Problem Set 9
COMP301 Fall 2019
12.12.2019 17:30 - 18:45

Problem 1[1]: Extend the `letrec` language to allow the declaration of any number of mututally recursive unary procedures, for example:

```
letrec
   even(x) = if zero?(x) then 1 else (odd -(x,1))
   odd(x) = if zero?(x) then 0 else (even -(x,1))
in (odd 13)
```

evaluates to 1 because 13 is an odd number. (This question appeared in an earlier PS, but was left unsolved by majority due to time limit.)

Problem 2[2]: It is suggested in the EOPL book: the use of assignment to make a program more modular by allowing one procedure to communicate information to a distant procedure without requiring intermediate procedures to be aware of it. Very often such an assignment should only be temporary, lasting for the execution of a procedure call. Add to the language a facility for dynamic assignment (also called fluid binding) to accomplish this. Use the production:

$$\textit{Expression} ::= \texttt{setdynamic}\ \textit{Identifier}\ =\ \textit{Expression}\ \texttt{during}\ \textit{Expression}$$

$$\texttt{setdynamic-exp}\ (\textit{var}\ \textit{exp1}\ \textit{body})$$

The effect of the `setdynamic` expression is to assign temporarily the value of *exp1* to *var*, evaluate *body*, reassign *var* to its original value, and return the value of *body*. The variable *var* must already be bound. For example:

```
let x = 11
in let p = proc (y) -(y,x)
   in -(setdynamic x = 17 during (p 22), (p 13))
```

In this code,the value of `x` which is in `p`, is set to 17 in the call `(p 22)` but is reset to 11 in `(p 13)`, so the value of the expression is $5 - 2 = 3$.

Problem 3[3]: `Call-by-value-result` is a variation on `call-by-reference`. In `call-by-value-result`, the actual parameter must be a variable. When a parameter is passed, the formal parameter is bound to a new reference initialized to the value of the actual parameter, just as in `call-by-value`. The procedure body is then executed normally. When the procedure body returns, however, the value in the new reference is copied back into the reference denoted by the actual parameter. This may be more efficient than `call-by-reference` because it can improve memory locality. Implement `call-by-value-result` by modifying `call-by-reference`. In the actual exercise in the book, you are also asked to come up with a code that gives different results for each case. We have written that for you, check the bottom of the file. You will also refer to that to check if your program is correct. **NOTE:** We have put all necessary modules in a single file for this problem. This is done in order to test DMOJ at some point too.

---

[1] EOPL p.84-85 Exercise 3.32
[2] EOPL p.122 Exercise 4.21
[3] EOPL p.133 Exercise 4.37