Índice (I)

Marina de la Cruz
Alfonso Ortega

Índice (II)

Objetivo del análisis semántico

- La semántica del lenguaje forma parte de la especificación del mismo. Normalmente **la semántica de un lenguaje se describe de manera informal** (ver enunciado de la práctica)
- El objetivo del análisis semántico es **comprobar si la semántica del programa que se está compilando cumple las especificaciones de la semántica del lenguaje fuente**. Algunas de estas comprobaciones son:
  - comprobación de tipos en sentencias de asignación
  - comprobación de tipos en operaciones aritmético-lógicas
  - comprobación de tipos en las sentencias condicionales
  - comprobación de la declaración de las variables antes de su uso
  - comprobación de unicidad de identificadores
  - comprobación del indexado de vectores
- Para realizar el análisis semántico se utilizan **gramáticas de atributos**, que son gramáticas en las que se asignan atributos o valores semánticos a los nodos del árbol de análisis sintáctico. Hay dos tipos de atributos:
  - sintetizados: se construyen a partir de los atributos de los hijos
  - heredados: se construyen a partir de los atributos del padre y/o de los hermanos

Decisiones para la la construcción de un analizador semántico

- Elaborar una **lista completa de las comprobaciones semánticas** que va a realizar el compilador. Estas comprobaciones se extraen de la especificación del lenguaje. Por ejemplo:
  - si el lenguaje permite mezcla de tipos en expresiones, las comprobaciones semánticas son diferentes que si el lenguaje no permite la mezcla de tipos.
  - si el lenguaje permite la utilización de variables sin previa definición de las mismas, las comprobaciones semánticas son diferentes que si el lenguaje requiere la definición previa de una variable antes de su uso.
- Definir los **atributos necesarios** para implementar las comprobaciones semánticas especificadas en el punto anterior.
- Definir si los atributos son **heredados o sintetizados**.
- Definir el modo en que se van a **calcular los atributos**.
- Si es necesario, **transformar la gramática y ajustar el analizador sintáctico**.

Análisis semántico con Bison (I)

- El analizador semántico se **añade al analizador sintáctico** construido con Bison **incorporando en las acciones de las reglas las comprobaciones propias del análisis semántico** (posteriormente también se incorporará la generación de código en las acciones de las reglas)
- Por ejemplo, para comprobar la restricción semántica que dice que la operación suma solamente es posible entre expresiones de tipo ENTERO, se hace:

```
exp: exp '+' exp {
                    1. comprobar que las dos
                    expresiones de la parte derecha
                    son de tipo ENTERO (consultando
                    el atributo correspondiente)

                    2. asignar a la expresión de la parte
                    izquierda el tipo ENTERO (cálculo de
                    un atributo sintetizado)

                      /* generación de código */
                  }
```

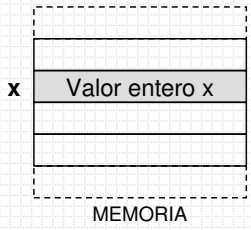

Análisis semántico con Bison (II)

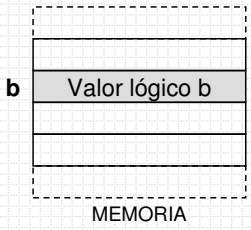
- Las decisiones para la construcción del analizador semántico con Bison son:
  - La lista de las comprobaciones semánticas: se deduce de la especificación del lenguaje.
  - Los **atributos necesarios** se deducen a partir de las comprobaciones semánticas que haya que hacer (ver lista anterior). El tipo de los atributos se define mediante la directiva **%union** dentro del fichero de especificación de Bison.
  - Los atributos son **sintetizados**.
    Esta restricción viene impuesta porque el tipo de analizador sintáctico que genera Bison es ascendente LALR(1).
    Si la gramática de atributos que se diseña para implementar el análisis semántico requiere atributos heredados, hay que diseñar un mecanismo adecuado.
  - El **acceso a los atributos** se realiza a través de $$, $1, $2, …
  - El **cálculo de los atributos** se realiza en las acciones asociadas a las reglas.

Nivel de indirección en ALFA (I)

- En el lenguaje de programación ALFA, el **nivel de indirección de un dato** es el número de accesos a memoria necesarios para obtener el valor del dato.
- Una **constante** (entera o lógica) tiene un **nivel de indirección igual a 0**.
- Una variable de **clase escalar** tiene un **nivel de indirección igual a 1**.
  Por ejemplo, las variables x y b definidas como:

```
ENTERO x;
LOGICO b;
```

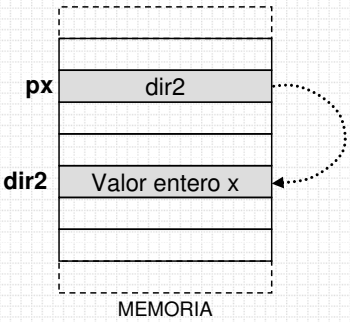


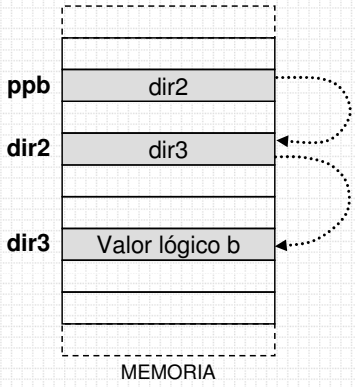


Nivel de indirección en ALFA (II)

- Una variable de **clase puntero** tiene un **nivel de indirección igual a 1 más el número de almohadillas que aparecen en su declaración**.
  Por ejemplo, de las siguientes declaraciones, se deduce que la variable px tiene un nivel de indirección igual a 2, y la variable ppb tiene un nivel de indirección igual a 3:

```
ENTERO # px;
LOGICO ## ppb;
```





Nivel de indirección en ALFA (III)

- Los elementos de las variables de **clase vector** de una o dos dimensiones tienen un nivel de indirección igual a 1 (como las variables de clase escalar)

- El **acceso** (->) a un dato tiene un **nivel de indirección una unidad menos que el nivel de indirección del dato accedido**. Por ejemplo, si se define la variable ppx como:

  ENTERO ## ppx;

  - la expresión de acceso ->ppx tiene un nivel de indirección igual a 2
  - la expresión de acceso ->->ppx tiene un nivel de indirección igual a 1

- La operación de **obtención de la dirección** (<-) de un dato tiene un **nivel de indirección una unidad más que el nivel de indirección del dato**. Por ejemplo, si se define la variable ppx como:

  ENTERO ## ppx;

  - la expresión <-ppx tiene un nivel de indirección igual a 4

Comprobaciones semánticas de la declaración y unicidad de los identificadores (I)

Descripción general

- **Descripción de la semántica:** Las variables debe ser declaradas antes de su uso y además los identificadores deben ser únicos.

- **Comprobaciones semánticas:** se necesitan varias acciones diferentes:
  - Diseñar la inserción en la tabla de símbolos de cada identificador de variable con su correspondiente información asociada (tipo de la variable, clase, nivel de indirección, etc)
  - Comprobar la unicidad de identificadores (haciendo uso de la tabla de símbolos)
  - Comprobar la declaración de variables antes de su uso (haciendo uso de la tabla de símbolos)

Comprobaciones semánticas de la declaración y unicidad de los identificadores (II)

Inserción en la tabla de símbolos y unicidad

- Las producciones de la gramática correspondientes a una lista de identificadores de variables en ALFA son:
  - <identificadores> : :=<identificador>
  - <identificadores> ::=<identificador> , <identificadores>
  - <identificador> ::= TOK_IDENTIFICADOR

- **Cada vez que se declara una variable hay que guardarla en la tabla de símbolos.** Por lo tanto, el punto adecuado para insertar los identificadores de las variables en la tabla de símbolos es la acción semántica de a la última producción de las tres anteriores. En ese momento es necesario disponer de la información asociada a la variable:
  - lexema
  - clase
  - tipo
  - nivel de indirección
  - dimensión (1 ó 2) en el caso de vectores
  - número de elementos en el caso de vectores de una dimensión
  - número de filas y columnas en el caso de vectores de dos dimensiones

- **En el momento de la inserción en la tabla de símbolos se puede comprobar la unicidad** de los identificadores.

Comprobaciones semánticas de la declaración y unicidad de los identificadores (III)

Comprobación de declaración previa al uso

- Cada vez que se accede a un identificador hay que comprobar si se ha declarado previamente haciendo uso de la tabla de símbolos. Las producciones implicadas en éstas comprobaciones son:
  - <sentencia> ::= LLAMAR TOK_IDENTIFICADOR
  - <asignacion> ::= TOK_IDENTIFICADOR = <exp>
  - <asignacion> ::= TOK_IDENTIFICADOR = RESERVAR
  - <asignacion> ::= TOK_IDENTIFICADOR <- TOK_IDENTIFICADOR
  - <elemento_vector> ::= TOK_IDENTIFICADOR [<exp>]
  - <elemento_vector> ::= TOK_IDENTIFICADOR [<exp> ,<exp>]
  - <bucle> ::= DESDE TOK_IDENTIFICADOR = <exp> HASTA <exp> HACER <sentencias> FIN
  - <lectura_escritura> ::= LEER TOK_IDENTIFICADOR
  - <liberacion> ::= LIBERAR TOK_IDENTIFICADOR
  - <acceso> ::= -> TOK_IDENTIFICADOR
  - <exp> ::= TOK_IDENTIFICADOR

Comprobaciones semánticas en las expresiones aritméticas (I)

Comprobación de tipos

- **Descripción de la semántica:** Las expresiones aritméticas sólo pueden operar con datos de tipo ENTERO.
- **Comprobaciones semánticas:** hay que comprobar que los operandos son de tipo ENTERO en las siguientes producciones:
  - <exp> ::= <exp> + <exp>
  - <exp> ::= <exp> - <exp>
  - <exp> ::= <exp> / <exp>
  - <exp> ::= <exp> * <exp>
  - <exp> ::= - <exp>

Comprobaciones semánticas en las expresiones aritméticas (II)

Comprobación de niveles de indirección

- **Descripción de la semántica:** No está permitida la aritmética de punteros.
- **Comprobaciones semánticas:** hay que comprobar que los operandos tienen un nivel de indirección igual a 0 ó 1 las siguientes producciones:
  - <exp> ::= <exp> + <exp>
  - <exp> ::= <exp> - <exp>
  - <exp> ::= <exp> / <exp>
  - <exp> ::= <exp> * <exp>
  - <exp> ::= - <exp>

Comprobaciones semánticas en las expresiones lógicas (I)

Comprobación de tipos (I)

- **Descripción de la semántica:** Las expresiones lógicas sólo pueden operar con datos de tipo LOGICO.
- **Comprobaciones semánticas:** hay que comprobar que los operandos son de tipo LOGICO en las siguientes producciones:
  - <exp> ::= <exp> Y <exp>
  - <exp> ::= <exp> O <exp>
  - <exp> ::= NO <exp>

Comprobaciones semánticas en las expresiones lógicas (II)

Comprobación de niveles de indirección

- **Descripción de la semántica:** Las expresiones lógicas no están permitidas sobre datos de clase puntero (esta descripción no aparece de forma explícita en el enunciado de la práctica)
- **Comprobaciones semánticas:** hay que comprobar que los operandos tienen un nivel de indirección igual a 0 ó 1 las siguientes producciones:
  - <exp> ::= <exp> Y <exp>
  - <exp> ::= <exp> O <exp>
  - <exp> ::= NO <exp>

Comprobaciones semánticas en las expresiones de comparación (I)

Comprobación de tipos

- **Descripción de la semántica:** Las comparaciones sólo pueden operar con datos de tipo ENTERO.
- **Comprobaciones semánticas:** hay que comprobar que los elementos comparados son de tipo ENTERO en las siguientes producciones:
  - <comparacion> ::= <exp> == <exp>
  - <comparacion> ::= <exp> != <exp>
  - <comparacion> ::= <exp> <= <exp>
  - <comparacion> ::= <exp> >= <exp>
  - <comparacion> ::= <exp> < <exp>
  - <comparacion> ::= <exp> > <exp>

Comprobaciones semánticas en las expresiones de comparación (II)

Comprobación de niveles de indirección

- **Descripción de la semántica:** No está permitida la comparación de punteros.
- **Comprobaciones semánticas:** hay que comprobar que los elementos comparados tienen un nivel de indirección igual a 0 ó 1 las siguientes producciones:
  - <comparacion> ::= <exp> == <exp>
  - <comparacion> ::= <exp> != <exp>
  - <comparacion> ::= <exp> <= <exp>
  - <comparacion> ::= <exp> >= <exp>
  - <comparacion> ::= <exp> < <exp>
  - <comparacion> ::= <exp> > <exp>

Comprobaciones semánticas en sentencias de asignación (I)

Comprobación de tipos

- **Descripción de la semántica:** Para que una sentencia de asignación sea válida, los tipos de dato de la parte izquierda y derecha deben ser iguales.

- **Comprobaciones semánticas:** hay que comprobar que son iguales los tipos de dato de la parte izquierda y derecha de la asignación en las siguientes producciones:

  - <asignacion> ::= <identificador> = <exp>
  - <asignacion> ::= <elemento_vector> = <exp>
  - <asignacion> ::= <acceso> = <exp>
  - <asignacion> ::= <identificador> = <- <identificador>

  - En la producción <asignacion> ::= <identificador> = RESERVAR
    no procede la comprobación de tipos ya que la parte izquierda puede ser de tipo ENTERO o de tipo LOGICO. La restricción semántica en este caso está relacionada con la clase (la parte izquierda tiene que ser de clase puntero)

Comprobaciones semánticas en sentencias de asignación (II)

Comprobación de niveles de indirección

- **Descripción de la semántica:** Para que una sentencia de asignación sea válida, las partes izquierda y derecha de la asignación deben tener el mismo nivel de indirección, o bien, la parte derecha un nivel de indirección igual a 0 y la parte izquierda igual a 1.

- **Comprobaciones semánticas:** hay que comprobar que ambas partes de la asignación tienen el mismo nivel de indirección, o bien, la parte derecha un nivel de indirección igual a 0 y la parte izquierda igual a 1, en las siguientes producciones:

  - <asignacion> ::= <identificador> = <exp>
  - <asignacion> ::= <elemento_vector> = <exp>
    En esta producción, la parte izquierda de la asignación siempre tiene un nivel de indirección igual a 1, y por lo tanto, la parte derecha sólo puede tener un nivel de indirección igual a 0 ó 1.
  - <asignacion> ::= <acceso> = <exp>
  - <asignacion> ::= <identificador> == <- <identificador>

  - En la producción <asignacion> ::= <identificador> = RESERVAR
    La comprobación semántica consiste en verificar que la parte izquierda tiene un nivel de indirección superior a 1, es decir, que es de clase puntero.

Comprobaciones semánticas relativas a punteros (I)

Reserva y liberación de memoria

- **Descripción de la semántica:** La palabras reservadas del lenguaje RESERVAR y LIBERAR se utilizan respectivamente para asignar a un puntero memoria para almacenar un nuevo valor y para liberarla.

- **Comprobaciones semánticas:**
  - Las comprobaciones semánticas relacionadas con la reserva de memoria se especifican en la descripción de las comprobaciones semánticas en sentencias de asignación ya que, la reserva de memoria sólo puede aparecer en una sentencia de asignación.
  - En la liberación de memoria se comprueba que el identificador implicado se declaró previamente como una variable de clase puntero. La producción de liberación de memoria es la siguiente:

  - <liberacion> ::= LIBERAR <identificador>

Comprobaciones semánticas relativas a punteros (II)

Acceso al contenido apuntado

- **Descripción de la semántica:** El concepto de “acceso al contenido apuntado” requiere la existencia de un “objeto apuntador”, es decir, de una variable de tipo puntero.

- **Comprobaciones semánticas:** las comprobaciones semánticas relacionadas con el acceso al contenido apuntado se realizan en las siguientes producciones:

  - <acceso> ::= -> <identificador>

    Se comprueba que el identificador ha sido declarado como una variable de clase puntero.

  - <acceso> ::= -> <acceso>

    Se comprueba que el nivel de indirección es correcto.

Comprobaciones semánticas para la clase VECTOR (I)

Comprobación del tamaño

- **Descripción de la semántica:** El tamaño de los vectores de una dimensión no podrá exceder nunca el valor de 64. Esta misma restricción se aplica a cada una de las dimensiones de los vectores de dos dimensiones.

- **Comprobaciones semánticas:** hay que comprobar que el tamaño declarado para un vector no excede el límite permitido (64 para cada dimensión). También hay que comprobar que el tamaño declarado para cada dimensión es mayor que 0 (esta restricción no se especifica explícitamente en la descripción de la semántica del lenguaje). Las producciones donde se pueden realizar estas comprobaciones son:

  - <clase_vector> ::= VECTOR <tipo> DE <constante_entera>
  - <clase_vector> ::= VECTOR <tipo> DE <constante_entera> POR <constante_entera>

Comprobaciones semánticas para la clase VECTOR (II)

Comprobaciones en el indexado de vectores de una dimensión (I)

- **Descripción de la semántica:** Para indexar los elementos de un vector de una dimensión se utiliza la cadena `[<exp>]` a continuación del nombre del vector. Los corchetes deberán contener en su interior una expresión de tipo ENTERO. En ningún caso podrá aparecer en el lugar de la expresión que encierran los corchetes una variable de tipo puntero, aunque sí el acceso al contenido de dicha variable, siempre que dicho contenido sea de tipo ENTERO.

- **Comprobaciones semánticas:** la producción indicada para comprobar la semántica descrita es la siguiente:

  - <elemento_vector> ::= <identificador> [<exp>]

  Las comprobaciones que hay que realizar son:

  - El identificador que aparece corresponde a la declaración de un vector de una dimensión.
  - La expresión que actúa como índice es de tipo ENTERO.
  - La expresión que actúa como índice tiene un nivel de indirección igual a 0 ó 1.

Comprobaciones semánticas para la clase VECTOR (III)

Comprobaciones en el indexado de vectores de una dimensión (II)

- **Descripción de la semántica:** Los corchetes deberán contener en su interior una expresión de tipo ENTERO, con un valor entre 0 y el tamaño definido para el vector menos 1 (ambos incluidos)

- **Comprobaciones semánticas:** en una operación de indexado de un vector de una dimensión, la comprobación de que el valor del índice está dentro del rango permitido sólo se puede realizar en **tiempo de ejecución**, no en tiempo de compilación. Para implementar esta restricción semántica, en tiempo de compilación se genera el código ensamblador adecuado para realizar la validación del valor del índice. La producción indicada para la generación del código es:

  - <elemento_vector> ::= <identificador> [<exp>]

Comprobaciones semánticas para la clase VECTOR (IV)

Comprobaciones para vectores de dos dimensiones

- **Descripción de la semántica:** Las restricciones semánticas para vectores de dos dimensiones son similares a las correspondientes a los vectores de una dimensión.

- **Comprobaciones semánticas:** la producción indicada para comprobar la semántica es la siguiente:

  - <elemento_vector> ::= <identificador> [<exp> , <exp>]

  Las comprobaciones que hay que realizar son:

  - El identificador que aparece corresponde a la declaración de un vector de dos dimensiones.
  - Las expresiones que actúan como índices son de tipo ENTERO.
  - Las expresiones que actúan como índices tienen un nivel de indirección igual a 0 ó 1.
  - En tiempo de compilación se genera el código ensamblador adecuado para comprobar que los valores de los índices no exceden los rangos permitidos.

Comprobaciones semánticas para las condiciones (I)

- **Descripción de la semántica:** Los resultados de las condiciones de las sentencias SI-ENTONCES, SI-ENTONCES-SINO y MIENTRAS-HACER deben ser de tipo LOGICO.

- **Comprobaciones semánticas:** hay que comprobar que las condiciones de las sentencias de bucle y condicionales son de tipo LOGICO y con un nivel de indirección 0 ó 1. Las producciones implicadas en la comprobación de esta restricción semántica son las siguientes:

  - <condicional> ::= SI <exp> ENTONCES <sentencias> FIN
  - <condicional> ::= SI <exp> ENTONCES <sentencias> SINO <sentencias> FIN
  - <bucle> ::= MIENTRAS <exp> HACER <sentencias> FIN

- **Modificación de la gramática:** para poder realizar comprobaciones sobre las condiciones de bucles y condicionales representadas por el no terminal <exp> es necesario modificar la gramática para ubicar la acción semántica al final de la regla. Para la sentencia SI-ENTONCES, el punto para la comprobación sería uno de los dos siguientes:
  - <condicional> ::= SI <exp> **{acción}** ENTONCES <sentencias> FIN
  - <condicional> ::= SI <exp> ENTONCES **{acción}** <sentencias> FIN

Comprobaciones semánticas para las condiciones (II)

Por ejemplo, si se elige el segundo punto:

- <condicional> ::= SI <exp> ENTONCES **{acción}** <sentencias> FIN

Hay que modificar la gramática de la siguiente manera:

- Se añade un nuevo símbolo no terminal <si_exp_entonces> para situar la acción semántica al final de una regla:

  <si_exp_entonces> ::= SI <exp> ENTONCES **{acción}**

- La producción original (36) quedaría de la siguiente manera:

  (36) <condicional> ::= <si_exp_entonces> <sentencias> FIN

- Se aplica el mismo razonamiento para la sentencias SI-ENTONCES-SINO y MIENTRAS-HACER.

Comprobaciones para sentencias de entrada/salida

- **Descripción de la semántica:** Las operaciones de entrada/salida se efectúan sobre elementos de tipo escalar.

- **Comprobaciones semánticas:** en cada una de las producciones de entrada/salida hay que realizar distintas comprobaciones. Estas producciones son las siguientes:

  - <lectura_escritura> ::= LEER <identificador>

    En esta producción se comprueba que el identificador corresponde a una declaración de una variable de clase escalar.

  - <lectura_escritura> ::= LEER <elemento_vector>

    En esta producción las comprobaciones semánticas son las mismas que se realizan el las operaciones de indexado de vectores (ver apartado “Comprobaciones semánticas para la clase VECTOR”)

  - <lectura_escritura> ::= ESCRIBIR <exp>

    En esta producción se comprueba que el nivel de indirección de la expresión es 0 ó 1.

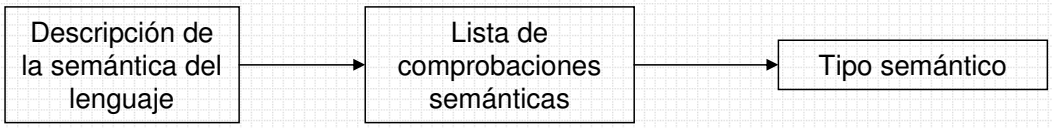
Otras comprobaciones semánticas

- **En la descripción de la semántica de ALFA proporcionada en el enunciado de la práctica sólo se mencionan los aspectos más relevantes. Además de éstos, hay un conjunto de restricciones semánticas que también deben estar presentes en el compilador de ALFA, como por ejemplo:**

  - Las variables deben ser definidas antes de ser utilizadas
  - Las variables deben ser únicas dentro de su ámbito de aplicación
  - etc

Definición del tipo semántico (I)

- El tipo semántico es el **tipo de los atributos de los símbolos en el árbol de análisis sintáctico.**
  Tiene que contemplar todos los casos posibles de valores semánticos de cualquier símbolo del lenguaje (terminal o no terminal).
- Tiene que permitir la realización de las comprobaciones semánticas y la generación de código.



- En la implementación del compilador de ALFA se pueden utilizar los siguientes atributos:
  - Lexema de las variables
  - Valor numérico de las constantes numéricas
  - Valor lógico de las constantes lógicas
  - Tipo de las expresiones
  - Nivel de indirección de las expresiones

NOTA: La elección de los atributos no es única, existen distintas alternativas
Algunos atributos son necesarios para la generación de código, no para análisis semántico

Definición del tipo semántico (II)

- El valor semántico de los terminales lo proporciona el analizador léxico:
  - los identificadores tiene como valor semántico su lexema
  - las constantes numéricas tienen como valor semántico su valor numérico
  - las constantes lógicas tienen como valor semántico 0 (FALSO) o 1 (VERDADERO)

- El valor semántico de los no terminales se calcula en el proceso de análisis sintáctico. Cada vez que se reduce una producción, se puede calcular el valor semántico del símbolo no terminal de la parte izquierda de la producción. Este valor semántico se almacena en la pseudovariable $$.

Definición del tipo semántico (III)

- El tipo semántico se define con la declaración **%union** en el fichero de especificación de Bison.

- La declaración %union de Bison se transforma en una definición de tipo union de C. Este tipo de datos de C sólo permite el uso de uno de sus campos, y no es válido para símbolos con un valor semántico múltiple. Por ejemplo, el no terminal “acceso” requiere un atributo para el tipo y otro para el nivel de indirección. Para permitir la **multiplicidad de atributos**, se puede definir en la declaración %union un sólo campo (atributos) cuyo tipo (tipo_atributos) sea un tipo definido como struct con tantos campos como sea necesario para contemplar todos los casos posibles de valores semánticos de cualquier símbolo del lenguaje (terminal o no terminal).

  NOTA: El tipo “tipo_atributos” se puede definir en un fichero de cabecera aparte.

Definición del tipo semántico (IV)

- En la declaración **%union** del fichero **alfa.y**, se define un sólo campo, **atributos**, cuyo tipo sea **tipo_atributos** (definido en el fichero alfa.h).

- Se cualifican con **<atributos>** las declaraciones **%token** de los símbolos terminales que tienen valor semántico.

- Se añaden las correspondientes declaraciones **%type**. **Todos los no terminales tienen valor semántico para permitir la propagación de atributos en el árbol de análisis.**

alfa.y

```
%{
#include “alfa.h”
%}
%union
{
  tipo_atributos atributos;
}

%token <atributos> TOK_IDENTIFICADOR
%token <atributos> TOK_NUMERO
%token <atributos> TOK_FALSO
%token <atributos> TOK_VERDADERO

/* resto de los tokens sin valor */
/* semántico                     */

%type <atributos> programa
%type <atributos> declaraciones
%type <atributos> sentencias

/* resto de los no terminales */

...

%%
...
%%
...
```

Definición del tipo semántico (V)

- En el fichero **alfa.h** se define el tipo **tipo_atributos** como un struct con los siguientes campos :
  - **lexema_identificador**: para identificadores.
  - **valor_numerico**: para constantes enteras.
  - **valor_logico**: para constantes lógicas.
  - **tipo**: para comprobación de tipos básicos.
  - **nivel_indireccion**: para comprobación del nivel de indirección.

alfa.h

```
#ifndef _ALFA_H
#define _ALFA_H

#define MAXID 50

/* otros defines */


typedef struct
{
   char lexema_identificador[MAXID+1];
   int valor_numerico;
   int valor_logico;
   int tipo;
   int nivel_indireccion;
}tipo_atributos;

#endif
```


Definición del tipo semántico (VI)

- El fichero **alfa.l** se modifica para que el analizador léxico actualice el valor semántico de los terminales.

alfa.l

```
...

FALSO     {columna+=yyleng; yylval.atributos.valor_logico =0; return TOK_FALSO;}
VERDADERO {columna+=yyleng; yylval.atributos.valor_logico =1; return TOK_VERDADERO;}

...

{IDENTIFICADOR}     {   columna+=yyleng;

                        /* control de la longitud del identificador  *^/

                        strcpy(yylval.atributos.lexema_identificador ,yytext);
                        return TOK_IDENTIFICADOR;
                    }

{NUMERO}   { columna+=yyleng;
             yylval.atributos.valor_numerico = atoi(yytext);
             return TOK_NUMERO;
           }
...
```

Ejemplo de uso del tipo semántico para la comprobación de tipos

- Supongamos la declaración `ENTERO X;`
- Supongamos la declaración `LOGICO B;`
- Supongamos la expresión `X+B`

1. El analizador léxico reconoce el identificador X y se lo comunica al analizador sintáctico devolviendo TOK_IDENTIFICADOR y cargando su valor semántico en el campo correspondiente de la variable yylval.
   El analizador sintáctico apila en su pila de análisis el TOK_IDENTIFICADOR y su valor semántico.
2. El analizador sintáctico reduce la producción exp::=TOK_IDENTIFICADOR. En la acción semántica asociada a la producción, busca en la tabla de símbolos y obtiene el tipo y el resto de atributos del identificador X para deducir el tipo y el resto de atributos del símbolo de la parte izquierda (como copia de los mismos atributos del símbolo de la parte derecha)
3. Cuando se reduce la producción exp::=exp+exp, se dispone de los atributos necesarios (`$1 y $3`) para realizar la comprobación de tipos (en este caso la comprobación termina con error)