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 val, but rather d 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.*