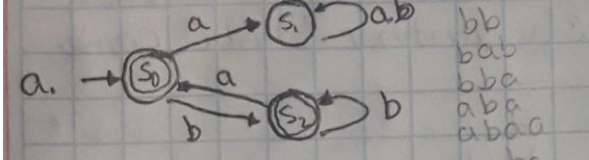
Capitulo 2 Ejercicios

1.- Describe de manera informal los lenguajes aceptados por el siguiente FAS:

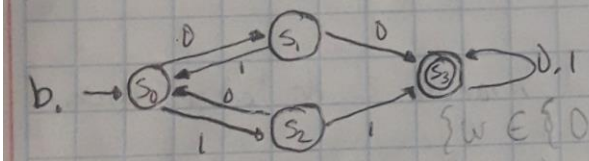
a.



b
bb
bab
bba
aba
abaa
aaba
babba

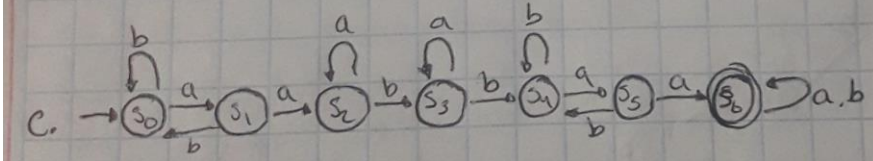
$\{w \in \{a,b\}^* | w$ contiene "abb" como subcadena

b.



$\{w \in \{0,1\}^* | w$ comienza con "0 o 1" y terminen con "0 o 1"

c.



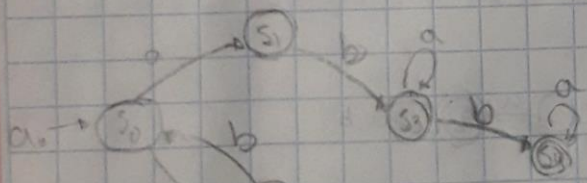
$\{w \in \{a,b\}^* | w$ comienza con "a, b" y contenga "aba" como subcadena

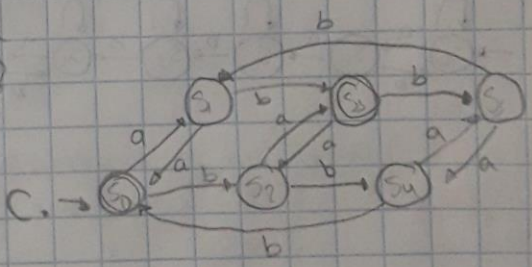
2- Construye un FA aceptando cada uno de los siguientes lenguajes:

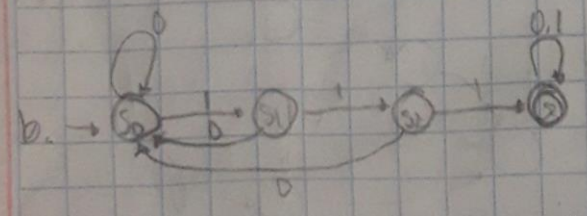
a. {w ∈ {a,b} W comienza con "a" y continue "babá" como segunda subcadena}.

b. {w ∈ {0,1} W contiene "111 como subcadena y no contiene "00" como subcadena}.

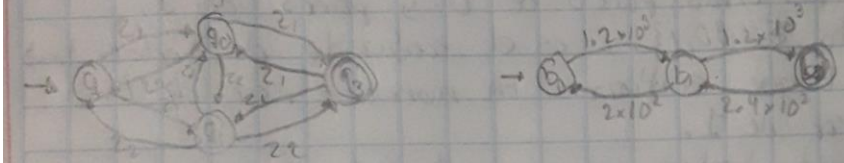
c. {w ∈ {a,b,c} en w el número de modulo 2 de "a" es igual al numero de "b" modulo 3}.







3.- Construye un FAS para reconocer (a) palabras que representen números complejos y (b) palabras que representen números decimales escritos en notación científica.



4.- Diferentes lenguajes de programación usan diferentes notaciones para representar números enteros. Construye una expresión regular para cada uno de los siguientes:

a. Enteros no negativos en C representados en base 10 y 16.
b. Enteros no negativos en Ada que puede incluir abrayado. (el abrayado no puede aparecer como último caracter).

a. (A)+(B+C+D) E* L = (10)+(100*/000*) 10

5. Escribe una expresión regular para cada uno de los sig. lenguajes

a. Dado un alfabeto $\Sigma = \{0, 1\}$, L es el conjunto de todas las cadenas de alternando pares de 0 y pares de 1.

b. Dado un alfabeto $\Sigma = \{0, 1\}$, L es el conjunto de todas las cadenas 0 y 1 que contienen un numero par de 0 o un número par de 1.

c. Dado el alfabeto ingles en minúsculas, la L es el conjunto de todas las cadenas en que las letras aparecen en orden lexicografico ascendente.

d. Dado un alfabeto $\Sigma = \{a, b, c, d\}$, L es el conjunto de cuerdas $xyzwy$, donde xyw son cadenas de uno o mas caracteres en b, y es cualquier un solo caracter en b, y z es el caracter z tomando desde fuera el alfabeto. Cada cadena $xyzwy$ contiene dos palabras xy y wy construida a partir de las letras del b. Las palabras terminan en la misma letra, y estan separadas por z.)

e. Dado un alfabeto $\Sigma = \{+, -, \times, \div, (,), id\}$, L es el conjunto de expresiones algebraicas usando la suma, la resta, la multiplicación, división, y parentesis sobre las identidad.

a. $(0|1)^* 00^* 11^*$

b. $(01)^* 00^* 11^*$

c. $(A)^* B^* C^* D^* \ldots$

d. $(a|b|c|d)(xyzwy + xy)^*$

6.- Escribe una expresión regular para describir cada uno de los siguientes lenguajes de programación que construye:

a. Cualquier secuencia de tablas y espacios en blanco
b. Comentarios en lenguaje de programación C
c. Constantes espacios en blanco (con caracteres de escape)
d. Números de punto flotante

a. [\r\space\t]|(\w\t)\t*

b.

c. (#)*

d. .1|2|...|

7.- Considera las 3 expresiones regulares:

(ab|ac)*

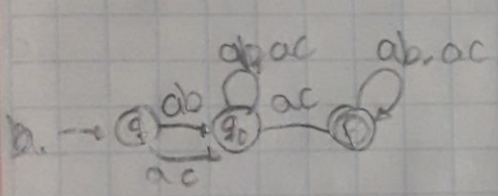
(0|1)* 100 1*

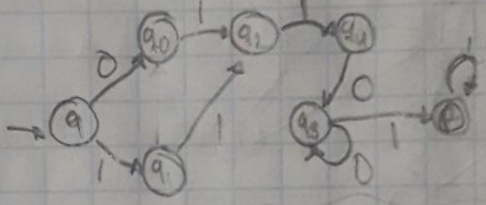
(01|10|00)* 11

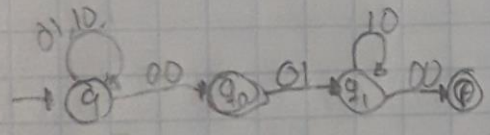
a. Usa la construcción de Thompson para construir un NFA para cada RE

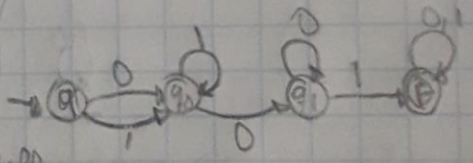
b. Convertir los NFAs a DFAs

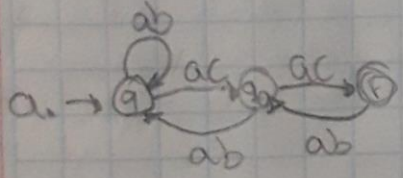
c. Minimizar los DFAs

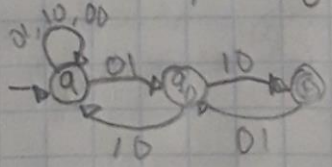












c. {q, F} {q0}

{q, F} {q0} {q1}

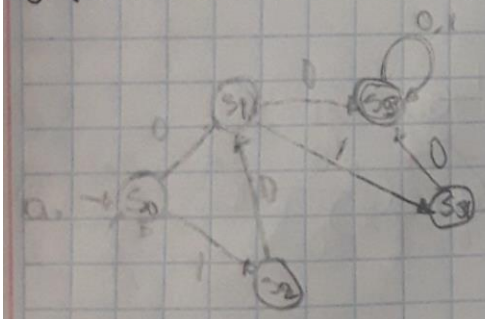
{q} {q0} {F}

8.- Una forma de probar que dos RES son equivalentes es construir sus minimizados DFDS y luego comparalos. Si difieren sólo por el estado de sus nombres, entonces los RES son equivalentes. Use esta técnica para comprobar si son o no equivalentes.

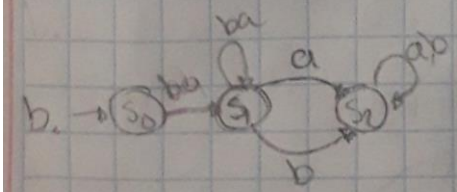
a. $(0 \mid 1)*$ y $(0* \mid 10*)*$

b. $(ba) + (a \vee b* \mid a*)$ y $(ba)* ba + (b* \in)$

a.



b.



9. En algunos casos dos estados conectados por E-move pueden ser combinados.

a. ¿Bajo que conjunto de condiciones pueden dos estados por un E-move ser combinados?,
Si estamos modelando un sistema y no esta claro si el estado actual (despues de procesar una cadena de entrada) debe ser q o q', podemos hacerlo

b. Dar un algoritmo para eliminar E-moves
Ejemplo
$A_1 \to A_2 A_3$
$A_2 \to A_3 A_1 / b$
$A_3 \to A_1 A_1 / a$
$A_3 \to A_1 A_1 / a$
$A_3 \to A_2 A_1 A_3 A_1 / b A_3 A_1$
$A_2 \to a / b A_3 A_1 / a A' / b A_3 A_1 A'$
$A \to A_1 A_2 A_3 / A_2 A_3 A_1 A'$

$A_1 \to A_2 A_3$
$A_2 \to A_3 A_1 /$ segundo
$A_3 \to a / b A_3 A_1 / a A' / b A_3 A_1 A'$
$A \to A_1 A_2 A_3 / A_2 A_3 A_1 A'$

c. ¿Cómo se relaciona su algoritmo con la funcion de cierre utilizada para implementar la construcción del subconjunto?
En que forma así las transacciones a los siguientes estados y al del cierre regresando los cadenas

Capitulo 3 Ejercicios

1.- Escribe una gramática libre de contexto para las sintaxis de las expresiones regulares

Expresión Regular

(a | b) (a | b | 01)

Gramática Libre de Contexto

S → aA | bA
A → aA | bA | 0A | 1A | e

2.- Escribe una gramática libre de contexto para la forma Backus-Naur (bnf) notación para gramaticas sin contexto

Lenguaje tipo Pascal

<sentencia> ::= <var> := <expresion>
<expresion> ::= <expresion> + <termino> | <expresion> - <termino> | <termino>
<termino> ::= <termino> * <factor> | <termino> / <factor> | <factor>
<factor> ::= (<expresion>) | <var> | <num>
<var> ::= A | B | C | D | ... | Z
<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

3.- Cuando se le presunto sobre la definicion de una gramatica libre de contexto ambigua en un examen, dos estudiantes dieron respuestas diferentes. El primero lo definio como "una gramatica donde cada frase tiene un árbol de sintaxis unico por la derivación mas a la izquierda". La segunda lo definio como "una gramatica donde cada frase tiene un árbol de sintaxis unico por cualquier derivación" ¿Cuál de ellas es correcta?

La segunda es la correcta
"Una gramatica donde cada frase tiene un árbol de sintaxis unico por cualquier derivación"

4.- La siguiente gramatica no es adecuada para un analizador predictivo de arriba a abajo. Identifique el problema y corrijalo reescribiendo la gramatica. Demuestre que tu nueva gramatica satisface la condición LL(1).

L → R a
| Q ba

R → aba
| caba
| R bc

Q → bbc
| bc

5.- Considere la siguiente gramatica:

$A \rightarrow Ba \mid Ac$

$B \rightarrow dab$

$C \rightarrow cB \mid Cb$

¿Esta gramatica satisface la condicion LL(1)? Justifique su respuesta. Si no lo hace, reescribalo como una gramatica LL(1) para el mismo idioma.

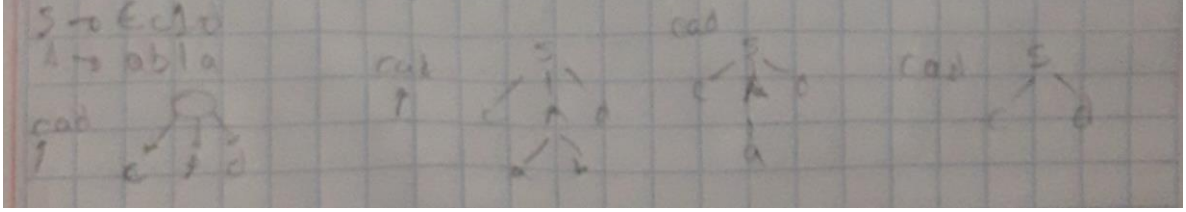
La gramatica G = (T, V, P, S) cuyo lenguaje generado L(G) puede ser analizado por un analizador sintactico descendente recursivo

6.- Las gramaticas que pueden ser analizadas de arriba a abajo, en un escaneo lineal de izquierda a derecha, con una letra K se llama gramáticas LL(K). En el texto, la condición LL(1) se describe en terminos de primeros conjuntos.

¿Como definiria los primeros conjuntos necesarios para describir una condicion LL(K)?

S → ECD

A → abBa



7.- Supongamos que un ascensor está controlado por dos comandos: ↑ para subir un piso y ↓ para bajar. Supongamos que el edificio es arbitrariamente alto y que el ascensor comienza en el piso x.

Escriba una gramática LL(1) que genere secuencias de comandos arbitrarias que (1) nunca causen que el ascensor baje del piso x y (2) siempre devuelvan al ascensor al piso x al final de la secuencia. Por ejemplo, ↑↑↓↓ y ↑↓↑↓ son secuencias de comandos válidas, pero ↑↓↓↑ y ↑↓ no lo son. Por comodidad puede considerar una secuencia nula como válida.

Prueba que su gramática es LL(1).

8.- Los analizadores de arriba y de abajo construyen árboles de sintaxis en diferentes órdenes. Escribe un par de programas TopDown y BottomUp, que toman un árbol de sintaxis e imprimen los nodos en orden de construcción.

TopDown debe mostrar el orden para un parser top-down,
BottomUp debe mostrar el orden para un parser bottom-up

```
TopDown
int elToken;
int yylex(){
    return(getchar());
}
void terminal(int t){
    if(elToken==t)
        elToken=yylex();
    else
        errorSintactico();
}
void L(){
    terminal((int)'a');
    if(elToken==(int)'b')
        terminal((int)'b');
}
```