Instituto Politécnico Nacional

Escuela Superior de Cómputo

Compiladores

Practica 3: AFN-épsilon a AFD

Nombre: *Mauro Sampayo Hernández*

Grupo: *3CV17*

Profesor: *Hernández Olvera Luis Enrique*

Fecha de entrega: *13 de octubre del 2021*

1. Introducción:

1.1 Operación Cerradura épsilon

Dado un AFN se define la operación de Cerradura épsilon como:

- $\boldsymbol{Cerradura\_\varepsilon}\ (\boldsymbol{s})$ **:** Conjunto de estados del AFN a los que se puede llegar desde el estado s del AFN, sólo en las transiciones ε.
- $\boldsymbol{Cerradura\_\varepsilon}\ (\boldsymbol{T})$ **:** Conjunto de estados del AFN a los que se puede llegar desde cierto estado s del AFN en el conjunto T, sólo en las transiciones ε; = $\cup_{s\ en\ T}\ Cerradura\_\varepsilon$.

Donde s es un estado, a es un símbolo del alfabeto del AFN y T es un conjunto de estados del AFN.

1.2 Operación Mover

Dado un AFN se define la operación de Mover como:

- $\boldsymbol{Mover}\ (\boldsymbol{T}, \boldsymbol{a})$ **:** Conjunto de estados del AFN para los cuales hay una transición sobre el símbolo de entrada a**,** a partir de cierto estado s en T.

Donde s es un estado y T es un conjunto de estados del AFN.

1.3 Operación Ir A

Dado un AFN se define la operación de Ir A como:

- $\boldsymbol{Ir\_A(T, a) = C\_\varepsilon(Mover(T, a))}$.Escriba aquí la ecuación.

Donde a es un símbolo del alfabeto del AFN y T es un conjunto de estados del AFN.

1.4 Algoritmo de Conversión de un AFN-épsilon a un AFD

1. Se realiza el cálculo de la Cerradura Épsilon del estado inicial del AFN, y el resultado de esta será el estado inicial S_0 del AFD, así como el estado inicial del AFD
2. Se calcula para cada estrado la operación Ir A para cada $\alpha\ \epsilon\ \Sigma$, la cual arrojara un estado S_i pudiendo estso repetirse
3. Se realiza el paso 2 con todos los estados hasta que ya no surjan estados nuevos.
4. Una vez finalizada la conversión, se definen como estados finales del AFD, todos aquellos estados S_i que contengan el estado final del AFN original.

1.5 Ejemplo de Conversión de un AFN-épsilon a un AFD

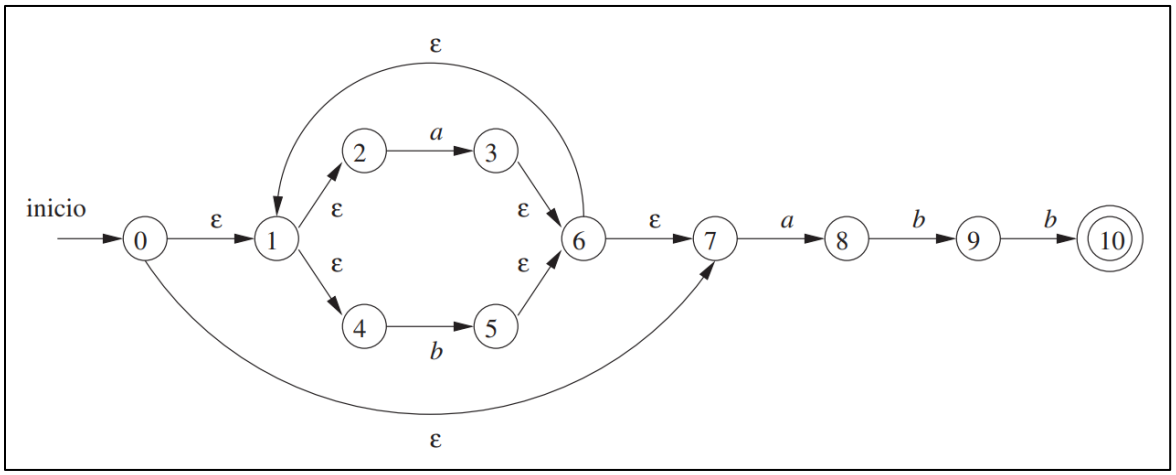


*Figura 1. AFN para (a|b)*abb*

$$\boldsymbol{C\_\varepsilon(0) = \{0, 1, 2, 4, 7\} = A}$$

$$\boldsymbol{Ir\_A(A, a) = C\_\varepsilon(Mover(A, a)) = C\_\varepsilon(3, 8) = \{1, 2, 3, 4, 6, 7, 8\} = B}$$

$$\boldsymbol{Ir\_A(A, b) = C\_\varepsilon(Mover(A, b)) = C\_\varepsilon(5) = \{1, 2, 4, 5, 6, 7\} = C}$$

$$\boldsymbol{Ir\_A(B, a) = C\_\varepsilon(Mover(B, a)) = C\_\varepsilon(3, 8) = B}$$

$$\boldsymbol{Ir\_A(B, b) = C\_\varepsilon(Mover(B, b)) = C\_\varepsilon(5, 9) = \{1, 2, 4, 5, 6, 7, 9\} = D}$$

$$\boldsymbol{Ir\_A(C, a) = C\_\varepsilon(Mover(C, a)) = C\_\varepsilon(3, 8) = B}$$

$$\boldsymbol{Ir\_A(C, b) = C\_\varepsilon(Mover(C, b)) = C\_\varepsilon(5) = C}$$

$$\boldsymbol{Ir\_A(D, a) = C\_\varepsilon(Mover(D, a)) = C\_\varepsilon(3, 8) = B}$$

$$\boldsymbol{Ir\_A(D, b) = C\_\varepsilon(Mover(D, b)) = C\_\varepsilon(5, 10) = \{1, 2, 4, 5, 6, 7, 10\} = E}$$

$$\boldsymbol{Ir\_A(E, a) = C\_\varepsilon(Mover(E, a)) = C\_\varepsilon(3, 8) = B}$$

$$\boldsymbol{Ir\_A(E, b) = C\_\varepsilon(Mover(E, b)) = C\_\varepsilon(5) = C}$$

➔ “E” es el estado final del AFD, ya que contiene al estado final del AFN original

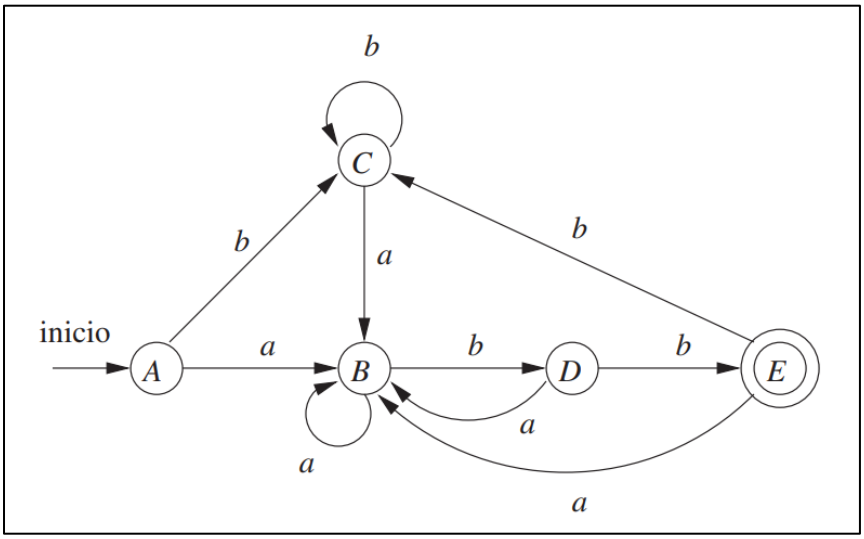


Figura 2. AFD resultante de la conversión

2. Problemas enfrentados al programar la práctica:

Los problemas enfrentados durante la realización del programa estuvieron prácticamente en su totalidad en la parte de realizar la conversión de AFN-épsilon a AFD, pues para la generación y formalización del AFN-épsilon recibido en un txt se reutilizo el código de la práctica pasada que se encargaba de este mismo proceso, y a este se le realizaron unas pequeñas modificaciones.

Para la realización de la operación **Cerradura épsilon** la problemática fue la de generar un algoritmo que pudiese identificar todos los estados alcanzables por transiciones épsilon desde los estados de entrada. Para esto se optó por un algoritmo que evaluará el resultado de realizar una transición épsilon en cada uno de los valores de entrada, para así conocer los estados alcanzables con una transición épsilon desde los estados de entrada e incluirlos en la lista de estados resultantes siempre y cuando estos estados alcanzables no fuesen vacíos. Se repite este mismo procedimiento con todos los estados que sean añadidos a la lista, hasta que no queden más estados a evaluar. Además, dentro de la lista de estados resultantes se incluirán los estados de entrada.

Para la operación **Mover** se empleo como base el algoritmo realizado para la operación **Cerradura Épsilon**, y se le realizaron modificaciones para que en este caso únicamente evaluará el resultado de realizar una transición con el símbolo especificado para la operación, y que debe pertenecer al alfabeto del AFN, en cada uno de los valores de entrada, para así conocer los estados alcanzables con dicha transición e incluirlos en la lista de estados resultantes de la operación siempre y cuando estos estados alcanzables no fuesen vacíos. En este caso este proceso solo se realiza una vez por cada estado de entrada y no se incluyen los estados de entrada dentro de la lista de estados resultantes.

Finalmente, para el proceso de conversión del AFN-épsilon a AFD se enfrentaron dos problemas. El primero de estos problemas fue referente a el proceso de conversión en si mismo, pues si bien para dicho proceso únicamente se hizo uso de las operaciones **Cerradura épsilon** y Mov**er** para generar la operación **Ir A,** se tuvieron que considerar algoritmos para el caso de que la operación Mover devolviese una cadena vacía (implicando la presencia de un estado de error en el AFD resultante) y el caso de que el resultado de la operación **Ir A** devolviese una cadena de estados previamente evaluada y que por tanto indica que el programa no debe generar un nuevo estado para el AFD al ya existir uno correspondiente a dicha cadena.

El segundo problema enfrentado fue en la parte de la generación de los estados finales del AFD, pues el algoritmo de conversión de AFN-épsilon a AFD debía saber identificar cuales de los estados generados para el AFD contenía el estado final del AFN-épsilon en su cadena de estados del AFN-épsilon correspondiente. Esto se soluciono sencillamente con una condicional.

3. Pruebas y Capturas de Pantalla:

Se realizaron dos pruebas con el programa, utilizando como valores de entrada dos archivos .txt proporcionados por el profesor para la generación de dos AFN-épsilon y su posterior conversión a un AFD.

Cada renglón del archivo de texto de entrada representa los datos de cada elemento de la tupla del AFN a generar. El primer renglón representa la lista de estados del AFN, el segundo los símbolos del alfabeto, el tercero el estado inicial, el cuarto el estado final y el resto de los renglones representa las transiciones del AFN (enlistadas iniciando con el estado de origen, seguido del símbolo de transición y finalizando con el estado resultante de dicha transición).

A continuación, se describen los valores de entrada proporcionados y los resultados de cada prueba (la ejecución del programa se hizo usando el compilador de Phyton en el IDE PyCharm):

3.1 Autómata Thompson 1

Para el primer autómata se nos proporciona el siguiente archivo de texto:

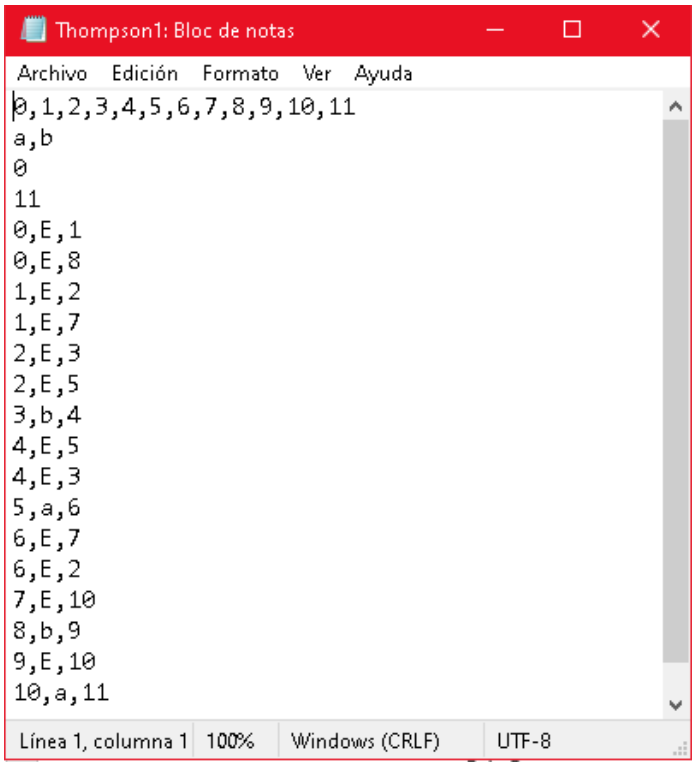
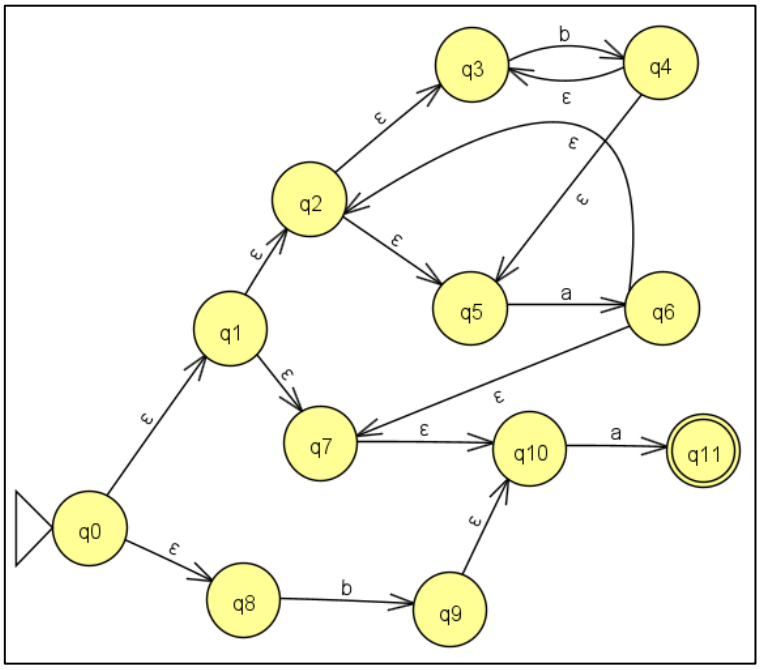


Figura 3. Archivo de texto de entrada para la generación del primer AFN-épsilon

Los datos enlistados en este archivo de texto representan un AFN-épsilon cuyo grafo y tabla de transiciones se muestran a continuación:



Estado	Símbolo entrada		
	a	b	ε
0	-	-	{1, 8}
1	-	-	{2, 7}
2	-	-	{3, 5}
3	-	4	-
4	-	-	{3, 5}
5	6	-	-
6	-	-	{2, 7}
7	-	-	10
8	-	9	-
9	-	-	10
10	11	-	-
11	-	-	-

Figura 4. AFN-épsilon resultante del archivo de texto de la Figura 3

El resto de los elementos de la tupla del AFN-épsilon quedan definidos de la siguiente manera:

Q: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

∑: {a, b, ε}

q_0**:** {0}

F:11

- **Generación del AFN-épsilon**

Al ejecutarse, el programa generará los datos de las tuplas del autómata de la Figura 4, y los mostrará en pantalla:

```
"C:\Users\146261\Documents\IPN_ESCOM\Compiladores\Prácti
---------- AFN epsilon FORMALIZADO ----------

Estados = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, _err)
Alfabeto = (a, b, E)
Estado inicial = 0
Estado finale = 11
Tabla de Transiciones:

Estado    a         b         E
--------  --------  --------  ----------
0         ['_err']  ['_err']  ['1', '8']
1         ['_err']  ['_err']  ['2', '7']
2         ['_err']  ['_err']  ['3', '5']
3         ['_err']  ['4']     ['_err']
4         ['_err']  ['_err']  ['5', '3']
5         ['6']     ['_err']  ['_err']
6         ['_err']  ['_err']  ['7', '2']
7         ['_err']  ['_err']  ['10']
8         ['_err']  ['9']     ['_err']
9         ['_err']  ['_err']  ['10']
10        ['11']    ['_err']  ['_err']
11        ['_err']  ['_err']  ['_err']
_err      ['_err']  ['_err']  ['_err']
```

Figura 5. Generación del primer AFN con transiciones épsilon en el programa de Phyton.

- **Conversión del AFN-épsilon a AFD**

Una vez se haya generado el autómata, se realizará internamente el proceso de conversión del AFN-épsilon a un AFD. Dicho proceso de conversión se mostrará a continuación:

$$\boldsymbol{C\_\varepsilon(0) = \{0, 1, 2, 3, 5, 7, 8, 10\} = A}$$

$$\boldsymbol{Ir\_A(A, a) = C\_\varepsilon(Mover(A, a)) = C\_\varepsilon(6, 11) = \{2, 3, 5, 6, 7, 10, 11\} = B}$$

$$\boldsymbol{Ir\_A(A, b) = C\_\varepsilon(Mover(A, b)) = C\_\varepsilon(4, 9) = \{3, 4, 5, 9, 10\} = C}$$

$$Ir\_A(B, a) = C\_\varepsilon(Mover(B, a)) = C\_\varepsilon(6, 11) = B$$

$$Ir\_A(B, b) = C\_\varepsilon(Mover(B, b)) = C\_\varepsilon(4) = \{3, 4, 5\} = D$$

$$Ir\_A(C, a) = C\_\varepsilon(Mover(C, a)) = C\_\varepsilon(6, 11) = B$$

$$Ir\_A(C, b) = C\_\varepsilon(Mover(C, b)) = C\_\varepsilon(4) = D$$

$$Ir\_A(D, a) = C\_\varepsilon(Mover(D, a)) = C\_\varepsilon(6) = \{2, 3, 5, 6, 7, 10\} = E$$

$$Ir\_A(D, b) = C\_\varepsilon(Mover(D, b)) = \ C\_\varepsilon(4) = D$$

$$Ir\_A(E, a) = C\_\varepsilon(Mover(E, a)) = C\_\varepsilon(6, 11) = B$$

$$Ir\_A(E, b) = C\_\varepsilon(Mover(E, b)) = C\_\varepsilon(4) = D$$

Una vez finalizado este proceso el programa mostrará los datos de la tupla del AFD resultante de la conversión:

```
---------- AFD FORMALIZADO ----------

Estados = (A, B, C, D, E)
Alfabeto = (a, b)
Estado inicial = A
Estados finales = (B)
Tabla de Transiciones:

Estado    a    b
--------  ---  ---
A         B    C
B         B    D
C         B    D
D         E    D
E         B    D
```

Figura 6. AFD resultante de la conversión

Los datos enlistados como resultado de la conversión representan un AFD cuyo grafo y tabla de transiciones se muestran a continuación:

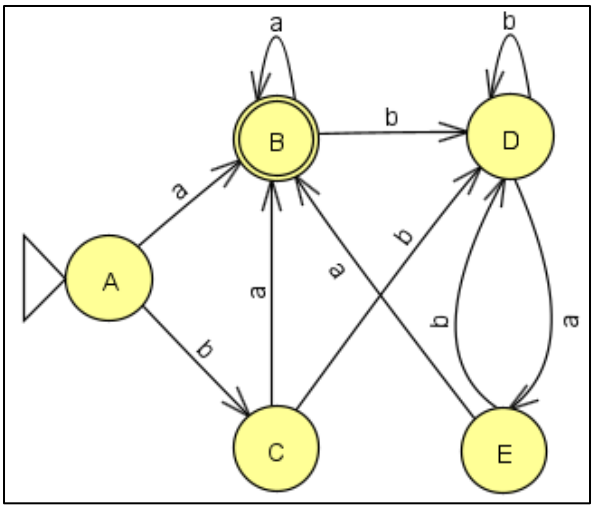


Figura 7. AFD resultante de la conversión

Estado	Símbolo entrada	
	a	b
A	B	C
B	B	D
C	B	D
D	E	D
E	B	D

El resto de los elementos de la tupla del AFN-épsilon quedan definidos de la siguiente manera:

Q: {A, B, C, D, E}

∑: {a, b}

q_0: A

F: {B}

3.2 Autómata Thompson 2

Para el segundo autómata se nos proporciona el siguiente archivo de texto:

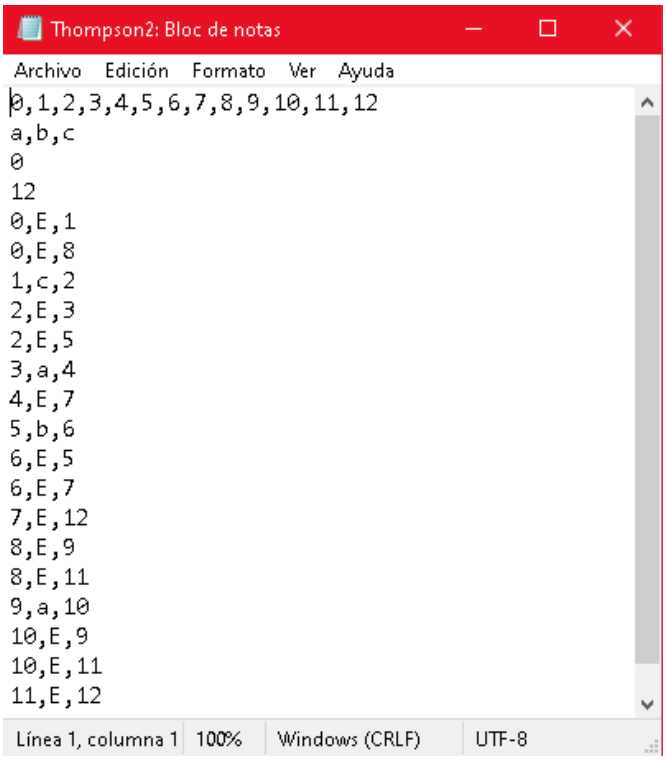


Figura 8. Archivo de texto de entrada para la generación del segundo AFN con transiciones épsilon

Los datos enlistados en este archivo de texto representan un AFN-épsilon cuyo grafo y tabla de transiciones se muestran a continuación:

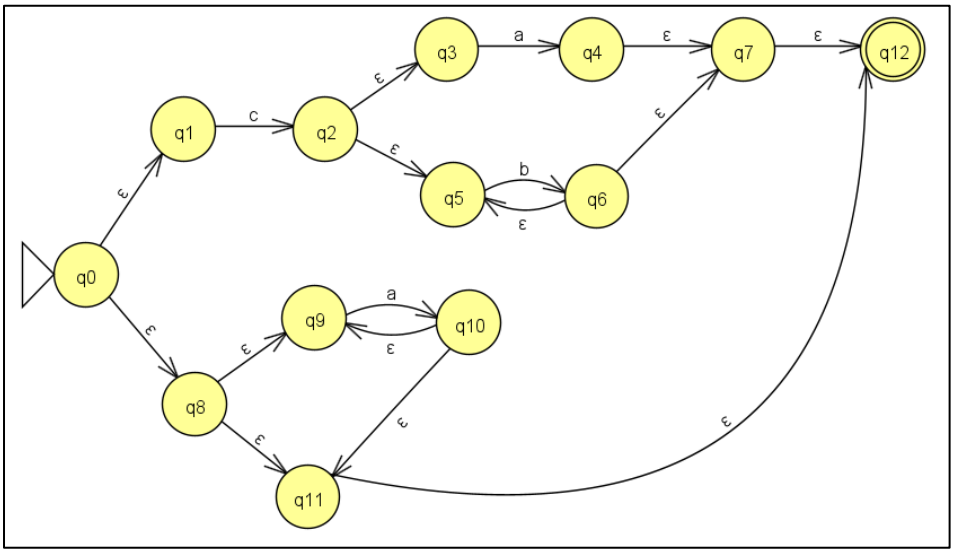


Figura 9. Autómata resultante del archivo de texto de la Figura 8

El resto de los elementos de la tupla del AFN-épsilon quedan definidos de la siguiente manera:

Q: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

∑: {a, b, c, ε}

q_0: {0}

F: 12

Estado	Símbolo entrada			
	a	b	c	ε
0	-	-	-	{1, 8}
1	-	-	2	-
2	-	-	-	{3, 5}
3	4	-	-	-
4	-	-	-	7
5	-	6	-	-
6	-	-	-	{5, 7}
7	-	-	-	12
8	-	-	-	{9, 11}
9	10	-	-	-
10	-	-	-	{9, 11}
11	-	-	-	12
12	-	-	-	-

- **Generación del autómata en el programa**

Al ejecutarse, el programa generará los datos de las tuplas del autómata de la Figura 9, y los mostrará en pantalla:

```
"C:\Users\146261\Documents\IPN_ESCOM\Compiladores\Práctica 3
---------- AFN epsilon FORMALIZADO ----------

Estados = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, _err)
Alfabeto = (a, b, c, E)
Estado inicial = 0
Estado finale = 12
Tabla de Transiciones:

Estado    a         b         c         E
--------  --------  --------  --------  -----------
0         ['_err']  ['_err']  ['_err']  ['1', '8']
1         ['_err']  ['_err']  ['2']     ['_err']
2         ['_err']  ['_err']  ['_err']  ['3', '5']
3         ['4']     ['_err']  ['_err']  ['_err']
4         ['_err']  ['_err']  ['_err']  ['7']
5         ['_err']  ['6']     ['_err']  ['_err']
6         ['_err']  ['_err']  ['_err']  ['5', '7']
7         ['_err']  ['_err']  ['_err']  ['12']
8         ['_err']  ['_err']  ['_err']  ['9', '11']
9         ['10']    ['_err']  ['_err']  ['_err']
10        ['_err']  ['_err']  ['_err']  ['9', '11']
11        ['_err']  ['_err']  ['_err']  ['12']
12        ['_err']  ['_err']  ['_err']  ['_err']
_err      ['_err']  ['_err']  ['_err']  ['_err']
```

Figura 10. Generación del segundo AFN con transiciones épsilon en el programa de Phyton

- **Conversión del AFN-épsilon a AFD**

Una vez se haya generado el autómata, se realizará internamente el proceso de conversión del AFN-épsilon a un AFD. Dicho proceso de conversión se mostrará a continuación:

$$\boldsymbol{C\_\varepsilon(0) = \{0, 1, 8, 9, 11, 12\} = A}$$

$$\boldsymbol{Ir\_A(A, a) = C\_\varepsilon(Mover(A, a)) = C\_\varepsilon(10) = \{9, 10, 11, 12\} = B}$$

$$\boldsymbol{Ir\_A(A, b) = C\_\varepsilon(Mover(A, b)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(A, c) = C\_\varepsilon(Mover(A, c)) = C\_\varepsilon(2) = \{2, 3, 5\} = C}$$

$$\boldsymbol{Ir\_A(B, a) = C\_\varepsilon(Mover(B, a)) = C\_\varepsilon(10) = B}$$

$$\boldsymbol{Ir\_A(B, b) = C\_\varepsilon(Mover(B, b)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(B, c) = C\_\varepsilon(Mover(B, c)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(C, a) = C\_\varepsilon(Mover(C, a)) = C\_\varepsilon(4) = \{4, 7, 12\} = D}$$

$$\boldsymbol{Ir\_A(C, b) = C\_\varepsilon(Mover(C, b)) = C\_\varepsilon(6) = \{5, 6, 7, 12\} = E}$$

$$\boldsymbol{Ir\_A(C, c) = C\_\varepsilon(Mover(C, c)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(D, a) = C\_\varepsilon(Mover(D, a)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(D, b) = C\_\varepsilon(Mover(D, b)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(D, b) = C\_\varepsilon(Mover(D, b)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(E, a) = C\_\varepsilon(Mover(E, a)) = C\_\varepsilon(\emptyset) = \emptyset}$$

$$\boldsymbol{Ir\_A(E, b) = C\_\varepsilon(Mover(E, b)) = C\_\varepsilon(6) = E}$$

$$\boldsymbol{Ir\_A(D, b) = C\_\varepsilon(Mover(D, b)) = C\_\varepsilon(\emptyset) = \emptyset}$$

Una vez finalizado este proceso el programa mostrará los datos de la tupla del AFD resultante de la conversión:

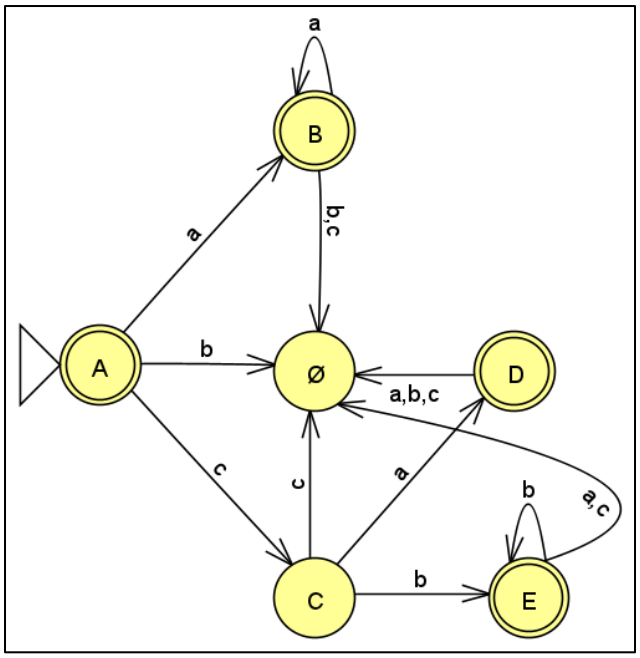
```
---------- AFD FORMALIZADO ----------

Estados = (A, B, C, D, E, err)
Alfabeto = (a, b, c)
Estado inicial = A
Estados finales = (A, B, D, E)
Tabla de Transiciones:

Estado    a     b     c
--------  ----  ----  ----
A         B     _err  C
B         B     _err  _err
C         D     E     _err
D         _err  _err  _err
E         _err  E     _err
err       _err  _err  _err
```

Figura 11. AFD resultante de la conversión

Los datos enlistados como resultado de la conversión representan un AFD cuyo grafo y tabla de transiciones se muestran a continuación:



Estado	Símbolo entrada		
	a	b	c
A	B	Ø	C
B	B	Ø	Ø
C	D	E	Ø
D	Ø	Ø	Ø
E	Ø	E	Ø
Ø	Ø	Ø	Ø

Figura 12. AFD resultante de la conversión

El resto de los elementos de la tupla del AFN-épsilon quedan definidos de la siguiente manera:

Q: {A, B, C, D, E, Ø }

∑: {a, b, c}

$\boldsymbol{q_0}$**:** A

F: {A, B, D, E}

4. Conclusión:

Como conclusión debo destacar que este proceso de conversión, de un AFN-épsilon a un AFD, resulta de gran utilidad al brindarnos una manera de hacer más simples los AFNs con transiciones épsilon, al ser que la conversión de estos a un AFD involucra la eliminación de todas las transiciones épsilon, lo que se traduce en una disminución bastante significativa de los estados que lo conforman, así como también en que el AFD resultante será mucho más sencillo de entender, lo que brinda diversas ventajas tales como la de facilitar la programación de estos.

Considero además que este proceso de conversión es quizá la herramienta más poderosa de la materia, pues a partir de esta podemos realizar la programación de autómatas que reconozcan lenguajes regulares para la elaboración de compiladores, de una manera mucho más fácil, por medio de los AFDs resultantes, que puedan ser obtenidos a partir de los AFNs épsilon que se construyan a partir de las nomenclaturas de Thompson para dicho compilador anteriormente mencionado.

5. Referencias:

- Aho, A., 2011. Compiladores. 2nd ed. Pearson Educación de México, SA de CV, pp.152-156.
- Hernandez Maza, J., Saldaña Chiquinta, C., Tello Arevalo, K. and Flores Sánchez, J., n.d. Algoritmos de Conversión. [online] Es.scribd.com. Disponible en: <https://es.scribd.com/document/471647185/ALGORITMOS-DE-CONVERSION-AFND-AFD#fullscreen&from_embed> [Accedido el 12 de octubre de 2021].
- Tenorio, A., 2014. CONVERSION DE UN AFND A UN AFD Y DE UN AFND-E A UN AFD. [online] prezi.com. Disponible en: <https://prezi.com/2-7vs1kjlmza/conversion-de-un-afnd-a-un-afd-y-de-un-afnd-e-a-un-afd/> [Accedido el 12 de octubre de 2021].