Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la computación
CC3102-1 Teoría de la Computación

Tarea n°1

Integrantes: Tomás Cortez

Profesor: Alejandro Hevia
Auxiliares: Ivana Bachmann Espinoza.
Rodrigo Fuentes Z.
Vicente Rojas C.

Fecha de entrega: 27 de octubre de 2019
Santiago, Chile

1. Pregunta 1

1.1.

Para esta pregunta se comenzó por modelar el AFD antes de la ER para esto se procedió de la siguiente forma. Los estados sabrán el resultado de aplicar módulo 9 al número computado hasta el momento y pasarán al resultado correspondiente al leer un nuevo dígito.

Sabemos que $n\,\%9 = k$ y tenemos que $(3n + d)\,\%9 = k'$ con $; d\epsilon 0, 1, 2$

Se tiene un sistema de ecuaciones el cual sigue así:

$(3n\,\%9 + d\,\%9)\,\%9 = k'$ Por propiedades del módulo

$((3(n\,\%9))\,\%9 + d)\,\%9 = k'$ ya que d es menor a 9

$((3k)\,\%9 + d)\,\%9 = k'$ Reemplazando k

De lo anterior se obtiene la siguiente tabla:

k\d	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8
3	0	1	2
4	3	4	5
5	6	7	8
6	0	1	2
7	3	4	5
8	7	8	9

donde las filas representan el resto actual y las columnas el dígito ingresado, por ejemplo en la columna 2, fila 3 se tiene el nuevo resto al ingresar un 2 cuando se contaba con un resto de 3.

Luego se pensaron en estos restos como estados del AFD y se utilizó la tabla anterior como función de transición.

Se obtuvo el siguiente AFD:

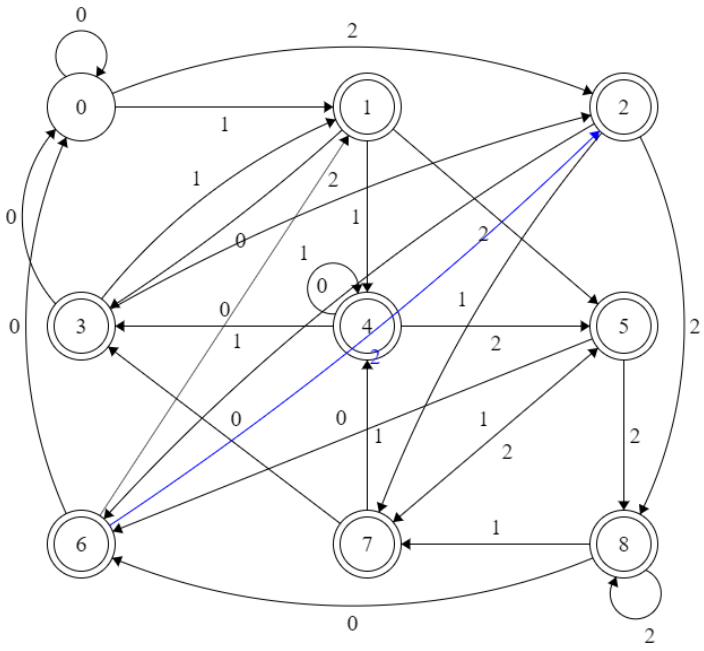

Figura 1.1: Esquema AFD

Mientras se transformaba el AFD a ER se llegó a un mejor entendimiento del lenguaje y por tanto a un AFD más compacto y de mejor visualización el cual es el siguiente:

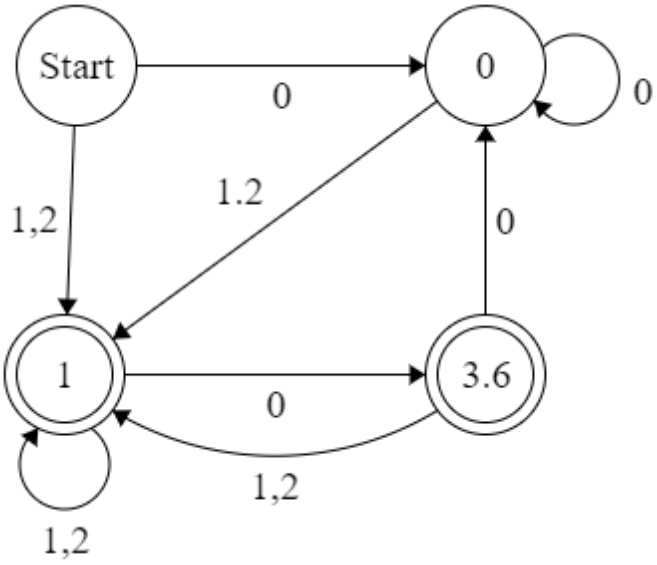

Figura 1.2: Esquema AFD mejorado

Luego ese AFD se pudo transformar de forma más fácil a ER y se llegó a la siguiente:
$(1|2)|(00^*(1|2))((1|2)^*|(0(1|2))|000*(1|2))^*(\epsilon|0)$

1.2.

Tomando en cuenta no se puede tener el substring ba se llega a la conclusión de que no puede haber una a, luego de una b, por lo tanto la expresión debe terminar con b. De la misma forma como debe tener el substring ab, se debe empezar con una a, por lo tanto se llega a la siguiente expresión regular: a^+b^+

Para el autómata se sigue la misma lógica, llevando los casos no favorables a un sumidero.

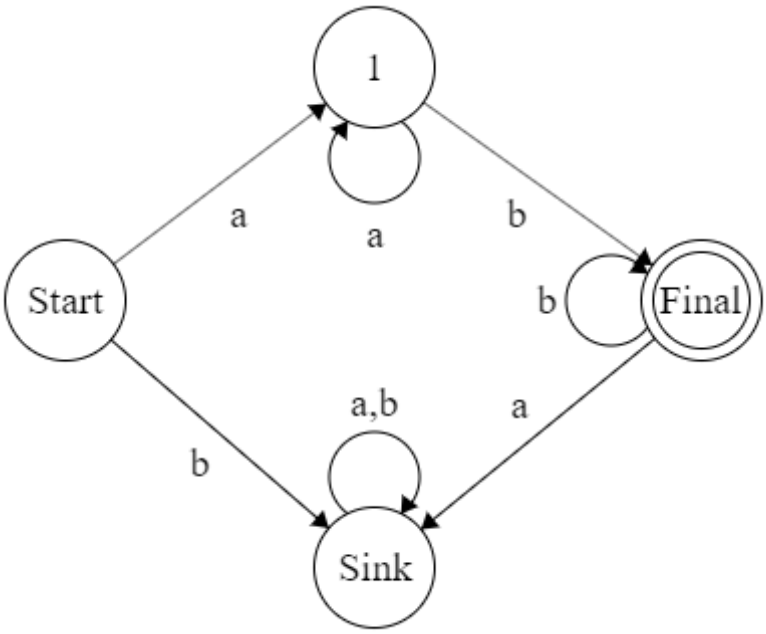

Figura 1.3: Esquema AFD

1.3.

Para esta expresión regular se pensó en los 5 casos posibles que se empiece por a, b y c con el substring abc en medio (3 casos) y los casos en que se empiece con el substring abc y se termine con este. Llegando a la siguiente ER: $a\Sigma^*abc\Sigma^*(b|c)|b\Sigma^*abc\Sigma^*(a|c)|c\Sigma^*abc\Sigma^*(a|b)|abc\Sigma^*(b|c)|(a|b)\Sigma^*abc$

Para el AFD Se pensaron en 3 caminos principales, empezar con a, b y c. Cada camino tiene 5 estados el estado 0 en el cual no se ha avanzado en el string abc, el estado 1 en el cual se ha avanzado en la letra a, etc. Estos estados vuelven al punto de partida o al estado 1 según corresponda. Sin embargo en el estado 3 ya se tiene el string abc listo, por lo tanto luego solo queda terminar en una letra distinta a la que se empezó.

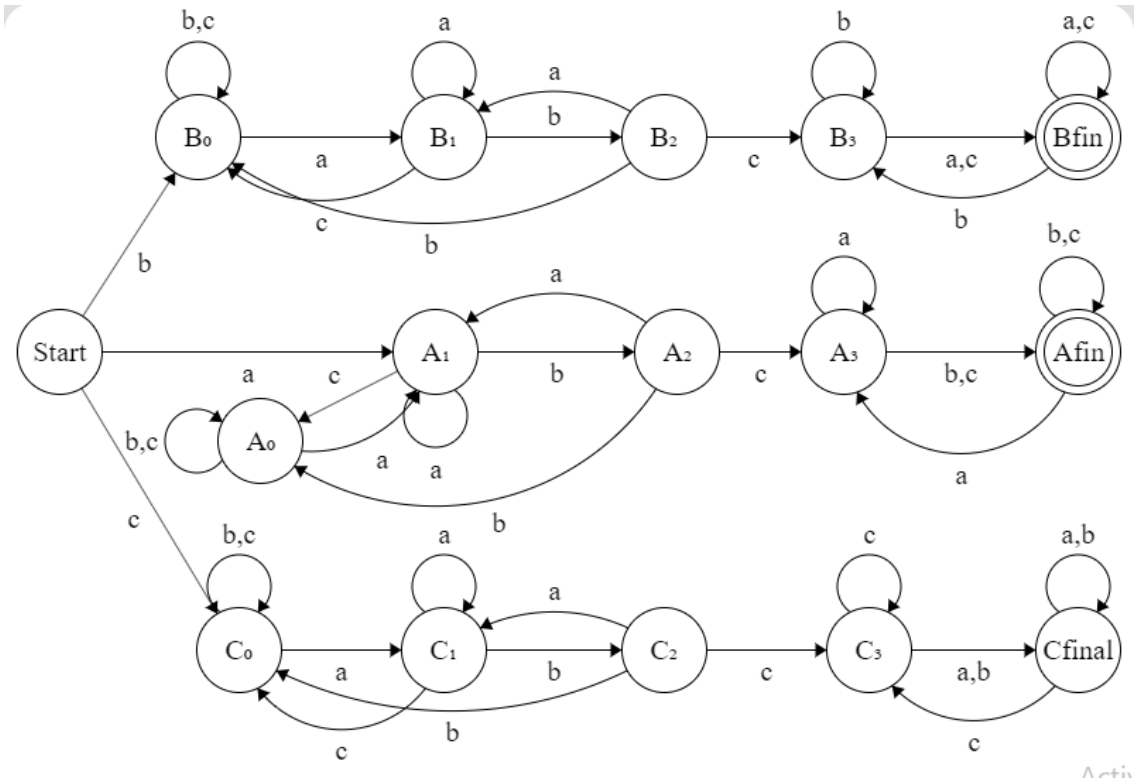

Figura 1.4: Esquema AFD

2. Pregunta 2

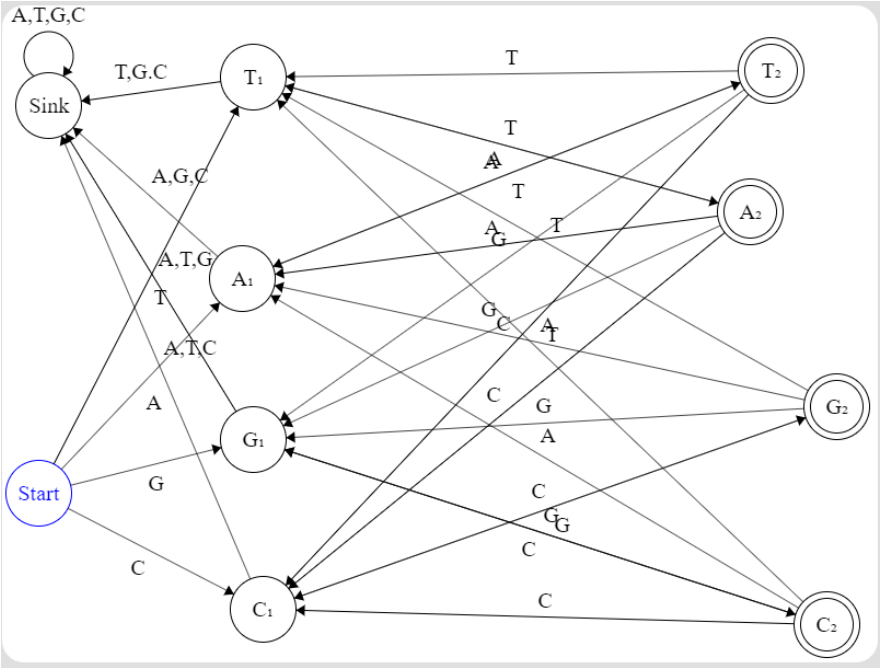

Figura 2.1: Esquema AFD general

Este autómata puede aceptar en principio cualquier letra de alfabeto, sin embargo después de aceptar una letra luego solamente puede aceptar a su par, una vez acepta un par puede aceptar cualquier letra otra vez, en caso de aceptar un par inválido va a un sumidero. Ahora se ve al autómata, pero con menos nodos para mayor claridad

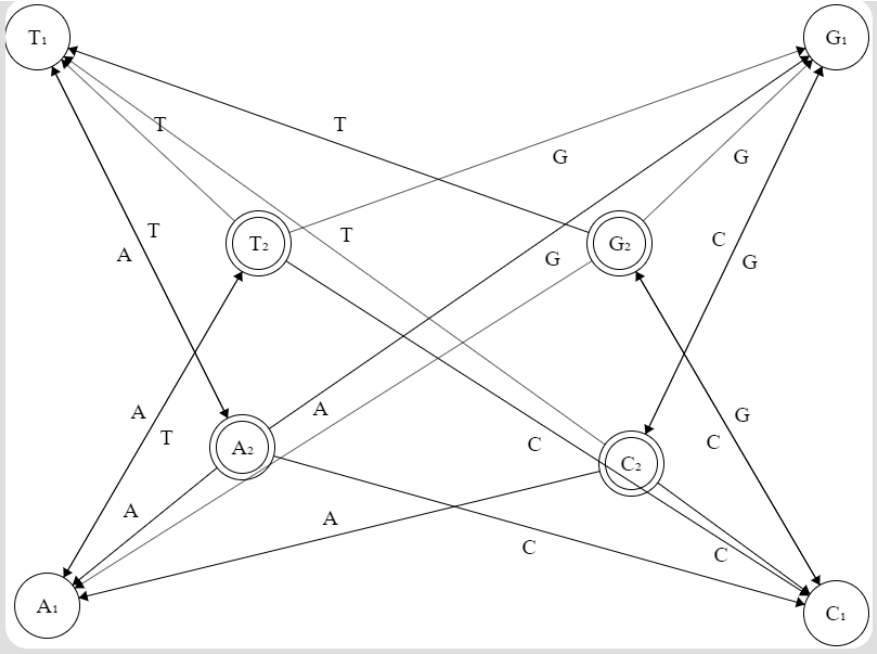

Figura 2.2: Esquema AFD general enfocado en los nodos intermedios

3. Pregunta 3

3.1. Operación TripleLargo(L)

Por cerradura de la intersección se sabe que lo que se obtiene es un lenguaje regular, ya que L es regular y las palabras múltiplo de 3 también, ya que basta con tener 3 estados guardando el múltiplo.

Luego para construir el autómata se proseguirá de la siguiente manera, se deberá tener tres copias del autómata principal etiquetados de forma de contar si es múltiplo de tres o no. Se tendrá la etiqueta de resto 0, resto 1 y resto 2, el resto cero serán los estados que importan, pues solamente contaremos los estados finales etiquetados con resto cero.

L es regular, por lo tanto existe un AFD $M = (Q_M, \Sigma, \delta_M, q_{0M}, F_M)$

Luego se construye el AFD M' $M = (Q_{M'}, \Sigma, \delta_{M'}, q_{0M'}, F_{M'})$:

$Q_{M'} = Q_M \times \{0, 1, 2\}$ Esta es la etiqueta que permitirá saber si un estado es múltiplo de tres o no.

Luego el estado inicial está etiquetado con 0, pues es múltiplo de tres.

$q_{0M'} = (q_{0M}, 0)$

$\delta_{M'}((q, 0), a) = (\delta_M(q, a), 1)$

$\delta_{M'}((q, 1), a) = (\delta_M(q, a), 2)$

$\delta_{M'}((q, 2), a) = (\delta_M(q, a), 0)$

Para todos los casos anteriores $q \in Q_M, a \in \Sigma$

La función de transición nos da a entender que actúa de la misma forma que el primer AFD con el cambio de ir cambiando su etiquetado con cada transición.

Por último los estados finales son solamente los múltiplos de tres.

$F_{M'} = F_M \times 0$

Para ver que funciona como se quiere se procederá de la siguiente manera.

Primero se analizará el comportamiento de una palabra w aceptada por L cuyo tamaño sea múltiplo de tres. Como es aceptada lleva desde el estado inicial a un estado de aceptación en una cierta cantidad de pasos para el autómata M, para el autómata M' se computa w desde el estado inicial de M' hasta uno de los estados finales en una cierta cantidad de pasos. Como la cantidad de pasos que se hizo es múltiplo de 3 el estado en el cual termina está etiquetado con un 0. Como se dijo que es un estado de aceptación de M y está etiquetado con un 0 esto significa que es un estado de aceptación de M'. Por lo tanto acepta los strings que se buscan.

Sin embargo podría aceptar más que solamente esos strings y en ese caso no estaría correcto el autómata por lo que se debe seguir analizando con más casos. Si se computa cualquier palabra w de largo que no sea múltiplo de 3 se llega a un estado etiquetado con un 1 o un 2, así que aunque sea una palabra reconocida por el autómata M usando la misma lógica anterior no llegará a un estado de aceptación en el autómata M'. Por último si se toma una palabra que no pertenece al lenguaje aceptado por M se llegará del estado inicial a un estado el cual no es de aceptación de M, por lo tanto en M',sin importar la cantidad de pasos que se tome, se llegará a un estado el cual no es de aceptación, pues no hay estados de aceptación en M' que sean producto de un estado en M que no sea de aceptación.

Ya habiendo cubierto estos casos se ha demostrado que la operación TripleLargo(L) mantiene la regularidad.

3.2. Operación Doble(L)

La intuición indica que el lenguaje no es regular, ya que si tuviese un autómata el cual está consumiendo el string para ver si lo acepta, no hay nada que le indique cuando termina un subs string y da comienzo el otro y una vez termina el string ya no hay forma de saber los estados pasados.

Dado N, se contruye p = vw, donde |v|=|w| = N. En caso que se tenga un y de tamaño impar, bastará con un bombeo de tamaño 0, pues el string final será de un largo impar y no será parte del lenguaje. En caso de un y par se deberá buscar un tamaño de bombeo talque las palabras formadas no sean parte del lenguaje, esto dependerá de L.

4. Pregunta 4

Para este AFD se tienen dos condiciones que el largo sea mayor o igual a nueve y que los substring de tamaño 9 tengan a lo más 5 ceros. Mientras se construye el primer string de largo 9 se puede sobrepasar el límite de los 5 ceros, así que como es necesario para los dos casos se partirá por esta condición.

Tomando solamente la segunda condición se debe cumplir que en los últimos 9 dígitos leídos no haya más de 5 símbolos, por lo tanto cuando se lee un nuevo símbolo se debe olvidar el más antiguo y y verificar la condición con el nuevo ingresado. (Como si fuese una estructura de datos de tipo cola que solamente puede albergar 9 símbolos)

Como es una cola se deben almacenar los 9 dígitos que son de importancia, es una cantidad finita de información por lo tanto es posible para el AFD. Por lo tanto habrá un estado por cada substring w de largo 9 que no rompa con la restricción. Se denominarán q_w. Si se rompe la segunda condición se llegará a un estado sumidero llamado q_s. Por último hace falta tener en cuenta los estados que contabilizan las cadenas de largo menor a 9.

Como los estados son demasiados se definirá el autómata:

$M = (Q, \Sigma, \delta, q_e, F)$

Lo estados son los denominados anteriormente, estados que tienen sub string de largo 9 con a lo más 5 ceros y los que aún no llegan a un largo de 9, además de el sumidero.

$Q = q_w | w \in \{0,1\}^*, |w| \leq 9, \#_0(w) \leq 5 \cup q_s$ La función de transición contempla los casos en los que se cumple la condición de los ceros, la condición del largo y de los ceros y cuando se rompe la condición de los ceros. La letra b representa el dígito del diccionario binario agregado, mientras que las letra d_i representa el dígito con la posición i en el substring.

$$\delta(q_w, b) = \begin{cases} q_{wb} & si \#_0(wb) \leq 5 \quad |w| \leq 9 \\ q_{d_2 \ldots d_9 b} & si \#_0(d_2 \ldots d_9 b) \leq 5 \quad |w| = 9 \\ q_r & si\ no \end{cases}$$

Del sumidero no se puede salir sin importar que se compute

$\delta(q_s, b) = q_s$

Los finales son los estados que tienen al menos un substring de largo 9 y no son el sumidero

$F = \{q_w | q_w \in Q, |w| = 9\}$

5. Pregunta 5

Como L_1 es regular existe un autómata $M = (Q, \Sigma, \delta, q_0, F)$ el cual corresponde a ese lenguaje.

Se busca tener un conjunto $F_/ \subset Q$ tal que desde los estados $q_/ \in F_/$ se puee llegar a cualquier $q_f \in F$ utilizando un string de L_2 mediante δ-

Para construir este conjunto se procederá de la siguiente manera:

Se debe revisar si en los estados q_i pertenecientes a Q existe un string w en L_2 tal que $\delta(q_i, x) \in F$, que es lo que se busca para la división. Si la palabr existe se agrega el estado a $F_/$

Como $F_/$ es un subconjunto de Q, este existe y es finito, por lo tanto se puede construir un autómata $M_/ = (Q, \Sigma, \delta, q_0, F_/)$ el cual acepta solamente a L_1/L_2, pero esto hay que demostrarlo.

Si se tiene $w \in L_1/L_2$ exite un $w' \in L_2$ tal que ww' $\in L_1$. En el autómata M se tiene que $\delta(q_0, w) = q_1 y luego \delta(q, w')/in F$ Por la construcción de $F_/, q \in F_/$, por lo tanto w es reconocido por $M_/$

Si w es reconocido por $M_/$ entonces $\delta(q_0, w) = q \in F_/$. Luego por construcción de $F_/$ hay un string $w' \in L_2$ tal que $\delta(q_1, x) \in F$. Juntando ambas partes se tiene que $\delta/(q_0, ww') \in F$, por ende $ww' \in L_1$. Luego por construcción $w \in L_1/L_2$