TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO SUPERIOR CHICONTEPEC

Tecnológico superior de Chicontepec

Nombre:

Mayra cruz Santiago

Matricula:

1717v0086

Materia:

Autómatas 1

Docente:

Ing. Efrén Flores Cruz

Ingeniería:

Sistemas Computacionales

Tema:

Unidad 3

Grado:

6° semestre

Fecha de entrega:

30 de Marzo de 2020

Unidad 3 Autómatas Finitos

3.1 Definición Formal

Formalmente, un autómata finito es una 5-tupla $(Q, \Sigma, q_0, \delta, F)$ donde:

Q = Es un conjunto finito de estados;

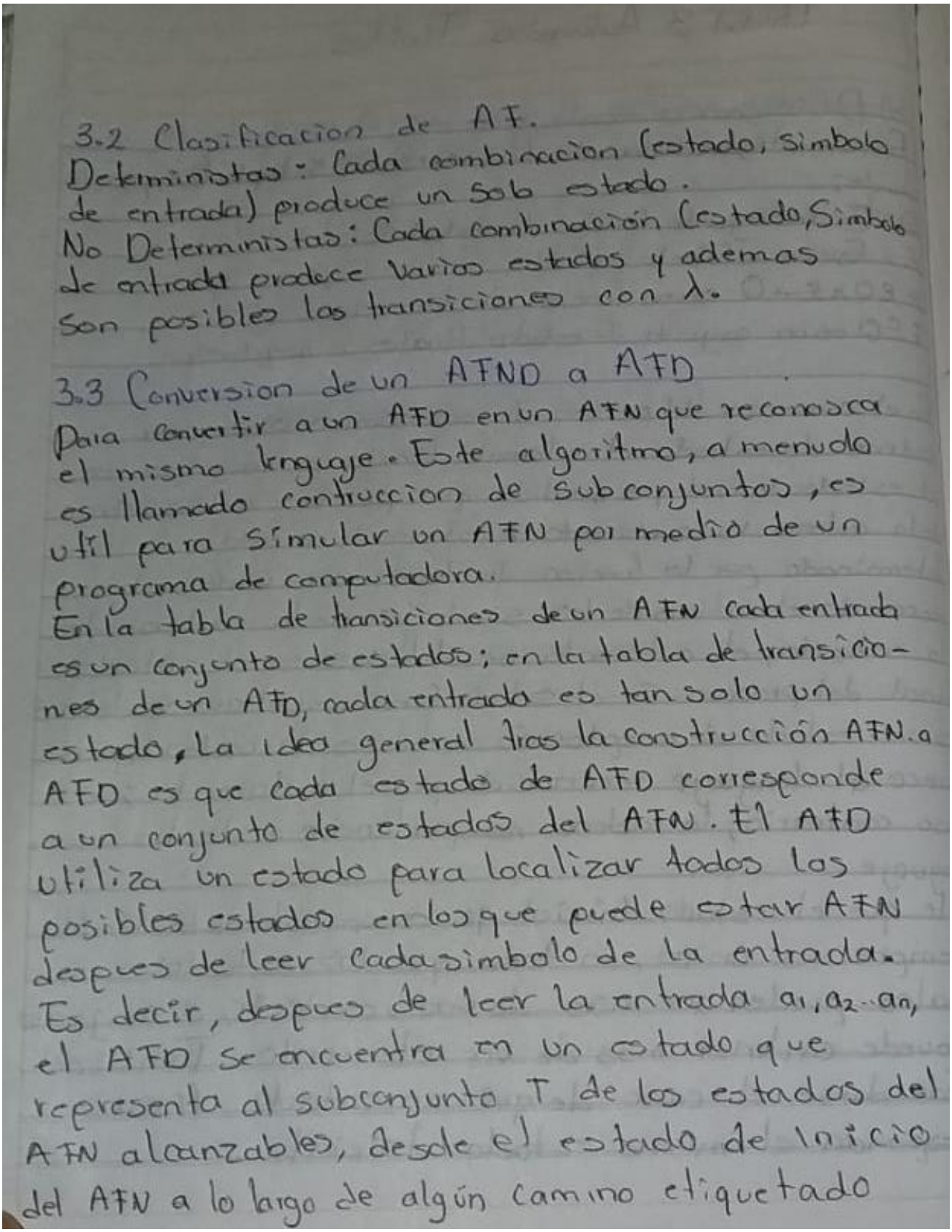
Σ = Es un alfabeto finito

$q_0 \in Q \times \Sigma \to Q$ es una función de transición;

$F \subseteq Q$ es un conjunto de estados finales o aceptación.

En el comienzo del proceso de reconocimiento de una cadena de entrada, el autómata finito se encuentra en el estado inicial y a medida que procesa cada símbolo de la cadena va cambiando de estado de acuerdo a lo determinado por la función de transición. Cuando se ha procesado el último de los símbolos de la cadena de la entrada, el autómata se detiene en el estado final del proceso. Si el estado final en el que se detuvo es un estado de aceptación, entonces la cadena, pertenece al lenguaje reconocido por el autómata; en caso contrario, la cadena no pertenece a dicho lenguaje.

Note que el estado inicial q_0 es un autómata finito siempre es único, en tanto que los estados finales pueden ser más de uno, es decir, el conjunto F puede contener mas de un elemento. También puede darse el caso de que un estado final corresponda al mismo estado inicial.



3.2 Clasificacion de AF.
Deterministas: Cada combinacion (estado, simbolo de entrada) produce un sob estado.
No Deterministas: Cada combinación (estado, Simbolo de entrada produce varios estados y ademas son posibles las transiciones con λ.

3.3 Conversion de un AFND a AFD
Para convertir a un AFD en un AFN que reconozca el mismo lenguaje. Este algoritmo, a menudo es llamado contruccion de subconjuntos, es util para simular un AFN por medio de un programa de computadora.
En la tabla de transiciones de un AFN cada entrada es un conjunto de estados; en la tabla de transiciones de un AFD, cada entrada es tan solo un estado, La idea general tras la construcción AFN.a AFD es que cada estado de AFD corresponde a un conjunto de estados del AFN. El AFD utiliza un estado para localizar todos los posibles estados en los que puede estar AFN despues de leer cada simbolo de la entrada. Es decir, despues de leer la entrada $a_1, a_2 \cdots a_n$, el AFD se encuentra en un estado que representa al subconjunto T de los estados del AFN alcanzables, desde el estado de inicio del AFN a lo largo de algún camino etiquetado

con $a_1, a_2 \ldots a_n$. El numero de estados de AFD puede ser exponencialmente en el numero de estados del AFN pero en la practica este peor caso ocurre raramente.

Algoritmo (Construccion de subconjuntos) Construccion de un AFD a partir de un AFN.

Entrada. Un AFN N

Salida. Un AFD D que acepta el mismo lenguaje.

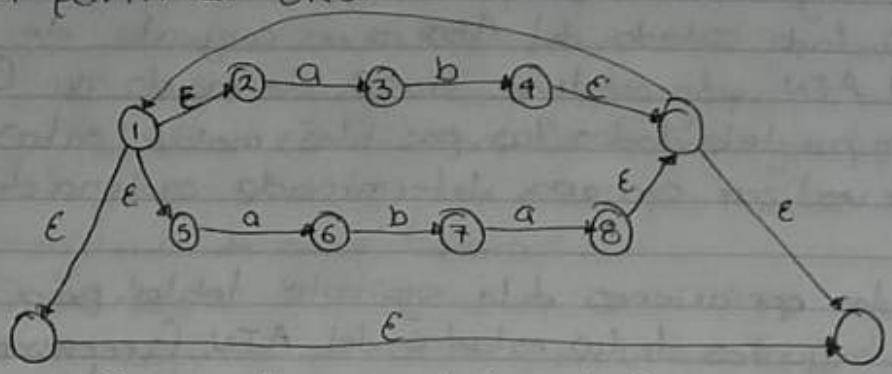
Metodo: El algoritmo construye una tabla de transiciones tranD para D. Cada estado del AFD es un conjunto de estados del AFN y se construye tranD de modo que D simulara "en paralelo" todos los posibles movimientos que N puede realizar con una determinada cadena de entrada.

Se utilizan las operaciones de la siguiente tabla para localizar los conjuntos de los estados del AFN (s representa un estado del AFN, y T un conjunto de estados del AFN)

Operacion	Descripcion.
Cerradura-□ (s)	Conjunto de estados del AFN alcanzables desde el estado s del AFN con transiciones □ solamente
Cerradora-□ (T)	Conjunto de estados del AFN alcanzables desde algun estado s en T con transiciones □ solamente.
Mueve (T, a)	Conjunto de estados del AFN hacia los cuales hay una transición con el simbolo de entrada a desde algun estado s en T del AFN

3.4 Representación de ER Usando AFND

ERs, AFDs, y AFNDs son mecanismos equivalentes para denotar, los lenguajes regulares. En estas 3 secciones demostraremos esto mediante Convertir ER → AFND → AFD → ER. Las dos primeras Conversiones son muy relevantes en la practica, pues permiten constuir verificadores o buscadores eficientes a partir de ERs.

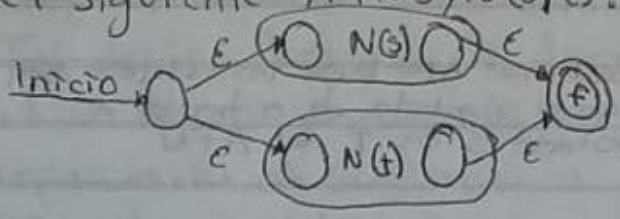


Representacion de la ER

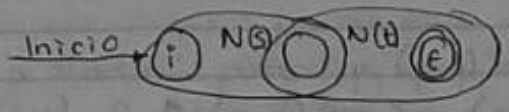
Existen muchas variantes de este algoritmo denominado "Algoritmo de Thompson".

- Supongamos que N(s) y N(t) son AFND's para las expresiones regulares s y t, respectivamente

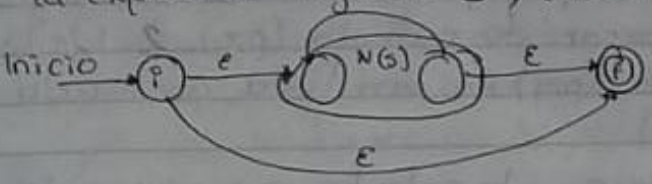
a) para expresion regular s|t (alternancia), constru el siguiente AFND, N(s|t):



.Para la expresion regular st(concatenacion), construir el AFND, N(st)



.Para la expresion regular s*, constuir el AFND, N(s*)



3.5 Minimizacion de estados en un AF

Dos estados de autómata finito determinista son estados equivalentes si al unirse en uno solo estado, pueden reconocer el mismo lenguaje regular que si estuviesen separados. Esta union de estados implica la union tanto de sus transiciones de entrada como de salida. Si dos estados no son equivalentes, se dice que son estados distingibles. Un estado final con un estado no-final nunca seran equivalentes. Un AFD está minimizado, si todos sus estados son distingibles y alcanzables. Un algoritmo de minimización de AFD es el siguiente.

1.- Eliminar los estados inaccesibles es del automata.

2.- Construir una tabla con todos los pares (p,q) de estado restantes.

3.- Marca en la tabla Aquellas entradas donde

un estado es final y el otro es no-final, es decir aquellos pares de estados que son claramente distingibles.

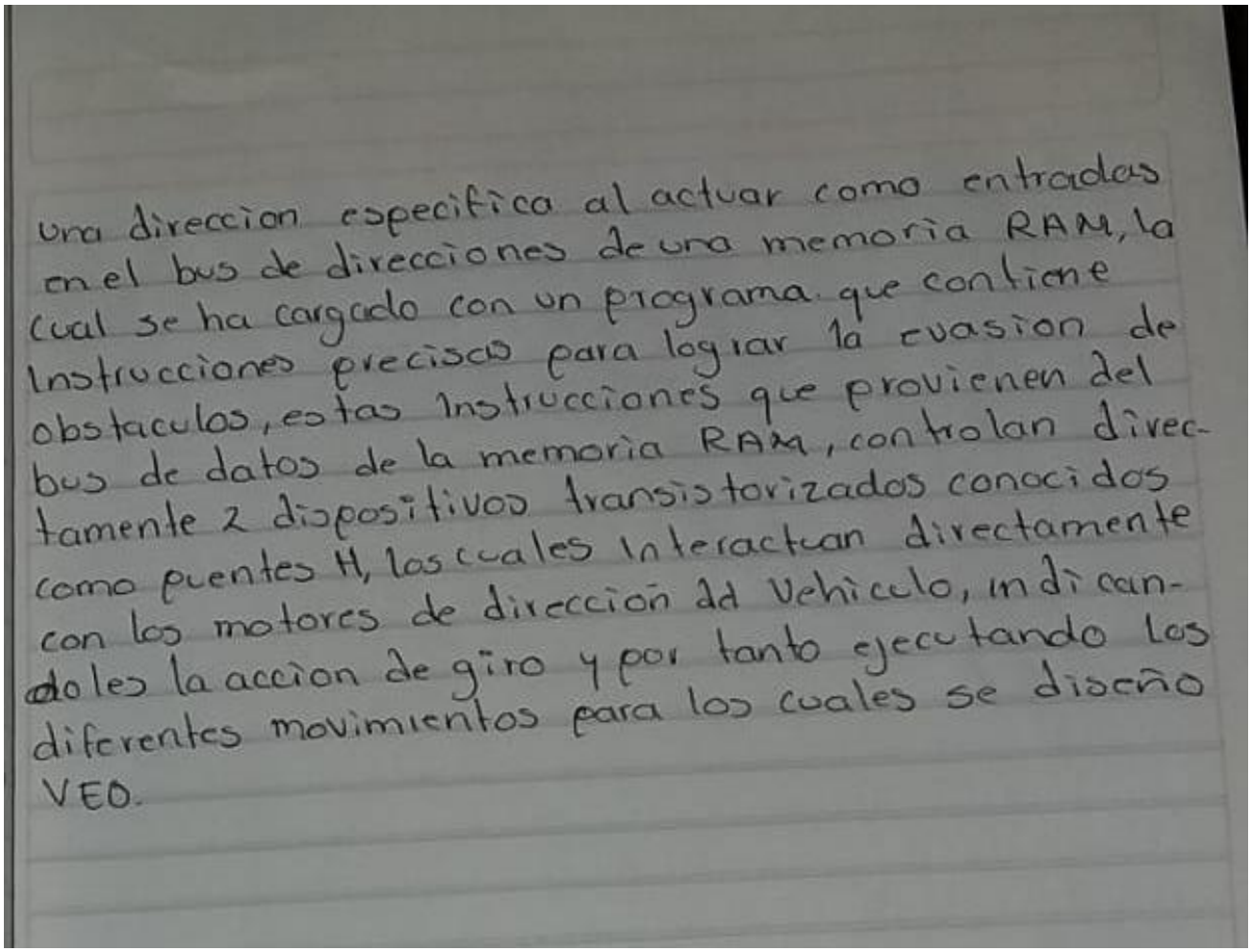
4. Para cada par (p,q) y cada simbolo a del alfabeto tal que r=δ(p,a) y s=δ(q,a): 1. Si (r,s) ya ha sido marcado, entonces p y q tambien son distingibles, por lo tanto marcar la entrada (p,q). 2. De lo contrario, colocar (p,q) en una lista asociada a la entrada (r,s).

5. Agrupar los pares de estados no marcados.

Luego del tercer paso, si la tabla creada, queda completamente marcada, entonces el AFD, inicial ya era minimo. La complejidad computacional del problema de minimizar un AFD es polinomio. Existen algoritmos mas eficientes aún que el mostrado. El problema de minimizar un automata finito no determinista es NP-completo y PSPACE-completo.

3.6 Aplicaciones (Definicion de un caso de estudio)

El Vehiculo Evasor Obstáculos Obtiene Informacion del medio por el cual transita a travez de unos fotodiodos y unas fotorresistencias que actuan como sensores, estos sensores arrojan como resultado niveles de Voltaje que Varian en proporción directa con la proximidad al obstaculo, los niveles de Voltaje despues de pasar por un comparador de niveles se convierten en niveles digitales, los cuales determinan,


una direccion especifica al actuar como entradas en el bus de direcciones de una memoria RAM, la cual se ha cargado con un programa que contiene instrucciones precisas para lograr la evasion de obstaculos, estas instrucciones que provienen del bus de datos de la memoria RAM, controlan directamente 2 dispositivos transistorizados conocidos como puentes H, los cuales interactuan directamente con los motores de direccion del vehiculo, indicandoles la accion de giro y por tanto ejecutando los diferentes movimientos para los cuales se diseño VEO.