Multi-Phase Model

July 14, 2015

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

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

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

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

parse : $ph\ stx\ \Sigma \to ast$

$$\mathsf{parse}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{lambda}, id_{arg}, stx_{body}), ctx), \Sigma]\!] = \mathbf{Fun}(\mathbf{Var}(\mathsf{resolve}_{ph}[\![id_{arg}, \Sigma]\!]), \mathsf{parse}_{ph}[\![stx_{body}, \Sigma]\!])$$
$$\text{subject to } \mathsf{resolve}_{ph}[\![id_{lambda}, \Sigma]\!] = \texttt{lambda}$$

$$\mathsf{parse}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{quote}, stx), ctx), \Sigma]\!] = \mathsf{strip}[\![stx]\!]$$
$$\text{subject to } \mathsf{resolve}_{ph}[\![id_{quote}, \Sigma]\!] = \texttt{quote}$$

$$\mathsf{parse}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{syntax}, stx), ctx), \Sigma]\!] = stx$$
$$\text{subject to } \mathsf{resolve}_{ph}[\![id_{syntax}, \Sigma]\!] = \texttt{syntax}$$

$$\mathsf{parse}_{ph}[\![\mathbf{Stx}(\mathbf{List}(stx_{rator}, stx_{rand}, ...), ctx), \Sigma]\!] = \mathbf{App}(\mathsf{parse}_{ph}[\![stx_{rator}, \Sigma]\!], \mathsf{parse}_{ph}[\![stx_{rand}, \Sigma]\!], ...)$$

$$\mathsf{parse}_{ph}[\![id, \Sigma]\!] = \mathbf{Var}(\mathsf{resolve}_{ph}[\![id, \Sigma]\!])$$

resolve : $ph\ id\ \Sigma \to name$

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

$$\mathsf{resolve}_{ph}[\![\mathbf{Stx}('name, ctx), \Sigma]\!] = name$$

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

$$\mathsf{biggest\text{-}subset}[\![\overline{scp}_{ref}, \{\overline{scp}_{bind}, ...\}]\!] = \overline{scp}_{biggest}$$
$$\text{subject to } \overline{scp}_{biggest} \subseteq \overline{scp}_{ref}, \overline{scp}_{biggest} \in \{\overline{scp}_{bind}, ...\},$$
$$\overline{scp}_{bind} \subseteq \overline{scp}_{ref} \Rightarrow \overline{scp}_{bind} \subseteq \overline{scp}_{biggest}$$

strip : $stx \to val$

$$\mathsf{strip}[\![\mathbf{Stx}(atom, ctx)]\!] = atom$$
$$\mathsf{strip}[\![\mathbf{Stx}(\mathbf{List}(stx, ...), ctx)]\!] = \mathbf{List}(\mathsf{strip}[\![stx]\!], ...)$$

expand : $ph\ stx\ \xi\ \overline{scp}\ \Sigma \to \langle stx, \Sigma \rangle$

$\mathsf{expand}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{lam}, id_{arg}, stx_{body}), ctx), \xi, \overline{scp}_p, \Sigma]\!] = \langle \mathbf{Stx}(\mathbf{List}(id_{lam}, id_{new}, stx_{body2}), ctx), \Sigma_4 \rangle$
subject to $\mathsf{resolve}_{ph}[\![id_{lam}, \Sigma]\!] = \texttt{lambda}$, $\mathsf{alloc\text{-}name}[\![\Sigma]\!] = \langle name_{new}, \Sigma_1 \rangle$,
$\mathsf{alloc\text{-}scope}[\![\Sigma_1]\!] = \langle scp_{new}, \Sigma_2 \rangle$, $\mathsf{add}_{ph}[\![id_{arg}, scp_{new}]\!] = id_{new}, \Sigma_2 + \{id_{new} \to name_{new}\} = \Sigma_3$,
$\xi + \{name_{new} \to \textsc{Var}(id_{new})\} = \xi_{new}$,
$\mathsf{expand}_{ph}[\![\mathsf{add}_{ph}[\![stx_{body}, scp_{new}]\!], \xi_{new}, \{scp_{new}\} \cup \overline{scp}_p, \Sigma_3]\!] = \langle stx_{body2}, \Sigma_4 \rangle$

$\mathsf{expand}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{quote}, stx), ctx), \xi, \overline{scp}_p, \Sigma]\!] = \langle \mathbf{Stx}(\mathbf{List}(id_{quote}, stx), ctx), \Sigma \rangle$
subject to $\mathsf{resolve}_{ph}[\![id_{quote}, \Sigma]\!] = \texttt{quote}$

$\mathsf{expand}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{syntax}, stx), ctx), \xi, \overline{scp}_p, \Sigma]\!] = \langle \mathbf{Stx}(\mathbf{List}(id_{syntax}, stx_{pruned}), ctx), \Sigma \rangle$
subject to $\mathsf{resolve}_{ph}[\![id_{syntax}, \Sigma]\!] = \texttt{syntax}$, $\mathsf{prune}_{ph}[\![stx, \overline{scp}_p]\!] = stx_{pruned}$

$\mathsf{expand}_{ph}[\![\mathbf{Stx}(\mathbf{List}(id_{ls}, id, stx_{rhs}, stx_{body}), ctx), \xi, \overline{scp}_p, \Sigma]\!] = \mathsf{expand}_{ph}[\![stx_{body2}, \xi_2, \overline{scp}_{p2}, \Sigma_4]\!]$
subject to $\mathsf{resolve}_{ph}[\![id_{ls}, \Sigma]\!] = \texttt{let-syntax}$, $\mathsf{alloc\text{-}name}[\![\Sigma]\!] = \langle name_{new}, \Sigma_1 \rangle$,
$\mathsf{alloc\text{-}scope}[\![\Sigma_1]\!] = \langle scp_{new}, \Sigma_2 \rangle$, $\mathsf{add}_{ph}[\![id, scp_{new}]\!] = id_{new}, \Sigma_2 + \{id_{new} \to name_{new}\} = \Sigma_3$,
$\mathsf{expand}_{ph+1}[\![stx_{rhs}, \xi_{primitives}, \emptyset, \Sigma_3]\!] = \langle stx_{exp}, \Sigma_4 \rangle$,
$\xi + \{name_{new} \to \mathsf{eval}[\![\mathsf{parse}_{ph+1}[\![stx_{exp}, \Sigma_4]\!]]\!]\} = \xi_2$, $\mathsf{add}_{ph}[\![stx_{body}, scp_{new}]\!] = stx_{body2}$,
$\{scp_{new}\} \cup \overline{scp}_p = \overline{scp}_{p2}$

$\mathsf{expand}_{ph}[\![stx_{macapp}, \xi, \overline{scp}_p, \Sigma]\!] = \mathsf{expand}_{ph}[\![\mathsf{flip}_{ph}[\![stx_{exp}, scp_i]\!], \xi, \{scp_u\} \cup \overline{scp}_p, \Sigma_3]\!]$
subject to $stx_{macapp} = \mathbf{Stx}(\mathbf{List}(id_{mac}, stx_{arg}, ...), ctx)$, $\xi(\mathsf{resolve}_{ph}[\![id_{mac}, \Sigma]\!]) = val$,
$\mathsf{alloc\text{-}scope}[\![\Sigma]\!] = \langle scp_u, \Sigma_2 \rangle$, $\mathsf{alloc\text{-}scope}[\![\Sigma_2]\!] = \langle scp_i, \Sigma_3 \rangle$,
$\mathsf{eval}[\![\mathbf{App}(val, \mathsf{flip}_{ph}[\![\mathsf{add}_{ph}[\![stx_{macapp}, scp_u]\!], scp_i]\!])]\!] = stx_{exp}$

$\mathsf{expand}_{ph}[\![\mathbf{Stx}(\mathbf{List}(stx_{rtor}, stx_{rnd}, ...), ctx), \xi, \overline{scp}_p, \Sigma]\!] = \langle \mathbf{Stx}(\mathbf{List}(stx_{exprtor}, stx_{exprnd}, ...), ctx), \Sigma_1 \rangle$
subject to $\mathsf{expand}^{\star}_{ph}[\![(), (stx_{rtor}\ stx_{rnd}\ ...), \xi, \overline{scp}_p, \Sigma]\!] = \langle (stx_{exprtor}\ stx_{exprnd}\ ...), \Sigma_1 \rangle$

$\mathsf{expand}_{ph}[\![id, \xi, \overline{scp}_p, \Sigma]\!] = \langle id_{new}, \Sigma \rangle$
subject to $\xi(\mathsf{resolve}_{ph}[\![id, \Sigma]\!]) = \textsc{Var}(id_{new})$

expand* : $ph\ (stx\ ...)\ (stx\ ...)\ \xi\ \overline{scp}\ \Sigma \to \langle (stx\ ...), \Sigma \rangle$

$\mathsf{expand}^{\star}_{ph}[\![(stx_{done}\ ...), (), \xi, \overline{scp}_p, \Sigma]\!] = \langle (stx_{done}\ ...), \Sigma \rangle$

$\mathsf{expand}^{\star}_{ph}[\![(stx_{done}\ ...), (stx_0\ stx_1\ ...), \xi, \overline{scp}_p, \Sigma]\!] = \mathsf{expand}^{\star}_{ph}[\![(stx_{done}\ ...\ stx_{done0}), (stx_1\ ...), \xi, \overline{scp}_p, \Sigma_1]\!]$
subject to $\mathsf{expand}_{ph}[\![stx_0, \xi, \overline{scp}_p, \Sigma]\!] = \langle stx_{done0}, \Sigma_1 \rangle$

prune : $ph\ stx\ \overline{scp} \to stx$

$\mathsf{prune}_{ph}[\![\mathbf{Stx}(atom, ctx), \overline{scp}_p]\!] = \mathbf{Stx}(atom, ctx + \{ph \to ctx(ph) \setminus \overline{scp}_p\})$
$\mathsf{prune}_{ph}[\![\mathbf{Stx}(\mathbf{List}(stx, ...), ctx), \overline{scp}_p]\!] = \mathbf{Stx}(\mathbf{List}(stx_{pruned}, ...), ctx + \{ph \to ctx(ph) \setminus \overline{scp}_p\})$
subject to $\mathsf{prune}_{ph}[\![stx, \overline{scp}_p]\!], ... = stx_{pruned}, ...$

add : $ph\ stx\ scp \to stx$

$\mathsf{add}_{ph}[\![\mathbf{Stx}(atom, ctx), scp]\!] = \mathbf{Stx}(atom, ctx + \{ph \to \{scp\} \cup ctx(ph)\})$
$\mathsf{add}_{ph}[\![\mathbf{Stx}(\mathbf{List}(stx, ...), ctx), scp]\!] = \mathbf{Stx}(\mathbf{List}(\mathsf{add}_{ph}[\![stx, scp]\!], ...), ctx + \{ph \to \{scp\} \cup ctx(ph)\})$

$\mathsf{flip} : \mathit{ph}\ \mathit{stx}\ \mathit{scp} \to \mathit{stx}$

$$\mathsf{flip}_{ph}[\![\textbf{Stx}(\mathit{atom}, \mathit{ctx}), \mathit{scp}]\!] = \textbf{Stx}(\mathit{atom}, \mathit{ctx}+\{\mathit{ph} \to \mathit{scp} \oplus \mathit{ctx}(\mathit{ph})\})$$

$$\mathsf{flip}_{ph}[\![\textbf{Stx}(\textbf{List}(\mathit{stx}, ...), \mathit{ctx}), \mathit{scp}]\!] = \textbf{Stx}(\textbf{List}(\mathsf{flip}_{ph}[\![\mathit{stx}, \mathit{scp}]\!], ...), \mathit{ctx}+\{\mathit{ph} \to \mathit{scp} \oplus \mathit{ctx}(\mathit{ph})\})$$