Instituto Politécnico Nacional

Escuela Superior de Cómputo

Compiladores

Profr. Edgardo Adrián Franco Martínez

18 Análisis sintáctico III

Análisis descendente

efranco.docencia@gmail.com

http://computacion.cs.cinvestav.mx/~efranco

Contenido

- Clasificación de métodos de análisis sintáctico
- Análisis descendente
  - Análisis descendente recursivo
  - Análisis descendente predictivo
- Métodos deterministas
- Problemas del análisis descendente
  - Eliminación de la recursión por izquierda
  - Indeterminismo en las alternativas
- Observaciones

Clasificación de métodos de análisis sintáctico

	descendentes		ascendentes
no direccionales	Unger		CYK
direccionales	no deterministas	Predice/Concuerda 1° en anchura 1° en profundida	Desplaza/Reduce 1° en anchura 1° en profundida
	deterministas	Predice/Concuerda Gram. LL(1)	Desplaza/Reduce Gram. LR(k) LR(0), SLR(1), LALR(1)

Análisis descendente

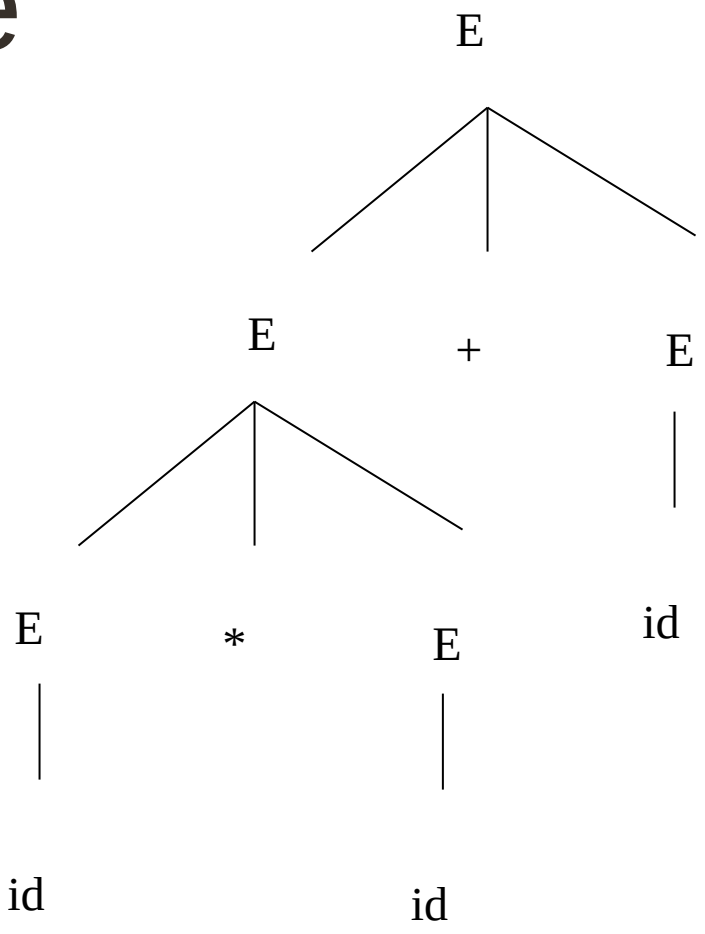
- ***Se parte del símbolo de inicio*** *de la gramática* que se coloca en la raíz y se va construyendo el árbol ***desde arriba hacia abajo***, hasta las hojas, *eligiendo la derivación que da lugar a una concordancia* con la cadena de entrada.

- Se basa en la idea de que predice una derivación y establece una concordancia con el símbolo de la entrada *(predict/match).*

Análisis descendente

- El análisis sintáctico descendente corresponde con un ***recorrido prefijo*** del árbol de análisis sintáctico.

"Primero expandimos el nodo que visitamos y luego procesamos los hijos".



Análisis descendente

Fundamento de los métodos descendentes:

Autómata predice/concuerda

- En cada paso del proceso de derivación de la cadena de entrada se realiza una *predicción de la posible producción a aplicar y se comprueba si* existe una *concordancia entre el símbolo actual en la entrada con el primer* terminal que se puede generar a partir de esa regla de producción, si existe esta concordancia se avanza en la entrada y en el árbol de derivación, en caso contrario se vuelve hacia atrás y se elige una nueva regla de derivación.

Análisis descendente

- **Ejemplo:** Supongamos la siguiente gramática que permite generar expresiones aritméticas:

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E' \mid -\ T\ E' \mid \epsilon$$
$$T \rightarrow F\ T'$$
$$T' \rightarrow *\ F\ T' \mid /\ F\ T' \mid \epsilon$$
$$F \rightarrow (\ E\ ) \mid \mathbf{num} \mid \mathbf{id}$$

- Para la entrada ***num + id * num.***

Crear el árbol de análisis sintáctico...

Análisis descendente

- Los métodos descendentes se caracterizan porque analizan la cadena de componentes léxicos de **izquierda a derecha**, obtienen la derivación más a la izquierda y el árbol de derivación se construye desde la raíz hasta las hojas.

Análisis descendente

- La especificación de la sintaxis de un lenguaje mediante gramáticas independientes del contexto permite *recursividad* y *estructuras anidadas*.

  - *P.g.:* sentencias **if-else** anidadas, paréntesis anidados en expresiones aritméticas, que no pueden ser representadas mediante expresiones regulares.

Análisis descendente

- La **recursividad** va a implicar:
  - Algoritmos de reconocimiento más complejos, que hagan uso de llamadas recursivas o usen explícitamente una pila, la *pila de análisis sintáctico.*
  - La estructura de datos usada para representar la sintaxis del lenguaje ha de ser también recursiva *(el árbol de análisis sintáctico)*, en vez de lineal *(caso de un vector de caracteres para almacenar los lexemas en el analizador léxico).*

Análisis Descendente Recursivo

- Se ejecuta un conjunto de procedimientos recursivos para procesar la entrada.
- A cada no terminal se le asocia un procedimiento para su análisis.
- Realiza varios exámenes a la cadena de entrada, para verificar si pertenece al lenguaje.

Análisis Descendente Recursivo

- Empareja cada una de las cadenas con la regla de producción.
- Hace una evaluación recursiva.
- Siguen la parte derecha de la regla.
- Se utilizan los conjuntos de predicción para determinar la regla.

Análisis Descendente Predictivo

- Es un tipo especial del análisis sintáctico descendente recursivo.

- Observa el siguiente *token* en la cadena de entrada para determinar cual va a ser la regla de producción a aplicar en la construcción del árbol de derivación.

Análisis Descendente Predictivo

- El símbolo de pre análisis determina sin ambigüedad el procedimiento seleccionado para cada entrada.

- Las gramáticas con análisis de una pasada son del tipo LL(k) las mas frecuentemente usadas son las de tipo LL1.

Métodos deterministas

- A partir de ahora nos restringimos al caso de métodos deterministas *(no hay vuelta atrás)* **teniendo en cuenta un solo símbolo de pre análisis** *(componente léxico que se esta analizando en la cadena de entrada en ese momento)*, sabemos exactamente en todo momento que producción aplicar. El símbolo de pre análisis se actualiza cada vez que se produce una concordancia.

Problemas del análisis descendente
(¿Qué producción elegir cuando hay varias alternativas?)

$$A \to \alpha \mid \beta \mid \ldots$$

Conjunto PRIMEROS(α): el conjunto de tokens que pueden comenzar cualquier frase derivable de α, $\alpha \in (V_T \cup V_{NT})^*$.

Conjunto SIGUIENTES(A): el conjunto de tokens que pueden aparecer después de un no-terminal A, $A \in V_{NT}$

Problemas del análisis descendente (Recursividad por la izquierda)

- La recursividad por izquierda da lugar a un bucle infinito de recursión.
  - **Problemas de indeterminismo** cuando varias alternativas en una misma producción comparten el mismo prefijo. No sabríamos cuál elegir.
  - Probaríamos una y si se produce un error haríamos ***backtracking***
  - Si queremos que el método sea determinista hay que evitarlas.

"Existen transformaciones de gramáticas para su eliminación. Estas transformaciones ***no modifican el lenguaje que se está reconociendo, pero si que*** *cambian el aspecto de la gramática, que es en general menos intuitiva".*

Eliminación de la recursividad izquierda

- Una gramática es ***recursiva por la izquierda*** *si tiene un no-terminal* ***A*** tal que existe una producción $A \rightarrow A\alpha$.

Algoritmo para eliminar recursividad a izquierdas:

- **primer paso:** se agrupan todas las producciones de A en la forma:

$$A \rightarrow A\alpha_1 \mid A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

donde ninguna β_i comienza con una A.

- **segundo paso:** se sustituyen las producciones de A por:

$$\begin{array}{l} A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A' \\ A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \epsilon \end{array}$$

Eliminación de la recursividad izquierda

- **Ejemplo:** Eliminar la recursividad a izquierdas de:

- $E \rightarrow E+T \mid T$
- $T \rightarrow T*F \mid F$
- $F \rightarrow (E) \mid id$

Eliminación de la recursividad izquierda

- **Ejemplo:** Eliminar la recursividad a izquierdas de:

- E → E+T | T
- T → T*F | F
- F → (E) | id

•Se identifica si cada una de las producciones que tienen la forma:
$A \rightarrow A\,\alpha \mid \beta$
•Las producción 1 y 2, tienen recursividad izquierda, por lo que:
$E \rightarrow E+T \mid T$
$A \rightarrow A\,\alpha \mid \beta$
Se igualan los símbolos según la formula:
$A = E$
$\alpha = +\,T$
$\beta = T$

E → TE'
E' → +TE'
E'→ ε

$$A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_m | \beta_1 | \beta_2 | \ldots | \beta_n$$

La recursión se elimina reemplazando la producción A por:

$$A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_n A'$$
$$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \epsilon$$

Eliminación de la recursividad izquierda

- **Ejemplo:** Eliminar la recursividad a izquierdas de:

- E → E+T | T
- T → T*F | F
- F → (E) | id

•A la producción T → T*F | F, también se le aplica el mismo proceso :
A = T
α = *F
β = F
Se aplica la fórmula 1:
T→ FT'
T'→ *FT'
T'→ ε

$$A \to A\alpha_1 | A\alpha_2 | \ldots | A\alpha_m | \beta_1 | \beta_2 | \ldots | \beta_n$$

La recursión se elimina reemplazando la producción A por:

$$A \to \beta_1 A' | \beta_2 A' | \ldots | \beta_n A'$$

$$A' \to \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \epsilon$$

Eliminación de la recursividad izquierda

- **Ejemplo:** Eliminar la recursividad a izquierdas de:

- E → E+T | T
- T → T*F | F
- F → (E) | id

•La gramática con la recursividad izquierda eliminada es:

E → TE'
E' → +TE'
E'→ λ
T→ FT'
T'→ *FT'
T'→ λ
F→ (E)| id

Indeterminismo en las alternativas (Factorización)

- Cuando dos alternativas para un no-terminal empiezan igual, no se sabe qué alternativa expandir. P.g.:

$$\begin{aligned} \text{Stmt} \rightarrow\ & \textbf{if}\ \text{E}\ \textbf{then}\ \text{Stmt}\ \textbf{else}\ \text{Stmt} \\ \mid\ & \textbf{if}\ \text{E}\ \textbf{then}\ \text{Stmt} \\ \mid\ & \textbf{otras} \end{aligned}$$

- Si en la entrada tenemos el *token* **if** no se sabe cual de las dos alternativas elegir para expandir . La idea es reescribir la producción para retrasar la decisión hasta haber visto de la entrada lo suficiente para poder elegir la opción correcta.

$$\begin{aligned} \text{Stmt} &\rightarrow \textbf{if}\ \text{E}\ \textbf{then}\ \text{Stmt}\ \text{Stmt'} \mid \textbf{otras} \\ \text{Stmt'} &\rightarrow \textbf{else}\ \text{Stmt} \mid \epsilon \end{aligned}$$

Indeterminismo en las alternativas (Factorización)

Algoritmo para factorizar la gramática:

- **primer paso:** para cada no-terminal A buscar el prefijo α más largo común a dos o más alternativas de A, $A \rightarrow \alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n|\gamma$
- **segundo paso:** Si $\alpha \neq \epsilon$, sustituir todas las producciones de A, de la forma $A \rightarrow \alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n|\gamma$, donde γ representa todas las alternativas de A que no comienzan con α por:

$$A \rightarrow \alpha\ A' \mid \gamma$$
$$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

Stmt → **if** E **then** Stmt **else** Stmt
| **if** E **then** Stmt
| **otras**

Stmt → **if** E **then** Stmt Stmt' | **otras**
Stmt' → **else** Stmt | ϵ

Indeterminismo en las alternativas (Factorización)

- Factorizar la siguiente gramática:
  - $P \rightarrow iEtP \mid iEtPeP \mid a$
  - $E \rightarrow b$

Indeterminismo en las alternativas (Factorización)

- Factorizar la siguiente gramática:
  - $P \rightarrow iEtP \mid iEtPeP \mid a$
  - $E \rightarrow b$

•*Se factoriza de la siguiente manera:*

$P \rightarrow iEtP\ (\lambda \mid eP) \mid a$

$P' \rightarrow eP \mid \lambda$

$E \rightarrow b$

•*Así que la gramática queda como:*

$P \rightarrow iEtPP' \mid a$

$P' \rightarrow eP \mid \lambda$

$E \rightarrow b$

Observaciones

- Es importante usar **métodos direccionales y deterministas** en compiladores.

- **El determinismo** por la necesidad de eficiencia (coste lineal frente a exponencial) y porque las herramientas para la generación de traductores asumen esta condición.

- **Los métodos direccionales** por la propia naturaleza secuencial en que se va procesando los símbolos del archivo fuente de izquierda a derecha.