Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 3. Análisis Sintáctico Descendente

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Procesadores del Lenguaje

1. Características del análisis sintáctico

El análisis sintáctico se encarga de:

- Leer tokens
- Comprobar que el orden de los tokens corresponde a la sintaxis predeterminada.
- Genera errores en caso de que la secuencia no corresponda a la sintaxis.
- Genera árboles de análisis sintáctico.

El análisis sintáctico se especifica mediante una gramática libre de contexto y se implementa mediante un autómata de pila.

2. Gramáticas libres de contexto

las gramáticas libres de contexto vienen definidas por la cuádrupla G(N,Σ,P,S) donde:

- N: Alfabeto de símbolos no terminales.
- Σ: Alfabeto de símbolos terminales.
- P: Conjunto de producciones.
- S: Axioma de la gramática (Símbolo inicial).

En los árboles de análisis sintáctico, los nodos corresponden a símbolos no terminales mientras que, las hojas a terminales y no terminales. Cada ramificación simboliza una producción.

Las **formas sentenciales** de la gramática se obtienen leyendo las hojas de izquierda a derecha.

Las derivaciones canónicas son aquellas que siguen un orden predeterminado para sustituir las hojas no terminales. Estas pueden ser a la izquierda, donde se sustituye siempre la hoja no terminal que esté más a la izquierda, o a la derecha, donde se sustituye la que esté más a la derecha.

Una gramática es ambigua si existe alguna cadena a la que se le puede asociar dos o más árboles de análisis sintáctico diferentes. Demostrar que una gramática es ambigua es difícil, sin embargo, existen algunas construcciones gramaticales que producen ambigüedad:

- Reglas de la forma:
  - $E \rightarrow E \ ... \ E$
- Gramáticas con ciclos
  - $S \rightarrow A$
  - $S \rightarrow a$
  - $A \rightarrow S$
- Conjuntos de reglas con caminos alternativos entre dos puntos:
  - $S \rightarrow A$
  - $S \rightarrow B$
  - $A \rightarrow B$
- Producciones recursivas donde la regla no recursiva pueda producir la cadena vacía.
  - $S \rightarrow H \ R \ S.$
  - $S \rightarrow s$
  - $H \rightarrow h \mid \lambda$
  - $R \rightarrow h \mid r \mid \lambda$

Una gramática es recursiva si, a partir de un símbolo no terminal, se puede obtener una forma sentencial que incluya el mismo símbolo no terminal. Pueden ser recursivas a izquierdas, donde el símbolo no terminal aparece a la izquierda de la forma sentencial, o a derechas, donde el símbolo no terminal aparece a la derecha.

3. Notación EBNF (Forma de Backus-Naur Extendida)

Permite definir gramáticas con partes derechas regulares. Existen varios símbolos para definir las gramáticas. Esta notación no añade capacidad expresiva, por lo que se puede transformar de EBNF a BNF:

- Disyunción $A \to \alpha\ (\rho_1 \mid \rho_2 \mid \rho_3)\ \beta$
  - $A \to \alpha\ A'\ \beta$
  - $A' \to \rho_1$
  - $A' \to \rho_2$
  - $A' \to \rho_3$
- Clausura $A \to \alpha\ (\rho)^*\ \beta$
  - $A \to \alpha\ A'\ \beta$
  - $A' \to \rho\ A'$
  - $A' \to \lambda$
- Clausura Positiva $A \to \alpha\ (\rho)^+\ \beta$
  - $A \to \alpha\ \rho\ A'\ \beta$
  - $A' \to \rho\ A'$
  - $A' \to \lambda$
- Opcionalidad $A \to \alpha\ (\rho)?\beta$
  - $A \to \alpha\ A'\ \beta$
  - $A' \to \rho$
  - $A' \to \lambda$

4. Tipos de análisis sintácticos

Para saber si una cadena x pertenece a un lenguaje reconocido por G existen varios algoritmos.

4.1. Algoritmos de Resolución General

- Algoritmo de Cocke-Younger-Kasami $O(n^3)$.
- Algoritmo de Early $O(n^3)$ *ó* $O(n^2)$ depende de si es ambigua o no.

Los algoritmos de resolución general tienen órdenes polinomiales, esto provoca que para una compilación en un tiempo razonable necesitemos un orden lineal.

4.2. Algoritmos de orden lineal

Estos algoritmos imponen restricciones a la gramática pueden ser de análisis descendente o ascendente.

4.2.1. Análisis Descendente

Se construyen los árboles partiendo de la raíz hacia las hojas, además, se tiene en cuenta un token de preanálisis (lookahead) para realizar la elección de la regla a expandir. Sirve para analizar gramáticas LL.

4.3. Análisis Ascendente

Se parte de las hojas hacia la raíz. Cuando se obtiene la parte derecha de una regla, se sustituye por su símbolo no terminal. Sirve para analizar gramáticas LR.

5. Análisis sintáctico descendente predictivo

El análisis sintáctico descendente con retroceso consiste en sustituir cada símbolo no terminal por cada una de sus producciones hasta encontrar la producción adecuada. Sin embargo, esta técnica puede tener un coste exponencial y solo sería viable para gramáticas sencillas.

Para obtener un análisis de orden lineal necesitamos saber cual es la producción que debe utilizarse para expandir un símbolo no terminal. Para ello necesitamos conocer el token que se pretende analizar y elegir la producción a expandir.

Dado un token y un símbolo no terminal, debe existir solamente una producción que comience por ese token (condición LL1). Para saber la elección debemos calcular los conjuntos de predicción.

5.1. Conjunto de Primeros(α)

$a \in Prim(\alpha)$ *si* $a \in \Sigma$ *y* $\alpha \rightarrow a\beta$
$\lambda \in Prim(\alpha)$ *si* $a \rightarrow \lambda$

5.2. Conjunto de Siguientes(α)

$a \in Sig(A)$ *si* $a \in \Sigma$ *y* $S \rightarrow \alpha A a \beta$
$\$ \in Sig(A)$ *si* $S \rightarrow \alpha A$ (Si S es el axioma).
Si la regla es del tipo: $B \rightarrow \alpha\ A\ \beta$

- Si $\lambda \notin Prim(\beta), Sig(A) = Sig(A) \cup Prim(\beta)$
- Si $\lambda \in Prim(\beta), Sig(A) = Sig(A) \cup (Prim(\beta) - \{\lambda\}) \cup Sig(B)$

5.3. Conjunto de Prediccion(α)

Se refiere al conjunto de símbolos terminales que pueden aparecer cuando se va a expandir la producción.

- Si $\lambda \notin Prim(\alpha), Pred(A \rightarrow \alpha) = Prim(\alpha)$
- Si $\lambda \in Prim(\alpha), Pred(A \rightarrow \alpha) = \{Prim(\alpha) - \{\lambda\} \cup Sig(A)\}$

6. Condición LL1

Se debe cumplir para poder hacer un análisis descendente predictivo. La condición es que los conjuntos predicción de cada producción con el mismo símbolo no terminal, deben de ser disjuntos.

Podemos transformar algunas gramáticas no LL(1) en una gramática equivalente que sí lo sea.

- Factorización a la izquierda Transformar:
  - $A \rightarrow \alpha\beta_1$
  - $A \rightarrow \alpha\beta_2$

  En:
  - $A \rightarrow \alpha A'$
  - $A' \rightarrow \beta_1$
  - $A' \rightarrow \beta_2$
- Eliminación de la recursividad
  - $A \rightarrow A\ \alpha$
  - $A \rightarrow \beta$

  En:
  - $A \rightarrow \beta A'$
  - $A' \rightarrow \alpha A'$
  - $A' \rightarrow \lambda$

7. ASDP dirigido por tabla

Se basa en una pila que almacena las hojas del árbol de análisis y una tabla de análisis que tiene una fila por cada símbolo no terminal y una columna por cada símbolo no terminal y el final de entrada.

Las celdas de la tabla contienen enlaces a las reglas de producción. El contenido de la tabla indica qué regla expandir en función del símbolo terminal recibido. El funcionamiento es el siguiente:

1. Almacenar en la pila el símbolo de fin de entrada ($).
2. Almacenar en la pila el símbolo inicial (axioma).
3. Leer el siguiente token.
4. Repetir hasta que la pila y la cadena de entrada estén vacías:

   a) Si el símbolo de la cima de la pila es un terminal entonces
   - Si el símbolo es el siguiente token, consumirlo y leer el siguiente.
   - Si no es el siguiente token, genera un error.

b) Si el símbolo es un no terminal (A), entonces:

- Buscar en la tabla la fila A y la columna siguiente token y, si la celda contiene una regla, almacenar el consecuente de forma inversa.
- Si no contiene ninguna regla, entonces genera un error.

El contenido de la tabla de análisis será el siguiente: Para cada símbolo no terminal, se estudian sus producciones y, para cada una de sus producciones, su conjunto de predicción.

Para cada elemento del conjunto de predicción se incluye en la celda (A,a) el enlace a la producción. Las celdas vacías suponen errores sintácticos.

Es necesario que la gramática sea LL(1) porque sino, se producirían colisiones al asignar el valor a las celdas.

8. Analizador Descendente Recursivo

Su funcionamiento se basa en un conjunto de funciones recursivas. Cada símbolo no terminal genera una función y esta selecciona la regla de producción a ejecutar en función del valor del siguiente token. La pila está implícita en las llamadas a las funciones.

9. Gestión de errores

Existen dos tipos de errores: los producidos por el emparejamiento del token con la cima de la pila y los producidos en la tabla de símbolos.

Los primeros se pueden avisar con el mensaje «Encontrado nextToken cuando se esperaba token» y los segundos con «Encontrado nextToken cuando se esperaba Pred(A)».

Para la recuperación de errores podemos utilizar símbolos de sincronismo añadiendo la función skipTo(Token sync) que se salta los tokens de la cadena de entrada hasta llegar al token de sincronismo. Es habitual utilizar el punto y coma como símbolo de sincronismo.