Midi-FreshML Dynamic Semantics

With delayed permutations

16/02/2014

A is a list of used atoms (name values).
E is a stack of environments (an environment is a list of $(id,\ val)$ pairs).
$\in$ denotes list membership, and $dom(\mathrm{E}')$ is the list of all ids in E′.
F is a list of frame stacks.
It is assumed all expressions have been type checked prior to evaulation.

A program consists of a sequence of name and data type declarations and expressions.
Let e_i be the i^{th} top-level expression in the program, then:

$$\mathcal{EXP}[[],\ [],\ [],\ e_0] \longrightarrow^* \mathcal{SUCCESS}[\mathrm{A}_1,\ \mathrm{E}_1,\ v]$$
$$\mathcal{EXP}[\mathrm{A}_i,\ \mathrm{E}_i,\ [],\ e_i] \longrightarrow^* \mathcal{SUCCESS}[\mathrm{A}_{i+1},\ \mathrm{E}_{i+1},\ v]$$
$$\mathcal{EXP}[\mathrm{A}_i,\ \mathrm{E}_i,\ [],\ e_i] \longrightarrow^* \mathcal{FAIL}$$

On success output the resultant value and evaluate the next expression.
On failure terminate evaluation and output an error message.

Id	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ \mathrm{F},\ x] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ \mathrm{F},\ \mathrm{cf}(\pi * v)] \iff (x,\ \pi * v) \in \mathrm{E}'\dagger$ $\mathcal{EXP}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ \mathrm{F},\ x] \longrightarrow \mathcal{FAIL} \iff (x,\ \_) \notin \mathrm{E}'$
Ctor	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ C\ e] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (C\ \_) :: \mathrm{F},\ e]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (C\ \_) :: \mathrm{F},\ v] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ C\ v]$
Fresh	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \mathrm{fresh} : N] \longrightarrow \mathcal{VAL}[a :: \mathrm{A},\ \mathrm{E},\ \mathrm{F},\ a] \iff a \notin \mathrm{A}$
If	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \mathrm{if}\ e_1\ \mathrm{then}\ e_2\ \mathrm{else}\ e_3] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\mathrm{if}\ \_\ \mathrm{then}\ e_2\ \mathrm{else}\ e_3) :: \mathrm{F},\ e_1]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\mathrm{if}\ \_\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2) :: \mathrm{F},\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e_1] \iff v = true$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\mathrm{if}\ \_\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2) :: \mathrm{F},\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e_2] \iff v = false$
Swap	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \mathrm{swap}\ (e_1,\ e_2)\ \mathrm{in}\ e_3] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\mathrm{swap}\ (\_,\ e_2)\ \mathrm{in}\ e_3) :: \mathrm{F},\ e_1]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\mathrm{swap}\ (\_,\ e_1)\ \mathrm{in}\ e_2) :: \mathrm{F},\ a] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\mathrm{swap}\ (a,\ \_)\ \mathrm{in}\ e_2) :: \mathrm{F},\ e_2]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\mathrm{swap}\ (a_1,\ \_)\ \mathrm{in}\ e) :: \mathrm{F},\ a_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\mathrm{swap}\ (a_1,\ a_2)\ \mathrm{in}\ \_) :: \mathrm{F},\ e]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\mathrm{swap}\ (a_1,\ a_2)\ \mathrm{in}\ \_) :: \mathrm{F},\ v] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \mathrm{cf}([(a_1\ a_2)] * v)]$
Name Abs	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \ll e_1 \gg e_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\ll\_\gg e_2) :: \mathrm{F},\ e_1]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\ll\_\gg e) :: \mathrm{F},\ a] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\ll a \gg \_) :: \mathrm{F},\ e]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\ll a \gg \_) :: \mathrm{F},\ v] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \ll a \gg v]$
Value	$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ [],\ v] \longrightarrow \mathcal{SUCCESS}[\mathrm{A},\ \mathrm{E},\ v]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ (end\text{–}\lambda) :: \mathrm{F},\ v] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ v]$
Pair	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ (e_1,\ e_2)] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\_,\ e_2) :: \mathrm{F},\ e_1]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\_,\ e) :: \mathrm{F},\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (v,\ \_) :: \mathrm{F},\ e]$ $\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (v_1,\ \_) :: \mathrm{F},\ v_2] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ (v_1,\ v_2)]$
Fun	$\mathcal{EXP}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ \mathrm{F},\ \mathrm{fun}\ (x : t) \to e] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ \mathrm{F},\ \mathrm{fun}\ (x : t) \to e\ [\mathrm{E}']]$

BinOp

Let $\odot \in \{/,\ *,\ +,\ -,\ >,\ \geq,\ <,\ \leq,\ =,\hat{}\,\}$

$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e_1 \odot e_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\_ \odot e_2) :: \mathrm{F},\ e_1]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\_ \odot e) :: \mathrm{F},\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (v \odot \_) :: \mathrm{F},\ e]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (v_1 \odot \_) :: \mathrm{F},\ v_2] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ v_3] \iff v_3 = (\mathrm{cf}(v_1) \odot \mathrm{cf}(v_2))\ \ddagger$$

UnOp

$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \sim e] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\sim \_) :: \mathrm{F},\ e]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\sim \_) :: \mathrm{F},\ v] \longrightarrow \mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ -\mathrm{cf}(v)]$$

App

$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e_1\ e_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\_\ e_2) :: \mathrm{F},\ e_1]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\_\ e) :: \mathrm{F},\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (v\ \_) :: \mathrm{F},\ e]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (v_1\ \_) :: \mathrm{F},\ v_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ ((x,\ v_2) :: \mathrm{E}') :: \mathrm{E},\ (\mathit{end}\text{–}\lambda) :: \mathrm{F},\ e]$$
$$\iff v_1 = \text{fun } (x : t) \rightarrow e\ [\mathrm{E}']$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (v_1\ \_) :: \mathrm{F},\ v_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ ((f,\ v_1) :: (x,\ v_2) :: \mathrm{E}') :: \mathrm{E},\ (\mathit{end}\text{–}\lambda) :: \mathrm{F},\ e]$$
$$\iff v_1 = f(x : t_1) : t_2 = e\ [\mathrm{E}']$$

Match

$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \text{match } e \text{ with } \mathit{branch}] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\text{match } \_ \text{ with } \mathit{branch}) :: \mathrm{F},\ e]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ (\text{match } \_ \text{ with } \mid p \rightarrow e) :: \mathrm{F},\ v] \longrightarrow$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E}' :: \mathrm{E},\ [],\ (\text{let } p = \_ \text{ in } e) :: (\mathit{end}\text{–}\lambda) :: \mathrm{F},\ \mathit{false},\ v]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ (\text{match } \_ \text{ with } \mid p \rightarrow e \mid \mathit{branch}) :: \mathrm{F},\ \mathit{false},\ v] \longrightarrow$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E}' :: \mathrm{E},\ [(\mathit{branch},\ v)],\ (\text{let } p = \_ \text{ in } e) :: (\mathit{end}\text{–}\lambda) :: \mathrm{F},\ \mathit{false},\ v]$$

Let

$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \text{let } p = e_1 \text{ in } e_2] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\text{let } p = \_ \text{ in } e_2) :: \mathrm{F},\ e_1]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\text{let } p = \_ \text{ in } e) :: \mathrm{F},\ v] \longrightarrow \mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ [],\ (\text{let } p = \_ \text{ in } e) :: \mathrm{F},\ \mathit{false},\ v]$$
$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \text{let } f(x : t_1) : t_2 = e_1 \text{ in } e_2] \longrightarrow$$
$$\mathcal{EXP}[\mathrm{A},\ ((f,\ f(x : t_1) : t_2 = e_1\ [\mathrm{E}']) :: \mathrm{E}') :: \mathrm{E},\ \mathrm{F},\ e_2]$$

TopLet

$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \text{let } p = e] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ (\text{let } p = \_) :: \mathrm{F},\ e]$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\text{let } p = \_) :: \mathrm{F},\ v] \longrightarrow \mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ [],\ (\text{let } p = \_ \text{ in } v) :: \mathrm{F},\ \mathit{true},\ v]$$
$$\mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ \text{let } f(x : t_1) : t_2 = e] \longrightarrow \mathcal{EXP}[\mathrm{A},\ ((f,\ v) :: \mathrm{E}') :: \mathrm{E},\ \mathrm{F},\ v]$$
$$\iff v = f(x : t_1) : t_2 = e\ [\mathrm{E}']$$

Pattern

$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\text{let } \_ = \_ \text{ in } e) :: \mathrm{F},\ b,\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e] \quad \text{(don't care pattern)}$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ [],\ (\text{let } x = \_ \text{ in } e) :: \mathrm{F},\ b,\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ ((x,\ v) :: \mathrm{E}') :: \mathrm{E},\ \mathrm{F},\ e]$$

$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ (\text{let } l = \_ \text{ in } e) :: \mathrm{F},\ b,\ v] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e] \iff l \text{ is a literal} \land l = v$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ (\mathit{branch},\ v') :: [],\ (\text{let } l = \_ \text{ in } e) :: (\mathit{end}\text{–}\lambda) :: \mathrm{F},\ b,\ v] \longrightarrow$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\text{match } \_ \text{ with } \mathit{branch}) :: \mathrm{F},\ v'] \iff l \text{ is a literal} \land l \neq v$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\text{let } l = \_ \text{ in } e) :: \mathrm{F},\ b,\ v] \longrightarrow \mathcal{FAIL} \iff l \text{ is a literal} \land l \neq v$$

$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\text{let } C\ p = \_ \text{ in } e) :: \mathrm{F},\ b,\ C\ v] \longrightarrow$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\text{let } p = \_ \text{ in } e) :: \mathrm{F},\ b,\ v]$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ (\mathit{branch},\ v') :: [],\ (\text{let } C\ p = \_ \text{ in } e) :: (\mathit{end}\text{–}\lambda) :: \mathrm{F},\ b,\ C'\ v] \longrightarrow$$
$$\mathcal{VAL}[\mathrm{A},\ \mathrm{E},\ (\text{match } \_ \text{ with } \mathit{branch}) :: \mathrm{F},\ v'] \iff C \neq C'$$
$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ [],\ (\text{let } C\ p = \_ \text{ in } e) :: \mathrm{F},\ b,\ C'\ v] \longrightarrow \mathcal{FAIL} \iff C \neq C'$$

$$\mathcal{MATCH}[\mathrm{A},\ \mathrm{E}' :: \mathrm{E},\ \mathrm{M},\ (\text{let } \ll x \gg p = \_ \text{ in } e) :: \mathrm{F},\ b,\ \ll a \gg v] \longrightarrow$$
$$\mathcal{MATCH}[a' :: \mathrm{A},\ ((x,\ a') :: \mathrm{E}') :: \mathrm{E},\ \mathrm{M},\ (\text{let } p = \_ \text{ in } e') :: \mathrm{F},\ b,\ [(a\ a')] * v]$$
$$\iff a' \notin \mathrm{A} \land e' = \begin{cases} \ll a' \gg ([(a\ a')] * v) & \text{if } b = \mathit{true} \\ e & \text{if } b = \mathit{false} \end{cases}$$

$$
\begin{array}{l}
\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\mathrm{let}\ ()=\_\ \mathrm{in}\ e)::\mathrm{F},\ b,\ ()] \longrightarrow \mathcal{EXP}[\mathrm{A},\ \mathrm{E},\ \mathrm{F},\ e] \\
\mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\mathrm{let}\ (p_1,\ p_2)=\_\ \mathrm{in}\ e)::\mathrm{F},\ b,\ (v_1,\ v_2)] \longrightarrow \\
\qquad\qquad \mathcal{MATCH}[\mathrm{A},\ \mathrm{E},\ \mathrm{M},\ (\mathrm{let}\ p_1=\_\ \mathrm{in}\ (\mathrm{let}\ p_2=v_2\ \mathrm{in}\ e))::\mathrm{F},\ b,\ v_1]
\end{array}
$$

† The auxiliary function cf(-) takes a value with delayed permutation v_p and pushes the permutation through the first level of its structure, thus making the outermost constructor manifest. It is defined as follows:

$$\mathrm{cf}(\pi * l) \stackrel{\mathrm{def}}{=} l \iff l \text{ is an int, real, bool or string literal}$$

$$\mathrm{cf}(\pi * ()) \stackrel{\mathrm{def}}{=} ()$$

$$\mathrm{cf}(\pi * a) \stackrel{\mathrm{def}}{=} \pi(a) \ \dagger\dagger$$

$$\mathrm{cf}(\pi * C(v)) \stackrel{\mathrm{def}}{=} C(\pi * v)$$

$$\mathrm{cf}(\pi * (v,\ v')) \stackrel{\mathrm{def}}{=} (\pi * v,\ \pi * v')$$

$$\mathrm{cf}(\pi * \ll a \gg v) \stackrel{\mathrm{def}}{=} \ll \pi(a) \gg \pi * v$$

$$\mathrm{cf}(\pi * (\mathrm{fun}\ (x:t) \to e\ [\mathrm{E}])) \stackrel{\mathrm{def}}{=} \mathrm{fun}\ (x:t) \to e\ [(x,\ \pi * v) \mid (x,\ v) \in E]$$

$$\mathrm{cf}(\pi * (f\ (x:t_1):t_2 = e\ [\mathrm{E}]) \stackrel{\mathrm{def}}{=} (f\ (x:t_1):t_2 = e\ [(x,\ \pi * v) \mid (x,\ v) \in E]$$

$$[](a) \stackrel{\mathrm{def}}{=} a$$

$$(a_1\ a_2) :: \pi(a) \stackrel{\mathrm{def}}{=} \begin{cases} \pi(a_1) & \text{if } a = a_2 \\ \pi(a_2) & \text{if } a = a_1 \\ \pi(a) & \text{otherwise} \end{cases}$$

‡ In the case of = perform object-level α-equivalence:

$$v_1 = \ll a_1 \gg v$$

$$v_2 = \ll a_2 \gg v'$$

$$v_1 = v_2 \iff \mathrm{let}\ x = \mathrm{fresh} : a\ \mathrm{in}\ (\mathrm{swap}\ (x,\ a_1)\ \mathrm{in}\ v) = (\mathrm{swap}\ (x,\ a_2)\ \mathrm{in}\ v')$$

For all other values use structural equality.