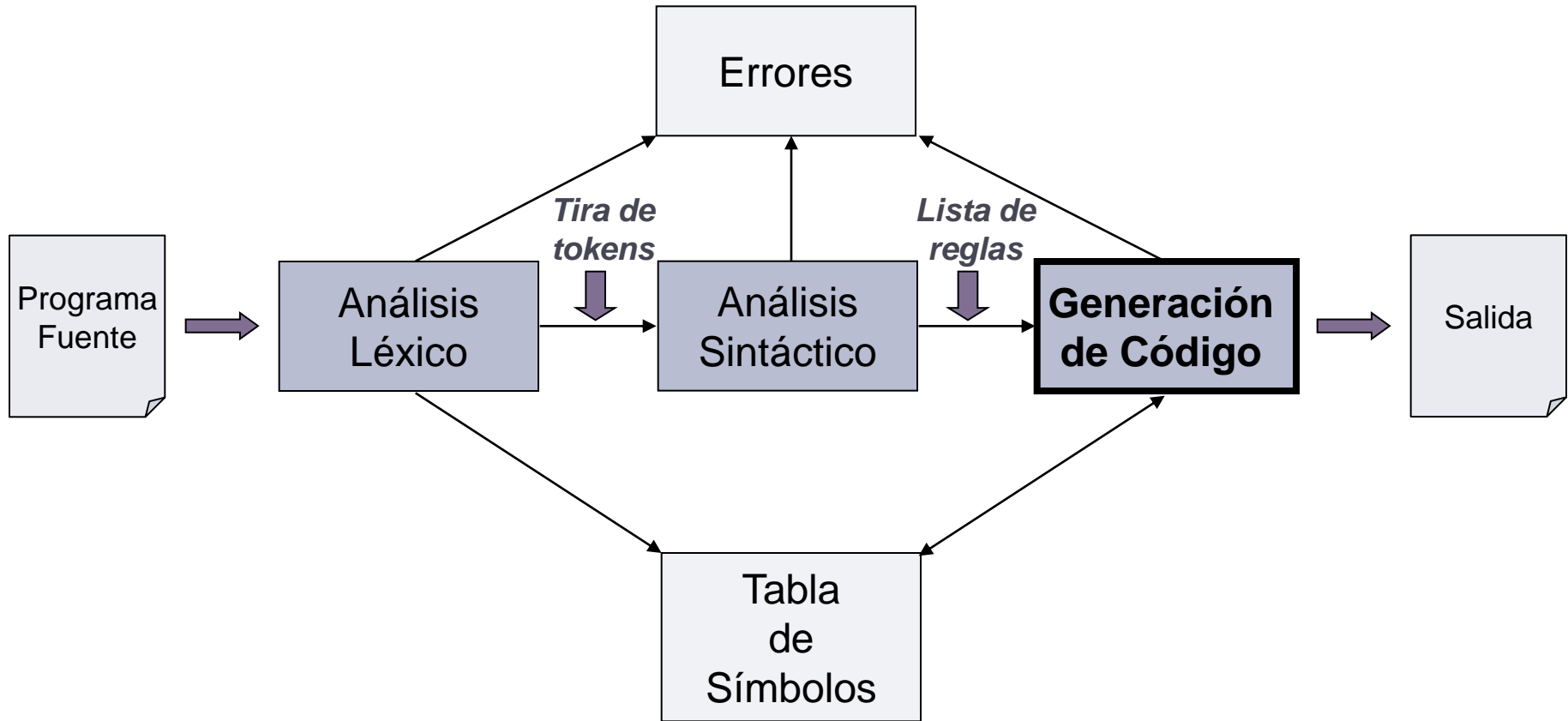


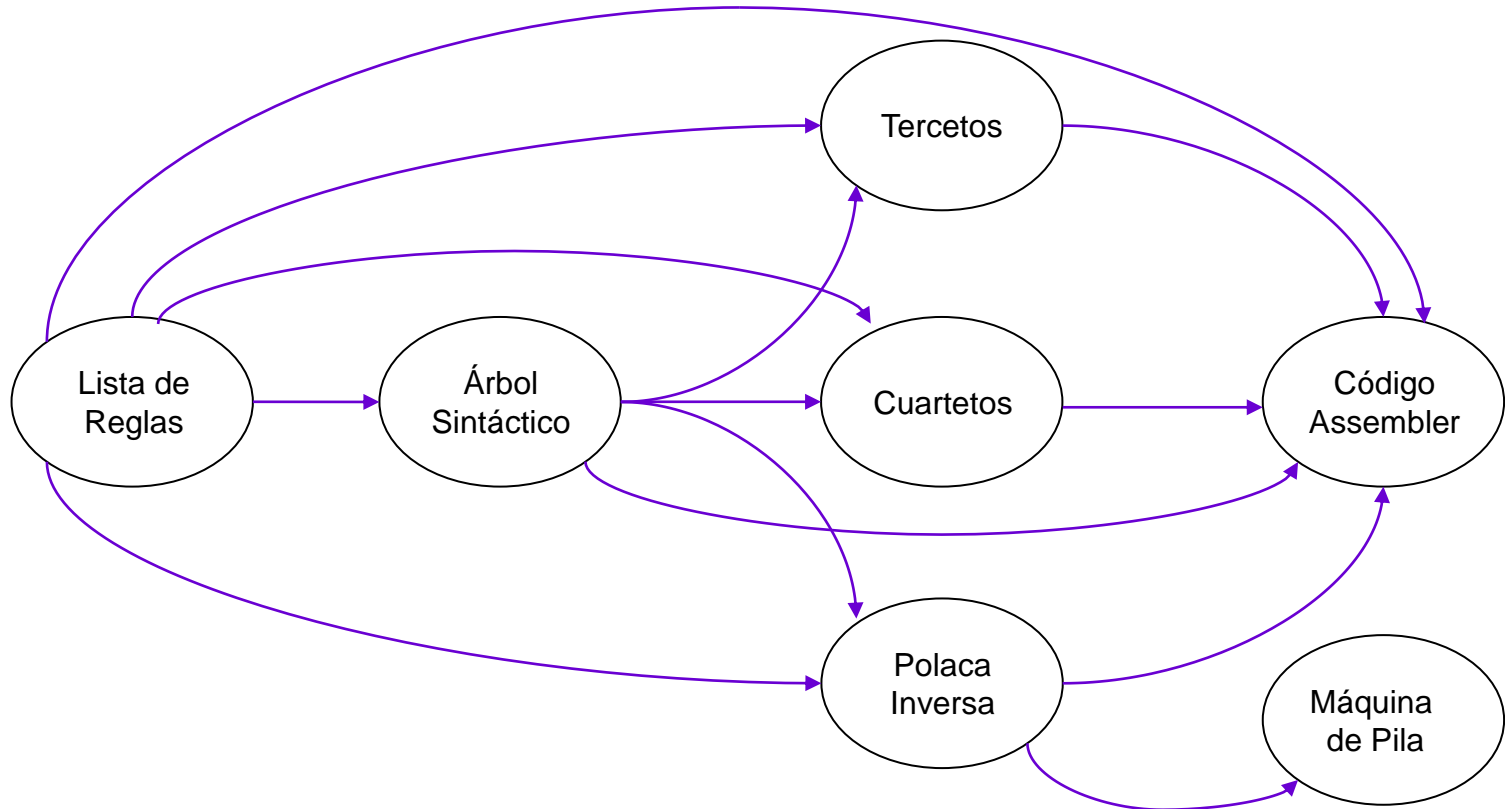
Diseño de Compiladores I

Generación de Código

Fases de la Compilación



Generación de Código



Traducción dirigida por la Sintaxis

- ▶ Se asocia información a los símbolos de la gramática, mediante reglas semánticas asociadas a las producciones.
- ▶ Cada símbolo tendrá un conjunto de atributos asociados.
- ▶ Atributos:
 - ▶ cadena,
 - ▶ tipo,
 - ▶ valor numérico,
 - ▶ posición en memoria,
 - ▶ etc.

Producciones + Reglas semánticas → Dependencias entre los atributos

Definiciones dirigidas por la Sintaxis

- ▶ El valor del atributo de un nodo de un **Árbol de Parsing** se define mediante una regla semántica asociada a la producción donde se genera ese nodo.
- ▶ **Atributos Sintetizados**
 - ▶ Se calculan a partir de los valores de los atributos de los hijos del nodo en el **Árbol de Parsing**.
- ▶ **Atributos Heredados**
 - ▶ Se calculan a partir de los valores de los atributos de los hermanos y el padre del nodo en el **Árbol de Parsing**.
 - ▶ Expresan dependencias entre una construcción y su contexto.

Atributos Sintetizados

► Ejemplo: Calculadora de escritorio

Reglas de la Gramática	Regla Semántica
1. $L \rightarrow E \text{'\n'}$	Print (E.valor)
2. $E \rightarrow E + T$	E.valor := E.valor + T.valor
3. $E \rightarrow T$	E.valor = T.valor
4. $T \rightarrow T * F$	T.valor := T.valor * F.valor
5. $T \rightarrow F$	T.valor := F.valor
6. $F \rightarrow (E)$	F.valor := E.valor
7. $F \rightarrow \text{num}$	F.valor := num.valor

Atributo sintetizado para el token num
(Lo proporciona el Analizador Léxico)

► Los atributos sintetizados se calculan fácilmente en Parsing Ascendente.

Atributos Heredados

- ▶ Ejemplo: Distribuir información de tipos a los identificadores de una declaración

Reglas de la Gramática	Regla Semántica
1. $D \rightarrow T L$	$L.her := T.tipo;$
2. $T \rightarrow int$	$T.tipo := integer;$
3. $T \rightarrow real$	$T.tipo := real;$
4. $L \rightarrow id \text{ , } L_1$	$añadir_tipo(id.ptr, L.her); L_1.her := L.her;$
5. $L \rightarrow id$	$añadir_tipo(id.ptr, L.her);$

Referencia a la entrada de id en la Tabla de Símbolos (Proporcionada por el Analizador Léxico)

- ▶ Los atributos heredados se calculan fácilmente en Parsing Descendente.

Chequeos Semánticos

Semántica Estática

- ▶ Se pueden efectuar en tiempo de compilación
- ▶ Ejemplos:
 - ▶ Comprobación de tipos
 - ▶ Comprobación de flujo de control
 - ▶ Comprobación de unicidad
 - ▶ Comprobaciones relacionadas con nombres

Semántica Dinámica

- ▶ Sólo se pueden efectuar en tiempo de ejecución
- ▶ Ejemplos:
 - ▶ Control de subíndices de arreglos

Tipos y Conversiones

LISTA DE REGLAS → Árbol Sintáctico

LISTA DE REGLAS → Tercetos

LISTA DE REGLAS → Polaca Inversa

Tipos

$w := x + y * z$

Con:

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



Conversiones

- ▶ **Implícitas (Coerciones)**

- ▶ Las realiza automáticamente el compilador.
- ▶ En principio, se limitan a situaciones donde no hay pérdida de información.

- ▶ **Explícitas**

- ▶ Las debe escribir el programador.



Lista de Reglas \rightarrow Árbol Sintáctico

W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

- a) Árbol Sintáctico sin información de tipos ni incorporación de conversiones (implícitas)
- b) Árbol Sintáctico con información de tipos
- c) Árbol Sintáctico con información de tipos e incorporación de conversiones (implícitas)



Lista de Reglas \rightarrow Árbol Sintáctico (a)

sin información de tipos ni incorporación de conversiones

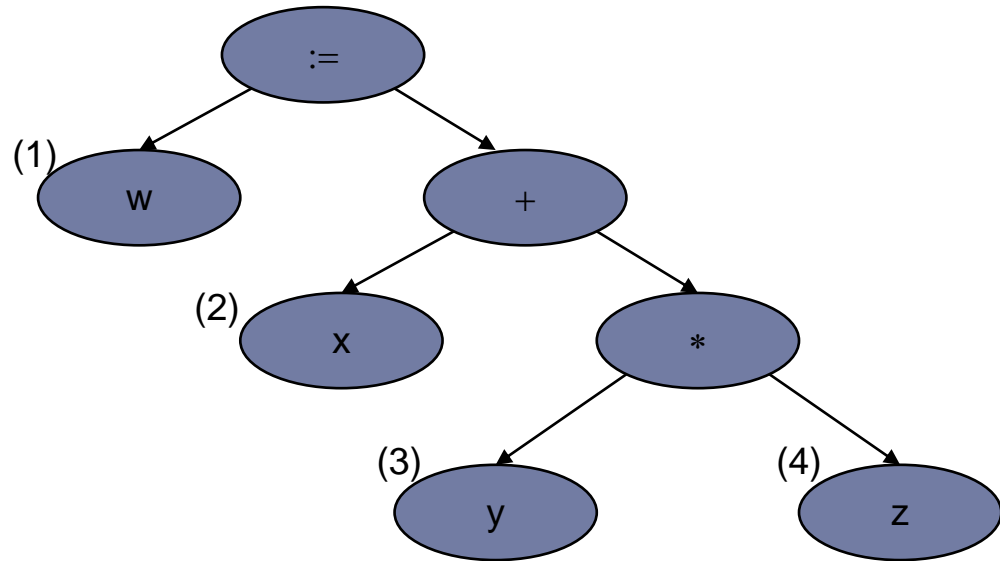
W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



- ▶ El chequeo de tipos se posterga a una traducción posterior
 - ▶ La incorporación de conversiones se posterga a una traducción posterior.
-



Lista de Reglas \rightarrow Árbol Sintáctico (b) con información de tipos

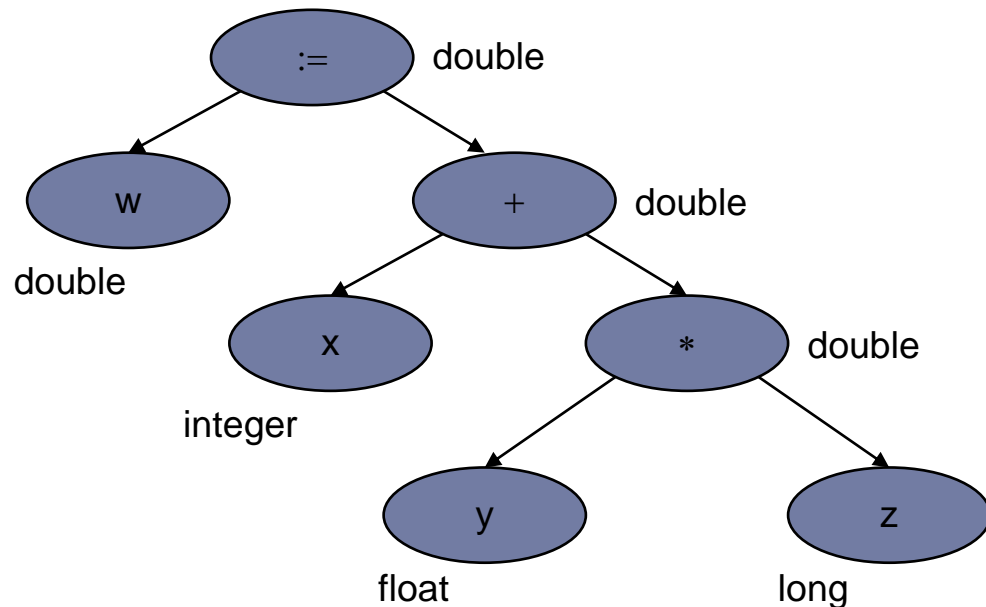
W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



- ▶ La incorporación de conversiones se posterga a una traducción posterior.
-



Lista de Reglas \rightarrow Árbol Sintáctico (c)

con información de tipos e incorporación de conversiones

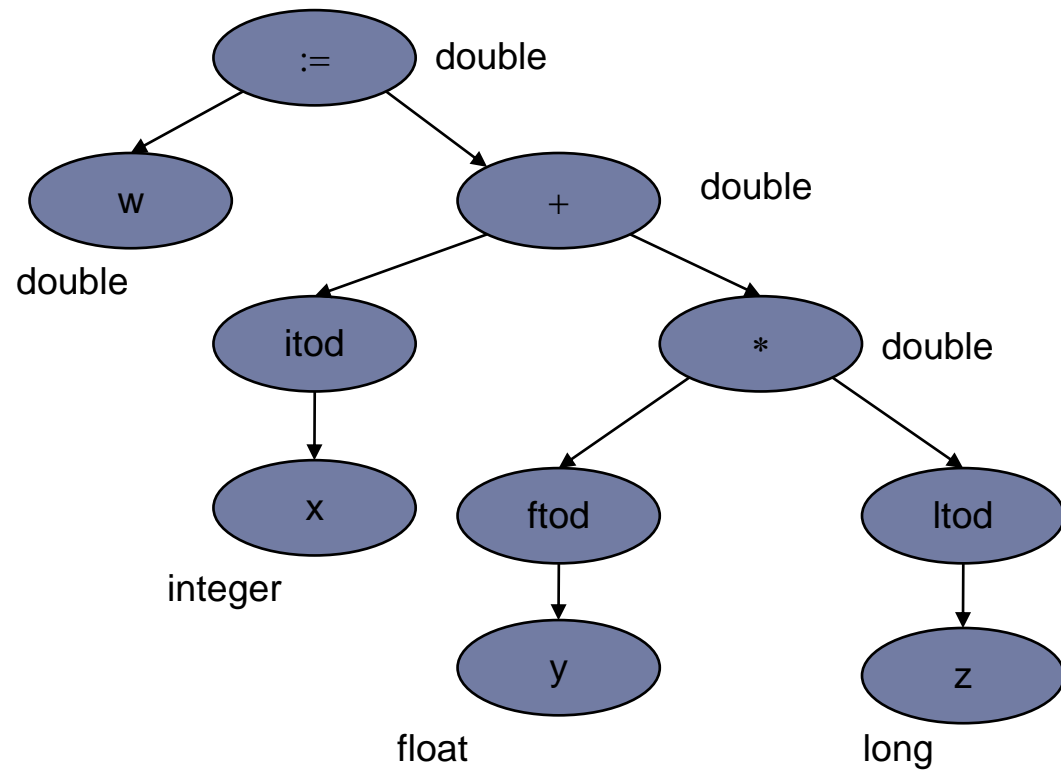
W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



Lista de Reglas \rightarrow Árbol Sintáctico (a)

sin tipos ni conversiones

Reglas de la Gramática	Acciones Semánticas
1. $A \rightarrow id := E$	$A.ptr = crear_nodo(':='; crear_hoja(id.ptr); E.ptr)$ (*)
2. $E \rightarrow E + T$	$E.ptr = crear_nodo('+; E.ptr; T.ptr)$
3. $E \rightarrow T$	$E.ptr = T.ptr;$
4. $T \rightarrow T * F$	$T.ptr = crear_nodo('*; T.ptr; F.ptr)$
5. $T \rightarrow F$	$T.ptr = F.ptr;$
6. $F \rightarrow id$	$F.ptr = crear_hoja(id.ptr)$ (*)
7. $F \rightarrow cte$	$F.ptr = crear_hoja(cte.ptr)$ (**)

(*) $id.ptr$ = Ref a la entrada del identificador en la Tabla de Símbolos

(**) $cte.ptr$ = Ref a la entrada de la constante en la Tabla de Símbolos



Lista de Reglas \rightarrow Árbol Sintáctico (a)

sin tipos ni conversiones

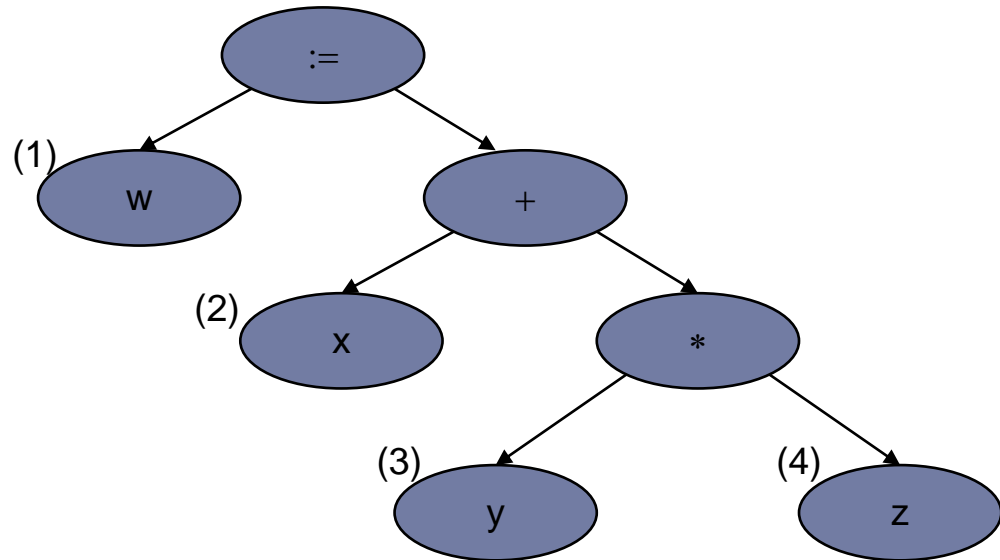
W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



Lista de Reglas \rightarrow Árbol Sintáctico (b) con información de tipos

Reglas de la Gramática	Acciones Semánticas
1. $A \rightarrow id := E$	A.ptr = crear_nodo(':=' ; crear_hoja(id.ptr) ; E.ptr) ; (*) A.Tipo = id.tipo ;
2. $E \rightarrow E + T$	E.ptr = crear_nodo('+' ; E.ptr ; T.ptr) ; E.Tipo = f ₊ (E.tipo , T.tipo)
3. $E \rightarrow T$	E.ptr = T.ptr ; E.tipo = T.tipo ;
4. $T \rightarrow T * F$	T.ptr = crear_nodo('*' ; T.ptr ; F.ptr) ; T.tipo = f _* (T.tipo , F.tipo) ;
5. $T \rightarrow F$	T.ptr = F.ptr ; T.tipo = F.tipo ;
6. $F \rightarrow id$	F.ptr = crear_hoja(id.ptr) ; F.tipo = id.tipo ; (*)
7. $F \rightarrow cte$	F.ptr = crear_hoja(cte.ptr) ; F.tipo = cte.tipo ; (**)

(*) id.ptr = Ref a la entrada del identificador en la Tabla de Símbolos

(**) cte.ptr = Ref a la entrada de la constante en la Tabla de Símbolos

Lista de Reglas → Árbol Sintáctico (b)

Tablas de Compatibilidad de Tipos

f_+	integer	long	float	double
integer	integer	long	float	double
long	long	long	float	double
float	float	float	float	double
double	double	double	double	double

f_*	integer	long	float	double
integer	long	long	float	double
long	long	long	double	double
float	float	double	double	double
double	double	double	double	double

- Toda la semántica de los tipos está en la matriz
- ▶ Habrá una matriz para cada operador de la gramática

Lista de Reglas → Árbol Sintáctico (b)

Tablas de Compatibilidad de Tipos

f_+	integer	long	float	double	string
integer	integer	long	float	double	X
long	long	long	float	double	X
float	float	float	float	double	X
double	double	double	double	double	X
string	X	X	X	X	string

- Si una combinación está prohibida, en la celda correspondiente habrá una “X”, o un código de error para esa situación



Lista de Reglas \rightarrow Árbol Sintáctico (b) con información de tipos

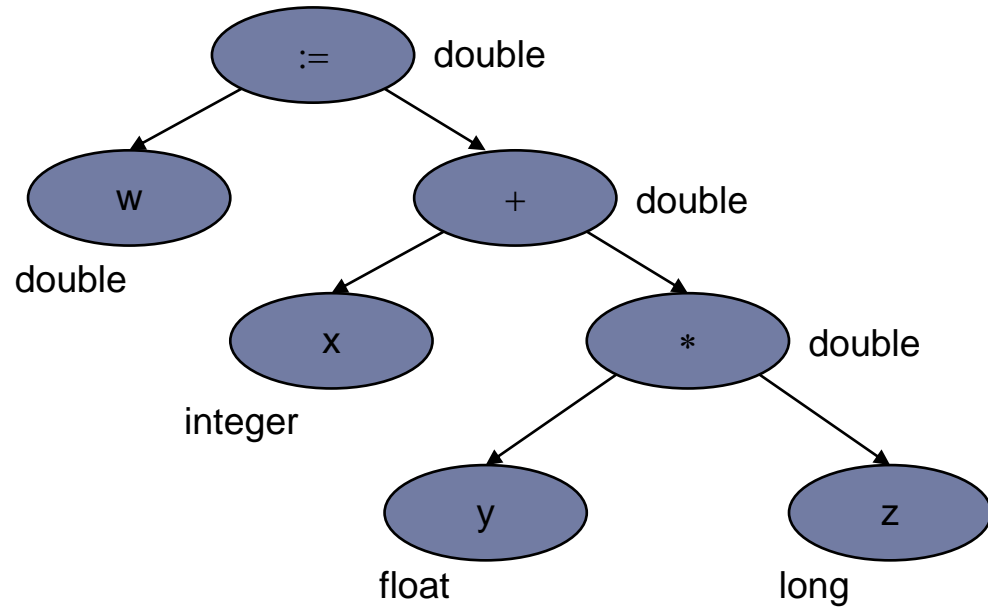
W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



Lista de Reglas \rightarrow Árbol Sintáctico (b) con información de tipos

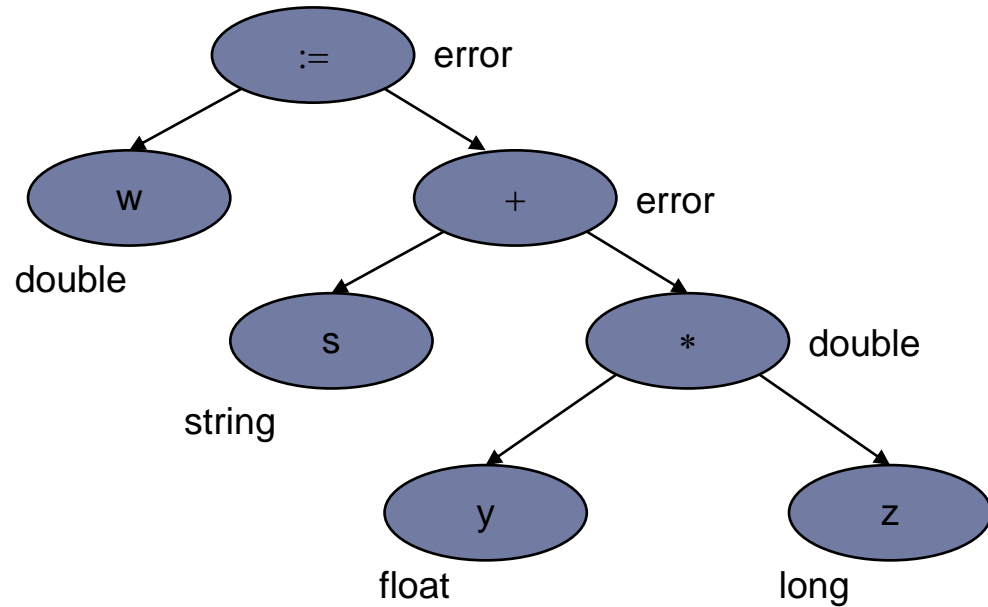
W := S + y * Z

w de tipo double

s de tipo string

y de tipo float

z de tipo long



Lista de Reglas \rightarrow Árbol Sintáctico (C)

con información de tipos e incorporación de conversiones

Reglas de la Gramática	Acciones Semánticas
1. $A \rightarrow id := E$	<pre>E'.ptr = agregar_nodo_conversion(f:=[3] ; id.tipo ; E.tipo); A.ptr = crear_nodo(':=' ; crear_hoja(id.ptr) ; E'.ptr) ; (*) A.Tipo = id.tipo ;</pre>
2. $E \rightarrow E + T$	<pre>E'.ptr = agregar_nodo_conversion(f+[1] ; E.tipo ; T.tipo) ; T'.ptr = agregar_nodo_conversion(f+[3] ; E.tipo ; T.tipo) ; E.ptr = crear_nodo('+' ; E'.ptr ; T'.ptr) ; E.Tipo = f_+[2](E.tipo , T.tipo) ;</pre>
3. $E \rightarrow T$	<pre>E.ptr = T.ptr ; E.tipo = T.tipo ;</pre>
4. $T \rightarrow T * F$	<pre>T'.ptr = agregar_nodo_conversion(f*[1] ; T.tipo ; F.tipo) ; F'.ptr = agregar_nodo_conversion(f*[3] ; T.tipo ; F.tipo) ; T.ptr = crear_nodo('*' ; T'.ptr ; F'.ptr) ; T.tipo = f_*[2](T.tipo , F.tipo) ;</pre>
5. $T \rightarrow F$	<pre>T.ptr = F.ptr ; T.tipo = F.tipo ;</pre>
6. $F \rightarrow id$	<pre>F.ptr = crear_hoja(id.ptr) ; F.tipo = id.tipo ; (*)</pre>
7. $F \rightarrow cte$	<pre>F.ptr = crear_hoja(cte.ptr) ; F.tipo = cte.tipo ; (**)</pre>

(*) id.ptr = Ref a la entrada del identificador en la Tabla de Símbolos (**) cte.ptr = Ref a la entrada de la constante en la Tabla de Símbolos

Lista de Reglas → Árbol Sintáctico (b)

Tabla de Tipos y Conversiones

f*	integer			long			float			double		
integer	itol	long	itol	itol	long	-	itof	float	-	itod	double	-
long	-	long	itol	-	long	-	ltof	double	-	ltod	double	-
float	-	float	itof	ftod	double	ltod	ftod	double	ftod	ftod	double	-
double	-	double	itod	.	double	ltod	-	double	ftod	-	double	-

- Toda la semántica de los tipos está en la matriz
 - Habrá una matriz para cada operador de la gramática
 - Si una combinación está prohibida, en la celda correspondiente habrá una “X”, o un código de error para esa situación
-



Lista de Reglas \rightarrow Tercetos

W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

- a) Tercetos sin información de tipos ni incorporación de conversiones (implícitas)
- b) Tercetos con información de tipos
- c) Tercetos con información de tipos e incorporación de conversiones (implícitas)



Lista de Reglas \rightarrow Polaca Inversa

W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

(a) Polaca Inversa sin información de tipos ni
incorporación de conversiones (implícitas)



Árbol Sintáctico → Tercetos

(a) (a)

Árbol Sintáctico sin tipos ni conversiones → Tercetos sin tipos ni conversiones

(El chequeo de tipos y la incorporación de conversiones se posterga a la traducción Tercetos → Assembler)

(a) (b)

Árbol Sintáctico sin tipos ni conversiones → Tercetos con tipos

(La incorporación de conversiones se posterga a la traducción Tercetos → Assembler)

(a) (c)

Árbol Sintáctico sin tipos ni conversiones → Tercetos con tipos y conversiones

(b) (b)

Árbol Sintáctico con tipos → Tercetos con tipos

(La incorporación de conversiones se posterga a la traducción Tercetos → Assembler)

(b) (c)

Árbol Sintáctico con tipos → Tercetos con tipos y conversiones

(c) (c)

Árbol Sintáctico con tipos y conversiones → Tercetos con tipos y conversiones



Árbol Sintáctico \rightarrow Polaca Inversa

(a) (a)

Árbol Sintáctico sin tipos ni conversiones \rightarrow Polaca sin tipos ni conversiones

(El chequeo de tipos y la incorporación de conversiones se posterga a la traducción Polaca Inversa \rightarrow Assembler)

(b) (b)

Árbol Sintáctico con tipos \rightarrow Polaca con tipos

(La incorporación de conversiones se posterga a la traducción Polaca Inversa \rightarrow Assembler)

(c) (c)

Árbol Sintáctico con tipos y conversiones \rightarrow Polaca con tipos y conversiones

