IC-5701
Compiladores e Intérpretes

Profesor:
Ing. Allan Rodríguez Dávila

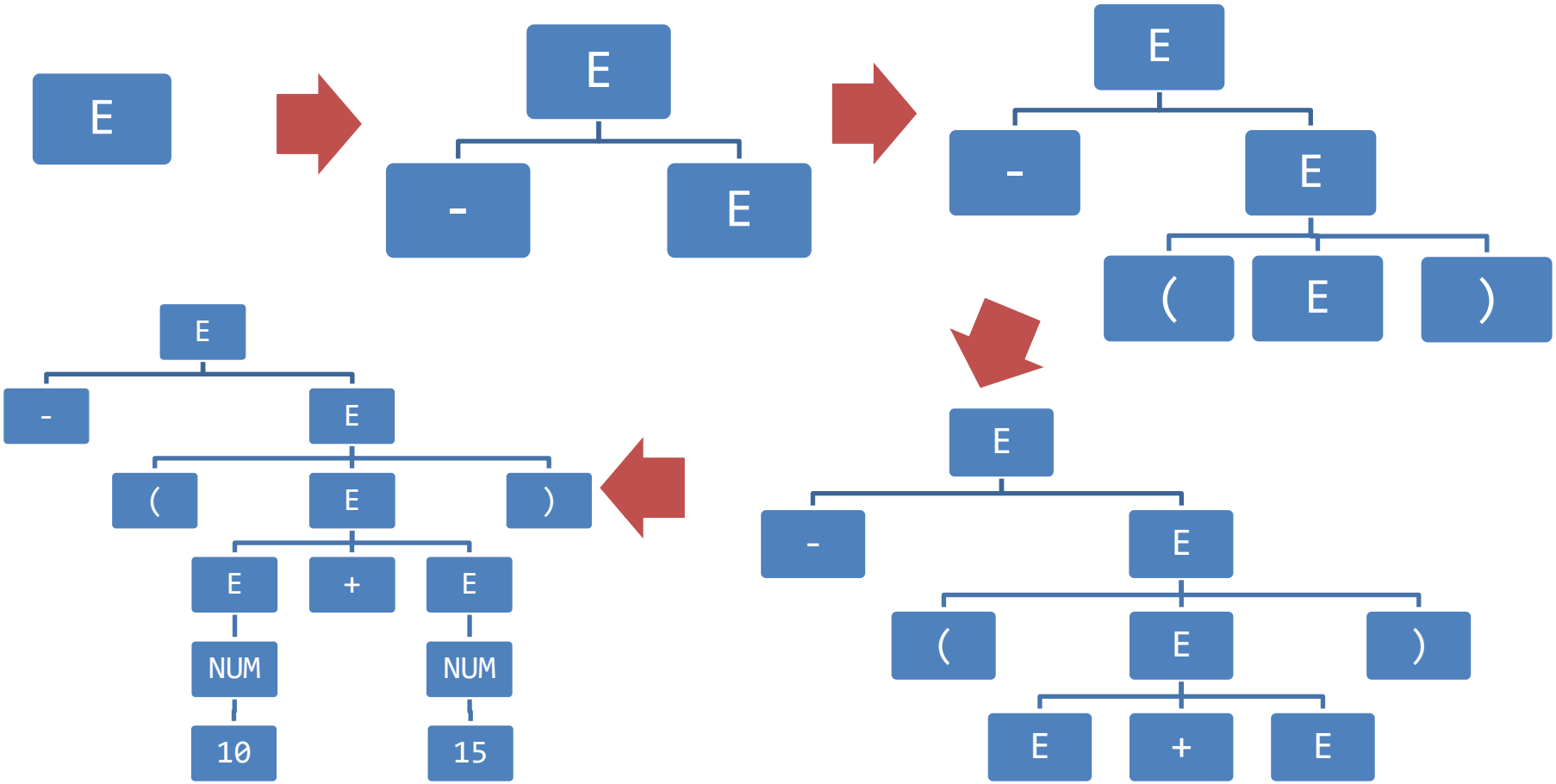
Reconocimiento Descendente

Reconocimiento Descendente

- Se inicia la construcción iniciando con la raíz etiquetada con el símbolo inicial de la gramática.
- A partir de los nodos padre se construyen sus nodos hijos con los símbolos del cuerpo de la producción que calzan con la entrada.

Construcción Descendente

-(10 + 15)



Reconocimiento Descendente

- Se realiza un scaneo de izquierda a derecha.
- El símbolo de lectura en la cadena de entrada se le conoce cómo preanálisis.
  - Por lo general se busca un terminal.
- Se construye el árbol en preorden.
- Se busca una derivación por la izquierda.

Descenso recursivo

- Conjunto de procedimientos, cada uno para cada no terminal.
- Se inicia con el procedimiento del no terminal inicial.
- Finaliza cuando se procesa toda la cadena.

TEC

Descenso recursivo

```
void analizadorSintactico() {
   Elegir una producción A, A → X1 X2 … Xk ;
  for ( i = 1 a k ) {
    if ( Xi es un no terminal )
      llamar al procedimiento Xi();
    else if (Xi es igual al símbolo de entrada actual a)
       avanzar la entrada hasta el siguiente símbolo;
    else /* ha ocurrido un error */;
  }
}
```

Descenso recursivo

- Puede requerir rastreo hacia atrás (backtracking)
- Se debe probar las diversas producciones de un mismo no terminal.
- Si se llega a fallo se hace backtracking o se evalúa la siguiente producción.
- Analizador sintáctico predictivo no necesitan rastreo hacia atrás.

TEC

FIRST

- Función Auxiliar que ayuda a elegir la producción correcta.
- FIRST(α) es la función, en donde α es cualquier cadena de símbolos gramaticales, del conjunto de terminales que empiezan las cadenas derivadas a partir de α.

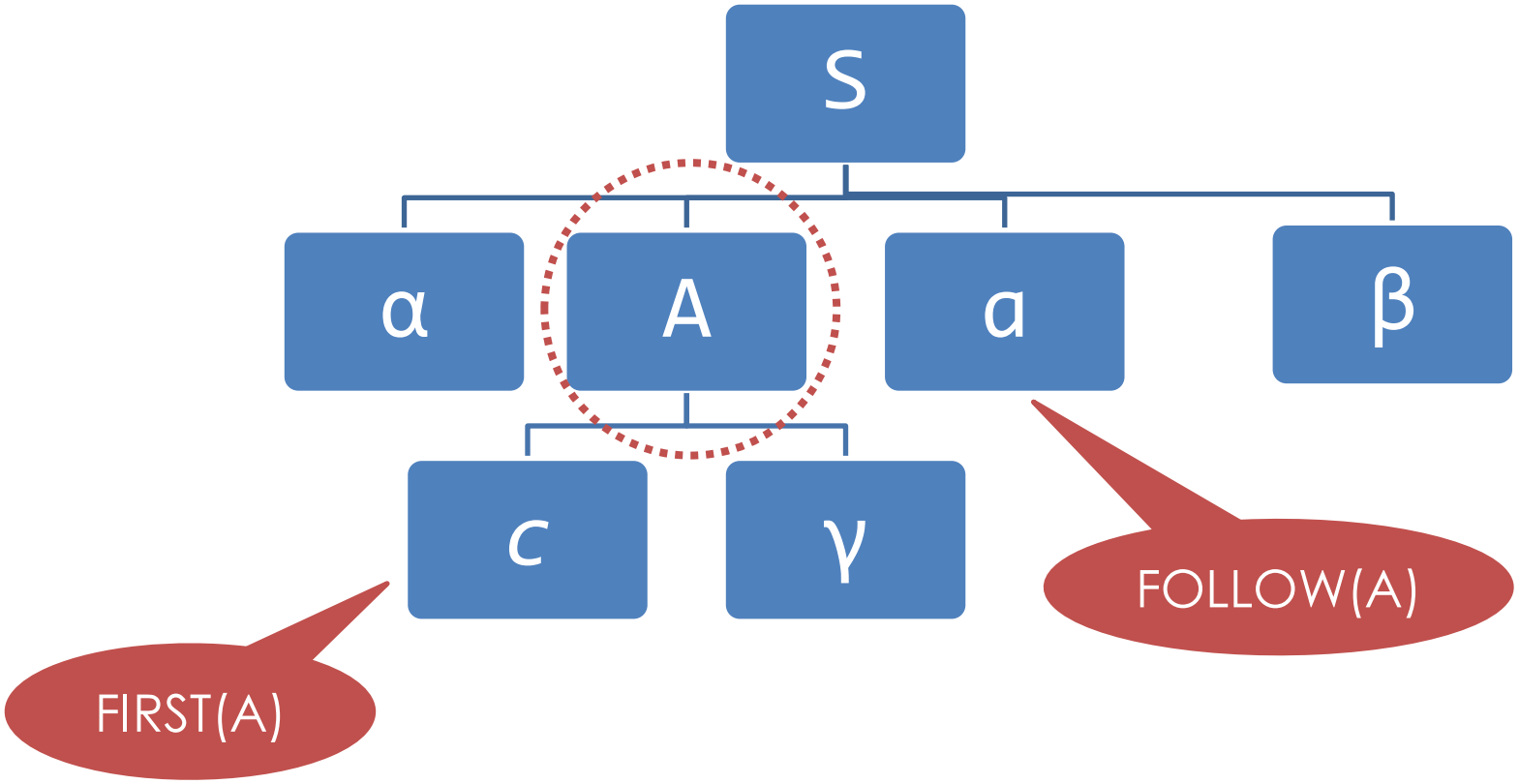
TEC

FOLLOW

- Función Auxiliar que ayuda a elegir la producción correcta.
- FOLLOW(A) es la función, para el no terminal A, del conjunto de terminales α pueden aparecer de inmediato a la derecha de A en cierta forma de frase.

TEC

FIRST & FOLLOW



TEC

FIRST & FOLLOW

- Para la gramática:

expresion ::= *termino expresion'*

expresion' ::= + *termino expresion'* | ε

termino ::= *factor termino'*

termino' ::= * *factor termino'* | ε

factor ::= (*expresion*) | **id**

- Calcular FIRST y FOLLOW de los no terminales

TEC

FIRST & FOLLOW

No Terminal	FIRST	FOLLOW
expresion	{(, id}	{), $}
expresion'	{+, ε}	{), $}
termino	{(, id}	{+,), $}
termino'	{*, ε}	{+,), $}
factor	{(, id}	{*, +,), $}

Gramáticas LL(k)

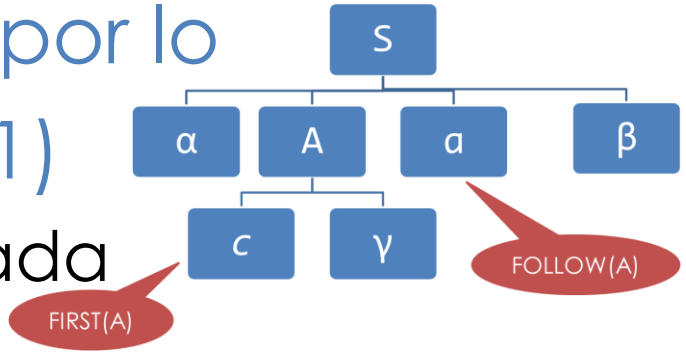
- Gramáticas utilizadas en los analizadores sintácticos predictivos.
  - Con descenso recursivo que no necesitan rastreo hacia atrás.
- Primera L: explora entrada de izquierda a derecha.
- Segunda L: para producir una derivación por la izquierda.
- k: cantidad de símbolos de anticipación

TEC

Gramáticas LL(1)

- Una gramática es LL(1) sí y sólo sí para cualesquiera dos producciones A → α y A → β se cumple:
  - FIRST(α) ∩ FIRST(β) = ∅
  - Si ε ∈ FIRST(β) => FIRST(α) ∩ FOLLOW(A) = ∅

- Los constructores de flujo por lo general cumplen con LL(1)
  - Inician con palabra reservada



Reconocimiento Ascendente

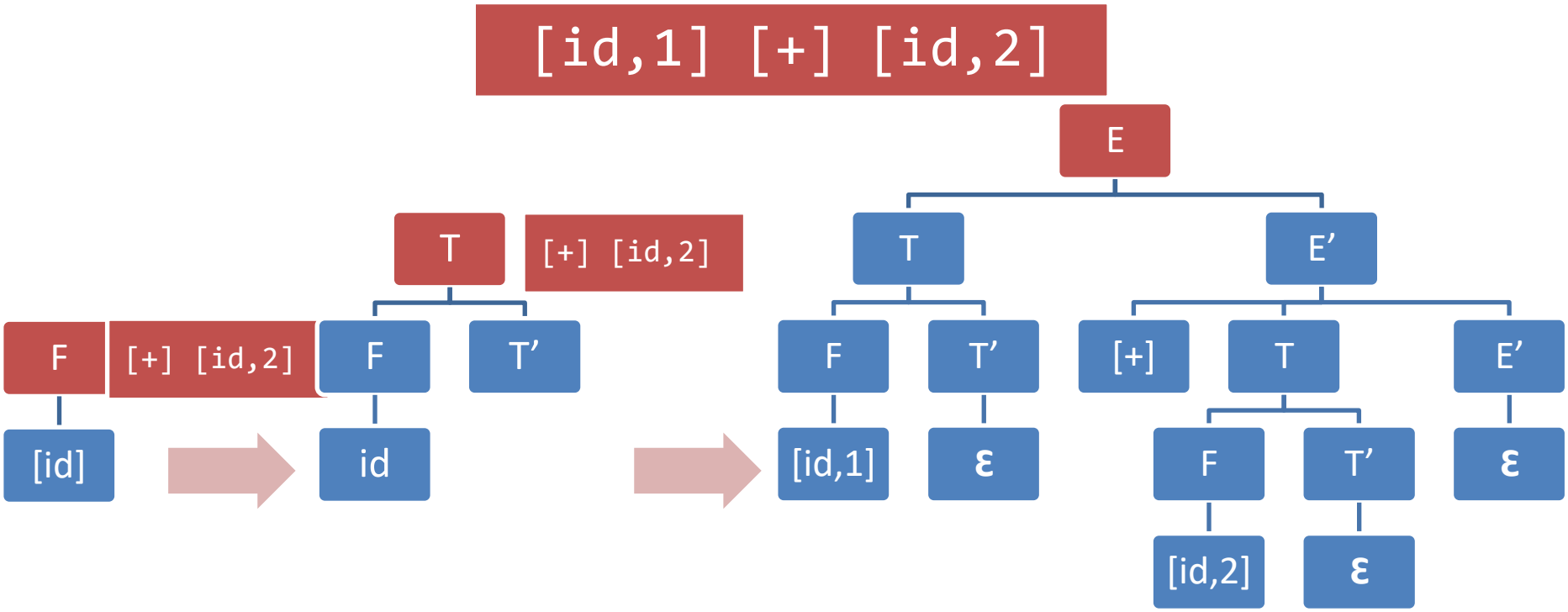
TEC

Reconocimiento Ascendente

- Se inicia la construcción iniciando desde las hojas avanzando hacia la raíz.
- Conocido como Análisis Sintáctico de Desplazamiento-Reducción.
- Se reduce una cadena al símbolo inicial de la gramática

TEC

Reconocimiento Ascendente



Desplazamiento-Reducción

- Análisis sintáctico ascendente
- Maneja una pila para controlar las reducciones.
- La parte superior de la pila se muestra a la derecha.

TEC

Desplazamiento-Reducción

- Operaciones:
  - Desplazar: Desplaza el siguiente símbolo de entrada y lo coloca en la parte superior de la pila
  - Reducir: El extremo derecho de la producción que se va a reducir debe estar en la parte superior de la pila. Localizar el extremo izquierdo de la cadena dentro de la pila y decidir con qué terminal se va a sustituir la cadena

TEC

Desplazamiento-Reducción

- Operaciones:
  - Aceptar: Anunciar que el análisis sintáctico se completó con éxito.
  - Error: Descubrir un error de sintaxis y llamar a una rutina de recuperación de errores.

Gramática

expresion ::= *expresion* + *termino*

expresion ::= *termino*

termino ::= *termino* * *factor*

termino ::= *factor*

factor ::= (*expresion*)

factor ::= **id**

TEC

Desplazamiento-Reducción

id * id

Pila	Entrada	Acción
$	id * id$	Desplazar
$id	* id$	Reducir F→id
$F	* id$	Reducir T→F
$T	* id$	Desplazar
$T *	id$	Desplazar
$T * id	$	Reducir F→id
$T * F	$	Reducir T→T * F
$T	$	Reducir E→T
$E	$	Aceptar

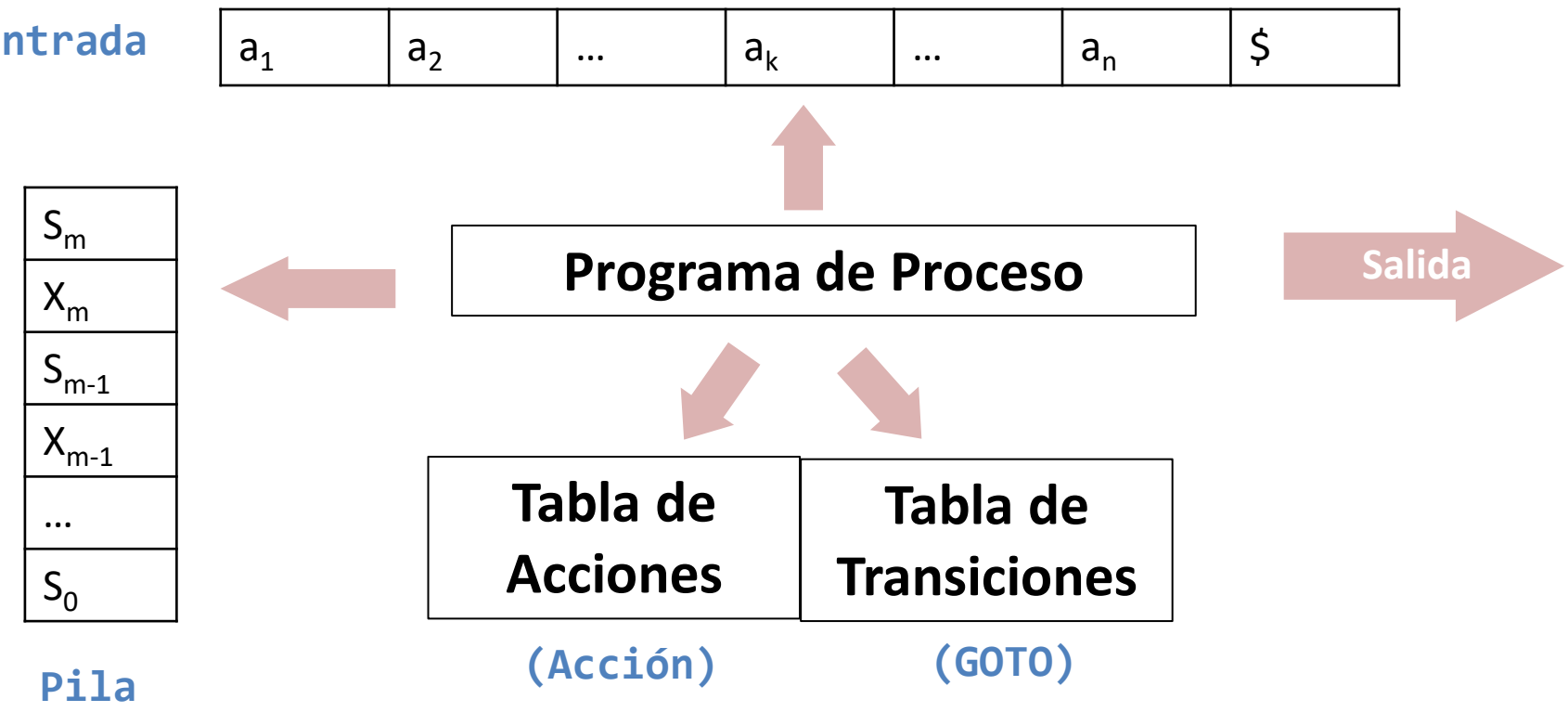
Análisis LR(k)

- Analizador Sintáctico Ascendente más frecuente
- L: explora entrada de izquierda a derecha.
- R: para producir una derivación por la derecha.
- k: cantidad de símbolos de anticipación
- Programa de proceso y tabla de análisis

Análisis LR(k)

- Pueden reconocer todas las construcciones de lenguajes de programación.
- Pueden detectar errores explorando de izquierda a derecha.
- Es el análisis Desplazamiento-Reducción más utilizado.

Análisis LR(k)



TEC

Gramática Ejercicios

expresion ::= *expresion* + *termino*

expresion ::= *termino*

termino ::= *termino* * *factor*

termino ::= *factor*

factor ::= (*expresion*)

factor ::= **id**

Portafolio #3

- Desarrolle el algoritmo Desplazamiento-Reducción el para las cadenas:
  - `13 + 7 * 23`
  - `(100*((1000)))`

Bonus

- Profundizar sobre Análisis LR.

Bibliografía

- Programming Language Processors in Java: compilers and interpreters. Watt, David, Brown, Deryck. Pearson Education. 2000

- Compilers: principles, techniques and tools (2da. ed.). Aho, Alfred. Pearson Education. 2007


TEC
Tecnológico
de Costa Rica