5118014 Programming Language Theory

Ch 11. Boxes

Shin Hong

Mutation and Box

- Mutation is important because many programs can be implemented concisely and efficiently with mutable variables
- Mutable variables makes it difficult to reason about program behaviors as different parts of the program may interfere with each other
- A mutable memory is represented as a box which contains a single value and updates the containing value over time

BFAE: Syntax

$$e \ ::= \ \cdots \ | \ \mathrm{box}\, e \ | \ !\, e \ | \ e \coloneqq e \ | \ e \,;\, e$$

- creating a new box
- reading the value in a box
- changing the value in a box
- sequencing expressions

BFAE: Address and Store

$$v ::= \cdots \mid a \in Addr$$

$$M \in Sto = Addr \rightharpoonup V$$

- A store is the memory of a program that records the values of boxes
  - A box is given with a unique address
- Unlike an environment whose change is propagated to only its subexpressions, an updated of a store is visible to the entire remaining executions

BFAE: Semantics

$$\Rightarrow \subseteq Env \times Sto \times E \times V \times Sto$$

- $\sigma, M_1 \vdash e \Rightarrow v, M_2$ holds if and only if e evaluates to v while updating the store from M_1 to M_2 under σ
  - store-passing style semantics

BFAE: Semantics on Box

$$\frac{\sigma, M \vdash e \Rightarrow v, M_1 \qquad a \notin Domain(M_1)}{\sigma, M \vdash \textsf{box}\ e \Rightarrow a, M_1[a \mapsto v]} \ [\textsc{NewBox}]$$

$$\frac{\sigma, M \vdash e \Rightarrow a, M_1 \qquad a \in Domain(M_1)}{\sigma, M \vdash !e \Rightarrow M_1(a), M_1} \ [\textsc{OpenBox}]$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow a, M_1 \qquad \sigma, M_1 \vdash e_2 \Rightarrow v, M_2}{\sigma, M \vdash e_1 := e_2 \Rightarrow v, M_2[a \mapsto v]} \ [\textsc{SetBox}]$$

BFAE: Sequencing and Application

$$\frac{\sigma, M \vdash e_1 \Rightarrow v_1, M_1 \qquad \sigma, M_1 \vdash e_2 \Rightarrow v_2, M_2}{\sigma, M \vdash e_1 ; e_2 \Rightarrow v_2, M_2} \text{ [Seq]}$$

$$\frac{\sigma, M \vdash e_1 \Rightarrow \langle \lambda x.e, \sigma' \rangle, M_1 \qquad \sigma, M_1 \vdash e_2 \Rightarrow v', M_2 \qquad \sigma'[x \mapsto v'], M_2 \vdash e \Rightarrow v, M_3}{\sigma, M \vdash e_1\ e_2 \Rightarrow v, M_3} \text{ [App]}$$

- A sequencing or application expression itself does not modify a given store, but its subexpressions can do so

Examples

- val x = Box 1 in (val y = Box 2 in (x := y + x ; y = x ; !y))
- $(\lambda x. (\lambda y. x \coloneqq 8; !\, y)\ x)$ box 7