

Lenguajes de Programación

Practica 4

Karla Ramírez Pulido

José Eliseo Ortiz Montaña

Semestre 2024-1
Facultad de Ciencias UNAM

Fecha de inicio: 03 de octubre 2023
Fecha de entrega: 18 de octubre 2023

Instrucciones

La realización y entrega de la práctica deberá realizarse en equipos de a lo más 3 personas*. La entrega debe respetar el orden de las funciones que es especificado en este documento. El uso de funciones auxiliares está totalmente permitido, sin embargo, deben ser declaradas justo después de la función principal que hace una de ellas.

```
.....  
;; Ejercicio 1  
;; [Documentación de la función]  
(define (my-fun a b c d) ... )  
;; [Documentación de la función auxiliar - incluir una breve descripción de  
;; la motivación de la función]  
(define (an-aux-fun lst1 lst2) ... )  
...  
...  
...  
.....
```

Las funciones deben incluir comentarios, tanto de documentación como del proceso de desarrollo. Éstos deben ser claros, concisos y descriptivos.

Se deberá subir la versión final de su práctica al apartado del classroom correspondiente antes de la fecha límite. Esto solo debe realizarlo un integrante del equipo; el resto de los integrantes del equipo deberá marcar como entregada la actividad y en un comentario privado especificar quienes son los miembros del equipo.

El archivo a entregar debe de ser un comprimido zip, con el nombre practica04.zip, en éste se deberán incluir los archivos `interp.rkt`, `parser.rkt`, `desugar.rkt`, `grammars.rkt` y `readme.md`, en el cual se deberán incluir los datos de cada miembro del equipo.

*Cualquier situación con respecto a este punto será tratada de acuerdo a las particularidades del caso. Para esto, acercarse al ayudante del rubro del laboratorio a la brevedad.

Estructura

La práctica está conformada por los siguientes archivos:

- `grammars.rkt`: archivo en el que se encuentra la definición de tipos que representan el **ASA**^{**} del lenguaje CFWSBAE, el cual corresponde a la siguiente sintaxis concreta:

```
<expr > :: = <num >
           | <id >
           | <bool >
           | <string >
           | {<op > <expr >+ }
           | {with  {{<id > <expr >} +} <expr >}
           | {with* {{<id > <expr >} +} <expr >}
           | {fun  {{<id >+} <expr >}}
           | {<expr > <expr >+ }
           | {if   <expr > <expr > <expr >}
           | {cond  {{<expr > <expr >}+ {else <expr >}}}

<num > :: = ... | -1 | 0 | 1 | ...
<id >  :: = a | ... | z | A | ... | Z | aa | ...
<string > :: = "a" | ... | "z" | "A" | ... | "Z" | "aa" ' | ...
<bool > :: = #t | #f
<op >  :: = + | - | / | * | modulo | min | max | expt | sqrt | sub1 | add1
           | < | > | <= | >= | = | not | and | or | zero? | num? | str? | bool? | str-length
```

Y el ASA del lenguaje CFSBAE, cuya sintaxis concreta es similar a la anterior, únicamente omitiendo las expresiones `with`, `with*` y `cond`.

- `parser.rkt`: archivo donde se encuentra la definición de la función `parse`, encargada de realizar el análisis sintáctico correspondiente.
- `interp.rkt`: archivo en el que se encuentra la definición de la función `interp`, la cual estará encargada de realizar el análisis semántico de una expresión del lenguaje correspondiente, a su vez contendrá la definición de la función `lookup` la cual será encargada de buscar un identificador dentro del ambiente de evaluación y por último la definición del tipo de dato para representar este mismo.
- `desugar.rkt`: archivo en el que se encuentra la definición de la función `desugar`, la cual estará encargada de realizar la transformación de expresiones del lenguaje CFWSBAE al lenguaje CFSBAE.
- `test.rkt`: archivo que contendrá una serie de pruebas unitarias para probar el trabajo realizado en la práctica.

^{**} Árbol de Sintaxis Abstracta, ASA.

Ejercicios

1. (2pt) Completar el cuerpo de la función `parse` del archivo `parser.rkt`, la cual es la encargada de realizar el análisis sintáctico del lenguaje CFWSBAE. Toma en cuenta lo siguiente:
 - En el caso de `fun` no se puede declarar dos parámetros con el mismo identificador, y si esto sucede se debe mandar un error.
2. (3pt) Completar el cuerpo de la función `desugar` del archivo `desugar.rkt`, la cual recibe una expresión del lenguaje CFWSBAE y la transforma en una del lenguaje CFSBAE. Toma en cuenta lo siguiente:
 - Las expresiones `with` son equivalentes a una aplicación de una función, por ejemplo:
`{with {{x 10} {y 20}} {+ x y}}`
Se puede transformar en:
`{{fun {x y} {+ x y}} 10 20}`
 - Las expresiones `with*` se pueden ver como una serie de expresiones `with` anidadas, y a su vez esto se puede transformar como se indica anteriormente, por ejemplo:
`{with* {{x 10} {y x}} {+ x y}}`
Se puede transformar en:
`{with {{x 10}} {with {{y x}} {+ x y}}}`
Y esto se puede reescribir como:
`{{fun {x} {{fun {y} {+ x y}} x}} 10}`
 - Las expresiones `cond` se puede ver como azúcar sintáctica para evitar el uso de expresiones `if` anidadas, por lo que al momento de aplicar la función `desugar` las expresiones `cond` se deberán transformar a expresiones `if` anidadas, por ejemplo:
`{cond {#t 10} {#f 20} {else 30}}`
Se puede transformar en lo siguiente:
`{if #t 10 {if #f 20 30}}`
3. (2pt) Completar el cuerpo de la función `lookup` del archivo `interp.rkt`, la cual busca el identificador dado dentro de un ambiente de evaluación y en caso de encontrarlo devuelve el valor asociado a éste, en caso contrario lanza un error indicado que el identificador se encuentra libre.
4. (3pt) Completar el cuerpo de la función `interp` del archivo `interp.rkt`, la cual corresponde al análisis semántico de la expresión recibida como parámetro. En este caso, la función recibirá dos parámetros, la expresión del lenguaje CFSBAE y un ambiente de evaluación y regresará el resultado de evaluar dicha función. Se debe considerar lo siguiente:
 - Al momento de evaluar aplicaciones de funciones se debe verificar que el número de parámetros formales de la función sea igual al número de parámetros reales.
 - La interpretación se tiene que realizar utilizando alcance estático y evaluación glotona.