PROCESADORES DE LENGUAJE

Ingeniería Informática
Especialidad de computación
Tercer curso, segundo cuatrimestre

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba

Relación de ejercicios nº 3: **ANÁLISIS SINTÁCTICO DESCENDENTE**

Análisis sintáctico mediante descenso recursivo (*Backtracking*)

1. Considera la siguiente gramática de contexto libre

   P = {
   S → **a** A B
   A → **a** A **b** | **a b**
   B → **c** B **d** | ε
   }

   a. Codifica en pseudocódigo las funciones o procedimientos que permitan aplicar el método de análisis sintáctico por **descenso recursivo** "con retroceso" (*Backtracking*)
   b. Comprueba si el analizador construido reconoce o no las siguientes cadenas:
      - **a a b c d**
      - **a a b c c d**
   c. ¿Qué lenguaje genera esta gramática?

2. Considera la siguiente gramática de contexto libre

   P = {
   E → E + T | T
   T → T * P | P
   P → F ^ P | F
   F → **número** | (E)
   }

   a. Elimina la recursividad por la izquierda y factoriza por la izquierda la gramática.
   b. Codifica en pseudocódigo las funciones o procedimientos que permitan aplicar el método de análisis sintáctico por **descenso recursivo** "con retroceso" (*Backtracking*).
   c. Comprueba si el analizador construido reconoce o no la siguiente expresión:
      (3^2 + 4^2)^0.5

Gramáticas LL(1)

3. Indica **razonadamente** si las siguientes gramáticas son o no gramáticas LL(1), es decir, si admiten o no un análisis descendente predictivo:
   a. P = {S → a A | b B A → A a | a B → B b | b }
   b. P = {S → a A | b B A → a A | a B B → b B | b }
   c. P = {S → A b | B c A → a A | ε B → a B | ε }

4. Pon un ejemplo de una gramática de contexto libre que **no** sea ambigua **ni** recursiva por la izquierda y que esté factorizada por la izquierda pero que **no** sea una gramática LL(1).

5. Considera una gramática LL(1) que se transforma de la siguiente manera:
   - En cada producción, se añade un símbolo marcador (M_i) distinto **delante** de cada símbolo no terminal.
   - Cada símbolo M_i sólo tiene asociada la siguiente producción: $M_i \rightarrow \varepsilon$

   a. Comprueba que la gramática generada también es LL(1), es decir, su tabla predictiva LL(1) no tiene conflictos.

6. Escribe dos gramáticas que estén en la forma normal de Greibach de modo que una permita construir una tabla predictiva LL(1) sin conflictos y la otra no.

Análisis descendente predictivo y "recursivo"

7. Considera la siguiente gramática de contexto libre:

P = {
<asignación_lógica> → **identificador :=** <predicado>
<predicado> → <predicado> **or** <disyunción>
<predicado> → <disyunción>
<disyunción> → <disyuncion> **and** <conjunción>
<disyunción> → <conjunción>
<conjunción> → <simple> | **not (** <predicado> **)**
<simple> → <operando> <relación> <operando>
<simple> → **true** | **false** | **(** <predicado> **)**
<operando> → **identificador** | **número**
<relación> → **<** | **>** | **=**
}

Nota: se recomienda renombrar los símbolos **no** terminales.

   a. Elimina la recursividad por la izquierda y factoriza por la izquierda la gramática.
   b. Obtén los conjuntos "primero" y "siguiente".
   c. Construye la tabla de análisis sintáctico descendente predictivo.
   d. **Codifica** en pseudocódigo los procedimientos asociados a los símbolos no terminales de forma que se pueda realizar el **análisis descendente predictivo** según el método **recursivo**.
   e. Muestra la ejecución de los procedimientos, el árbol de activación y la derivación por la izquierda generados al analizar la siguiente sentencia:

valor := (a> 0) and (a < 10)

Análisis descendente predictivo y "no" recursivo

8. Considera la siguiente gramática de contexto libre:

P = {
S → D S | ε
D → T L **;**
T → **integer | real**
L → I L'

I → **identificador** C
C → **[número]** C | ε
L' → **,** I L' | ε
}

Esta gramática puede generar "algunas" declaraciones de variables multidimensionales del **lenguaje C**, como, por ejemplo:

int entrada[5];
float temperatura [30][24];

a. Obtén los conjuntos "primero" y "siguiente".
b. Construye la tabla de análisis **descendente predictivo**.
c. Utiliza la tabla predictiva para realizar un análisis descendente no recursivo de las declaraciones anteriores.

9. Considera la siguiente gramática que permite generar algunas declaraciones en pseudocódigo:

P = {
S → S D ; | D ;
D → L **:** T
L → L **, identificador** | **identificador**
T → **entero** | **real** | **carácter** | **array (** N **) of** T
N → N, E **..** E | E **..** E
E → Signo **número**
Signo → **+** | **-** | ε
}

a. Elimina la recursividad por la izquierda y factoriza por la izquierda la gramática.
b. Obtén los conjuntos "primero" y "siguiente".
c. Construye la tabla de análisis sintáctico descendente predictivo.
d. Analiza la siguiente declaración:
temperatura: array (-5 .. 5) of real;

Gramáticas LL(1), análisis descendente predictivo y métodos de recuperación de errores

10. Considérese la siguiente gramática de contexto libre

P = {
1) S → S D
2) S → D
3) D → L **:** T **;**
4) L → L **,** ***identificador***
5) L → ***identificador***
6) T → ***entero***
}

- Esta gramática permite generar declaraciones de variables, como, por ejemplo: ***a, b, c : entero;***
  donde ***a***, ***b*** y ***c*** son identificadores.

a) Elimina la recursividad por la izquierda y factoriza la gramática por la izquierda.
b) A partir de la gramática obtenida en el apartado "a":

- Construye los conjuntos "**primero**" y "**siguiente**" de los símbolos no terminales.
- Construye la tabla de análisis **descendente predictivo**.
- Utiliza el método recuperación de errores de **"nivel de frase"** para **completar** la tabla predictiva.
- Utiliza la tabla predictiva para realizar un análisis descendente **no** recursivo de la siguiente declaración errónea:
  id id ,, id entero

11. Considérese la siguiente gramática de contexto libre

P = {

1) S → S D
2) S → D
3) D → T **:** L **;**
4) L → L **,** ***identificador***
5) L → ***identificador***
6) T → ***real***

}

- Esta gramática permite generar declaraciones de variables, como, por ejemplo: ***real: x, y, z;***
  donde ***x***, ***y*** y ***z*** son identificadores.

a) Elimina la recursividad por la izquierda y factoriza la gramática por la izquierda.
b) A partir de la gramática obtenida en el apartado "a":
   - Construye los conjuntos "**primero**" y "**siguiente**" de los símbolos no terminales.
   - Construye la tabla de análisis **descendente predictivo**.
   - Utiliza el método recuperación de errores de **"nivel de frase"** para **completar** la tabla predictiva.
   - Utiliza la tabla predictiva para realizar un análisis descendente **no** recursivo de la siguiente declaración errónea:
     real id id ,, id

Diseño de una gramática que admita un análisis descendente predictivo

12. Análisis sintáctico de los prototipos de funciones en C:
    a. Diseña una gramática que permita generar los prototipos de las funciones de C que sólo utilizan los tipos int, char y punteros a int o char.
    b. Construye la tabla predictiva de análisis LL
    c. Analiza la siguiente sentencia:
       char * letras (int , char **, char);
    d. Utiliza el método de **nivel de fase** para analizar la siguiente sentencia errónea:
       int int mayor (int , (, ;