INFG-TALF Teoría de Autómatas y Lenguajes Formales

Ejercicios de Autómatas Finitos (T3-P1)

1. Sea el alfabeto {0,1}. Construir un autómata finito determinista que reconozca aquellos elementos del lenguaje universal que tengan un número de ceros (0) impar.

2. Sea el alfabeto de símbolos {a,b}. Desarrollar un autómata que reconozca cadenas de longitud "3" del lenguaje universal. (A realizar después de Tema 4: Obtener la G3 correspondiente a dicho Autómata.)

4. Sea el alfabeto {a,b}. Diseñar un Autómata Finito Determinista que reconozca aquellas secuencias que tienen un número par de "a" y un número impar de "b".

5. Sea el alfabeto {0,1}. Diseñar un Autómata Finito Determinista que reconozca el lenguaje L de aquellas cadenas que tiene el mismo número de apariciones de la subcadena 01 que la subcadena 10. Ejemplos: 101 pertenece L (**10**1, 1**01**); sin embargo 1010 no pertenece a L (**10**10, 1**01**0, 10**10**).

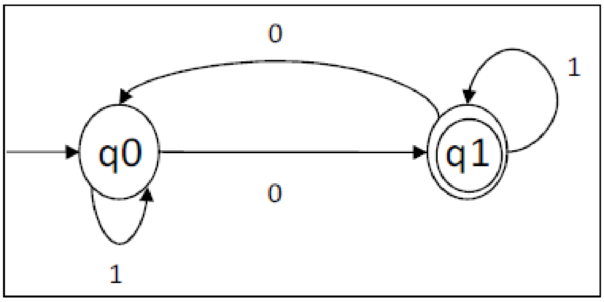
6. Se desea diseñar un dispositivo que, dada una cadena formada por números binarios, encuentre las ocurrencias de la palabra clave 1011 y sirva de base para un recuento de sus apariciones. Nótese que si la cadena fuera, por ejemplo, 010101101011, se detectaría dos ocurrencias de la palabra clave (subrayadas), no considerando el "1" de la séptima posición como inicio de otra ocurrencia. Se pide construir el Autómata Finito Determinista correspondiente.

7. En algunos lenguajes de programación, los comentarios aparecen entre los delimitadores "/*" y "*/" como marca inicial y final del comentario. Sea L el lenguaje de todas las cadenas de comentarios delimitados. Así pues todo elemento de L, empieza por /* y acaba por */, pero no debe tener ningún */ intermedio. Por simplicidad consideraremos que el alfabeto sería {a, b, /,*}. Indicar el Autómata Finito Determinista que reconoce L.

Ejercicios de Autómatas Finitos (T3-P1)
SOLUCIONES

1. **Sea el alfabeto {0,1}. Construir un autómata finito determinista que reconozca aquellos elementos del lenguaje universal que tengan un número de ceros (0) impar.**

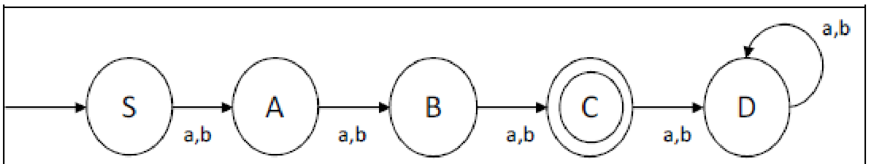
Solución:



---

2. **Sea el alfabeto de símbolos {a,b}. Desarrollar un autómata que reconozca cadenas de longitud "3" del lenguaje universal. Obtener la G3 correspondiente a dicho Autómata.**

Solución:

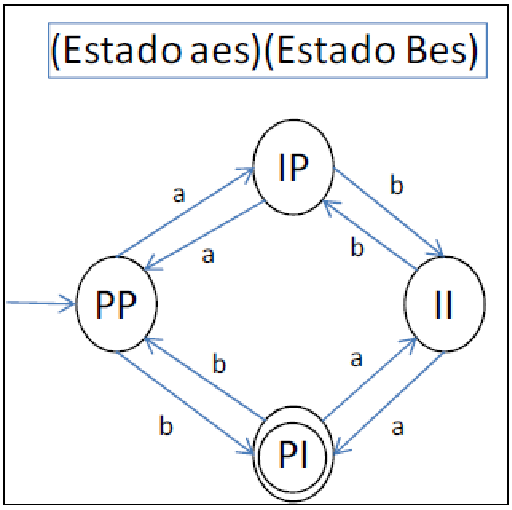


S::=aA | bA
A::=aB | bB
B::=aC | bC
B::=a | b
C::=aD | bD
D::=aD | bD

Gramática Bien Formada
S::=aA | bA
A::=aB | bB
B::=a | b

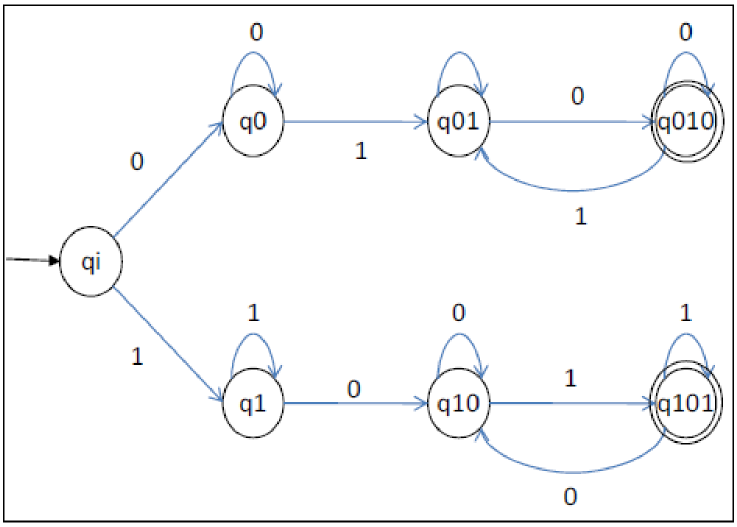
4. Sea el alfabeto {a,b}. Diseñar un Autómata Finito Determinista que reconozca aquellas secuencias que tienen un número par de "a" y un número impar de "b".

Solución:



5. Sea el alfabeto {0,1}. Diseñar un Autómata Finito Determinista que reconozca el lenguaje L de aquellas cadenas que tiene el mismo número de apariciones de la subcadena 01 que la subcadena 10. Ejemplos: 101 pertenece L (<u>10</u>1, 1<u>01</u>); sin embargo 1010 no pertenece a L (<u>10</u>10, 10<u>10</u>, 1<u>01</u>0).

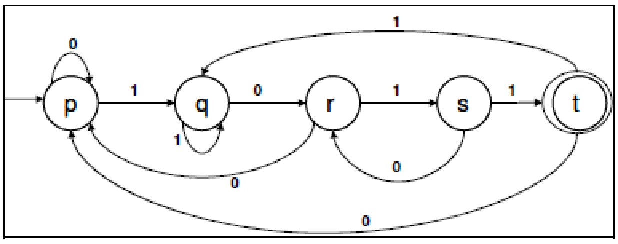
Solución:



6. Se desea diseñar un dispositivo que, dada una cadena formada por números binarios, encuentre las ocurrencias de la palabra clave 1011 y sirva de base para un recuento de sus apariciones. Nótese que si la cadena fuera, por ejemplo, 0101011011011, se detectaría dos ocurrencias de la palabra clave (subrayadas), no considerando el "1" de la séptima posición como inicio de otra ocurrencia. Se pide construir el Autómata Finito Determinista correspondiente.

Solución:

Un posible autómata es:



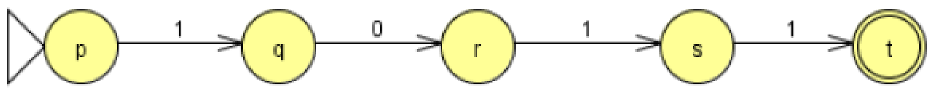
El objetivo es detectar la palabra clave 1011, tantas veces como aparezca en la secuencia de entrada de números binarios. Por lo tanto, el autómata finito deberá llegar a un estado final cada vez que detecte una ocurrencia de dicha palabra clave, por lo que no puede quedarse en el estado final tras encontrar la primera secuencia 1011, si sigue habiendo números detrás.
No es tarea del autómata finito contabilizar el número de palabras clave, sino de otro dispositivo de orden superior, que incluya este autómata, que las detecta una a una.

Todo AFD está compuesto por una quíntupla: AFD=(Σ, Q, f, q0, {F})

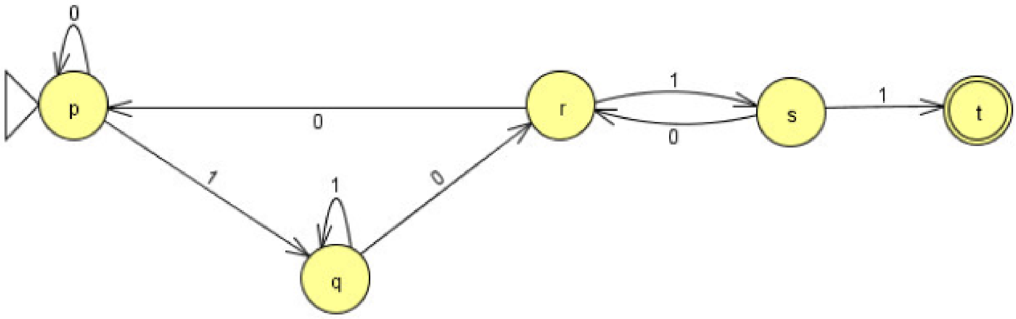
- El alfabeto de entrada, en este caso es sencillo, pues sólo contiene los dígitos 0 y 1 que forman los binarios: AFD=(**{0,1}**, …)
- El conjunto de estados Q se irá definiendo más adelante en función de las transiciones necesarias, pero al menos necesitaremos un estado inicial, p, y un estado final, t, que ya podemos añadir al conjunto: AFD=({0,1}, **{p, t, …}**,…)
- La función de transición, f, la representaremos más adelante mediante un diagrama de transiciones: AFD=({0,1}, {p, t, …}, **f**, …)
- El estado inicial ya está definido, denominándose p: AFD=({0,1}, {p, t, …}, f, **p**, …)
- Por último, el conjunto de estados finales se completará al final, pero al menos podemos incluir el que ya hemos definido, llamado t: AFD=({0,1}, {p, t, …}, f, p, **{t, …}**)

Por lo tanto, nos falta definir el diagrama de transiciones que representa a f, que también nos permitirá completar el conjunto de estados y de estados finales.

Dado que la palabra clave está formada por 4 símbolos, que deben aparecer siempre de forma consecutiva, necesitaremos un estado diferente para reconocer cada subsecuencia de símbolos leídos de la cadena. Es decir, tendremos un estado (q) para determinar que se ha leído la subsecuencia "1", otro estado (r) para indicar la lectura de la subsecuencia "10", un estado más (s) para representar el reconocimiento de la subsecuencia con 3 elementos "101", y por último se alcanza el estado final (t) cuando se ha leído la secuencia completa "1011" de forma consecutiva. Así, de momento, nos quedaría el siguiente diagrama de transciones:

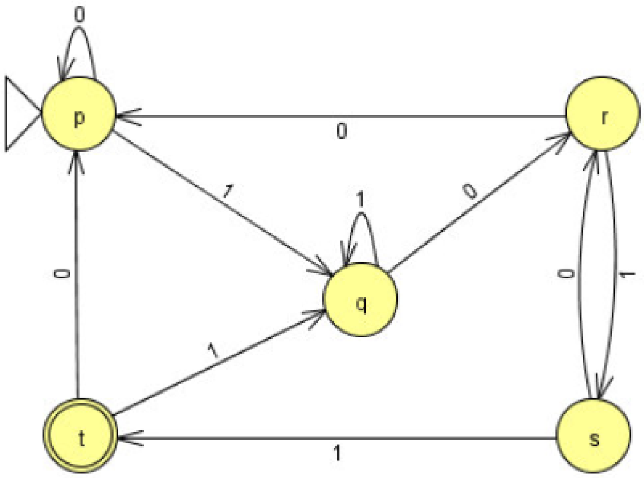


Pero como se trata de un AF **determinista** hay que definir una transición desde cada estado con cada uno de los símbolos del alfabeto. También es necesario añadir dichas transiciones para controlar las situaciones de error, en las que la secuencia inicialmente sea igual a la palabra clave, pero llegado un determinado símbolo varíe, y haya que retroceder hasta el estado adecuado, que no siempre será el inicio. Así, añadiendo desde cada estado la transición con el símbolo que faltaba, nos queda:



Por último, también hay que añadir transiciones desde el estado final, t, porque el autómata debe reconocer varias palabras clave "1011" y no sólo la primera. Así, si se lee un 0 se transita al estado "p", a esperar que aparezca un "1" que inicia la palabra clave. Pero si se lee un 1, se puede transitar directamente al estado "q" que se representa la lectura del primer símbolo de la palabra clave.

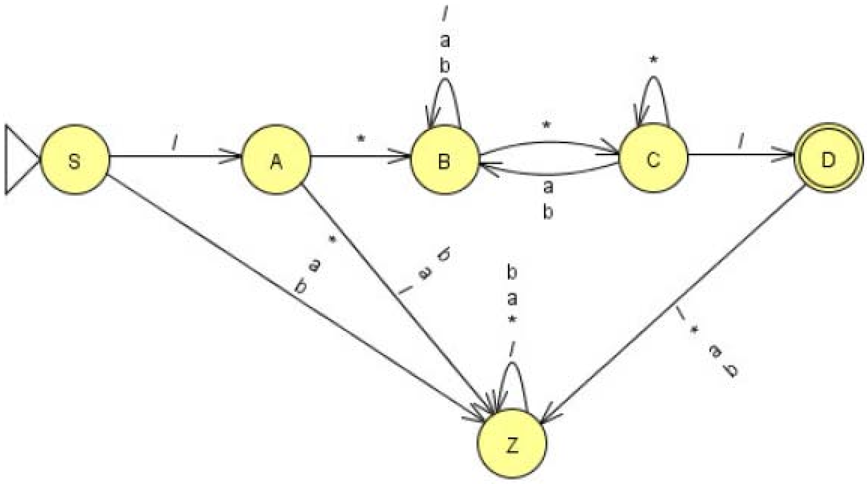
Así, finalmente, la definción formal del autómata finito es:
AFD=({0,1},{p,q,r,s,t},f,p,{t}), siendo f:



7. En algunos lenguajes de programación, los comentarios aparecen entre los delimitadores "/*" y "*/" como marca inicial y final del comentario. Sea L el lenguaje de todas las cadenas de comentarios delimitados. Así pues todo elemento de L, empieza por /* y acaba por */, pero no debe tener ningún */ intermedio. Por simplicidad consideraremos que el alfabeto sería {a, b, /,*}. Indicar el Autómata Finito Determinista que reconoce L.

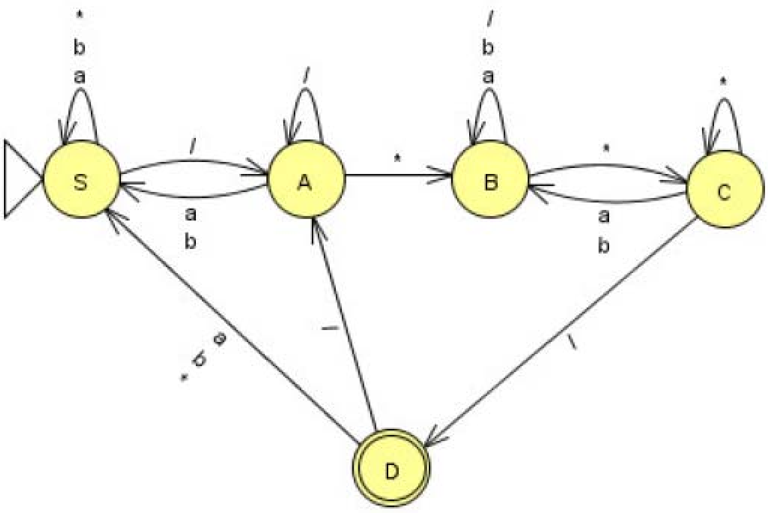
Solución:

Un posible autómata es AFD_1=({/,*,a,b},{S,A,B,C,D,Z},f,S,{D}), siendo f:



En esta solución, la cadena de entrada debe contener la marca de inicio de comentario (“*”) desde el principio, lo cual se reconoce en las transiciones entre los estados S, A y B. Si la cadena de entrada no comienza con “*”, la palabra se rechaza, transitando desde S o A al estado sumidero, Z.
Además, en esta solución sólo se reconoce un comentario, por lo que al llegar al final del comentario, es decir, cuando el autómata está situado en el estado final D, el resto de cadena no se analiza, dado que se transita al estado sumidero Z, del que no se sale.
Cabe destacar que las transiciones de B, C y entre ellos leen el texto que se encuentra dentro del comentario, analizando si se ha llegado a la marca de fin de comentario (“*/”) o no, y hay que retroceder a B para seguir leyendo símbolos del interior del comentario.

Otro posible autómata es AFD_2=({/,*,a,b},{S,A,B,C,D},f’,S,D), siendo f’:



En esta solución se permite reconocer un comentario en medio de un texto mayor representado en la cadena de entrada, dado que se espera en el estado inicial S hasta que se lee el símbolo "\", sin transitar a un estado sumidero, como se hacía en la solución anterior. Además, esta solución permite reconocer más de un comentario, dentro de un texto o cadena de entrada. Para ello, se llega al estado final D cada vez que se cierra un comentario, y se empieza de nuevo por S o A, dependiendo de si hay texto entre medias de los dos comentarios (transición de D a S) o no (transición de D a A).