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

Definitional translations; References and continuations (Lectures 9–10) Section and Practice Problems

Week 5

1 Definitional translations

Consider an applied lambda calculus with booleans, conjunction, a few constant natural numbers, and addition, whose syntax is defined as follows.

$$e ::= x \mid \lambda x.\, e \mid e_1\ e_2 \mid \mathsf{true} \mid \mathsf{false} \mid e_1 \ \mathsf{and}\ e_2 \mid 0 \mid 1 \mid 2 \mid e_1 + e_2 \mid \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3$$

Give a translation to the pure lambda calculus. Use the encodings of booleans and natural numbers that we considered in class. (You can assume that both the source and target languages have full-beta reduction semantics.)

2 Evaluation context

Consider the lambda calculus with let expressions and pairs (§1.3 of Lecture 9), and a semantics defined using evaluation contexts. For each of the following expressions, show one step of evaluation. Be clear about what the evaluation context is.

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

(b) $\mathsf{let}\ x = 5\ \mathsf{in}\ (\lambda y.\, y + x)\ 9$

(c) $(4, ((\lambda x.\, x)\ 8, 9))$

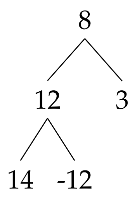
(d) $\mathsf{let}\ x = \#1\ ((\lambda y.\, y)\ (3, 4))\ \mathsf{in}\ x + 2$

3 References

(a) Evaluate the following program. (That is, show the sequence of configurations that the small-step evaluation of the program will take. The initial store should by $\emptyset$, i.e., the partial function with an empty domain.)

$$\mathsf{let}\ a = \mathsf{ref}\ 17\ \mathsf{in}\ \mathsf{let}\ b = \mathsf{ref}\ !a\ \mathsf{in}\ !b + (b := 8)$$

(b) Construct a program that represents the following binary tree, where an interior node of the binary tree is represented by a value of the form $(v, (\ell_{left}, \ell_{right}))$, where v is the value of the node, ℓ_{left} is a location that contains the left child, and ℓ_{right} is a location that contains the right child.



(It may be useful to define a function that creates internal nodes. Feel free to use let expressions to make your program easier to read and write.)

4 Continuations

(a) Suppose we add let expressions to our CBV lambda-calculus. How would you define $\mathcal{CPS}[\![\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2]\!]$? (Note, even though $\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$ is equivalent to $(\lambda x.\, e_2)\ e_1$, don't use $\mathcal{CPS}[\![(\lambda x.\, e_2)\ e_1]\!]$, as there is a better CPS translation of $\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$. Why is that?)

(b) Translate the expression $\mathsf{let}\ f = \lambda x.\, x + 1\ \mathsf{in}\ (f\ 19) + (f\ 21)$ into continuation-passing style. That is, what is $\mathcal{CPS}[\![\mathsf{let}\ f = \lambda x.\, x + 1\ \mathsf{in}\ (f\ 19) + (f\ 21)]\!]$?

(Use your definition of $\mathcal{CPS}[\![\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2]\!]$ from above.)