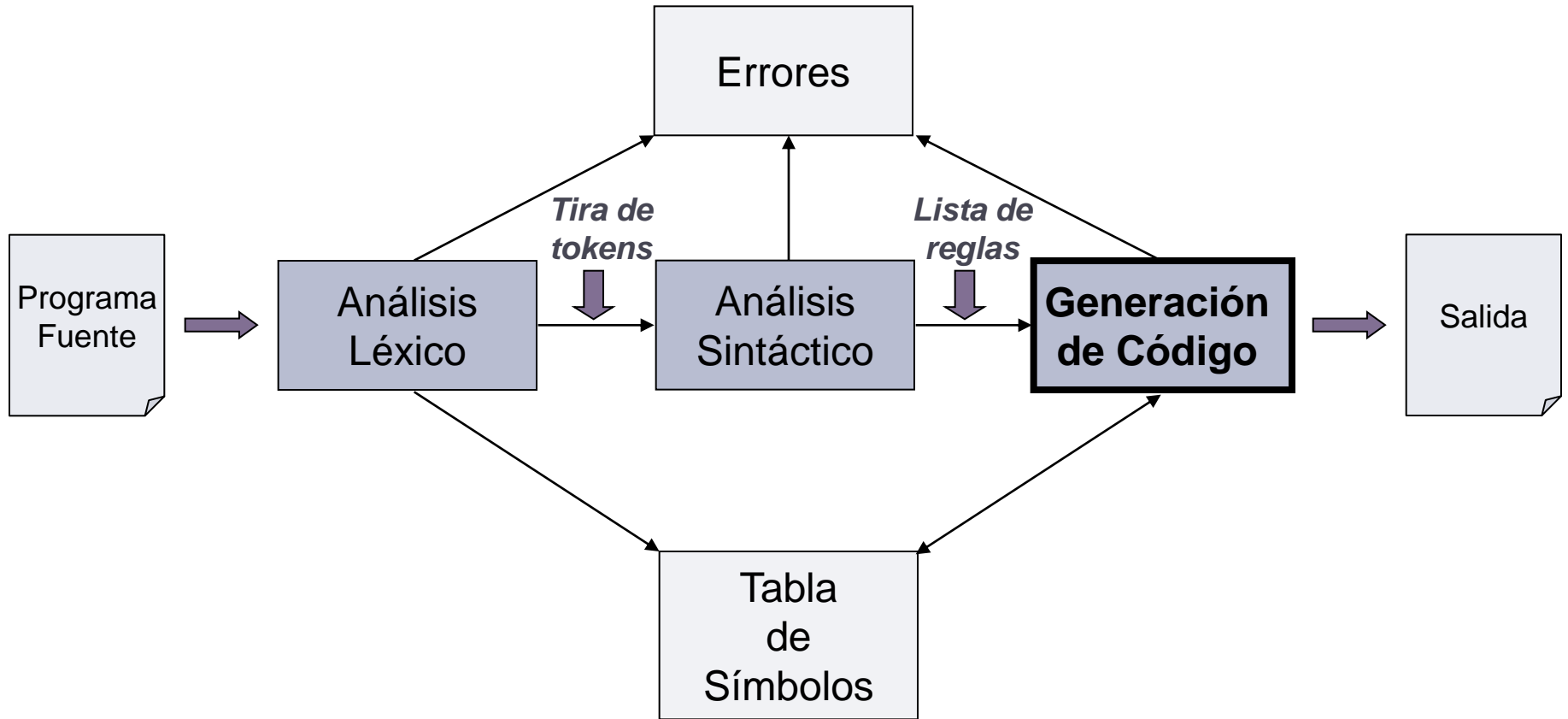


Diseño de Compiladores I

Generación de Código

Fases de la Compilación



Generación de Código

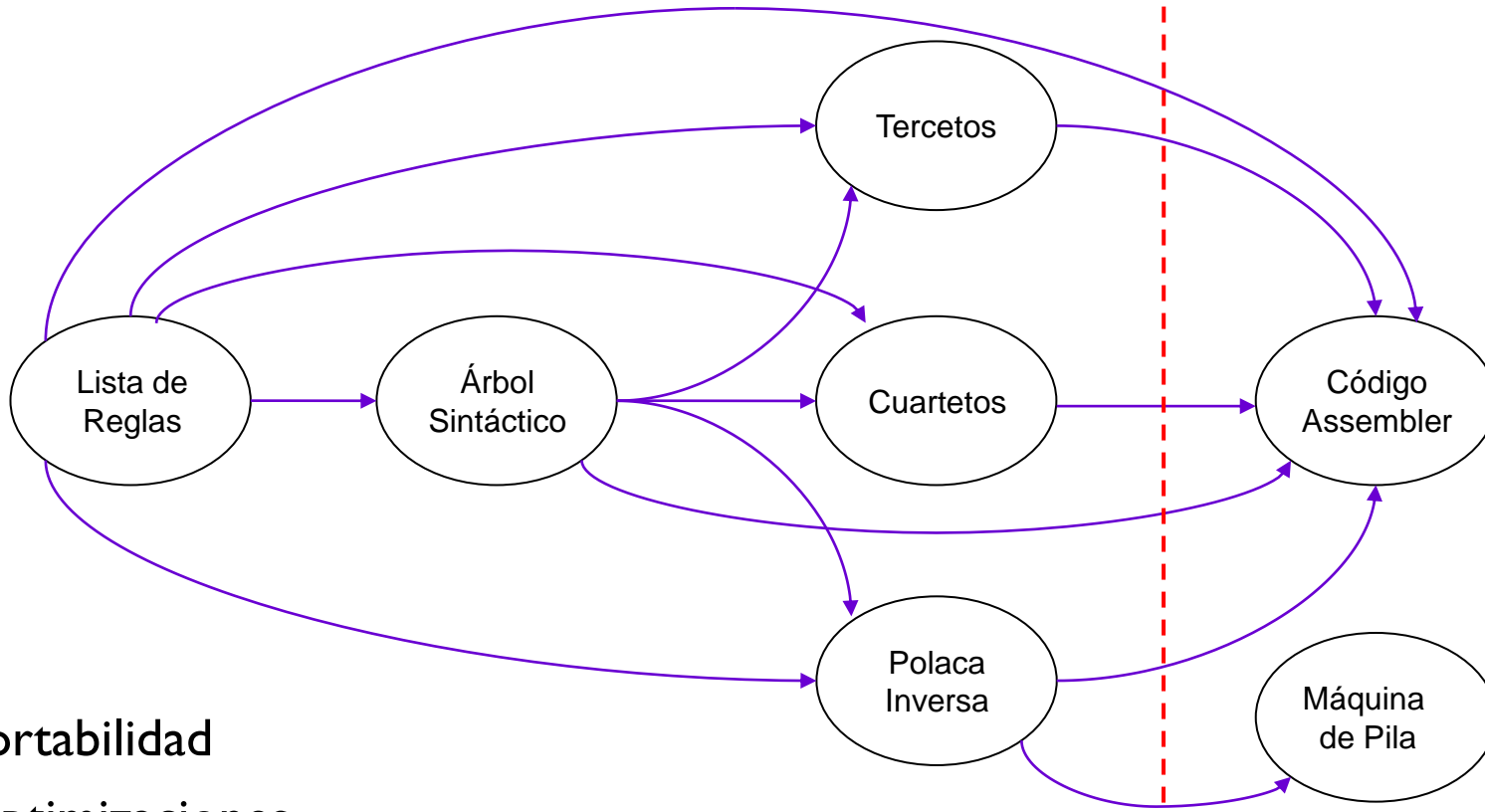
Grafo 1

Sentencias ejecutables

$x := x + 2;$

Sólo depende
del lenguaje

Depende del S.O.
y de la máquina



- ▶ Portabilidad
- ▶ Optimizaciones

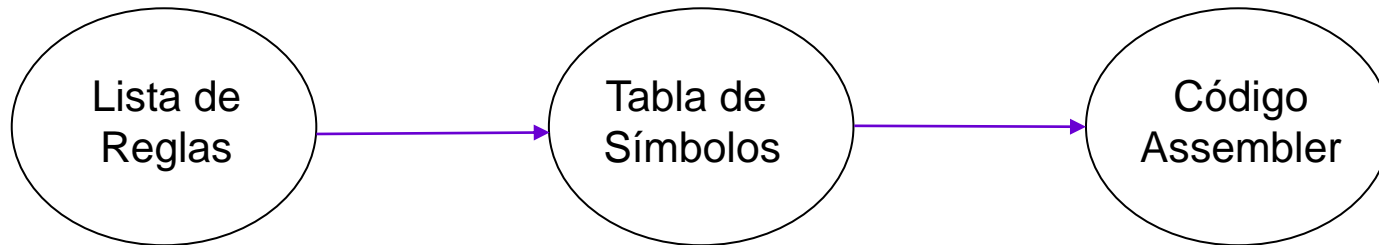
Generación de Código

Generación de Código

Grafo 2: Sentencias declarativas

Sentencias declarativas

```
int x, y;
```



Generación de Código

Traducción Dirigida por la Sintaxis

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 de caracteres (lexema),
 - ▶ tipo,
 - ▶ valor numérico (para optimizaciones, por ejemplo),
 - ▶ 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



Generación de código

Lista de Reglas → Árbol Sintáctico

Lista de Reglas → Tercetos

Lista de Reglas → Polaca Inversa

Tipos y Conversiones

Tipos

$w := x + y * z$

Con:

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

¿Es válida la expresión?

¿Es válida la asignación?



¿Cómo se detecta una
incompatibilidad de tipos?



Tablas de Compatibilidad de Tipos

- ▶ Para poder realizar una operación, ambos operandos deben tener el mismo tipo en el momento de hacer la operación.

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
- ▶ Si una combinación está permitida, en la celda correspondiente estará el tipo resultante, y ambos operandos deberán ser convertidos al tipo del resultado.

Tablas 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	float	double
float	float	float	double	double
double	double	double	double	double

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



Conversiones

Si una combinación está permitida, en la celda correspondiente estará el tipo resultante, y ambos operandos deberán ser convertidos al tipo del resultado.

- ▶ **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.



Tipos y Conversiones

LISTA DE REGLAS → ÁRBOL SINTÁCTICO

Lista de Reglas → Árbol Sintáctico

(Información de tipos y conversiones implícitas)

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

(Información de tipos y conversiones implícitas)

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)



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 información de tipos ni incorporación de conversiones

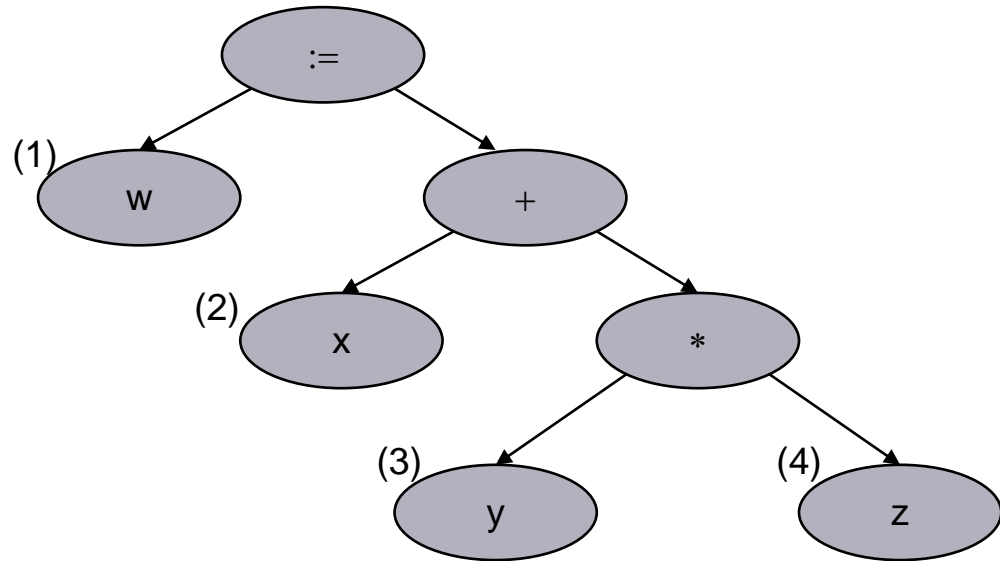
W := X + y * Z

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 implícitas se posterga a una traducción posterior.
-



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

b) Árbol Sintáctico con información de tipos



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 ; (**)

Matriz de tipos

Matriz de tipos

(*) 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 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	float	double
float	float	float	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 \rightarrow Árbol Sintáctico (b) con información de tipos

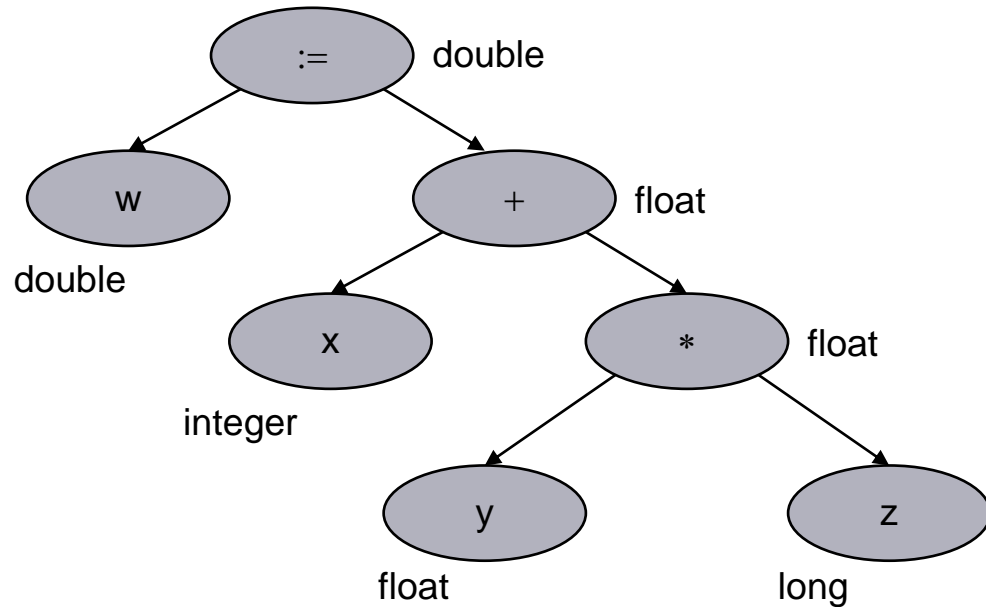
W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long



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



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

► Ejemplo con error:

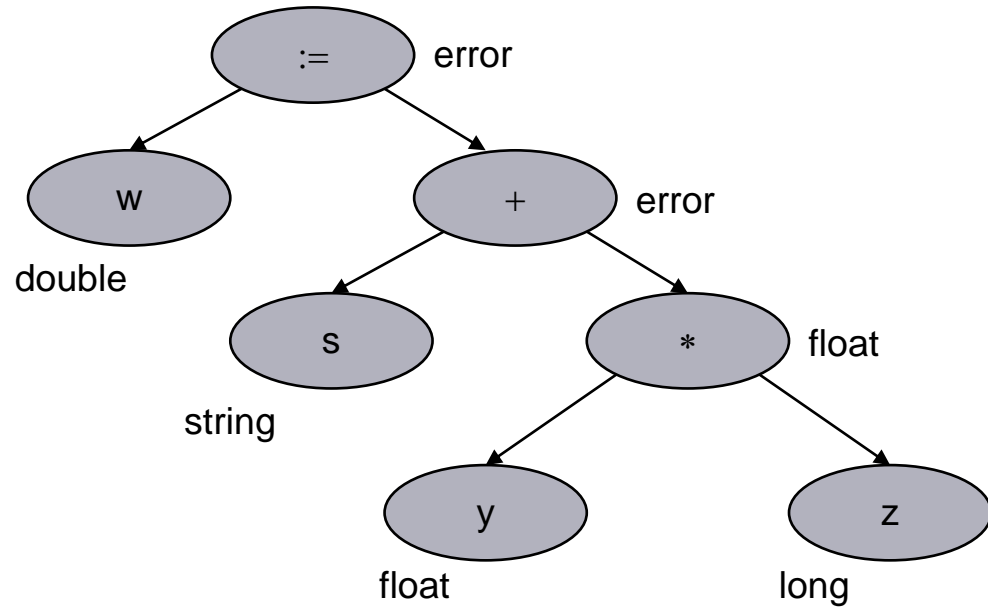
W := S + y * Z

w de tipo double

s de tipo string

y de tipo float

z de tipo long



Lista de Reglas \rightarrow Árbol Sintáctico

W := X + y * Z

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

- c) Árbol Sintáctico con información de tipos e incorporación de conversiones (implícitas)



Tablas de Compatibilidad de Tipos

- ▶ Para poder realizar una operación, ambos operandos deben ser del mismo tipo.

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
- ▶ Si una combinación está permitida, en la celda correspondiente estará el tipo resultante, y ambos operandos deberán ser convertidos al tipo del resultado.

Lista de Reglas → Árbol Sintáctico (b)

Tabla de Tipos y Conversiones

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

- ▶ Toda la semántica de los tipos y **conversiones** 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 Árbol Sintáctico (c)

con información de tipos e incorporación de conversiones

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

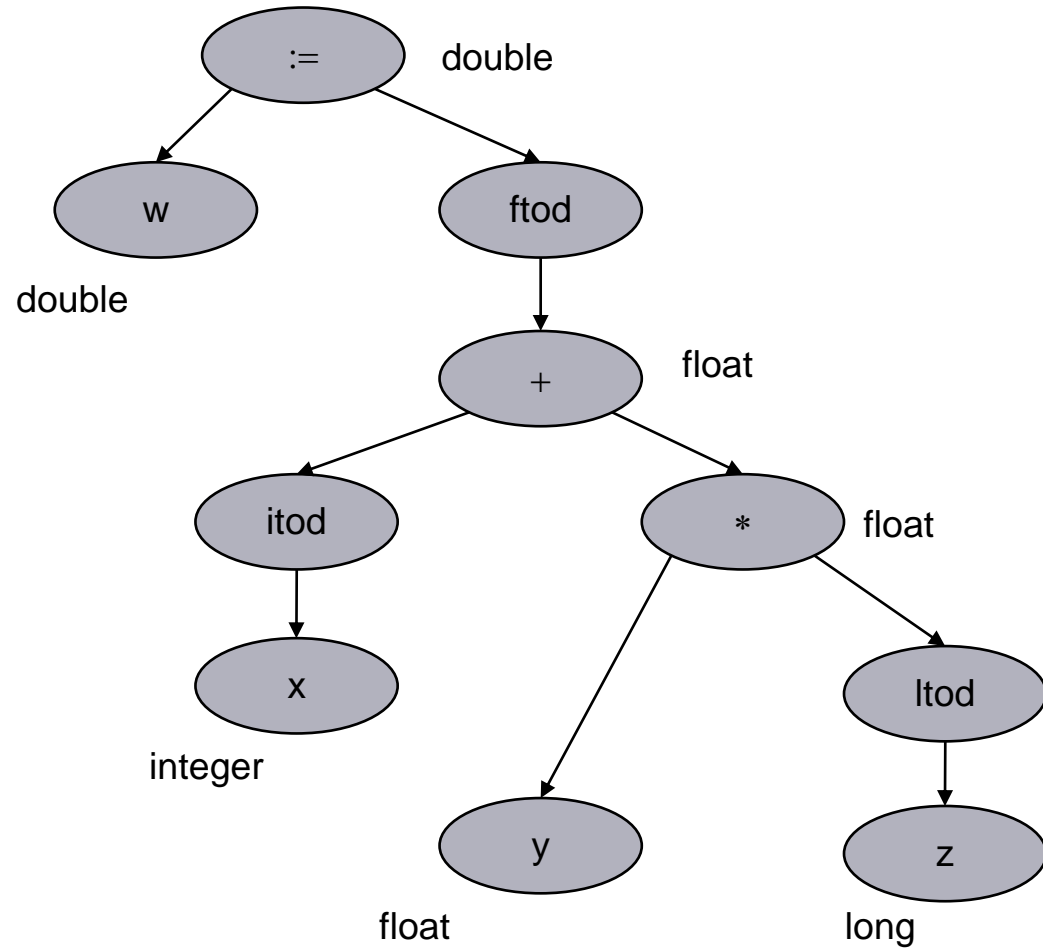
(*) $id.ptr$ = Ref a la entrada del identificador en la Tabla de Símbolos (**) $cte.ptr$ = Ref a la entrada de la constante en la Tabla de Símbolos

Lista de Reglas \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



Tipos y Conversiones

LISTA DE REGLAS → TERCETOS

Lista de Reglas → Tercetos

(Información de tipos y conversiones implícitas)

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)



Tipos y Conversiones

LISTA DE REGLAS → POLACA INVERSA

Lista de Reglas \rightarrow Polaca Inversa

(Información de tipos y conversiones implícitas)

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)



Tipos y Conversiones

ÁRBOL SINTÁCTICO → TERCETOS

ÁRBOL SINTÁCTICO → POLACA INVERSA

Árbol Sintáctico → Tercetos

(Información de tipos y conversiones implícitas)

(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 → Polaca Inversa

(Información de tipos y conversiones implícitas)

(a) (a)

Árbol Sintáctico sin tipos ni conversiones → Polaca sin tipos ni conversiones

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

(b) (b)

Árbol Sintáctico con tipos → Polaca con tipos

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

(c) (c)

Árbol Sintáctico con tipos y conversiones → Polaca con tipos y conversiones



Conversiones Explícitas

ÁRBOL SINTÁCTICO
TERCETOS
POLACA INVERSA

Conversiones explícitas

$W := X + y * Z$

Con:

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

¿Es válida la expresión?

¿Es válida la asignación?

Para un lenguaje que sólo admite conversiones explícitas, esta sentencia dará errores de incompatibilidad de tipos



Conversiones explícitas

$W := X + y * Z$ ← **¡Tipos incompatibles!**

y de tipo float

z de tipo long

- ▶ Para poder realizar una operación, ambos operandos deben tener el mismo tipo en el momento de hacer la operación.

F _*	integer	long	float	double	string
integer	long	long	float	double	X
long	long	long	float	double	X
float	float	float	double	double	X
double	double	double	double	double	X
string	X	X	X	X	X

- ▶ Si una combinación está permitida, en la celda correspondiente estará el tipo resultante, y ambos operandos deberán ser convertidos (explícita o implícitamente) al tipo del resultado.



Conversiones explícitas

¡Tipos incompatibles!

$w := x + y * \text{toFloat}(z)$

x de tipo integer


y * tofloat(z) de tipo float

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



Conversiones explícitas

¡Tipos incompatibles!

 $w := \text{toFloat}(x) + y * \text{toFloat}(z)$

w de tipo double

resultado de la expresión de tipo float



Conversiones explícitas

$w := \text{todouble}(\text{tofloat}(x) + y * \text{tofloat}(z))$

w de tipo double

x de tipo integer

y de tipo float

z de tipo long

¡Ahora sí!



Conversiones explícitas

LR → AS → Assembler

Conversiones explícitas en el código fuente



Conversiones en el Árbol Sintáctico



Conversiones en Assembler

Conversiones explícitas

LR → T → Assembler

Conversiones explícitas en el código fuente



Conversiones en el los Tercetos



Conversiones en Assembler

Conversiones explícitas

LR → PI → Assembler

Conversiones explícitas en el código fuente



Conversiones en la Polaca Inversa



Conversiones en Assembler

Conversiones explícitas

LR → AS → T → Assembler

Conversiones explícitas en el código fuente



Conversiones en el Árbol Sintáctico



Conversiones en los Tercetos



Conversiones en Assembler



Conversiones explícitas

LR → AS → PI → Assembler

Conversiones explícitas en el código fuente



Conversiones en el Árbol Sintáctico



Conversiones en la Polaca Inversa



Conversiones en Assembler



Ejercicio

- ▶ Dada la siguiente gramática de expresiones:

$A \rightarrow id = E$

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow id \mid cte$

Para un lenguaje que admite datos de tipo int, y de tipo float, y conversiones implícitas.

- ▶ Si el camino de generación de código es:

Lista de Reglas \rightarrow Polaca Inversa \rightarrow Assembler

¿Cómo incorpora las conversiones el compilador?