Lenguajes de Programación
Auxiliar Nº2

Auxiliar: Fabián Mosso

18/03/2016

Recuerde enunciar el contrato y tests.

1. Un número Natural se puede definer como zero o el sucesor de un `Natural`
   (a) Defina la notación BNF para el tipo Natural
   (b) Implemente usando `deftype` el tipo `Natural`
   (c) Implemente la función `(natToInt N)` que dado un número de tipo `Natural` retorna su representación de racket
   (d) Implemente la función `(natSum N1 N2)` que dado dos números de tipo `Natural` retorna el `Natural` resultante de la suma
2. Booleans: Considere el Interprete de expresiones aritméticas visto en clases:

```
(deftype Expr
  (num n)
  (add l r)
  (sub l r))

(define (parse s-expr)
  (match s-expr
    [(? number?) (num s-expr)]
    [(list '+ l r) (add (parse l) (parse r))]
    [(list '- l r) (sub (parse l) (parse r))]))

(define (calc expr)
  (match expr
    [(num n) n]
    [(add l r) (+ (calc l) (calc r))]
    [(sub l r) (- (calc l) (calc r))]))
```

```
(define (run prog)
  (calc (parse prog)))
```

1. Agregue el tipo `bool` al interprete, además agregue las operaciones `my-and`, `my-or`, y `my-not`
2. Agregue las operaciones `my-if` e `if-zero`