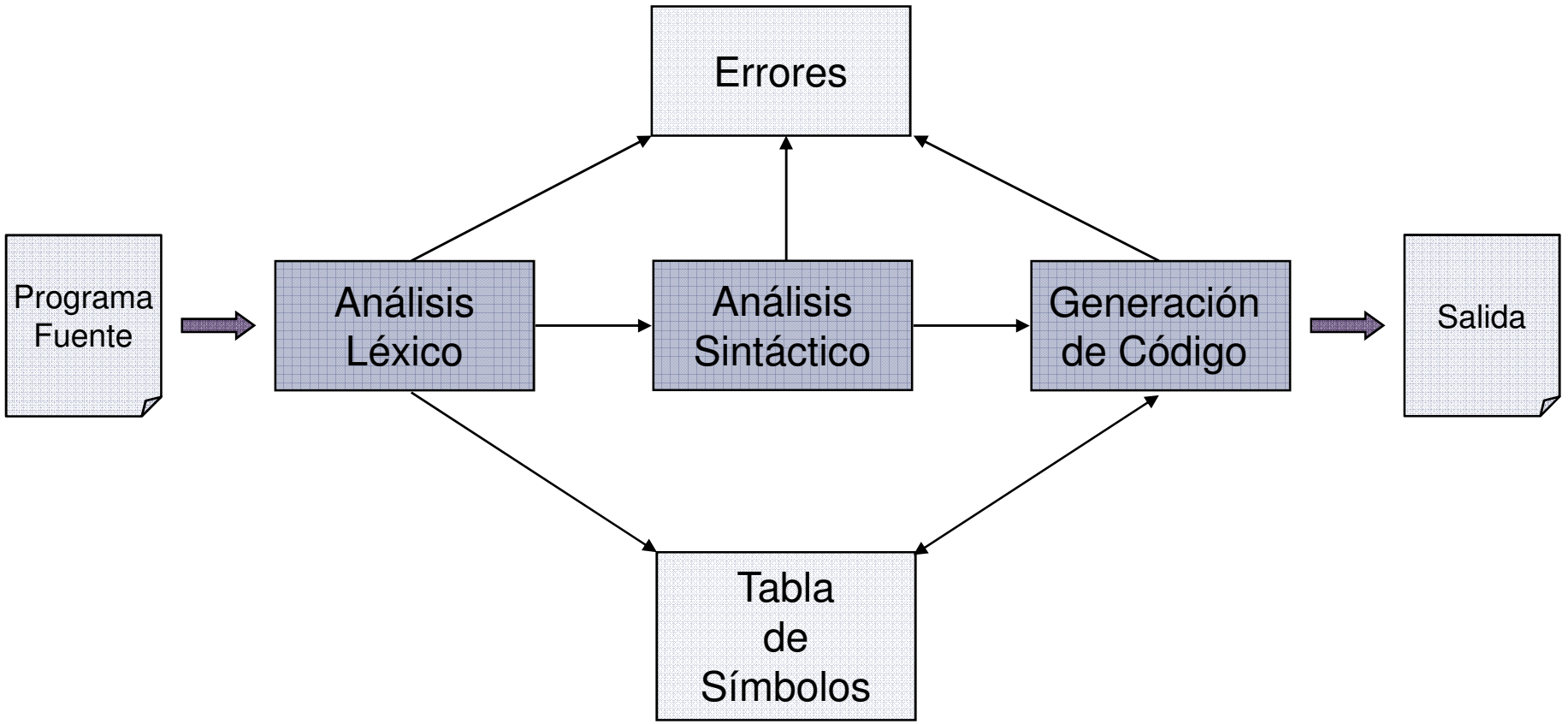
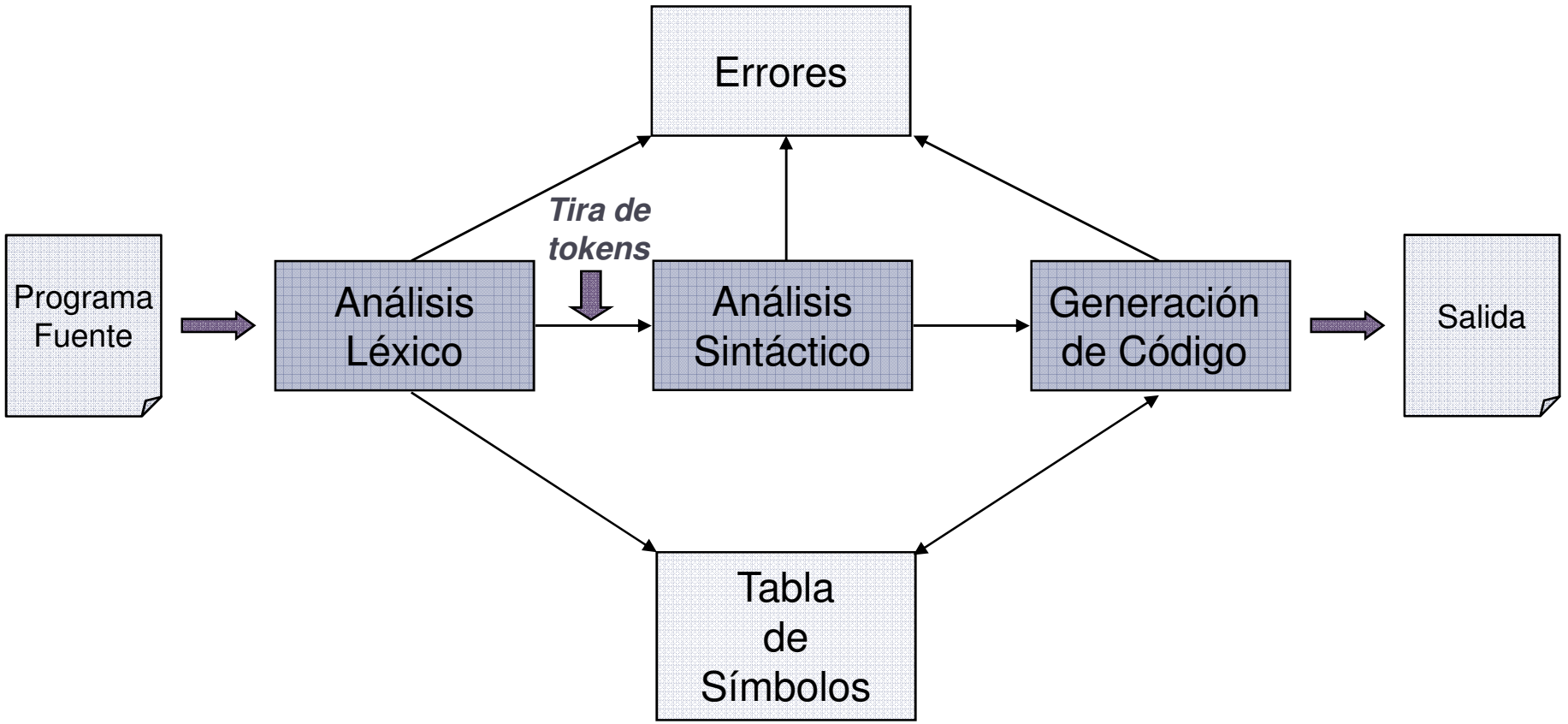
Diseño de Compiladores I

Análisis Sintáctico

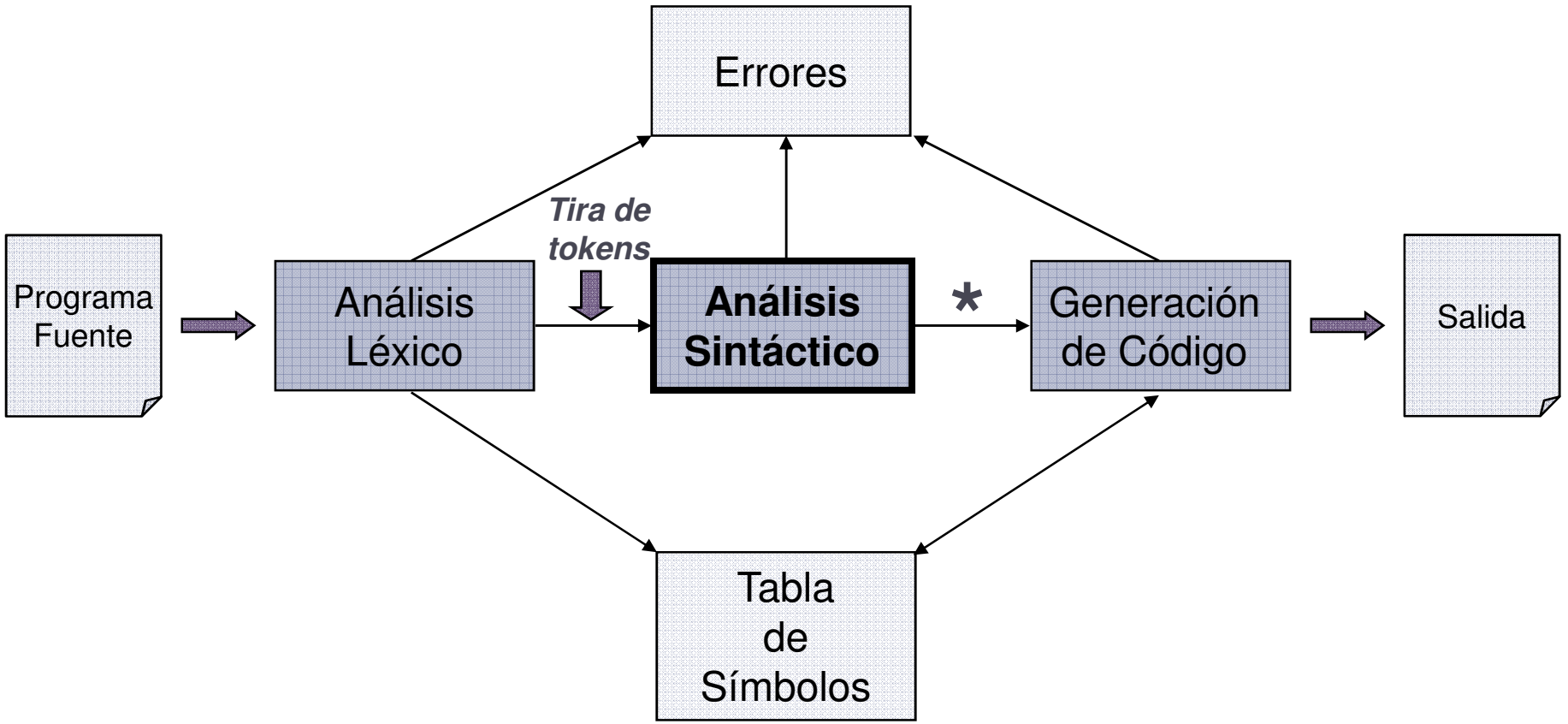
Fases de la Compilación



Fases de la Compilación



Fases de la Compilación



Análisis Sintáctico

Agrupa los tokens del programa fuente en frases gramaticales que el compilador usará en las siguientes etapas.

Análisis Sintáctico

Obtiene una cadena de tokens del Analizador Léxico, y verifica que la cadena de tokens presentes en la entrada pueda ser generada mediante la gramática del lenguaje.

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59] [27] [80] [28] [60] [27] [85] [27] [70] [27]
[73] [28] [61] [27] [85] [27]

IF ID >= CTE THEN ID := ID + ID / CTE ELSE ID := ID

Análisis Sintáctico

- Las construcciones de un lenguaje de programación pueden ser descriptas mediante una ***gramática independiente de contexto***, utilizando BNF.
- Las reglas de la gramática se representan por medio de ***producciones***.
- Cada producción define un ***símbolo no terminal*** en función de ***símbolos terminales*** o tokens, y otros símbolos no terminales.
- Existe una producción que define al no terminal ***programa***.

Análisis Sintáctico
Gramática

1) PROGRAMA → LS
2) LS → LS ST | ST
3) ST → S
4) ST → A
5) S → if C then ST else ST
6) C → E CP E
7) CP → < | > | <= | >= | == | <>
8) A → id := E
9) E → E + T
10) E → E – T
11) E → T
12) T → T * F
13) T → T / F
14) T → F
15) F → id
16) F → cte

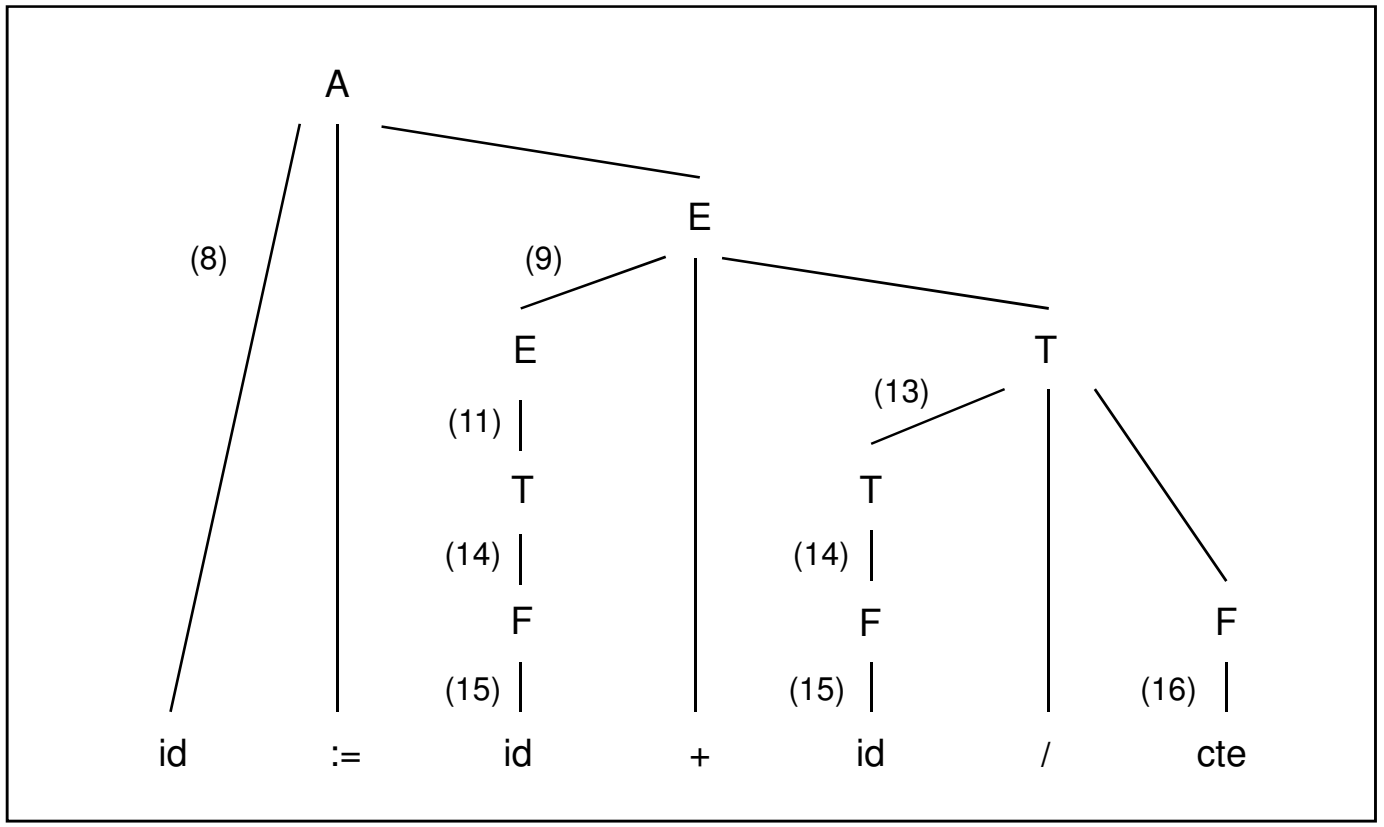
¿Se podría representar esta gramática con un Autómata Finito?

Análisis Sintáctico

- Conceptualmente, el Analizador Sintáctico construye un ***árbol de parsing***.
- El árbol de parsing describe la estructura sintáctica del código de entrada.
- El árbol de parsing demuestra como la secuencia de tokens de entrada puede ser derivada a partir de las reglas de la gramática.

Árbol de Parsing

Tasa := Base + Recargo / 100 → id := id + id / cte



8) $A \rightarrow id := E$
9) $E \rightarrow E + T$
10) $E \rightarrow E - T$
11) $E \rightarrow T$
12) $T \rightarrow T * F$
13) $T \rightarrow T / F$
14) $T \rightarrow F$
15) $F \rightarrow id$
16) $F \rightarrow cte$

Lista de reglas: 15 14 11 15 14 16 13 9 8

¿El compilador construye el Árbol de Parsing?

Análisis Sintáctico
Implementación

- Se requiere una estructura más compleja que un Autómata finito.
- Se requiere un ***Autómata de Pila***

Estrategias de Parsing

- Parsing Ascendente (botton up)
  - Construye el árbol desde las hojas a la raíz
- Parsing Descendente (top down)
  - Construye el árbol desde la raíz a las hojas

Estrategias de Parsing

- El Parsing Descendente utiliza gramáticas LL.
- El Parsing Ascendente utiliza gramáticas LR.
- Los métodos ascendente y descendente más eficientes no funcionan para todas las gramáticas.
- Las gramáticas LL y LR son lo bastante expresivas como para describir la mayoría de las construcciones sintácticas de los lenguajes de programación modernos.

Estrategias de Parsing

- Los Analizadores Sintácticos que se implementan manualmente utilizan con frecuencia gramáticas LL (descendente)
- Los Analizadores Sintácticos para la clase más extendida de gramáticas LR (ascendente), se construyen generalmente mediante herramientas.

Parsing Ascendente
Gramáticas LR

- Va del programa a la hipótesis
- El programa se lee de izquierda a derecha (**L**).
- Las reglas se leen de derecha a izquierda (**R**): el lado derecho se reemplaza por el izquierdo.
- Estrategia: ***Reducción***

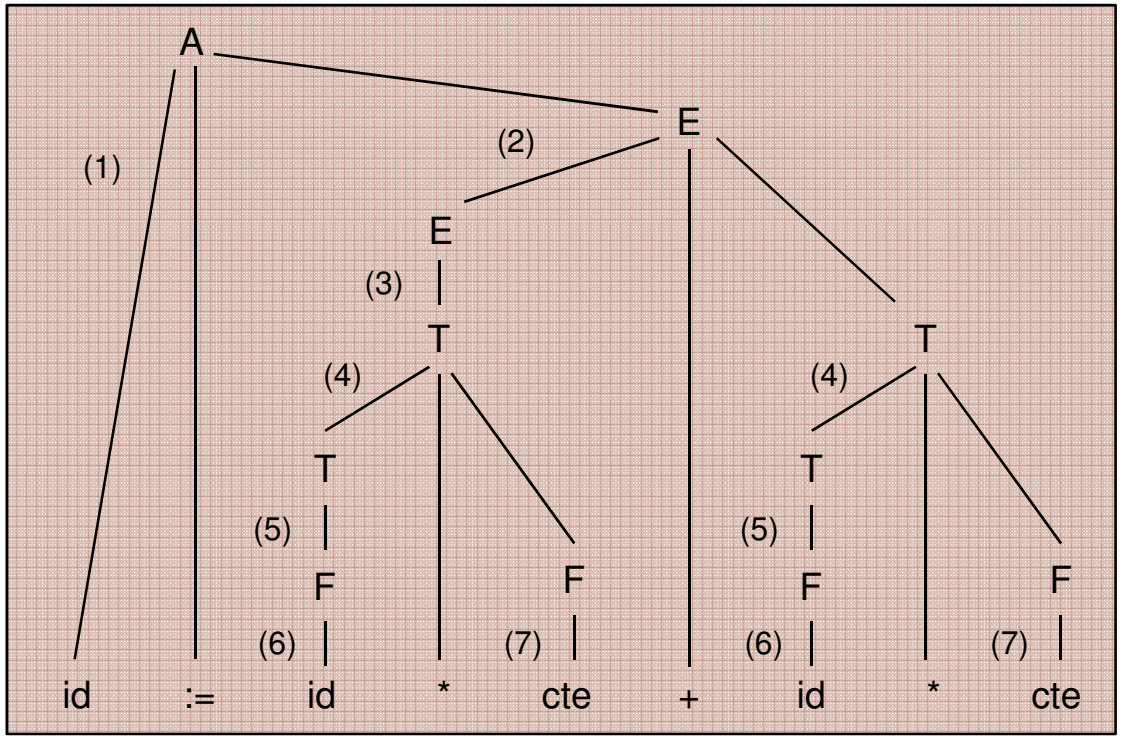
Parsing Ascendente
Ejemplo

precio := costo1 * 1.5 + costo2 * 1.2 → id := id * cte + id * cte

Gramática

1) A → id := E
2) E → E + T
3) E → T
4) T → T * F
5) T → F
6) F → id
7) F → cte

Árbol de Parsing



Lista de Reglas: 6 5 7 4 3 6 5 7 4 2 1

Parsing Ascendente
Ejemplo

Lista de Reglas: 6 5 7 4 3 6 5 7 4 2 1

id := id * cte + id * cte
6 id := F * cte + id * cte
5 id := T * cte + id * cte
7 id := T * F + id * cte
4 id := T + id * cte
3 id := E + id * cte
6 id := E + F * cte
5 id := E + T * cte
7 id := E + T * F
4 id := E + T
2 id := E
1 A

1) $A \rightarrow id := E$
2) $E \rightarrow E + T$
3) $E \rightarrow T$
4) $T \rightarrow T * F$
5) $T \rightarrow F$
6) $F \rightarrow id$
7) $F \rightarrow cte$

Parsing Descendente
Gramáticas LL

- Va de la hipótesis al programa
- El programa se lee de izquierda a derecha (**L**).
- Las reglas se leen de izquierda a derecha (**L**): el lado izquierdo se reemplaza por el derecho.
- Estrategia: ***Expansión***

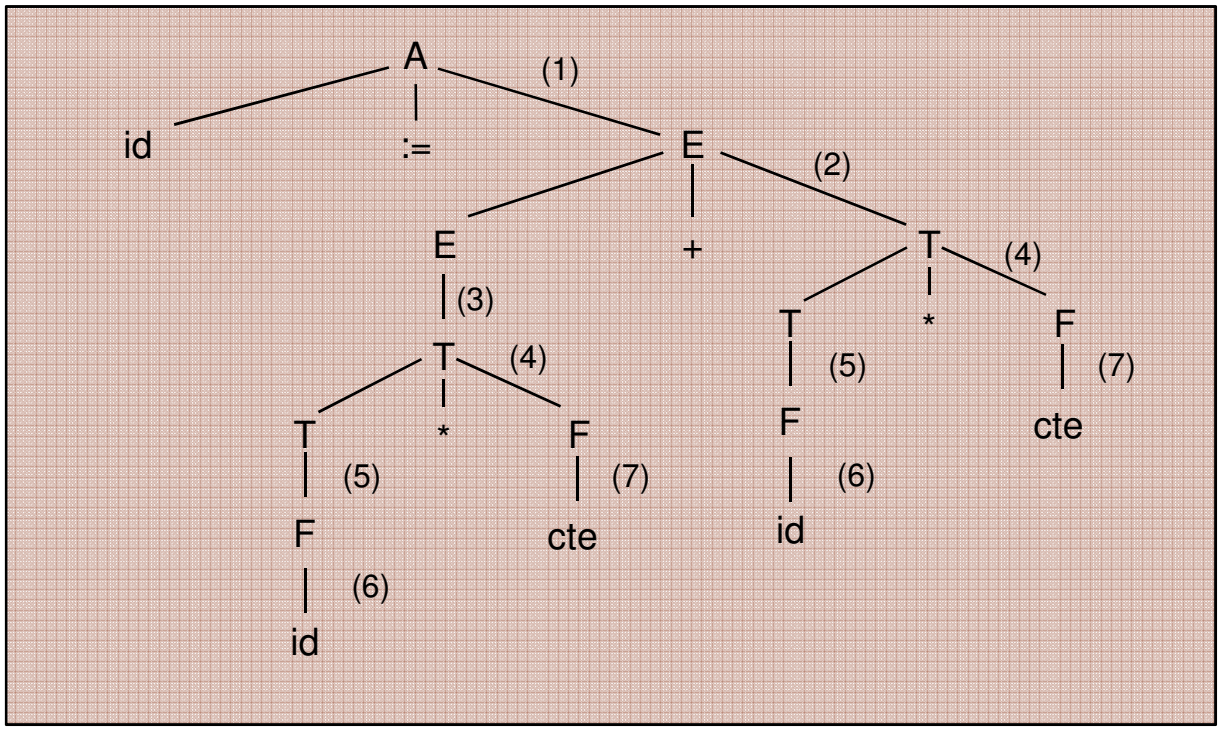
Parsing Descendente
Ejemplo

precio := costo1 * 1.5 + costo2 * 1.2 → id := id * cte + id * cte

Gramática

1) A → id := E
2) E → E + T
3) E → T
4) T → T * F
5) T → F
6) F → id
7) F → cte

Árbol de Parsing



Lista de Reglas: 1 2 3 4 5 6 7 4 5 6 7

Parsing Descendente
Ejemplo

Lista de Reglas: 1 2 3 4 5 6 7 4 5 6 7

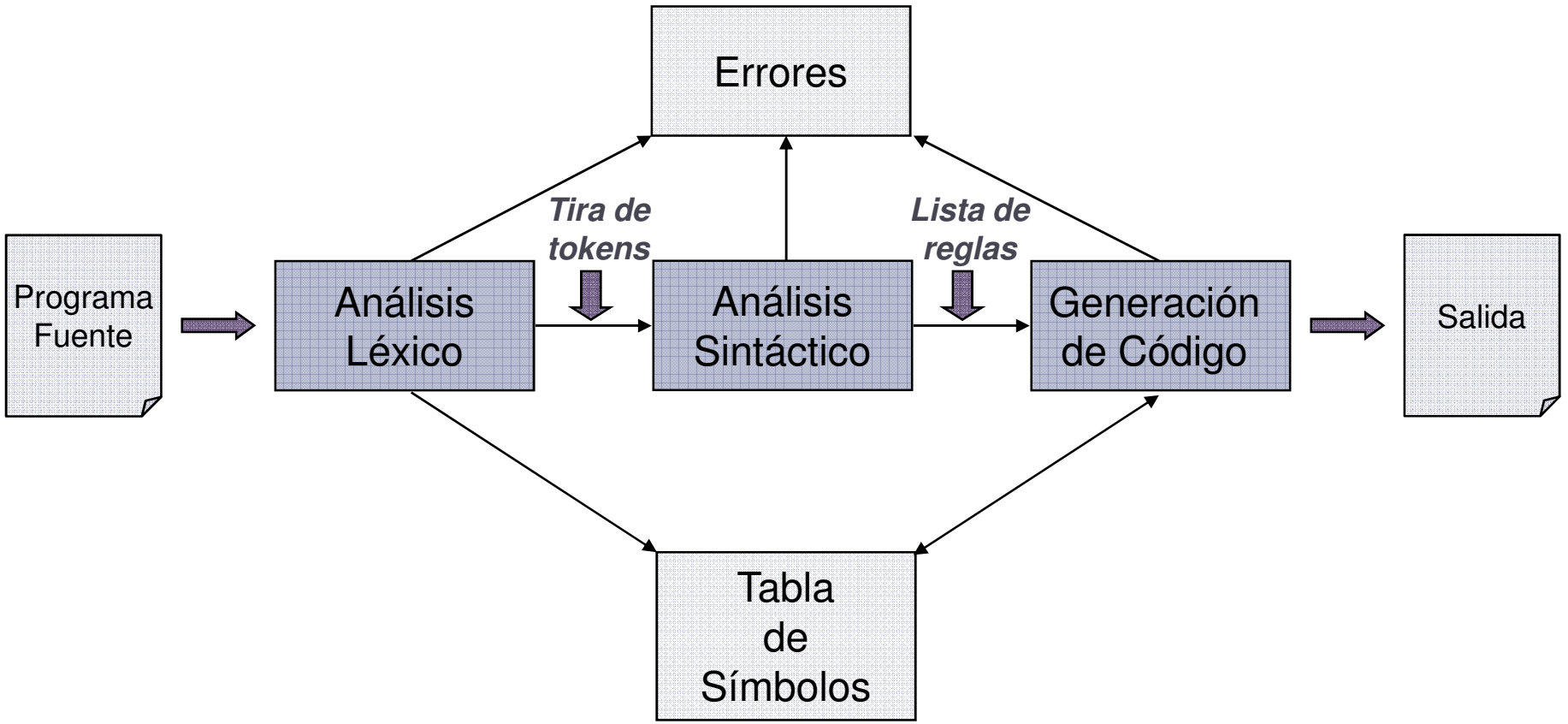
	A
1	id := E
2	id := E + T
3	id := T + T
4	id := T * F + T
5	id := F * F + T
6	id := id * F + T
7	id := id * cte + T
4	id := id * cte + T * F
5	id := id * cte + F * F
6	id := id * cte + id * F
7	id := id * cte + id * cte

1) $A \rightarrow id := E$
2) $E \rightarrow E + T$
3) $E \rightarrow T$
4) $T \rightarrow T * F$
5) $T \rightarrow F$
6) $F \rightarrow id$
7) $F \rightarrow cte$

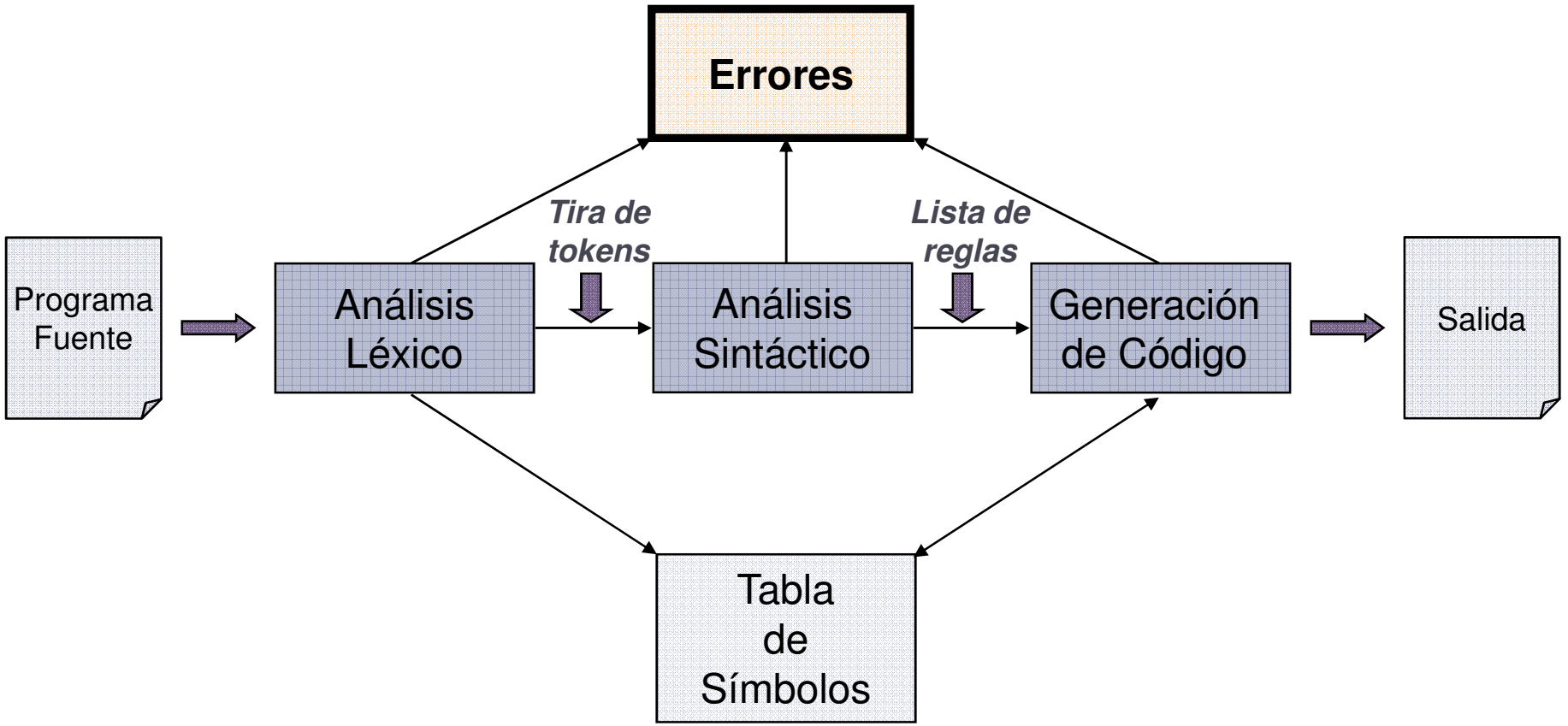
Parsing Descendente

- Predice el programa.
- Sólo puede construir el programa deseado, si se elige la secuencia correcta de reglas.
- Otras secuencias dan otros programas.
- Determinar la secuencia correcta es difícil.

Fases de la Compilación



Fases de la Compilación



Análisis Sintáctico
Errores

- Ante un error, el Análisis Sintáctico se detiene.

- Para continuar con la compilación, se deben usar técnicas de recuperación:
  - Modo pánico
  - Producciones de error
  - Recuperación a nivel de frase
    (Requiere producciones de error)
  - Corrección global