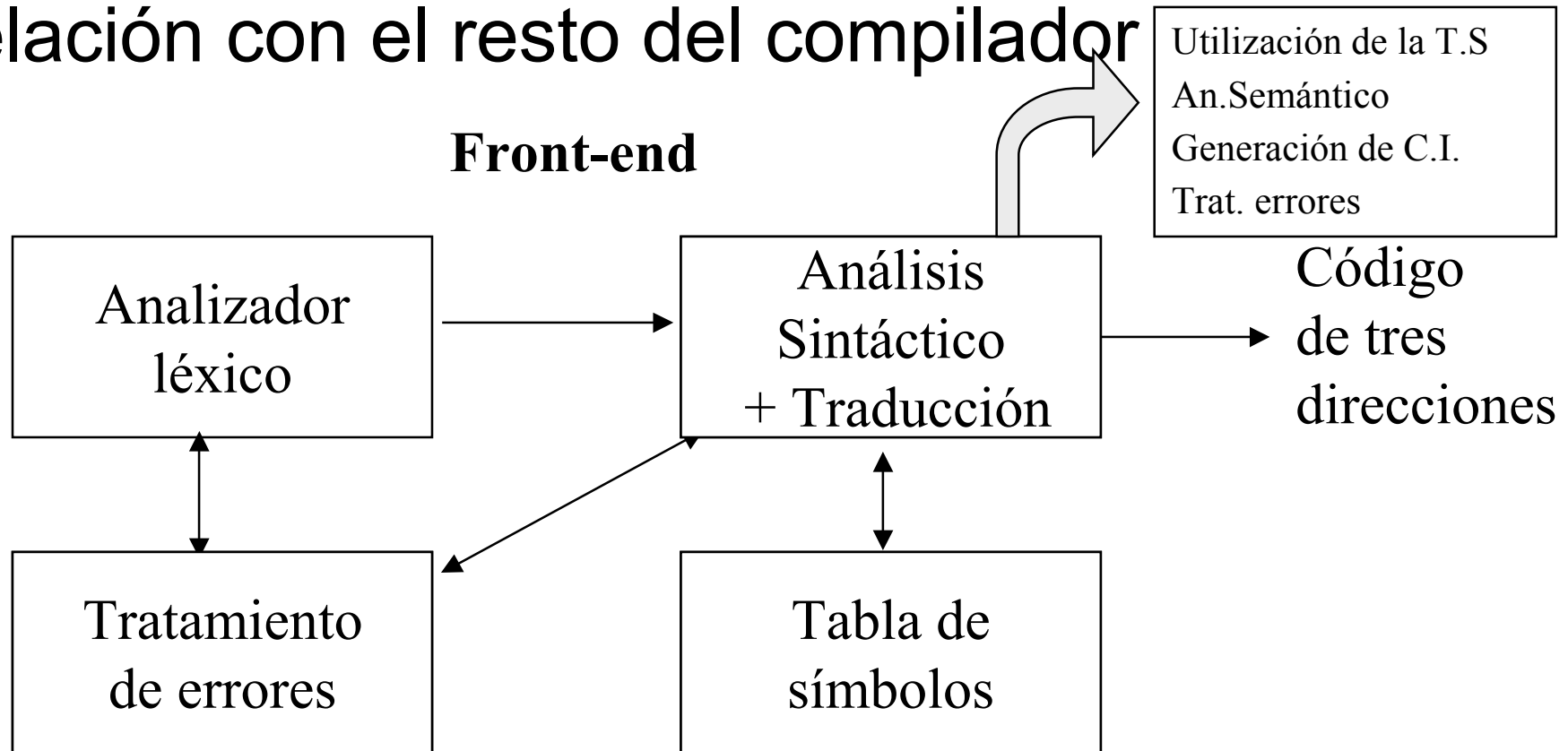


Tema 4: Análisis sintáctico

- 4.1 El analizador sintáctico
- 4.2 Definición sintáctica
- 4.3 Análisis descendente
- 4.4 Análisis ascendente
- 4.5 Tratamiento de errores sintácticos
- 4.6 YACC: generador de analizadores sintácticos

4.1 El analizador sintáctico

Su relación con el resto del compilador



4.1 El analizador sintáctico

- **Análisis ascendente y descendente**
 - » **Análisis descendente:**
 - Símbolo inicial -> hojas (tokens-terminales)
 - Análisis LL
 - » **Análisis ascendente:**
 - Cadena de tokens -> Símbolo inicial
 - Análisis LR

4.2 Definición sintáctica

- $G = (T, N, S, P)$
 - » Conjunto de símbolos terminales T (tokens)
 - » Conjunto de símbolos no terminales N
 - » Símbolo inicial S
 - » Conjunto de reglas de producción P :
 - $u \rightarrow v$, donde $u \in N$, $v \in (N \cup T)^*$

Definición sintáctica

- Derivación inmediata:
 - » $x \Rightarrow y$, donde $x, y \in (N \cup T)^*$
 - » $x = \alpha u \beta$
 - » $y = \alpha v \beta$
 - » $u \rightarrow v \in P$
- Derivación en n pasos

Definición sintáctica

- Derivación a izquierdas:
 - » siempre se sustituye el no terminal más a la izquierda
 - Análisis descendente
- Derivación a derechas:
 - » siempre se sustituye el no terminal más a la derecha
 - Análisis ascendente

4.2 Ambigüedad

- **Árbol de derivación**
 - » Está asociado a varias secuencias de derivación
 - » Una única secuencia de derivación a izdas.
- **Ambigüedad**
 - » Para al menos un cadena hay varios árboles de derivación
 - » Para al menos una cadena hay más de una derivación a izquierdas

Ambigüedad (II)

- Ejemplo:

- » $\text{sent} \rightarrow \text{if exp then sent}$

- » $\quad \quad | \text{if exp then sent else sent}$

if e1 then if e2 then s1 else s2

if e1 then if e2 then s1 else s2

4.3 Análisis descendente

- Análisis predictivo: condiciones LL(1)
- Análisis descendente recursivo
- Análisis descendente no recursivo

Análisis predictivo. PRIMERO

- Si X es un símbolo terminal, entonces $\text{PRIMERO}(X)$ es $\{X\}$.
- Si $X \rightarrow \xi$ es una producción, entonces añadir ξ a $\text{PRIMERO}(X)$.

Análisis predictivo. PRIMERO (II)

- Si X es un no terminal y $X \rightarrow Y_1 Y_2 \dots Y_k$ es una producción,
entonces añadir b a $\text{PRIMERO}(X)$ si $b \neq \xi$ y existe algún i tal que:
 - b pertenece a $\text{PRIMERO}(Y_i)$ y
 - ξ pertenece a $\text{PRIMERO}(Y_1), \dots, \text{PRIMERO}(Y_{i-1})$, esto es, si a partir de $Y_1 Y_2 \dots Y_{i-1}$ se puede derivar la cadena vacía.
- Finalmente, habrá que añadir ξ a $\text{PRIMERO}(X)$ si ξ pertenece a $\text{PRIMERO}(Y_i)$ para todo i entre 1 y k .

Análisis predictivo. SIGUIENTE

- Se define $\text{SIGUIENTE}(A)$, **para un no terminal A** , como el conjunto de terminales a que pueden aparecer inmediatamente a la derecha de A en una forma sentencial.

Análisis predictivo. SIGUIENTE (II)

- » 1. Introducir # en SIGUIENTE(S), donde S es el símbolo inicial de la gramática y # representa el final de la cinta de entrada
- » 2. Si existe una producción de la forma $A \rightarrow \alpha B \beta$, entonces introducir todos los símbolos de PRIMERO(β), excepto ξ , en SIGUIENTE(B).
- » 3. Si existe una producción de la forma $A \rightarrow \alpha B \beta$, entonces, si PRIMERO(β) contiene la ξ , introducir los símbolos de SIGUIENTE(A) en SIGUIENTE(B).

Condiciones LL(1)

- Una gramática G es LL(1) si y sólo si para cualquier par de producciones de la forma $A \rightarrow \alpha \mid \beta$ se cumplen las siguientes condiciones:
 - » 1.No existe ningún terminal a que pueda iniciar cadenas derivadas a partir de α y β .
 - » 2.Como máximo sólo una parte derecha, α ó β pueden derivar la ξ .
 - » 3.Si $\alpha \Rightarrow \xi$, entonces a partir de A no se deriva ninguna cadena que comience por algún terminal perteneciente a $\text{SIGUIENTE}(A)$.

Expresión de las condiciones LL(1)

● Para cualquier par de producciones de la forma $A \rightarrow \alpha \mid \beta$ se cumplen las siguientes condiciones:

- » 1. $\text{PRIMERO}(\alpha) \cap \text{PRIMERO}(\beta) = \emptyset$
- » 2. Si $\alpha \xRightarrow{*} \xi$, $\text{SIGUIENTE}(A) \cap \text{PRIMERO}(\xi) = \emptyset$

Recursividad a izquierdas

- Gramática recursiva a izquierdas
 - » $A \xRightarrow{+} A\alpha$, donde α pertenece a $(N \cup T)^*$.
- Recursividad a izquierdas inmediata
 - » $A \rightarrow A \alpha_1 | A \alpha_2 | \dots | A \alpha_m | \beta_1 | \beta_2 | \dots | \beta_n$
 - » Para i en $1..n$, β_i no comienza con A
- Estas reglas pueden transformarse en:
 - » $A \rightarrow \beta_1 A' | \beta_2 A' | \dots | \beta_n A'$
 - » $A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots | \alpha_m A' | \xi$

Eliminación de la recursividad a izquierdas

para i desde 1 hasta n hacer

para j desde 1 hasta i -1 hacer

Remplazar cada $A_i \rightarrow A_j \gamma$

por las producciones $A_i \rightarrow \delta_1 \gamma \mid \dots \mid \delta_k \gamma$,

donde $A_j \rightarrow \delta_1 \mid \dots \mid \delta_k$ son todas las A_j -prod.;

fin para

Eliminar la recursividad a izquierdas inmediata
sobre las A_i -producciones

fin para

Factorización a izquierdas de una gramática

Buscar prefijo común más largo a las A-producciones

Si $\alpha \neq \xi$,

Sustituir $A \rightarrow \alpha\beta_1 \mid \dots \mid \alpha\beta_n \mid \gamma$

por $A \rightarrow \alpha A' \mid \gamma$

donde $A' \rightarrow \beta_1 \mid \dots \mid \beta_n$

y γ representa todas las producciones que no tienen como prefijo α

Análisis descendente predictivo recursivo

- Un procedimiento por cada no terminal
- Un parámetro por cada atributo
- Selección de la regla a aplicar: lookahead
- Gramática LL(1)

Análisis descendente predictivo no recursivo

- Pila de análisis: elementos por analizar
- Tabla de análisis sintáctico:
 - » matriz(no terminales, terminales)
- Entrada para un no terminal A y un terminal a :
 - » regla a aplicar ó
 - » error

Construcción de tablas de análisis sintáctico

- » Entrada: Una gramática G
- » Salida: La tabla de análisis sintáctico M

Para cada no terminal A de la gramática G

Para cada regla de la forma $A \rightarrow \alpha$

Para cada terminal a (incluido la marca de final $\#$)

Si $a \in \text{PRIMERO}(\alpha)$ o $\xi \in \text{PRIMERO}(\alpha)$ y $a \in \text{SIGUIENTE}(A)$

entonces $M[A, a] := A \rightarrow \alpha$

Fin si

Fin Para cada

Fin Para cada

Fin Para cada

Construcción de tablas de análisis sintáctico (II)

- Las entradas no definidas corresponden a situaciones de error
- Si en el proceso de construcción de la tabla se intenta redefinir una entrada de la tabla, la gramática no cumple las condiciones LL(1)

Algoritmo de análisis sintáctico descendente no recursivo

- **Entrada:** La tabla de análisis sintáctico M y la entrada a analizar
- **Salida:** Si la entrada pertenece al lenguaje definido por la gramática a partir de la que se ha construido la tabla de análisis M , la secuencia de derivación a izquierdas de la entrada; si no un aviso de error

Algoritmo de análisis sintáctico descendente no recursivo (II)

Principio

Lookahead apunta al primer símbolo de la entrada

La pila de análisis ha sido inicializada con S#

Repetir

X := cima de la pila

Si X es un terminal **entonces** -- emparejar (X)

Si lookahead = X **entonces**

 Desempilar X; Avanzar en la entrada;

si no Avisar del error

fin si

si no -- X es un no terminal

Algoritmo de análisis sintáctico descendente no recursivo (III)

si no -- X es un no terminal

Si $M(X, \text{lookahead}) = X \rightarrow \alpha$ **entonces**

Desempilar X ;

Empilar los símbolos de α (de dcha. a izda.)

-- ξ no es un símbolo

si no Avisar del error y Recuperar el control

Fin si

Fin si

Hasta ($X = \#$ y $\text{lookahead} = \#$)

Final

4.4 Análisis ascendente

- **Por precedencia de operadores**

- » Gramáticas que no tienen producciones $\rightarrow \xi$
- » Gramáticas en que no hay dos no-terminales adyacentes en la parte derecha de una regla

- **Analizadores LR**

- » Análisis por reducción-desplazamiento (R/D)
- » Construyen el árbol de análisis partiendo de las hojas hasta llegar a la raíz
- » Proceso similar a la inversa de la derivación a derechas de la cadena de entrada

Analizadores LR

- » Un *handle* de una cadena es una subcadena que coincide con la parte derecha de una regla y cuya **reducción** por el no terminal que se encuentra a la izquierda en dicha regla constituye un paso correcto en la **inversa de la derivación a derechas**.
- » Para obtener la inversa de la derivación a derechas bastará con ir localizando *los handles* en las formas sentenciales obtenidas sucesivamente.

Ejemplos

- **Ejemplo 1**

» $S \rightarrow aABe$
» $A \rightarrow Abc \quad | \quad b$
» $B \rightarrow d$
» $S \Rightarrow aABe \Rightarrow aAde \Rightarrow aAbcde \Rightarrow aAbcbcde \Rightarrow abbcbcde$

- **Ejemplo 2**

» $A \rightarrow BC$
» $B \rightarrow bb$
» $C \rightarrow cA \quad | \quad d$
» $A \Rightarrow BC \Rightarrow BcA \Rightarrow BcBC \Rightarrow BcBd \Rightarrow Bcbbd \Rightarrow bbcbbd$

- **Ejemplo 3**

» $A \rightarrow BC$
» $B \rightarrow bb \quad | \quad db$
» $C \rightarrow cA \quad | \quad d$
» $A \Rightarrow BC \Rightarrow BcA \Rightarrow BcBC \Rightarrow BcBd \Rightarrow Bcdbd \Rightarrow bbcdbd$

Implementación de un analizador por R/D usando una pila

- Inicialmente la pila está vacía (o contiene una marca) y en la entrada está la cadena a analizar. El objetivo es sintetizar en el tope de la pila el símbolo inicial de la gramática coincidiendo con el momento en que se alcanza el final de la cadena de entrada.

Pila	Entrada
-------------	----------------

#	$\omega\#$
---	------------

...	
-----	--

#S	#
----	---

- El proceso de análisis se basa en el uso de cuatro operaciones: reducir, desplazar, aceptar y error.

Ejemplo 1

Pila	Entrada	Acción
#	abbcbcde#	desplazar
#a	bbcbcdde#	desplazar
#ab	bcbcde#	reducir
#aA	bcbcde#	desplazar
#aAb	cbcdde#	desplazar
#aAbc	bcde#	reducir
#aA	bcde#	desplazar
#aAb	cde#	desplazar
#aAbc	de#	reducir
#aA	de#	desplazar
#aAd	e#	reducir
#aAB	e#	desplazar
#aABe	#	reducir
#S	#	aceptar

Ejemplo 2

Pila	Entrada	Acción
#	bbcbbd#	desplazar
#b	bcbbd#	desplazar
#bb	cbbd#	reducir
#B	cbbd#	desplazar
#Bc	bbd#	desplazar
#Bcb	bd#	desplazar
#Bcbb	d#	reducir
#BcB	d#	desplazar
#BcBd	#	reducir
#BcBC	#	reducir
#BcA	#	reducir
#BC	#	reducir
#A	#	aceptar

Ejemplo 3

- 1. S → begin | L_ | end
- 2. L_ | → , | L_ |
- 3. | ξ
- 4. | → ID_ASIG := REF_TABLA
- 5. ID_ASIG → id
- 6. REF_TABLA → ID_TABLA [entero]
- 7. ID_TABLA → id

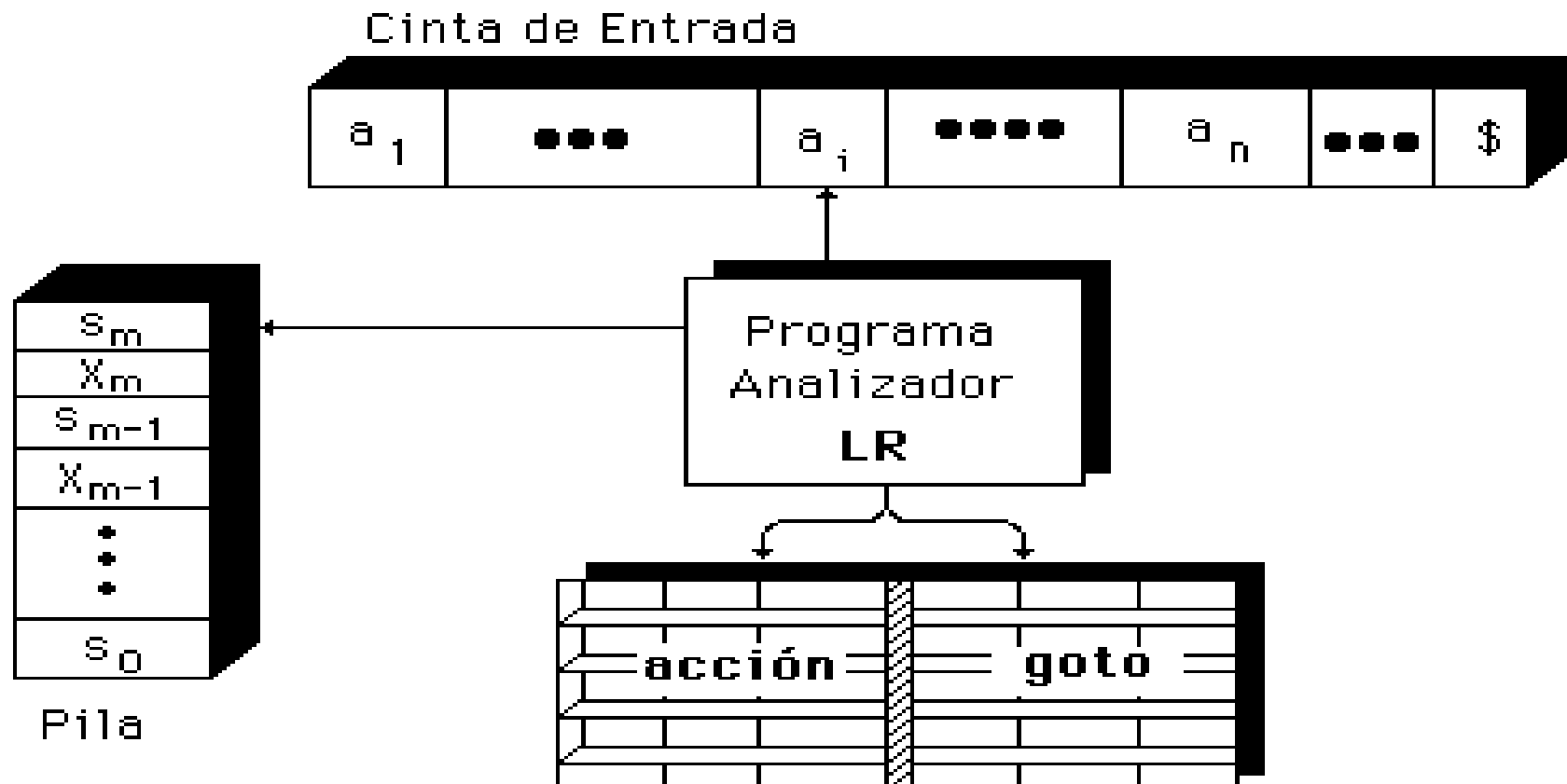
Ejemplo 3

Pila	Entrada	Acción
#	begin id1 := id2 [entero1] end#	empilar
# begin	id1 := id2 [entero1] end#	empilar
# begin id1	:= id2 [entero1] end#	reducir
# begin ID_ASIG	:= id2 [entero1] end#	empilar
# begin ID_ASIG:=	id2 [entero1] end#	empilar
# begin ID_ASIG:=id2	[entero1] end#	reducir
# ... :=ID_TABLA	[entero1] end#	empilar
# ... :=ID_TABLA[entero1] end#	empilar
# ...:=ID_TABLA[entero1] end#	empilar
# ...:=ID_TABLA[entero1]	end#	reducir
# ...:=REF_TABLA	end#	reducir
# begin I	end#	reducir
# begin I L_I	end#	empilar
# begin I L_I end	#	reducir
# S	#	aceptar

Análisis LR

- » Permiten reconocer una amplia gama de los lenguajes descritos por medio de gramáticas independientes del contexto
- » Es el método de análisis por R/D más general de los que no usan vuelta-atrás, además permite una implementación eficiente
- » La clase de gramáticas a partir de las cuales es posible construir un analizador descendente predictivo es prácticamente un subconjunto de la clase de gramáticas LR
- » Existen tres técnicas de construcción de tablas de análisis LR: SLR, CLR y LALR

Análisis LR



Análisis LR

configuración $(s_0 X_1 s_1 \dots X_m s_m, a_i \dots a_n \#)$

$X_1 X_2 \dots X_m a_i a_{i+1} \dots A_n$

movimiento: símbolo de entrada actual a_i
estado en el tope de la pila s_m

acción $[s_m, a_i]$

1. Si **acción** $[s_m, a_i] = \text{empilar } s$

$(s_0 X_1 s_1 \dots X_m s_m a_i s, a_{i+1} \dots a_n \#)$

2. Si **acción** $[s_m, a_i] = \text{reducir } A \rightarrow b$

$(s_0 X_1 s_1 \dots X_{m-r} s_{m-r} A s, a_i \dots a_n \#)$

donde $s = \text{goto } [s_{m-r}, A]$ y r es la longitud de b

3. Si **acción** $[s_m, a_i] = \text{aceptar}$ indica el final del análisis

4. Si **acción** $[s_m, a_i] = \text{error}$, el analizador ha descubierto un error y llama a la rutina de recuperación de errores

Algoritmo de análisis LR

Entrada: Una cadena de entrada w y una tabla de análisis LR con las funciones *acción* y *goto* para la gramática G .

Salida: Si w pertenece a $L(G)$ un *análisis ascendente*, en caso contrario un mensaje **de error**.

Método: Inicialmente, el analizador almacena s_0 en la pila, donde s_0 es el estado inicial, y $w\#$ es el contenido de la entrada. El analizador ejecuta el programa siguiente hasta que la cadena es reconocida, o hasta que se detecta un error.

Algoritmo de análisis LR

inicializar **ip** con el primer símbolo de **w#**;

repetir

*Sea **s** el estado en el tope de la pila y **a** el símbolo apuntado por **ip**;*

si acción [**s**, **a**] = desplazar **s'** **entonces**

empilar **a** , **s'** en el tope de la pila;

avanzar **ip** al siguiente símbolo de la entrada

Algoritmo de análisis LR

si no

si acción $[s, a] = \text{reducir } A \rightarrow b$ **entonces**

desempilar $2 * |b|$ símbolos de la pila;

sea **s'** ahora el estado del tope de la pila;

empilar **A**, y empilar *goto* $[s', A]$;

si no

si acción $[s, a] = \text{aceptar}$ **entonces salir**

si no error ()

fin si

fin repetir

Algoritmo de análisis LR

ESTADO	acción						goto		
	id	+	*	()	\$	E	T	F
0	s ₅				s ₄		1	2	3
1		s ₆				acc			
2		r ₂	s ₇		r ₂	r ₂			
3		r ₄	r ₄		r ₄	r ₄			
4	s ₅				s ₄		8	2	3
5		r ₆	r ₆		r ₆	r ₆			
6	s ₅				s ₄			9	3
7	s ₅				s ₄				10
8		s ₆				s ₁₁			
9		r ₁	s ₇		r ₁	r ₁			
10		r ₃	r ₃		r ₃	r ₃			
11		r ₅	r ₅		r ₅	r ₅			

$E \rightarrow E + T$ (1)

$| T$ (2)

$T \rightarrow T * F$ (3)

$| F$ (4)

$F \rightarrow (E)$ (5)

$| id$ (6)

Movimientos del analizador LR

PILA	INPUT	ACCIÓN
(1) 0	id * id + id #	empilar
(2) 0 id 5	* id + id #	reducir por $F \rightarrow id$
(3) 0 F 3	* id + id #	reducir por $T \rightarrow F$
(4) 0 T 2	* id + id #	empilar
(5) 0 T 2 * 7	id + id #	empilar
(6) 0 T 2 * 7 id 5	+ id #	reducir por $F \rightarrow id$
(7) 0 T 2 * 7 F 10	+ id #	reducir por $T \rightarrow T * F$
(8) 0 T 2	+ id #	reducir por $E \rightarrow T$
(9) 0 E 1	+ id #	empilar
(10) 0 E 1 + 6	id #	empilar
(11) 0 E 1 + 6 id 5	#	reducir por $F \rightarrow id$
(12) 0 E 1 + 6 F 3	#	reducir por $T \rightarrow F$
(13) 0 E 1 + 6 T 9	#	reducir por $E \rightarrow E + T$
(14) 0 E 1	#	aceptar

Gramáticas LR

- Una gramática será LR si es posible construir para ella una tabla de análisis que dirija el proceso de análisis por reducción-desplazamiento
- Un analizador LR **no** necesita recorrer toda la pila para saber si en el tope de ésta se encuentra un *handle*
- EL ESTADO EN EL TOPE DE LA PILA CONTIENE LA INFORMACIÓN NECESARIA
- La función GOTO de una tabla de análisis LR es básicamente un autómata finito, capaz de reconocer la aparición de un handle en el tope de la pila
- Otra fuente de información es el lookahead
- Las gramáticas LR pueden describir más lenguajes que las LL

4.5 Tratamiento de errores sintácticos

- Detección del error
- Aviso del error
- Recuperación del error

Recuperación del error

- Modo de pánico
- Recuperación a nivel de frase
- Producciones de error
- Reparación del error

Modo “pánico”

- “Saltar” hasta encontrar un punto donde se pueda continuar el análisis
- Sencillo
- Correcto eligiendo conjuntos de sincronización adecuados
- Desventaja: omite el análisis de parte de la entrada

Múltiples mensajes de error

- Se exigirá analizar correctamente n tokens hasta generar un nuevo mensaje de error:
 - » al encontrar un error se pone un contador a cero
 - » al tener un mensaje de error, si el contador no ha llegado a n , no se escribe
 - » se incrementa el contador al avanzar un token

Modo de pánico (análisis descendente recursivo ADR)

- Un procedimiento por no terminal
- Por cada elemento de la parte derecha:
 - » una llamada al procedimiento correspondiente si es un no terminal
 - » una llamada al procedimiento *match* (emparejar) si es un terminal

Modo de pánico (ADR)

- Dos aspectos:
 - » (A) Cuándo se detectan los errores
 - » (B) Qué acciones realizar cuando se detecta un error

A. En qué momento pueden detectarse los errores (ADR)

1. Al principio de la ejecución de un procedimiento, caso de que el símbolo en curso no sea ninguno de los que se esperan en ese momento :

procedimiento no_terminal

empezar

Si el símbolo en curso no es ninguno de los que se esperan

entonces ...

...

final

A. En qué momento pueden detectarse los errores (ADR)

2. Al final de la ejecución de un procedimiento :

procedimiento no_terminal
empezar

...

Si el símbolo en curso no es ninguno de los que se esperan
entonces ...

final

A. En qué momento pueden detectarse los errores (ADR)

3. En el procedimiento emparejar:

procedimiento emparejar(simbolo:t_terminal)

empezar

Si símbolo no coincide con el lookahead **entonces** ...

final

B. Qué acciones realizar cuando se detecta un error (ADR)

- Avisar del error
- Recuperación del error, es decir, posibilitar que el análisis continúe, con el objetivo de seguir detectando errores

B.1 Al principio de la ejecución de un procedimiento (ADR)

procedimiento no_terminal

empezar

Si el símbolo en curso no es ninguno de los que se esperan

entonces

 avisar del error

 saltar hasta algún punto que permita continuar el análisis

fin si

Si el símbolo en curso es uno de los que se esperan

entonces

 TRATAMIENTO NORMAL

fin si

final

B.1 Al principio de la ejecución de un procedimiento (ADR)

- “Algún punto que permita continuar el análisis”:
 - »El conjunto PRIMERO del no terminal en cuestión
 - »El conjunto SIGUIENTE del no terminal en cuestión
 - »Otros símbolos, no pertenecientes a los conjuntos anteriores, pero que representan fronteras que no se quieren traspasar

B.2 En el procedimiento emparejar (ADR)

1. Actuar como si el símbolo encontrado hubiera coincidido con el esperado:

procedimiento emparejar (simbolo: t_terminal)

empezar

Si símbolo no coincide con el lookahead

entonces avisar del error

fin si

obtener siguiente token

final

B.2 En el procedimiento emparejar (ADR)

2. Actuar considerando que el símbolo esperado se le ha olvidado al programador y, en consecuencia, emparejar el símbolo esperado con "la nada"

Procedimiento emparejar(simbolo:t_terminal)
empezar

Si símbolo no coincide con el lookahead

entonces avisar del error

si no obtener siguiente token

fin si

final

B.2 En el procedimiento emparejar (ADR)

3. Análisis por casos aprovechando ambas opciones.

procedimiento emparejar(simbolo:t_terminal)

empezar

si símbolo no coincide con el lookahead

entonces avisar del error

si el símbolo esperado es ... y el que viene es ...

entonces obtener siguiente token

fin si

si no obtener siguiente token

fin si

final

Modo de pánico (análisis descendente con tabla)

Similar al recursivo. Casos:

- » a) El tope de la pila es un terminal \neq token de entrada. Posibilidades:
 - emparejar la pila con la entrada
 - avanzar en la entrada ('inserta' el contenido de la pila)
 - estrategia mixta, dependiente de los terminales

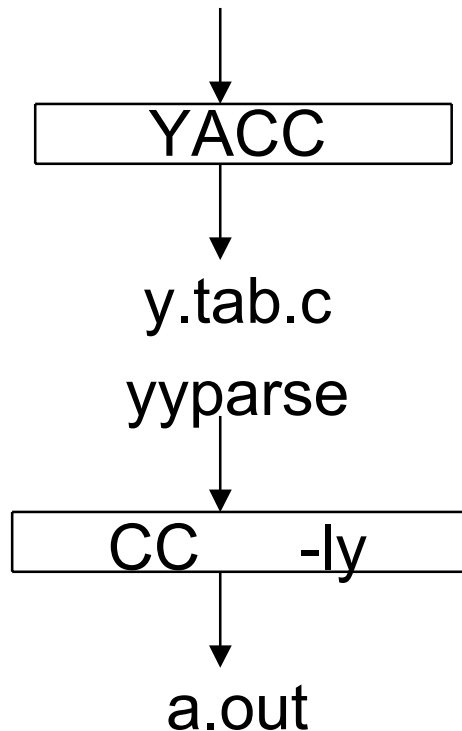
Modo de pánico (análisis descendente con tabla)

- » b) El tope de la pila es un no terminal A y el token de entrada no puede aparecer al analizar el no terminal.
- » Tratamiento. Saltar hasta:
 - Primero(A) -> analizar A
 - Siguiente(A), Posteriores(A) -> acabar el análisis de A

4.6 YACC: generador de analizadores sintácticos

COMPILADOR DE COMPILADORES
GENERADOR DE ANALIZADORES
GENERADOR DE TRADUCTORES

especificación YACC



Especificación YACC

declaraciones

%%

reglas

%%

programas

declaraciones

- » Código C: Decl. de variables,...
- » Declaración de nombres de token %token
- » Prioridades %left,...
- » Símbolo inicial %start

Especificación YACC

reglas

lista de:

regla de producción ACCION

A : PD1 {código C};

A : PD2;

A : PD1 {código C}

| PD2;

PDi Parte derecha de la regla

Acciones

- Pueden obtener valores y utilizar los obtenidos por otras acciones
- Pueden aparecer en puntos intermedios de la parte derecha de una regla
- {}
- Cálculo y uso de atributos a través de: ##, #1, #2 y #3
- Acción por defecto: ## = #1
- Manejo de objetos globales

Analizador léxico

- yylex
- yylval (atributos)
- Referencias a nombres de tokens
- Definición previa en la sección de declaraciones o posterior en la sección de programas
- nombres de tokens <> Palabras reservadas de C
- Trabajo en combinación con LEX

Funcionamiento del analizador generado

- Cuando se reduce se ejecuta la acción asociada a la regla, se desempilan los atributos asociados a los símbolos de la parte derecha y se empilan los atributos del símbolo de la parte izquierda

Ambigüedad y conflictos

- YACC genera el analizador aunque la gramática no sea LALR
- Reglas para la resolución de conflictos:
 - Conflicto R/D \rightarrow D
 - Conflicto R/R \rightarrow Reducción por medio de la primera regla en la secuencia de entrada
- YACC avisa del número de conflictos detectados y puede, opcionalmente, documentarlos

Precedencia

expr: expr Op expr

expr: Op-unario expr

- Definición de prioridad de operadores:
 - » %left '+' '-'
 - » %left '*' '/'