

Problemas EXPLICIT-REFS

Ricardo Payan

27 de octubre de 2022

Ejercicio 4.8.

Show exactly where in our implementation of the store these operations take linear time rather than constant time.

- En newref al usar length y append hacemos que la funcion trabaje con tiempo lineal.
- En deref, list-ref tiene tiempo de ejecucion lineal
- En setref!, setref-inner funciona con un ciclo, asi que setref! tiene tiempo de ejecucion lineal.

Ejercicio 4.9

Implement the store in constant time by representing it as a Scheme vector. What is lost by using this representation?

```
1 (define (empty-store)
2   (vector))
3
4 (define the-store 'uninitialized)
5
```

```

6  (define (get-store)
7    the-store)
8
9  (define reference?
10   (lambda (v)
11     (integer? v)))
12
13 (define (extend-store store val)
14   (let* ([store-size (vector-length store)]
15         [new-store (make-vector (+ store-size ↵
16                                   1) (vector-copy! store 0 store 0 ↵
17                                   (vector-length store)))]
18         (vector-set! new-store (vector-length (new-store) ↵
19                                   val))))
17
18 (define newref val
19   (let* ([new-store-info (extend-store the-store val)]
20         [new-store (first new-store-info)]
21         [next-ref (last new-store-info)])
22     (set! the-store new-store)
23     next-ref))
24
25 (define (deref ref)
26   (vector-ref the-store ref))
27
28 (define (setref! ref val)
29   (cond
30     [(and (reference? ref) (< ref (vector-length ↵
31                                   the-store)))] (vector-set! the-store ref val)]
31     [error 'setref! "No se puede cambiar la ↵
32             referencia"]]))

```

Ejercicio 4.10

Implement the begin expression as specified in exercise 4.4.

```

1  #Sintaxis Concreta
2    Expression := begin Expression {; Expression }* end
3
4  #Sintaxis Abstracta
5    begin-exp(exps)

```

```

1 #Implementacion
2 (value-of (begin-exp (exp1 (cons exp2 null))) env) =
3   ((value-of exp1 env)
4    (pair-val (cons (value-of exp2 env)
5                    (null-val))))

```

Ejercicio 4.11

Implement list from exercise 4.5.

```

1 #Sintaxis concreta
2   Expression := list (Expression *(,))
3
4 #Sintaxis Abstracta
5   list-exp(exps)

```

```

1 #Implementacion
2
3 (value-of (list-exp (cons exp1 (cons exp2 null))) env) =
4   (pair-val (cons (value-of exp1 env)
5                   (pair-val(cons (value-of exp2 env)
6                                 (null-val)))))

```
