# PROCESADORES DE LENGUAJES Primer Examen (Ev. Alt.)



**2) Análisis Semántico. El lenguaje D (2 puntos)**

D es un lenguaje de programación secuencial con tipado dinámico de sus variables. El programa D está compuesto por una sección de declaración de variables sin tipo explícito (ver programa de ejemplo abajo) y una secuencia de asignaciones simples. Las expresiones en D son de tres tipos: (a) enteras, (b) booleanas y (c) listas. Los elementos de una lista pueden ser enteros, booleanos u otras listas. Las expresiones enteras y booleanas son las convencionales (ver gramática de D abajo). Las expresiones listas incluyen (a) listas por extensión y (b) yuxtaposiciones (o suma) de listas (ver gramática de D abajo).

Una asignación está bien tipada si y sólo si el tipo de la expresión asignada es entera, booleana o lista.
Una asignación es errónea si y sólo si la expresión contiene alguna variable no declarada.
Una asignación está indefinida si y sólo si no está bien tipada y no es errónea.

En las asignaciones bien tipadas, el tipo de la variable asignada cambia al tipo de la expresión asignada.
En las asignaciones erróneas, el tipo de la variable asignada no cambia (permanece el tipo que tenía previamente).
En las asignaciones indefinidas, el tipo de la variable asignada es indefinido.

A continuación, se muestra un programa D de ejemplo (con anotaciones sobre el tipado de las asignaciones):

```
VARIABLES l1,l2,l3,i,j,b;

ASIGNACIONES
   l1 = [cierto, falso O b];  // indefinida
   l1 = [cierto];             // bien tipada
   i = 1 * z;                 // errónea
   j = [i + 1,3];             // indefinida
   l3 = [2,[1,8]];            // bien tipada
   l2 = l3;                   // bien tipada
   l3 = []+1;                 // indefinida
```

Suponga la siguiente gramática para el lenguaje D:

```
programa : variables asignaciones EOF ;

variables : VARIABLES (decl_vars)*  PUNTOYCOMA;

decl_vars : IDENT
   | IDENT COMA decl_vars
   ;

asignaciones : ASIGNACIONES (asignacion)* ;

asignacion: IDENT ASIGNACION expresion PUNTOYCOMA ;
```

```
expresion: expresion (MENOS|POR|DIV) expresion
    | expresion MAS expresion
    | expresion (Y|O) expresion
    | NO expresion
    | MENOS expresion
    | lista_por_extension
    | PARENTESISABIERTO expresion PARENTESISCERRADO
    | IDENT
    | NUMERO
    | CIERTO
    | FALSO
    ;

lista_por_extension: lista_vacia
   | lista_no_vacia
   ;

lista_vacia: CORCHETEABIERTO  CORCHETECERRADO  ;

lista_no_vacia: CORCHETEABIERTO expresiones CORCHETECERRADO  ;

expresiones: expresion COMA expresiones
   | expresion
   ;
```

**SE PIDE:**

Una solución para un analizador semántico que decida si cada asignación del programa está bien tipada, es indefinida o errónea. Por ejemplo, la ejecución del analizador semántico sobre el programa de ejemplo emitirá los siguientes mensajes por pantalla:

```
l1 = [cierto, falso O b];   indefinida
l1 = [cierto];              bien tipada (lista)
i = 1 * z;                  errónea
j = [i + 1,3];              indefinida
l3 = [2,[1,8]];             bien tipada (lista)
l2 = l3;                    bien tipada (lista)
l3 = []+1;                  indefinida
```

```
OBJETIVO:
    Construir un analizador semántico capaz de decidir si cada asignación de un
    programa D está bien tipada, mal tipada o es errónea.

    Una asignación está bien tipada si y sólo si el tipo de la expresión asignada
    es entera, booleana o lista.
    Una asignación es errónea si y sólo si contiene alguna variable no declarada.
    Una asignación está mal tipada si y sólo si no está bien tipada y no es errónea.

DECISIONES:

    Decisión 1) Memoria para almacenar el tipo de cada variable declarada.
                Las variable declaradas tienen tipo inicial indefinido.

    Decisión 2) Calcular recursivamente el tipo de una expresión.

    Decisión 3) En cada asignación:
       si la expresión es errónea entonces
          el tipo de la variable asignada no cambia.
       si la expresión es entera/booleana/lista entonces
          el tipo de la variable asignada es entera/booleana/lista igualmente.
       en cualquier otro caso
          la variable asignada tiene tipo indefinido.

    Argumento de diseño I:

       { Decision 1, Decision 2,  Decisión 3 } es un conjunto de decisiones
       necesarias. Si falta alguna de ellas, no se puede conseguir el objetivo.
       También es un conjunto suficiente. No hace falta ninguna
       decisión adicional para conseguir el objetivo.


GRAMATICA ATRIBUIDA:

(global) memoria_variables  //Decisión 1

programa : variables asignaciones

variables : VARIABLES (decl_vars)*  PUNTOYCOMA

decl_vars : IDENT {almacenar en memoria_variables IDENT con tipo indefinido}
//Decisión 1
   | IDENT {almacenar en memoria_variables IDENT con tipo indefinido} COMA decl_vars
//Decisión 1

asignaciones : ASIGNACIONES (asignacion)*

asignacion: IDENT ASIGNACION tipo=expresion PUNTOYCOMA //Decisión 3
    {si tipo es erróneo entonces no hacer nada
     si tipo es entero, booleano o lista entonces
       almacenar en memoria_variables IDENT con dicho tipo
     en otro caso
       almacenar en memoria_variables IDENT con tipo indefinido}

expresion dev tipo: tipo1=expresion (MENOS|POR|DIV) tipo2=expresion  //Decisión 2
      {si tipo1 o tipo2 es igual a error entonces tipo = error
       si tipo1 y tipo2 son guales a enteros entonces tipo = entero
       en otro caso tipo = indefinido}
```

```
    | tipo1=expresion MAS tipo2=expresion
      {si tipo1 o tipo2 es igual a error entonces tipo = error
       si tipo1 y tipo2 son iguales a enteros entonces tipo = entero
       si tipo1 y tipo2 son iguales a lista entonces tipo = lista
       en otro caso tipo = indefinido}
    | tipo1=expresion (Y|O) tipo2=expresion
      {si tipo1 o tipo2 es igual a error entonces tipo = error
       si tipo1 y tipo2 son iguales a booleanos entonces tipo = booleano
       en otro caso tipo = indefinido}
    | NO tipo1=expresion
      {si tipo1 es igual a error entonces tipo = error
       si tipo1 es igual a booleano entonces tipo = booleano
       en otro caso tipo = indefinido}
    | MENOS tipo1=expresion
    {si tipo1 es igual a error entonces tipo = error
     si tipo1 es igual a booleano entonces tipo = booleano
     en otro caso tipo = indefinido}
    | tipo=lista_por_extension
    | PARENTESISABIERTO tipo=expresion PARENTESISCERRADO
    | IDENT {si IDENT está en memoria_variables entonces
               tipo = consultar el tipo de IDENT en memoria_variables
             en otro caso tipo = error}
    | NUMERO {tipo = entero}
    | CIERTO {tipo = booleano}
    | FALSO  {tipo = booleano}

lista_por_extension dev tipo: tipo=lista_vacia
   | tipo=lista_no_vacia

lista_vacia dev tipo: CORCHETEABIERTO  CORCHETECERRADO  {tipo = lista}

lista_no_vacia dev tipo: CORCHETEABIERTO tipos=expresiones CORCHETECERRADO
   { si algún elemento en tipos es igual a error entonces tipo=error
     si algún elemento en tipos es igual a indefinido entonces tipo=indefinido
     en otro caso tipo=lista }

expresiones dev tipos: tipo=expresion {añadir tipo a tipos} COMA aux=expresiones
                       {añadir cada elemento de aux a tipos}
   | tipo=expresion {añadir tipo a tipos}


Argumento de diseño II: la gramática propuesta implica Decisión 1, Decisión 2 y
Decisión 3. Por tanto, la gramática propuesta, a través de estas decisiones (ver
Argumento de diseño I), satisface el objetivo.
```

**3) ANTLR (1 punto)**

Programar como método Visitor en ANTLR4/Java la siguiente regla de una gramática atribuida:

```
lista_por_extension dev tipo: tipo=lista_vacia
      | tipo=lista_no_vacia
```

```
@Override
public String visitLista_por_extension(Anasint2.Lista_por_extensionContext ctx) {
    String tipo=null;
    if (ctx.lista_vacia()!=null)
        tipo=visitLista_vacia(ctx.lista_vacia());
    else
        tipo=visitLista_no_vacia(ctx.lista_no_vacia());
    return tipo;
}
```