Práctica 3

Análisis LALR(1) para miniPascal. Construcción del árbol de sintaxis

Objetivos

1) Trabajar el análisis sintáctico LALR
2) Introducirse al uso de atributos
3) Trabajar con la herramienta Bison

Contenidos

El objetivo final de esta práctica es implementar, utilizando Bison, un analizador sintáctico para miniPascal que además genere el árbol de sintaxis (concreta) del fuente analizado.

En el directorio de salidas de la asignatura podeis encontrar el fichero "esqueletoGA_miniPascal.y" con el fuente Bison de la gramática de miniPascal. Lo que hay que hacer es insertar en dicho fuente las acciones necesarias para para la construcción de árbol de sintaxis, que deberá quedar almacenado en la variable "elAST", y posteriormente generar la descripción XML del árbol.

El fichero resultado, si es correcto, se puede ver con cualquier visualizador de XML (hay uno en el directorio "xmlviewerJava" de "salidas". Para su invocación, es suficiente con ejecutar, desde dicho directorio, "xmlviewer.sh"). Veamos un ejemplo. Al ejecutar la invocación

```
GA_miniPascal 3 simple.pas simple.xml
```

siendo el contenido de "simple.pas"

```
(*un ejemplo*)
program uno;
   var x: integer;
begin
   x := 2 DIV 3
end.
```


el contenido de "simple.xml" generado es el que aparece en el Anexo-II, y su visualización con un visualizador XML será la mostrada en la figura 1.

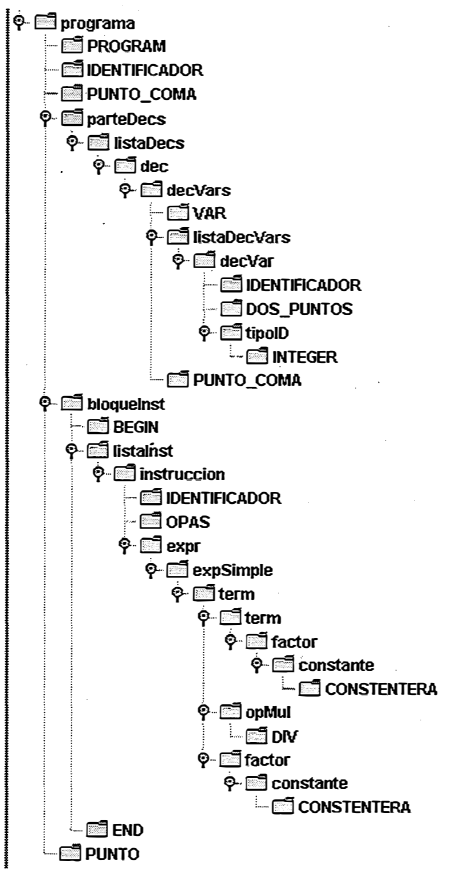

Figura 1. Un ejemplo de árbol de sintaxis. Nótese que cada nodo se denomina con la parte izquierda de la producción que lo genera.

Resultados

Como resultado de esta práctica hay que entregar el fichero denominado 'pract3.tar', de manera que la ejecución de la instrucción 'tar -xvf pract3.tar' genere un directorio denominado 'pract3' con los siguientes ficheros:

- 'AL_miniPascal3.l' que contiene el fuente Flex del analizador léxico usado. Nótese que este analizador léxico requiere algunas modificaciones respecto a los usados en las prácticas anteriores: ahora debemos asegurar, para cada terminal, que el árbol correspondiente ha sido generado y asignado al atributo en "yylval.atrib", con la información necesaria para la generación del árbol XML (la información es un string con el nombre del "tag" XML adecuado).
- 'GA_miniPascal.y' que contiene el fuente Bison con la implementación del analizador sintáctico y el generador del árbol de sintaxis en formato XML. En el directorio de salidas se encuentra el fichero "esqueletoGA_miniPascal.y" con la gramática del lenguaje que estamos manejando, que debeis usar como base para obtener 'GA_miniPascal.y'. También podeis descargarlo desde la página Web de la asignatura.
- 'GA_miniPascalMake' que contiene el fichero Make para generar el ejecutable del analizador sintáctico, de manera que la invocación

```
make -f GA_miniPascalMake
```

genere el ejecutable 'GA_miniPascal' que realiza la tarea especificada. La invocación al analizador debe ser como sigue (el significado del número 3 en la invocación es el mismo que en la práctica anterior):

```
GA_miniPascal 3 fichFuente fichDestino
```

- Todos aquellos ficheros que sean necesarios (conteniendo, por ejemplo, implementaciones de tipos de datos, funciones para el tratamiento de errores, etc.).

El analizador léxico no debe dar ningún mensaje (ni realizar ninguna tarea adicional) cuando el fuente sea léxica y sintácticamente correcto. En caso de error léxico, el comportamiento ha de ser el establecido en la segunda práctica.

En caso de error sintáctico, el programa aborta.

Nota: En el directorio de salidas podeis encontrar el subdirectorio "bancoPruebasMiniPascalXML", con los fuentes XML de los árboles correspondientes a los ejemplos de la batería de test.

Fecha límite de entrega de resultados:

Todos los grupos: 16 de Enero de 2004.

Anexo-I

El esquema general de un documento XML es el que sigue

```
<?xml version="1.0"?>
<Nodo atrib1="val1" atrib2="val2">
  <NodoHoja1>Valor</NodoHoja1>
  <NodoHoja2/>
  <Nodo>...</Nodo>
    ...
</Nodo>
```

Restricciones de uso de XML

- Todo nodo que se abre se debe cerrar.
- No hay límite de profundidad en la jerarquía de nodos.
- El nombre de un nodo debe ser un identificador válido (similar a Pascal) como <MiNodo> o <Nodo_de_prueba>. No serían válidos <;> o <[indice]>
- Un nodo puede tener cualquier número de pares atributo-valor (en esta práctica no es necesario usar atributos en el fuente XML).

- **Anexo-II**

```
<?xml version="1.0"?>
<programa>
  <PROGRAM/>
  <IDENTIFICADOR/>
  <PUNTO_COMA/>
  <parteDecs>
    <listaDecs>
      <dec>
        <decVars>
          <VAR/>
          <listaDecVars>
            <decVar>
              <IDENTIFICADOR/>
              <DOS_PUNTOS/>
              <tipoID>
                <INTEGER/>
              </tipoID>
            </decVar>
          </listaDecVars>
          <PUNTO_COMA/>
        </decVars>
      </dec>
    </listaDecs>
  </parteDecs>
  <bloqueInst>
    <BEGIN/>
    <listaInst>
      <instruccion>
        <IDENTIFICADOR/>
        <OPAS/>
        <expr>
          <expSimple>
            <term>
              <term>
                <factor>
                  <constante>
                    <CONSTENTERA/>
                  </constante>
                </factor>
              </term>
              <opMul>
                <DIV/>
              </opMul>
              <factor>
                <constante>
                  <CONSTENTERA/>
                </constante>
              </factor>
            </term>
          </expSimple>
        </expr>
      </instruccion>
    </listaInst>
    <END/>
  </bloqueInst>
  <PUNTO/>
</programa>
```