Compilador Kaskell

Álvaro García Tenorio [*] Miguel Pascual Domínguez[**]

19 de junio de 2018

Índice general

[*] alvgar14@ucm.es
[**] miguepas@ucm.es

Capítulo 1

Introducción

Comencemos primero explicando en que consiste nuestro lenguaje.

Un programa en código kaskell, consiste en 3 partes diferenciadas. Una primera parte que consiste en la declaración de registros. Nuestro lenguaje, permite la declaración de varios registros seguidos, que irán declarados uno detrás de otro, pueden estar separados por saltos de linea, espacios o sin separación. Además se permite la declaración de registros anidados, es decir, que uno de los campos de un registro, sea un registro previamente declarado.

La segunda parte consiste en los bloques. En nuestro lenguaje, los bloques están formados por la composición secuencial de instrucciones. Se permiten los bloques anidados, y las intrucciones pueden ser declaraciones de un array, un kinkeger (int) o un kool (bool), con o sin inicialización (asignación); asignaciones, khile (while), kor (for), if (if), kelse (else) o la llamada a una función.

Y por último, la tercera parte que consiste en la declaración de funciones. Las funciones pueden o no devolver un valor de tipo simple, es decir, un kinkeger o un kool, consideraremos que las funciones está formadas por dos partes, la parte de la cabeza que contiene los tipos de los argumentos, si los tiene, el identificador de la función y el tipo de retorno, si lo tiene; y la cola, que será los argumentos de tenerlos, el bloque de código y el valor de retorno, si lo tiene.

Capítulo 2

Estructura del compilador

Comentaremos ahora, la estructura del compilador, al igual del código del mismo. Para ello se comentará brevemente cada apartado del compilador.

2.1. Léxico

Comencemos con el léxico, en el léxico procesamos los identificadores de cualquier elemento del código, al igual que los símbolos especiales como "ƨ , y también las palabras reservadas.

Las palabras reservadas son: kif,kelse,khile,kor,kinkeger,kool, X (producto cartesiano), korr (or lógico), kand (and lógico), knot (not lógico), kod (operación modulo), kreturn (palabra reservada para el retorno de funciones), krue (true), kalse (false), y struk (registros).

Para poder analizar las palabras reservadas, nos ayudamos de un HashMap que tiene las palabras reservadas en el mismo, y al encontrar un identificador, comprobamos si es palabra reservada o no, viendo si está en el HashMap.

El resto de identificadores tienen una pequeña restricción de que solo pueden empezar por letras mayúsculas o minúsculas.

Y los comentarios de este lenguaje se escribe entre $. comentario .$.

2.2. Sintáctico

Continuemos con el análisis sintáctico del compilador.

Cup

El archivo Cup, consiste en la gramática de atributos que analiza la sintaxis del lenguaje, por tanto en vez de detallar la gramática, que esta ya presente en los archivos del compilador en el archivo "parser.cup". Vemos más conveniente, comentar informalmente la sintaxis del lenguaje.

Comencemos por las declaraciones. Hay 4 tipos de declaraciones: declaracion de tipo simple e identificador, array de tipo simple e identificador, tipo çomplejo"(tipo ya declarado, por ejemplo un registro) e identificador o array de tipo complejo e identificador. La sintaxis de estas cuatro declaraciones es: tipoSimple identificador, [tam 1º componente]...[tam i componente]...[tam n componente] tipoSimple identificador, IdentificadorTipoComplejo identificador, [tam 1º componente]...[tam i componente]...[tam n componente] IdentificadorTipoComplejo identificador.

Aclaremos que el tipo simple son kinkeger o kool.

Solo tenemos un tipo complejo que es la declaracion de los registros que es de la siguiente manera: identificador := struk declaraciones .

Veamos ahora las instrucciones: tenemos las instrucciones simples, complejas, y los bloques.

Un bloque se escribe de la siguiente manera: instrucciones .

Las instrucciones simples son las asignaciones, las declaraciones, y las declaraciones con asignaciones. Todas las instrucciones terminan con un ";". Las declaraciones ya las hemos aclarado antes, veamos como son las asignaciones y las declaraciones mixtas.

Las asignaciones son al igual que en los lenguajes tipo c++ o java, es decir identificador = expresión, identificador[i][j] = expresión, identRegistro.member = expresión.

Y las declaraciones mixtas, consisten en la declaración de un tipo simple y una asignación, es decir, tipoSimple identificador = expresión.

Veamos por último las instrucciones complejas. Estas son el if, el while y el for.

El if: kif (expresión) bloque o kif (expresión) bloque1 kelse bloque2.

El while: khile (expresión) bloque.

El for: kor (declaración mixta o asignación; expresión, asignación o expresión) bloque.

Las funciones se declaran de la siguiente manera:

La cabeza de la función: identificador : tipoArgumento1 X ... X tipoArgumentoN -¿ tipoRetorno. Recordemos que los argumentos y el retorno pueden no existir y por tanto se declararía como identificador :-¿.

Y la cola de la función: (identificador1, ...,identificadorN) -¿ bloque o bloque retorno.

Los bloques retorno son igual que los bloques normales con la salvedad de que tienen al final la instrucción kreturn indent/expresión/....

Además contamos con una gramática de expresiones aritméticas y lógicas, con las respectivas prioridades de operadores. La gramática de expresiones viene detalla en cup. Destacaremos que ciertas prioridades son distintas a lo habitual por ejemplo las de las comparaciones.

Árbol Sintáctico

2.3. Identificadores

2.4. Tipos

2.5. Generación Código de la Máquina-P

2.6. Errores

2.7. Ejemplo de Código