CC3006: Tarea #4

Para entregar el Lunes, Abril 26, 2010

Bidkar Pojoy

Carlos E. López Camey

Defina una Gramática Independiente de Contexto. ¿Cuál es el uso que se le da a dicho concepto dentro del análisis sintáctico ?

Es una gramática G que describe sistemáticamente la síntaxis de un lenguaje. En el análisis sintáctico, se usa G para hacer salida al árbol sintáctico de la entrada, eso es, la salida del análisis sintáctico.

Formalmente, G es una 4-tupla $\langle T, NT, S_i, P\rangle$ en donde:

1. T, es el conjunto de símbolos llamados "terminales" que son símbolos básicos que forman a cadenas.
2. NT, es el conjundo de símbolos llamados "no terminales" que son variables sintácticas que denotan un conjunto de cadenas. Imponen una estructura jerarquica en el lenguaje.
3. S_i, un elemento de NT que es llamado "símbolo inicial" y el conjunto de cadenas que denota es el lenguaje generado por G.
4. P, el conjunto de producciones, que especifican la manera en que cada terminal o no terminal pueden ser combinados para formar cadenas. Cada producción consiste en
   - Un no terminal llamado "lado izquierdo" de la producción.
   - El símbolo $->$
   - Un cuerpo o "lado derecho" de la producción, que consiste en cero o más terminales y no terminales. Los símbolos del cuerpo describen una manera en los que las cadenas del no terminal pueden ser construídas

¿Cuál es la diferencia entre una derivación por la izquierda y una derivación por la derecha? ¿Qué relación tiene dicha derivación con un árbol sintáctico?

En una derivación por la izquierda, el no-terminal más a la izquierda es escogido siempre en cada derivación, en una derivación por la derecha, se escoge siempre el no-terminal más a la derecha.

El orden en el cual los símbolos fueron remplazados i.e. si fue una derivación por la derecha o por la izquierda, genera el mismo árbol sintáctico ya que este ignora el mismo. Cada árbol sintáctico tiene asociada una sola y única derivación por la izquierda y una única derivación por la derecha.

¿Qué es lo que significa que una gramática G sea ambigua?

Que esa gramática produce dos o más árboles sintácticos diferentes para alguna cadena $w \in L(G)$

Dada la siguiente gramática G, genera una nueva gramática G' no recursiva por la izquierda

G**:**

$E->E+T|T$
$T->TF|F$
$F->$ ***|a|b**

G'**:**

$E->TE'$
$E'->+TE'|\epsilon$
$T->FT'$
$T'->FT'|\epsilon$
$F->\mathbf{a}F'|\mathbf{b}F'$
$F'->$ *****$F'|\epsilon$

Dada la siguiente gramática de una producción, genere una gramática equivalente que se encuentre factorizada por la izquierda

$$A->\alpha\beta_1|\alpha\beta_2$$

Es equivalente a G**:**

$A->\alpha A'$

$A'->\beta_1|\beta_2$

Dada una producción de la forma $A->X_1X_2X_3\ldots X_n$ dentro de una gramática G $LL(1)$ ¿Cuál sería el algoritmo utilizado para procesar dicha producción durante un análisis sintáctico por decenso recusrivo?

Tenemos que tener una función o método para cada no-terminal $X_m->X_jX_{j+1}\ldots X_k$ con la siguiente estructura:

Escoger una producción para X_m con la ayuda de $FOLLOW(X_m)$ y $FIRST(X_m)$ i.e. con la tabla de predicciones.
for $i=1$ a n **do**
  if X_i es no-terminal **then**
    llamar al método X_i
  else
    if X_i es terminal y es igual al símbolo de entrada actual α **then**
      avanzar el puntero de símbolo actual al siguiente símbolo
    else
      error
    end if

end if
end for

¿Qué métodos de recuperación de errores existen en un análisis sintáctico predictivo? ¿Cómo utilizaría los conjuntos $PRIMERO$ y $SIGUIENTE$ de una gramática $LL(1)$ para implementar dichos métodos?

1. **Modo pánico**: En este modo de recuperación de errores, se trata se dejan pasar los símbolos y se trata de encontrar una sincronización de tokens que puedan ser identificados.

   Para esto, utilizamos $SIGUIENTE$ Y $PRIMERO$ para tratar de encontrar esa sincronización de tokens, por ejemplo, con $PRIMERO$ podríamos saber con que símbolo puede empezar algún no-terminal y tratar de hacer match con él.

2. **Recuperación a nivel de fase**: En este modo, generamos la tabla de análisis sintáctico con la ayuda de $PRIMERO$ y $SIGUIENTE$, eso es, una matriz con entradas $[M, \alpha]$ en donde M es un no-terminal y α un terminal, y la entrada que se encuentra ahi, es la sustitución que tendríamos que hacer explicitamente. Si la gramática no es ambigua, tendremos una única entrada para cada espacio.

   Si hay entradas vacías en la matriz, significa que para el no-terminal M con el símbolo de entrada α se produce un error. La recuperación a nivel de fase consiste en tratar de llenar las entradas en blanco de esta matriz con apuntadores a rutinas de error, estas rutinas pueden modificar, insertar o eliminar símbolos de entrada según convenga o incluso sacar de la pila no-terminales.

¿Cuál es el objetivo principal del análisis sintáctivo ascendente o $Bottom - Up\ Parsing$?

Reconocer gramáticas LR i.e. produce una derivación por la derecha. Estas gramáticas no tienen que estar factorizadas por la izquierda y pueden ser recursivas por la izquierda también (al contrario del $Top - Down\ Parsing$).

¿Defina lo que es un mango ($handle$) y discuta su uso dentro del análisis sintáctico ascendente.

Es una cadena compuesta por símbolos gramaticales que hace *match*, es decir, concuerda con el cuerpo de la definición de un no-terminal. En el análisis sintáctico ascendente, se usa para reducir las expresiones a no-terminales y así encontrar el árbol sintáctico hasta llegar a la raíz. La última reducción de un parseo sintáctico ascendente por ejemplo, tiene un mango w que tiene que hacer match con el cuerpo del símbolo inicial S, eso es, existe una producción $S- > w$ para el símbolo inicial S.

¿Cuáles son las cuatro operaciones o acciones que puede tener un analizador sintáctico por despazamiento-reducción, y en qué consisten?

1. **Desplazar** ($Shift$): Meter el siguiente símbolo de la entrada en el stack.

2. **Reducir**: Identificar en el elemento que está hasta arriba del stack un handler y remplazarlo por el no-terminal en la cadena de entrada.

3. **Aceptar**: Aceptar que la entrada está gramaticalmente corrrecta

4. **Error**: Rechazar la entrada, ya que no está gramaticalmente correcta, opcionalmente llamar a una rutina de recuperación de errores.

Bibliografía

- *Aho, Sethi, Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986. ISBN 0-201-10088-6*