Lenguajes de Programación

Ejercicios propuestos

Karla Ramírez Pulido Manuel Soto Romero Alejandro Hernández Mora
Silvia Díaz Gómez Pedro Ulises Cervantes González

Semestre 2021-2
Facultad de Ciencias, UNAM

Considera la siguiente gramática del lenguaje de expresiones aritméticas (**AE**):

```
<expr>::= <id>
        | <num>
        | {<op> <expr>+}
<id>  ::= a | b | c | ...
<num> ::= 1 | 2 | 3 | ...
<op>  ::= + | - | * | / | modulo | expt | add1 | sub1
```

En Racket podemos definir la gramática anterior mediante el siguiente tipo abstracto de datos, el cuál está conformado por los siguientes constructores:

- `id` que representa una variable algebráica.
- `num` que representa un número.
- `op` que representa una operación aritmética. El primer parámetro es para cualquiera de las funciones de la gramática. El segundo parámetro es para la lista de argumentos que recibe la función anterior.

```
;; Definición del tipo AE
(define-type AE
[id   (i symbol?)]
[num  (n number?)]
[op   (f procedure?) (args (listof AE?))])
```

1. Ejercicios popuestos

1. Define el procedimiento `(subst expr sub-id value)`, el cual realiza la sustitución `expr[sub-id := value]` correspondiente. Es decir, que en la expresión `expr` reemplaza cada presencia de la variable `sub-id` por otra expresión `value`.

   ```
   ;; subst: AE symbol AE -> AE
   (define (subst expr sub-id value) ...)
   ```