Evaluation with environments

Jonathan Lam

2022/01/03

1 Motivation

Evaluation with substitution is not efficient because it forces the re-evaluation of the substituted expression every time it is encountered. A more efficient involves an environment model, where variable values are evaluated and stored in an environment when bound and looked-up when encountered.

(Is evaluation with substitution considered normal order evaluation? This seems similar to normal/applicative order evaluation described in SICP 1.1.5.)

Big step semantics

The irreducible judgment (for internal expressions) in Hazel is not $d\ \mathsf{val}$, but rather $d\ \mathsf{final}$. Thus, final expressions evaluate to themselves. Variables evaluate to the final value that they are bound to (assuming they are bound; otherwise they are free and thus final). Lambdas evaluate to a closure type. The evaluation of a let-expression or function application extends the current environment with the newly-bound variable. For function applications, the current environment is first extended with the closure environment before binding the new variable. When extending an environment ($E :: E'$ or $E, x \leftarrow d$), bindings on the right overwrite bindings on the left.

$\boxed{d \Downarrow d'}$ Internal expression d evaluates to d'

$$\frac{d \text{ final}}{E \vdash d \Downarrow d}\ \text{Eval-Final} \qquad \frac{}{E, x \leftarrow d \vdash x \Downarrow d}\ \text{Eval-Var}$$

$$\frac{}{E \vdash (\lambda x : \tau.d) \Downarrow [E](\lambda x : \tau.d)}\ \text{Eval-Lam}$$

$$\frac{d_2 \Downarrow d_2' \qquad E :: E', x \leftarrow d_2' \vdash d_1 \Downarrow d}{E \vdash [E'](\lambda x : \tau.d_1)(d_2) \Downarrow d}\ \text{Eval-Ap}$$

$$\frac{d_2 \Downarrow d_2' \qquad E, x \leftarrow d_2' \vdash d_1 \Downarrow d}{E \vdash \text{let } x = d_2 \text{ in } d_1 \Downarrow d}\ \text{Eval-Let}$$

The following metatheorem states that environments only include final terms.

Theorem 1 *If the variable binding $x \leftarrow d$ exists in E, then d final.*

$\boxed{\dot{e}\ \mathsf{value}}$ H-Expression $\dot{e}$ is a closed value

$$\frac{}{\underline{n}\ \mathsf{value}}\ \text{V-num} \qquad \frac{}{(\lambda x.\dot{e})\ \mathsf{value}}\ \text{V-lam}$$

Figure 1: Value forms

$\boxed{\dot{e}\ \mathsf{final}}$ H-Expression $\dot{e}$ is final

$$\frac{\dot{e}\ \mathsf{value}}{\dot{e}\ \mathsf{final}}\ \text{F-val} \qquad \frac{\dot{e}\ \mathsf{final}}{(\!|\dot{e}|\!)\ \mathsf{final}}\ \text{F-filled} \qquad \frac{}{(\!|\,|\!)\ \mathsf{final}}\ \text{F-unfilled} \qquad \frac{\dot{e}\ \mathsf{indet}}{\lceil\dot{e}\rceil\ \mathsf{final}}\ \text{F-indet}$$

Figure 2: Final forms

$\boxed{\dot{e}\ \mathsf{indet}}$ H-Expression $\dot{e}$ is indeterminate

$$\frac{\dot{e}_1\ \mathsf{final} \qquad \dot{e}_2\ \mathsf{final} \qquad \dot{e}_1 \neq \underline{\underline{n_1}}}{(e_1 + e_2)\ \mathsf{indet}}\ \text{I-plus}_1$$

$$\frac{\dot{e}_1\ \mathsf{final} \qquad \dot{e}_2\ \mathsf{final} \qquad \dot{e}_2 \neq \underline{\underline{n_2}}}{(\dot{e}_1 + \dot{e}_2)\ \mathsf{indet}}\ \text{I-plus}_2$$

$$\frac{\dot{e}_1\ \mathsf{final} \qquad \dot{e}_2\ \mathsf{final} \qquad \dot{e}_1 \neq (\lambda x.\dot{e}_1')}{\dot{e}_1(\dot{e}_2)\ \mathsf{indet}}\ \text{I-app}$$

Figure 3: Indeterminate forms

We extend the syntax of H-Expressions as follows:

$$\dot{e} ::= \cdots \mid \lceil\dot{e}\rceil$$

Here are some things that we want to prove:

Def (Ascription erasure).
$|\dot{e}|_{\mathsf{erase}}$ is the same as $\dot{e}$, but without its type ascriptions. All cases are congruences, except for the ascription case, where $|\dot{e} : \dot{\tau}|_{\mathsf{erase}} = \dot{e}$.

Def (Declarative typing).
The judgement $\dot{\Gamma} \vdash \dot{e} : \dot{\tau}$ is a type assignment system for erased terms. It is declarative, and unlike the bidirectional rules, is not algorithmic. We employ it to relate bidirectionally-typed terms to (erased) terms that enjoy type soundness with respect to the dynamics.

$\boxed{\dot{e}_1 \longrightarrow \dot{e}_2}$ H-Expression $\dot{e}_1$ steps to $\dot{e}_2$

$$\frac{\dot{e}_1 \longrightarrow \dot{e}_1'}{(\dot{e}_1 + \dot{e}_2) \longrightarrow (\dot{e}_1' + \dot{e}_2)}\ \text{S-plus}_1 \qquad \frac{\dot{e}_1\ \mathsf{final} \qquad \dot{e}_2 \longrightarrow \dot{e}_2'}{(\dot{e}_1 + \dot{e}_2) \longrightarrow (\dot{e}_1 + \dot{e}_2')}\ \text{S-plus}_2$$

$$\frac{n_1 + n_2 = n_3}{(\underline{n_1} + \underline{n_2}) \longrightarrow \underline{n_3}}\ \text{S-plus}_3 \qquad \frac{\dot{e}_1\ \mathsf{final} \qquad \dot{e}_2\ \mathsf{final} \qquad \big(\dot{e}_1 \neq \underline{n_1} \vee \dot{e}_2 \neq \underline{n_2}\big)}{(\dot{e}_1 + \dot{e}_2) \longrightarrow \lceil (\dot{e}_1 + \dot{e}_2) \rceil}\ \text{S-plus}_4$$

$$\frac{\dot{e}_1 \longrightarrow \dot{e}_1'}{\dot{e}_1(\dot{e}_2) \longrightarrow \dot{e}_1'(\dot{e}_2)}\ \text{S-ap}_1 \qquad \frac{\dot{e}_2 \longrightarrow \dot{e}_2' \qquad \dot{e}_1\ \mathsf{final}}{\dot{e}_1(\dot{e}_2) \longrightarrow \dot{e}_1(\dot{e}_2')}\ \text{S-ap}_2$$

$$\frac{\dot{e}_2\ \mathsf{final}}{(\lambda x.\dot{e}_1)(\dot{e}_2) \longrightarrow [\dot{e}_2/x]\dot{e}_1}\ \text{S-ap}_3 \qquad \frac{\dot{e}_1\ \mathsf{final} \qquad \dot{e}_2\ \mathsf{final} \qquad \dot{e}_1 \neq (\lambda x.\dot{e}_1')}{\dot{e}_1(\dot{e}_2) \longrightarrow \lceil \dot{e}_1(\dot{e}_2) \rceil}\ \text{S-ap}_4$$

Figure 4: Small-step operational semantics.

Conjecture (Bidirectional implies declarative).
(i) If $\dot{\Gamma} \vdash \dot{e} \Rightarrow \dot{\tau}$ then $\dot{\Gamma} \vdash |\dot{e}|_{\mathsf{erase}} : \dot{\tau}$.
(ii) If $\dot{\Gamma} \vdash \dot{e} \Leftarrow \dot{\tau}$ then $\dot{\Gamma} \vdash |\dot{e}|_{\mathsf{erase}} : \dot{\tau}$.

Conjecture (Substitution).
If $\dot{\Gamma}, x : \tau_x \vdash \dot{e} : \dot{\tau}$
and $\dot{e}'\ \mathsf{final}$ (do we actually need this condition?)
and $\dot{\Gamma} \vdash \dot{e}' : \dot{\tau}_x$
then $\dot{\Gamma} \vdash \dot{e}[\dot{e}'/x] : \dot{\tau}$

Conjecture (Canonical forms)
If $\cdot \vdash \dot{e} : \dot{\tau}$
and $\dot{e}\ \mathsf{final}$ then

- if $\dot{e} = \lceil \dot{e}' \rceil$ then $\dot{e}'\ \mathsf{indet}$ and $\cdot \vdash \dot{e}' : \dot{\tau}$
- else:
  - if $\dot{\tau} = \texttt{num}$ then exists $\underline{n}$ such that $\dot{e} = \underline{n}$.
  - else if $\dot{\tau} = (\dot{\tau}_1 \to \dot{\tau}_2)$ then exists x and $\dot{e}'$ such that $\dot{e} = (\lambda x.\dot{e}')$
  - else if $\dot{\tau} = \llparenthesis\rrparenthesis$ then either:
    - $\dot{e} = \llparenthesis\rrparenthesis$, or
    - exists $\dot{\tau}'$ and $\dot{e}'$ such that $\dot{e} = \llparenthesis \dot{e}' \rrparenthesis$, $\dot{e}'\ \mathsf{final}$ and $\cdot \vdash \dot{e}' : \dot{\tau}'$

Conjecture (Progresss).
If $\cdot \vdash \dot{e}_1 : \dot{\tau}$
then either $\dot{e}_1\ \mathsf{final}$
or exists $\dot{e}_2$ such that $\dot{e}_1 \longrightarrow \dot{e}_2$.

Conjecture (Preservation).
If $\cdot \vdash \dot{e}_1 : \dot{\tau}$
and $\dot{e}_1 \longrightarrow \dot{e}_2$
then $\cdot \vdash \dot{e}_2 : \dot{\tau}$