Analizador Sintáctico Predictivo No Recursivo

Arturo Ferreira1, Manuel Avalos1

1 Facultad Politécnica, Universidad Nacional de Asunción,  
San Lorenzo, Paraguay

1{aferreiraduarte, mavalosgodoy}@gmail.com

1. Introducción

El documento tratará brevemente sobre los detalles de implementación del Analizador Sintáctico Predictivo No Recursivo, así como también sobre los inconvenientes que se encontraron durante el desarrollo del mismo. También se comentará sobre las suposiciones definidas para facilitar la implementación del mismo.

1. Ambiente de Desarrollo

El lenguaje de programación seleccionado fue Java (JDK 1.6) y utilizando como IDE el Netbeans 6.5. El trabajo fue desarrollado para su uso en un ambiente web, utilizando JSP, Servlets y el servidor web Tomcat 6.0.

1. Especificaciones de la Entrada

El analizador recibirá como entrada una gramática BNF limpia, bien formada y sin recursión por la izquierda ni necesidad de factorización, además deberá recibir una cadena de entrada de componentes léxicos para ser validados.

* 1. Ingreso de Gramática BNF

La gramática BNF será ingresado por medio del teclado. Separando el lado derecho y el lado izquierdo de cada producción por dos puntos (**“ : ”**). Cuando un No-terminal cuenta con más de una producción, cada producción debe ir en líneas diferentes, cada línea esta separada por un “salto de línea (Enter ) ”. Los terminales y los No-terminales del lado derecho deben estar separados por espacios. Todo símbolo ubicado en el lado derecho de cualquier producción es considerado terminal si no aparece en el lado izquierdo de alguna otra producción de la gramática. El vacío en el programa se representa por medio del símbolo @.

**Ejemplo:**

Gramáticas expresadas como sigue:

V 🡪 R T|ε  
R 🡪 id R | id  
T 🡪 type | id

Deberán ser ingresadas de la siguiente forma.

V: R[espacio] T

V: @

R: id[espacio] R

R: id

T: type

T: id

* 1. Cadena de Componentes Léxicos

La cadena de componentes léxicos debe ser ingresado separando cada componente léxico por espacios.

**Ejemplo:**

Cadena a ingresar:

(id+id)\*id

Cadena formateada:

[espacio] ( [espacio] id[espacio] +[espacio] id[espacio] ) [espacio] \*[espacio] id

1. Breve descripción sobre los detalles de Implementación.

Toda la implementación del analizador se centra en la clase ASP. El atributo más importante de la clase es el conjunto de No-terminales, que es representado por medio de una tabla hash que guarda objetos de la clase No-Terminal. La clave para la tabla hash es el nombre que se le asigna al No-terminal.

Del objeto No-terminal podemos citar los atributos más importantes, que son : El conjunto de lados derechos que corresponde al no-terminal, en el caso que un No-terminal tenga asignado más de un lado derecho. Luego están los conjuntos Primero y Siguiente, la fila de la tabla de ASP correspondiente al No-terminal.

La clase ASP cuenta con dos implementaciones para hallar el conjunto Primero, Primero() y PrimeroProducción(), la primera función es utilizada para hallar el conjunto primero de un No-terminal en particular, por lo tanto esta función recibe como parámetro de entrada el nombre del No-terminal.

La segunda función es utilizada para hallar el conjunto primero de un conjunto de terminales y No-terminales, generalmente siendo el lado derecho de alguna producción. Esta función fue creada con el fin de poder ser utilizada en la implementación de la creación de la tabla ASP.

Los algoritmos implementados para calcular los conjuntos Primero y Siguiente y otros algoritmos principales fueron sacados de [1]. Se podrá encontrar mayor información sobre los algoritmos implementados en el Javadoc o el propio código fuente, para obtener así un mejor entendimiento de los mismos.

* 1. Primero

La función para hallar el conjunto Primero es por medio de llamadas recursivas, previendo de no ingresar en un ciclo durante el proceso. Esto se realiza verificando que se realiza una llamada recursiva si y solo si el nuevo conjunto Primero que se quiere hallar es distinto del conjunto Primero que realmente se desea calcular. Cada conjunto Primero de cada No-terminal es calculado desde cero, es decir, si el conjunto Primero ya fue hallado en algún momento previo este no es almacenado.

* 1. Siguiente

La función para hallar el conjunto Siguiente es también por medio de llamadas recursivas y este es solo utilizado una vez que se han hallado todos los conjuntos Primeros de los No-terminales. El principal problema del cálculo de Siguiente por medio de la técnica adoptada fue la recursividad por la derecha, ya que se daba la posibilidad, al aplicar la regla 3, de ingresar en un ciclo infinito.

Para solucionar el problema se realiza una verificación de que la llamada recursiva se realiza si y solo si el nuevo conjunto Siguiente a ser hallado, es distinto del conjunto Siguiente que se desea calcular y también que sea distinto del elemento padre o conjunto Siguiente Padre que inicio la primera llamada para ingresar a la función recursiva. Con estas verificaciones se evita entrar en ciclos corto y largo. Cada conjunto Siguiente de cada No-terminal es calculado desde cero, es decir, si el conjunto Siguiente ya fue hallado en algún momento previo este no es almacenado y por ende será re-calculado.

1. Posibles mejoras

Las principales posibilidades de mejoras que pueden realizarse están dadas en el cálculo de los conjuntos Primero y Siguiente. En el sentido de que si los conjuntos ya son calculados o hallados en pasos previos, estos ya deben ser almacenados de manera a agilizar el proceso de análisis en general.

1. Referencias
2. Compiladores: Principios, Técnicas y Herramientas. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Segunda Edición. Editorial Pearson.