Compiladores 2020-1
Facultad de Ciencias UNAM
Proyecto Final

González Alvarado Raúl
gonzalv@ciencias.unam.mx

Hernández Cano Alejandro
ale.hdz333@ciencias.unam.mx

Hernández Leyva Mirén Jessamyn
mjhl1999@ciencias.unam.mx

Muñoz Barón Luis Miguel
luis_mi1999@ciencias.unam.mx

Fecha de entrega: 6 de febrero del 2021

Uso del compilador

El compilador se ejecuta de primeras y a modo de ejemplo usando el archivo `ejemplo3.mt`, sin embargo que puede usar con cualquier programa en el lenguaje LF, con extención `.mt`. Si se desea agregar alguno cambiar el path (ruta del archivo de ejemplo) en el archivo `Compilador.rkt`, .

Pasos a seguir:

1. Se debe descargar el proyecto del repositorio en GitHub, el cual se encuentra en el siguiente link: `https://github.com/ebooshi/Proyecto-de-Lambda`
2. Abrir el archivo `Compilador.rkt` en DrRacket. O bien ejecutarlo con `racket compiler.rtk`
3. Obtenemos 3 archivos de salida uno correspondiente al front-end (extención `.fe`), otro correspondinete al middle end (extrención `.me`) y por último el correspondiente al back-end (extrención `.c`).

Etapas del proceso de compilación

Escritura de archivos

Primero, la función `read-file` lee un archivo de extensión que contiene expresiones de LF (la extensión debe ser `".mt"`) directo de la ruta especificada.

Posteriormente, la función `write-file codigo` escribe en un archivo lo que contiene la código indicado en el path.

Luego, la función `write-file-int` que crea el archivo correspondiente a alguna de las fases de compilación crea el archivo en la ruta path cuyo contenido sera la variable código.

Por ultimo, la función `compilar` lee el archivo indicado en el path y aplica los procesos definidos en la lista passes. Esta debe llevar la ruta del archivo de la siguiente forma: `(compilar ".../ejemplos/ejemploLF1.mt")`

Pre-procesamiento

Front End

Para llevar a cabo el proceso de front-end utilizamos las siguientes fases, en este proceso se hace el análisis léxico, sintáctico y semántico.

- `remove-armed-if`: Unifica las expresiones `if`, identificandolas sin sentencias `else`, les añadade un `else` que regresa `void`, así ya solo queda un constructor de `if`, como resultado obteniendo el lenguaje LF1.
- `remove-string`: Toma los String encontrados y los transforma en una lista, obteniendo el lenguaje L3.
- `curry-let`: Se currifican las expresiones `let` y `letrec`, este proceso recibe como entrada el lenguaje L6 y regresa el lenguaje L7.
- `identify-assigments`: Se encuentran las expresiones `let` que son de tipo Lambda, o bien `let` de función y las transforma a expresiones `letrec`, las expresiones `let` que no corresponden a esta descripción se mantienen iguales, este proceso regresa L7.
- `un-anonymous` : Función que unifica el nombre las funciones anónimas, por lo tanto ahora nuestras funciones anónimas tendrán un nombre `foo#`, recibe L7 y regresa L8.
- `verify-arity`: Es un procesos que funciona como verificador en el lenguaje L8, no modifica nada, solo verifica que la aridad de los operadores sea la correcta, solamente tenemos operadores binarios y unarios.

  `verify-vars`: Este procesos verifica que no existan variables libres en las expresiones definidas en nuestro programa del lenguaje L8.

Middle End

Para llevar a cabo el proceso de middle-end utilizamos las siguientes fases,

- `curry`: Currifica las aplicaciones de funciones y las expresiones lambda. Recibe un lenguaje L8 y regresa una instancia del lenguaje L9.
- `type-const`: Este proceso le va a agregar el tipo de manera literal a las constante de nuestro lenguajes, recibe un lenguaje L9 y va a regresar una instancia de un lenguaje L10.
- `type-infer`: Con uso del algoritmo J este proceso es capaz de inferir el tipo de las expresiones del programa, depende de cada caso.
- `uncurry`: Función que quita la currificacion que le hicimos a los `let` y lambda en uno de los procesos anteriores.

Back End

En esta parte simplemente se traduce el código en lenguaje L11 a lenguaje C. Esto utilizando un `list-to-array` que como su nombre lo indica realiza la traducción de las listas a arreglos.