{parse}[\![stx_{body}, \Sigma]\!])$$

subject to $\text{resolve}[\![id_{lam}, \Sigma]\!] = \texttt{lambda}$

$$\text{parse}[\![\mathbf{STX}(\mathbf{LIST}(id_{quote}, stx), ctx), \Sigma]\!] = \text{strip}[\![stx]\!]$$

subject to $\text{resolve}[\![id_{quote}, \Sigma]\!] = \texttt{quote}$

$$\text{parse}[\![\mathbf{STX}(\mathbf{LIST}(id_{syntax}, stx), ctx), \Sigma]\!] = stx$$

subject to $\text{resolve}[\![id_{syntax}, \Sigma]\!] = \texttt{syntax}$

$$\text{parse}[\![\mathbf{STX}(\mathbf{LIST}(stx_{fun}, stx_{arg}, ...), ctx), \Sigma]\!] = \mathbf{APP}(\text{parse}[\![stx_{fun}, \Sigma]\!], \text{parse}[\![stx_{arg}, \Sigma]\!], ...)$$

$$\text{parse}[\![id, \Sigma]\!] = \mathbf{VAR}(\text{resolve}[\![id, \Sigma]\!])$$

$\text{resolve} : id\ \Sigma \rightarrow name$

$$\text{resolve}[\![\mathbf{STX}('name, ctx), \Sigma]\!] = name_{biggest}$$

subject to $\Sigma(name) = \{\overline{scp}_{bind} \leftarrow name_{bind}, ...\}$,
$\text{biggest-subset}[\![ctx, \{\overline{scp}_{bind}, ...\}]\!] = \overline{scp}_{biggest}$,
$\{\overline{scp}_{bind} \leftarrow name_{bind}, ...\}(\overline{scp}_{biggest}) = name_{biggest}$

$$\text{resolve}[\![\mathbf{STX}('name, ctx), \Sigma]\!] = name$$

$\text{biggest-subset} : \overline{scp}\ \{\overline{scp}, ...\} \rightarrow \overline{scp}$

$$\text{biggest-subset}[\![\overline{scp}_{ref}, \{\overline{scp}_{bind}, ...\}]\!] = \overline{scp}_{biggest}$$

subject to $\overline{scp}_{biggest} \subseteq \overline{scp}_{ref}$, $\overline{scp}_{biggest} \in \{\overline{scp}_{bind}, ...\}$,
$\overline{scp}_{bind} \subseteq \overline{scp}_{ref} \Rightarrow \overline{scp}_{bind} \subseteq \overline{scp}_{biggest}$

$\text{strip} : stx \rightarrow val$

$$\text{strip}[\![\mathbf{STX}(atom, ctx)]\!] = atom$$

$$\text{strip}[\![\mathbf{STX}(\mathbf{LIST}(stx, ...), ctx)]\!] = \mathbf{LIST}(\text{strip}[\![stx]\!], ...)$$

expand : $stx\ \xi\ \Sigma \to \langle stx, \Sigma \rangle$

$$\text{expand}[\![\textbf{Stx}(\textbf{List}(id_{lam}, id_{arg}, stx_{body}), ctx), \xi, \Sigma]\!] = \langle \textbf{Stx}(\textbf{List}(id_{lam}, id_{new}, stx_{body2}), ctx), \Sigma_4 \rangle$$
$$\text{subject to } \text{resolve}[\![id_{lam}, \Sigma]\!] = \texttt{lambda}, \text{alloc-name}[\![\Sigma]\!] = \langle name_{new}, \Sigma_1 \rangle,$$
$$\text{alloc-scope}[\![\Sigma_1]\!] = \langle scp_{new}, \Sigma_2 \rangle, \text{add}[\![id_{arg}, scp_{new}]\!] = id_{new},$$
$$\Sigma_2 + \{id_{new} \to name_{new}\} = \Sigma_3, \xi + \{name_{new} \to \text{VAR}(id_{new})\} = \xi_{new},$$
$$\text{expand}[\![\text{add}[\![stx_{body}, scp_{new}]\!], \xi_{new}, \Sigma_3]\!] = \langle stx_{body2}, \Sigma_4 \rangle$$

$$\text{expand}[\![\textbf{Stx}(\textbf{List}(id_{quote}, stx), ctx), \xi, \Sigma]\!] = \langle \textbf{Stx}(\textbf{List}(id_{quote}, stx), ctx), \Sigma \rangle$$
$$\text{subject to } \text{resolve}[\![id_{quote}, \Sigma]\!] = \texttt{quote}$$

$$\text{expand}[\![\textbf{Stx}(\textbf{List}(id_{syntax}, stx), ctx), \xi, \Sigma]\!] = \langle \textbf{Stx}(\textbf{List}(id_{syntax}, stx), ctx), \Sigma \rangle$$
$$\text{subject to } \text{resolve}[\![id_{syntax}, \Sigma]\!] = \texttt{syntax}$$

$$\text{expand}[\![\textbf{Stx}(\textbf{List}(id_{ls}, id, stx_{rhs}, stx_b), ctx), \xi, \Sigma]\!] = \text{expand}[\![stx_{b2}, \xi_2, \Sigma_3]\!]$$
$$\text{subject to } \text{resolve}[\![id_{ls}, \Sigma]\!] = \texttt{let-syntax}, \text{alloc-name}[\![\Sigma]\!] = \langle name_{new}, \Sigma_1 \rangle,$$
$$\text{alloc-scope}[\![\Sigma_1]\!] = \langle scp_{new}, \Sigma_2 \rangle, \text{add}[\![id, scp_{new}]\!] = id_{new}, \Sigma_2 + \{id_{new} \to name_{new}\} = \Sigma_3,$$
$$\xi + \{name_{new} \to \text{eval}[\![\text{parse}[\![stx_{rhs}, \Sigma_3]\!]]\!]\} = \xi_2, \text{add}[\![stx_b, scp_{new}]\!] = stx_{b2}$$

$$\text{expand}[\![stx_{macapp}, \xi, \Sigma]\!] = \text{expand}[\![\text{flip}[\![stx_{exp}, scp_i]\!], \xi, \Sigma_2]\!]$$
$$\text{subject to } stx_{macapp} = \textbf{Stx}(\textbf{List}(id_{mac}, stx_{arg}, ...), ctx), \xi(\text{resolve}[\![id_{mac}, \Sigma]\!]) = val,$$
$$\text{alloc-scope}[\![\Sigma]\!] = \langle scp_u, \Sigma_1 \rangle, \text{alloc-scope}[\![\Sigma_1]\!] = \langle scp_i, \Sigma_2 \rangle,$$
$$\text{eval}[\![\textbf{App}(val, \text{flip}[\![\text{add}[\![stx_{macapp}, scp_u]\!], scp_i]\!])]\!] = stx_{exp}$$

$$\text{expand}[\![\textbf{Stx}(\textbf{List}(stx_{fun}, stx_{arg}, ...), ctx), \xi, \Sigma]\!] = \langle \textbf{Stx}(\textbf{List}(stx_{fun2}, stx_{arg2}, ...), ctx), \Sigma_1 \rangle$$
$$\text{subject to } \text{expall}[\![(), (stx_{fun}\ stx_{arg}\ ...), \xi, \Sigma]\!] = \langle (stx_{fun2}\ stx_{arg2}\ ...), \Sigma_1 \rangle$$

$$\text{expand}[\![id, \xi, \Sigma]\!] = \langle id_{new}, \Sigma \rangle$$
$$\text{subject to } \xi(\text{resolve}[\![id, \Sigma]\!]) = \text{VAR}(id_{new})$$

expall : $(stx\ ...)\ (stx\ ...)\ \xi\ \Sigma \to \langle (stx\ ...), \Sigma \rangle$

$$\text{expall}[\![(stx_e\ ...), (), \xi, \Sigma]\!] = \langle (stx_e\ ...), \Sigma \rangle$$

$$\text{expall}[\![(stx_e\ ...), (stx_0\ stx_1\ ...), \xi, \Sigma]\!] = \text{expall}[\![(stx_e\ ...\ stx_{e0}), (stx_1\ ...), \xi, \Sigma_1]\!]$$
$$\text{subject to } \text{expand}[\![stx_0, \xi, \Sigma]\!] = \langle stx_{e0}, \Sigma_1 \rangle$$

add : $stx\ scp \to stx$

$$\text{add}[\![\textbf{Stx}(atom, ctx), scp]\!] = \textbf{Stx}(atom, \{scp\} \cup ctx)$$
$$\text{add}[\![\textbf{Stx}(\textbf{List}(stx, ...), ctx), scp]\!] = \textbf{Stx}(\textbf{List}(\text{add}[\![stx, scp]\!], ...), \{scp\} \cup ctx)$$

flip : $stx\ scp \to stx$

$$\text{flip}[\![\textbf{Stx}(atom, ctx), scp]\!] = \textbf{Stx}(atom, scp \oplus ctx)$$
$$\text{flip}[\![\textbf{Stx}(\textbf{List}(stx, ...), ctx), scp]\!] = \textbf{Stx}(\textbf{List}(\text{flip}[\![stx, scp]\!], ...), scp \oplus ctx)$$