CS450

Structure of Higher Level Languages

Lecture 16: Evaluating expressions; variable arguments

Tiago Cogumbreiro

$$\frac{\blacktriangleright_{H_1}\ e \Downarrow_E v \quad \blacktriangleright_{H_2} \quad E \leftarrow [x := v] \blacktriangleright_{H_3}}{\blacktriangleright_{H_1}\ (\texttt{define}\ x\ e) \Downarrow_E \texttt{void} \blacktriangleright_{H_3}}$$

$$\frac{\blacktriangleright_{H_1}\ t_1 \Downarrow_E v_1 \quad \blacktriangleright_{H_2} \quad t_2 \Downarrow_E v_2 \blacktriangleright_{H_3}}{t_1; t_2 \Downarrow_E v_2}$$

$$\blacktriangleright_H\ v \Downarrow_E v \blacktriangleright_H$$

$$\blacktriangleright_H\ x \Downarrow_E E(x) \blacktriangleright_H$$

$$\blacktriangleright_H\ \lambda x.t \Downarrow_E (E, \lambda x.t) \blacktriangleright_H$$

$$\frac{\blacktriangleright_{H_1}\ e_f \Downarrow_E (E_f, \lambda x.t_b) \blacktriangleright_{H_2}\ e_a \Downarrow_E v_a \blacktriangleright_{H_3}\ E_b \leftarrow E_f + [x := v_a] \blacktriangleright_{H_4}\ t_b \Downarrow_{E_b} v_b \blacktriangleright_{H_5}}{\blacktriangleright_{H_1}\ (e_f\ e_a) \Downarrow_E v_b \blacktriangleright_{H_5}}$$

Notes

- Make sure `(d:eval-term)` handles expressions by calling `(d:eval-exp)`
- Make sure the case for `d:define?` returns the value `(d:void)` not `(void)`, not `d:void`, not `void`
- Make sure the case for `d:apply?` invokes `(d:eval-term)` when handling t_b

UMass Boston

Exercise

```
;; e1 ⇓E v1

;; E' ← E + [x := v1]

;; e2 ⇓E' v2
```

Exercise

```
;; e1 ⇓E v1
(define v1+mem1 (d:eval-exp mem env e1))
(define mem1 (eff-state v1+mem1))
(define v1 (eff-result v1+mem1))
;; E' ← E + [x := v1]
(define env2+mem2 (environ-push mem1 env y v1)
(define env2 (eff-result env2+mem2))
(define mem2 (eff-state env2+mem2))
;; e2 ⇓E' v2
(define v2+mem3 (d:eval-exp mem2 env2 e2))
(define mem3 (eff-state v2+mem3))
(define v2 (eff-result v2+mem3))
```

UMass Boston