Compiladores
Práctica 2
Análisis Sintáctico Descendente LL(1)
Nombre_____
Grupo _____

José Sánchez Juárez

30 de marzo de 2016

1. Objetivo

Reconocer las gramáticas libres de contexto, las gramáticas LL(1), convertir ciertas gramáticas libres de contexto a gramáticas LL(1) y construir un analizador sintáctico descendente LL(1).

2. Analizador Sintáctico descendente

El analizador sintáctico LL(1) es descendente. Para poder determinar si el analizador sintáctico es del tipo LL(1), debe cumplir con dos reglas:

1. Si la producción es de la forma $A \to \sigma_1 \mid \sigma_2 \mid \cdots \mid \sigma_n$, se debe cumplir con la siguiente condición: $PRIMERO(\sigma_1) \cap PRIMERO(\sigma_2) \cap \cdots \cap PRIMERO(\sigma_n) = \Phi$.

2. Si la producción es de la siguiente forma $X \to \sigma \mid \lambda$, se aplica la siguiente condición $PRIMERO(X) \cap SIGUIENTE(X) = \Phi$

Cuando una gramática no es LL(1), se debe seguir el siguiente procedimiento para programarla:

1. Se aplican las reglas UNO y DOS, para determinar si es LL(1).

2. Se debe determinar si las producciones de la gramática tienen recursividad izquierda. Esto se comprueba si las producciones tienen la siguiente forma $A \to A\alpha \mid \beta$.

3. Se debe eliminar la recursividad izquierda, haciendo la siguiente transformación: $A \to \beta A'$, $A' \to \alpha A' \mid \lambda$.

4. Si las alternativas son de la forma $A \to \sigma_1 \mid \sigma_2 \mid \cdots \mid \sigma_n$, pero se intersectan entonces para corregir se debe factorizar.

5. Una vez que la gramática es LL(1) se procede a construir el analizador sintáctico descendente LL(1).

La forma de programar una gramática LL(1), se hace de la siguiente forma:

Algorithm .1: Análisis descendente LL(1)

```
1  Palabra ← SigPal();
2  PUSH(eof,Pila);
3  PUSH(SimboloInicial,Pila);
4  Cima ← Cima.valor;
5  while True do
6      if Cima == eof ∧ Palabra == eof then
7          EsEnPan(Sentencia Aceptada);
8          Salir();
9      else if Cima ∈ T ∨ Cima == eofthen
10         if Cima == Palabra then
11             POP(Pila);
12             Palabra ← SigPal() ;
13         else
14             EsEnPan(Sentencia No Aceptada);
15             Break;
16     else
17         if Cima ∈ N then
18             A ← Leer(Tabla[Cima, Palabra]);
19             k ←| A |;
20             POP(Pila);
21             for i ← k hasta 1 do
22                 if B_i <> λ then
23                     PUSH(B_i,Pila);
24         else
25             EsEnPan(No se puede expandir B_i);
26     Cima ← Cima.valor;
```

3. Problemas

De la siguiente gramática libre de contexto G:

1. $Expr \to (Expr)$
2. $\mid Expr\ Op\ id$
3. $\mid id$

4. $Op \rightarrow +$
5. $|-$
6. $|*$
7. $|/$

Donde $id = a, b, c, d, \cdots$. Demostrar si es LL(1), en caso de no ser LL(1) convertirla a LL(1). Hacer lecturas de cadenas donde se usa: Entrada, Pila y Producción. Para comprobar su sintaxis después de construir el analizador sintáctico LL(1).

4. Procedimiento

Lo siguiente es bansándose en la gramática G:

1. Justificar si la gramática G es LL(1).
2. Si no lo es convertirla a gramática LL(1).
3. Después de convertirla justificar si es LL(1).
4. Construir la tabla de primeros y siguientes para formar la tabla LL(1).
5. Escriba las justificaciones de la generación de las cadenas $a + (b * c$ y $a + b) * c)$ por la gramática G.
6. Escriba los árboles de derivación de la $(a + b) * c$ y de $(a * b)/c$.
7. Construir el analizador Sintáctico LL(1), de acuerdo al psudocódigo presentado en esta práctica.

5. Actividades

Realice las siguientes actividades:

1. Investigar en internet las gramáticas libres de contexto, para determinar la forma de sus producciones.
2. Investigue que es un árbol de derivación.
3. Investigue que es una árbol de análisis.
4. Convertir la gramática en gramática LL(1), tal como se expresa en la sección de procedimiento.
5. Generar la secuencia de caracteres $(a+b)*c$, con la gramática G. Aplicando la lectura deonde se usa: Entrada, Pila y Producción.

6. Justificar si se genera la siguiente cadena "$a + (b * c''$ *or* "$a + b) * c),''$ con la gramática G. Aplicando la lectura donde se usa: Entrada, Pila y Producción.

7. Para formar el árbol de derivación de la cadena $(a + b) * c$, se hacen las siguientes derivaciones:

$$Expr$$
$$2\ Expr\ Op\ id$$
$$6\ Expr * id$$
$$1\ (Expr) * id$$
$$2\ (Expr\ Op\ id) * id$$
$$4\ (Expr + id) * id$$
$$3\ (id + id) * id$$

8. Construya el árbol de análisis de las cadenas $a + (b * c)$ y $(a + b) * c)$