Compiladores 2022-1
Facultad de Ciencias UNAM
Práctica 6: Tabla de símbolos

Lourdes del Carmen González Huesca Reyes Granados Naomi Itzel
Hernández Luna Nora Hilda

9 de diciembre de 2021
Fechas de entrega: 07 de enero del 2022

1. Lenguaje

El lenguaje resultante de aplicar los procesos definidos en las prácticas 2, 3, 4 y 5 al lenguaje fuente es el lenguaje `L10` que se define con la gramática siguiente:

```
<programa> ::= <expr>

<expr> ::= <const>
         | <var>
         | (const <type> <const>)
         | (begin <expr> <expr>*)
         | (primapp <prim> <expr>*)
         | (if <expr> <expr> <expr>)
         | (lambda ([<var> <type>]) <expr>)
         | (let ([<var> <type> <expr>])  <expr>)
         | (letrec ([<var> <type> <expr>]) <expr>)
         | (list <expr>*)
         | (<expr> <expr>)

<const> ::=  <boolean>
         | <integer>
         | <char>

<boolean> ::= #t | #f

<integer> ::= <digit> | <digit><integer>

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<var> ::= <car> | <car><var> | <car><digit> | <car><digit><var>

<car> ::= a | b | c | ... | z
```

```
<prim> ::= + | - | * | / | length | car | cdr

<type> ::= Bool | Int | Char | List | Lambda | (<type> → <type>) | (
   List of <type>)
```

Y definido con *Nanopass* como sigue:

```
(define-language L10
  (terminals
   (variable (x))
   (primitive (pr))
   (constant (c))
   (type (t)))
  (Expr (e body)
    x
    (const t c)
    (begin e* ... e)
    (primapp pr e* ...)
    (if e0 e1 e2)
    (lambda ([x t]) body)
    (let ([x t e]) body)
    (letrec ([x t e]) body)
    (letfun ([x t e]) body)
    (list e* ...)
    (e0 e1)))
```

Este es el lenguaje de entrada para esta práctica.

2. Ejercicios

1. *(2.5 pts.)* Definir el proceso `uncurry` encargado de *descurrificar* las expresiones `lambda` de nuestro lenguaje.

   Para el algoritmo de inferencia de tipos resultaba mas sencillo trabajar con las funciones currificadas, sin embargo para la traducción a C es mas sencillo trabajar con funciones multiparamétricas como las que nos ofrece dicho lenguaje.

   Para este proceso es necesario definir un nuevo lenguaje `L11` en el cual el constructor `lambda` vuelve a ser multiparamétrico.

   ```
   entrada : (uncurry  '(lambda ([x Int]) (lambda ([y Int]) (+ x y))))
   salida : '(lambda ([x Int] [y Int]) (+ x y))
   ```

2. *(2.5 pts.)* Definir una función `symbol-table-var` que genera la tabla de símbolos de una expresión del lenguaje. Para modelar la tabla de símbolos usaremos una estructura `Hash Table` del lenguaje Racket. El funcionamiento de éstas se puede consultar en la documentación oficial de Racket.

La tabla de símbolos va a tomar como llave el identificador de la variable y como valor tendra un par que almacena el tipo de la variable y su valor.

Hint: Utilizar `nanopass-case`.

3. *(2.5 pts.)* Definir el proceso `assigment` que modifica los constructores `let`, `letrec` y `letfun`, eliminando el valor asociado a los identificadores y el tipo correspondiente.

   Para este proceso es necesario definir un nuevo lenguaje `L12` en el que los constructores de asignación solo reciben una variable.

   Para poder realizar este proceso sin perder información, se hará uso de una tabla de símbolos. Esta tabla se pasará entre procesos para conservar los valores correspondientes. De esta forma, los resultados de este proceso deben incluir la tabla de símbolos.

```
entrada : (assigment (letrec ([foo (Int → Int) (lambda ([x Int]) x)
   ])
                             (foo 5)))
salida : (letrec (foo 5))
```

4. *(2.5 pts.)* Definir el proceso `list-to-array` encargado de traducir las listas de nuestro lenguaje en arreglos.

   Para este proceso se debe definir un nuevo lenguaje `L13` el cual elimina el constructor `list` y agrega un nuevo constructor `(array len t [e*...])`, este constructor requiere agregar `len` como una expresión terminal del lenguaje. `len` será la longitud del arreglo. Para simplificar esta práctica se ignorara el caso de la lista vacía el cual se retomará en el proyecto final.

   En la traducción, los elementos del arreglo serán los mismos de la lista original y la `len` será la longitud de la ésta.

```
entrada : (list-to-array  '(list 1 2 3 4 5))
salida : '(array 5 Int [1 2 3 4 5])
```

3. Entrega

- La entrega será en el *classroom* del curso. Basta entregar los archivos comprimidos y el código bien documentado. Indicando los integrantes del equipo.
- Esta práctica debe realizarse en equipos de a lo más 3 personas.
- En los scripts entregados, deberán considerar incluir lo siguiente como parte de la documentación:
  - Los nombres y números de cuenta de los integrantes del equipo.
  - Descripción detallada del desarrollo de la práctica.
  - En caso de haber sido necesario, la especificación formal de los nuevos lenguajes definidos, utilizando gramáticas.

    - Comentarios, ideas o críticas sobre la práctica.

- Se recibirá la práctica hasta las 23:59:59 horas del día fijado como fecha de entrega, cualquier práctica recibida después no será tomada en cuenta.
- **Cualquier práctica total o parcialmente plagiada, será calificada automáticamente con cero y no se aceptarán más prácticas durante el semestre.**