Universidad Nacional Autónoma de México
Compiladores
2018-1
Definición Dirigida por sintaxis Compilador básico
Profesor: Adrián Ulises Mercado Martínez

Descripción

La definición dirigida por sintaxis que se presenta a continuación es para un lenguaje similar a C con las siguientes restricciones:

1. Sólo admite variables del tipo int.
2. Las variables que se declaran son globales.
3. Sólo admite una función global main de tipo void.
4. Dentro de las palabras reservadas se considera a main como una de ellas.
5. Se están utilizando las etiquetas como atributos sintetizados usando el método presentando en el libro de Aho, "Compiladores, Principios, Técnicas y Herramientas" de la segunda edición descrito en las páginas de la 410-417 de la versión en español.
6. ltrue, lfalse, y lnext son listas que contienen las direcciones de las instrucciones donde debe haber un salto a una etiqueta.
7. Para los saltos se considera que el código intermedio está alojado en un arreglo por lo cual los saltos indican la posición i del arreglo a la que hay que saltar.

Definición Dirigida por sintaxis

PRODUCCIÓN	REGLAS SEMÁNTICAS
P → D F	$dir = 0$ temporales = 0

D → int id D	**si** !existe(id) **entonces**
	  insertar(id)
	sino
	  error(id duplicado)
	fin si

D → ε	
F → **void main(){** Q **}**	generar_etiqueta(main) generar_etiqueta(fin) retroceso(Q.lnext, fin)
Q → L	Q.lnext = L.next Q.first = L.first
Q → ε	
L → L_1 S	retroceso($L_1.lnext$, S.$first$) L.$lnext$ = S.$lnext$ L.$first$ = $L_1.first$
L → S	L.$lnext$ = S.$lnext$ L.$first$ = S.$first$

PRODUCCIÓN	REGLAS SEMÁNTICAS
S → **if**(B) {Q}	retroceso(B.*ltrue*, Q.*first*) S.*lnext* = combinar(B.*lfalse*, Q.*lnext*) S.*first* = B.*first*
S → **if**(B) {Q_1} **else** G {Q_2}	retroceso(B.*ltrue*, Q_1.*first*) retroceso(B.*lfalse*, Q_2.*first*) temp = combinar(Q_1.*lnext*, G.*next*) S.*lnext* = combinar(temp, Q_2.*lnext*)
S → **while**(B) Q	retroceso(Q.*ltrue*, B.*first*) retroceso(B.*ltrue*, Q.*first*) S.*next* = B.*lfalse* genera_codigo('goto' B.*firs*)
S → S_1 S_2	S_1.*next* = newLabel() S_2.*next* = S.*next* S.*codigo* = S_1.*codigo* || label(S_1.*next*) || S_2.*codigo*
S → **id =** E **;**	**si** existe(id) **entonces** inst =generar_codigo(id '=' E.dir) **si** E.*first* != -1 **entonces** S.*first* = E.*first* **sino** S.*first* = inst **fin si** **sino** error("El id no ha sido declarado") **fin si**
S → **break;**	S.*first* = -1 inst = genera_codigo("goto") S.*lnext* = crearlista(inst)
E → E_1 + E_2	E.*dir* = new Temp() inst = genera_codig(E.*dir* '=' E_1.*dir* '+' E_2.*dir*) **si** E_1.*first* != -1 **entonces** E.*first* = E_1.*first* **sino** **si** E_2.*first* != -1 **entonces** E.*first* = E_2.*first* **sino** E.*first* = inst **fin si** **fin si**
E → E_1 - E_2	E.*dir* = new Temp() inst = genera_codig(E.*dir* '=' E_1.*dir* '-' E_2.*dir*) **si** E_1.*first* != -1 **entonces** E.*first* = E_1.*first* **sino** **si** E_2.*first* != -1 **entonces** E.*first* = E_2.*first* **sino** E.*first* = inst **fin si** **fin si**
E → E_1 * E_2	E.*dir* = new Temp() inst = genera_codig(E.*dir* '=' E_1.*dir* '*' E_2.*dir*)

PRODUCCIÓN	REGLAS SEMÁNTICAS

	si $E_1.first$!= -1 **entonces**
	  E.$first$ = $E_1.first$
	sino
	  si $E_2.first$!= -1 **entonces**
	  E.$first$ = $E_2.first$
	  sino
	  E.$first$ = inst
	  fin si
	fin si

E → E_1 / E_2	E.dir = new Temp()
	inst = genera_codig(E.dir '=' $E_1.dir$ '/' $E_2.dir$)
	si $E_1.first$!= -1 **entonces**
	  E.$first$ = $E_1.first$
	sino
	  si $E_2.first$!= -1 **entonces**
	  E.$first$ = $E_2.first$
	  sino
	  E.$first$ = inst
	  fin si
	fin si

E → **id**	**si** existe(id) **entonces**
	  E.dir = id.$lexval$
	  E.$first$ =-1
	sino
	  error("El id no ha sido declarado")
	fin si

E → **num**	E.dir = num.$lexval$ E.$first$ =-1
B → B_1 || B_2	retroceso($B_1.lfalse$, $B_2.first$) B.$ltrue$ = combinar($B_1.ltrue$,$B_2.ltrue$) B.$lfalse$ = $B_2.lfalse$
B → B_1 **&&** B_2	retroceso($B_1.ltrue$, $B_2.first$) B.$lfalse$ = combinar($B_1.lfalse$,$B_2.lfalse$) B.$ltrue$ = $B_2.ltrue$

B → E_1 **relop** E_2	inst = genera_codigo("if" $E_1.dir$ oprel $E_2.dir$ goto –)
	inst = genera_codigo("goto")
	si $E_1.first$!= -1 **entonces**
	  B.$first$ = $E_1.first$
	sino
	  si $E_2.first$!= -1 **entonces**
	  B.$first$ = $E_2.first$
	  sino
	  E.$first$ = inst -1
	  fin si
	fin si