Problemas EXPLICIT-REFS

Ricardo Payan
Chelsea Durazo

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?

```
(define (empty-store)
    (vector))

(define the-store 'uninitialized)
```

```

(define (get-store)
  the-store)

(define reference?
    (lambda (v)
        (integer? v)))

(define (extend-store store val)
    (let* ([store-size (vector-length store)]
                [new-store (make-vector (+ store-size ↩
                    1) (vector-copy! store 0 store 0 ↩
                    (vector-length store)))])
    (vector-set! new-store (vector-length (new-store) ↩
       val))))

(define newref val
    (let* ([new-store-info (extend-store the-store val)]
                    [new-store (first new-store-info)]
                    [next-ref (last new-store-info)])
        (set! the-store new-store)
            next-ref))

(define (deref ref)
(vector-ref the-store ref))

(define (setref! ref val)
    (cond
    [(and (reference? ref) (< ref (vector-length ↩
       the-store))) (vector-set! the-store ref val)]
    [error 'setref! "No se puede cambiar la ↩
       referencia"]))
```

Ejercicio 4.10

Implement the begin expression as specified in exercise 4.4.

```
#Sintaxis Concreta
    Expression := begin Expression {; Expresion }* end

#Sintaxis Abstracta
    begin-exp(exps)
```

```
#Implementacion
(value-of (begin-exp (exp1 (cons exp2 null))) env) =
    ((value-of exp1 env)
        (pair-val (cons (value-of exp2 env)
                                        (null-val))))
```

Ejercicio 4.11

Implement list from exercise 4.5.

```
#Sintaxis concreta
    Expression := list (Expression *(,))

#Sintaxis Abstracta
    list-exp(exps)
```

```
#Implementacion

(value-of (list-exp (cons exp1 (cons exp2 null))) env) =
    (pair-val (cons (value-of exp1 env)
            (pair-val(cons (value-of exp2 env)
                                            (null-val)))))
```