Advice about Debugger Construction

Arthur Nunes-Harwitt

Nassau Community College
One Education Drive
Garden City, NY 11530

nunesa@ncc.edu
http://www.matcmp.ncc.edu/~nunesa/

Background

```
(define (interp exp env k)
  (cond ((constant? exp) (k exp))
        ((variable? exp) (k (lookup exp env)))
        ((quote? exp) (k (unquote exp)))
     ...))
```

Goals:
Implement debug commands step and next such that...

- they are efficient

- they are correct

```
position:  ▷(fact 5)
debug command> step
position:  ▷fact
debug command> step
position:  fact◁
debug command> step
position:  ▷5
debug command> step
position:  5◁
debug command> step
position:  ▷(if (= n 0) 1 (* n (fact (- n 1))))
debug command> step
position:  ▷(= n 0)
debug command> next
position:  (= n 0)◁
debug command> continue
value:  120
```

λ-calculus

$$
\begin{gathered}
c_{op} \in \textsf{Operators} \\
c \in \textsf{Constants} \supset \textsf{Operators} \\
x \in \textsf{Variables} \\
V \in \textsf{Values} ::= c \mid x \mid (\lambda x.M) \\
M, N \in \Lambda ::= V \mid (M\ M)
\end{gathered}
$$

$$
\mathcal{E} ::= [] \mid \mathcal{E}[(V\ [])] \mid \mathcal{E}[([]\ M)]
$$

Example:

$((\lambda x.x)$ (succ (succ 0)))

$\mathcal{E} = ((\lambda x.x)$ (succ []))

$\mathcal{E}$[(succ 0)] $\longrightarrow$ $\mathcal{E}$[1]

Specification

$$\mathcal{E}[\cdot(M\ \ N)] \xrightarrow{\text{step}} \mathcal{E}[(\cdot M\ \ N)] \quad \text{[step 1]}$$

$$\mathcal{E}[\cdot V] \xrightarrow{\text{step}} \mathcal{E}[V\cdot] \quad \text{[step 2]}$$

$$\mathcal{E}[(V\cdot\ \ M)] \xrightarrow{\text{step}} \mathcal{E}[(V\ \ \cdot M)] \quad \text{[step 3]}$$

$$\frac{(c_{op}\ \ c) \xrightarrow{\delta} V}{\mathcal{E}[(c_{op}\ \ c\cdot)] \xrightarrow{\text{step}} \mathcal{E}[V\cdot]} \quad \text{[step 4]}$$

$$\frac{((\lambda x.M)\ \ V) \xrightarrow{\beta_v} N}{\mathcal{E}[((\lambda x.M)\ \ V\cdot)] \xrightarrow{\text{step}} \mathcal{E}[\cdot N]} \quad \text{[step 5]}$$

$$\frac{M \xrightarrow{*} V}{\mathcal{E}[\cdot M] \xrightarrow{\text{next}} \mathcal{E}[V\cdot]} \quad \text{[next 1]}$$

$$\frac{(V_1\ \ V_2) \xrightarrow{*} V}{\mathcal{E}[(V_1\ \ V_2\cdot)] \xrightarrow{\text{next}} \mathcal{E}[V\cdot]} \quad \text{[next 2]}$$

$$\mathcal{E}[(V\cdot\ \ M)] \xrightarrow{\text{next}} \mathcal{E}[(V\ \ \cdot M)] \quad \text{[next 3]}$$

Theoretical Implementation (part 1)

CEK-machine

$$E \in \mathsf{Env} ::= [] \mid (x, (V, E)) :: E$$

$$K \in \mathcal{K} ::= \mathsf{k}_\emptyset \mid \mathsf{fun}(V, E, K) \mid \mathsf{arg}(M, E, K)$$

$$\mathsf{lookup}(x, (y, (V, E)) :: E') = \begin{cases} (V, E) & \text{if } x = y \\ \mathsf{lookup}(x, E') & \text{otherwise} \end{cases}$$

$$C \in \mathcal{C} ::= \langle M, E, K \rangle$$

$$\begin{array}{lll}
\langle (M\ \ N), E, K \rangle & \longmapsto & \\
\langle M, E, \mathsf{arg}(N, E, K) \rangle & & \\
\langle V, E, \mathsf{arg}(N, E', K) \rangle & \longmapsto & \\
\langle N, E', \mathsf{fun}(V, E, K) \rangle & & \text{if } V \notin \mathsf{Variables} \\
\langle c, E, \mathsf{fun}(c_{op}, E', K) \rangle & \longmapsto & \langle \delta(c_{op}, c), [], K \rangle \\
\langle V, E, \mathsf{fun}((\lambda x.M), E', K) \rangle & \longmapsto & \\
\langle M, (x, (V, E)) :: E', K \rangle & & \text{if } V \notin \mathsf{Variables} \\
\langle x, E, K \rangle & \longmapsto & \\
\langle \mathsf{lookup}(x, E)_1, \mathsf{lookup}(x, E)_2, K \rangle & &
\end{array}$$

Theoretical Implementation (part 2)

$$\langle V, E, K_d \rangle \longmapsto \langle V\cdot, E, K \rangle \quad \text{if } V \notin \text{Variables}$$

$$\langle \cdot(M\ \ N), E, K \rangle \overset{\text{step}}{\longmapsto} \langle \cdot M, E, \mathsf{arg}(N, E, K) \rangle$$

$$\langle \cdot V, E, K \rangle \overset{\text{step}}{\longmapsto} \langle V, E, K_d \rangle$$

$$\langle V\cdot, E, \mathsf{arg}(N, E', K) \rangle \overset{\text{step}}{\longmapsto} \langle \cdot N, E', \mathsf{fun}(V, E, K) \rangle$$

$$\langle c\cdot, E, \mathsf{fun}(c_{op}, E', K) \rangle \overset{\text{step}}{\longmapsto} \langle \delta(c_{op}, c)\cdot, [], K \rangle$$

$$\langle V\cdot, E, \mathsf{fun}((\lambda x.M), E', K) \rangle \overset{\text{step}}{\longmapsto} \langle \cdot M, (x, (V, E)) :: E', K \rangle$$

$$\langle \cdot M, E, K \rangle \overset{\text{next}}{\longmapsto} \langle M, E, K_d \rangle$$

$$\langle V\cdot, E, \mathsf{arg}(N, E', K) \rangle \overset{\text{next}}{\longmapsto} \langle \cdot N, E', \mathsf{fun}(V, E, K) \rangle$$

$$\langle c\cdot, E, \mathsf{fun}(c_{op}, E', K) \rangle \overset{\text{next}}{\longmapsto} \langle \delta(c_{op}, c)\cdot, [], K \rangle$$

$$\langle V\cdot, E, \mathsf{fun}((\lambda x.M), E', K) \rangle \overset{\text{next}}{\longmapsto} \langle M, (x, (V, E)) :: E', K_d \rangle$$

Correctness (of step)

$$\mathcal{E}[\cdot(M\ \ N)] \xrightarrow{\mathsf{step}} \mathcal{E}[(\cdot M\ \ N)]$$

$$\langle \cdot(M\ \ N), E, K\rangle \xmapsto{\mathsf{step}}$$
$$\langle \cdot M, E, \mathsf{arg}(N, E, K)\rangle$$

$$\mathcal{E}[\cdot V] \xrightarrow{\mathsf{step}} \mathcal{E}[V\cdot]$$

$$\langle \cdot V, E, K\rangle \xmapsto{\mathsf{step}} \langle V, E, K_d\rangle$$

$$\mathcal{E}[(V\cdot\ \ M)] \xrightarrow{\mathsf{step}} \mathcal{E}[(V\ \ \cdot M)]$$

$$\langle V\cdot, E, \mathsf{arg}(M, E', K)\rangle \xmapsto{\mathsf{step}}$$
$$\langle \cdot M, E', \mathsf{fun}(V, E, K)\rangle$$

$$\frac{(c_{op}\ \ c) \xrightarrow{\delta} V}{\mathcal{E}[(c_{op}\ \ c\cdot)] \xrightarrow{\mathsf{step}} \mathcal{E}[V\cdot]}$$

$$\langle c\cdot, E, \mathsf{fun}(c_{op}, E', K)\rangle \xmapsto{\mathsf{step}} \langle \delta(c_{op}, c)\cdot, [], K\rangle$$

$$\frac{((\lambda x.M)\ \ V) \xrightarrow{\beta_v} N}{\mathcal{E}[((\lambda x.M)\ \ V\cdot)] \xrightarrow{\mathsf{step}} \mathcal{E}[\cdot N]}$$

$$\langle V\cdot, E, \mathsf{fun}((\lambda x.M), E', K)\rangle \xmapsto{\mathsf{step}}$$
$$\langle \cdot M, (x, (V, E)) :: E', K\rangle$$

Correctness (of next)

$$\frac{M \xrightarrow{*} V}{\mathcal{E}[\cdot M] \xrightarrow{\text{next}} \mathcal{E}[V\cdot]}$$

$\langle \cdot M, E, K\rangle \quad \overset{\text{next}}{\longmapsto} \quad \langle M, E, K_d\rangle$

$$\frac{(V_1\ V_2) \xrightarrow{*} V}{\mathcal{E}[(V_1\ V_2\cdot)] \xrightarrow{\text{next}} \mathcal{E}[V\cdot]}$$

$\langle c\cdot, E, \mathsf{fun}(c_{op}, E', K)\rangle \quad \overset{\text{next}}{\longmapsto}$

$\langle \delta(c_{op}, c)\cdot, [], K\rangle$

$\langle V\cdot, E, \mathsf{fun}((\lambda x.M), E', K)\rangle \quad \overset{\text{next}}{\longmapsto}$

$\langle M, (x, (V, E)) :: E', K_d\rangle$

$\mathcal{E}[(V\cdot\ \ M)] \xrightarrow{\text{next}} \mathcal{E}[(V\ \ \cdot M)]$

$\langle V\cdot, E, \mathsf{arg}(M, E', K)\rangle \quad \overset{\text{next}}{\longmapsto}$

$\langle \cdot M, E', \mathsf{fun}(V, E, K)\rangle$

Practical Implementation (part 1)

RK-machine

$$\langle [], \mathsf{eval}(c, E, K)\rangle \longmapsto$$
$$\langle c :: [], K\rangle$$
$$\langle [], \mathsf{eval}(x, E, K)\rangle \longmapsto$$
$$\langle \mathsf{lookup}(x, E) :: [], K\rangle$$
$$\langle [], \mathsf{eval}((\lambda\vec{x}.M), E, K)\rangle \longmapsto$$
$$\langle \mathsf{closure}(\vec{x}, M, E) :: [], K\rangle$$
$$\langle [], \mathsf{eval}((\mathsf{if}\ \ M_0\ \ M_1\ \ M_2), E, K)\rangle \longmapsto$$
$$\langle [], \mathsf{eval}(M_0, E, \mathsf{if}(M_1, M_2, E, K))\rangle$$
$$\langle [], \mathsf{eval}((M_0\ \ M_1\ \ \ldots\ \ M_n), E, K)\rangle \longmapsto$$
$$\langle [], \mathsf{evalapp}([], [M_0, M_1, \ldots, M_n], E, K)\rangle$$
$$\langle \mathsf{true} :: [], \mathsf{if}(M_1, M_2, E, K)\rangle \longmapsto$$
$$\langle [], \mathsf{eval}(M_1, E, K)\rangle$$
$$\langle \mathsf{false} :: [], \mathsf{if}(M_1, M_2, E, K)\rangle \longmapsto$$
$$\langle [], \mathsf{eval}(M_2, E, K)\rangle$$
$$\langle [], \mathsf{evalapp}(c_{op} :: \vec{R}, [], E, K)\rangle \longmapsto$$
$$\langle \delta(c_{op}, \vec{R}) :: [], K\rangle$$
$$\langle [], \mathsf{evalapp}(\mathsf{closure}(\vec{x}, M, E) :: \vec{R}, [], E', K)\rangle \longmapsto$$
$$\langle [], \mathsf{eval}(M, (\vec{x}, \vec{R}) :: E, K)\rangle$$

Practical Implementation (part 2)

$$\langle V, E, K_d \rangle \longmapsto \langle V\cdot, E, K \rangle \quad \text{if } V \notin \text{Variables}$$

$$\langle R :: [], K_d \rangle \longmapsto \langle R\cdot :: [], K \rangle$$

$$\langle \cdot V, E, K \rangle \overset{\text{step}}{\longmapsto} \langle V, E, K_d \rangle$$

$$\langle [], \mathsf{eval}(\cdot V, E, K) \rangle \overset{\text{step}}{\longmapsto} \langle [], \mathsf{eval}(V, E, K_d) \rangle$$

$$\langle V\cdot, E, \mathsf{fun}((\lambda x.M), E', K) \rangle \overset{\text{step}}{\longmapsto} \langle \cdot M, (x, (V, E)) :: E', K \rangle$$

$$\langle R_n\cdot :: [], \mathsf{mevalapp}(\mathsf{closure}(\vec{x}, M, E) :: \vec{R}, [], E', K) \rangle \overset{\text{step}}{\longmapsto} \langle [], \mathsf{eval}(\cdot M, (\vec{x}, \vec{R}@[R_n]) :: E, K) \rangle$$

Practical Implementation (part 3)

```
(define (interp exp env k)
  (cond ((constant? exp)
         (apply-k k exp))
        ((quote? exp)
         (apply-k k (unquote exp)))
        ((variable? exp)
         (apply-k k (lookup exp env)))
        ((abstraction? exp)
         (apply-k k (make-closure env exp)))
        ((if? exp)
         (interp (if-test exp)
                 env
                 (make-if-cont exp env k)))
        ((application? exp)
         (interp (rator exp)
                 env
                 (make-app-cont exp env k)))
        (else (error "unknown expression"))))
```

Practical Implementation (part 4)

$$
\begin{array}{lcl}
\mathcal{I}[\![c]\!]\rho\kappa & = & \kappa c \\
\mathcal{I}[\![x]\!]\rho\kappa & = & \kappa\rho(x) \\
\mathcal{I}[\![(\lambda x.M)]\!]\rho\kappa & = & \kappa(\lambda\iota.(\lambda\kappa'.(\lambda v.\iota[\![M]\!]\rho[x \mapsto v]\kappa'))) \\
\mathcal{I}[\![(M\ \ N)]\!]\rho\kappa & = & \\
\multicolumn{3}{l}{\mathcal{I}[\![M]\!]\rho(\lambda m.\mathcal{I}[\![N]\!]\rho(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n)))} \\
\mathcal{I}[\![\cdot M]\!]\rho\kappa & = & \mathcal{B}[\![M]\!]\rho\kappa
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{B}[\![c]\!]\rho\kappa & \overset{\text{step}}{=} & \mathcal{A}c\kappa \\
\mathcal{B}[\![x]\!]\rho\kappa & \overset{\text{step}}{=} & \mathcal{A}\rho(x)\kappa \\
\mathcal{B}[\![(\lambda x.M)]\!]\rho\kappa & \overset{\text{step}}{=} & \\
\multicolumn{3}{l}{\mathcal{A}(\lambda\iota.(\lambda\kappa'.(\lambda v.\iota[\![M]\!]\rho[x \mapsto v]\kappa')))\kappa} \\
\mathcal{B}[\![(M\ \ N)]\!]\rho\kappa & \overset{\text{step}}{=} & \\
\multicolumn{3}{l}{\mathcal{B}[\![M]\!]\rho(\lambda m.\mathcal{I}[\![N]\!]\rho(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n)))} \\
\mathcal{B}[\![M]\!]\rho\kappa & \overset{\text{next}}{=} & \mathcal{I}[\![M]\!]\rho(\lambda v.\mathcal{A}v\kappa)
\end{array}
$$

$$\mathcal{A}v(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n)) \overset{\text{step}}{=}$$
$$((\lambda n.(((m\ \mathcal{B})\ \kappa)\ n))\ v)$$

$$\mathcal{A}v(\lambda m.\mathcal{I}[\![M]\!]\rho(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n))) \overset{\text{step}}{=}$$
$$((\lambda m.\mathcal{B}[\![M]\!]\rho(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n)))\ v)$$

$$\mathcal{A}v(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n)) \overset{\text{next}}{=}$$
$$((\lambda n.(((m\ \mathcal{I})\ (\lambda v'.\mathcal{A}v'\kappa))\ n))\ v)$$

$$\mathcal{A}v(\lambda m.\mathcal{I}[\![M]\!]\rho(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n))) \overset{\text{next}}{=}$$
$$((\lambda m.\mathcal{B}[\![M]\!]\rho(\lambda n.(((m\ \mathcal{I})\ \kappa)\ n)))\ v)$$

Advice about Debugger Construction

Arthur Nunes-Harwitt

Nassau Community College
One Education Drive
Garden City, NY 11530

nunesa@ncc.edu
http://www.matcmp.ncc.edu/~nunesa/