Harvard School of Engineering and Applied Sciences — CS 152: Programming Languages

Denotational Semantics; Lambda Calculus Basics Section and Practice Problems

Week 4: Tue Feb 13–Fri Feb 16, 2018

1 Denotational Semantics

(a) Give the denotational semantic for each of the following IMP programs. That is, express the meaning of each of the following programs as a function from stores to stores.

(i) $\mathsf{a} := \mathsf{b} + 5; \mathsf{a} := \mathsf{a} \times \mathsf{b}$

(ii) $\textbf{if}\ \mathsf{foo} < 0\ \textbf{then}\ \mathsf{bar} := \mathsf{foo} \times \mathsf{foo}\ \textbf{else}\ \mathsf{bar} := \mathsf{foo} \times \mathsf{foo} \times \mathsf{foo}$

(iii) $\mathsf{bar} := \mathsf{foo} \times \mathsf{foo}; \textbf{if}\ \mathsf{foo} < 0\ \textbf{then skip else}\ \mathsf{bar} := \mathsf{bar} \times \mathsf{foo}$
(Hint: the answer to this question should be the same function as the answer to 1(b)ii above. You may have written the function down differently, but it should be the same mathematical function.)

(iv) $\mathsf{a} := 0; \mathsf{b} = 0; \textbf{while}\ \mathsf{a} < 3\ \textbf{do}\ \mathsf{b} := \mathsf{b} + \mathsf{c}$

(b) Consider the following loop.

$$\textbf{while}\ \mathsf{foo} < 5\ \textbf{do}\ \mathsf{foo} := \mathsf{foo} + 1; \mathsf{bar} := \mathsf{bar} + 1$$

We will consider the denotational semantics of this loop.

(i) What is the denotational semantics of the loop guard $\mathsf{foo} < 5$? That is, what is the function $\mathcal{B}[\![\mathsf{foo} < 5]\!]$?

(ii) What is the denotational semantics of the loop body $\mathsf{foo} := \mathsf{foo} + 1; \mathsf{bar} := \mathsf{bar} + 1$? That is, what is the function $\mathcal{C}[\![\mathsf{foo} := \mathsf{foo} + 1; \mathsf{bar} := \mathsf{bar} + 1]\!]$?

(iii) Recall that the semantics of the loop is the fixed point of the following higher-order function F. (This is from Section 1.2 of Lecture 6, where we have provided a specific loop guard b and loop body c for the higher-order function $F_{b,c}$).

$$\begin{aligned}
F &: (\textbf{Store} \rightharpoonup \textbf{Store}) \to (\textbf{Store} \rightharpoonup \textbf{Store}) \\
F(f) &= \{(\sigma, \sigma) \mid (\sigma, \textbf{false}) \in \mathcal{B}[\![\mathsf{foo} < 5]\!]\} \ \cup \\
&\qquad \{(\sigma, \sigma') \mid (\sigma, \textbf{true}) \in \mathcal{B}[\![\mathsf{foo} < 5]\!] \wedge \\
&\qquad\qquad \exists \sigma''.\ ((\sigma, \sigma'') \in \mathcal{C}[\![\mathsf{foo} := \mathsf{foo} + 1; \mathsf{bar} := \mathsf{bar} + 1]\!] \wedge (\sigma'', \sigma') \in f)\}
\end{aligned}$$

That is, the semantics of the loop are:

$$\begin{aligned}
\mathcal{C}[\![\textbf{while}\ \mathsf{foo} < 5\ \textbf{do}\ \mathsf{foo} := \mathsf{foo} + 1; \mathsf{bar} := \mathsf{bar} + 1]\!] &= \bigcup_{i \geq 0} F^i(\emptyset) \\
&= \emptyset \cup F(\emptyset) \cup F(F(\emptyset)) \cup F(F(F(\emptyset))) \cup \ldots \\
&= \text{fix}(F)
\end{aligned}$$

Compute $F(\emptyset)$, $F(F(\emptyset))$, and $F(F(F(\emptyset)))$.
In general, what is the domain of the partial function $F^i(\emptyset)$? (Note that $F^i(\emptyset)$ is F applied to the empty set i times, e.g., $F^3(\emptyset)$ is $F(F(F(\emptyset)))$.)

2 Lambda Calculus Basics

(a) **Variable Bindings** Fully parenthesize each expression based on the standard parsing of λ-calculus expressions, i.e. you should parenthesize all applications and λ abstractions. Then, draw a $\boxed{\text{box}}$ around all binding occurrences of variables, underline all usage occurrence of variables, and circle all free variables. For each bound usage occurrence, *neatly* draw an arrow to indicate its corresponding binding occurrence. (You may also use other methods to indicate binding occurrences of variables, usage occurrences of variables, free variables, and which uses correspond to which bindings.)

- $\lambda a.\, z\ \lambda z.\, a\ y$
- $(\lambda z.\, z)\ \lambda b.\, b\ \lambda a.\, a\ a$
- $\lambda b.\, b\ \lambda a.\, a\ b$
- $\lambda z.\, z\ \lambda z.\, z\ z$
- $\lambda a.\, \lambda b.\, (\lambda a.\, a)\ \lambda b.\, a$
- $x\ \lambda x.\, \lambda x.\, x\ (\lambda x.\, x)$
- $y\ (\lambda y.\, y)(\lambda y.\, z)$

(b) **Alpha equivalence:** Which of these three lambda-calculus expressions are alpha equivalent?

i. $\lambda x.\, y\ \lambda a.\, a\ x$
ii. $\lambda x.\, z\ \lambda b.\, b\ x$
iii. $\lambda a.\, y\ \lambda b.\, b\ a$

(Hint: to figure out whether two expressions are alpha equivalent, you need to know which variables are free and which variables are bound.)

(c) **Evaluation** For each of the following terms, do the following: (a) write the result of one step of the *call-by-value* reduction of the term; (b) write the result of one step of the *call-by-name* reduction of the term; and (c) write *all* possible results of one step under *full β-reduction*. If the term cannot take a step, please note that instead.

- $(\lambda z.\, \lambda x.\, x\ x)\ (\lambda y.\, y)$
- $\lambda a.\, \lambda b.\, (\lambda c.\, c)\ (\lambda d.\, d)$
- $(\lambda x.\, x\ x\ x\ x)\ (\lambda x.\, \lambda y.\, x\ y)$
- $(\lambda x.\, \lambda y.\, x\ y)\ ((\lambda w.\, \lambda z.\, w\ z)\ (\lambda x.\, x))$
- $(\lambda a.\, (\lambda b.\, b\ a)\ a)\ (\lambda z.\, z)\ (\lambda w.\, w)$

(d) Suppose we have an applied lambda calculus with integers and addition. Write the sequence of expressions that the following lambda calculus term evaluates to under call-by-value semantics. Then do the same under call-by-name semantics.

$$(\lambda f.\, f\ (f\ 8))\quad (\lambda x.\, x + 17)$$