Teoría de Autómatas y Lenguajes Formales

Prueba de Evaluación Final

<u>Autores:</u>

Araceli Sanchis de Miguel
Agapito Ledezma Espino
Jose A. Iglesias Martínez
Beatriz García Jiménez
Juan Manuel Alonso Weber

UNIVERSIDAD CARLOS III DE MADRID
TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES.
GRADO EN INGENIERÍA INFORMÁTICA.

	APELLIDOS (En mayúsculas)			
	NOMBRE (En mayúsculas)			
	NIA		DNI	

INSTRUCCIONES PARA LA REALIZACIÓN DEL EXAMEN

- Para que una pregunta de test puntúe, todas las respuestas correctas deben estar marcadas y ninguna de las incorrectas lo ha de estar. En ningún caso las preguntas de test restan puntos.
- **Tiempo de examen (TEST + PROBLEMAS): 3 horas y media.**
- **A los 40 minutos del inicio se recogerán las hojas de TEST.**

TEST (Responder en la hoja del enunciado) 3 puntos.

1. Marque las afirmaciones verdaderas:
   a. **Si Σ={a, b, c} entonces Σ^0= λ.**
   b. Σ={a, b, λ} es un alfabeto.
   c. **Si Σ={a, b, c}, entonces {a, b, c} puede ser un lenguaje sobre Σ.**
   d. Si Σ={a, b, c}, entonces ∅ no es un lenguaje sobre Σ.

2. Marque las afirmaciones verdaderas:
   a. ABC::=AB es una regla de una gramática de tipo 1, sensible al contexto.
   b. **Toda gramática regular también es en una gramática independiente del contexto.**
   c. Toda gramática de tipo 0 puede transformarse en una gramática sensible al contexto equivalente.
   d. **Toda gramática de tipo 0 sin estructura de frase puede transformarse en una gramática equivalente de tipo 0 con estructura de frase.**

3. Marque las afirmaciones verdaderas:
   a. **Si la cardinalidad del conjunto cociente Q/E_0 es 3, es imposible que la cardinalidad de Q/E_1 sea 2.**
   b. Si $Q/E_0 = Q/E_1$ entonces el autómata finito es mínimo.
   c. En un autómata de 6 estados, Q/E_3 siempre es Q/E.
   d. Para determinar si dos autómatas finitos son equivalentes bastará saber si sus autómatas mínimos son isomorfos.

4. Marque las afirmaciones verdaderas:
   a. Podemos utilizar expresiones regulares para describir cualquier lenguaje independiente del contexto.
   b. Para que dos AFD sean equivalentes, su número de estados debe ser el mismo.
   c. Todo AFND puede transformarse en un AFD equivalente.
   d. Dado un AFD siempre es posible encontrar la expresión regular correspondiente al lenguaje que acepta.

5. Marque las afirmaciones verdaderas:
    a. Si α es una expresión regular, entonces $\alpha\alpha{*}=\alpha{*}$
    b. Si α es una expresión regular, entonces $\alpha\alpha{*}=\alpha{*}\alpha$
    c. $D_b(a^*(a+b)^*) = (a+b)^*$
    d. $\delta(a^*bb) = \lambda$

6. Marque las afirmaciones verdaderas:
    a. Existe algún Autómata a Pila (AP) capaz de reconocer el lenguaje vacío (L= ∅).
    b. Un AP puede carecer de estados finales.
    c. A partir de un AP que reconoce un lenguaje por estados finales puede construirse otro AP que reconoce un lenguaje por vaciado de pila.
    d. Un AP puede tener un solo estado.

7. Marque las afirmaciones verdaderas:
    a. El lenguaje $L=\{a^{n+2}b^n\}$ puede ser reconocido por un AP.
    b. Un lenguaje que tiene λ como una de sus palabras puede ser aceptado por algún AP.
    c. **Si en un AP $\Gamma=\{A\}$, Q={p,q}, siendo p estado inicial, entonces la correspondiente gramática tendrá entre sus reglas S::=(p, A, p) | (p, A, q)**
    d. f(p, x, R) = (p, R) corresponde al movimiento (p, xB, RP) |- (p, B, RP).

8. Marque las afirmaciones verdaderas:

   a. **$L((0+1)^*)=(L(0) \cup L(1))^*$.**

   b. **Si la ecuación característica correspondiente a un AF es**

   $X_1=1\ X_1+0\ X_2+0+1\ X_0$, entonces el autómata es no determinista.

   c. **Un autómata que acepta el lenguaje expresado por λ puede tener dos estados p (inicial) y q (final) con $f(p, \lambda)=q$.**

   d. Siendo α y β expresiones regulares: $(\alpha^*\beta)^*=\lambda+(\alpha\beta)^*\beta$.

9. Marque las afirmaciones verdaderas:

   a. **En una máquina de Turing transductora, si la entrada no está bien formada, debe acabar en estado no-final.**

   b. Toda Máquina de Turing tiene un AFD equivalente**.**

   c. Una máquina de Turing no puede modificar el contenido de la cinta.

   d. Una máquina de Turing puede desplazarse varias celdas a la vez después de leer un símbolo.

10. Marque las afirmaciones verdaderas:

   a. **En una máquina de Turing: f(p,a) → (p,a,i) indica que mientras lea "a", se sigue en el estado p, se re-escribe "a", y se mueve la cabeza de lectura a la izquierda.**

   b. En una máquina de Turing: f(p,a) → (p,a,i) indica que cuando se encuentra "a", se sigue en el estado p, se re-escribe "i", y se mueve la cabeza de lectura a la izquierda.

   c. En una máquina de Turing f(r,a) → (r,c,d) indica que se cambian las "*aes*" encontradas por "*ces*" y se cambia de estado.

   d. **En una máquina de Turing f(q,1) → (q,1,d) indica que siempre que se encuentra el símbolo "1" se re-escribe.**

	APELLIDOS (En mayúsculas)			
	NOMBRE (En mayúsculas)			
	NIA		DNI	

Problema 1 (2’5 puntos)

Las especificaciones de un lenguaje de programación son las siguientes:

- Todo programa empieza con la palabra reservada “***program***” y finaliza con la palabra reservada “***end***”.
- El lenguaje incluye cinco tipos de sentencias:
  - Lectura de una variable: ***read identificador;***
  - Escritura de una variable: ***write identificador;***
  - Sentencias de asignación: ***identificador := expresión_aritmética;***
    identificador := número_real;
  - Sentencias condicionales: ***if condicional then sentencias else sentencia(s) endIf***
    - condicional: ***identificador = expresión_aritmética***
      identificador = número_real
  - Sentencias de iteración: ***while condición do sentencia(s) endWhile***
- En estas dos últimas sentencias (condicional y de iteración), ***condicional*** es una única sentencia de condición.
- Las sentencias de lectura, escritura y deben acabar en “**;**”.
- Respecto a los tipos de datos:
  - Los identificadores consisten en una letra seguida o no de cualquier combinación de letras o dígitos.
  - Los números reales utilizan el “**.**” para separar la parte entera de la decimal.
  - Las expresiones aritméticas deben contener al menos uno de los operadores: “+”, “-“, “*” y “/”. Está permitido el uso de paréntesis siempre que el número de abiertos y cerrados esté emparejado.

Se pide:

1. Determinar una gramática para generar las **<u>palabras incluidas en el lenguaje definido</u>**, es decir, cualquier programa válido en este lenguaje de programación.

2. Sólo para reconocer **<u>expresiones aritméticas</u>** válidas, diseñar un autómata de pila por vaciado. Para el diseño de este AP, simplificar identificador y números reales como símbolos terminales de la gramática.

<u>SOLUCIÓN</u>

1. Determinar una gramática para generar las **palabras incluidas en el lenguaje definido**, es decir, cualquier programa válido en este lenguaje de programación.

```
PROGRAMA                ::= program INSTRUCCIONES end
INSTRUCCIONES           ::= INSTRUCCION ; INSTRUCCIONES
                        |  INSTRUCCION
INSTRUCCION             ::= LECTURA
                        |  ESCRITURA
                        |  ASIGNACION
                        |  CONDICIONAL
                        |  ITERACION
LECTURA                 ::= read IDENTIFICADOR
ESCRITURA               ::= write IDENTIFICADOR
CONDICIONAL             ::= if ASIGNACION then INSTRUCCIONES else
                           INSTRUCCIONES endIf
ITERACION               ::= while ASSIGMENT do INSTRUCCIONES endWhile
ASIGNACION              ::= IDENTIFICADOR = EXPRESION_ARITMETICA
                        |  IDENTIFICADOR = NUMERO_REAL
EXPRESION_ARITMETICA    ::= TERMINO | TERMINO RESTO_EXPRESION
RESTO_EXPRESION         ::= + EXPRESION_ARITMETICA
                        |   - EXPRESION_ARITMETICA
                        |   / EXPRESION_ARITMETICA
                        |  * EXPRESION_ARITMETICA
TERMINO                 ::= ( EXPRESION_ARITMETICA )
                        |  IDENTIFICADOR
                        |  NUMERO_REAL
IDENTIFICADOR           ::= LETRA
                        |  LETRA RESTO_IDENT
REST0_IDENT         ::= LETRA RESTO_IDENT
                        |  DIGITO RESTO_IDENT
                        |  LETRA
                        |  DIGITO
LETRA               ::= a | b |c | … | z | A | B |C |…| Z
DIGITO              ::= 0 | 1 | 2 | … | 9
NUMERO_REAL                     ::= DIGITOS
                        |  DIGITO . DIGITOS
DIGITOS                 ::= DIGITO DIGITOS
                        |  DIGITO
```

2. Sólo para reconocer **expresiones aritméticas** válidas, diseñar un autómata de pila por vaciado. Para el diseño de este AP, simplificar identificador y números reales como símbolos terminales de la gramática.

Nos quedamos con esta parte de la gramática:

EXPRESION_ARITMETICA ::= TERMINO | TERMINO RESTO_EXPRESION
RESTO_EXPRESION ::= + EXPRESION_ARITMETICA
| - EXPRESION_ARITMETICA
| / EXPRESION_ARITMETICA
| * EXPRESION_ARITMETICA
TERMINO ::= (EXPRESION_ARITMETICA)
| identificador
| numero_real

o la que es equivalente en GNF:
E (expresión_aritmética)
R (resto_expresion)
i (identificador)
n (número_real)

E ::= (E C | i | n | (E C R | i R | n R
R ::= + E | -E | / E | * E
C ::=)

El autómata de pila equivalente por vaciado de pila es:

$$AP_v = (\{i, n\}, \{E, R\}, \{q\}, E, q, f, \Phi)$$

con función de transición:

f(q,’(‘,E) = (q, EC)
f(q,i,E) = (q, λ)
f(q,n,E) = (q, λ)
f(q,’(‘,E) = (q, ECR)
f(q,i,E) = (q, R)
f(q,n,E) = (q, R)
f(q,+,R) = (q, E)
f(q,-,R) = (q, E)
f(q,/,R) = (q, E)
f(q,*,R) = (q, E)
f(q,’)’,C) = (q, λ)

Problema 2 (2'25 puntos)

Dada la siguiente gramática **G1=({1,2}, {A, M, N}, A, P)** siendo el conjunto de producciones P las siguientes:

P = { A → MN | M
N → 1
M → N | 1M21 | 121
N → 2M1 | λ
M → 2 }

Se pide:

1. Contextualizar la gramática dentro de la Jerarquía de Chomsky.
2. Identificar aquellas producciones de G1 que son válidas en una gramática de tipo 3.
3. Quitar las reglas identificadas en el apartado 2 (sin sustituirlas por ninguna otra), y generar así una nueva gramática (G2) que no será equivalente.
4. Limpiar y bien formar la gramática resultante (G2) y obtener a partir de ella las gramáticas equivalentes en FNC (G3) y FNG (G4).
5. Comprobar que las palabras: λ, 121 y 12121211 pueden ser generadas por las gramáticas G2, G3 y G4.

SOLUCIÓN:

1. Tipo 2

2. La gramática sería:

A → MN | M
M → N | 1M21 | 121
N → 2M1 | λ

3. Limpiamos y convertimos
a) eliminar lambda:
A → MN | M
M → λ | 1M21 | 121 | N
N → 2M1

b) eliminar lambda:
A → MN | M | N | λ
M → 1M21 | 121 | N
N → 2M1 | 21

c) eliminar redenominaciones:
A → MN | 1M21 | 121 | 2M1 | 21 | λ
M → 1M21 | 121 | 2M1 | 21
N → 2M1 | 21

d) conversión a FNC

A → MN | CD | XD | YF | YX | λ
M → CD | XD | YF | YX
N → YF | YX
C → XM
D → YX
F → MX
X → 1
Y → 2

e) conversión a FNG

A → MN | 1M21 | 121 | 2M1 | 21 | λ
M → 1M21 | 121 | 2M1 | 21
N → 2M1 | 21

- TEOREMA: todo **L de contexto libre sin λ** puede ser generado por una G2 en la que todas las reglas sean de la forma:
  - **A → a**α donde $A \in \Sigma_{NT}$ $a \in \Sigma_T$ $\alpha \in \Sigma_{NT}{}^*$
  - Si ∈λL habrá que añadir **S::= λ**

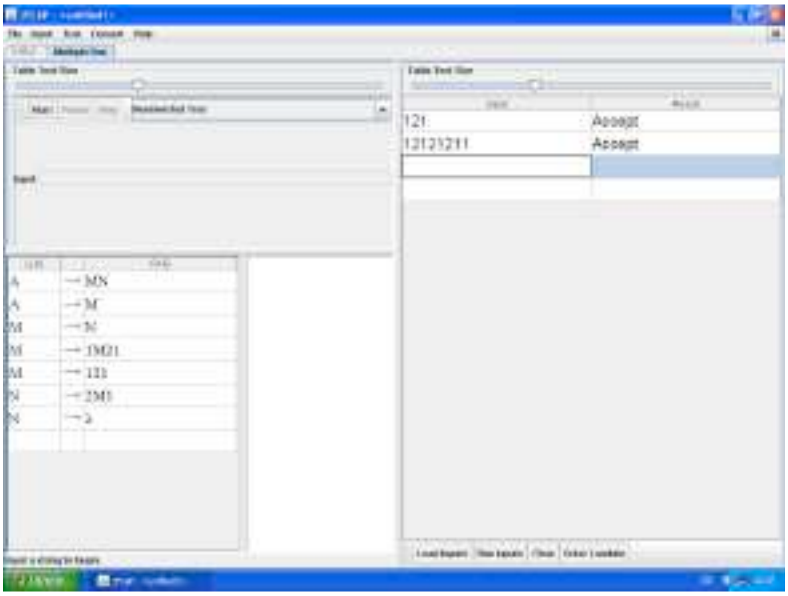
-No hay recursividad a izquierdas
-Establecemos orden de no terminales Σ_{NT} = {A, M, N}
- Reglas en FNG:

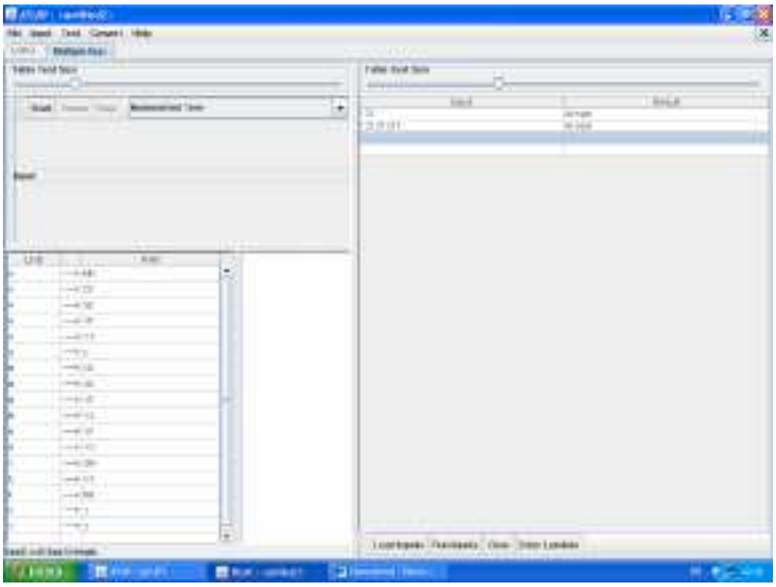
A → λ
- Sustituimos partes derechas de M en la regla A →MN
A →1M21N/121N/2M1N/21N/1M21/121/2M1/21
M → 1M21 | 121 | 2M1 | 21
N → 2M1 | 21

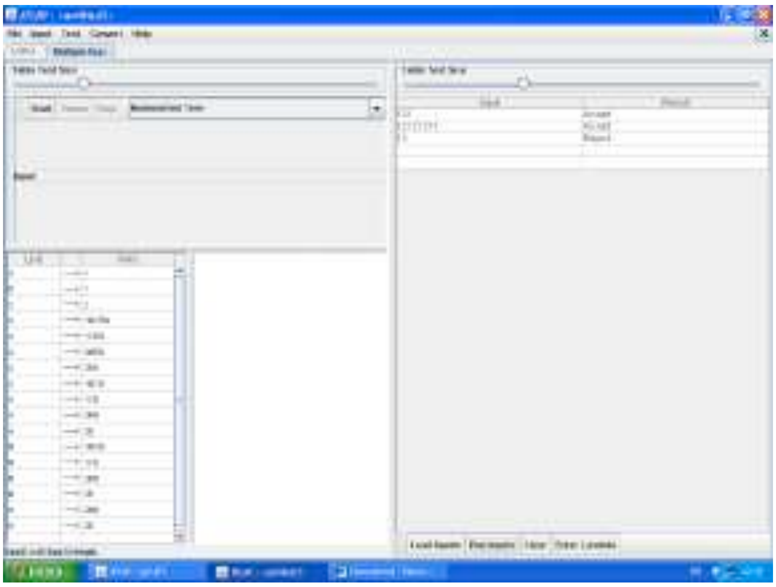
-Creamos dos reglas B → 1 y C→ 2 para convertir las reglas a tipo 1 y sustituimos quedando la gramática en FNG:

A → λ
B → 1
C→ 2
A →1MCBN/1CBN/2MBN/2BN/1MCB/1CB/2MB/2B
M → 1MCB | 1CB | 2MB | 2B
N → 2MB | 2B

- Comprobar Palabras:


121 Accept
12121211 Accept
A → MN
A → M
M → N
M → 1M21
M → 121
N → 2M1
N → λ





Problema 3 (2'25 puntos)

Diseñar una Máquina de Turing que calcule el cociente de la división entre números naturales en codificación unaria. Considerar que el dividendo **siempre** es mayor que el divisor, y que **la división es siempre exacta** (sin resto).
Utilizar la codificación unaria en la que: 1 se representa como 1, 2 se representa como 11, 3 se representa como 111, etc. En la cinta inicialmente habrá *dividendo÷divisor*, y deberá finalizar con *dividendo÷divisor=cociente*. Es decir, se debe preservar el contenido original de la cinta.

Por ejemplo, si al inicio hubiera: **1111÷11**
le correspondería el fin: **1111÷11=11**
(*en codificación decimal: 4÷2=2*);
Y si hubiera al inicio: **111111111÷111**
le correspondería el fin: **111111111÷111=111**
(*en codificación decimal: 9÷3=3*);

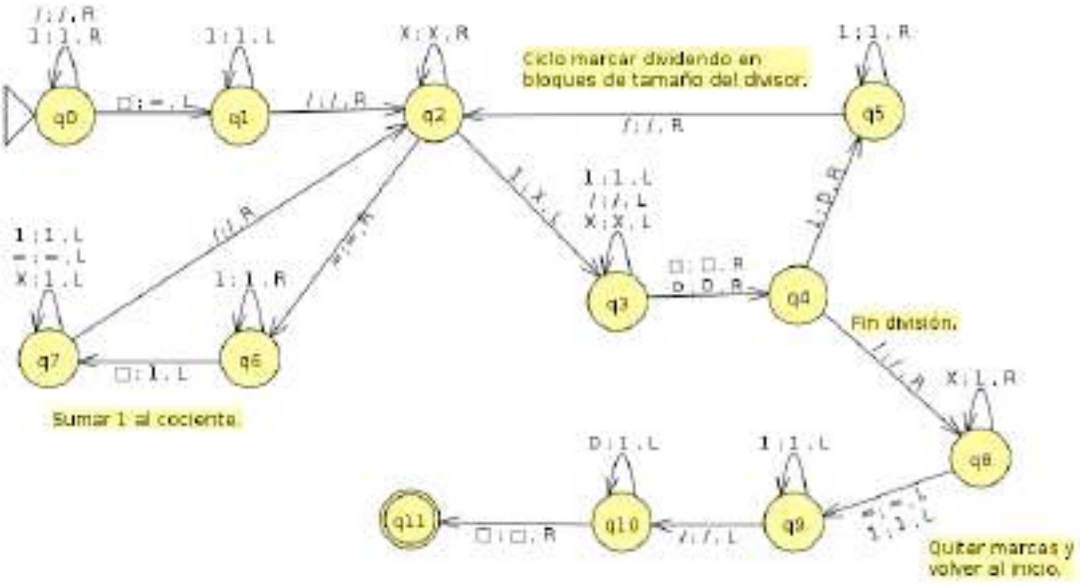
Se pide:

a) Definición formal de la séptupla de la Máquina de Turing. Incluir el diagrama de transiciones (no la lista, ni la tabla de la función de transición).
b) Descripción detallada del algoritmo implementado en la Máquina de Turing.
c) Explicación del significado de:
    - cada símbolo del alfabeto de la cinta no definido en este enunciado,
    - cada uno de los estados y transiciones, o grupo de estados y transiciones.

SOLUCIÓN

a) Definición formal:

MT=({1,÷}, {1,÷,=,□,X,D}, □, {q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11 }, q0, f, {q11}),
donde f (con cabeza de la pila al final de la palabra) queda representado en el siguiente diagrama de transiciones:



b) Descripción detallada del algoritmo implementado en la Máquina de Turing:

Idea general: Ir marcando los dígitos del dividendo en grupos del tamaño indicado por el divisor. Por cada grupo, sumar 1 al cociente. Repetir hasta que se hayan marcado todos los dígitos del dividendo. Al final, desmarcar todos los dígitos de dividendo y divisor, y situar puntero al inicio.

Algoritmo detallado:

1. Marcar final de la entrada de la cinta con el símbolo "=".
2. Retroceder hasta el símbolo de la división (÷), para localizar el inicio del divisor (una posición a la derecha).
3. Recorrer cada dígito del divisor (marcando con una X), paralelamente a los del dividendo (marcando con una D), hasta alcanzar tantos dígitos como indique el divisor.
4. Al llegar al final del divisor (identificado por el símbolo "=" en la cinta), sumar uno a la cantidad existente en el cociente; es decir, añadir un 1 detrás del último símbolo escrito en la cinta.
5. Retroceder hasta el símbolo ÷, cambiando las X por 1 en el divisor, pero sin desmarcar el dividendo, porque no ha finalizado la división.
6. Volver al paso 2, hasta que todos los símbolos del dividendo estén marcados.
7. Retroceder hasta el inicio cambiando por 1 las marcas "X" en el divisor y "D" en el dividendo, hasta dejar el puntero de la cinta al inicio del dividendo.

c) Explicación del significado de:

- **cada símbolo del alfabeto de la cinta no definido en este enunciado:**

☐: símbolo blanco, que indica que la celda está vacía.

X: marca de dígitos del *divisor* recorridos.

D: marca de dígitos del *dividendo* recorridos.

(=: símbolo que separa el divisor del cociente en el contenido de salida de la cinta. Ya definido en el enunciado.)

- **cada uno de los estados y transiciones, o grupo de estados y transiciones:**
- **q0** [paso 1 del algoritmo]: recorrer la cinta hasta el final, y escribir símbolo "=" al transitar a q1.
- **q1** [paso 2 del algoritmo]: retroceder hasta el inicio del divisor, casilla a la derecha del símbolo ÷.
- **q2**: inicio ciclo marcar dígito divisor-dividendo, o inicio sumar 1 al cociente.
- **ciclo q2-q5** [paso 3 del algoritmo]: en cada vuelta se marca con "D" un dígito del dividendo por cada dígito del divisor a marcar con "X". De q2 a q3 se marca el divisor. Entre q3 y q4 se retrocede hasta último dígito del dividendo no marcado. De q4 a q5 se marca el dividendo. Entre q5 y q2 se avanza hasta último dígito del divisor no marcado.
- **ciclo q2-q7** [pasos 4, 5 y 6 del algoritmo]: en cada vuelta, se incrementa en 1 el cociente. De q2 a q6 se salta el símbolo de separación de divisor y cociente (=). En q6 se avanza hasta el final de la cinta. De q6 a q7 se incrementa el cociente añadiendo un 1 al final. En q7 se retrocede hasta el inicio del divisor, desmarcando todos sus dígitos (cambiando X por 1). De q7 a q2, se sitúa el puntero de nuevo al inicio del divisor, para continuar con la división, en otro ciclo q2-q5.
- **q4, q8** [paso 7 del algoritmo, primera parte]: al detectar que todos los símbolos del dividendo están marcados, ha finalizado la división; y en q8 se desmarcan todos los símbolos del divisor (cambiando X por 1), avanzando hacia la derecha.
- **q8-q11** [paso 7 del algoritmo, segunda parte]: con el divisor sin marcas, se recorre toda la cinta hasta el inicio del dividendo, quitando las marcas del dividendo (cambiando D por 1), hasta llegar a la cabeza y situar allí el puntero (transición de q10 a q11), como pide el enunciado.

Imagen con varias palabras verificadas:

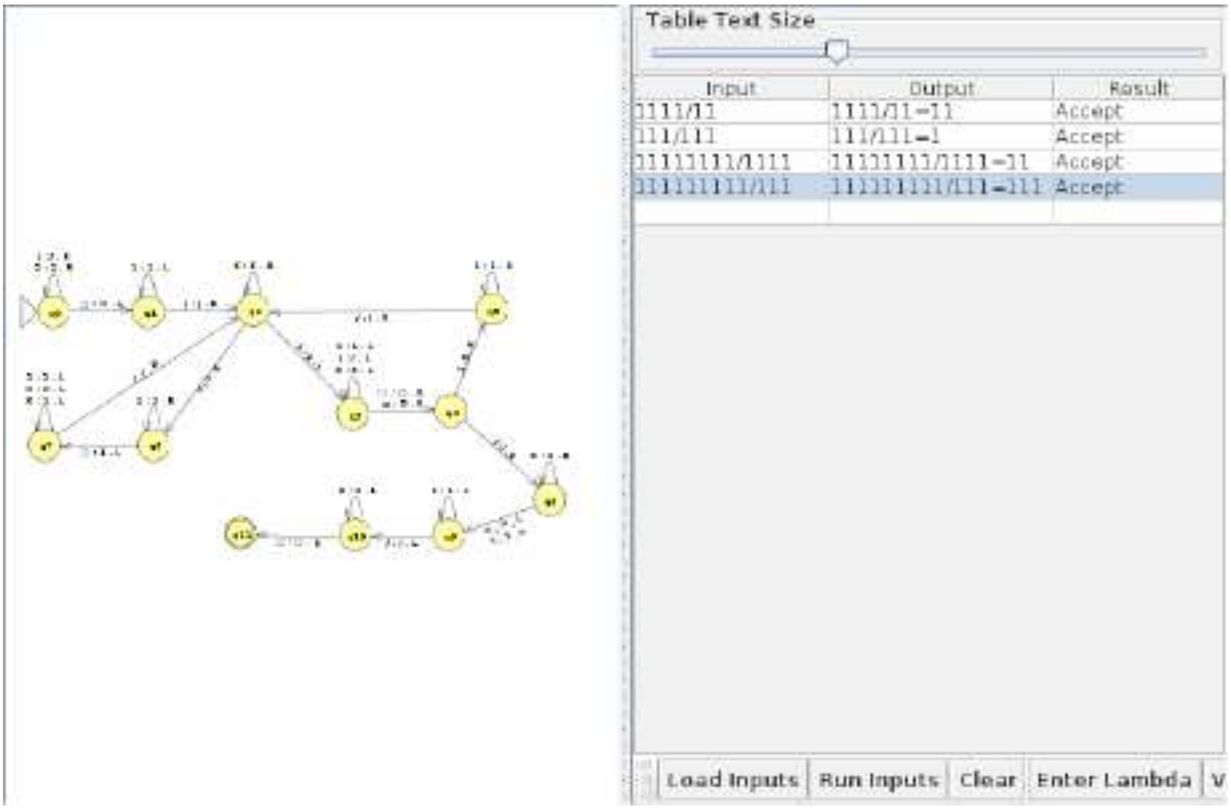


Table Text Size
Input
Output
Result
1111/11
1111/11=11
Accept
111/111
111/111=1
Accept
11111111/1111
11111111/1111=11
Accept
111111111/111
111111111/111=111
Accept
Load Inputs
Run Inputs
Clear
Enter Lambda