Lenguajes y Compiladores

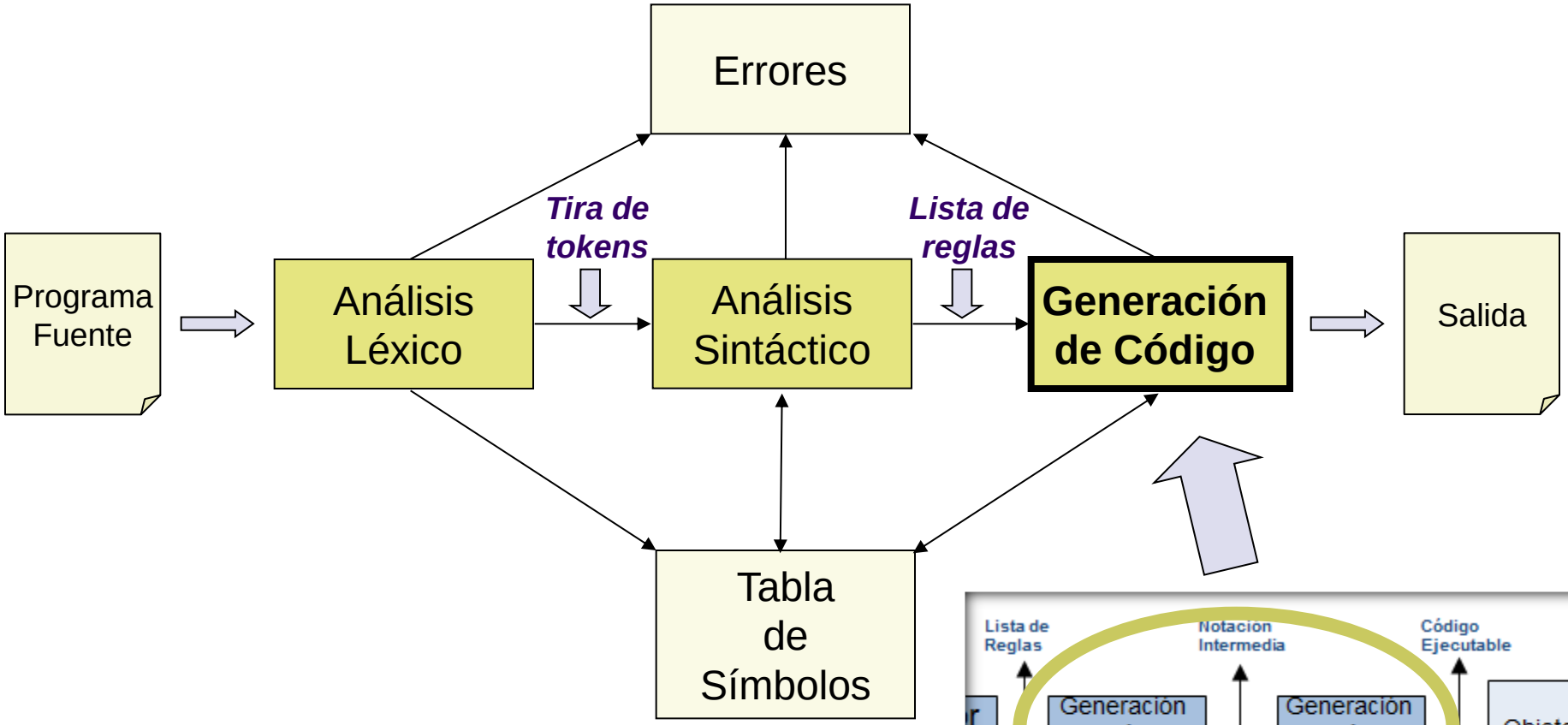
Generación de Código Intermedio

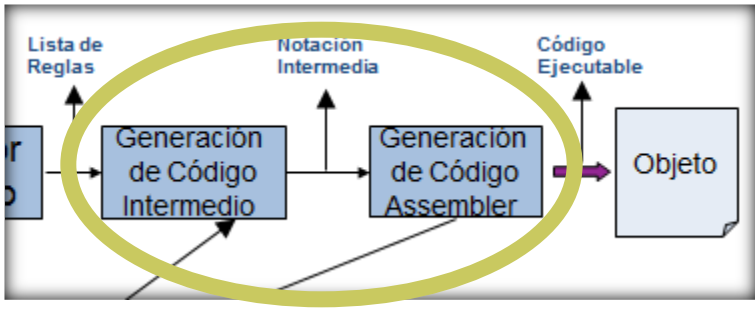
Árbol Sintáctico-Polaca Inversa-Tercetos

Sentencias Básicas

UNLaM

Fases de la compilación





Generación de Código Intermedio

Gramática

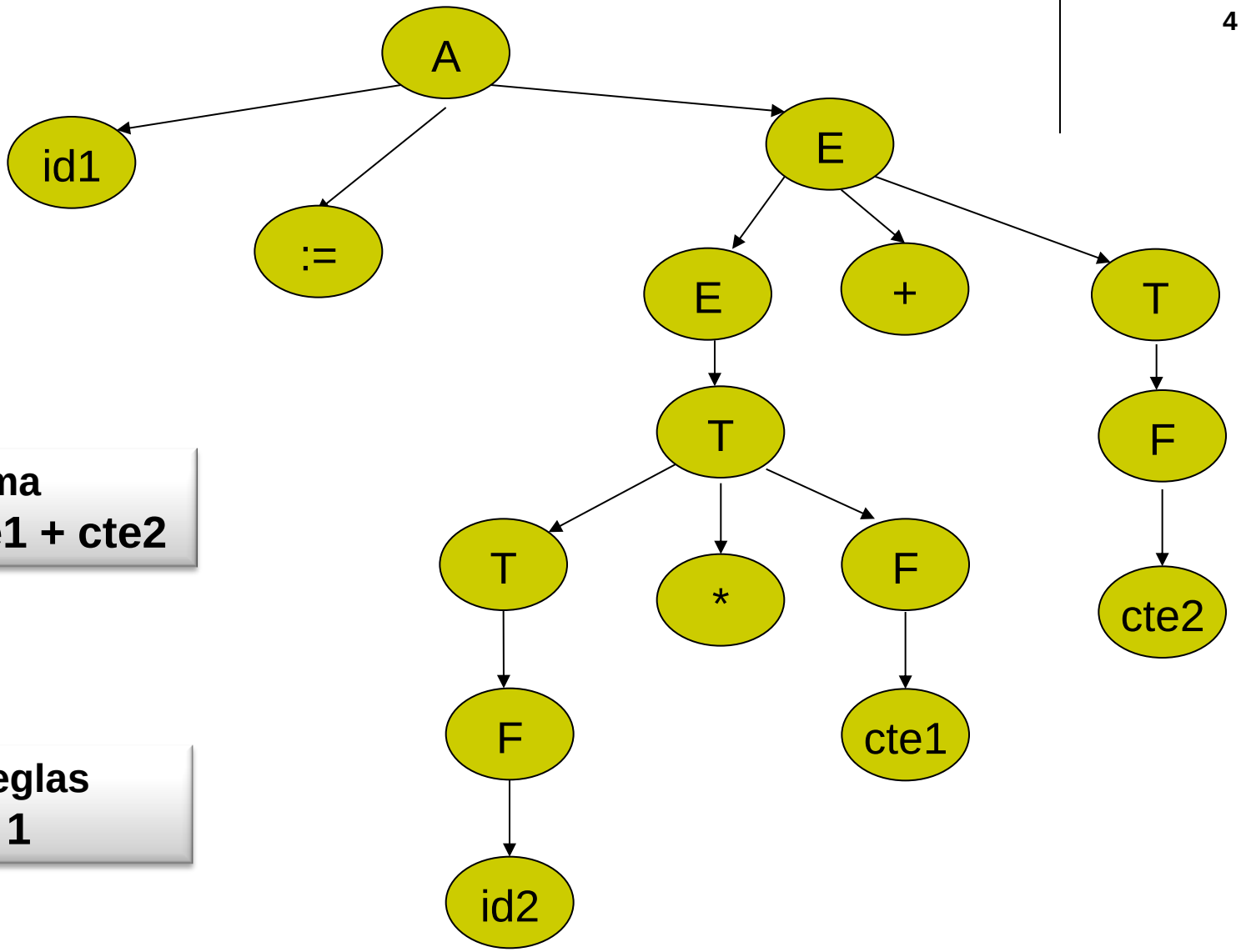
1) $A \rightarrow id := E$
2) $E \rightarrow E + T$
3) $E \rightarrow T$
4) $T \rightarrow T * F$
5) $T \rightarrow F$
6) $F \rightarrow id$
7) $F \rightarrow cte$

id1:= id2 * cte1 + cte2

Generación de Código Intermedio

Programa
id1:= id2 * cte1 + cte2

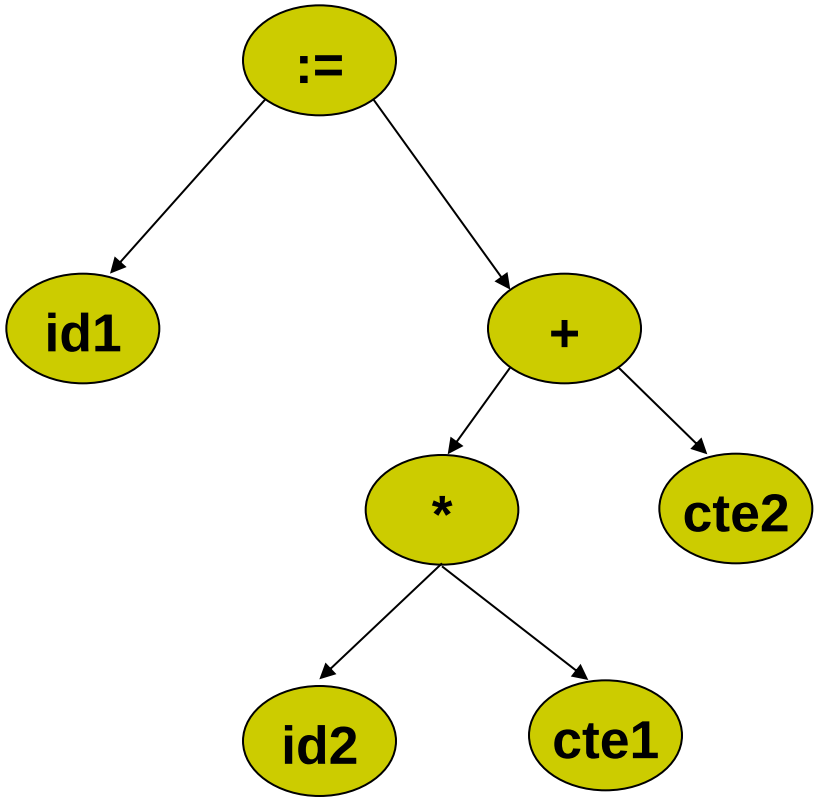
Lista de Reglas
6 5 7 4 3 7 5 2 1



Generación de Código Intermedio

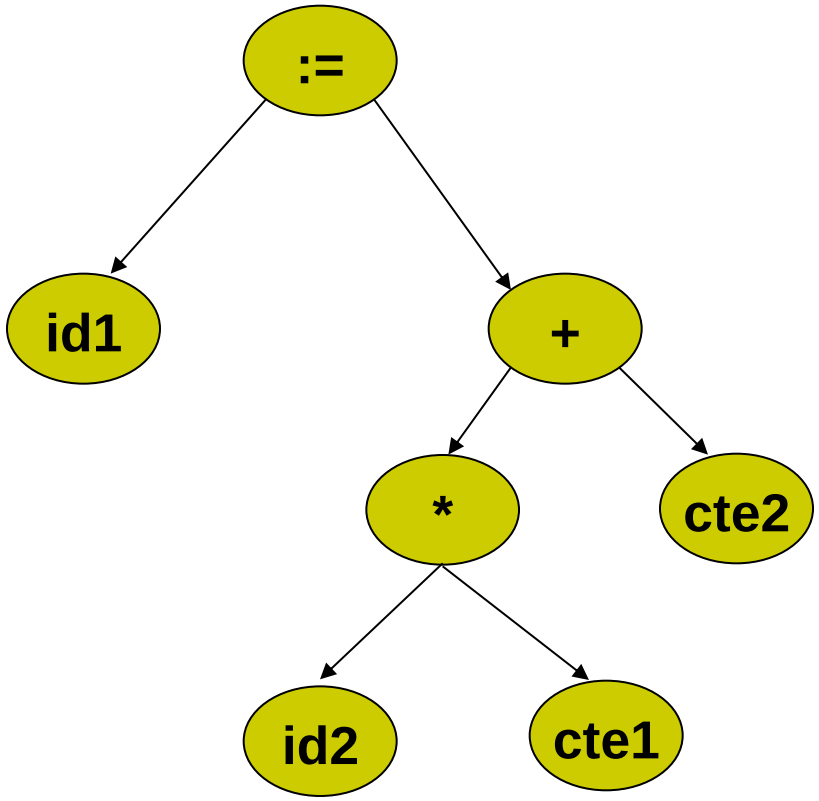
Se puede achicar el árbol de parsing

Toda vez que aparece un E_{NT} con hijo único, se sube el E_T lo más alto posible



Árbol Sintáctico

Generación de Código Intermedio

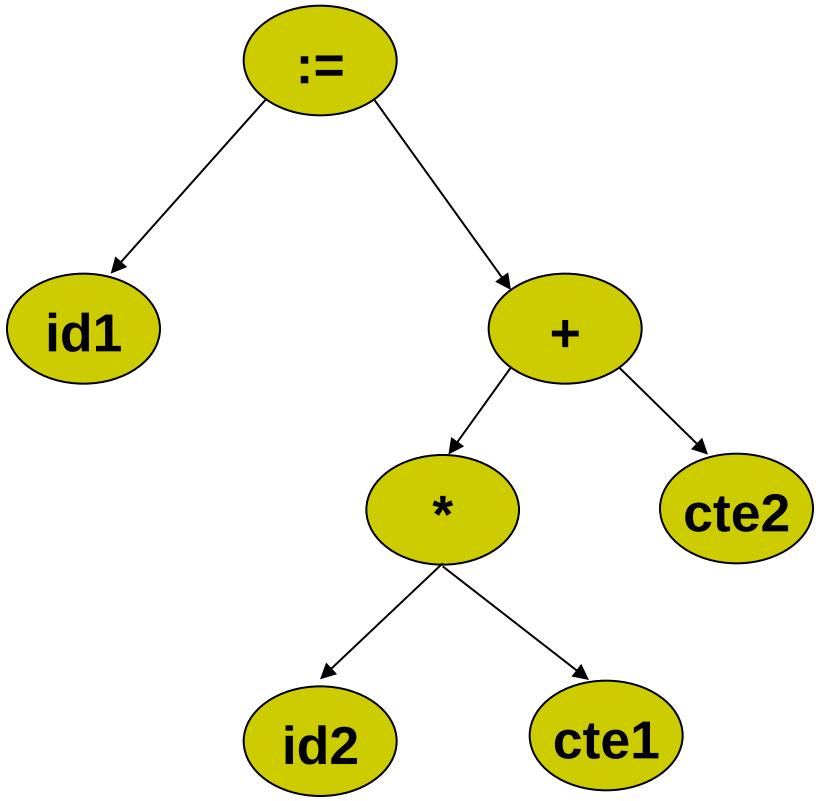


Si se lee el Árbol Sintáctico INORDER
(Hijo Izquierdo, Padre, Hijo Derecho)

id1	:=	id2	*	cte1	+	cte2

Programa Original

Generación de Código Intermedio



Si se lee el Árbol Sintáctico POSORDER
(Hijo Izquierdo, Hijo Derecho, Padre)

id1	id2	cte1	*	cte2	+	:=

Polaca Inversa (RPN)
(Reverse polish notation)

Generación de Código Intermedio

Si se recorre la Polaca Inversa de izquierda a derecha hasta el primer operador de más a la derecha , se buscan los dos operandos a su izquierda y se los guarda en una tripla

id1	id2	cte1	*	cte2	+	:=

[1] (* id2 cte1)

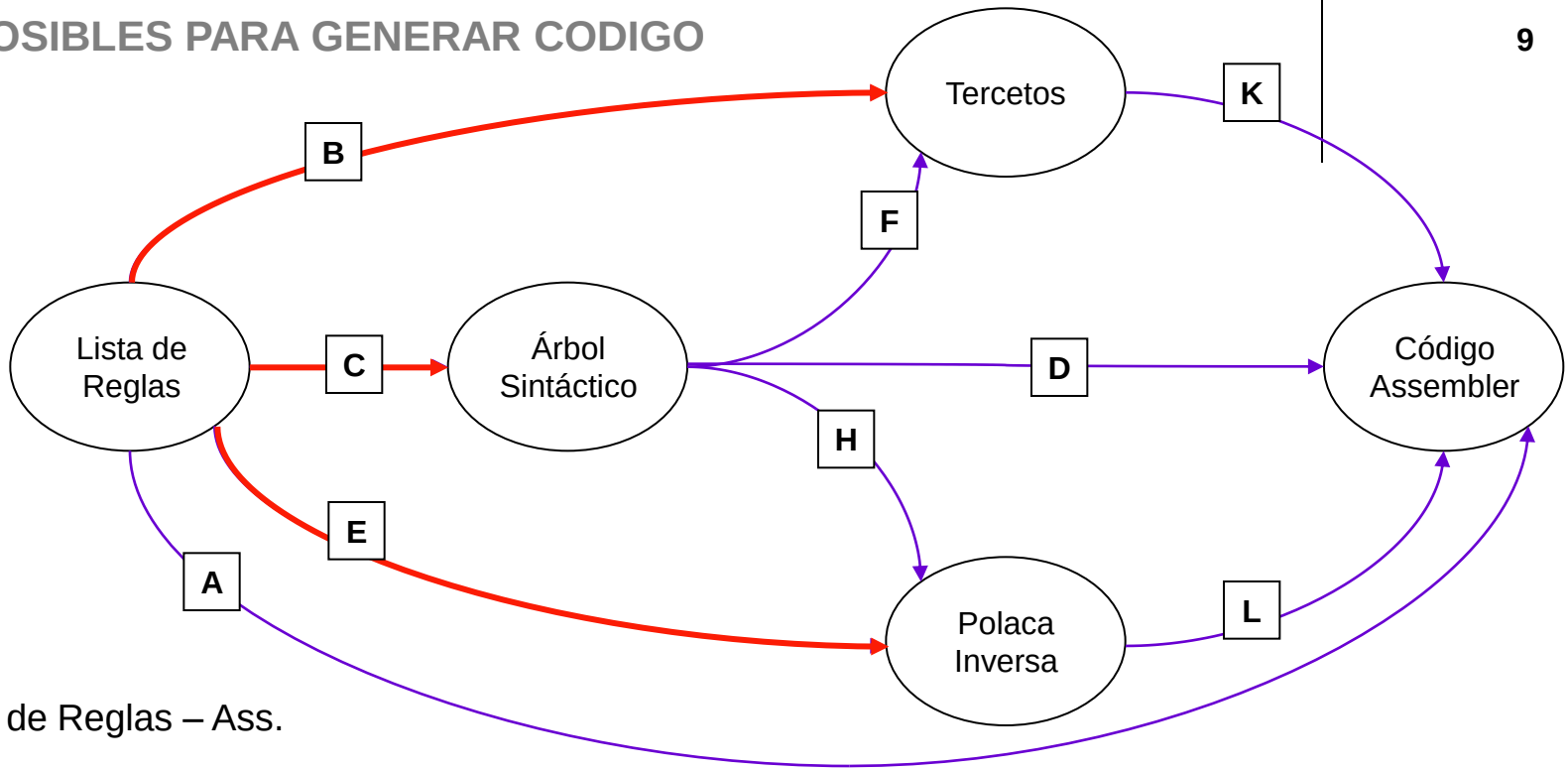
[2] (+ [1] cte2)

[3] (:= id1 [2])

Tercetos

Generación de Código Intermedio

CAMINOS POSIBLES PARA GENERAR CODIGO



- Camino A : Lista de Reglas – Ass.

Camino E-L : LR – PI - Ass.

Camino B-K : LR – Tercetos – Ass.

Camino C-D : LR – AS – Ass.

Camino C-F-K : LR – AS – Tercetos – Ass.

Camino C-H-L : LR – AS – PI - Ass

Traducciones

- Lista de Reglas → Árbol Sintáctico
- Lista de Reglas → Tercetos
- Lista de Reglas → Polaca Inversa

Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Ante todo es necesario saber que el árbol se debe armar de manera que, una vez armado, pueda ser recorrido desde el subárbol de más a la izquierda con dos nodos hoja.

De esta forma, el código assembler se podrá construir con facilidad.

Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Se requiere :

- Un puntero por cada elemento no terminal
- Dos rutinas que se asocian a cada regla
  - crear_nodo () // crea nodos intermedios en el árbol
  - crear_hoja() // crea las hojas del árbol

Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

1. <ASIG> → id := <EXPRESION>
2. <EXPRESION> → <EXPRESION> + <TERMINO>
3. <EXPRESION> → <TERMINO>
4. <TERMINO> → < TERMINO> * <FACTOR>
5. <TERMINO> → <FACTOR>
6. <FACTOR> → id
7. <FACTOR> → cte

Start : <ASIG>

Un puntero por cada elemento no terminal

Aptr
Eptr
Tptr
Fptr

Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

1.	<ASIG> → id := <EXPRESION>	Aptr = crearNodo(“:=”, crearHoja(id), Eptr)
2.	<EXPRESION> → <EXPRESION> + <TERMINO>	Eptr = crearNodo(“+”, Eptr, Tptr)
3.	<EXPRESION> → <TERMINO>	Eptr = Tptr
4.	<TERMINO> → < TERMINO> * <FACTOR>	Tptr = crearNodo(“*”, Tptr, Fptr)
5.	<TERMINO> → <FACTOR>	Tptr = Fptr
6.	<FACTOR> → id	Fptr = crearHoja(id)
7.	<FACTOR> → cte	Fptr = crearHoja(cte)

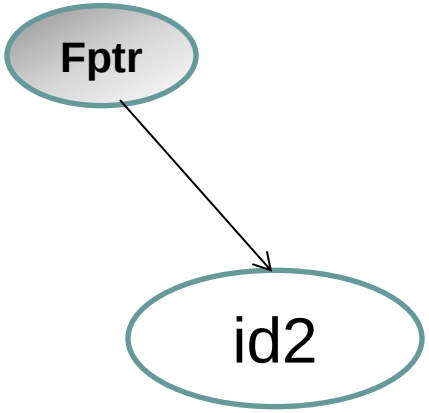
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : x:= z * 17.1 + 8 (id1 := id2 * cte1 + cte2)
Lista de Reglas : 6 5 7 4 3 7 5 2 1

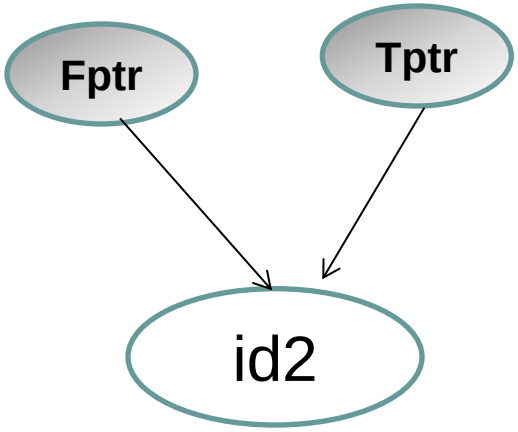
Regla 6 : <FACTOR> → id

Fptr = crearHoja(id)



Regla 5 : <TERMINO> → <FACTOR>

Tptr = Fptr



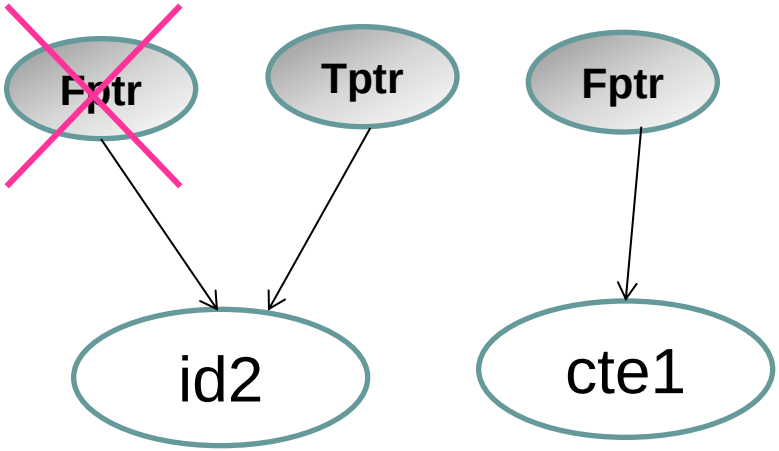
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2~~ * cte1 + cte2
Lista de Reglas : ~~6 5~~ 7 4 3 7 5 2 1

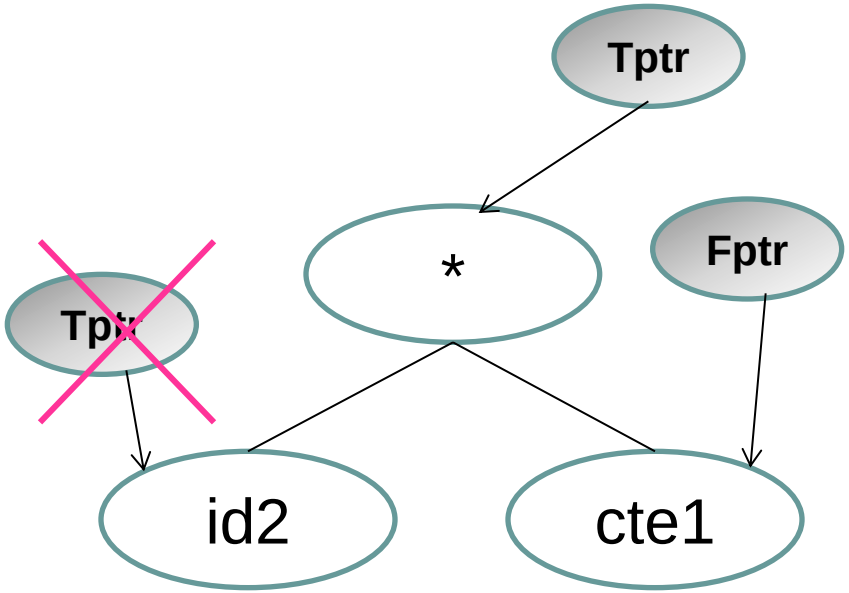
Regla 7 : <FACTOR> → cte

Fptr = crearHoja(cte)



Regla 4 : <TERMINO> → < TERMINO> * <FACTOR>

Tptr = crearNodo("*", Tptr, Fptr)



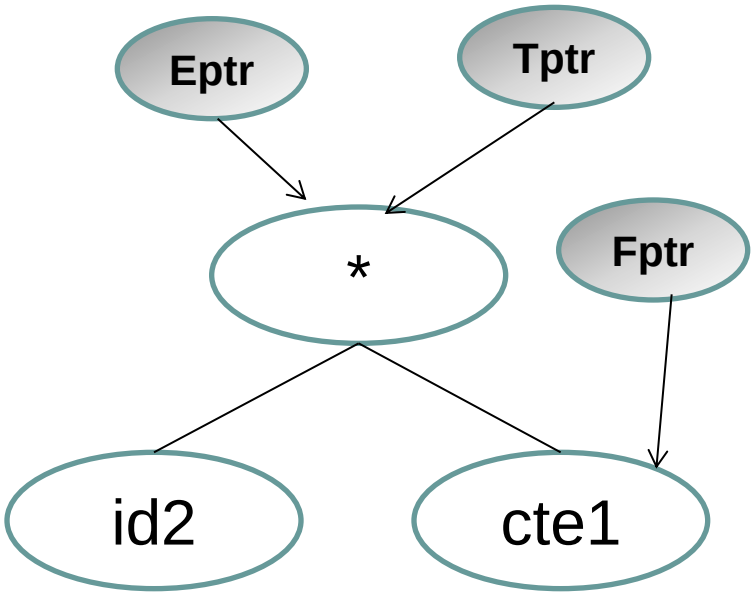
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2 * cte1~~ + cte2
Lista de Reglas : ~~6 5 7 4~~ 3 7 5 2 1

Regla 3 : <EXPRESION> → <TERMINO>

Eptr = Tptr



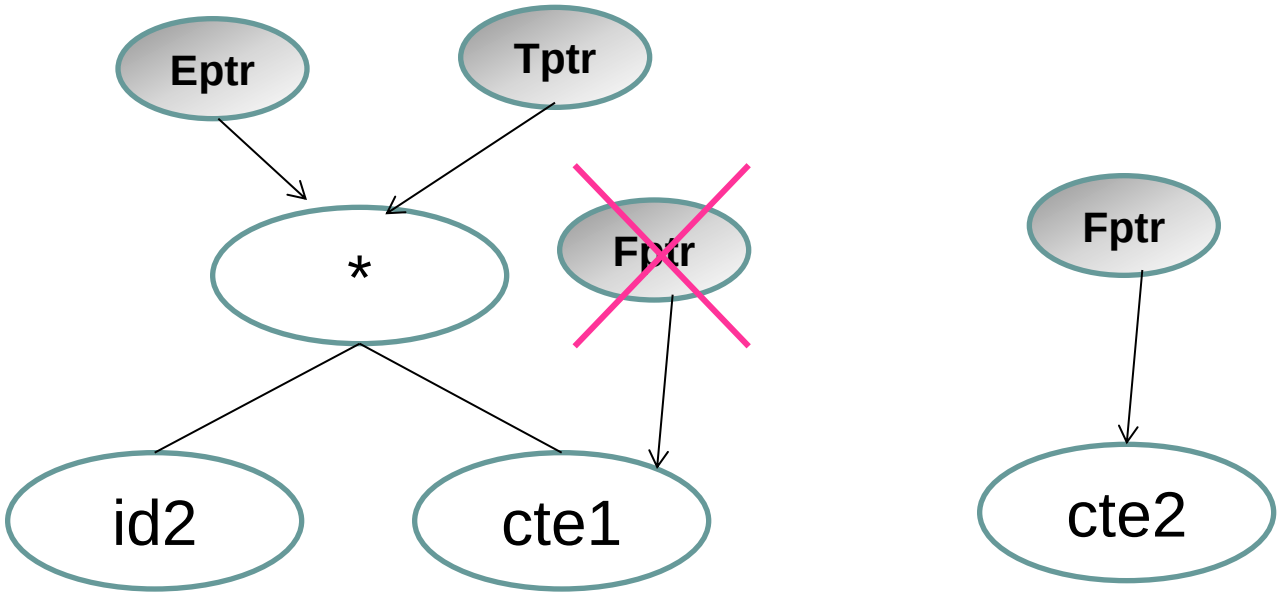
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2 * cte1~~ + cte2
Lista de Reglas : ~~6 5 7 4~~ 3 7 5 2 1

Regla 7 : <FACTOR> → cte

Fptr = crearHoja(cte)



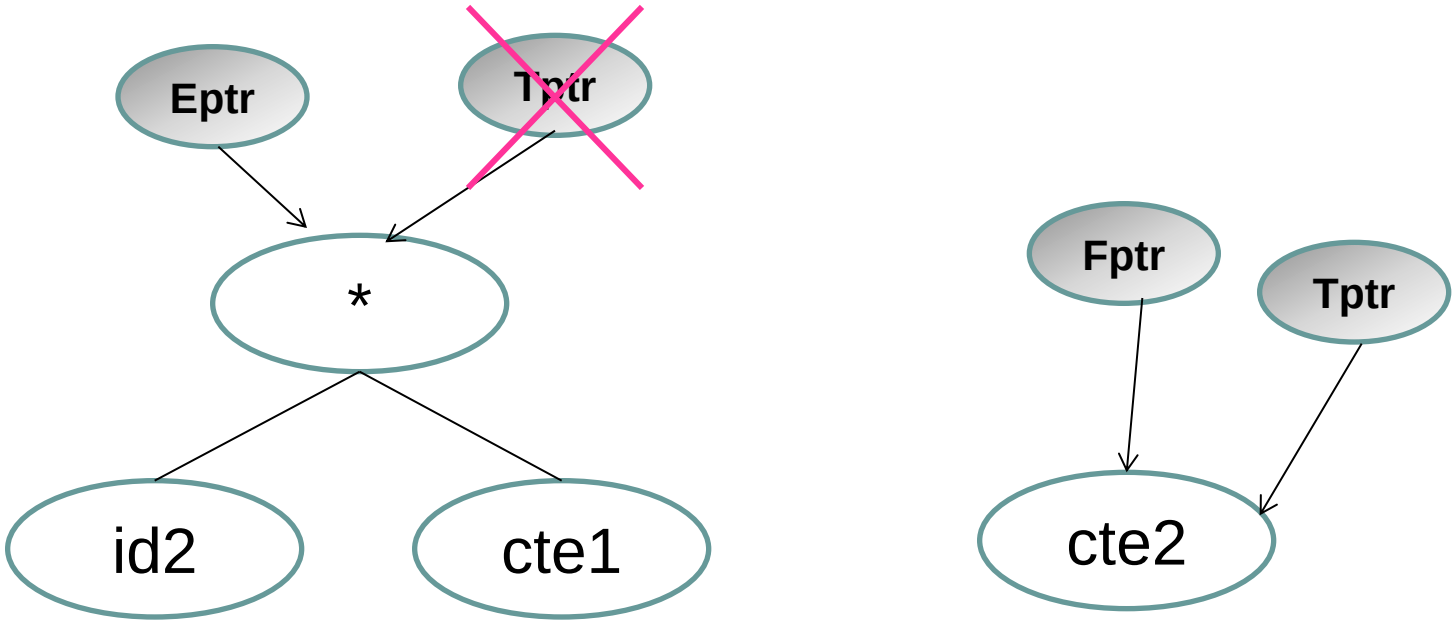
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2 * cte1~~ + cte2
Lista de Reglas : ~~6 5 7 4 3 7~~ 5 2 1

Regla 5 : <TERMINO> → <FACTOR>

Tptr = Fptr



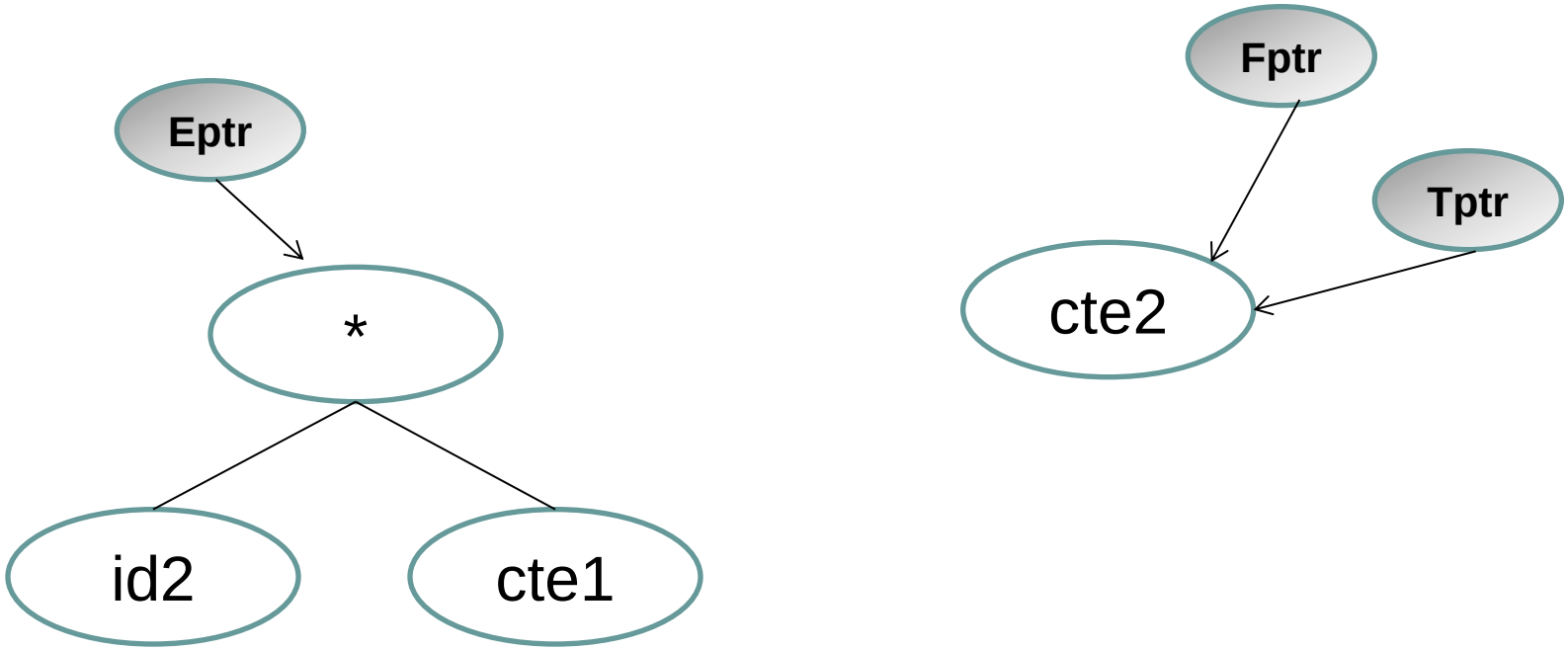
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2 * cte1~~ + cte2
Lista de Reglas : ~~6 5 7 4 3 7 5~~ 2 1

Regla 2 : <EXPRESION> → < EXPRESION> + <TERMINO>

Eptr = crearNodo("+", Eptr, Tptr)



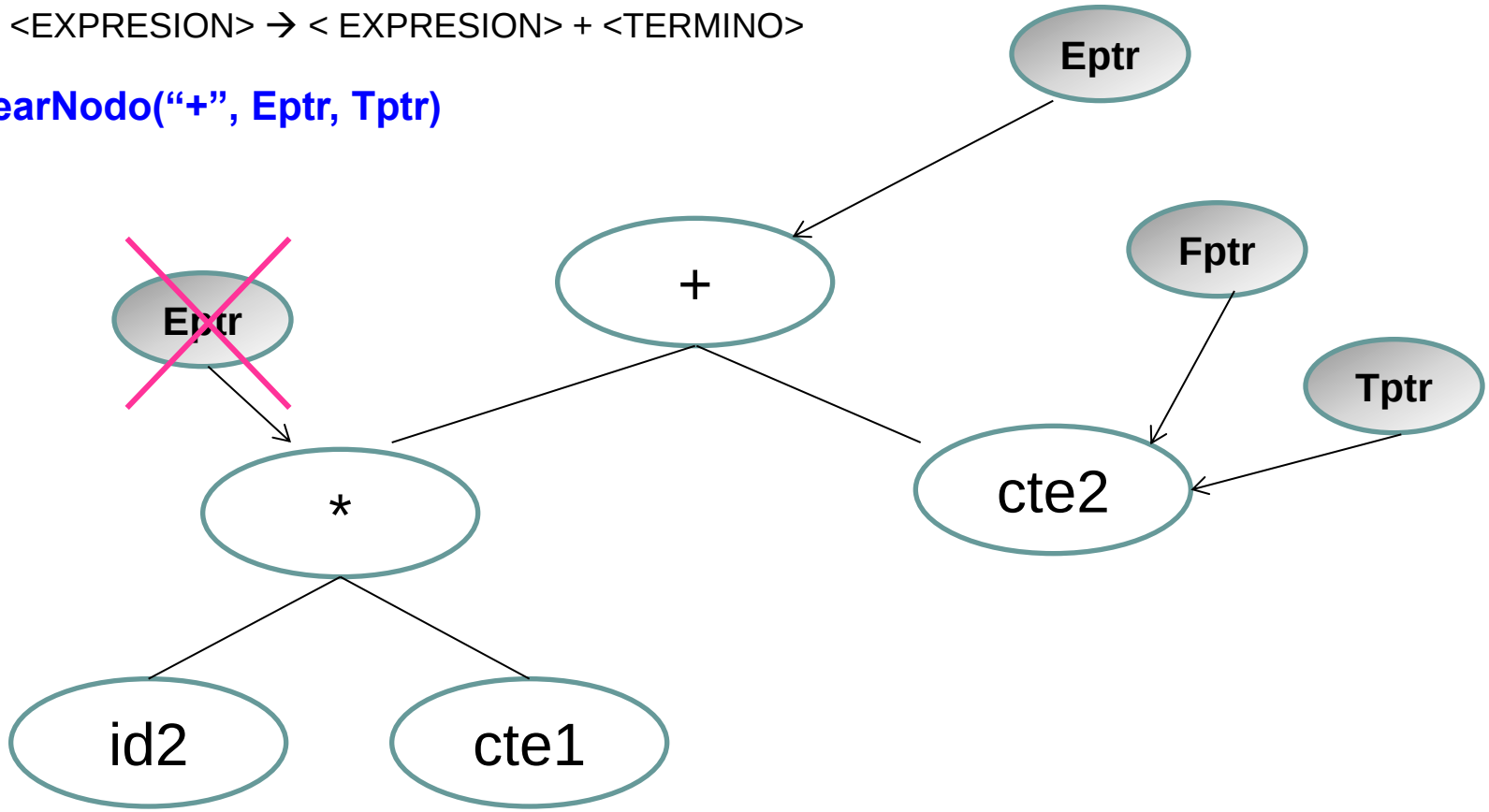
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2 * cte1~~ + cte2
Lista de Reglas : ~~6 5 7 4 3 7 5~~ 2 1

Regla 2 : <EXPRESION> → < EXPRESION> + <TERMINO>

Eptr = crearNodo("+", Eptr, Tptr)



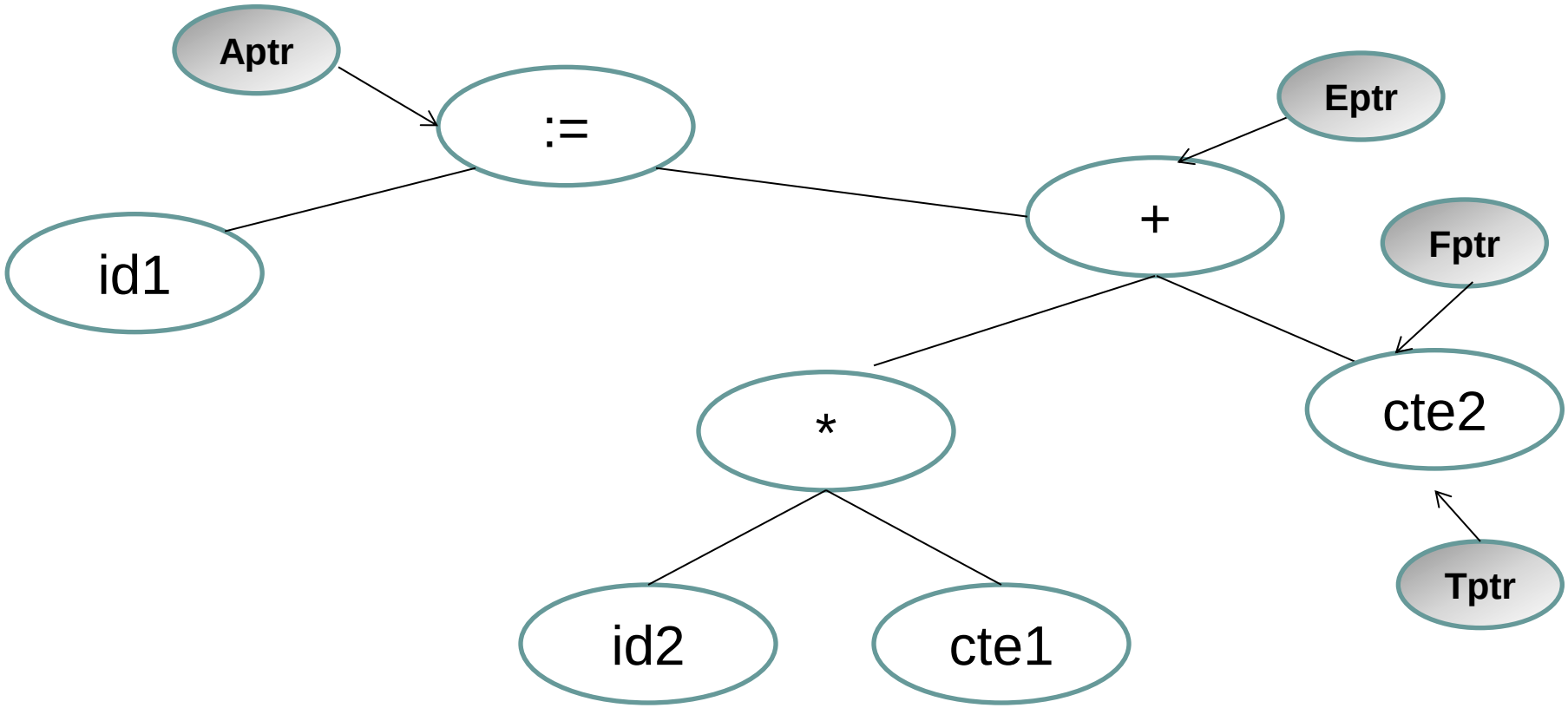
Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Programa : id1 := ~~id2 * cte1 + cte2~~
Lista de Reglas : ~~6 5 7 4 3 7 5 2~~ 1

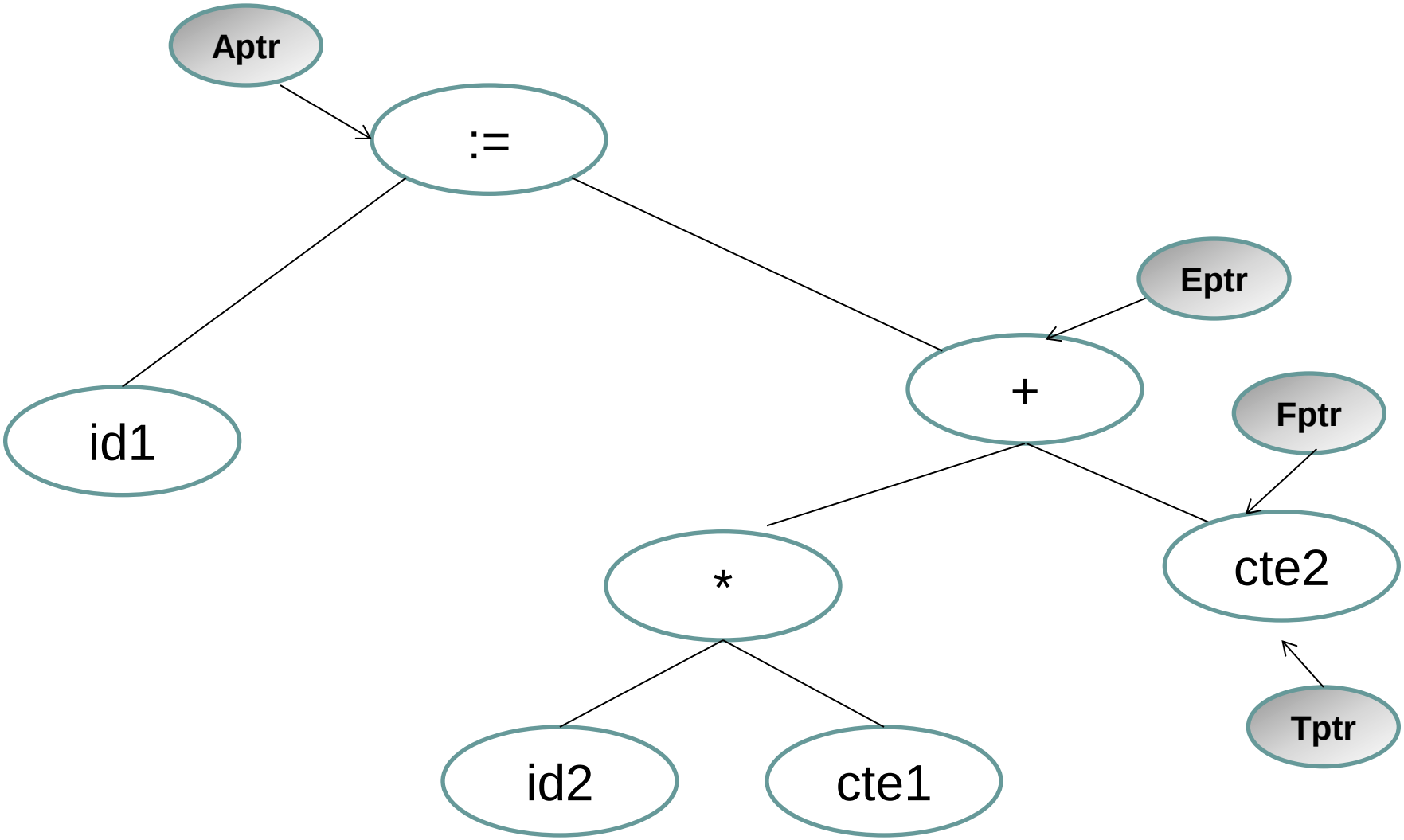
Regla 1 : <ASIG> → id := <EXPRESION>

Aptr = crearNodo(":=", crearHoja(id), Eptr)



Generación de Código Intermedio

LISTA DE REGLAS → ÁRBOL SINTÁCTICO



Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Se requiere :

- Una rutina que se asocia a cada regla
  - insertar_en_polaca () // inserta un token en la estructura elegida para la polaca inversa

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

1.	<ASIG> → id := <EXPRESION>	insertar_en_polaca (id) ; insertar_en_polaca (:=)
2.	<EXPRESION> → <EXPRESION> + <TERMINO>	insertar_en_polaca (+)
3.	<EXPRESION> → <TERMINO>	
4.	<TERMINO> → < TERMINO> * <FACTOR>	insertar_en_polaca (*)
5.	<TERMINO> → <FACTOR>	
6.	<FACTOR> → id	insertar_en_polaca (id)
7.	<FACTOR> → cte	insertar_en_polaca (cte)

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Programa : x:= z * 17.1 + 8 (id1 := id2 * cte1 + cte2)
Lista de Reglas : 6 5 7 4 3 7 5 2 1

Regla 6 : <FACTOR> → id

insertar_en_polaca (id)

Regla 5 <TERMINO> → <FACTOR>

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Programa : x:= z * 17.1 + 8
Lista de Reglas : ~~6 5~~ 7 4 3 7 5 2 1

Regla 7 : <FACTOR> → cte

insertar_en_polaca (cte)

z	17.1							

Regla 4 : <TERMINO> → < TERMINO> * <FACTOR>

insertar_en_polaca (*)

z	17.1	*						

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Programa : x:= z * 17.1 + 8
Lista de Reglas : ~~6 5 7 4 3 7~~ 5 2 1

Regla 3 : <EXPRESION> → <TERMINO>

z	17.1	*						

Regla 7 : <FACTOR> → cte

insertar_en_polaca (cte)

z	17.1	*	8					

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Programa : x:= z * 17.1 + 8
Lista de Reglas : ~~6 5 7 4 3 7~~ 5 2 1

Regla 5 : <TERMINO> → <FACTOR>

z	17.1	*	8					

Regla 2 : <EXPRESION> → <EXPRESION> + <TERMINO>

insertar_en_polaca (+)

z	17.1	*	8	+				

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Programa : x:= z * 17.1 + 8
Lista de Reglas : ~~6 5 7 4 3 7 5 2~~ 1

Regla 1 : <ASIG> → id := <EXPRESION>

insertar_en_polaca (id) ;
insertar_en_polaca (:=)

z	17.1	*	8	+	x	:=		

Generación de Código Intermedio

LISTA DE REGLAS → POLACA INVERSA

Ejemplo: **: x:= z * 17.1 + 8**

Lista de Reglas → Polaca Inversa

z	**17.1**	*****	**8**	**+**	**x**	**:=**

Polaca Inversa (Definición Formal)

x	**z**	**17.1**	*****	**8**	**+**	**:=**

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Se requiere :

- Un variable indice por cada elemento no terminal
- Una rutina que se asocia a cada regla
  - crear_terceto () // crea tercetos en algún archivo
    // numera el terceto
    // asigna el número a la variable índice

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

1. <ASIG> → id := <EXPRESION>
2. <EXPRESION> → <EXPRESION> + <TERMINO>
3. <EXPRESION> → <TERMINO>
4. <TERMINO> → < TERMINO> * <FACTOR>
5. <TERMINO> → <FACTOR>
6. <FACTOR> → id
7. <FACTOR> → cte

Start : <ASIG>

Una variable indice por cada elemento no terminal

Aind

Eind

Tind

Find

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

1.	<ASIG> → id := <EXPRESION>	crearTerceto(“:=”, id, Eind)
2.	<EXPRESION> → <EXPRESION> + <TERMINO>	Eind = crearTerceto(“+”, Eind, Tind)
3.	<EXPRESION> → <TERMINO>	Eind = Tind
4.	<TERMINO> → < TERMINO> * <FACTOR>	Tind = crearTerceto(“*”, Tind, Find)
5.	<TERMINO> → <FACTOR>	Tind = Find
6.	<FACTOR> → id	Find = crearTerceto(id)
7.	<FACTOR> → cte	Find = crearTerceto(cte)

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Programa : id1 := id2 * cte1 + cte2
Lista de Reglas : 6 5 7 4 3 7 5 2 1

Regla 6 : <FACTOR> → id

Find = crearTerceto(id)

[11] (id2, _ , _)

Find	11
Tind	
Eind	

Regla 5 : <TERMINO> → <FACTOR>

Tind = Find

[11] (id2, _ , _)

Find	11
Tind	11
Eind	

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Programa : id1 := ~~id2~~ * cte1 + cte2
Lista de Reglas : ~~6 5~~ 7 4 3 7 5 2 1

Regla 7 : <FACTOR> → cte

Find = crearTerceto(cte)

[11] (id2, _ , _)
[12] (cte1, _ , _)

Find	12
Tind	11
Eind	

Regla 4 : <TERMINO> → <TERMINO> * <FACTOR>

Tind = crearTerceto("*", Tind, Find)

[11] (id2, _ , _)
[12] (cte1, _ , _)
[13] (* ,[11] , [12])

Find	12
Tind	13
Eind	

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Programa : id1 := ~~id2 * cte1~~ + cte2
Lista de Reglas : ~~6 5 7 4~~ 3 7 5 2 1

Regla 3 : <EXPRESION> → <TERMINO>

Eind = Tind

[11]	**(id2, _ , _)**
[12]	**(cte1, _ , _)**
[13]	**(* ,[11] , [12])**

Find	12
Tind	13
Eind	13

Regla 7 : <FACTOR> → cte

Find = crearTerceto(cte)

[11]	**(id2, _ , _)**
[12]	**(cte1, _ , _)**
[13]	**(* ,[11] , [12])**
[14]	**(cte2, _ , _)**

Find	14
Tind	13
Eind	13

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Programa : id1 := ~~id2 * cte1~~ + ~~cte2~~

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

Regla 5 : <TERMINO> → <FACTOR>

Tind = Find

[11] (id2, _ , _)
[12] (cte1, _ , _)
[13] (* ,[11] , [12])
[14] (cte2, _ , _)

Find	14
Tind	14
Eind	13

Regla 2 : <EXPRESION> → <EXPRESION> + <TERMINO>

Eind = crearTerceto("+", Eind, Tind)

[11] (id2, _ , _)
[12] (cte1, _ , _)
[13] (* ,[11] , [12])
[14] (cte2, _ , _)
[15] (+ ,[13] , [14])

Find	14
Tind	14
Eind	15

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Programa : id1 := ~~id2 * cte1 + cte2~~
Lista de Reglas : ~~6 5 7 4 3 7 5 2~~ 1

<ASIG> → id := <EXPRESION>

crearTerceto(":=", id, Eind)

[11] (id2, _ , _)
[12] (cte1, _ , _)
[13] (* ,[11] , [12])
[14] (cte2, _ , _)
[15] (+ ,[13] , [14])
[16] (:= , id1 ,[15])

Find	14
Tind	14
Eind	15

Generación de Código Intermedio

LISTA DE REGLAS → TERCETOS

Lista de Tercetos

```
[ 11 ]    (id2, _ , _ )
[ 12 ]    (cte1, _ , _ )
[ 13 ]    (* ,[11] , [12])
[ 14 ]    (cte2, _ , _ )
[ 15 ]    (+ ,[13] , [14])
[ 16 ]    (:= , id1 ,[15])
```

Lista de Tercetos Optimizados

```
[ 11 ]    (* , id2, cte1)
[ 12 ]    (+ ,[13] , cte2)
[ 13 ]    (:= , id1 ,[12])
```

Generación de Código Intermedio

¿Preguntas?