Analizadores sintácticos LR(0) y SLR

Teoría de Lenguajes

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Clase de Hoy

- Antes: Parsing **descendente** (LL(1), ELL)
  - Recursivos e iterativos
  - Generan árbol de derivación desde la raíz hasta las hojas
  - Simulan derivación a izquierda
- Hoy: Parsing **ascendente** (LR(k))
  - Construyen el árbol sintáctico desde las hojas hasta la raíz
  - Más expresivos que los descendentes
  - Más difíciles de construir

Analizadores Sintácticos LR(k)

- **L:** *left-to-right* (el parser lee el texto de izquierda a derecha)
- **R:** *rightmost derivation* (el parser usa la derivación más a la derecha)
- **k:** número de símbolos de lookahead

Tipos de Analizadores Sintácticos LR(k)

- LR(0)
- SLR(1): Simple LR(1)
- LALR(1): *Lookahead* LR(1)
- LR(1)

Poder expresivo

$$LR(0) \subset SLR(1) \subset LALR(1) \subset LR(1)$$

$\uparrow$ *menor* — $\uparrow$ *mayor*

Tipos de Analizadores Sintácticos LR(k)

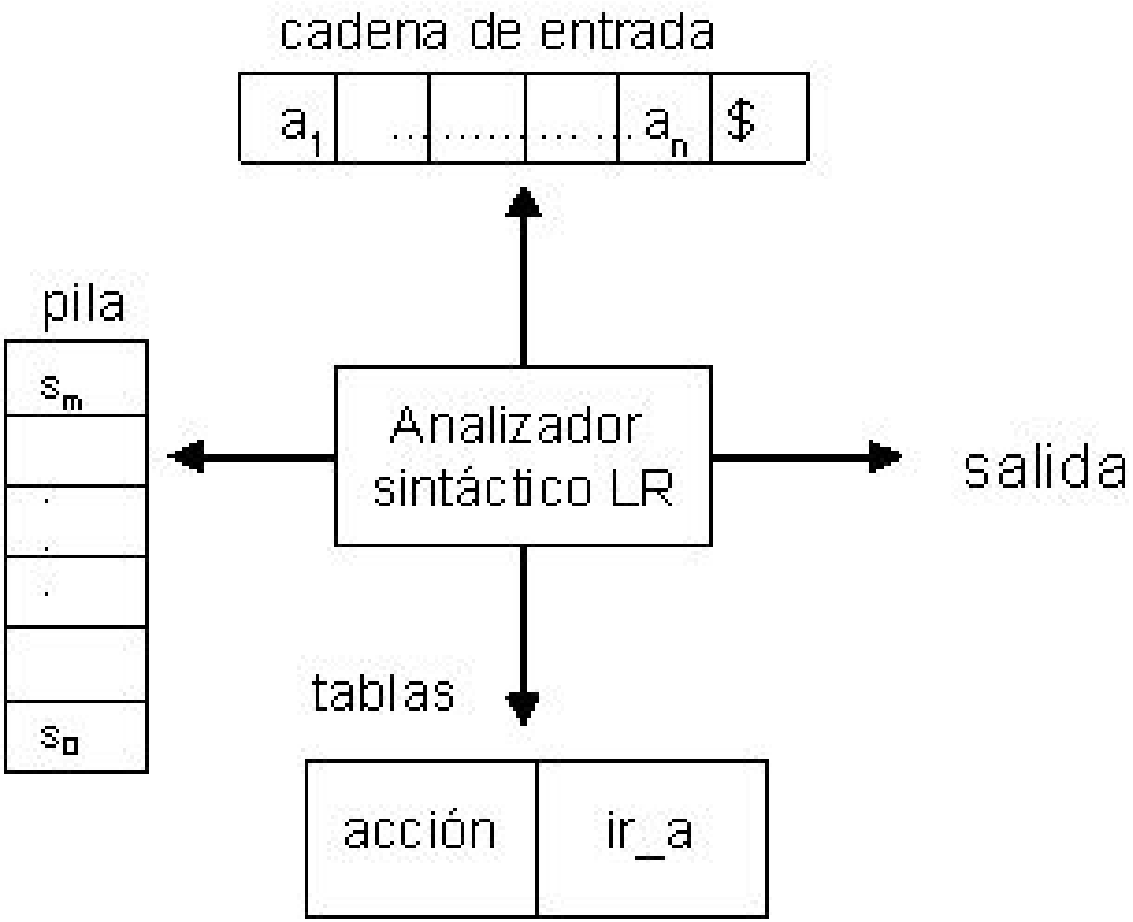
- LR(0) $\leftarrow$ HOY
- SLR(1): Simple LR(1) $\leftarrow$ HOY
- LALR(1): *Lookahead* LR(1)
- LR(1)

Poder expresivo

$$LR(0) \subset SLR(1) \subset LALR(1) \subset LR(1)$$

$\uparrow$ *menor* — $\uparrow$ *mayor*

Parsers LR(k) - Estructura General



Modelo de los analizadores sintácticos LR

Los parsers LR se representan como las tablas *ir_a* y *acción*:

- **Tabla acción** (Action): Mapea cada estado (del tope de la pila) y símbolo de entrada a una acción:
  - *desplazar i* (shift): desplazar y apilar el estado i.
  - *reducir* $A \rightarrow \beta$: reducir por la producción indicada.
  - *aceptar*: aceptar la cadena de entrada.
  - *error*: rechazar la cadena de entrada.

- **Tabla ir_a** (GOTO): Mapea un estado y un símbolo no terminal a un nuevo estado.

AFD para construir las tablas

¿Pero de qué estados estamos hablando?

Para armar las tablas *ir_a* y *acción*, es necesario construir un AFD.

Conceptos para la construcción del AFD

Item LR(0)

Producción con un pivote ($\bullet$) en alguna posición del lado derecho.

Representa hasta dónde se vio una producción en el proceso de análisis sintáctico, y cómo se espera que continúe la cadena de entrada.

Por ejemplo, para la producción $A \to BC$, existen estos ítems LR(0):

$A \to \bullet BC$
$A \to B \bullet C$
$A \to BC \bullet$

Para la producción $A \to \lambda$ el único ítem es: $A \to \bullet$

Conceptos para la construcción del AFD

Clausura de un Conjunto de Items

$J \leftarrow I$
repeat
  for ítem $A \rightarrow \alpha \cdot \mathbf{B} \beta \in J$, y producción $\mathbf{B} \rightarrow \gamma$ de G **do**
    agregar $B \rightarrow \cdot \gamma$ a J
  end for
until no se puedan agregar ítems a J
return J

Por ejemplo, si tenemos

$$A \rightarrow BaC$$
$$B \rightarrow b$$
$$C \rightarrow c$$

La clausura de $A \rightarrow \bullet BaC$ es

$$A \rightarrow \bullet BaC$$
$$B \rightarrow \bullet b$$

Conceptos para la construcción del AFD

Función *Ir_a*(*I* conjunto de ítems, *X* símbolo)

$J \rightarrow \emptyset$
for cada ítem $A \rightarrow \alpha \bullet X\beta$ en I **do**
  agregar $Clausura(\{A \rightarrow \alpha X \bullet \beta\})$ a J
end for
return J

Items de una gramática *G*

$C \leftarrow \{Clausura(\{S' \rightarrow \bullet S\})\}$
repeat
  for cada conjunto de ítems I en C y cada símbolo X **do**
    agregar $Ir\_a(I, X)$ a C
  end for
until no se puedan agregar más conjuntos de ítems a C

Construcción del AFD

Nuestro AFD va a estar definido por:

- El alfabeto: $V = V_N \cup V_T$.
- Los estados: conjuntos de ítems y sus clausuras.
- La función de transición entre un estado y otro, dada por la función *ir_a*.

AFD para armado de tablas

Construcción de las tablas

1. Aumentar la gramática G a G' agregando $S' \to S$ y cambiando el símbolo distinguido a S'
2. Construir los conjuntos de ítems LR(0) $C = \{I_0, ..., I_n\}$ (estados del AFD)

AFD para armado de tablas

Construcción de las tablas

3 Construir la tabla `acción`:

if $A \to \alpha \cdot a\beta$ está en I_i, y $\delta(I_i, a) = I_j$, $a \in V_T$ **then**
  asignar `desplazar j a accion[i, a]`
end if
if $A \to \alpha\cdot$ está en I_i, $A \neq S'$ **then**
  asignar `reducir` $A \to \alpha$ a `accion[i,a]`
    LR(0): para todos los terminales a
    SLR(1): para a en $Siguientes(A)$
end if
if $S' \to S\cdot$ está en I_i **then**
  asignar `aceptar` a `accion[i,$]`
end if

AFD para armado de tablas

Construcción del Autómata

4. Construir la tabla `ir_a`:
   Para cada no terminal A, `ir_a[i,A] = j` $\Leftrightarrow \delta(I_i, A) = I_j$
5. Las entradas vacías de la tabla acción son consideradas *error*
6. El estado inicial es el que contiene el ítem $S' \rightarrow \cdot S$

Gramáticas LR(0) – SLR(1)

Conflictos

Si la tabla `acción` tiene más de una entrada en algún casillero, entonces la gramática no es LR(0) / SLR(1).

Posibles conflictos:

- shift-reduce
- reduce-reduce

Parsers LR(K) - Algoritmo

apilar s_0
loop
  $s \leftarrow$ tope de la pila
  $a \leftarrow$ próximo símbolo apuntado en $w\$$
  if $accion[s, a] =$ desplazar s' **then**
    apilar s'
    avanzar al próximo símbolo de entrada
  else if $accion[s, a] =$ reducir $A \rightarrow \beta$ **then**
    sacar $|\beta|$ símbolos de la pila
    $s' \leftarrow$ tope de la pila
    apilar $ir\_a[s', A]$
  else if $accion[s, a] =$ aceptar **then**
    return
  else
    $error()$
  end if
end loop

Ejercicio 1

Sea G1 la siguiente gramática:

$$\begin{aligned} S &\rightarrow SA|A \\ A &\rightarrow (S)|() \end{aligned}$$

¿Es LR(0)? ¿Es SLR?

Analizar la siguiente cadena: (())()

Tabla acción LR(0) para G1

estado	(	)	$
0	desplazar 3		
1	desplazar 3		aceptar
2	reducir $S \to A$	reducir $S \to A$	reducir $S \to A$
3	desplazar 3	desplazar 6	
4	reducir $S \to SA$	reducir $S \to SA$	reducir $S \to SA$
5	desplazar 3	desplazar 7	
6	reducir $A \to ()$	reducir $A \to ()$	reducir $A \to ()$
7	reducir $A \to (S)$	reducir $A \to (S)$	reducir $A \to (S)$

Tabla ir_a LR(0) para G1

estado	S	A
0	1	2
1		4
2		
3	5	2
4		
5		4
6		
7		

Análisis de una cadena del lenguaje generado por G1

pila	entrada	acción
0	(())()$	desplazar 3
03	())()$	desplazar 3
033	))()$	desplazar 6
0336	)()$	reducir $A \to ()$
032	)()$	reducir $S \to A$
035	)()$	desplazar 7
0357	()$	reducir $A \to (S)$
02	()$	reducir $S \to A$
01	()$	desplazar 3
013	)$	desplazar 6
0136	$	reducir $A \to ()$
014	$	reducir $S \to SA$
01	$	aceptar

Derivación:

$S \Rightarrow SA \Rightarrow S() \Rightarrow A() \Rightarrow (S)() \Rightarrow (A)() \Rightarrow (())()$

Ejercicio 2

Sea G2 la siguiente gramática:

$$E \quad \rightarrow id|id(E)|E + id$$

¿Es LR(0)? ¿Es SLR?

Analizar la siguiente cadena: $id(id + id)$

Tabla acción SLR para G2

estado	id	(	)	+	$
0	desp2				
1				desp3	aceptar
2		desp4	r($E \to id$)	r($E \to id$)	r($E \to id$)
3	desp5				
4	desp2				
5			r($E \to E + id$)	r($E \to E + id$)	r($E \to E + id$)
6			desp7	desp3	
7			r($E \to id(E)$)	r($E \to id(E)$)	r($E \to id(E)$)

Siguientes(E) $= \{), +, \$\}$

Tabla ir_a SLR para G2

estado	E
0	1
1	
2	
3	
4	6
5	
6	
7	

Análisis de una cadena del lenguaje generado por G2

pila	entrada	acción
0	id(id+id)$	desplazar 2
02	(id+id)$	desplazar 4
024	id+id)$	desplazar 2
0242	+id)$	reducir $E \rightarrow id$
0246	+id)$	desplazar 3
02463	id)$	desplazar 5
024635	)$	reducir $E \rightarrow E + id$
0246	)$	desplazar 7
02467	$	reducir $E \rightarrow id(E)$
01	$	aceptar

Derivación:

$S \Rightarrow id(E) \Rightarrow id(E + id) \Rightarrow id(id + id)$

Bibliografía

A. Aho, R. Sethi, J. Ullman. Compiladores: Principios, técnicas y herramientas. Addison Wesley, 1990. *(El Libro del Dragón)*
Capítulo 4, secciones 4.5 y 4.7