UNIVERSIDAD
NACIONAL
DE COLOMBIA

Análisis sintáctico descendente

Parte 2

Felipe Restrepo Calle
ferestrepoca@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia
Sede Bogotá

1. **Condición LL(1) y LL(k)**
2. **Análisis sintáctico descendente recursivo - ASDR**
3. **Ejercicios**

Condición LL(1)

¿Qué ocurriría si ...

REGLA	*PREDICCIÓN*
$A \rightarrow \alpha_1$	{ ..., a, ... }
$A \rightarrow \alpha_2$	{ ..., a, ... }

... aparece el mismo símbolo en dos o más conjuntos de predicción del mismo no terminal?

Cuando aparezca el símbolo a en la cadena de entrada y el analizador tuviera que derivar el no terminal A, no sabría elegir qué regla aplicar.

Condición LL(1)

Es la condición que deben cumplir las gramáticas para que sea posible hacer un análisis lineal **O(N)** de izquierda a derecha (**Left**-to-right), usando la derivación por la izquierda (**Leftmost** derivation) y mirando solamente **1** símbolo de la entrada.

Condición LL(1)

→ **Una gramática *G* se dice cumple la condición LL(1), si para todos los no terminales, no existen símbolos comunes en los conjuntos de predicción de sus reglas.**

→ Si existe un símbolo común en los conjuntos de dos reglas del mismo no terminal, se puede decir que la gramática **no es LL(1).**

Condición LL(1) y ambigüedad

- La única forma de determinar que una gramática es ambigua es encontrando 2 o más árboles de derivación para una cadena de entrada.
- Si una gramática es LL(1), sabemos que no es ambigua.
- Si una gramática no es LL(1), no siempre es ambigua. Puede que sea ambigua, o puede que no.

Ejemplo:

$A \rightarrow B$ **uno**

$A \rightarrow$ **dos**

$B \rightarrow$ **dos**

No es LL(1), y no es ambigua.

Condición LL(k)

- Si en vez de mirar sólo un símbolo de la entrada se miraran **k** símbolos (**k > 1**), sería posible realizar el análisis en tiempo lineal para un conjunto más amplio de gramáticas.

Ejemplo:

A → *B* **uno**
A → **dos**
B → **dos**

No es LL(1), pero si es LL(2).

Condición LL(*)

- Existe una condición, LL(*), más general que la condición LL(k), y que permite realizar el análisis lineal con un conjunto todavía mayor de gramáticas.
- Un analizador **LL**(*), llamado analizador **LL-regular**, no está restringido a un número k de tokens finito para mirar hacia adelante en la cadena de entrada, toma decisiones de análisis al reconocer si los símbolos siguientes pertenecen a un lenguaje regular.
  - ✓ ANTLR: v3 LL(*)
    v4 Adaptative LL(*) – ALL(*)
  - ✓ Bison: LALR(1)

Características no LL(1)

Para asegurar que una gramática no es LL(1) se debe comprobar que no cumpla la condición LL(1). Sin embargo, hay algunas características que hacen que una gramática no sea LL(1):

1. **Recursividad por la izquierda**

$$A \rightarrow \mathbf{A} \ldots$$
$$A \rightarrow \ldots$$

2. **Factores comunes por la izquierda**

$$A \rightarrow \alpha \beta_1$$
$$\ldots$$
$$A \rightarrow \alpha \beta_m$$

3. **Ambigüedad**

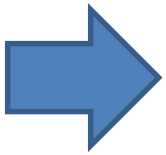
Transformaciones para conseguir la condición LL(1)

- La recursividad por la izquierda y los factores comunes por la izquierda se pueden eliminar de una gramática sin mucha dificultad.
- Sin embargo, la ambigüedad (si se ha detectado) no es fácil de eliminar y requiere un rediseño manual de la gramática.

Transformaciones para conseguir la condición LL(1)

- Eliminación de la recursividad por la izquierda:

$$A \rightarrow A\,\alpha_1 \quad \ldots \quad A \rightarrow A\,\alpha_n \quad A \rightarrow \beta_1 \quad \ldots \quad A \rightarrow \beta_m$$



$$A \rightarrow \beta_1\,A' \quad \ldots \quad A \rightarrow \beta_m\,A' \quad A' \rightarrow \alpha_1\,A' \quad \ldots \quad A' \rightarrow \alpha_n\,A' \quad A' \rightarrow \varepsilon$$

Transformaciones para conseguir la condición LL(1)

- Ejemplo de eliminación de recursividad por la izquierda:

$E \rightarrow E$ **opsuma** T

$E \rightarrow T$

$\alpha_1 =$ **opsuma** T

$\beta_1 = T$

$E \rightarrow T\ E'$

$E' \rightarrow$ **opsuma** $T\ E'$

$E' \rightarrow \varepsilon$

Modifica el AST

Transformaciones para conseguir la condición LL(1)

- Ejemplo de eliminación de recursividad por la izquierda:

Original

E → ***E*** **opsuma** *T*

E → *T*

Transformada

E → *T* ***E'***

E' → **opsuma** *T* ***E´***

E' → **ε**

Ejemplo: `4-5-10`

Se modifica el AST y la asociatividad del operador

Transformaciones para conseguir la condición LL(1)

- Ejemplo de eliminación de recursividad por la izquierda:

$E \rightarrow E_1$ **opsuma** T { E.trad = E_1.trad **opsuma**.trad T.trad }

$E \rightarrow T$ { E.trad = T.trad }

α_1 = **opsuma** T

β_1 = T

$E \rightarrow T$ { E'.i = T.trad } E' { E.trad = E'.s }

$E' \rightarrow$ **opsuma** T { E_1'.i = E'.i **opsuma**.trad T.trad} E_1' { E'.s = E_1'.s}

$E' \rightarrow \varepsilon$ { E'.s = E'.i }

Transformaciones para conseguir la condición LL(1)

- Ejemplo de eliminación de recursividad por la izquierda:

$E \rightarrow T \ \{ E'.i = T.trad \} \ E' \ \{ E.trad = E'.s \}$

$E' \rightarrow$ **opsuma** $T \ \{ E_1'.i = E'.i$ **opsuma**$.trad \ T.trad \} \ E_1' \ \{ E'.s = E_1'.s \}$

$E' \rightarrow \varepsilon \ \{ E'.s = E'.i \}$

Ejemplo: `4-5-10`

4-5-10	$E'.i = 4$	$E.trad = E'.s$
-5-10	$E_1'.i = 4 - 5$	$E'.s = E_1'.s$
-10	$E_1'.i = 4 - 5 - 10$	$E'.s = E_1'.s$
ε	$E'.s = E'.i = 4 - 5 - 10$	

Transformaciones para conseguir la condición LL(1)

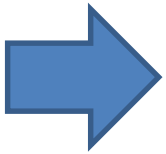
- Eliminación de factores comunes por la izquierda (factorización por la izquierda):

$A \rightarrow \alpha \beta_1$

...

...

$A \rightarrow \alpha \beta_m$



$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1$

...

$A' \rightarrow \beta_m$

Transformaciones para conseguir la condición LL(1)

- Ejemplo de eliminación de factores comunes por la izquierda (factorización por la izquierda):

Inst → **if** ***E*** **then** *Inst* **endif**
Inst → **if** ***E*** **then** *Inst* **else** *Inst* **endif**

Inst → **if** ***E*** **then** *Inst Inst'*
Inst' → **endif**
Inst' → **else** *Inst* **endif**

Analizador Sintáctico Descendente Recursivo - ASDR

- Una forma muy común de realizar un análisis lineal con una gramática LL(1) es usar un ASDR.
- Antes de programar el ASDR, es necesario calcular los conjuntos de predicción de todas las reglas y comprobar que la gramática es LL(1).

Analizador Sintáctico Descendente Recursivo - ASDR

Funcionamiento:

- En un ASDR, se debe diseñar una función para cada no terminal de la gramática.
- Se utiliza una función auxiliar para emparejar terminales.
- Cuando se tiene que derivar un no terminal, se llama a la función asociada a ese no terminal, y es la función la que se encarga de analizar el sublenguaje generado por dicho no terminal.

Ejemplo de implementación de un ASDR

A → *B* **uno**	{ **tres, cuatro**}
A → **dos**	{ **dos**}
B → **tres**	
B → **cuatro** *A*	

```
void A(){
    if (token() == TRES || token() == CUATRO) {
        B();
        emparejar(UNO);
    }
    else if (token() == DOS) {
        emparejar(DOS);
    }
    else errorSintaxis(TRES,CUATRO,DOS);
}
```

Ejemplo de implementación de un ASDR

```
...
void emparejar(TElementoLexico tokEsperado){
   if (token == tokEsperado)
      token = lexico.getNextToken();
   else
      errorSintaxis(tokEsperado);
}

public void main(...) {
   ...
   token = lexico.getNextToken();
   A(); // símbolo inicial de la gramática
   if (token != TOKFinArchivo)
      errorSintaxis(TOKFinArchivo);
}
```

Ejercicio 1

1. Dada la siguiente gramática:

S → A B C
S → D E
A → **dos** B **tres**
A → ε
B → B **cuatro** C **cinco**
B → ε
C → **seis** A B
C → ε
D → **uno** A E
D → B
E → **tres**

a) Eliminar la recursividad por la izquierda
b) Para la gramática resultante:
   - ✓ Calcular los conjuntos de PRIMEROS de cada no terminal.
   - ✓ Calcular los conjuntos de SIGUIENTES de cada no terminal.
   - ✓ Calcular los conjuntos de predicción de cada regla.
   - ✓ Decir si la gramática es LL(1) o no y por qué.
   - ✓ Implementar la función para cada no terminal en un ASDR.

Ejercicio 2

2. Dada la siguiente gramática:

S → B **uno**
S → **dos** C
S → ε
A → S **tres** B C
A → **cuatro**
A → ε
B → A **cinco** C **seis**
B → ε
C → **siete** B
C → ε

a) Calcular los conjuntos de PRIMEROS de cada no terminal.
b) Calcular los conjuntos de SIGUIENTES de cada no terminal.
c) Calcular los conjuntos de predicción de cada regla.
d) Decir si la gramática es LL(1) o no y por qué.
e) Implementar la función para cada no terminal en un ASDR.

Ejercicio 3

3. Dada la siguiente gramática:

S → A B C
S → S **uno**
A → **dos** B C
A → ε
B → C **tres**
B → ε
C → **cuatro** B
C → ε

a) Eliminar la recursividad por la izquierda
b) Para la gramática resultante:
   - ✓ Calcular los conjuntos de PRIMEROS de cada no terminal.
   - ✓ Calcular los conjuntos de SIGUIENTES de cada no terminal.
   - ✓ Calcular los conjuntos de predicción de cada regla.
   - ✓ Decir si la gramática es LL(1) o no y por qué.
   - ✓ Implementar la función para cada no terminal en un ASDR.