Teoría de Autómatas y Lenguajes Formales

v2.4

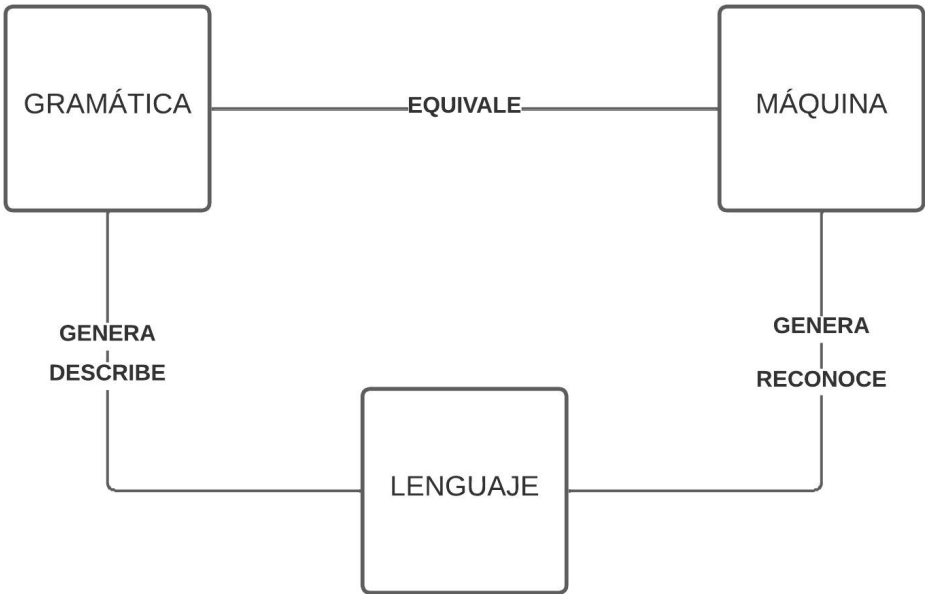
Andrés Lillo Ortiz
Universidad de Santiago de Compostela
2023/2024

Índice

Nota: Estos apuntes son sacados de las diapositivas y boletines de la asignatura. La referencia bibliográfica es el libro Teoría de autómatas, lenguajes y computación de John E. Hopcroft, Rajeev Motwani y Jeffrey D. Ullman de la editorial Pearson, Addison Wesley.

Tema 1: Introducción

Lenguajes, Gramáticas y Autómatas



GRAMÁTICA	LENGUAJE	MÁQUINA / COMPLEJIDAD
Tipo 0: Sin Restricciones	Recursivamente Enumerable	Máquina de Turing / Indecidible
Tipo 1: Sensible al Contexto	Sensible al Contexto	Autómata Linealmente Acotado / Exponencial
Tipo 2: Independiente del Conteto	Independiente del Contexto	Autómata con Pila / Polinómica
Tipo 3: Regular	Regular	Autómata Finito / Lineal

Tema 2: Autómatas Finitos

Ejemplos y ejercicios sobre Autómatas Finitos en el apartado Problemas Tema 2. Los autómatas de número de estados finitos son máquinas secuenciales que reconocen lenguajes regulares.

Clasificación

1. Determinista (AFD): el autómata no puede estar en más de un estado simultáneamente
2. No determinista (AFN): puede estar en varios estados al mismo tiempo

Autómatas Finitos Deterministas

Para cada estado hay un único estado al que el autómata puede transicionar partiendo del actual. Ejemplos en el Boletín 1: Diseño de Autómatas de Estados Finitos.

Definición Formal

A = (Q, S, δ, q_0, F)

1. Estados, Q
2. Conjunto finito de símbolos de entrada, S
3. Función de transición (δ) que, dados un estado y una entrada, devuelve un estado. δ(q, a) = p
4. Estado inicial (uno de los estados de Q), q_0
5. Estados finales o de aceptación (subconjunto de Q), F

Representación

Grafo

1. Un nodo por cada estado de Q
2. Un arco de q a p etiquetado con a para cada δ(q, a) = p
3. Una flecha dirigida al estado inicial
4. Los estados finales están marcados por un doble círculo

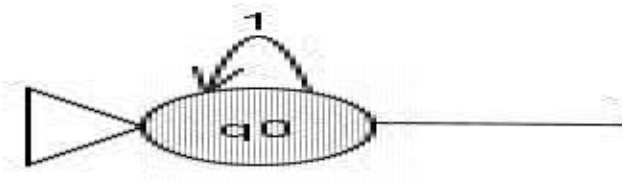


Tabla de Transiciones

1. Filas: estados

2. Columnas: entradas
3. Estado inicial: indicado por una flecha
4. Estados finales: indicado por: *

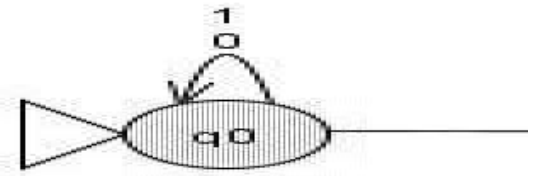
	0	1
$\rightarrow q_0$	q_2	q_0
$^* q_1$	q_1	q_1
q_2	q_2	q_1

Autómatas Finitos No Deterministas

Tienen la capacidad de estar en varios estados simultáneamente. En la representación formal, δ devuelve un conjunto de estados en lugar de solo uno. Esto produce que se creen varios hilos que se ejecutan simultáneamente. Si una transición no está definida (Θ), el hilo termina. Puede haber transiciones con la cadena vacía (Ɛ o λ), es decir, si un estado está unido a otro por Ɛ, el autómata estará en los dos estados a la vez. Siempre es posible convertir un AFN a un AFD equivalente. Ejemplos en el Boletín 1: Diseño de Autómatas de Estados Finitos.

Representación

Grafo



AFD equivalente:

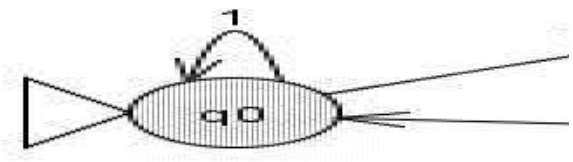


Tabla de Transiciones

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	Θ	$\{q_2\}$
$* q_2$	Θ	Θ

Clausuras respecto de Ɛ

Clausura del estado q respecto de Ɛ, CLAUSε(q):

1. Base: el estado q está en CLAUSε(q)
2. Paso inductivo: si δ es la función de transición del AFN-Ɛ, y el estado p está en CLAUSε(q), entonces CLAUSε(q) contiene todos los estados de δ(p, Ɛ)

Equivalencia entre AFD y AFN

Algoritmo para pasar de un AFN a un AFD equivalente:

1. Calcular todas las clausuras respecto de Ɛ
2. Definir el estado inicial como la clausura respecto de Ɛ del estado inicial del AFN
3. Calcular todas las funciones de transición para todos los nuevos estados empezando por el nuevo estado inicial. Utilizar las clausuras respecto de Ɛ de cada estado en lugar de usar solamente el estado. Así cada vez que una función de transición dé como resultado un conjunto de estados del AFN, corresponderá con un nuevo estado del AFD equivalente
4. Los estados finales serán los que contienen al estado final del AFN

Ejemplos en el Boletín 3: Minimización de AF y Expresiones Regulares.

Minimzación de un AFD

Algoritmo para minimizar un AFD:

1. Separar estados finales y no finales
2. Para cada estado comprobamos con cada entrada a qué conjunto transicionaría. Para estar en el mismo conjunto todos deben transicionar al mismo conjunto para cada entrada
3. Parar si no se separan más conjuntos

Ejemplos en el Boletín 3: Minimización de AF y Expresiones Regulares.

Tema 3: Lenguajes Regulares

Ejemplos y ejercicios sobre Lenguajes Regulares en el apartado Problemas Tema 3. Las Expresiones Regulares (ER) representan Lenguajes Regulares (LR).

Operadores de las ER por orden de precedencia

1. Los paréntesis modifican las reglas de preferencia
2. Clausura (L*): conjunto de cadenas formado por la concatenación de cualquier número de cadenas de L
3. Concatenación (L · M o LM): conjunto de cadenas formadas por la concatenación de una cadena de L y otra de M
4. Unión (L + M): conjunto de cadenas que pertenecen a L, a M o a ambos

Álgebra de las ER

1. Propiedad conmutativa de la unión: L + M = M + L
2. Propiedad asociativa de la unión: (L + M) + N = L + (M + N)
3. Propiedad asociativa de la concatenación: (LM)N = L(MN)
4. La concatenación no es conmutativa: L · M ≠ M · L
5. Θ es el elemento identidad de la unión: Θ + L = L + Θ = L
6. Ɛ es el elemento identidad de la concatenación: Ɛ · L = L · Ɛ = L
7. Θ es el elemento nulo de la concatenación: Θ · L = L · Θ = Θ
8. Propiedad distributiva por la izquierda de la concatenación respecto de la unión: L · (M + N) = L · M + L · N
9. Propiedad distributiva por la derecha de la concatenación respecto de la unión: (M + N) · L = M · L + N · L
10. Propiedad de idempotencia de la unión: L + L = L

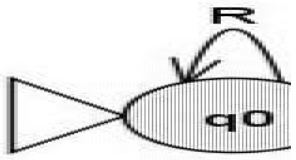
Construcción de ER

1. Base:
    a. Las constantes Ɛ y Θ son ER de los lenguajes L(Ɛ) = {Ɛ} y L(Θ) = {Θ}
    b. Si a es un símbolo, a es la ER del lenguaje L(a) = {a}
2. Paso inductivo:
    a. si E y F son ER, E + F es una ER, y L(E + F) = L(E) + L(F)
    b. si E y F son ER, EF es una ER, y L(E · F) = L(E) · L(F)
    c. si E es ER, E* es una ER, y L(E*) = (L(E))*

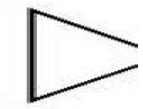
Conversión de Autómatas Finitos a ER

Las ER definen los LR exactamente igual que los Autómatas Finitos. Para pasar de un Autómata Finito a una ER equivalente se usa la eliminación de estados, es decir, se eliminan estados y se sustituyen por arcos con expresiones regulares. Al eliminar todos los estados menos el inicial y el final se siguen las siguientes reglas:

1. L = (R* + SU*T)*SU*



2. L = R*

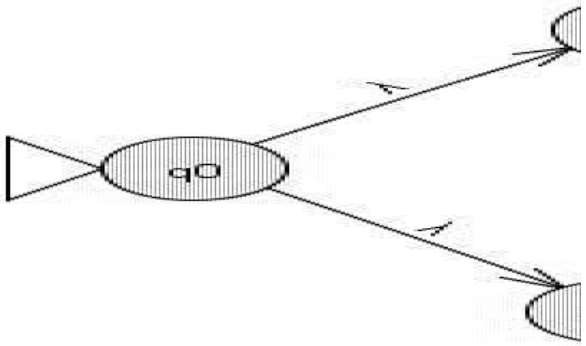


Ejemplos en el Boletín 3: Minimización de AF y Expresiones Regulares.

Conversión de ER a Autómatas Finitos

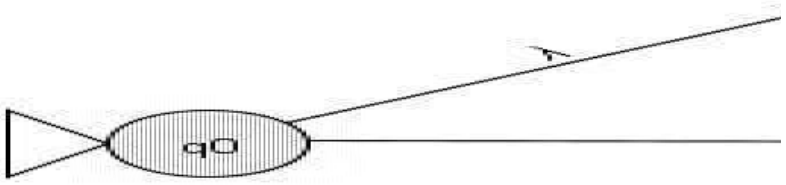
Se siguen las siguientes reglas:

1. R + S: L(R) + L(S)



2. RS: L(R)L(S)

3. R*: L(R*)



Ejemplos en el Boletín 3: Minimización de AF y Expresiones Regulares.

Lema del Bombeo para LR

Para un lenguaje regular infinito, el cumplimiento del lema del bombeo (LB) es una condición necesaria, pero no suficiente. Indica que hay una parte de cada palabra del lenguaje que se puede bombear (repetir las veces que se quiera) y seguirá perteneciendo al lenguaje.

Teorema

Sea L un lenguaje regular. Entonces existe una constante n (que depende de L), tal que, para toda cadena w ∈ L, con |w| ≥ n, podemos dividir w en tres cadenas, w = xyz, de modo que:

1. y ≠ Ɛ
2. |xy| ≤ n
3. para todo k≥0, la cadena xy^kz ∈ L

Tema 4: Gramáticas Independientes del Contexto

Ejemplos y ejercicios sobre Gramáticas Independientes del Contexto en el apartado Problemas Tema 4.

Definición Formal de una Gramática

G = (V, T, P, S). Las gramáticas están formadas por cuatro componentes:

1. El conjunto finito de símbolos no terminales (V o NT) o variables
2. El alfabeto de símbolos terminales (T)
3. Un conjunto finito de producciones o reglas (P), que indican las transformaciones posibles
4. El símbolo inicial o axioma (S)

Las reglas de la gramática están formadas por:

1. Una variable, cabeza de la producción
2. El símbolo de producción →
3. Una cadena de cero o más símbolos terminales y no terminales, que son el cuerpo de la producción

Clasificación de Gramáticas

Tipo 0: Sin Restricciones

$x \rightarrow y$

$x \in (NT|T)^{+}$

$y \in (NT|T)^{*}$

Tipo 1: Sensible al Contexto

$\alpha \rightarrow \beta;\ |\alpha| \leq |\beta|$

$\alpha = z_1 x z_2$

$\beta = z_1 y z_2$

$z_1,\ z_2 \in T^{*}$

$x \in NT$

$y \in (NT|T)^{+}$

Tipo 2: Independiente del Contexto

$x \rightarrow y$

$x \in NT$

$y \in (NT|T)^{*}$

Tipo 3: Regular

$\alpha \rightarrow \beta$

$\alpha \in NT$

$\beta \in \{aB,\ Ba,\ b\}$

$B \in NT$

$a \in T^{+}$

$b \in T^{*}$

Lenguaje de una Gramática

Si G = (V, T, P, S), el lenguaje de G será: L(G) = {w que están en T *|S⇒*w}.

Gramáticas Regulares

Una gramática G = (V, T, P, S) es lineal por la derecha si todas sus producciones son de la forma:

1. A → xB
2. A → x

Una gramática G = (V, T, P, S) es lineal por la izquierda si todas sus producciones son de la forma:

1. A → Bx
2. A → x

Donde A y B pertenecen a V y x pertenece a T *

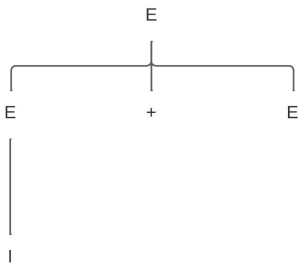
Árboles de Derivación

Una derivación de una sentencia ω (palabra que pertenece al lenguaje) es la secuencia de sustituciones de no terminales que, partiendo del símbolo inicial S, produce como resultado ω. El árbol de derivación para G tendrá las siguientes características:

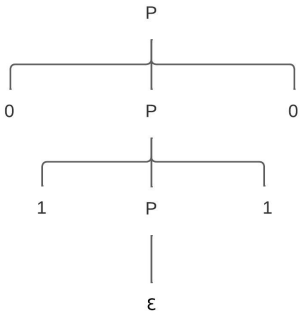
1. Cada nodo interior está etiquetado con una variable
2. Cada hoja está etiquetada con una variable, un terminal o Ɛ. Si es Ɛ, tiene que ser el único hijo de su nodo progenitor
3. Si un nodo interior está etiquetado con A y sus hijos están etiquetados con X1, X2 , ..., Xk (de izquierda a derecha), entonces A → X1X2...Xk es una producción de P.

Ejemplos

1. E ⇒ I + E



2. P ⇒ 0110



Ambigüedad

Una GIC G = (V, T, P, S) es ambigua si existe al menos una cadena w en T* para la que podemos encontrar dos árboles de derivación distintos con la raíz etiquetada con S y cuyo resultado es w.

Ejemplos de GIC

Lenguaje formado por palabras palíndromas sobre el alfabeto {0, 1}:

S → 0S0 | 1S1 | 0 | 1 | Ɛ

L = $\{0^n1^n \mid n \geq 0\}$:

S → 0S1 | Ɛ

Ejemplos en el Boletín 4: Diseño de Gramáticas Independientes del Contexto (GIC).

Formas normales para GIC

Las gramáticas en formas normales reducen la complejidad para la obtención de las derivaciones. Existen, entre otras, la Forma Normal de Chomsky (FNC) y la Forma Normal de Greibach (FNG). Para llegar a una forma normal se siguen los siguientes pasos:

1. Eliminar producciones con la cadena vacía Ɛ: se sustituyen las producciones con la cadena vacía Ɛ, que solo podrá aparecer en S → Ɛ
2. Eliminar producciones unitarias: se sustituyen las producciones del tipo A → B donde A, B ∈ NT
3. Eliminar símbolos inútiles:
    a. Eliminar símbolos no generadores: un símbolo es generador si se puede transformar en un símbolo terminal
    b. Eliminar símbolos no alcanzables: un símbolo es alcanzable si se puede llegar a él partiendo de S
4. Conversión a FNC o a FNG

Ejemplos en el Boletín 5: Diseño de GIC y Transformación a FNC.

Forma Normal de Chomsky

Una cadena de longitud n se analiza en 2n-1 pasos. El árbol de derivación es binario y su profundidad máxima es n. Las producciones son de la forma A → BC donde A, B, C ∈ NT, de la forma A → a donde A ∈ NT y a ∈ T o de la forma S → Ɛ.

Forma Normal de Greibach

Una cadena de longitud n tiene una derivación de n pasos. Un analizador sintáctico descendente parará a profundidad n. Las producciones son de la forma A → aα donde a ∈ T y α es una cadena de cero o más variables.

Tema 5: Autómatas con Pila

Ejemplos y ejercicios sobre Autómatas con Pila en el apartado Problemas Tema 5. Un autómata con pila (AP) es un AFN con transiciones Ɛ y con una pila en la que se puede almacenar una cadena de símbolos de pila. El AP puede recordar una cantidad infinita de información. Reconoce Lenguajes Independientes del Contexto.

Definición Formal

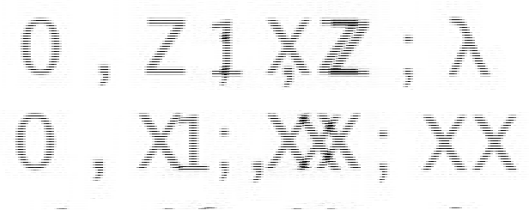
P = (Q, Σ, Γ, δ, q_0,Z_0, F)

1. Q: conjunto finito de estados
2. Σ: conjunto finito de símbolos de entrada
3. Γ: alfabeto de pila finito
4. δ: función de transición, δ(q, a, X) = (p, XX)
5. q_0: estado inicial
6. Z_0: símbolo inicial de la pila
7. F: conjunto de estados de aceptación

Representación

Grafo

Las transiciones indican la entrada, lo que hay en la cima de la pila (se saca de la pila) y qué se añade a la pila.



Ejemplos en el Boletín 6: Diseño de Autómatas con Pila.

Tabla de Transiciones

Las columnas indican el estado actual, la entrada, lo que hay en la cima de la pila (se saca de la pila) y el movimiento (estado siguiente y qué se añade a la pila).

Q	Σ	Γ	Movimiento
q	0	Z_0	(q, xZ_0)
q	0	x	(q, xx)
q	1	x	(q, x)

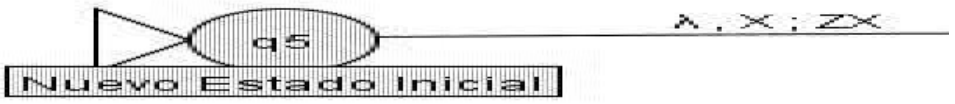
q	Ɛ	x	(p, Ɛ)
p	Ɛ	x	(p, Ɛ)
p	1	x	(p, xx)
p	1	Z_0	(p, Ɛ)

Tipos de Aceptación

1. Aceptación por Estado Final (APF): el autómata acaba en un estado final al terminar la cadena
2. Aceptación por Vaciado de Pila (APN): la pila acaba vacía después de terminar la cadena

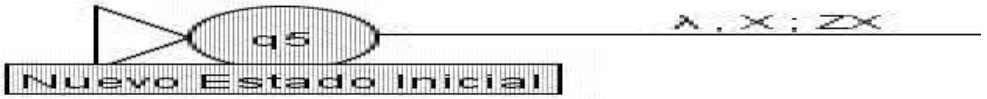
Conversión Vaciado de Pila a Estado Final

1. Se añade un nuevo símbolo inicial de pila X_0
2. En todos los estados se añade una nueva transición Ɛ, X_0; Ɛ (se cumple cuando la pila está vacía, es decir, se acepta) que lleve a un estado final



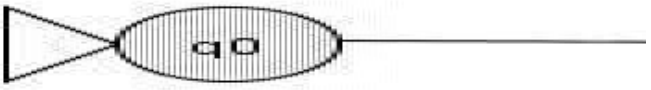
Conversión Estado Final a Vaciado de Pila

1. Se añade un nuevo símbolo inicial de pila X_0: si el APF vacía su pila en un estado no final no debería reconocer la secuencia. Si no se añadiese el nuevo símbolo inicial de pila, el APN pasaría a reconocer en esas situaciones
2. En todos los estados finales se añade una nueva transición Ɛ, Λ; Ɛ (da igual lo que haya en la pila, se vacía) que lleve a un estado no final con esta misma transición a sí mismo



Conversión de GIC a AP

Obtener el AP que reconozca por vaciado de pila la siguiente gramática: I → a | b | Ia | Ib | I0 | I1, E → I | E*E | E+E | (E). Se añaden los símbolos terminales y las producciones:



Autómatas con Pila Deterministas

Los Autómatas con Pila Deterministas (APD) aceptan un conjunto de lenguajes a medio camino entre los Lenguajes Regulares y los Lenguajes Independientes de Contexto. Un AP P = (Q, Σ, Γ, δ, q_0,Z_0, F) es determinista si:

1. δ(q, a, X) tiene como máximo un elemento para cualquier q en Q, a en Σ o a = Ɛ, y X en Γ
2. δ(q, a, X) no está vacío para algún a en Σ y δ(q, Ɛ, X) debe estar vacío

Lema del Bombeo para LIC

Para un Lenguaje Independiente del Contexto (LIC), el cumplimiento del lema de bombeo es una condición necesaria, pero no suficiente.

Teorema

Sea L un LIC. Entonces existe una constante n tal que si z es cualquier cadena de L de longitud |z| ≥ n, podemos escribir z = uvwxy, con las siguientes condiciones:

1. |vwx| ≤ n
2. vx ≠ Ɛ
3. Para todo k >= 0, $uv^kwx^ky \in L$

Tema 6: Máquinas de Turing

Una Máquina de Turing (MT) es un autómata que cuenta con un dispositivo de almacenamiento denominado cinta. Asociada con la cinta, existe una cabeza de lectura/escritura. La entrada está escrita en la cinta al comienzo y la salida se escribirá en la cinta durante la operación de la MT. La MT finaliza el procesamiento cuando llega a un estado final o no tiene transiciones definidas para la combinación de estado y entrada. En una MT no es necesario leer todo el contenido de la cinta para aceptar, pero los AF y AP sí requieren leer toda la entrada.

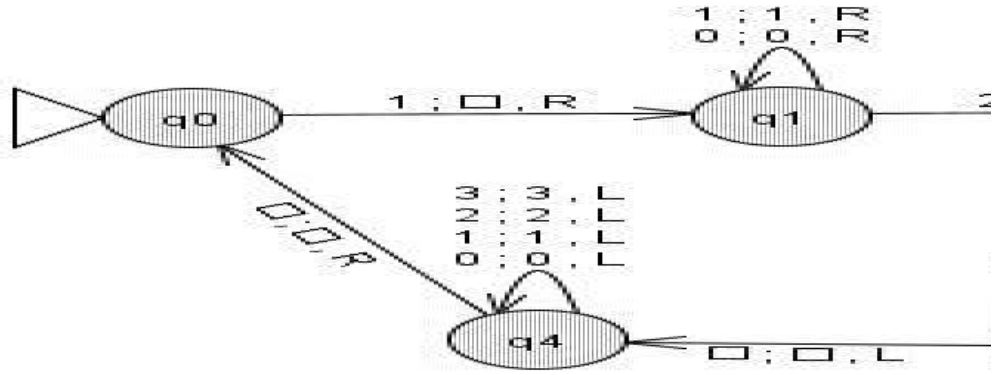
Definición Formal

M = (Q, Σ, Γ, δ, q_0, B, F)

1. Q: conjunto de estados
2. Σ: alfabeto de entrada
3. Γ: alfabeto de la cinta
4. δ: función de transición. Está formada por el nuevo estado al que transiciona, el símbolo a escribir y el movimiento a la derecha o a la izquierda en una posición. Ejemplo: δ(q_1, a) = (q_5, b, R). Pasa del estado q_1 leyendo una a al estado q_5 escribiendo una b y moviéndose una posición a la derecha.
5. q_0 ∈ Q : estado inicial
6. B ∈ Γ : espacio en blanco (B ∉ Σ)
7. F ⊆ Q: conjunto de estados finales

Representación

Los arcos representan el símbolo que se lee, el que se escribe en su lugar y la dirección del movimiento. Ejemplo: MT que reconozca expresiones del siguiente lenguaje: L = {$1^n2^n3^k$ I n > k}



Ejemplos en el [Boletín 7: Diseño de Máquinas de Turing Estándar].

Computación de Funciones

Una función f con dominio D es Turing-computable o computable sin más, si existe una MT M = (Q, Σ, Γ, δ, q_0, B, F) tal que: q_0w |-* q_ff(w), q_f ∈ F, para todo w ∈ D, es decir, a partir de q_0 con una entrada w y después de varias transiciones (|-*) llegue a q_f con la salida f(w).

Ejemplos:

1. Comparador:
   a. $q_{C,0}$w(x)0w(y) |-* $q_{A,0}$w(x)0w(y) si x ≥ y
   b. $q_{C,0}$w(x)0w(y) |-* $q_{E,0}$w(x)0w(y) si x < y
2. Sumador:
   a. $q_{A,0}$w(x)0w(y) |-* $q_{A,f}$w(x+y)
3. Borrador:
   a. $q_{E,0}$w(x)0w(y) |-* $q_{E,f}$0
4. f(x, y) = x + y si x ≥ y; f(x, y) = 0 si x < y:
   a. combinación de comparador, sumador y borrador

Variaciones de la MT

Ninguna de las variaciones añade nueva funcionalidad a la MT estándar pero pueden añadir o restar complejidad.

MT con Opción de No Movimiento

A las opciones R (right) y L (left) que indican el movimiento en la función de transición se añade S (stay) que indica que la cabeza se mantiene estática.

Simulación de una MT con opción de no movimiento (δ) con una MT estándar (δ')

1. Por cada δ(q_i, a) = (q_j, b, I o D), se incluye δ'(q_i, a) = (q_j, b, I o D)
2. Por cada δ(q_i, a) = (q_j, b, E), se incluyen δ'(q_i, a) = (q_{jS}, b, D) y δ'(q_{jS}, c) = (q_j, c, I) (una transición por cada c ∈ Γ)

MT con Cinta Semiinfinita

MT cuya cinta está limitada por un extremo.

Simulación de una MT estándar M por medio de una MT con cinta semiinfinita P

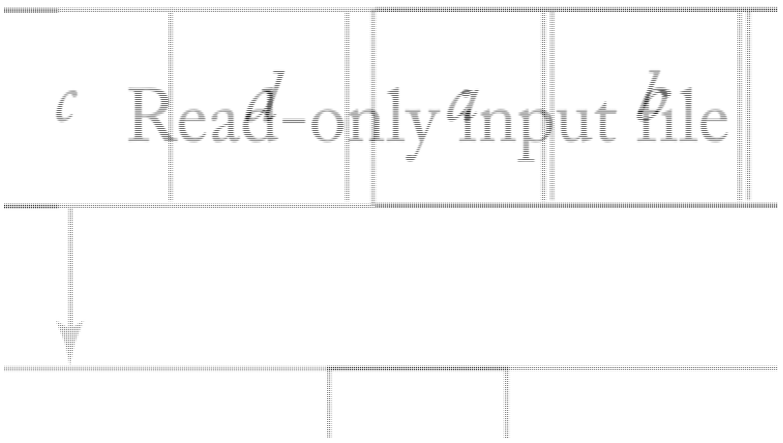
1. Pista superior: contenido de la cinta de M a la derecha de la referencia (situación inicial de la cabeza de M)
2. Pista inferior: contenido de la cinta de M a la izquierda de la referencia y en orden inverso

Ejemplo: $\delta(q_i, a) = (q_j, c, I)$ es simulado con

1. $\delta'(q'_i, (a, b)) = (q'_j, (c, b), I)$
2. $\delta'(q'_j, (\#, \#)) = (p'_j, (\#, \#), D)$

MT con Cinta de Entrada

La entrada está escrita en una cinta de sólo lectura. Las transiciones se realizan en función del estado, el símbolo leído de la entrada y el símbolo leído por la cabeza de lectura/escritura en la cinta.



Simulación de una MT estándar con una MT con cinta de entrada

Copiar el contenido de la cinta de entrada en la cinta.

Simulación de una MT con cinta de entrada M1 con una MT estándar M2

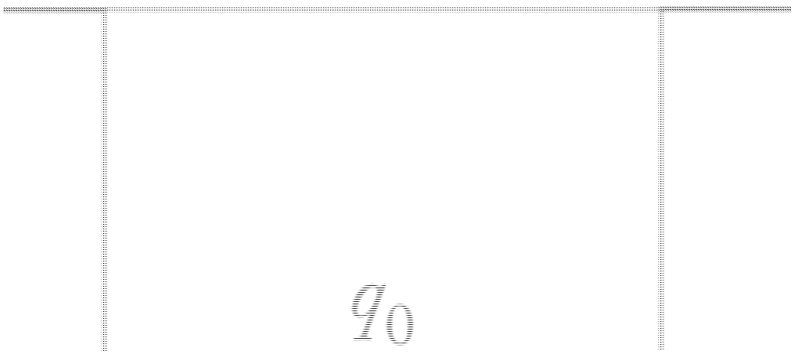
Cinta con 4 pistas: valores de entrada, posición de la cabeza de lectura, contenido de la cinta y posición de la cabeza de lectura/escritura.

La simulación requiere varios movimientos en M2 por cada movimiento en M1:

1. Posición de partida: extremo izquierdo de la cinta
2. Búsqueda de la posición de la cabeza de lectura en la pista 2
3. Lectura del símbolo correspondiente en la pista 1 y transición de estado
4. Búsqueda de la posición de la cabeza de lectura/escritura en la pista 4
5. Lectura del símbolo correspondiente en la pista 3 y transición de estado
6. Modificación de las pistas para representar el movimiento en M1
7. Vuelta a la posición de partida para simular el siguiente movimiento

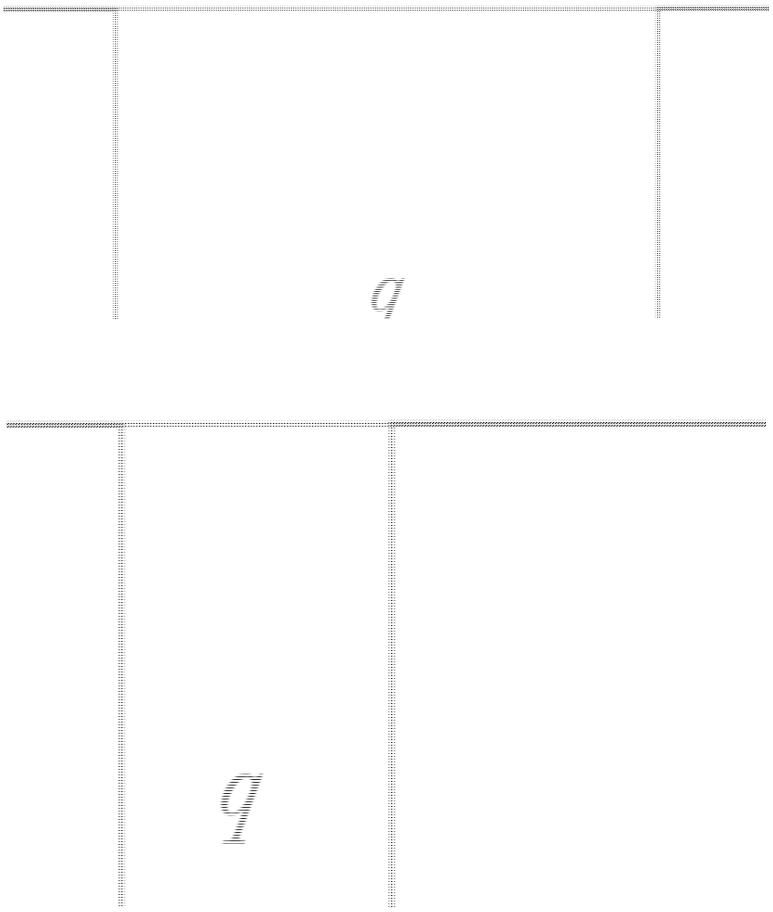
MT Multicinta

La MT trabaja con varias cintas al mismo tiempo. Ejemplo: δ(q_0, a, e) = (q_1, x, y, I, D)



Simulación de una MT multicinta con una MT estándar

1. Se necesitan 2n pistas (n es el número de cintas de la MT multicinta):
2. Pistas impares: representan el contenido de las cintas
3. Pistas pares: representan la posición de la cabeza en las cintas
4. Los pasos a ejecutar son similares al de la simulación de MT con cinta de entrada

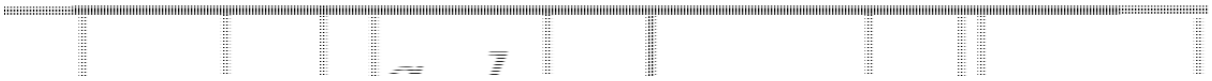


MT Multidimensional

La cinta es infinita en más de una dimensión.

Simulación de una MT bidimensional con una MT estándar

1. Pista 1: almacena el contenido de la cinta bidimensional
2. Pista 2: contiene las direcciones asociadas al contenido de la pista 1
3. Para simular un movimiento, se busca en la pista 2 la dirección de la celda a la que se debe desplazar la cabeza de lectura/escritura



MT No Determinista

Por cada posible transición la MT se replica a sí misma y sigue caminos distintos. Ejemplo: $\delta(q_0, a) = \{(q_1, b, D), (q_2, c, I)\}$.

Simulación de una MT no determinista con una MT estándar

1. Se crea una cinta con 2n pistas, donde n es el número de máquinas a simular
2. Cada nueva MT implica la inicialización de dos nuevas pistas
3. Una pista representa el contenido de la cinta y la otra el estado de la MT

MT Universal

Son reprogramables. Dada una descripción de cualquier MT M y una cadena w, una MTU puede simular la computación de M para w. Una MTU tiene tres cintas:

1. Se examina el contenido de las cintas 2 y 3: configuración de M
2. Se consulta la cinta 1 para determinar la transición a realizar
3. Se modifican las cintas 2 y 3 como resultado del movimiento realizado

Codificación de una MT

1. q_1 = 1 q_2 = 11, …
2. a_1 = 1 a_2 = 11, …
3. I = 1, D = 11

4. 0 es el símbolo separador
5. Ejemplo: δ(q_1, a_2) = (q_2, a_3, I) se codifica como 10110110111010

Lenguajes Sensibles al Contexto

Ejemplo: GSC que genera L = {$a^n b^n c^n$: n ≥ 1}

S → abc | aAbc
Ab → bA
Ac → Bbcc
bB → Bb
aB → aa | aaA

Ejemplos en el Boletín 9: Diseño de Gramáticas Sensibles al Contexto (GSC) y de Gramáticas Sin Restricciones (GSR).

Autómatas Linealmente Acotados

Reconocen Lenguajes Sensibles al Contexto que no contienen la cadena vacía. Son equivalentes a las Máquinas de Turing pero solo pueden operar en la parte de la cinta ocupada por la cadena de entrada. Debe contener dos símbolos especiales:

1. [: marcador izquierdo, con transiciones del tipo δ(q_i, [) = (q_j, [, D)
2.]: marcador derecho, con transiciones del tipo δ(q_i,]) = (q_j,], I)h

Tema 7: Decidibilidad y Complejidad

Lenguajes Recursivos y Recursivamente Enumerables

1. Un lenguaje es recursivamente enumerable (LRE) si existe una MT que reconozca el lenguaje. Si la cadena es del lenguaje se para en un estado final, pero si la cadena no es del lenguaje puede pararse en un estado no final o no pararse.
2. Un lenguaje es recursivo (LRC) si existe una MT que reconozca el lenguaje. Si la cadena es del lenguaje se para en un estado final y si no es del lenguaje se para en un estado no final.

Problema de la Parada en MT

El problema de la parada en Máquinas de Turing es que no puede saberse si parará en algún momento o no. Una solución sería una MT Universal que pueda replicar la MT original pero que garantice la parada. No existe ninguna MT Universal que garantice la parada, es decir, es un problema indecidible. Si el problema de la parada fuera decidible todos los LRE serían LRC.

Complejidad Computacional

Si una computación tiene complejidad (de tiempo) T(n) significa que puede ser resuelta en no más de T(n) movimientos de una MT para un problema de tamaño n. No nos interesa el tiempo exacto sino su orden de magnitud O(...). Desde el punto de vista de la decidibilidad todas las MT son equivalentes pero desde el punto de vista de la complejidad no. Ejemplo: L= $\{a^n b^n : n \geq 1\}$ MT estándar: O(n^2), MT con dos cintas: O(n).

Problema de Satisfacibilidad (SAT)

1. Expresiones en forma normal conjuntiva: $e = t_i \wedge t_j \wedge \ldots \wedge t_k$
    a. $t_i = s_m \vee s_p \vee \ldots \vee s_q$ donde s son variables o sus negaciones
2. Dada una expresión e en forma normal conjuntiva, ¿hay alguna asignación de valores a sus variables que haga e verdadera?
3. Ejemplos: $e_1 = (!x_1 \vee x_2) \wedge (x_1 \vee x_3)$, $e_2 = (x_1 \vee x_2) \wedge !x_1 \wedge !x_2$
4. MT determinista: O(2^n)
5. MT no determinista: O(n)

Complejidades P y NP

1. Complejidad P: problemas tratables. Aceptados por una MT Determinista en tiempo polinómico.
2. Complejidad NP: problemas intratables. Aceptados por una MT No Determinista en tiempo polinómico.
    a. Un Un lenguaje L es NP-completo si L $\in$ NP y todo L' $\in$ NP es reducible en tiempo polinómico a L. SAT es NP-completo

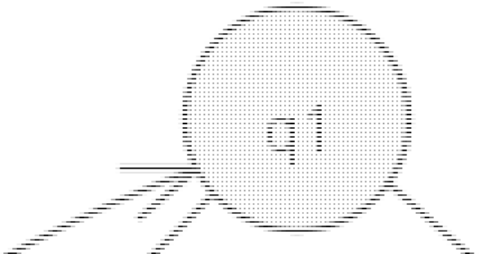
Problemas

Problemas Tema 2

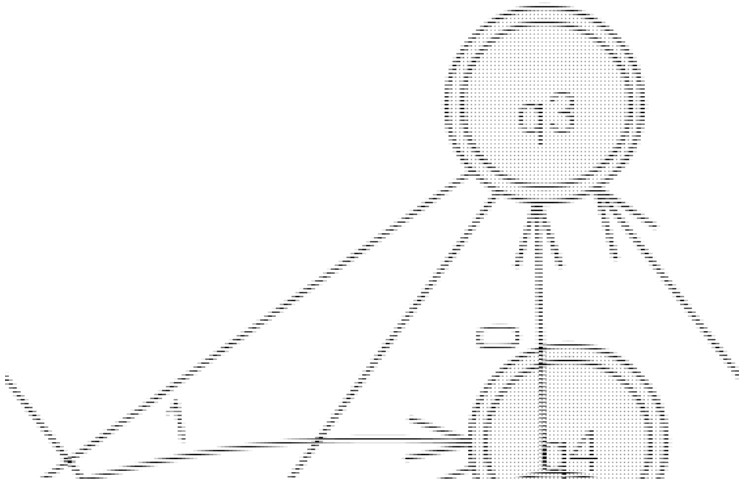
Problema 1: Construir los AFD que acepten los siguientes lenguajes sobre el alfabeto {0, 1}:

1. El conjunto de cadenas con 011 como subcadena

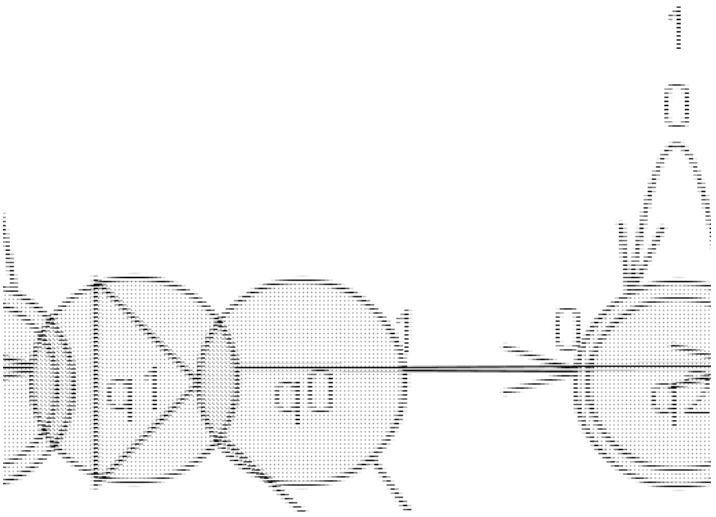
2. El conjunto de cadenas terminadas en 00



3. El conjunto de cadenas cuyo tercer símbolo desde el extremo derecho sea un 1



4. El conjunto de cadenas que empiezan o terminan por 01



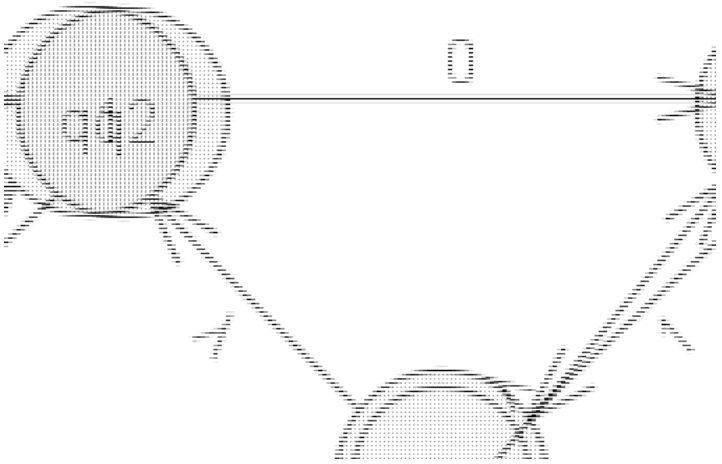
5. El conjunto de palabras que no contienen las subcadenas 100

6. El conjunto de cadenas que contengan un número par (se incluye 0) de subcadenas con un número par de ceros consecutivos. Por ejemplo, el autómata deberá reconocer la cadena 001100010000, pero no la cadena 0001000100

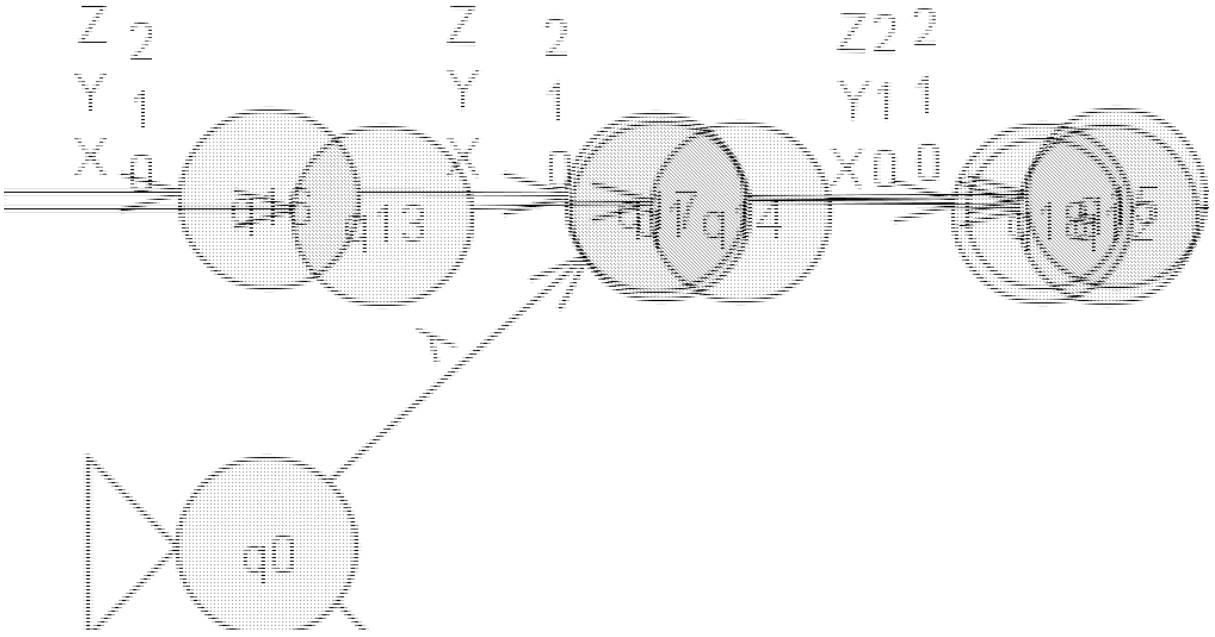
Problema 2: Diseñar el AFN-Ɛ para los siguientes lenguajes:

1. Conjunto de cadenas con cero o más letras a seguidas de cero o más letras b seguidas de 0 o más letras c

2. El conjunto de cadenas formadas por 01 repetido una o más veces o por 010 repetido una o más veces



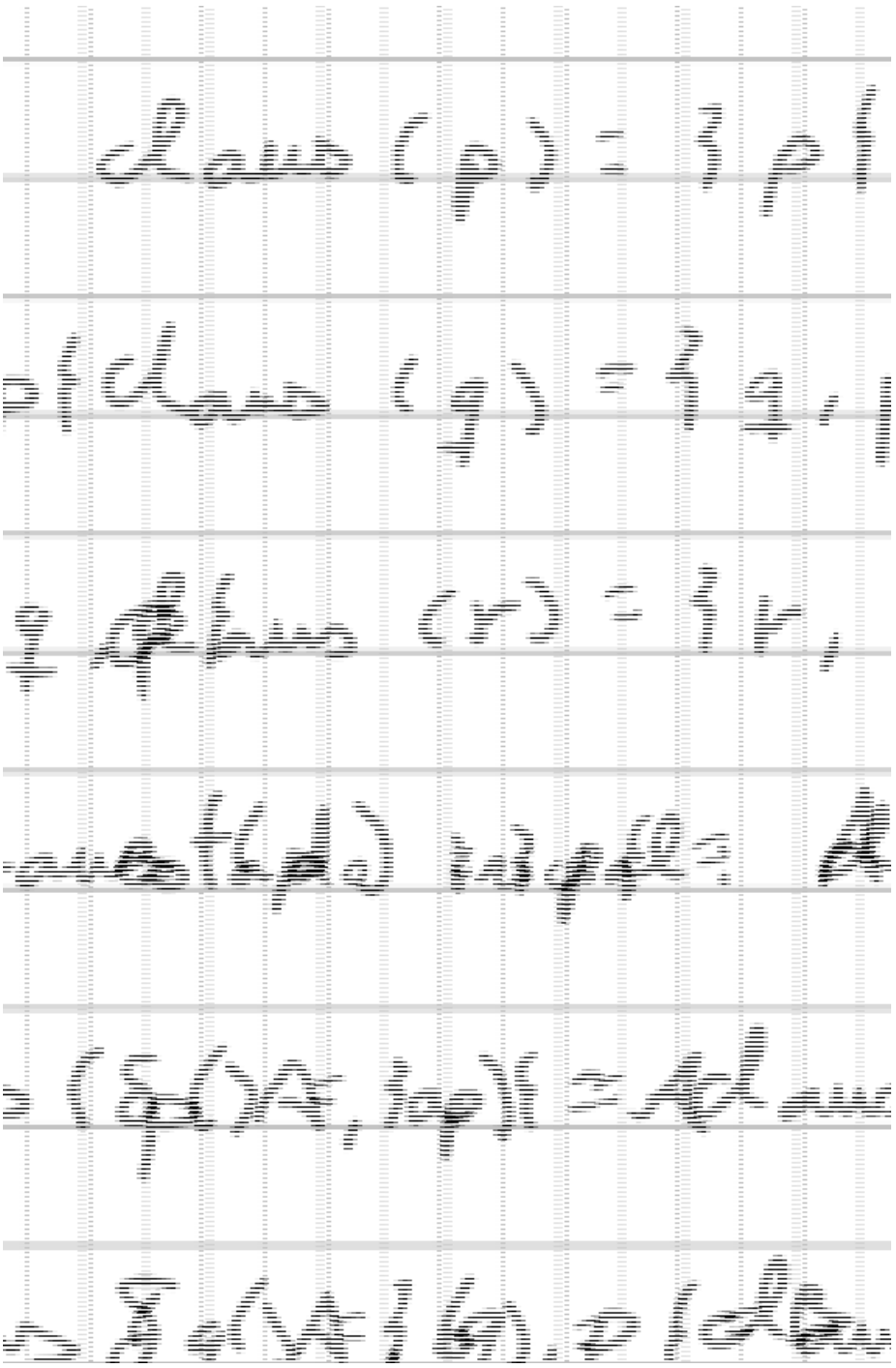
3. AF que sobre el alfabeto {X, Y, Z, 0, 1, 2} reconoce matrículas de vehículos válidas, para las provincias “X”, “YY” y “ZX”. El formato de las matrículas podrá ser:
    a. Provincia, 4 números y 0, 1 o 2 letras
    b. 4 números y tres letras



Problema 3: Dados los siguientes AFN-Ɛ, calcular la clausura respecto de Ɛ para cada estado y convertir los autómatas en AFD:

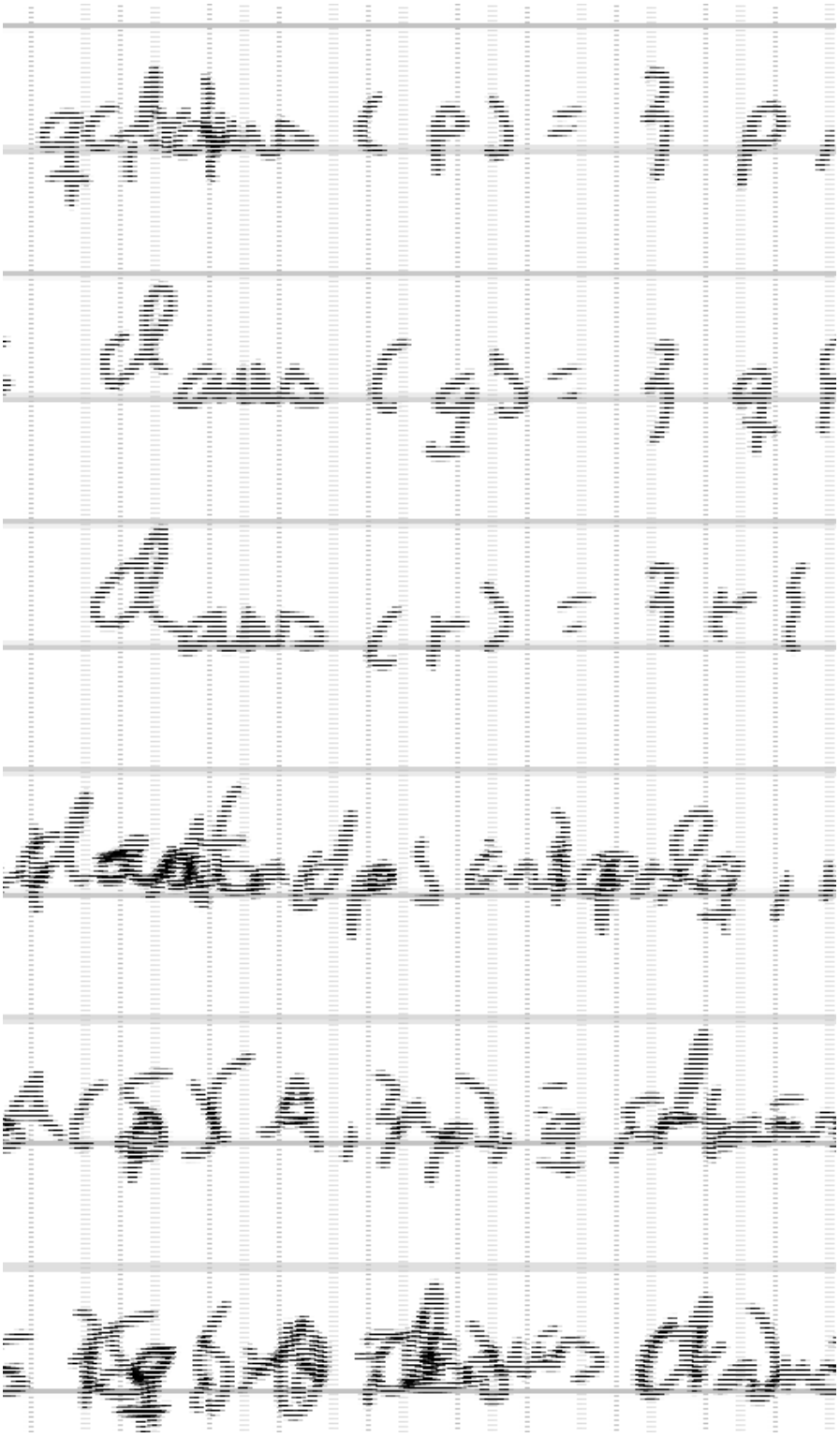
1.

	Ɛ	a	b	c
→p	Θ	{p}	{q}	{r}
q	{p}	{q}	{r}	Θ
*r	{q}	{r}	Θ	{p}



2.

	Ɛ	a	b	c
→p	{q, r}	Θ	{q}	{r}
q	Θ	{p}	{r}	{p, q}
*r	Θ	Θ	Θ	Θ

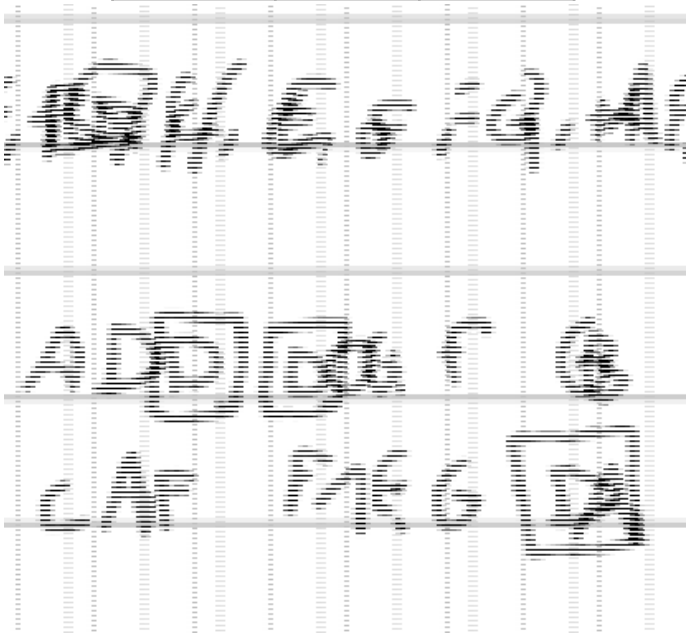


Problema 4: Dados los siguientes AFD, calcular los AFD equivalentes mínimos:

1.

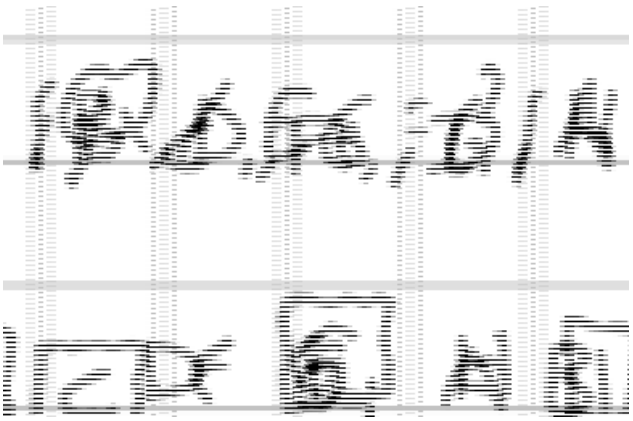
	0	1
→A	B	A
B	A	C

C	D	F
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D



2.

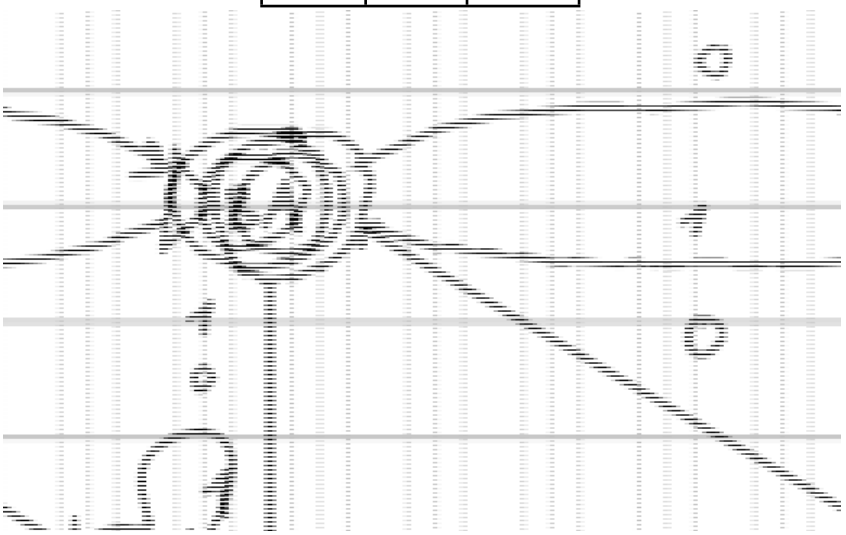
	0	1
→A	B	E
B	C	F
C	D	H
*D	E	H
E	F	I
F	G	B
G	H	B
H	I	C
I	A	E

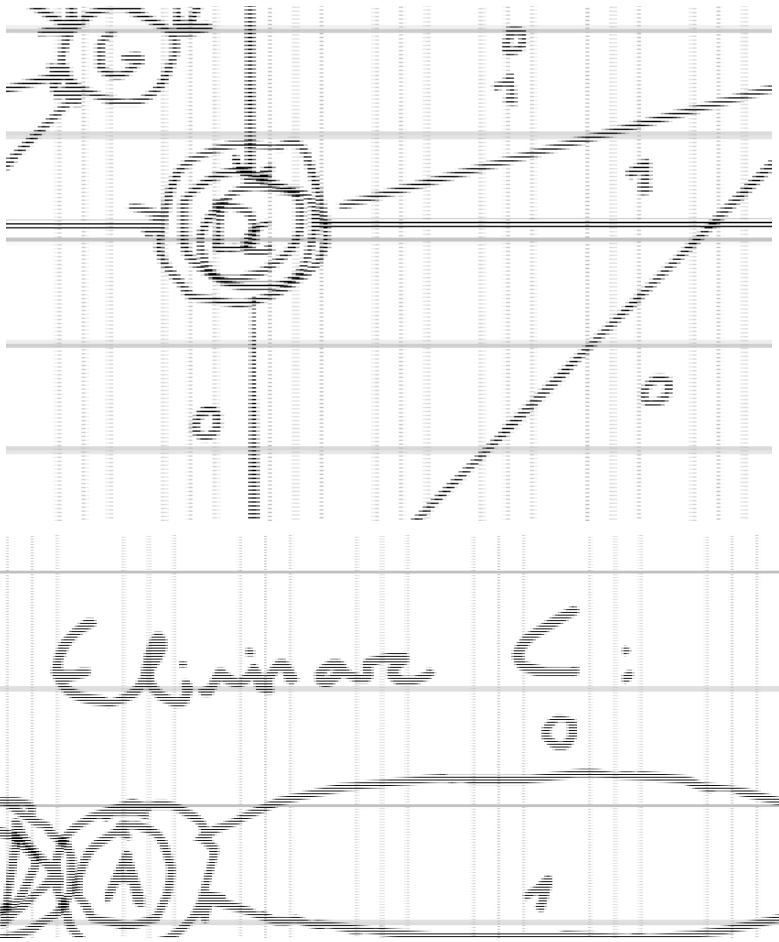


Problemas Tema 3

Problema 1: Obtener la expresión regular del AFD definido en la siguiente tabla de transiciones:

	0	1
→*A	B	C
*B	G	A
C	E	D
*D	G	G
*E	F	E
F	G	E
G	G	G





Problema 2: Dado el siguiente AFD, obtener la ER para el lenguaje del autómata:

	0	1
$\rightarrow q_1$	q_2	q_2
$^* q_2$	q_3	q_1
q_3	q_3	q_2

Problema 2: Convertir las siguientes ER en AFN:

1. 01*

2. 0*1*

Problema 3: Demostrar si se verifica el lema del bombeo para el lenguaje descrito por la siguiente ER: $0^m 1^m$

Recordemos que debe existir una constante n tal que, para toda cadena w perteneciente al lenguaje, con lwl ≥ n, podamos dividir w en tres cadenas, w = xyz, de modo que:

1. y ≠ Ɛ
2. lxy l ≤ n
3. para todo k≥0, la cadena xykz también pertenece al lenguaje

Si tomamos n=5 y w=0000011111, podremos considerar: x=0000; y=0; y z=11111, verificándose los puntos 1 y 2 previos, pero no el punto 3, ya que si k=0, por ejemplo, resultará xy^0z=000011111, que no pertenece al lenguaje.

Problema 4: Dados los siguientes lenguajes, razonar si son regulares, es decir, si es posible reconocerlos con un autómata finito.

1. El que consta de todas las cadenas con el mismo número de 0 y 1 (sin ningún orden particular)
2. El conjunto vacío
3. {00, 11}
4. (00 + 11)*
5. L = {h {a, b}* : N_a(h) < N_b(h)}
6. L = {$a^i b^j c^k$ | k=i-j; i, j, k >= 0}

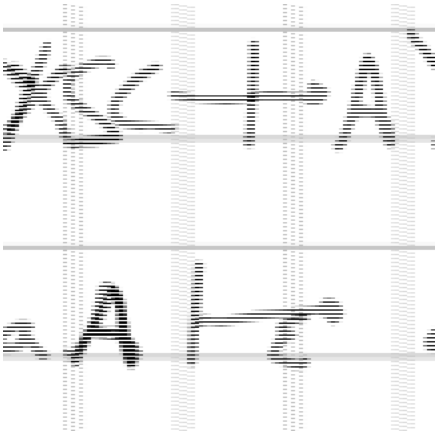
Problemas Tema 4

Problema 1: Diseñar las GIC que generen los siguientes lenguajes:

1. L={ $0^n 1^n$ | n≥1}

2. L = {$a^i b^j$ | 2i = j; i, j > 0}

3. L = {$a^i b^j c^k$ | i ≠ j ó j ≠ k}



4. L = {(a+b+c)* / N(a)+N(b)>N(c)}

Problema 2: Dada la gramática siguiente, eliminar producciones Ɛ y unitarias y símbolos inútiles:

S → AC | BS | B
A → aA | aF
B → cF | b
C → cC | D
D → aD | BD | C
E → aA | BSA
F → bB | b

1. No tiene producciones ε
2. Producciones unitarias: son (S,B), (C,D) y (D,C), por tanto, la producción S → B se elimina, rescribiéndose como S → cF y S → b. La producción C → D se elimina, reescribiéndose como C → aD, C → BD y C→C, aunque esta última se obvia, ya que no aporta nada. La producción D → C se elimina y se reescribe como D → cC y D → D, obviándose esta última. El resultado es el siguiente:

S → AC | BS | cF | b
A → aA | aF
B → cF | b
C → cC | aD | BD
D → aD | BD | cC
E → aA | BSA
F → bB | b

3. Sobre las nuevas producciones de la gramática, eliminamos los símbolos inútiles, buscando los generadores y los alcanzables.
    a. Veamos cuales son los generadores, comenzando por los terminales o símbolos del alfabeto: a, b, c. Serán generadores aquellos que se resuelvan

directamente como los terminales o combinaciones de estos mediante producciones. Por ejemplo: F → b. En concreto, son los siguientes: S, F, B, a, b, c. Si repetimos el proceso, ahora serán también generadores los que estén asociados a producciones cuya parte derecha esté conformada solo por una combinación de los símbolos que ya han sido identificados como generadores: A, S, F, B, a, b, c; repetimos el proceso de nuevo y nos da: E, A, S, F, B, a, b, c; lo repetimos una vez más y ya no aparece ningún otro símbolo generador nuevo, luego hemos acabado. Por tanto, C y D no son generadores –estos ya se sabe que son símbolos inútiles al no ser generadores-. Esto elimina las producciones: S → AC; D → BD; C → BD; D → cC; C → aD; D → aD; C → cC.

S → BS | cF | b
A → aA | aF
B → cF | b
E → aA | BSA
F → bB | b

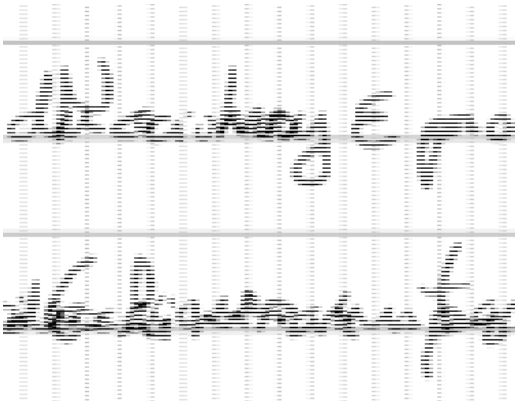
b. Buscamos ahora los alcanzables, después de haber eliminado las anteriores del conjunto de producciones. Para ello comenzamos por S, y vemos cuales son directamente alcanzables por producciones de la forma S → …; serían los siguientes: S, B, c, F, b Ahora haríamos lo propio a partir de producciones de las formas: B->… y F->…, pero nos da lo mismo: S, B, c, F, b. Por tanto hemos acabado. Las producciones que quedan se muestran a continuación:

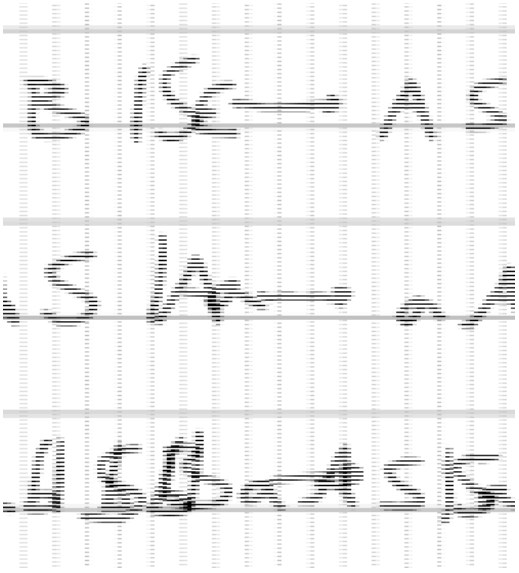
S → BS | cF | b
B → cF | b
F → bB | b

Problema 3: Encontrar una gramática equivalente S → AB | CA, A → a, B → BC | AB, C → aB | b, sin símbolos inútiles:

S → A | CA
A → a
C → a | b

Problema 4: Dada la gramática S → ASB | Ɛ, A → aAS | a, B → SbS | A | bb, obtener la gramática equivalente en FNC:

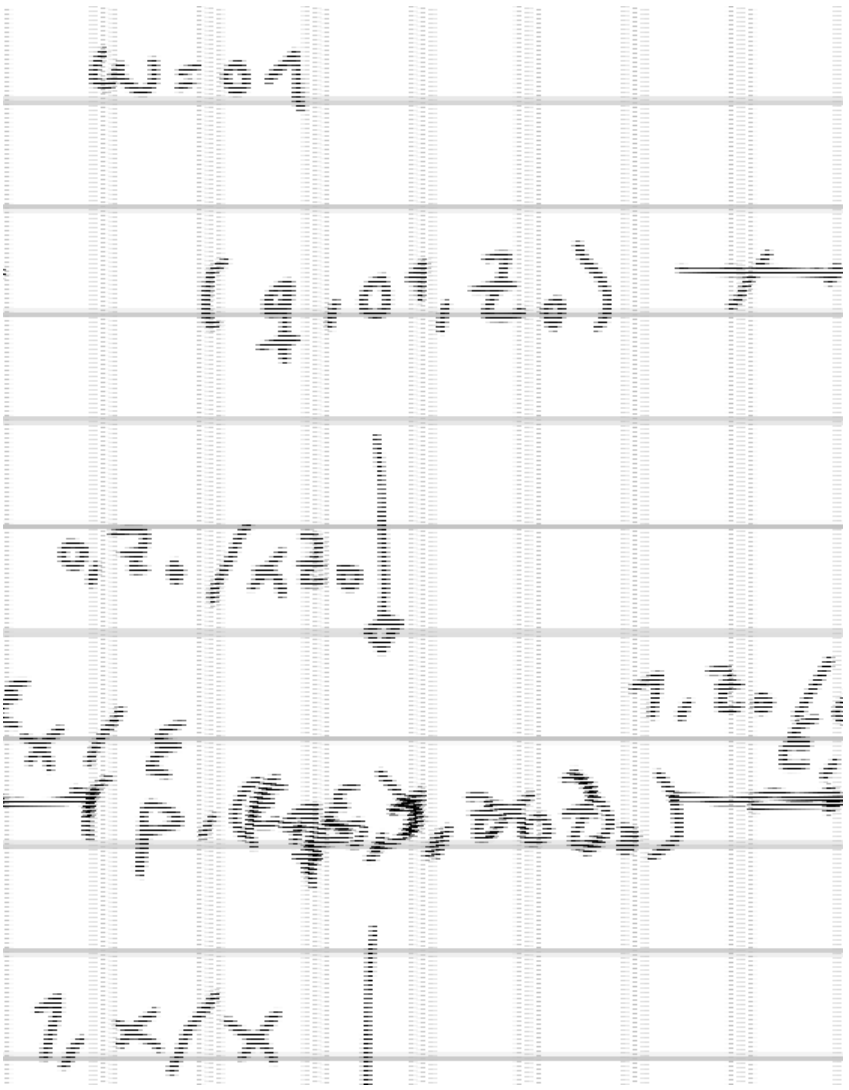




Problemas Tema 5

Problema 1: Dado el AP P = ({q, p}, {0, 1}, {Z_0, X}, δ, q, Z_0, {p}), donde δ se define en la siguiente tabla, mostrar las configuraciones alcanzables a partir de la inicial (q, w, Z_0), para w igual a 01 y 010:

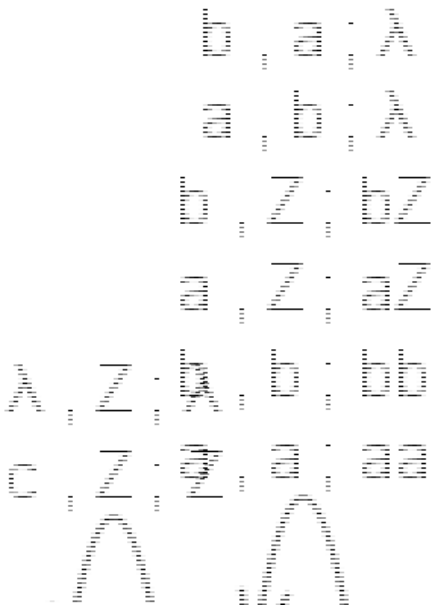
Q	Σ	Γ	Movimiento
q	0	Z_0	(q, XZ_0)
q	0	X	(q, XX)
q	1	X	(q, X)
q	ε	X	(p, ε)
p	ε	X	(p, ε)
p	1	X	(p, XX)
p	1	Z_0	(p, ε)



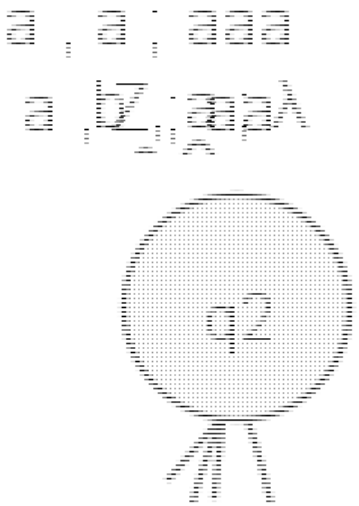
Problema 2: Diseñar un APF que acepte los lenguajes $\{0^n1^n \mid n > 0\}$ y L={ 0^n1^n | n≥0}:

Problema 3: Diseñar un APN que acepte el lenguaje formado por aquellas cadenas que cumplen alguno de los siguientes criterios:

1. contienen igual número de símbolos a y b, entrando estos en cualquier orden, y finalizan con un número k de símbolos c, k ≥ 0
2. $a^i b^j c^k$ | k > i



Problema 4: Diseñar el APF sobre el alfabeto {a, b} que acepte los lenguajes $\{a^i b^j \mid 2i = j;\ i, j > 0\}$ y $\{a^i b^j \mid i = 2j;\ i, j > 0\}$



Problema 5: Verificar si se cumple el lema del bombeo para: L=$\{a^n b^n c^n \mid n \geq 1\}$:
Si L es un LIC, entonces existe una constante n tal que si z es cualquier cadena de L de longitud |z| >= n, podemos escribir z=uvwxy, cumpliendo las condiciones:

1. $|vwx| \leq n$
2. $vx \neq \varepsilon$

Al tener que cumplirse: $|vwx| \leq n$, vwx no pueden contener al mismo tiempo los tres símbolos del alfabeto, pero vx contendrá al menos un símbolo. Por tanto, si consideramos k=0, resulta uv^0wx^0y = uwy, y esta cadena no podrá pertenecer a L ya que le faltarán los elementos de vx para estar equilibrada.

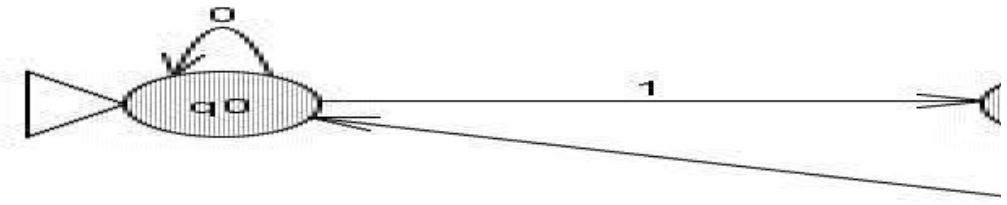
Boletines

Boletín 1: Diseño de Autómatas de Estados Finitos

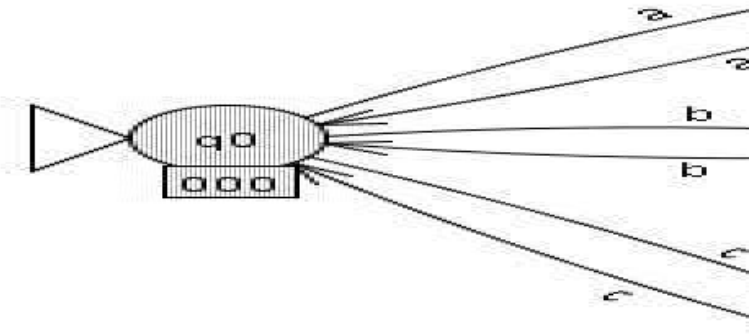
Para la construcción de los autómatas de estados finitos se utilizará la herramienta JFLAP. Será necesario entender y explicar el diseño del autómata. Los autómatas deberán ser AFD o AFN a vuestra elección, salvo que se realice alguna indicación específica.

1. AF que sobre el alfabeto {0, 1} reconoce las palabras que tienen un número par (mayor que 0) de subcadenas “01”.

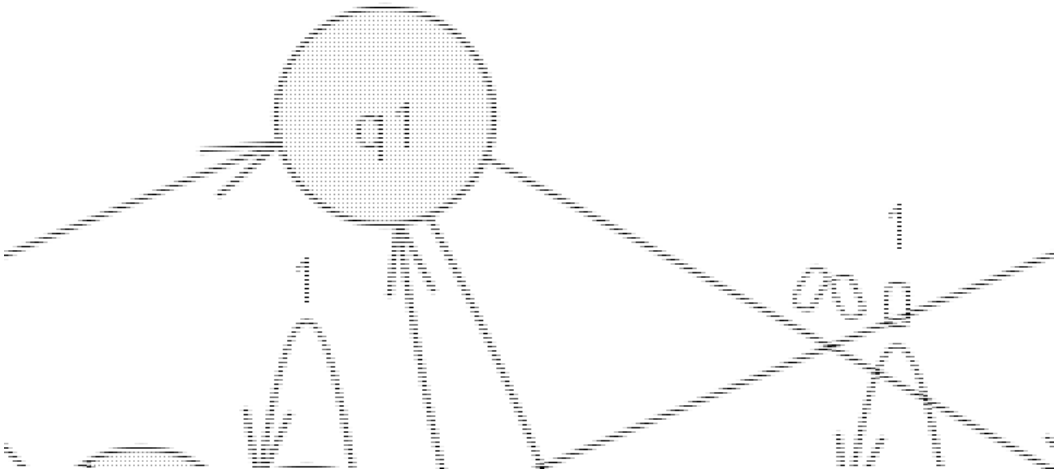
2. AF que sobre el alfabeto {0, 1} reconoce las palabras que finalizan con la subcadena “101”, pero ésta no aparece en ningún otro lugar de la palabra.



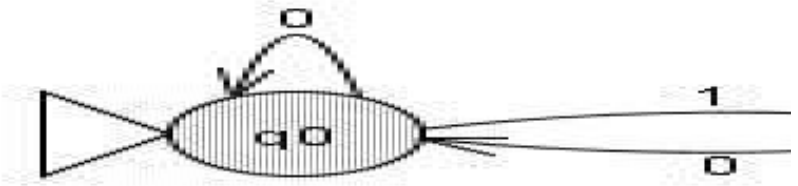
3. AF que sobre el alfabeto {a, b, c} reconoce las palabras que contengan un número impar de “a”, “b”, y “c” (no su suma total, sino su número individual).



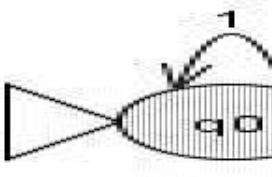
4. AF que sobre el alfabeto {0, 1} reconoce las palabras que no contienen la subcadena “000” y finalizan con la subcadena “01”.



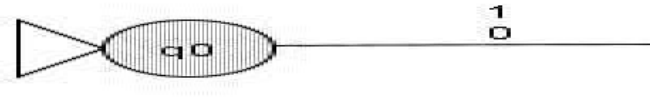
5. AF que sobre el alfabeto {0, 1} reconozca el lenguaje formado por aquellas cadenas que contienen en algún lugar de la cadena un número consecutivo de “1” par (y mayor que 0).



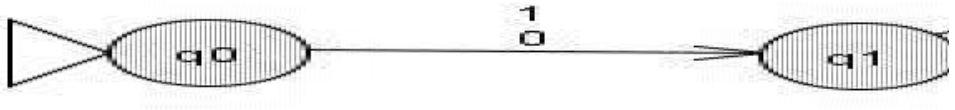
6. AF que sobre el alfabeto {0, 1} reconozca el lenguaje formado por aquellas cadenas que comienzan por la subcadena “01”, pero que no contienen dicha subcadena en ninguna otra posición.



7. AF que sobre el alfabeto {0, 1} reconoce las palabras cuyo segundo símbolo empezando por la izquierda coincide con el segundo símbolo empezando por la derecha (la cadena tendrá una longitud mínima de 4 símbolos). Restricción: el autómata debe ser determinista

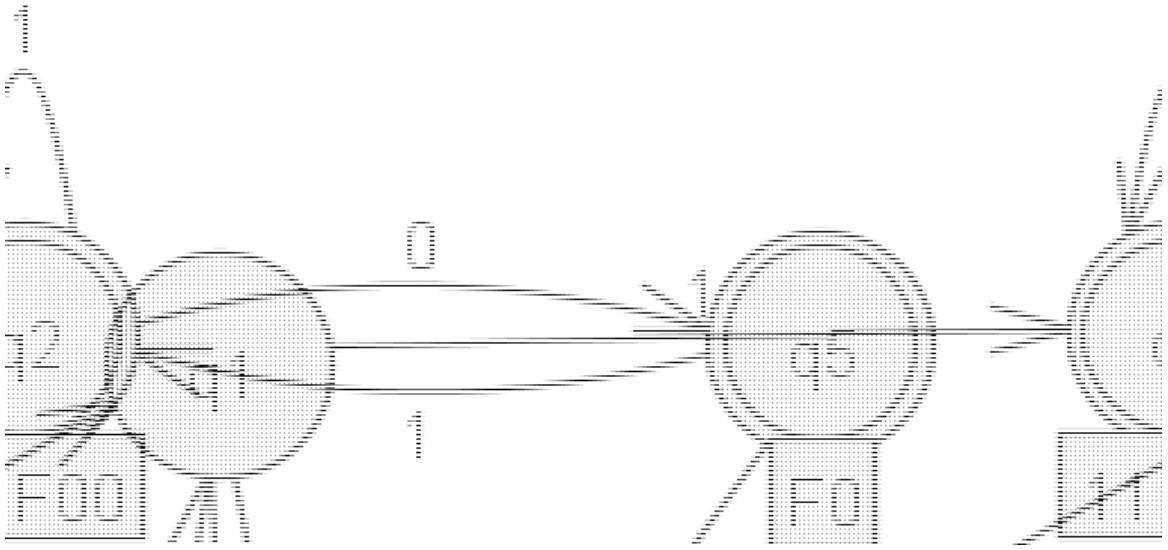


8. Diseñar un autómata no determinista para resolver el problema anterior, sin que contenga transiciones con la cadena vacía.

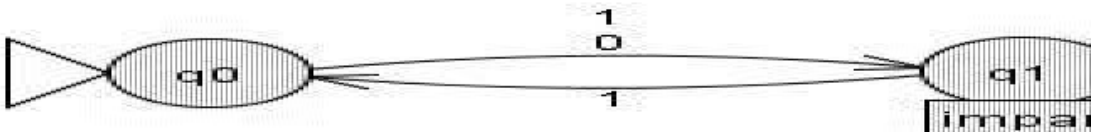


9. AF que sobre el alfabeto {0, 1} reconoce las palabras que contienen la subcadena “101 y no finalizan en “11”. Por ejemplo, la cadena 1011 debería ser rechazada.

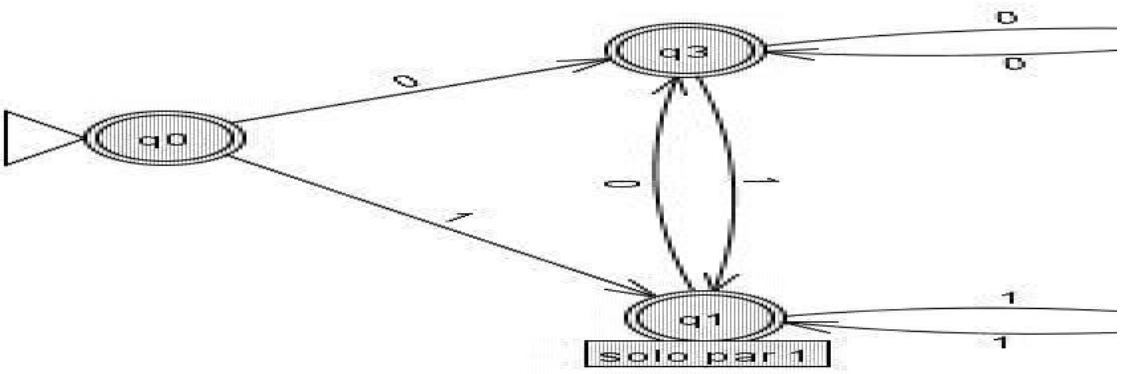
10. AF que reconozca, para el alfabeto {0, 1}, el lenguaje formado por aquellas cadenas que no contienen la subcadena 00, pero sí la subcadena 11.



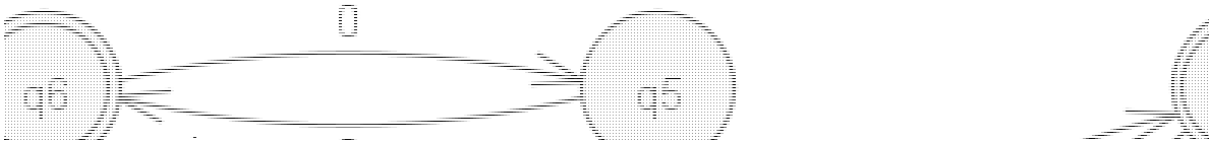
11. AF que reconozca, para el alfabeto {0, 1}, el lenguaje formado por aquellas cadenas en las que el número de símbolos leídos antes del último par de ceros es impar.



12. AF que reconozca, para el alfabeto {0, 1}, el lenguaje formado por aquellas cadenas en las cuales, si aparece una subcadena con un número par (mayor que cero) de unos consecutivos, siempre va precedida y seguida por subcadenas con un número par (mayor que cero) de ceros consecutivos. No es obligatorio que las subcadenas con un número par de ceros se lean exactamente antes y después de la subcadena de unos, pero sí antes de la aparición de una nueva subcadena con un número par de unos consecutivos. Por ejemplo, el autómata deberá reconocer la cadena 11100100011110100001011000O. En caso de que la entrada no contenga ninguna subcadena con un número par de unos, el AF deberá aceptarla. Por ejemplo, se deberían aceptar las cadenas 111 ó 0.



13. AF que reconozca, para el alfabeto {0, 1}, el lenguaje formado por aquellas cadenas que contienen una subcadena con un número par (mayor que cero) de ceros consecutivos. Esta subcadena debe ser única. Además, si entra alguna subcadena con un número par (mayor que cero) de unos consecutivos, la subcadena con el número par de ceros debe aparecer después de la última subcadena con un número par de unos consecutivos. Por ejemplo, el autómata deberá reconocer la cadena 110001111011001.



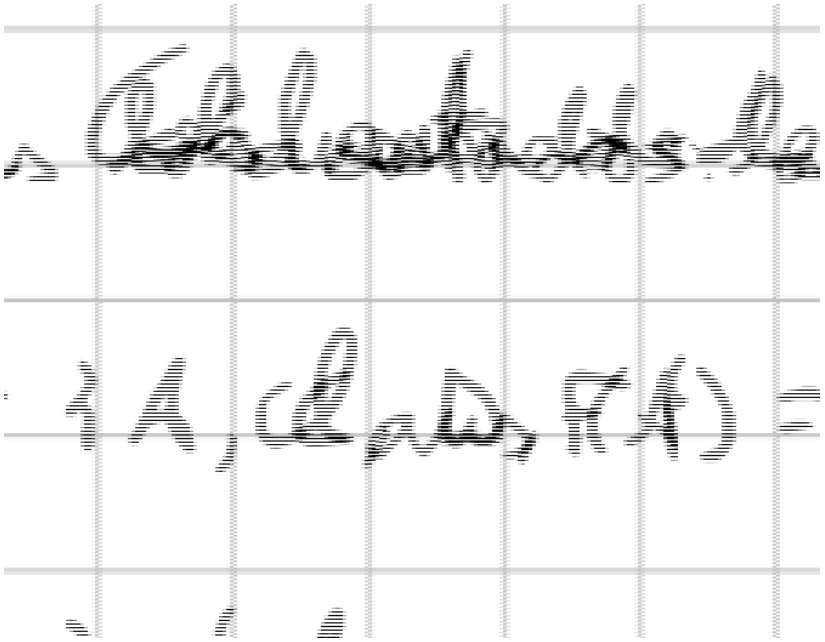
Boletín 3: Minimización de AF y Expresiones Regulares

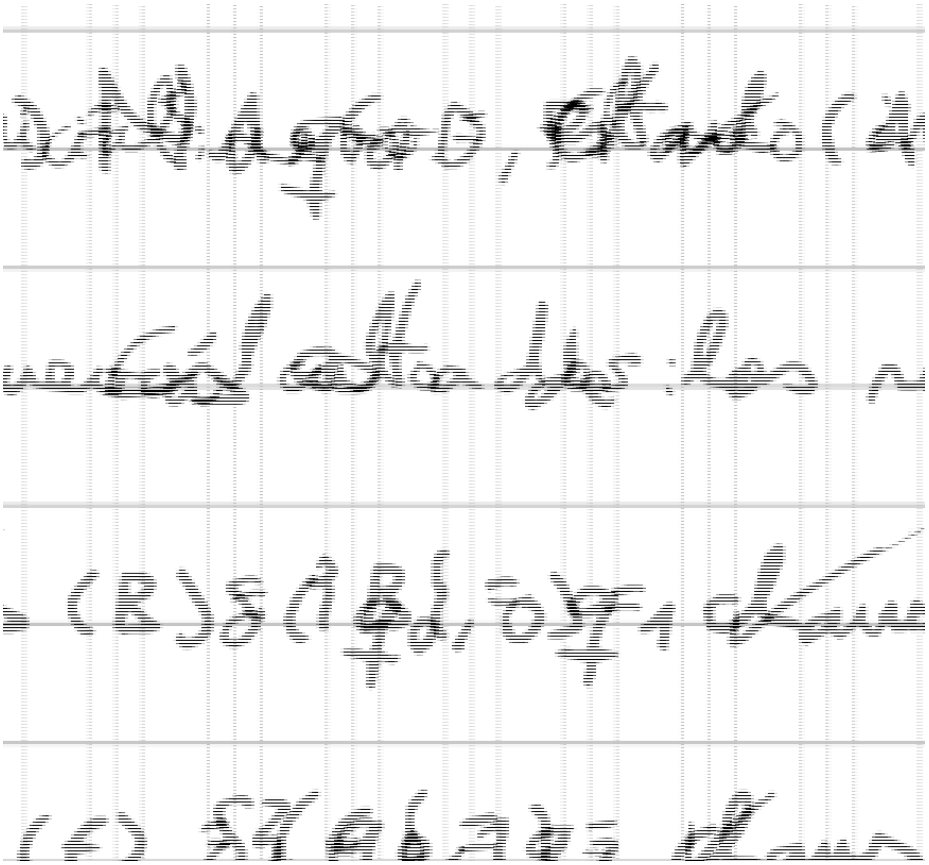
Para los ejercicios de AF mínimo y obtención de ER será necesario entregar escaneados todos los cálculos realizados a mano, y comprobar con JFLAP que cada uno de los pasos es correcto. Para los ejercicios de obtención del AF a partir de la ER, será necesario entregar el fichero de JFLAP con vuestro diseño.

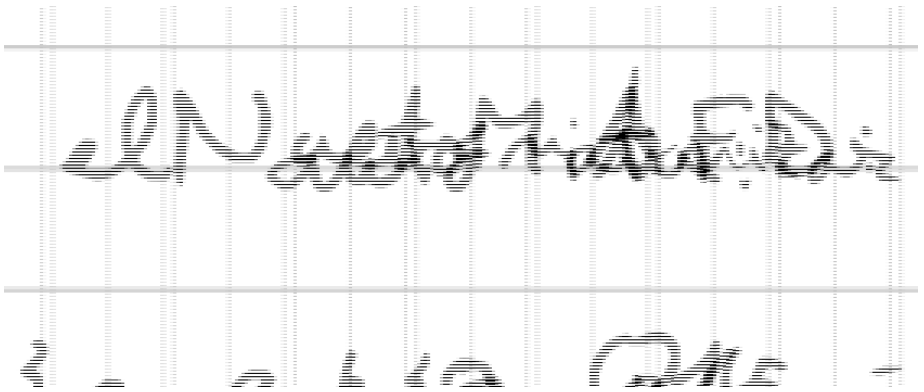
1. Obtener el autómata finito determinista equivalente mínimo del autómata AF = ({0, 1}, {A, B, C, D, E, F}, f, A, {F}), donde f está definida en la siguiente tabla de transiciones:

	0	1	Ɛ
→A	B	E	C

B	C		
C			A, D, F
D		E	F
E		F	
*F			D

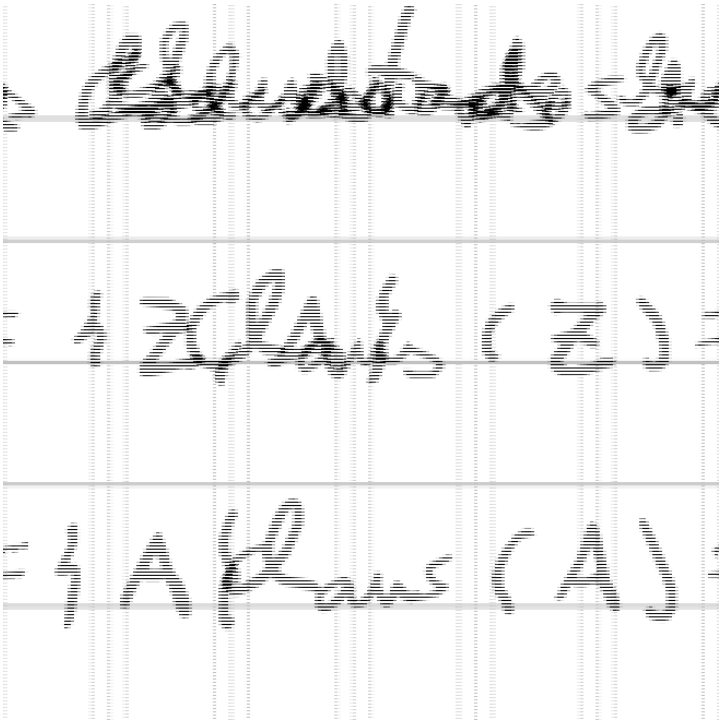


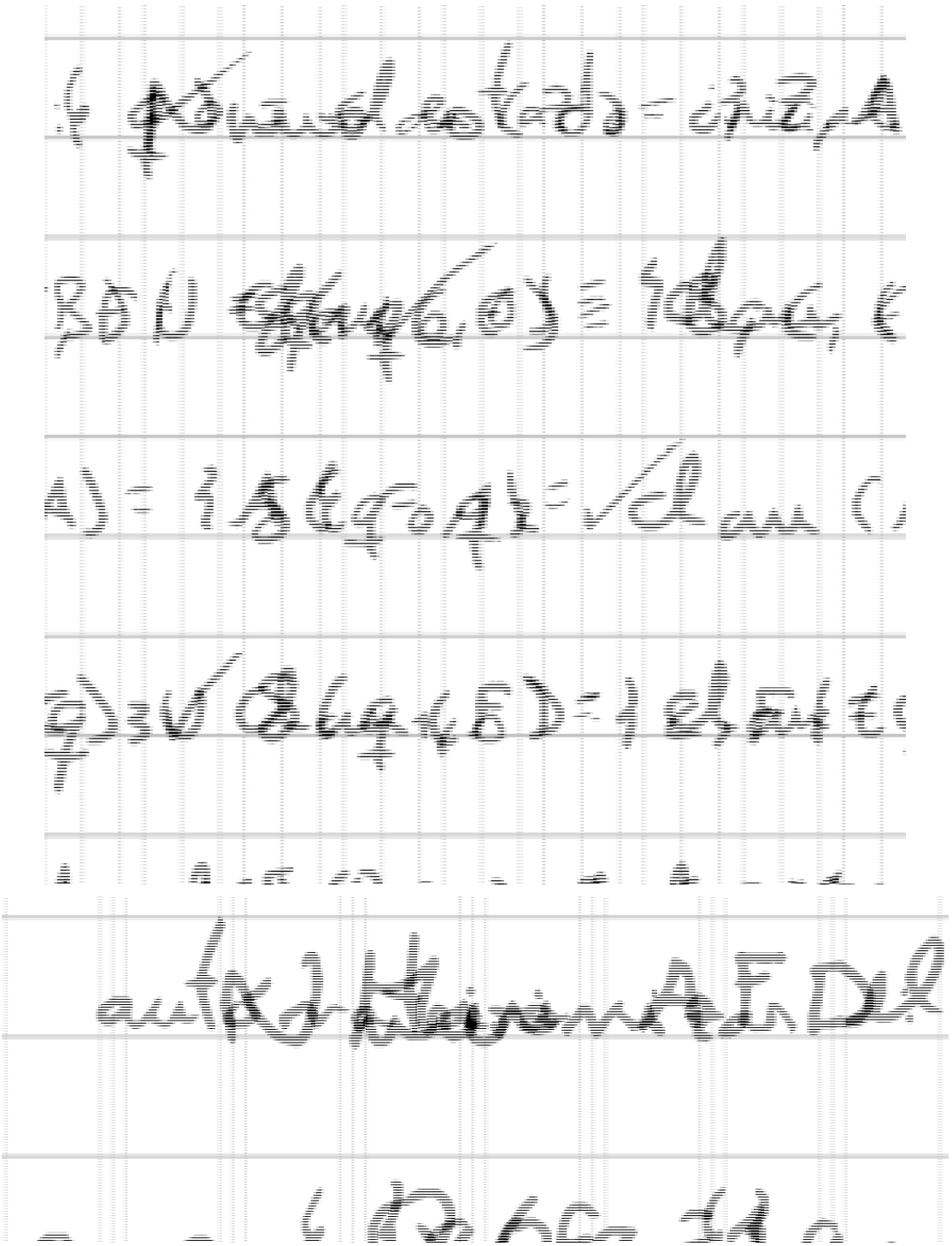




2. Obtener el autómata finito determinista equivalente mínimo del autómata AF = ({0, 1}, {Z, A, B, C, D, E, F, G}, f, Z, {B, D, E, G}), donde f está definida en la siguiente tabla de transiciones:

	0	1	ε
→Z			A
A	B, E	A	
*B		B	C
C	C	D	
*D			B
*E		E	F
F	F	G	
*G			E

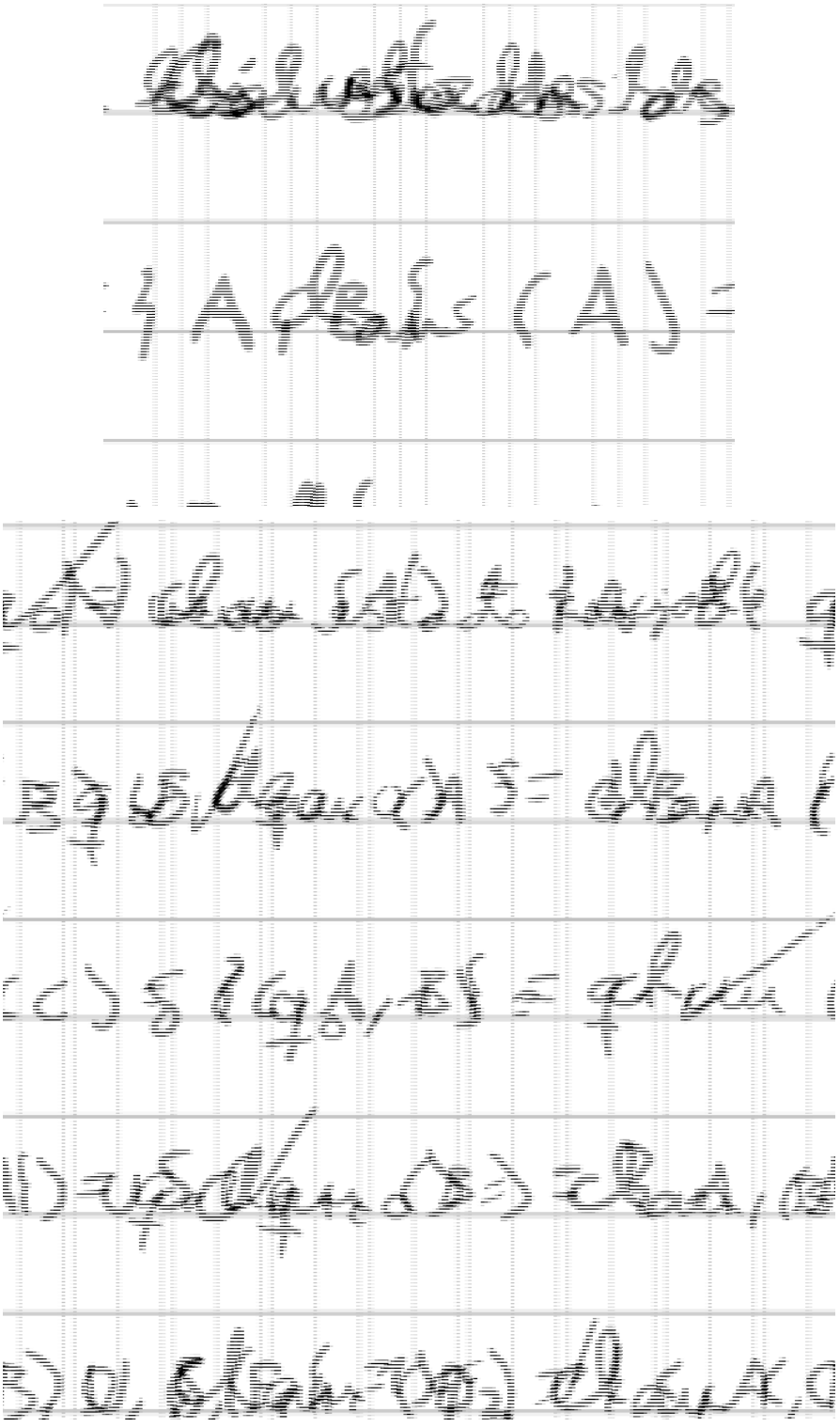


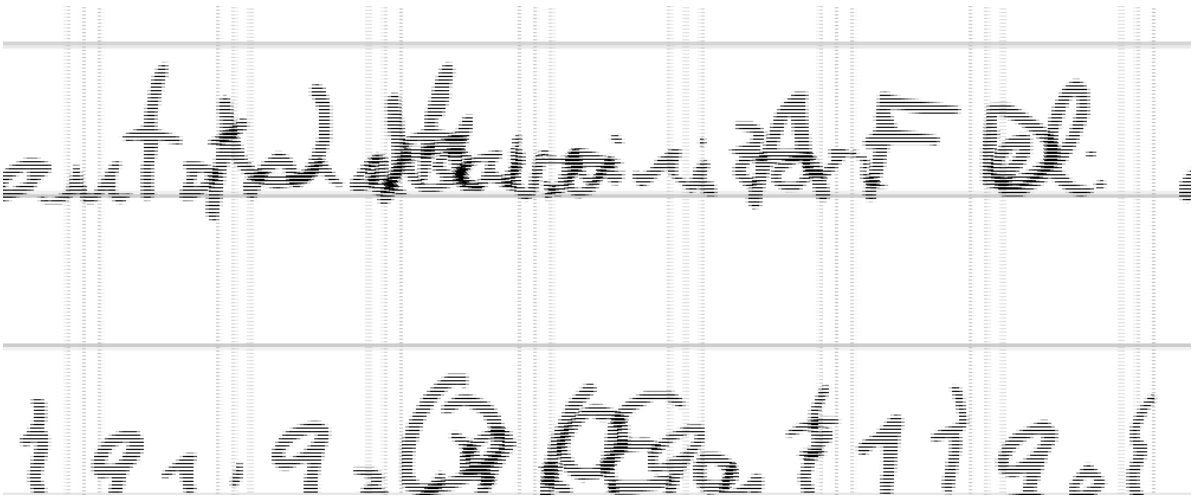


3. Obtener el autómata finito determinista equivalente mínimo del autómata AF = ({0, 1}, {A, B, C, D, E, F}, f, A, {C, E}), donde f está definida en la siguiente tabla de transiciones:

	0	1	ε
→A	B		B
B	A	C	A
*C		D	A

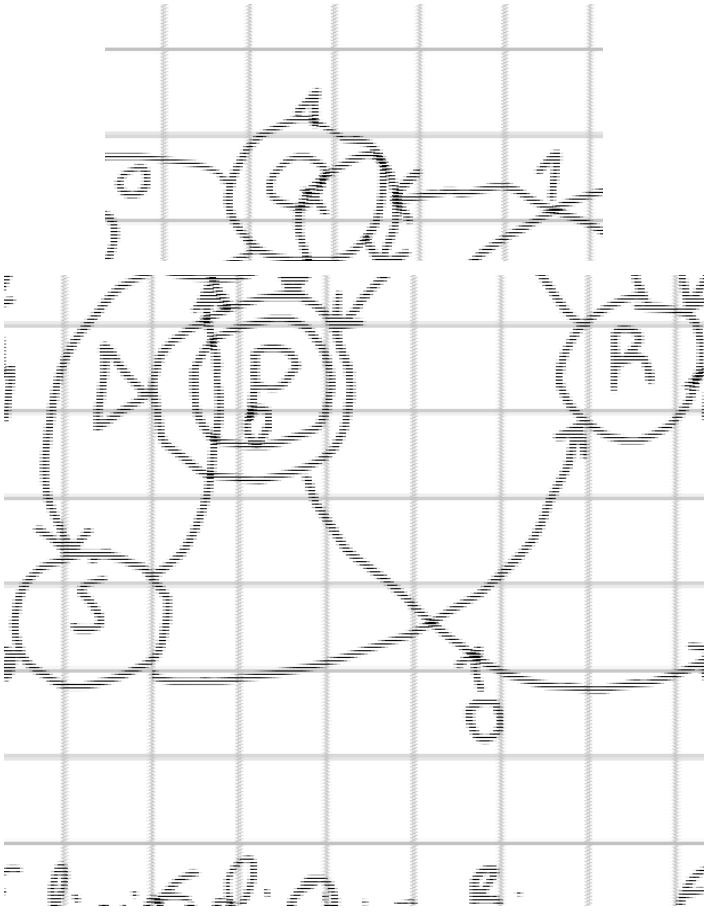
D	F	F	E
*E		E	F
F	E		





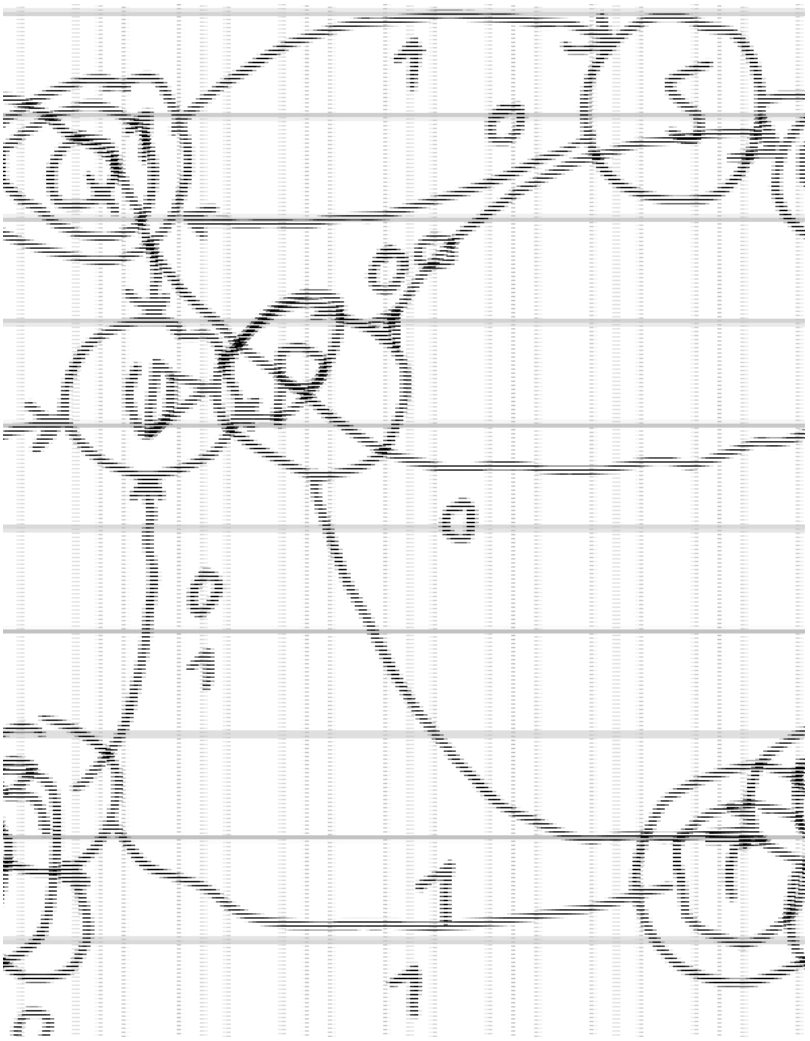
4. Obtener la expresión regular (minimizada) que representa el lenguaje reconocido por el siguiente autómata:

	0	1
→*P	S	P
Q	P	S
R	R	Q
S	Q	R



5. Obtener la expresión regular (minimizada) que representa el lenguaje reconocido por el siguiente autómata:

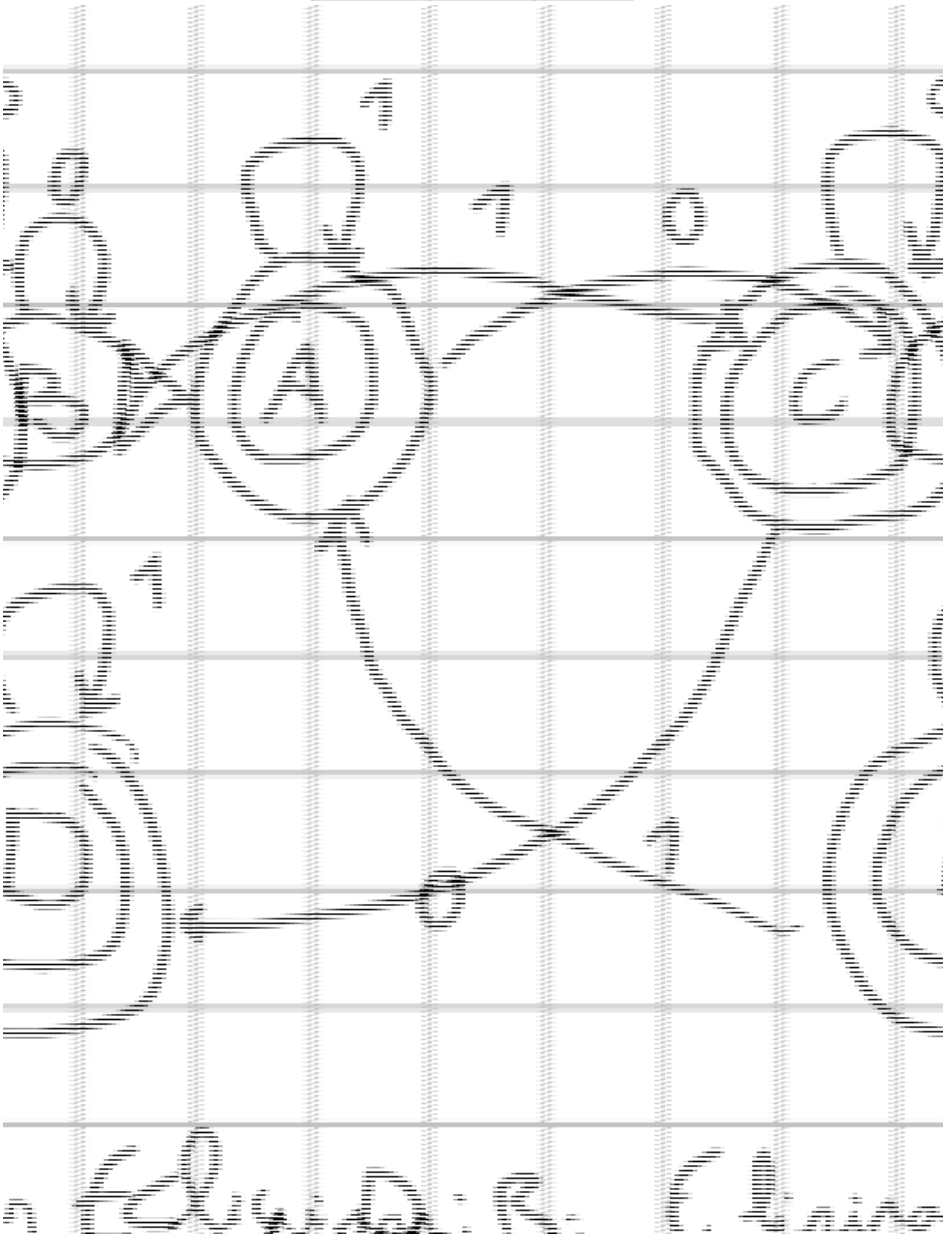
	0	1
→P	Q	R
*Q	U	S
R	R	T
S	Q	U
*T	U	U
U	U	U



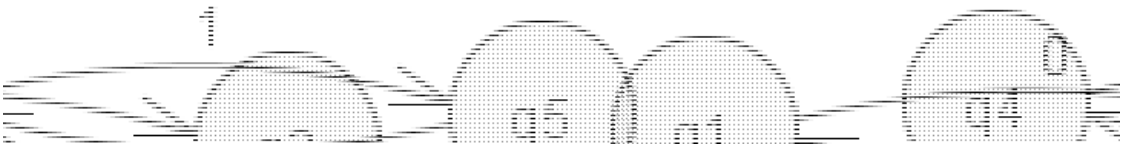
6. Obtener la expresión regular (minimizada) que representa el lenguaje reconocido por el siguiente autómata:

	0	1
→*A	B	A

B	B	C
*C	C	D
*D	A	D

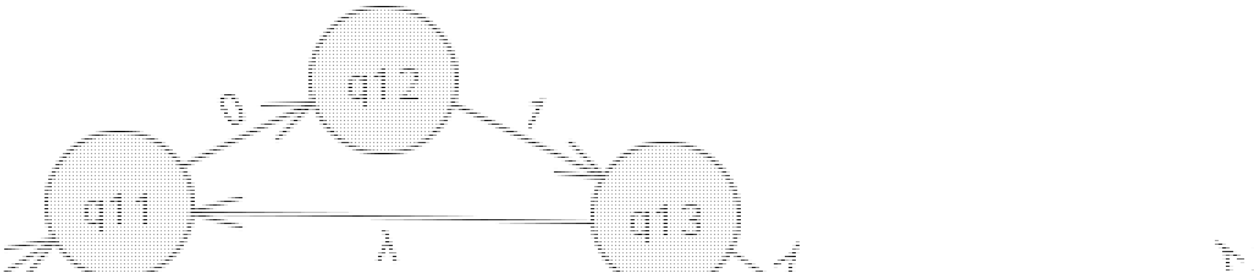


7. Obtener el autómata de estados finitos que reconoce el lenguaje dado por la expresión regular: 0* 1*



8. Obtener el autómata de estados finitos que reconoce el lenguaje dado por la expresión regular: (00)* (0+1)*

9. Obtener el autómata de estados finitos que reconoce el lenguaje dado por la expresión regular: (0* 1^{+}) {(01) * + 11}

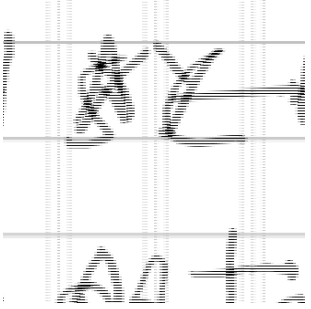


Boletín 4: Diseño de Gramáticas Independientes del Contexto (GIC)

Diseñar las gramáticas independientes del contexto que generen los siguientes lenguajes:

1. Un palíndromo es una cadena que se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, las palabras “radar”, “oso” y “abba” son palíndromos. Dado el alfabeto S = {a, b}, determinar una gramática que describa palíndromos. La gramática debería generar palabras como “abba”, “aba”, “bb”, “babab”, “a”, “b”, ..., y Ɛ

2. L = {$a^{i}b^{j}c^{k}$ | i=j o j=k, i, j, k > 0}

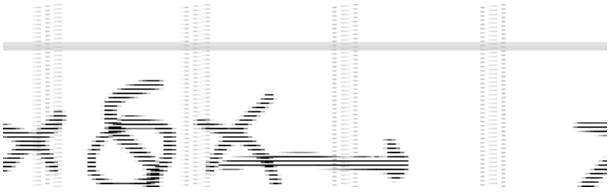


3. L = $\{a^i b^i a^j b^j \mid i, j > 0\}$

4. L = $\{a^i b^j c^k \mid k = i + (2 * j)\}$

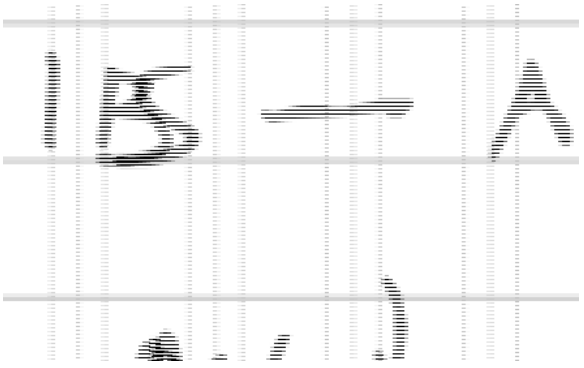
5. Lenguaje sobre el alfabeto {a, b} cuyas cadenas tengan una relación 2 a 1 entre el número de “a” y “b”

6. Lenguaje sobre el alfabeto {0, 1} cuyas cadenas cumplan que N(0) = N(1) + 1 (N(a) es el número de apariciones del símbolo “a”)

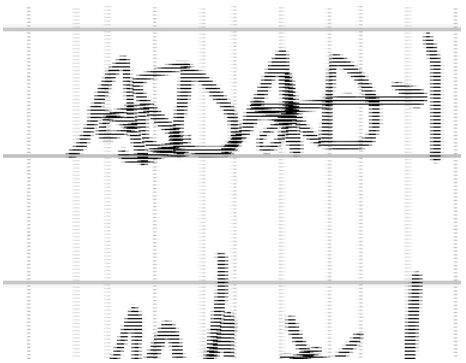


7. L = $\{a^i (b + c)^k \mid k > i\}$

8. L = $\{a^i b^j c^k \mid k = |i - j|\}$



9. L = $\{a^i (b + c)^j d^k \mid i+k > j;\ i, j, k \geq 0\}$

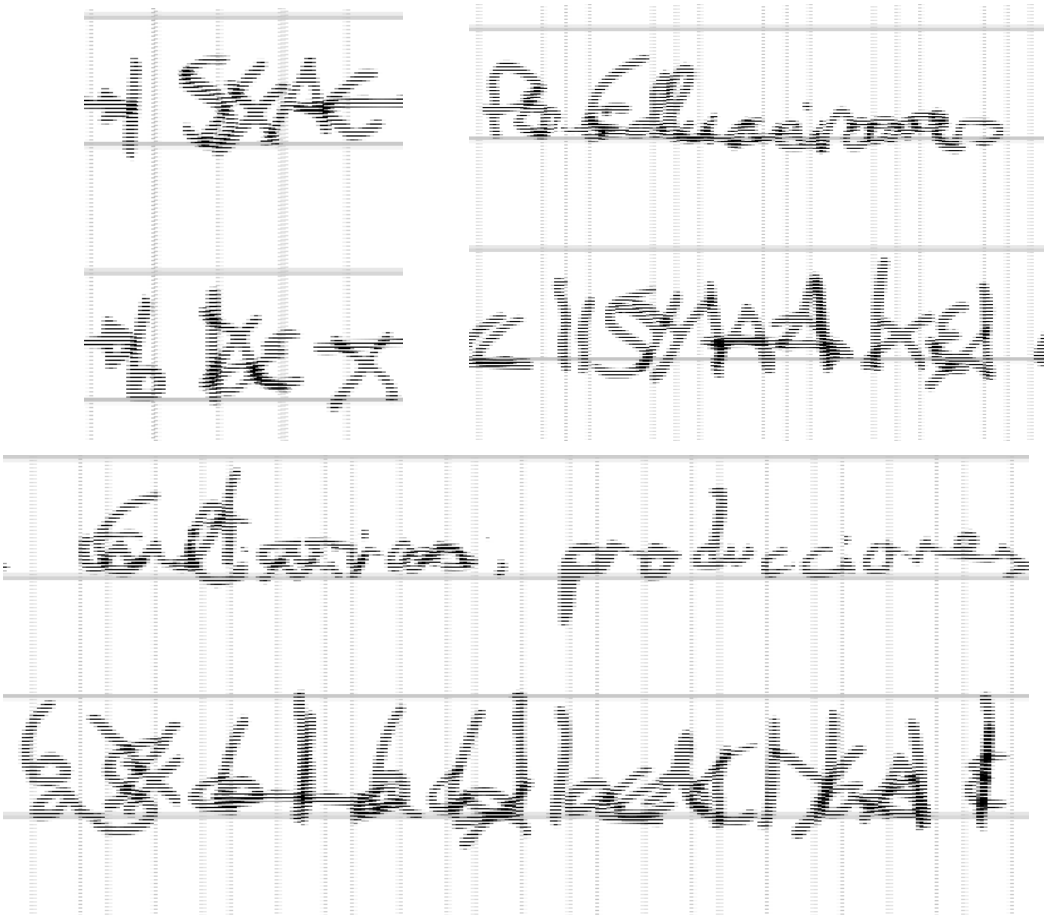


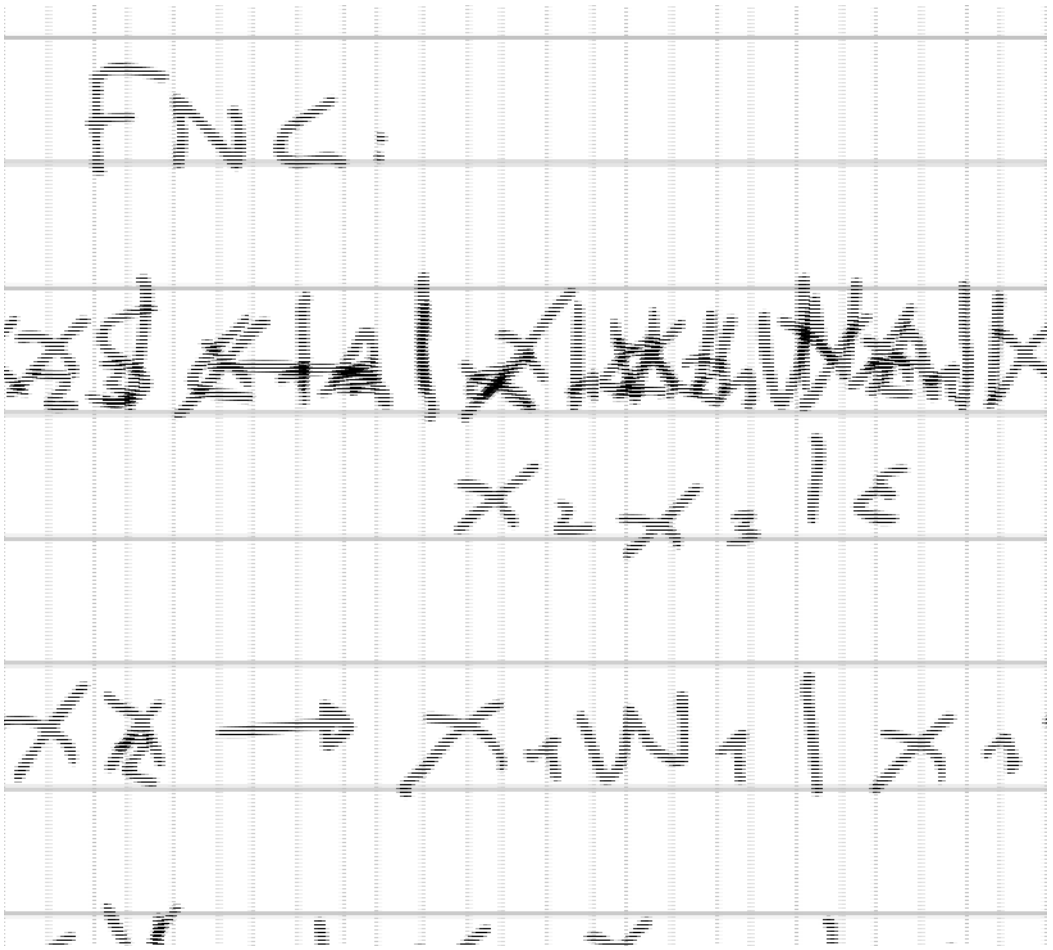
10. Lenguaje sobre el alfabeto {a, b, c} que cumple N(c) = N(a) + N(b), siendo N(x) el número de símbolos x en la cadena

Boletín 5: Diseño de GIC y Transformación a FNC

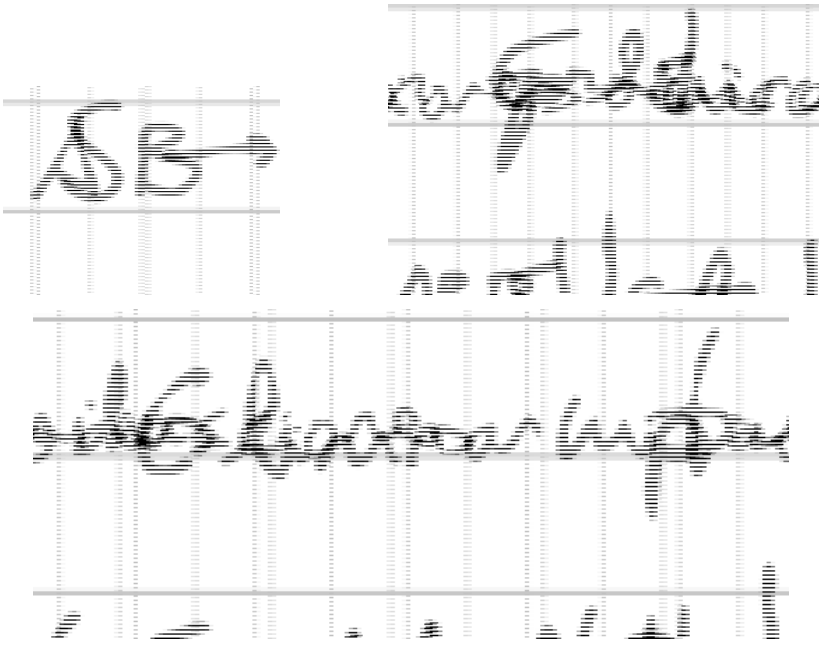
Será necesario entregar escaneados en formato PDF todos los cálculos realizados a mano, y comprobar con JFLAP que cada uno de los pasos es correcto. Diseñar las gramáticas independientes del contexto que generen los siguientes lenguajes. Además, en los tres primeros ejercicios debes transformar las GIC diseñadas a su Forma Normal de Chomsky (FNC).

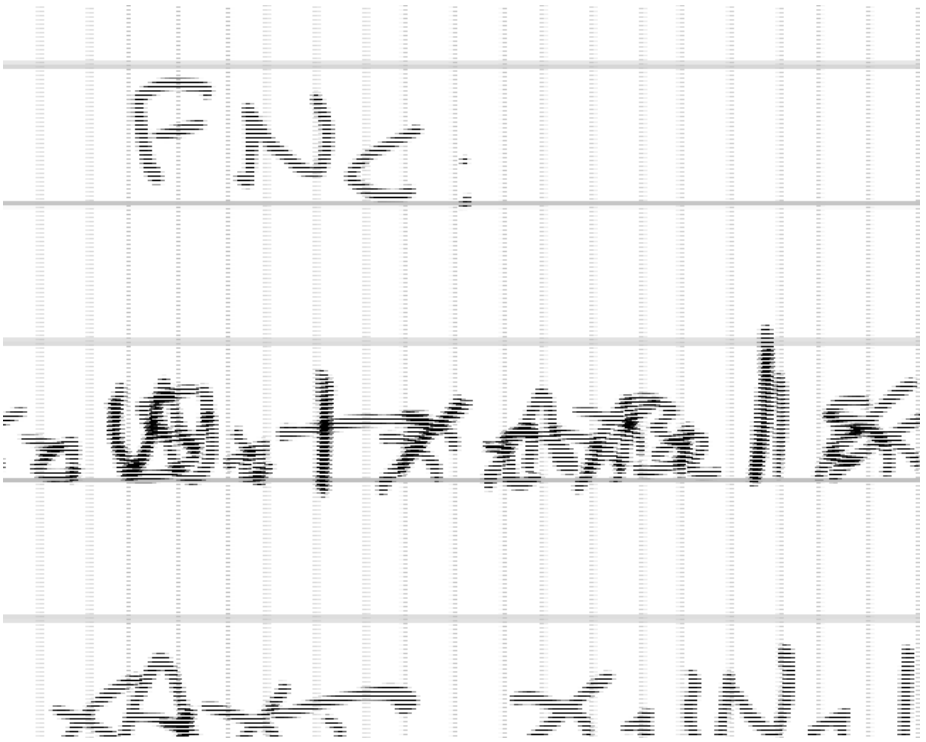
1. L = {$a^i b^j c^k$ | i = j o j ≤ k}



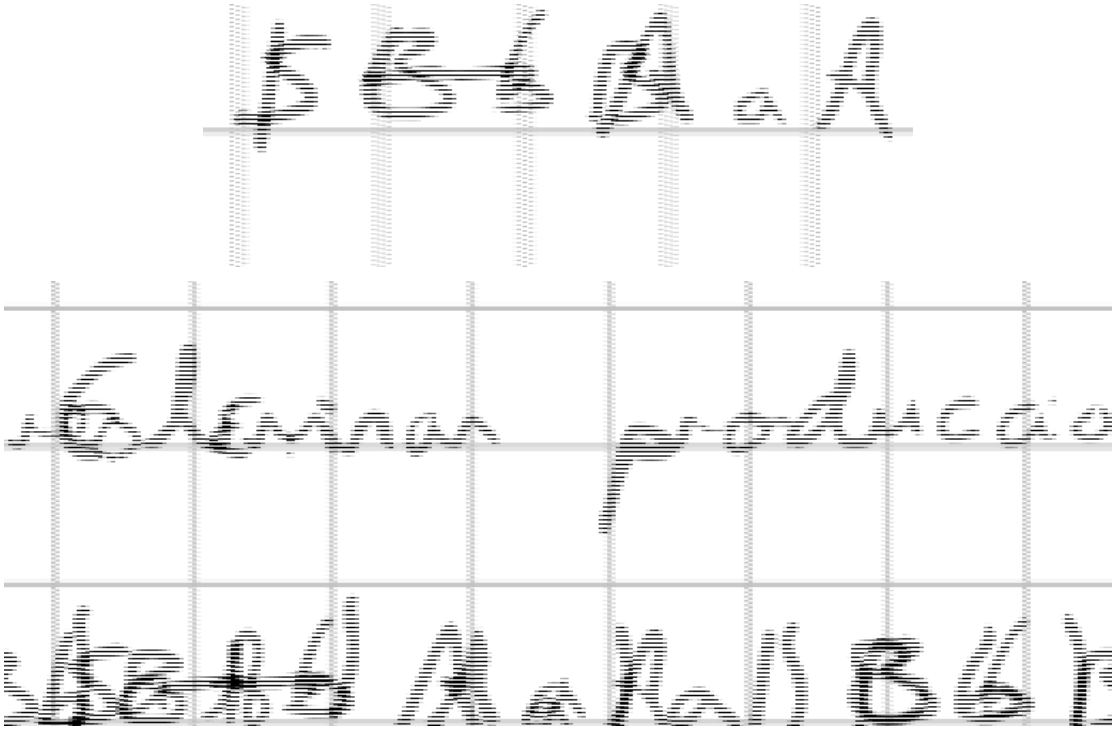


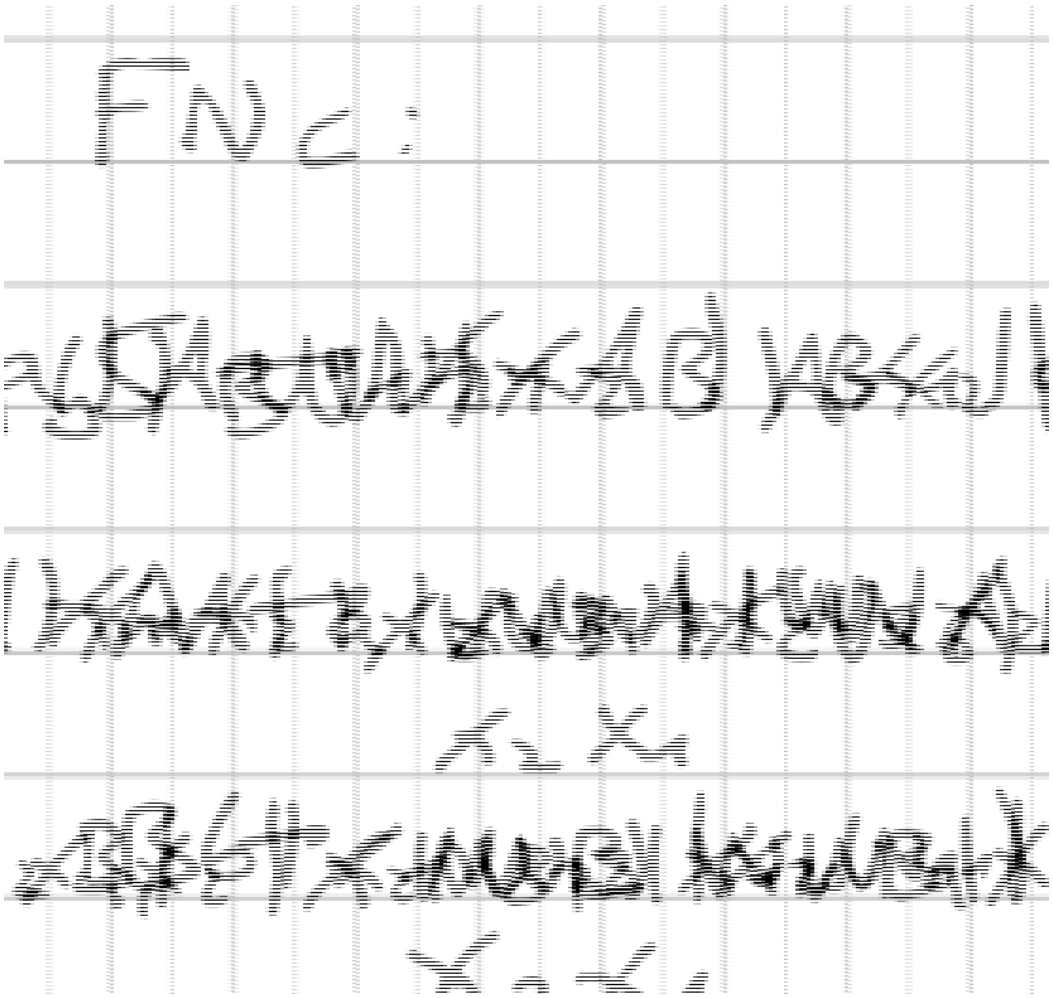
2. L = $\{a^i b^j a^k \mid j=i+k;\ i, j, k \geq 0\}$



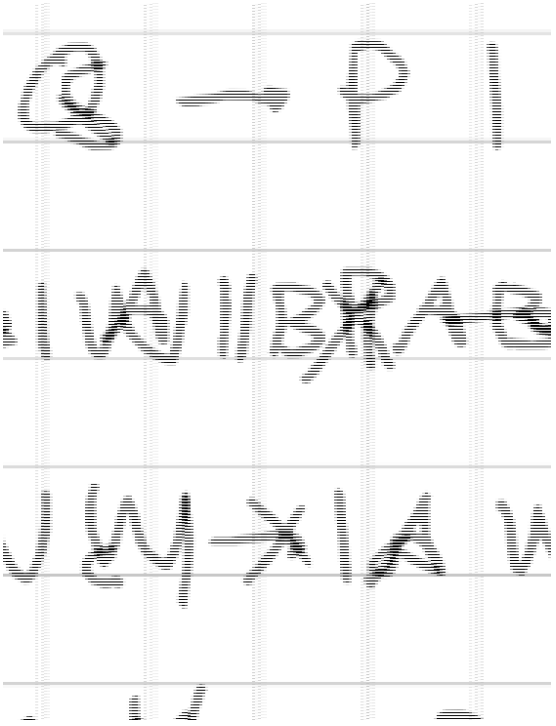


3. L = {w∈(a + b)* | N(a) ≠ N(b)}



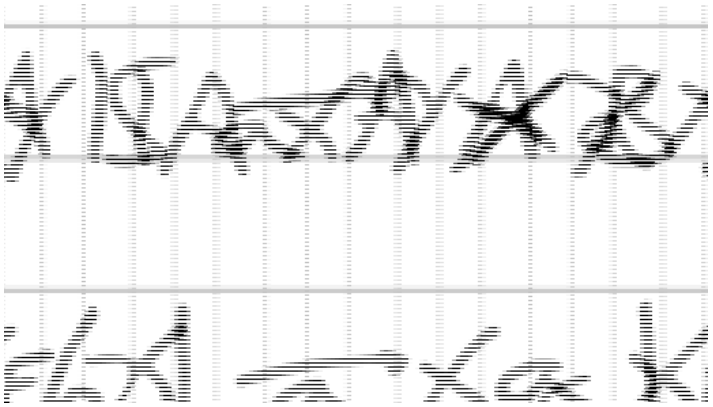


4. L = $\{a^i b^j c^k \mid k \neq i + j\}$

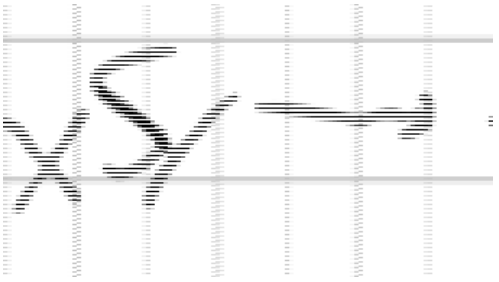


5. L = $\{a^i b^j c^k d^m \mid i > m \text{ y } j > k\}$

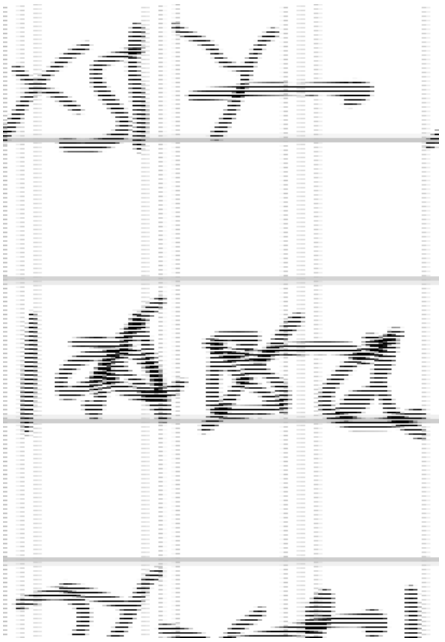
6. L = $\{a^i (b + c)^k a^j \mid k \neq i + j;\ i, j, k \geq 0\}$



7. L = $\{a^i b^j c^k a^m b^p \mid i = j+k \text{ y } m = p;\ i, j, k, m, p \geq 0\}$



8. L = $\{a^i b^j c^k d^m \mid i + k = j + m, \text{ o bien } i + j + k = m;\ i, j, k, m \geq 0\}$



Boletín 6: Diseño de Autómatas con Pila

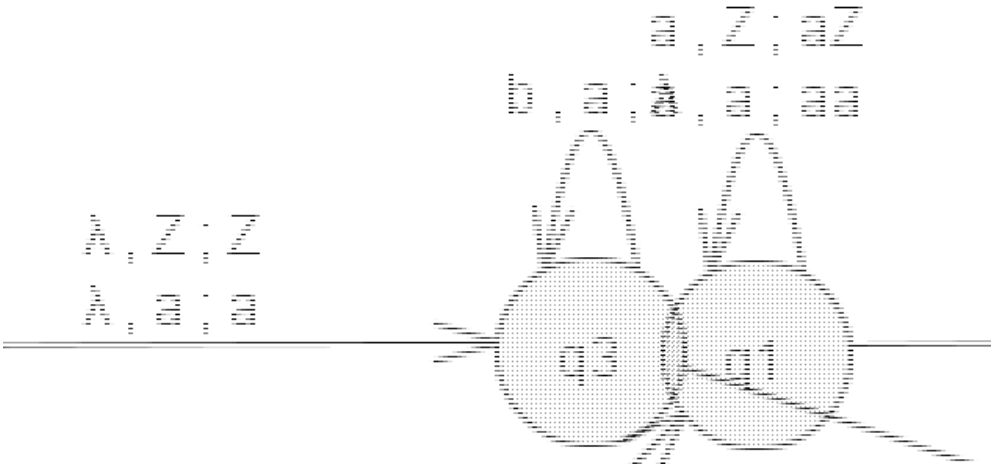
Diseñar los autómatas con pila que reconozcan los siguientes lenguajes. Restricción: el alfabeto de la pila será igual al alfabeto de entrada más el símbolo inicial de pila.

Por Estado Final

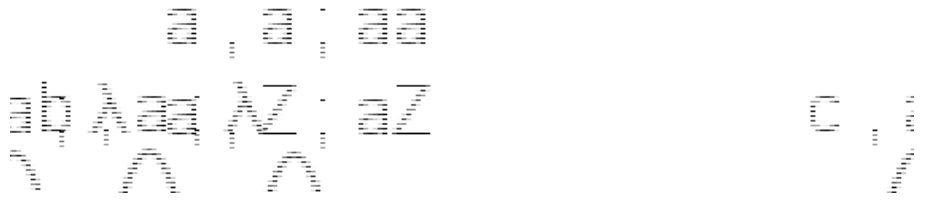
1. L = $\{a^i b^i c^k \mid k \geq 3\}$

2. L = $\{a^i b^j c^k \mid k > i + j\}$

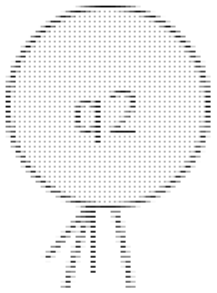
3. L = $\{a^i b^j c^k \mid i = j \text{ ó } j \neq k\}$



4. L = $\{a^i b^j c^k \mid k=i-j;\ i, j, k \geq 0\}$



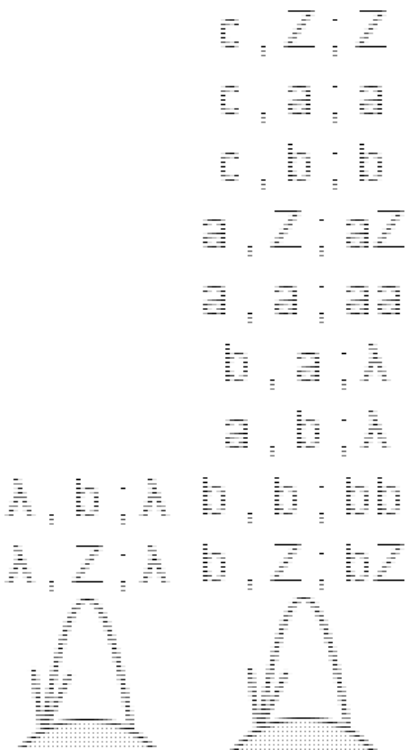
5. L = $\{a^i b^j c^k \mid N(a) = 2N(b) \text{ y } N(c) \text{ es impar}\}$



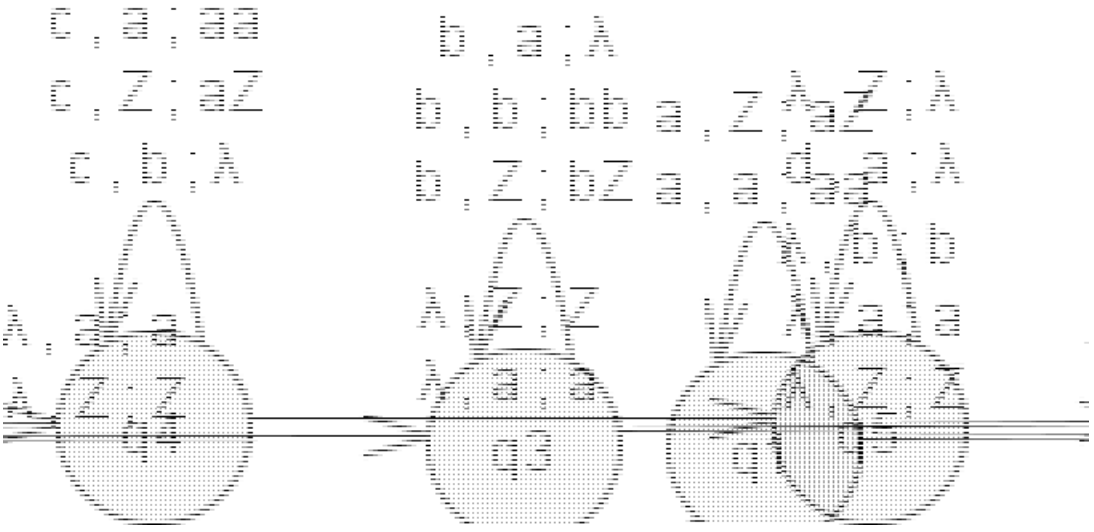
Por Vaciado de Pila

6. Lenguaje formado por aquellas cadenas que cumplen alguno de los siguientes criterios:
    a. N(c) = N(a) – N(b), entrando todos estos símbolos en cualquier orden
    b. $a^i b^j c^k$ | k – i < j

7. Lenguaje formado por aquellas cadenas que cumplen alguno de los siguientes criterios:
    a. N(a) < N(b), entrando los símbolos del alfabeto de entrada, {a, b, c}, en cualquier orden
    b. $a^i b^j c^k$ | 3k = i



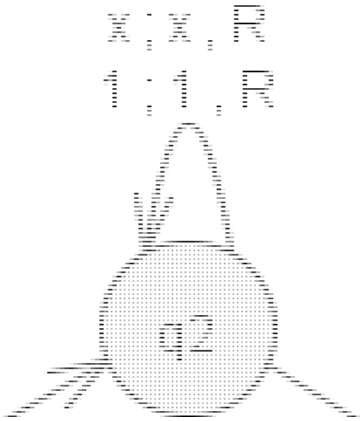
8. Lenguaje formado por aquellas cadenas que cumplen alguno de los siguientes criterios:
    a. $a^i b^j c^k d^m$ | i+k = j+m
    b. $a^i b^j c^k d^m$ | i+m = j+k



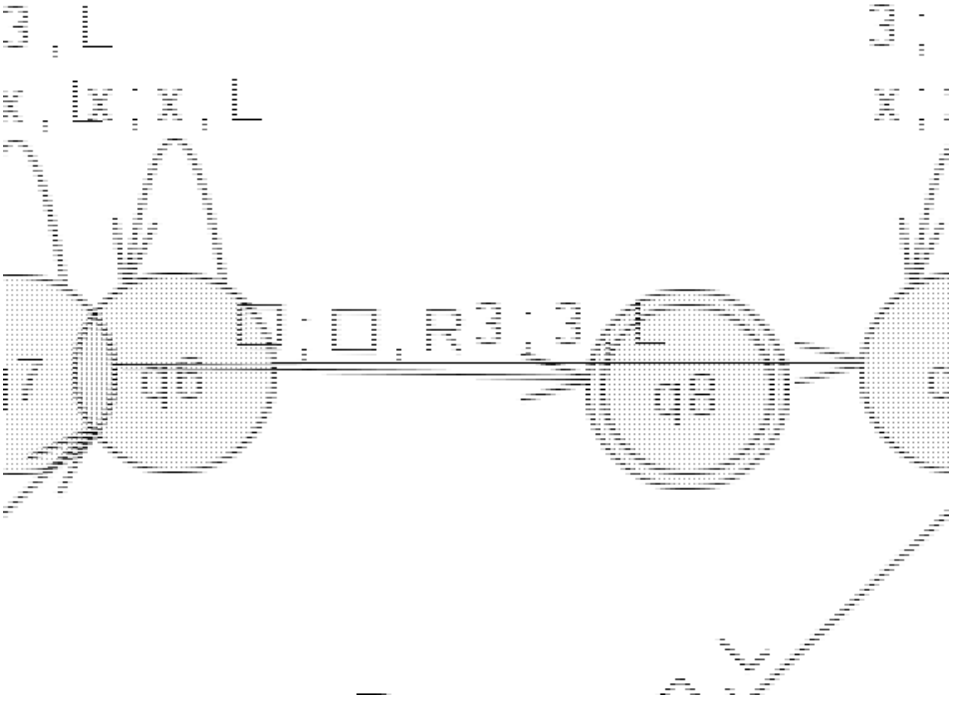
Boletín 7: Diseño de Máquinas de Turing Estándar

Diseñar las siguientes máquinas de Turing estándar:

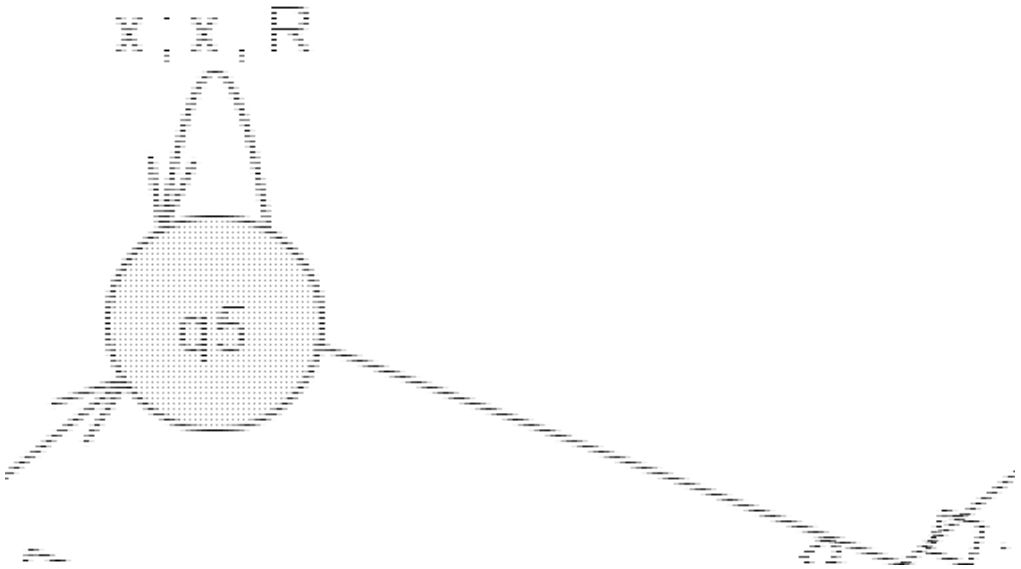
1. MT que reconozca el lenguaje L = {$1^{n+1}2^n3^{n-1}$ | n ≥ 1}, con ={1, 2, 3, x, B}



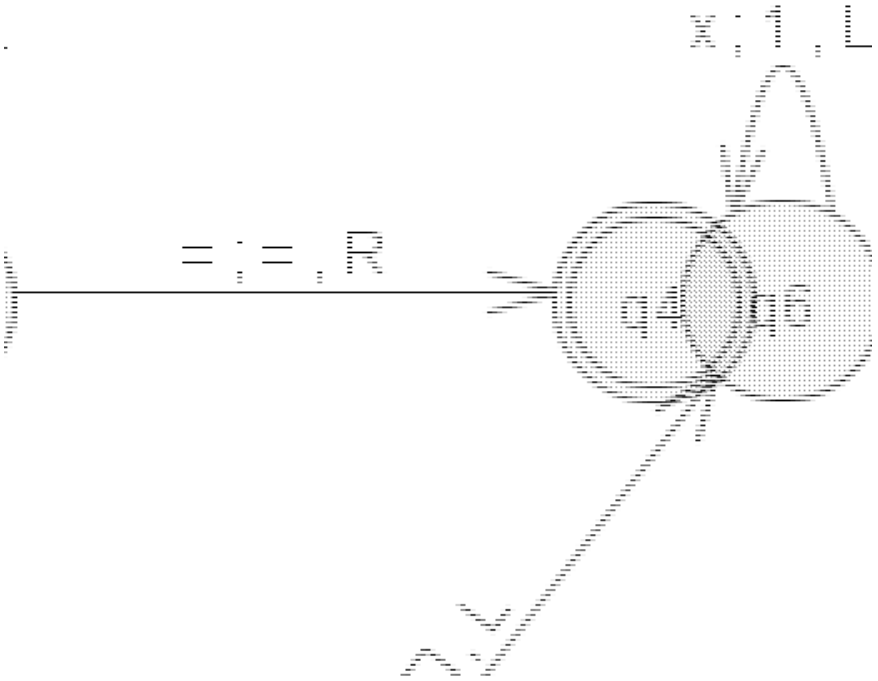
2. T que reconozca el lenguaje L = {w ∈ (1+2+3)*: N(1) = N(2) < N(3)}, con ={1, 2, 3, x, B}



3. MT que, dados dos números enteros (x, y), acepte cuando a ≥ b, con ={1, >, x, B}
    a. Ejemplo de contenido inicial de la cinta: “1111>11”
    b. La MT también debe reconocer el caso en el que ambos números son 0, es decir, cuando el contenido inicial de la cinta es “>”

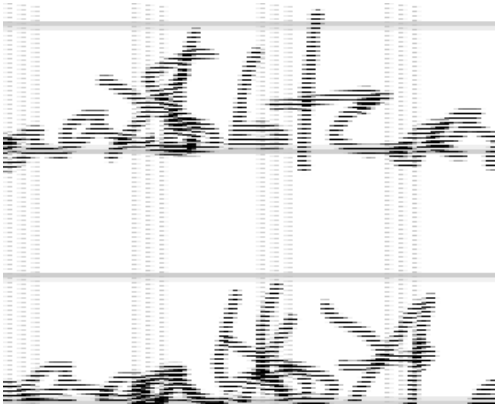


4. MT que convierta un número entero en formato unario a formato binario, con ={0, 1, =, x, B}
    a. Ejemplo de contenido inicial de la cinta: “=11111”
    b. Ejemplo de contenido final de la cinta: “101=11111”
    c. La MT también debe reconocer el caso en el que el número es 0, es decir, cuando el contenido inicial de la cinta es “=”

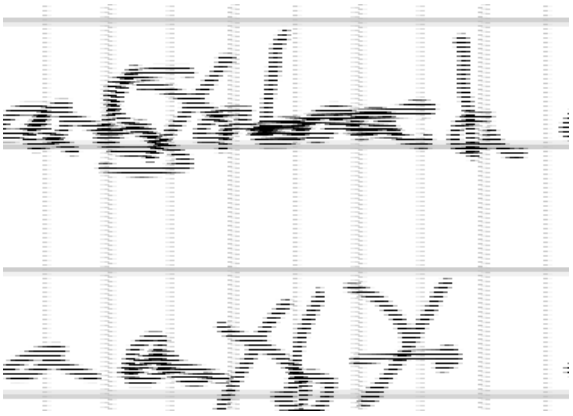


Boletín 9: Diseño de Gramáticas Sensibles al Contexto (GSC) y de Gramáticas Sin Restricciones (GSR)

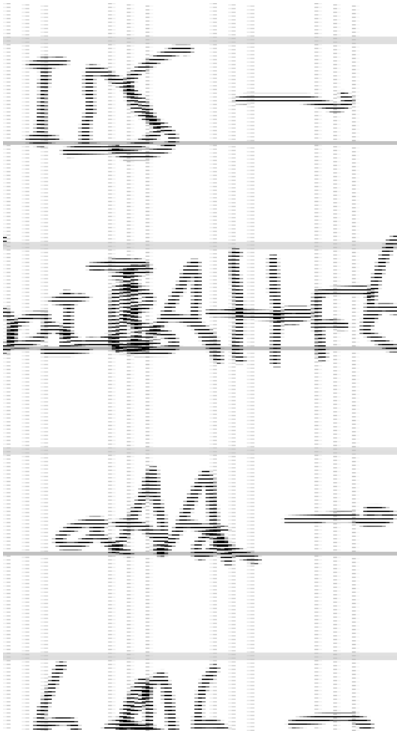
1. Diseñar la GSC que genere el lenguaje L = $\{a^{n+1}b^{n}c^{n-1} \mid n \geq 1\}$



2. Diseñar la GSC que genere el lenguaje L = $\{a^{n}b^{n}a^{2n} \mid n \geq 1\}$



3. Diseñar la GSR que genere el lenguaje L = {ww : w ∈ $\{a, b\}^{+}$}



4. Diseñar la GSC que genere el lenguaje L = {$w \in (a + b + c)^{+}$ | N(a) = N(b) < N(c)}