Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 4. Análisis Sintáctico Ascendente

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Procesadores del Lenguaje

1. Introducción

El análisis sintáctico ascendente construye la inversa de la derivación por la derecha. El problema fundamental es decidir cuando sustituir la parte derecha de una regla por su parte izquierda.

El conjunto de gramáticas que pueden ser analizadas mediante este tipo de análisis se denomina LR(1). Este conjunto es mucho más amplio que el de las gramáticas LL(1).

2. Análisis Sintáctico por desplazamiento y reducción

El algoritmo más utilizado es el de desplazamiento-reducción. Este algoritmo se basa en una pila de estados y una tabla de análisis. Para generar esta tabla existen diferentes métodos como el LR(1) que genera tablas muy grandes, el SLR que genera tablas compactas, pero no puede aplicarse a todas las gramáticas o el LALR que genera tablas compactas y puede aplicarse a la mayoría de las gramáticas.

Utiliza dos acciones básicas: desplazar y reducir.

- **Desplazar**: consume un token de la cadena de entrada.
- **Reducir**: Sustituye en la pila los símbolos de una parte derecha de una regla por su parte derecha.

El algoritmo usa una pila de estados y una tabla de análisis con dos partes:

- Acciones: Para cada símbolo terminal y $. Existen cuatro tipos de acciones:
  - **dj**: Desplaza y apila el estado j.
  - **rk**: Reduce por la regla k-ésima. Consiste en eliminar tantos estados de la pila como elementos de la parte derecha de la regla. Se analiza el estado de la cima de la pila (p). Por últmo, se apila el estado Ir-a(p,A) siendo A el símbolo no terminal de la regla k-ésima.
  - **Aceptar**: Termina el análisis aceptando la cadena.
  - **Error**: Produce un error cuando no encuentra una acción.
- Ir-a: Para cada símbolo no terminal.

Las filas corresponden a los estados

3. Autómata reconocedor de prefijos variables

Los elementos se obtienen colocando un punto en cualquier posición de la parte derecha de una regla. El punto significa que los símbolos a la izquierda de este ya han sido reconocidos.

Las reglas que producen lambda solo generan un elemento.

3.1. Clausura de un elemento

Sea I un conjunto de elementos, todos los elementos de I pertenecen a la *clausura(I)*. Si la clausura(I) contiene un elemento de la forma $A \to \alpha \cdot B\beta$, entonces se añade a clausura(I) todos los elementos de la forma $B \to \cdot\alpha$.

3.2. Operación Ir-a

Sea I un conjunto de elementos y A un símbolo cualquiera de la gramática, la operación Ir-a(I,A) da como resultado otro conjunto de elementos de forma que para cada elemento de I tal que $B \to \alpha \cdot A\beta$, se añade a Ir-a(I,A) los elementos del conjunto clausura($B \to \alpha A \cdot \beta$).

3.3. Colección Canónica de un conjunto de elementos

Se amplía la gramática añadiendo una regla que genera el símbolo inicial ($X \rightarrow S$). Se calcula el conjunto $I_0 = clausura(X \rightarrow \cdot S)$ que será el primer elemento de la colección.

Por cada conjunto de la colección y cada símbolo se calcula el conjunto Ir-a(C,A) que se añade a la colección si no se había hecho previamente. Este paso se repite hasta que no se puedan añadir más conjuntos a la colección.

Si se representan los conjuntos de la colección como estados y las operaciones Ir-a() como transiciones, se obtiene un autómata reconocedor de prefijos variables.

4. Algoritmo LR(0)

Para construir la tabla tenemos que colocar los estados en las filas y los símbolos terminales y no terminales en las columnas. Las celdas que tienen símbolos terminales son desplazamientos, mientras que las que tienen no terminales son acciones de Ir-a().

La acción de Aceptar se coloca en la celda ($,X).

Para todo estado que contenga un elemento con el punto al final de la regla k-ésima, tenemos que colocar rk en todas las celdas de ese estado.

Este algoritmo no necesita token de preanálisis, sin embargo, hay gramáticas que pueden generar conflictos.

5. Algoritmo SLR

Para evitar el problema de los conflictos de LR(0), tendremos que considerar los conjuntos siguientes de cada símbolo no terminal y tener en cuenta un token de preanálisis.

El algoritmo es igual a LR(0), pero al rellenar las acciones de reducción tenemos que colocar rk en todas las celdas de los tokens pertenecientes a Siguiente(A).

6. Gestión de Errores

Los errores sintácticos aparecen cuando no existe ninguna acción asociada al estado actual y al símbolo terminal de la entrada (celda vacía). El mensaje de error es del tipo «Encontrado token a, se esperaba uno de los siguientes b, c, d».

El token encontrado se refiere al token de entrada y los esperados aquellos que tienen acciones asociadas para el estado de la cima de la pila.

La recuperación de errores en el análisis ascendente es compleja. Para desarrollar una estrategia de recuperación de errores se añade una nueva regla con un token especial ($B \rightarrow \cdot error$).

7. Clasificación de Gramáticas

- Libres de Contexto: Es el conjunto más amplio y pueden ser ambiguas.
- LR(k): No son ambiguas pueden analizarse con un método ascendente de orden k.
- LR(1): No son ambiguas pueden analizarse con un método ascendente lineal.
- LALR(1): Subconjunto de las LR(1) y pueden analizarse por el algoritmo LALR.
- SLR: Subconjunto de las LALR y pueden analizarse por el algoritmo SLR.
- LL(k): No son ambiguas y pueden analizarse por un método descendente con lookahead k. Subconjunto de LR(k).
- LL(1): No son ambiguas y pueden analizarse por un método descendente con lookahead 1. Subconjunto de LL(k) y LR(1).