Universidad de Málaga
Departamento de Lenguajes y Ciencias de la Computación
Campus de Teatinos, 29071 MÁLAGA

Examen de Traductores, Intérpretes y Compiladores
Convocatoria ordinaria de Septiembre de 2001
3er Curso de I.T. Informática de Sistemas

Apellidos, Nombre: _____ ***Calificación:*** _____

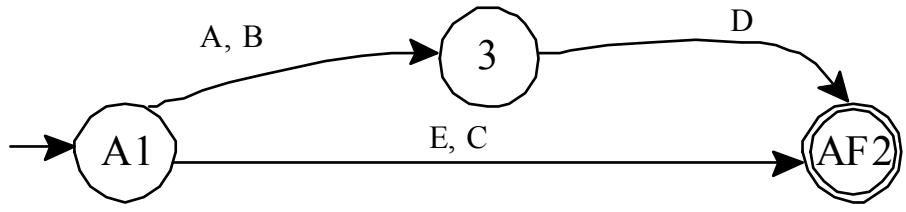
PRÁCTICA

Se desea construir un lenguaje que permita la creación de autómatas formales. Para ello se suministra la siguiente gramática:

```
automata     :      /* Epsilon */
                    |      automata estado ';'
                    |      automata transicion ';'
                    |      automata error ';'
                    ;
estado       :      ESTADO ID tipo
             ;
tipo         :      TERMINAL
             |      INICIAL
             |      /* Épsilon */
             ;
transicion   :      TRANSITA DE ID A ID CON ENTRADA listaCadenas
             ;
listaCadenas:       CADENA
             |      listaCadenas ‘,’ CADENA
             ;
```

Un ejemplo de entrada sería el siguiente:

```
ESTADO A1 INICIAL;
ESTADO AF2 TERMINAL;
ESTADO 3;
```




```
TRANSITA DE A1 A 3 CON ENTRADA "A", "B" ;
TRANSITA DE A1 A AF2 CON ENTRADA "E", "C";
TRANSITA DE 3 A AF2 CON ENTRADA "D";
```

que se corresponde con el autómata:

Se pide construir los programas Lex y Yacc que reconozcan sentencias de este lenguaje, teniendo en cuenta que es necesario realizar los siguientes controles semánticos:

- No hay problema en declarar más de una vez el mismo nombre de estado. Si esto ocurre

Open Course Ware
Gálvez Rojas, S.
(2011). Traductores, Compiladores e Intérpretes
OCW- Universidad de Málaga http://ocw.uma.es
Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike
BY NC SA

se considera rá una redefinición de su tipo: terminal, final o intermedio.

- Si uno de los estados de una transición no existe, **no** se considerará error. Se introduce en la lista de estados y ya está.
- Una misma cadena se puede utilizar en más de una transición.
- Una vez creado el autómata, se desea saber si éste es determinista o no a nivel de transición, esto es, si a partir de un estado es posible llegar a más de un destino a través de la misma cadena de entrada.
- Una vez creado el autómata, se desea saber si éste es conexo o no.

Para todo ello se suiminstra la siguiente tabla de símbolos que, **es obligatorio** utilizar cuando sea posible:

Fichero TabSimb.c

```
struct _transicion;
typedef struct _cadena {
      struct _cadena * sig;
      char texto[20];
} Cadena;

typedef struct _estado {
      struct _estado * sig;
      char nombre[20];
      struct _transicion * listaTransiciones;
      char tipo; /*´t´erminal, ´i´nicial, ´u´ otro */
} Estado;

typedef struct _transicion {
      struct _transicion * sig;
      Estado * destino;
      Cadena * listaCadenas;
} Transicion;

Cadena * nuevaCadena(){
      return (Cadena *)malloc(sizeof(Cadena));
}

Estado * nuevoEstado(){
      return (Estado *)malloc(sizeof(Estado));
}

Transicion * nuevaTransicion(){
      return (Transicion *)malloc(sizeof(Transicion));
}
Estado * buscar(Estado * t, char * s){
      while ((t != NULL)&&(strcmp(t->nombre, s))){
            t = t-> sig;
      }
      return t;
}
```


Bajo licencia Creative Commons Attribution-Non-Comercial-ShareAlike