2013W1-lecture27

November 22, 2013

Contents

1 Question of the Day

What still needs to be done after `(web-read "Enter the first number:")` finishes evaluating in the following program?

```
(let (web-read (lambda (s)
                 (seq
                  (seq (display s) (display "\n"))
                  (read))))
  (seq
   (seq
    (display
```

```
   (+ (web-read "Enter the first number:")
      (web-read "Enter the second number:")))
  (display "\n"))
 ""))
```

In English:

1. We need to evaluate the second `web-read` call.
2. Add the results of the first `web-read` call and the second `web-read` call.
3. Display the result of the addition.
4. Display the endline character (the bottom expression that does that).
5. Evaluate the empty string.

What still needs to be done after `(read)` finishes evaluating in this program?

1.1 Solution

We can write a program to represent the answer to the first question:

```
(let (web-read (lambda (s)
                 (seq
                  (seq (display s) (display "\n"))
                  (read))))
  (lambda (it)
    (seq
     (seq
      (display
       (+ it
          (web-read "Enter the second number:")))
      (display "\n"))
     ""))))
```

However, we don't know what still needs to be done for the second program because we don't know statically where `(read)` was called from. So, instead, let's ask what still needs to be done after the second evaluation of `(read)`.

Then, we can write a program again. (Well, it's not **quite** a program. Why not?)

```
(lambda (it)
  (seq
   (seq
    (display (+ first-number
                it))
    (display "\n"))
   ""))
```

It's a perfectly good program if our `lambda` closed over the value of the first number. Something like:

```
(let (first-number 5)
  (lambda (it)
    (seq
     (seq
      (display (+ first-number
                  it))
      (display "\n"))
     "")))
```

2 Logistics

2.1 Final Project: Proposal resubmit due Friday

If you choose to resubmit (in which case be sure to submit a file with the appropriate name!), you'll be remarked and receive the new mark.

2.2 Final Project: Background Research Report due Friday

Be sure to AT LEAST easily clear the resubmit bar!

Better: submit something you think is good and get feedback on it from your facilitator!

2.3 Steve owes lambda calculus practice problems

3 Motivating Continuations

We call the answer to the QotD's "What still needs to be done after *some particular computation* finishes evaluating in a program?" a *continuation.*

Why would we **want** to think about continuations?

We'll switch over to code!

4 Converting to CPS

How in the world does `send/suspend` manage to "grab" the continuation? Let's figure it out.

In order to get there, we need to expose the continuation. Right now, we just rely on Racket (or Java or C++ or . . .) to keep track of the continuation for us. Let's see if we can instead make the continuation **explicit**. In other words, we're going to write programs that automatically generate the `lambda` forms representing the continuation

5 What have we learned today?

- Continuations
  - Justify the utility of "grabbing" everything that has to happen **after the current evaluation finishes** in a program?
    - For a callback: as in web programming.
  - Express the continuation of a particular program point as a function (to the extent that that's possible), an English description, and a portion of the runtime call tree.
- (If we get to it:) Continuation-Passing Style
  - Define tail call/tail position
    - A tail call is a function call (e.g., a call to `interp`) that represents "all work remaining to be done" in evaluating the current expression.
    - Evaluation of an expression in tail position within a syntactic form is always a tail call. (Note: sometimes a syntactic form has multiple of these, as in `if` expressions! Sometimes an expression will have **no** tail position, as in `display`.)
  - Define continuation-passing style
    - A program with the invariant that every (non-trivial, for some appropriate definition of non-trivial) computation is in tail position and where each non-trivial computation consumes a continuation: a function which, given the value of the current computation, performs all computations after the current computation.
  - Convert a simple program into continuation-passing style.