Conversor de gramáticas - autómatas finitos

1. Objetivo

El trabajo consiste en generar un conversor de gramáticas regulares a autómatas finitos y viceversa. Dada una gramática regular se debe obtener una representación grafica del autómata finito equivalente. Dada una especificación del autómata finito, debe poder obtener la especificación de la gramática equivalente.

2. Descripción del funcionamiento esperado

2.1. si la entrada es archivo .gr

Si el conversor recibe un archivo de extensión .gr, se asume que se desea la conversión de gramática a autómata. Dada la gramática de entrada, deberá obtener:

1. Símbolos terminales.
2. Símbolos no terminales.
3. Símbolo inicial.
4. Si la gramática es válida.
5. Si la gramática es regular (en caso de que lo sea, si es regular a derecha o a izquierda).
6. Un gráfico del automata finito equivalente.

Los items anteriores se mostrarán en salida estandar, a excepción del último que será a un archivo de imagen (de extensión .png).
Si el autómata no se puede realizar, especificar con claridad cuál es el motivo.
Si la gramática no está en forma normal, habrá que normalizarla, esto es, llevarla a la forma:
$A \to bC$
$A \to \lambda$
o bien
$A \to Bc$
$A \to \lambda$
Donde A, B, C son símbolos no terminales y b, c son símbolos terminales.
Si la gramática es regular a izquierda, deberá convertirla a gramática regular derecha.

2.2. si la entrada es archivo .dot

Si el conversor recibe un archivo de extensión .dot, se asume que se desea la conversión de autómata a gramática.
Dado el autómata de entrada, deberá obtener:

1. Símbolos terminales.
2. Estados.
3. Estado inicial.
4. Conjunto de estados finales.
5. Tabla de la función de transición.
6. La especificación **completa** de la gramática equivalente.

Los items anteriores se mostrarán en salida estándar, a excepción del último que será a un archivo de texto (de extensión .gr).
Si la gramática no se puede obtener, especificar con claridad cuál es el motivo. No es necesario que la gramática resultante esté normalizada.

3. Formato de los archivos

3.1. Archivos .gr

En el caso de tratarse de un archivo de gramática (.gr) será un archivo en el que se especificarán las gramáticas de la siguiente forma:

NombreDeLaGramatica = ({SimbolosNoTerminales}, {SimbolosTerminales}, SimboloInicial, {Producciones})

Consideraciones:

- Los SimbolosNoTerminales son letras en mayúscula separadas por comas.
- Los SimbolosTerminales son letras en minúscula separadas por comas.
- Para el símbolo lambda se usará el caracter \ (barra invertida).
- Se debe contemplar que los caracteres del archivo pueden (o no) estar separados por espacios, tabuladores o fines de linea.

El siguiente podría ser el contenido de un archivo válido:
G1 = ({A, B, C}, {a, b, c},A, {A → aB|c, B → aA|b})

G2 = (
{A, B, C, D},
{a, b, c},
A,
{A → aB|*c*, B → C|b, C→D | cD | b, D → \}
)

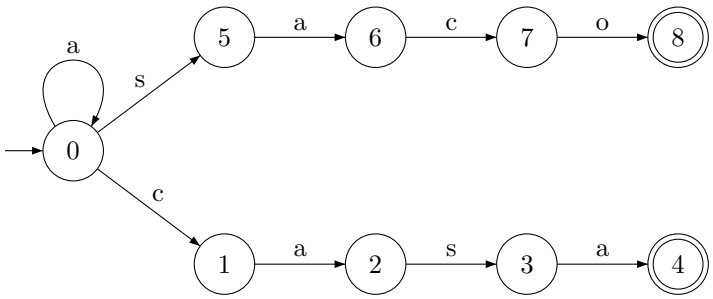
3.2. Archivos .dot

El formato de los archivos para los autómatas -que permitirá crear un gráfico de los mismos usando la libreria Graphviz- es el siguiente:

```
digraph {
// Nodos
// Un estado de no aceptacion, con un 0 como etiqueta
node [shape=circle] Node0 [label="0"];

// Un estado de aceptacion, con un 1 como etiqueta
node [shape=doublecircle] Node1 [label="1"];
....
// Transiciones
Node0 -> Node0 [label="a"]; // Un lazo en el estado cero
Node0 -> Node1 [label="a/b"]; // Transicion del estado cero al uno
....
}
```

El nodo inicial siempre será el de etiqueta 0. Los demás nodos tendrán por etiqueta cualquier otro **numero**.
Ejemplo: Para el siguiente NFA



La especificación en un .dot es:

```
digraph  {
rankdir = "LR";   // De derecha a izquierda
// Nodos
node [shape=circle] Node0 [label="0"];
node [shape=circle] Node1 [label="1"];
node [shape=circle] Node2 [label="2"];
node [shape=circle] Node3 [label="3"];
node [shape=doublecircle] Node4 [label="4"];
node [shape=circle] Node5 [label="5"];
node [shape=circle] Node6 [label="6"];
node [shape=circle] Node7 [label="7"];
node [shape=doublecircle] Node8  [label="8"];

// Transiciones
Node0 -> Node0 [label="a"];
Node0 -> Node1 [label="c"];
Node1 -> Node2 [label="a"];
Node2 -> Node3 [label="s"];
Node3 -> Node4 [label="a"];

Node0 -> Node5 [label="s"];
Node5 -> Node6 [label="a"];
Node6 -> Node7 [label="c"];
Node7 -> Node8 [label="o"];

}
```

4. Material a entregar

- Códigos fuente: Makefile, código Lex (sin compilar), y código C.
- Archivo ejecutable para Linux.
- Un readme explicando cómo compilar y ejecutar el programa.
- Archivos de ejemplo diferentes a las provistas por la cátedra.

- Un informe que contenga, en este orden:
  - Carátula
  - Índice
  - Consideraciones realizadas (no previstas en el enunciado).
  - Descripción del desarrollo del TP.
  - Dificultades encontradas en el desarrollo del TP.
  - Futuras extensiones, con una breve descripción de la complejidad de cada una.

5. Fecha de entrega

El material a entregar debe ser enviado por mail, en un archivo tipo .zip a la cuenta `ariasroigana@gmail.com` antes de finalizado el día 7 de octubre.

6. Material de consulta

Se sugiere leer los documentos de las páginas:

- http://www.linux-magazine.es/issue/29/046-049_GraphvizLM29.crop.pdf
- http://www.graphviz.org/pdf/libguide.pdf