Tarea 4: Ejercicios PROC

Daniel Rubén Ochoa Galván

October 12, 2022

Lenguajes de Programación

Exercise 3.20. [⋆] In PROC, procedures have only one argument, but one can get the effect of multiple argument procedures by using procedures that return other procedures. For example, one might write code like

```
let f = proc (x) proc (y) ...
in ((f 3) 4)
```

This trick is called *Currying*, and the procedure is said to be *Curried*. Write a Curried procedure that takes two arguments and returns their sum. You can write $x + y$ in our language by writing $-(x, -(0, y))$.

```
let suma = proc (x) proc (y) -(x, -(0, y)) ...
```

Exercise 3.27 [⋆] Add a new kind of procedure called a `traceproc` to the language. A `traceproc` works exactly like a `proc`, except that it prints a trace message on entry and on exit.
Extendiendo el lenguaje agregando a la sintaxis concreta:

$$\textit{Expression} ::= \texttt{traceproc}\ (\textit{Identifier})\ \textit{Expression}$$

Extendiendo el lenguaje agregando a la sintaxis abstracta:

$$(\text{traceproc-exp}\ \textit{var}\ \textit{body})$$

Especificando la semántica:

```
(value-of (proc-exp var body) env)
  = (proc-val (procedure var body env))
```

donde:

```
(apply-procedure (procedure var body env) val)
  = (value-of body [var = val] env)
```

Modificando la implementación de `apply-procedure` para integrar la especificación de `traceproc`:

```
(define (apply-procedure proc val)
  (unless (procedure? proc)
    (error 'value-of ”no es un procedimiento: ˜e” proc))
  (let ([var (procedure-var proc)]
        [body (procedure-body proc)]
        [saved-env (procedure-saved-env proc)])
    (cond
      [(traceproc-exp? proc)
       (display (”Entrando a proc...”))
       (value-of body (extend-env var val saved-env))
       (display (”Saliendo de proc...”))]
      [(proc-exp? proc)
       (value-of body (extend-env var val saved-env))])))
```