2010

Practique la Teoría de Autómatas y Lenguajes Formales

Leonardo Alonso Hernández Rodríguez
Sonia Jaramillo Valbuena
Sergio Augusto Cardona Torres
Armenia - Quindío

Practique la Teoría de Autómatas y Lenguajes Formales

Leonardo Alonso Hernández Rodríguez

Adscrito al
Programa de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Quindío

Sonia Jaramillo Valbuena

Adscrita al
Programa de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Quindío

Sergio Augusto Cardona Torres

Adscrito al
Programa de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Quindío

Practique la Teoría de Autómatas y Lenguajes Formales

ISBN: 978-958-44-7913-6

200 ejemplares

EDICIONES ELIZCOM

Armenia, Quindío – Colombia

Practique la Teoría de Autómatas y Lenguajes Formales

Contenido

1. Introducción

El desarrollo de compiladores es un campo computacional extremadamente complejo. En esta área, para realizar proyectos de calidad y que no sean propensos a errores, se requiere tener una buena fundamentación en la Teoría de autómatas y lenguajes formales, también conocida como Teoría de la Computación. Esta teoría es de índole matemática y trata diversos conceptos como tokens, palabras, lenguajes, alfabetos, expresiones regulares, autómatas, gramáticas, forma normal BNF y muchos más.

El título del libro ya da una idea de su enfoque. Es un texto que promueve la práctica, mediante ejercicios y proyectos, de los conceptos de la Teoría de Autómatas y Lenguajes Formales, conceptos que son más bien abstractos. Muchos de los ejercicios vienen con la correspondiente respuesta, por lo que son un excelente medio para que el estudiante se autoevalúe, perfeccione sus conceptos e incluso prepare sus exámenes. El libro promueve un aprendizaje activo por parte de los estudiantes, de acuerdo con las recomendaciones de los pedagogos.

El profesor también puede sacar gran provecho de los temas tratados, diseñando sus propios talleres, proyectos, ejercicios o exámenes, con base en los proyectos propuestos y en los variados tipos de ejercicios ofrecidos.

El libro está orientado a un curso de Teoría de Autómatas y Lenguajes Formales para la carrera de Ingeniería de Sistemas y Computación, aproximadamente en V semestre. Se recomienda altamente este texto, como preparación previa para tomar un curso de Compiladores. Adicionalmente, durante el curso de compiladores, el texto puede ser un excelente material de repaso y de consulta.

Con frecuencia se presentan temas y ejercicios orientados al área de compiladores. Así, el estudiante no solo aprenderá conceptos matemáticos en abstracto, sino que verá el sentido que tienen y una de sus grandes posibilidades de aplicación. Lo anterior también está de acuerdo con recomendaciones de los pedagogos, que insisten en que el estudiante debe saber en que se aplican los conocimientos que están aprendiendo.

En los capítulos, se incluye una descripción clara y concisa de los conceptos que se van a tratar y se continúa con ejemplos, ejercicios con respuestas y ejercicios sin respuestas.

Adicionalmente, se proponen tres proyectos en diferentes momentos de la lectura. El primero busca que el lector se forme un concepto más general de los que es un lenguaje de programación, mediante el estudio de algunas instrucciones y tokens de diferentes lenguajes. En el segundo y tercer proyecto, se propone que el lector diseñe su propio lenguaje de programación. Específicamente, en el segundo proyecto, se deben diseñar los tokens del lenguaje e implementar el analizador léxico correspondiente. El último proyecto se enfoca en el diseño de aspectos sintácticos, utilizando la notación BNF.

Se espera realizar una contribución importante al currículo y a los procesos de aprendizaje de estudiantes de Ingeniería de Sistemas y Computación, con el enfoque que ha orientado la elaboración de los diferentes capítulos.

2. Generalidades de Teoría de Lenguajes Formales

En este capítulo se introducen los primeros conceptos de la Teoría de Autómatas y Lenguajes Formarles. Específicamente, se tratan los conceptos de alfabeto, palabra, lenguaje y las operaciones y relaciones entre ellos. A medida que se avance en el capítulo, el lector notará que estos términos tienen en este contexto un significado diferente al que tienen en el uso corriente.

2.1 Conceptos básicos

2.1.1 Alfabeto

Definición.

Un alfabeto es un conjunto no vacío y finito de símbolos [KELLEY].

Ejemplos.

Los siguientes son alfabetos

1) {a, b, c, …, z}
2) {a, b, c}
3) {1, 2, 3, 4}
4) {a, b, 5, 6, @, *}
5) {casa, perro, awb}

Los siguientes conjuntos no son alfabetos.

1) El conjunto vacío no es un alfabeto. Un alfabeto debe ser un conjunto no vacío.

2) El conjunto de los números naturales no es un alfabeto. Un alfabeto debe ser finito.

2.1.2 Palabra o cadena

Definición

Una palabra sobre un alfabeto es una secuencia finita de símbolos de dicho alfabeto [KELLEY].

En la definición se puede observar que, en Teoría de Lenguajes Formales, las palabras no necesitan tener sentido o significado.

Ejemplos

1) bbca es una palabra sobre el alfabeto {a, b, c}
2) casacasaawbperro es una palabra sobre el alfabeto {casa, perro, awb}
3) Una secuencia de cero símbolos es una palabra sobre el alfabeto {a, b, c}. Esta palabra se conoce como la palabra vacía y se representa por ε (Épsilon), en algunos textos, por λ (Lambda).
4) perro es una palabra sobre el alfabeto {a, b, c, …, z}
5) while, una de las palabras claves del lenguaje Java, es una palabra sobre el alfabeto conformado por los 256 caracteres ASCII.
6) x=325+y; una instrucción de asignación en lenguaje Java, es una palabra sobre el alfabeto conformado por los 256 caracteres ASCII.
7) Un programa de computador es una palabra sobre el alfabeto conformado por los 256 caracteres ASCII.
8) Una frase es una palabra sobre un alfabeto cuyos símbolos son todas las palabras del lenguaje castellano

9) Un párrafo es una palabra sobre un alfabeto que tenga todas las palabras del lenguaje castellano y además los signos de puntuación, espacio en blanco etc.

Notas

1. El orden de los símbolos importa en las palabras, debido a que por definición una palabra es una secuencia. Por ejemplo, la palabra *casa* es diferente de la palabra *saca*, y la palabra *ab* es diferente de la palabra *ba*.
2. No hay palabras infinitas.
3. La palabra *012* es diferente de la palabra *12*. La primera es una palabra de 3 símbolos y la segunda, una de dos símbolos.

Ejercicios

Responda verdadero o falso

1) abb es una palabra sobre {ab, a, c}
2) El símbolo ASCII espacio es igual a ε

2.1.3 Lenguaje

Definición.

Un lenguaje sobre un alfabeto es un conjunto de palabras sobre ese alfabeto [KELLEY].

Ejemplos.

1) { aba, bca, cca } es un lenguaje sobre {a, b, c}
2) El conjunto infinito {a, aa, aaa, … } es un lenguaje sobre { a}.
3) ϕ , el conjunto vacío es un lenguaje.

4) {a, ala, aro, …, zurdo} , el conjunto de todas la palabras del castellano, es un lenguaje sobre el alfabeto {a, b, c, …, z}

Ejercicios

Responda verdadero o falso

1) { }={ ε }
2) ε ={ }
3) { }={ { } }

2.2 Operaciones con alfabetos

2.2.1 Unión, intersección, diferencia

Como los alfabetos son conjuntos, podemos utilizar con ellos las operaciones entre conjuntos.

Ejemplos

{a, b, c} $\cup$ {b, c, d} = {a, b, c, d}

{a, b, c} $\cap$ {b, c, d} = { b, c}

{a, b, c} - {b, c, d} = { a }

Propiedades.

1) La unión de alfabetos siempre da por resultado un alfabeto.
2) La intersección de alfabetos da por resultado un alfabeto, si el resultado no es vacío.
3) La diferencia de alfabetos da como resultado un alfabeto, si el resultado no es vacío.

2.2.2 Cerradura de estrella o lenguaje universal

La cerradura de estrella o lenguaje universal es el lenguaje formado por todas las posibles palabras que se puedan formar utilizando un alfabeto dado. Se representa por Σ^*, lo cual se lee cerradura de estrella de Σ o lenguaje universal sobre Σ.

Ejemplo.

Sea Σ = {a, b}, entonces,

Σ^* = { ε , a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, …}

Los puntos suspensivos representan palabras de 4, 10, 1 000 000, 1 googol de símbolos y aún de más símbolos

Un googol corresponde al número 10^{100}. El matemático estadinense Edward Kasner preguntó a su sobrino por un nombre para un número muy grande. El sobrino, Milton Sirotta inventó la palabra googol.

Un googol plex es $10^{1\ \text{googol}}$

Propiedades.

1) La cerradura de estrella de un alfabeto siempre es un lenguaje infinito.
2) Para todo alfabeto Σ, $\varepsilon \in \Sigma^*$.

Definición.

Sea Σ un alfabeto, definimos la cerradura de estrella de Σ, denotada por Σ^*, como el lenguaje de todas las palabras sobre Σ.

2.3 Operaciones con cadenas y relaciones entre ellas.

2.3.1 Cardinal

Cardinal es el número de símbolos de una cadena.

Ejemplos

1) |perro|=5, si perro es una palabra sobre {a, b, c, … , z }
2) |bccca|= 4, si bccca es una palabra sobre {a, b, c}
3) |bccca|=3, si bccca es una palabra sobre {ca, cc, b}
4) $|\varepsilon| = 0$.
5) |ab| es una expresión ambigua, si ab es una palabra sobre {ab, a, b}

2.3.2 Concatenación

La concatenación es similar a la concatenación en programación de computadores.

Ejemplos

$inter \cdot disciplinario = interdisciplinario$

$ab \cdot bc = abbc$

$ab \cdot \varepsilon = ab$

Propiedad.

Sean w, x y y palabras, entonces $|wx| = |w \cdot x| = |w| + |x|$

Ejercicios con respuestas

Determine si las siguientes afirmaciones son verdaderas o falsas

1) Un párrafo de un texto de un periódico puede ser considerado como una palabra, en teoría de lenguajes formales.

2) Sea $\Sigma = \{ab, bc\}$ un alfabeto, entonces bcababc es una palabra sobre Σ

3) Sea $\Sigma = \{ab, ba\}$ un alfabeto, entonces babaabbaabbaabab es una palabra sobre Σ

4) Sea $\Sigma = \{ab, c\}$ un alfabeto, entonces abccabaabc es una palabra sobre Σ

5) Sea Σ un alfabeto y sea a un símbolo, si $a \in \Sigma$ entonces $a \in \Sigma^*$

6) Para todo alfabeto Σ se tiene que $\{\varepsilon\} \in \Sigma^*$

7) Sea A un lenguaje sobre Σ, entonces, $A \subseteq \Sigma^*$

8) Sea $\Sigma = \{a, b\}$ un alfabeto y sea $A = \{wax \mid w \in \Sigma^* \wedge x \in \Sigma^*\}$ un lenguaje, Se puede decir que A es el lenguaje de todas las palabras que contienen por lo menos una a

9) Sea $\Sigma = \{a, b\}$ un alfabeto y sea $A = \{wa \mid w \in \Sigma^*\}$ un lenguaje, entonces A es el lenguaje de todas las palabras sobre Σ terminadas en *a*.

Respuestas: (1) V (2) F (3) V (4) F (5) V (6) F , note que $\varepsilon \in \Sigma^*$, sin las llaves, sí es verdadera (7) V (8) V (9) V

10) Considere la siguiente propiedad:

"Sean w y x palabras, entonces; $|wx| = |w||x|$ "

Responda V o F

i. w = *ab* y x = *cd* es un contraejemplo de la propiedad ___

ii. w = *a* y x = *b* es un contraejemplo de la propiedad ___

iii. w = *abc* y x = *d* es un contraejemplo de la propiedad ___

iv. w = *a* y x = *bc* es un contraejemplo de la propiedad

v. w = ε y x = ε es un contraejemplo de la propiedad

Respuesta: F V V V F

Ejercicios

Sea $\Sigma = \{a, b\}$, especifique por comprensión:

11) El lenguaje sobre Σ de las palabras que comienzan por *b* y terminan en *b*

12) El lenguaje sobre Σ de las palabras que tienen a *aa* como subcadena una o más veces.

13) El lenguaje sobre Σ de las palabras que comienzan por *b* y tienen a *aa* como subcadena una o más veces.

2.3.3 Potencia

La potencia consiste el concatenar una palabra consigo misma el número de veces indicado por la potencia, por ejemplo:

$$(abc)^3 = abc \cdot abc \cdot abc = abcabcabc$$

$$(abc)^0 = \varepsilon$$

$$\varepsilon^0 = \varepsilon$$

Definición.

[KELLEY] *Sea w una palabra sobre* Σ *y n un número natural, se define* w^n *como*

$$w^n = \begin{cases} ww^{n-1} \; si \; n \geq 1 \\ \varepsilon \; si \; n = 0 \end{cases}$$

Propiedad.

Sean w una palabra y n un número natural, entonces, $|w^n| = n|w|$

Ejercicios

1) Responda verdadero o falso a las siguientes afirmaciones.
   a) $(ab)^4(ab)^2=(ab)^6$
   b) $ab^3a=a^2b^3$
   c) $(ab)^4=a(ba)^3b$

2) Demuestre que son falsas las siguientes afirmaciones.
   a) Sean w y x palabras y n un número natural, entonces, $|(wx)^n|=|wx|^n$
   b) Sean w y x palabras y n un número natural, entonces, $|(wx)^n|=|w^n||x^n|$

3) Demuestre las siguientes proposiciones.
   a) Sean w y x palabras y n un número natural, entonces, $|w^nx^n|=n|wx|$
   b) Sean w, x palabras y m y n números naturales, entonces, $|(xy)^{m+n}|=m|xy|+n|x|+n|y|$

4) Especifique formalmente y por comprensión los siguientes lenguajes sobre el alfabeto {a, b}
   a) El lenguaje de las palabras conformadas por una secuencia de aes seguida de una secuencia de bes y por nada más.
   b) El lenguaje de las palabras conformadas por un número par de bes y por nada más. Nota: cero es un número par.
   c) El lenguaje de las palabras conformadas por una secuencia de aes seguido de una secuencia de bes, tales que el número de aes es el doble que el de bes. Nota: Las palabras no están conformadas por nada más que lo dicho anteriormente.

d) El lenguaje de las palabras conformadas por una secuencia de aes seguida de una secuencia de bes, seguida a su vez, de otra secuencia de aes. Además, hay más bes en total que aes en total. Las palabras no están conformadas por nada más que lo especificado anteriormente.

5) Especifique por compresión los siguientes lenguajes sobre el alfabeto {a, b}. Nota: Interprete la palabra “aes” como “cero o más aes” y la palabra “bes” como “cero o más bes”.

a) El lenguaje de las palabras conformadas por una secuencia de aes seguida por una secuencia de bes

b) El lenguaje de las palabras que comienzan por **a**, después de la cual sigue una secuencia de bes.

c) El lenguaje de las palabras conformadas por dos aes, entre las cuales se encuentra una secuencia de bes.

d) El lenguaje de las palabras que tienen una **a** y cualquier número de bes..

e) El lenguaje de las palabras que tienen dos aes y cualquier número de bes.

f) El lenguaje de las palabras que comienzan por **a,** después de la cual sigue una secuencia de bes, después del cual sigue una secuencia de aes.

g) El lenguaje de las palabras de aes, tales que el número de ellas en un cuadrado perfecto.

h) El lenguaje de las palabras que comienzan por una secuencia de aes, después del cual sigue una secuencia de bes de manera tal que el número de aes es igual al número de bes.

i) El lenguaje de las palabras conformadas por un número impar de aes.

j) El lenguaje de todas las palabras sobre Σ, que cumplen simultáneamente las siguientes condiciones:

   i. Cada palabra comienza con cero o más **bes**, continúa con cero o más **aes**, luego siguen cero o más **bes** y con esto termina la palabra.
   ii. hay más **bes** en total que **aes**.

k) El lenguaje de todas las palabras sobre Σ, que cumplen simultáneamente las siguientes condiciones:

   i. La palabra comienza con cero a más bes, continúa con cero o más aes y con esto termina la palabra.
   ii. El número de **bes** es par
   iii. El número de **aes** no es superior al doble del número de **bes**.

6) Especifique por comprensión los siguientes lenguaje sobre el alfabeto {a, b, c }:

a) El lenguaje de todas las palabras sobre Σ que cumplen simultáneamente las siguientes condiciones:

   i. La palabra comienza con cero o más **aes**, continúa con cero o más **bes**, luego siguen cero o más **ces** y con esto termina la palabra.
   ii. El número de **bes** es el doble que el de **ces**.
   iii. El número de **aes** es el doble que el de **bes**.

b) El lenguaje de todas las palabras sobre Σ que cumplen simultáneamente las siguientes condiciones:

   i. La palabra comienza con cero o más **bes**, continúa con cero o más **ces**, luego siguen cero o más **aes** y con esto termina la palabra.
   ii. El número de **bes** es par.
   iii. Juntas las **aes** y las **bes** superan en número a las **ces**.

Ejercicios con respuestas

Determine si las siguientes afirmaciones son verdaderas o falsas

7) Sea $\Sigma = \{a, b\}$ un alfabeto, luego $a^5 b^6 = b^6 a^5$

8) Sea $\Sigma = \{a, b\}$ un alfabeto, luego $ab^2 ab^3 = a^2 b^5$

9) Sea $\Sigma = \{a, b\}$ un alfabeto, luego $a(ab)^3 b = a^4 b^4$

10) Sea $\Sigma = \{a, b\}$ un alfabeto, luego $(ab)^3 b = a(ba)^2 b^2$

11) Sea $\Sigma = \{a, b\}$ un alfabeto y sea $A = \{b^i ab^j ab^k \mid i \geq 0 \wedge j \geq 0 \wedge k \geq o\}$ un lenguaje. Se puede decir que A es el lenguaje de todas las palabras sobre Σ conformadas por dos aes y cualquier número de bes.

12) Sea $w = 1$ una palabra, luego $w^n = 1$ para todo $n \geq 0$

13) Sean w, x y y palabras, luego $\left|w^i x^j y^k\right|^2 = \left|w^{2i} x^{2j} y^{2k}\right|$

14) Sea w una palabra $\left|w^{m+n}\right| = \left|w^m\right| \cdot \left|w^n\right|$.

Respuestas: 7) F, 8) F, 9) F, 10) V, 11) V, 12) F, 3) F, 14) F

15) Sea *n* un entero no negativo; *w*, *x*, *y* palabras sobre algún alfabeto. Una de las siguientes expresiones es igual a $\left|w^n xy^n\right|$. ¿Cuál es esta expresión?

A. $|x| + (|w| + |y|)^n$
B. $|w|^n + |x| + |y|^n$
C. $|w|^n |x| \, |y|^n$
D. $|x| + n(|w| + |y|)$
E. $|w^n \| xy^n|$

Respuesta: D

16) Sea A el lenguaje de las palabras conformadas por una secuencia de aes seguida de una secuencia de bes. Además, el número de aes es par, el número de bes es impar y hay más aes que bes. ¿Cuál de los siguientes es el lenguaje A, expresado por comprensión?

A. $\{a^{2i}b^{2j+1} \mid i \geq 0,\ j \geq 0, i > j\}$

B. $\{a^{2i}b^{2i+1} \mid i \geq 0\}$

C. $\{a^{2i}b^{2j+1} \mid i \geq 0,\ j \geq 0, 2i > 2j+1\}$

D. $\{a^{2i+2}b^{2i+1} \mid i \geq 0\}$

E. $\{a^{2i+1}b^{2j} \mid i \geq 0,\ j \geq 0, 2i+1 > 2j+1\}$

Respuesta: C

17) Sea A el lenguaje de todas las palabras conformadas por una secuencia de aes seguida de una secuencia de bes. Además, el número de aes es par y el número de bes es igual al número de aes más uno. Por ejemplo, una palabra de este lenguaje es la palabra *aabbb*.

Exprese por comprensión el lenguaje A.

Respuesta: $\{a^{2i}b^{2i+1} \mid i \geq 0\}$

18) Sea A el lenguaje de todas las palabras conformadas por una secuencia de aes seguida de una secuencia de bes seguida, a su vez, por una secuencia de ces. Además, el número de aes es par, el número de bes es el doble que el de aes y hay tantas ces como aes y bes juntas.

Exprese por comprensión el lenguaje A.

Respuesta: $\{a^{2i}b^{4i}c^{6i} | i \geq 0\}$

19) Considere la siguiente propiedad

"Sea *w* una palabra y *n* y *m* números naturales, entonces $|w^n|^m = |w^{nm}|$ "

Responda V o F

i. *w* = a, *n* = 1 *y* *m* = 1 es un contraejemplo de la propiedad ___

ii. *w* = ε , *n* = 1 *y* *m* = 1 es un contraejemplo de la propiedad ___

iii. *w* = aa, *n* = 1 *y* *m* = 1 es un contraejemplo de la propiedad ___

iv. *w* = a, *n* = 1 *y* *m* = 3 es un contraejemplo de la propiedad ___

v. *w* = a, *n* = 2 *y* *m* = 2 es un contraejemplo de la propiedad ___

Respuesta: F F F V F

2.3.4 Inversa

La inversa de una palabra se obtiene escribiendo sus símbolos de derecha a izquierda, por ejemplo:

1) $(aro)^I = ora$
2) $(abcb)^I = bcba$
3) $\varepsilon^I = \varepsilon$

Convención

Cuando no se especifica el alfabeto, asumimos un alfabeto sencillo de símbolos de un solo carácter. Así, para el primer ejemplo asumimos el alfabeto {a, r, o} y para el segundo, el alfabeto {a, b, c, d}. Para el tercer ejemplo, no interesa el alfabeto.

Propiedades.

Sean w, x y y, entonces:

1) $(wx)^I = x^I w^I$
2) $(wxy)^I = y^I x^I w^I$
3) $(w^I)^I = w$
4) $|w^I| = |w|$

Ejercicios con respuestas

Responda verdadero o falso

1) Para toda palabra w se tiene que $w^I \neq w$
2) Para toda palabra w se tiene que $|w^I| = |w|$
3) Sean w, x y y palabras, luego $|(w^i x^i y^i)^I| = i(|w| + |x| + |y|)$
4) Sean *w*, *x* y *y* palabras, entonces, $(wxy)^I = w^I x^I y^I$

Respuestas: 1) F, 2) V, 3) V, 4) F

Ejercicio

5) Demuestre la siguiente proposición.

   Sean w y x palabras y m y n números naturales, entonces,

   $|[(xy)^{(m+n)}]^I| = (m+n)|xy|$

2.3.5 Prefijo y prefijo propio.

Una palabra es prefijo de otra, si la segunda comienza por la primera, por ejemplo:

para es prefijo *paracaídas*

ab es prefijo de la palabra *abcd*.

Los prefijos propios son los prefijos diferentes de la palabra, por ejemplo:

para es un prefijo propio de paracaídas

luna no es un prefijo propio de luna.

Definición

Sean w y x palabras, w es prefijo de x, si y solo si, existe una palabra y tal que wy=x [KELLEY].

Definición. Sea w y x palabras, w es prefijo propio de x, si y solo si, w es prefijo de x y $w \neq x$ [KELLEY].

Ejercicios con respuestas

Determine si las siguientes afirmaciones son verdaderas o falsas.

1) ε es prefijo propio de toda palabra
2) Todo sufijo de una cadena es una subcadena de dicha cadena
3) Una palabra no puede ser a la vez, sufijo y prefijo de otra
4) Sea $\Sigma = \{a, b\}$ un alfabeto, la palabra abb tiene tres prefijos.

Respuestas: 1) F, 2) V, 3) F, 4) F

Ejercicios

5) Determine si las palabras dadas son prefijos de la palabra cba. En caso afirmativo, encuentre el valor de la palabra y de la definición.
    a) cb
    b) cba
    c) ε

6) Defina sufijo y sufijo propio.

7) Responda verdadero o falso a las siguientes proposiciones.
    a) Una palabra no puede ser a la vez sufijo y prefijo de otra.
    b) Para toda palabra w tenemos que w es sufijo de w.

2.3.6 Subpalabra o subcadena

El concepto de subcadena corresponde al concepto de subcadena de programación de computadores.

Ejemplos.

1) arrend es una subcadena de subarrendar
2) Estudiantes es una subcadena de numeroEstudiantesActivos
3) ab es una subcadena de ddabccd.

Definición.

Sean w y x palabras, w es una subcadena de x, si y solo si, existen cadenas y y z tales que x=ywz [KELLEY].

Ejercicios

¿Cuáles de las siguientes palabras son subcadenas de la cadena subarrendar? Halle los valores de *y* y de *z* en cada caso.

1) arre
2) sub
3) ε
4) baen
5) subarrendar

2.4 Operaciones con lenguajes

2.4.1 Cardinal de un lenguaje

El cardinal de un lenguaje es el número de palabras de dicho lenguaje.

Ejemplos.

| {ab, cd, ε }| = 3

| {ab, ε }| = 2

| { ε } | = 1

| {} | = | ϕ | = 0

Ejercicios.

Resuelva las siguientes expresiones.

1) |{abc}|

2) |abc|

2.4.2 Concatenación de lenguajes

La concatenación de dos lenguajes A y B es un tercer lenguaje C. En el lenguaje C van todas las concatenaciones posibles formadas por una palabra de A y un palabra de B.

Ejemplos.

{ corr, beb} · {er, o, imos} =

{ correr, corro, corrimos, beber, bebo, bebimos }

{ a, bc }{ ε }= { a, bc }

$\{ab, c\} \cdot \phi = \phi$

Definición.

Sean A y B lenguajes sobre algún alfabeto Σ *, definimos* $A \cdot B = AB = \{wx \mid w \in A \wedge x \in B\}$ [KELLEY].

Propiedades.

Sea Σ *un lenguaje y A, B y C alfabetos sobre* Σ *, entonces,*

1) $A(B \cup C) = AB \cup AC$
2) $(A \cup B)C = AC \cup BC$

Ejercicios con respuestas

Resuelva las siguientes expresiones y exprese el resultado por comprensión:

1) {a}{a}* {b}* {b}
2) {ab}* {a}{ba}*
3) {ab}$^+$ {ba}* {a}

Respuestas:

1) $\{aa^i b^j b \mid i \geq 0, j \geq 0\}$
2) $\{(ab)^i a(ba)^j \mid i \geq 0, j \geq 0\}$

3) $\{(ab)^i (ba)^j a \mid i \geq 1, j \geq 0\}$

Ejercicios

1) Sean A,B y C lenguajes sobre un alfabeto Σ, de un contraejemplo de:

   a) $A \cdot (B \cap C) = A \cdot B \cap A \cdot C$

   b) $A \cdot (B - C) = A \cdot B - A \cdot C$

2.4.3 Potencia de un lenguaje

Elevar un lenguaje *A* a una potencia *n* es simplemente concatenar el lenguaje consigo mismo *n* veces. *n* debe ser un número natural, es decir un entero no negativo. Siguen algunos ejemplos:

1) $\{ a, bc \}^3 = \{ a, bc \} \{ a, bc \} \{ a, bc \} =$

   {aa, abc, bca, bcbc } {a, bc}=

   {aaa, aabc, abca, abcbc, bcaa, bcabc, bcbca, bcbcbc}

2) $\{ a, bc \}^0 = \{ \varepsilon \}$

   $\{ \varepsilon \}$ es el uno de la concatenación de lenguajes.

3) $\phi^0 = \{\varepsilon\}$

El resultado se debe a la definición y no es similar al resultado del cálculo. En cálculo, cero elevado a la cero es indeterminado.

Definición.

[KELLEY] *Sea A un lenguaje sobre* Σ *y n un número natural, se define* A^n *como*

$$A^n = \begin{cases} AA^{n-1} \; si \; n \geq 1 \\ \{\varepsilon\} \;\; si \; n \; = \; 0 \end{cases}$$

Ejercicios.

Resuelva las siguientes expresiones. Exprese el resultado por extensión. Use puntos suspensivos si es necesario.

1) $\{\varepsilon, ab\}^2$
2) $\{\varepsilon, ab\}^{100}$
3) $\{\varepsilon, ab, (ab)^2\}^{50}$
4) $\{ba, (ba)^2\}^{100}$
5) $\{(ab)^{10}\}^{50}$
6) $(\{\varepsilon, ab\}\{(ab)^2, (ab)^3\})^{100}$
7) $(\{(ab)^2\}\{\varepsilon, (ab)^2\})^{50}$

Ejercicios con respuestas

Resuelva las siguientes expresiones y exprese el resultado por extensión.

8) $[\{(ab)^3\}\{\varepsilon, ab\}]^{50}$
9) $(\{\varepsilon, ab\}\{ab\}^{20})^{20}$
10) $(\{\varepsilon, ab\}\{\varepsilon, ab\})^{100}$
11) $[\{\varepsilon, a, a^2, a^3\} - \{\varepsilon, a, a^4\}]^{20}$
12) $(\{\varepsilon\}\{ab\})^{10}$

13) Sea el lenguaje A = $\{\varepsilon, ca\}^{25}$. Responda V o F.

i. $(ca)^{10} \in A$ ___

ii. $(ca)^{15} \in A$ ___

iii. $(ac)^{25} \in A$ ___

iv. $\varepsilon \in A$ ___

v. $(ca)^{30} \in A$ ___

14) $[\{\varepsilon, cb\}\{\varepsilon, cb\}]^{30}$ da como resultado uno de los siguientes lenguajes. ¿Cuál es este lenguaje?

A. $\{\varepsilon, cb, (cb)^2, ..., (cb)^{60}\}$

B. $\{cb, (cb)^2, ..., (cb)^{60}\}$

C. $\{(cb)^2, (cb)^3, ..., (cb)^{60}\}$

D. $\{\varepsilon, (cb)^2, (cb)^4 ..., (cb)^{60}\}$

E. $\{\varepsilon, cb, (cb)^2, ..., (cb)^{30}\}$

15) Resuelva la siguiente expresión y exprese el resultado por extensión. Escriba al menos los tres primeros elementos y el último elemento del resultado.

$$[\{(ab)^4, (ab)^5, (ab)^6, (ab)^7\} - \{(ab)^4, (ab)^7\}]^{30}$$

16) Resuelva la siguiente expresión. Exprese el resultado por compresión:

$$\{\varepsilon, bc\}^{100}\{\varepsilon, df\}^{20}\{a\}^{60}$$

Respuestas:

8) $\{(ab)^{150}, (ab)^{151}, ..., (ab)^{200}\}$

9) $\{(ab)^{400}, (ab)^{401}, ..., (ab)^{420}\}$

10) $\{\varepsilon,(ab),(ab)^2,...,(ab)^{200}\}$

11) $\{a^{40},a^{41},...,a^{60}\}$

12) $\{(ab)^{10}\}$

13) V V F V F

14) A

15) $[\{(ab)^4,(ab)^5,(ab)^6,(ab)^7\}-\{(ab)^4,(ab)^7\}]^{30}=$

$\{(ab)^{150},(ab)^{151},(ab)^{152},\ldots,(ab)^{180}\}$

16) $\{\varepsilon,bc\}^{100}\{\varepsilon,df\}^{20}\{a\}^{60}$ = $\{(bc)^i(df)^j a^{60} | 0 \leq i \leq 100, 0 \leq j \leq 20\}$

2.4.4 Cerradura de estrella o cerradura de Kleene de un lenguaje

Para hallar la cerradura de estrella de un lenguaje, se realizan todas las concatenaciones posibles de 0, 1, 2 o más palabras del lenguaje, por ejemplo:

1) $\{a,bc\}^*$ = { ε , a, bc, aa, abc, bca, bcbc,

aaa, aabc, abca, abcbc, bcaa, bcabc,

bcbca, bcbcbc,…}

2) $\{\varepsilon\}^* = \{\varepsilon\}$

3) $\phi^* = \{\varepsilon\}$

Definición.

[KELLEY] *Sea A un lenguaje, definimos la cerradura de estrella de A o cerradura de Kleene de A como*

$$A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \cup ... = \bigcup_{i=o}^{\infty} A^i$$

Ejercicio con respuesta

1) Determine si la siguiente afirmación es verdadera o falsa: *Para todo lenguaje A se tiene que* $\varepsilon \in A^*$

Respuesta: Verdadero

Ejercicios.

2) Resuelva las siguientes expresiones. Exprese el resultado por extensión. Use puntos suspensivos si es necesario.
   a) $\{ab\}^*$
   b) $\{\varepsilon, ab\}^*$
   c) $\emptyset^*$, $\{\varepsilon\}^*$
   d) $\{a\}\{a\}^*\{b\}$

3) Sea $A = \{a\}$ y $B = \{b\}$ especifique por comprensión:
   a) A^*ABB^*
   b) A^*BA^*B
   c) B^*AAB^*

Propiedad.

Sea Σ *un alfabeto y A un lenguajes sobre* Σ*, entonces,* $(A^*)^* = A^*$

La propiedad anterior se conoce como idempotencia. Esta propiedad también la posee el valor absoluto, en Cálculo, y la recuperación mediante bitácora de una base de datos base de datos.

2.4.5 Cerradura positiva de un lenguaje

Para hallar la cerradura positiva de un lenguaje, se realizan todas las concatenaciones posibles de 1, 2, 3, o más palabras del lenguaje, por ejemplo:

1) $\{a, bc\}^+$ = { a, bc, aa, abc, bca, bcbc, aaa, aabc, abca, abcbc, bcaa, bcabc, bcbca, bcbcbc,…}

2) $\{\varepsilon\}^+ = \{\varepsilon\}$

3) $\phi^+ = \{\}$

Definición.

[KELLEY] *Sea A un lenguaje, definimos la cerradura positiva A como*

$$A^+ = A^1 \cup A^2 \cup A^3 \cup \ldots = \bigcup_{i=1}^{\infty} A^i$$

Ejercicios.

Resuelva las siguientes expresiones.

1) $\{\varepsilon, ab\}^+$

2) Sea A = { a } especifique por comprensión A^* y A^+

3) Sea A = { ab } especifique por comprensión A^* y A^+

4) Halle $\emptyset^*$, $\{\varepsilon\}^+$

5) ¿Bajo qué condición $A^* = A^+$?

6) ¿Cuándo $\varepsilon \in A^+$?

7) Dar un contraejemplo para la siguiente afirmación: Sea A un lenguaje cualquiera, entonces: $A^+ = A^* - \{\varepsilon\}$

Ejercicios con respuestas

8) Resuelva la siguiente expresión y exprese el resultado por extensión.

$$[\{ab,a\}\{b,ab\}-\{ab^2,aab,ba\}]^{+|}$$

Respuesta: $\{ab,(ab)^2,(ab)^3,...\}$

Propiedades.

Sea Σ un alfabeto, A, B, C lenguajes sobre Σ y n un número natural, entonces,

1) $A^* = \{\varepsilon\} \cup A^+$

2) $A^+ = AA^* = A^*A$

3) $(A^+)^+ = A^+$

4) $(A^*)^+ = A^*$

5) $(A^+)^* = A^*$

2.4.6 Inverso de un lenguaje

Para hallar el inverso de un lenguaje, se invierten todas las palabras del lenguaje, por ejemplo:

{ aro, sala, acbd } ^{I} = { ora, alas, dbca}

Definición.

[KELLEY] *Sea A un lenguaje sobre un alfabeto Σ. Definimos el inverso de A como* $A^I = \{w^I \mid w \in A\}$

Ejercicio.

1) Resuelva la expresión $[\{a\}\{bc\}^*]^I$

Propiedad.

Sea A un lenguaje sobre algún alfabeto Σ*, entonces,* $(A^I)^I = A$

2.4.7 Complemento de un lenguaje

El complemento del lenguaje A es un conjunto formado por todas las palabras que no están en A, por ejemplo:

> El complemento del conjunto de todas las palabras que comienzan por a es el conjunto de todas las palabras que no comienzan por a, formalmente:
>
> Sea Σ = {a, b } un alfabeto y A = { w | w comienza por a} un lenguaje sobre Σ. El complemento de A es el siguiente.
>
> $\overline{A}$ = { w | w no comienza por a }

Definición.

Sea Σ *un alfabeto y A un lenguaje sobre* Σ*. Definimos el complemento de A como* $\overline{A} = \Sigma^* - A$ [KELLEY].

Ejercicios

1) Sea Σ ={a, b, c} y A = { w | w tiene a y w tiene b } un lenguaje sobre Σ, Halle $\overline{A}$.

2) Sea Σ ={a, b, c}, A={ w | w tiene b o w no tiene c } un lenguaje sobre Σ, Halle $\overline{A}$.

Propiedades

Sea Σ *un alfabeto, entonces,*

1) $\overline{\phi} = \Sigma^*$

2) $\overline{\Sigma^*} = \phi$

2.5 Miscelánea de ejercicios, con respuestas.

1) Sea el lenguaje A =$\{\varepsilon, ac, b\}$ Responda V o F.

i. $(bac)^{20} \in A^{20}$ ___

ii. $b^5(ac)^5b^5 \in A^{20}$ ___

iii. $bac \in A^*$ ___

iv. $\varepsilon \in A^+$ ___

v. $b^2 \in A^{20}$ ___

Respuesta: F V V V V

2) Sean *a*, *b*, *c*, *d*, *e*, *f*, *g* símbolos de algún alfabeto. Responda V o F.

i. $(ab)^5(ab)^6 = (ab)^{11}$ ___

ii. $a^5(bc)^5 \in \{bc, a\}^*$ ___

iii. $\{ca, (ca)^2\}^3 = \{(ca)^3, (ca)^6\}$ ___

iv. $\{\varepsilon, a\}\{\varepsilon, a\} = \{\varepsilon, a^2\}$ ___

v. $\{ab, c\} - \{c, ba\} = \{ab, ba\}$ ___

Respuesta: V V F F F

3) Sea el lenguaje A = $\{a, bc\}$. Responda V o F.

i. $a^5(bc)^{10}a^7(bc)^{10} \in A^{32}$ ___

ii. $\varepsilon \in A^{32}$ ___

iii. $a^5(bc)^5 \in A^{32}$ ___

iv. $(bc)^{40}abcaa \in A^*$ ___

v. $(abc)^{16} \in A^{32}$ ___

Respuesta: V F F V V

4) Responda verdadero o falso

A. $(abcd)^{50} \in \{\varepsilon, ab, cd\}^*$
B. $(abc)^{3} \in \{a, bc\}^{10}$
C. $(abcd)^{50} \in \{ab, cd\}^{50}$
D. $(abc)^{3} \in \{\varepsilon, a, bc\}^{10}$
E. $(ab)^{100} \in \{\varepsilon, a, b\}^{300}$
F. $\varepsilon \in \{a, ab\}^{+}$

Respuestas: A) V, B) F, C) F, D) V, E) V, F) F

2.6 Miscelánea de ejercicios

1) Sean A = { ε , a, bc } y B = { a, bc } lenguajes. Responda verdadero o falso a las siguientes proposiciones.

a) $(bc)^{10}a(bc)^5 a \in B^{17}$

b) $a(bc)^2 a^2 \in B^5$

c) $(bc)^{10}a(bc)^{10} \in B^{20}$

d) $a^5(bc)^5 a \in B^{12}$

e) $a^5(bc)^5 a \in A^{20}$

f) $a^3bca^5 \in B^*$

g) $cbc \in A^*$

h) $\varepsilon \in A^*$

i) $\varepsilon \in A^+$

j) $\varepsilon \in B^*$

k) $\varepsilon \in B^+$

2) Sean A, B y C lenguajes sobre algún alfabeto Σ. Demostrar que son falsas las siguientes proposiciones.

a) |AB|=|A||B|

b) $A^* - \{\varepsilon\} = A^+$

c) $| A^n | = | A |^n$

d) A(B-C)=AB-AC

3) Resuelva las siguientes expresiones y exprese el resultado por comprensión.

a) $\{a\}^*\{b\}^+$

b) $\{a\}^4\{\varepsilon, ab\}^{100}$

c) $\{a\}^*\{cc\}^{100}\{cc\}^+$

d) $\{ab\}^*\{\varepsilon, cc\}^{100}\{a\}^+$

2.7 Demostraciones

En esta sección, se tratará con propiedades de los lenguajes, en cuya demostración se pueden utilizar uno o más de los siguientes pasos permitidos

1) $w \in M \leftrightarrow w^I \in M^I$

2) $w \notin M \leftrightarrow w^I \notin M^I$

3) $w \in M^I \leftrightarrow w^I \in M$

4) $w \notin M^I \leftrightarrow w^I \notin M$

Para demostrar la igualdad de dos expresiones cuyo resultado es un lenguaje, se puede seguir el siguiente procedimiento:

1. Se toma una palabra cualquiera *w* de la expresión de la izquierda del signo igual y se demuestra que también pertenece a la expresión de la derecha de dicho signo.
2. Del paso anterior se concluye que la expresión de la izquierda del signo igual es un subconjunto de la expresión de la derecha de este signo.
3. Se toma una palabra cualquiera *w* de la expresión de la derecha del signo igual y se demuestra que también pertenece a la expresión de la izquierda del dicho signo.
4. Del paso anterior se concluye que la expresión de la derecha del signo igual es un subconjunto de la expresión de la izquierda de este signo.
5. De los pasos 2 y 4, se concluye que los lenguajes son iguales.

Sigue un ejemplo

Demuestre la siguiente propiedad.

Sea A un lenguaje sobre algún alfabeto Σ, entonces, $(\bar{A})^I = \overline{A^I}$

Demostración:

i. Sea w una palabra cualquiera de la expresión de la izquierda del signo igual, es decir $w \in (\bar{A})^I$. Ahora bien, $w \in (\bar{A})^I \rightarrow w^I \in \bar{A} \rightarrow w^I \notin A \rightarrow$ w$\notin A^I \rightarrow w \in \overline{A^I}$ luego $(\bar{A})^I \subseteq \overline{A^I}$

ii. Sea w una palabra cualquiera de la expresión de la derecha del signo igual, es decir $w \in \overline{A^I}$. Ahora bien, $w \in \overline{A^I} \rightarrow$ w$\notin A^I \rightarrow w^I \notin A \rightarrow w^I \in \bar{A} \rightarrow w \in (\bar{A})^I$ luego $\overline{A^I} \subseteq (\bar{A})^I$

De i ii, $(\bar{A})^I = \overline{A^I}$

Ejercicios con respuestas

1) Complete

$w^I \notin \overline{A} \cup \overline{B} \rightarrow$ _____ $\rightarrow$
$w \notin (\overline{A})^I \wedge w \notin (\overline{B})^I$

Respuesta: $w^I \notin \overline{A} \wedge w^I \notin \overline{B}$

2) En la siguiente inferencia se ha suprimido el paso intermedio. Complete este paso sobre la raya.

$w^I \in \bar{A} \cap B \rightarrow$ _____ $\rightarrow$
$w \in (\bar{A})^I \wedge w \in B^I$

Respuesta: $w^I \in \bar{A} \wedge W^I \in B$

3) Sean A, B y C Lenguajes cualesquiera, determine si las siguientes implicaciones son correctas o no.

A. a) $x \notin A \cap B \rightarrow x \notin A \wedge x \notin B$

B. b) $x \in A \cap B \rightarrow x \in A \cap B \vee x \in A \cap C$

C. c) $x \notin \overline{A \cap BC} \rightarrow x \in A \cap BC$

D. d) $x \notin A \cup B \rightarrow x \notin A \vee x \notin B$

E. e) $x \in A \cap B \wedge C \subseteq A \cap B \rightarrow x \in C$

Respuestas: A) F, B) V, C) V, D) F, E) F

4) Sea *w* una palabra y *A* y *B* lenguajes. La expresión $w \in A^I - \overline{B}$ implica una de las siguientes conclusiones. ¿Cuál es esta conclusión?

A. $w \in A^I \wedge w \in \overline{B}$
B. $w \in A^I \wedge w \notin \overline{B}$
C. $w \notin A^I \wedge w \notin \overline{B}$
D. $w^I \in A - \overline{B}$
E. $w \notin A^I - B$

Respuesta: B

5) Demuestre que las siguientes propiedades son falsas:

a) Sea *A* un lenguaje, entonces $A \cap A^I = \phi$

b) Sean *A* y *B* lenguajes, entonces $|AB| = |A| + |B|$

c) Sean *w*, *x*, *y* y *z* palabras, entonces $(wxyz)^I = w^I x^I y^I z^I$

Respuestas

a) Contraejemplo A={a}

$\{a\} \cap \{a\}^I = \phi$

{a} $\cap$ *{a}=* ϕ

{a}= ϕ

F

b) Contraejemplo A={a}, B={a}

|{a}{a}| = |{a}| + |{a}|

|{aa}| = 1 + 1

1=2

F

c) Contraejemplo w=a, x=b, y=c, z=d

(abcd) ^{I} = a ^{I} b ^{I} c ^{I} d ^{I}

dcba=abad

F

Ejercicios.

6) Sean A, B, C lenguajes y *w* y *x* palabras, ¿Cuáles de los siguientes pasos son correctos y cuáles no?

a) $w \in (A \cup B \cup C)^I \rightarrow w^I \in A \cup B \cup C$

b) $w^I \in (A \cup B) - C \rightarrow w \in (A \cup B)^I - C$

c) $w \in A \cup (B - C)^I \rightarrow w^I \in A \cup (B - C)^I$

d) $(wx)^I \notin \overline{A} \rightarrow wx \notin (\overline{A})^I$

7) Sean A,B y C lenguajes sobre un alfabeto Σ, demostrar:

$(A^I)^I = A$

8) Demuestre la siguiente propiedad

Sean A y B lenguajes sobre algún alfabeto Σ, entonces, $(A \cup B)^I = A^I \cup B^I$

2.8 Proyecto de exploración

En este punto, se propone un proyecto que consiste en investigar sobre diversos lenguajes de programación, para obtener así un concepto más general de lo que es un lenguaje de este tipo. Esta comprensión permitirá facilitar la comprensión de los siguientes temas. Según la información suministrada por lo estudiantes que han realizado este proyecto, el tiempo que se requiere para desarrollarlo es de alrededor de 5 horas.

2.8.1 Instrucciones

Como mecánica para desarrollar el proyecto se propone diligenciar la siguiente plantilla. La plantilla consta de varios títulos con las correspondientes indicaciones sobre lo que hay que escribir a continuación de cada uno de ellos. En muchos casos se solicita solo un ejemplo de una instrucción en algún lenguaje de programación, no las explicaciones sobre cómo funcionan las instrucciones, las cuales están disponibles ampliamente en Internet.

El procedimiento para diligenciar la plantilla consiste en agregar el contenido solicitado en cada sección y posteriormente borrar las indicaciones correspondientes, que se encuentran entre corchetes.

2.8.2 Plantilla

1) Introducción

[Una introducción a esta fase del proyecto]

2) Impresión de comillas en lenguaje Python

[Escriba la instrucción Python para imprimir en la consola una cadena que tenga dentro de ella comillas dobles, por ejemplo la cadena: ***Juan dijo "Chao" y se fue***.]

3) Extracción de elementos de una lista en lenguaje Python.

[Escriba una sola instrucción Python que permita extraer los elementos segundo a cuarto de una lista Python de 6 elementos.]

4) for en lenguaje Python

[Incluya aquí un ejemplo de código en lenguaje Python que imprima cada uno de los elementos de una lista Python de cadenas y la longitud de cada una de las cadenas]

5) Ejemplo de una lista en lenguaje Lisp

[Escriba la lista Lisp que evalúa la expresión 2*3- (4+6)]

6) Operadores relacionales del lenguaje Visual Basic

[Escriba los operadores]

7) Ejemplo de Sentencia If…Then … Else … en lenguaje Visual Basic

[De un ejemplo de la sentencia]

8) Ejemplo de regla en lenguaje Prolog.

[Escriba el ejemplo]

9) Ejemplo de una instrucción SELECT que utilice la cláusula GROUP BY en lenguaje SQL

[De un ejemplo de la instrucción]

10) Ejemplo de una sentencia for en lenguaje PHP

[De un ejemplo de la instrucción]

11) Ejemplo de una instrucción de asignación en lenguaje Pascal.

[De un ejemplo de la instrucción. Nota: Pascal también es conocido como Delphi]

12) Ejemplo de sentencia DIVIDE en lenguaje Cobol.

[De un ejemplo de la instrucción]

13) Ejemplo de hallar la inversa de una matriz en MatLab

[Escriba las instrucciones MatLab necesarias parar hallar la inversa de la siguiente matriz: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$]

14) Ejemplo de un comando whenever en lenguaje Urbi.

[De un ejemplo del comando]

15) Paralelismo en lenguaje Urbi

[Complete la siguiente tabla con los cuatro separadores de comando en Urbi, y para que se usa cada uno de ellos]

Separador	Uso

16) Observaciones y comentarios

[Esta sección es opcional, por si desea agregar algo no solicitado en los otros puntos]

17) Conclusiones

[Escriba alrededor de tres conclusiones de esta fase del proyecto]

3. Lenguajes regulares

La teoría de este capítulo se utiliza para especificar las reglas que deben cumplir los tokens de un lenguaje de programación. Los token son las palabras con las que escribimos los programas de computador, como por ejemplo números enteros, números reales, variables y operadores.

El estudio de los lenguajes formales es muy complejo, por lo que es buena estrategia comenzar el estudio por una parte de ellos, los lenguajes regulares.

Lo dicho anteriormente concuerda con el hecho de que el conjunto de todos los lenguajes no es numerable, mientras que el de los lenguajes regulares sí lo es. Un conjunto es numerable si sus elementos se pueden poner en correspondencia uno a uno con el conjunto de los números naturales.

En otras palabras, aunque el conjunto de todos los lenguajes y el de los lenguajes regulares son infinitos, el primero es más grande. Recordemos que en matemáticas, los infinitos vienen en varios tamaños. Los conjuntos numerables son conjuntos infinitos del tamaño de los números naturales.

3.1 Lenguaje regular

Los lenguajes regulares sobre un alfabeto se obtienen a partir de unas reglas muy precisas. A continuación se incluyen las reglas y algunos lenguajes regulares sobre $\Sigma = \{a, b\}$ obtenidos con las reglas.

1. Vacío y el lenguaje conformado por la palabra vacía son lenguajes regulares.
   a) ϕ
   b) $\{\varepsilon\}$

2. Los lenguajes formados por una sola palabra de un solo símbolo son regulares.
   c) {a}
   d) {b}

3. Los lenguajes que resultan de unir dos lenguajes regulares son regulares.
   e) {a,b} que es la unión de c) y d)
   f) { ε , b} que es la unión de b) y d)
   g) { ε , a, b} que es la unión de b) y e)

4. Los lenguajes que resultan de concatenar dos lenguajes regulares son regulares.
   h) {a, aa, ba} que es la concatenación de g) y c)
   i) {a, ab, b, bb} que es la concatenación de e) y f)

5. Los lenguajes que resultan de realizar una cerradura de estrella a un lenguaje regular son regulares.
   j) $\{\varepsilon, a, a^2, a^3, ...\}$ que es la cerradura de estrella de c)
   k) $\{\varepsilon, b, b^2, b^3, ...\}$ que es la cerradura de estrella de d)

Podemos seguir uniendo, concatenando o aplicando cerradura de estrella para obtener más leguajes regulares:

   l) $\{\varepsilon, b, b^2, ..., a, ab^2, ab^3, ..., a^2b, a^2b^2, ...\}$ que es la concatenación de j) y k).

Por lo tanto hay infinitos lenguajes regulares sobre {a, b}

Hay lenguajes que no son regulares, no se pueden obtener con los mecanismos descritos, por ejemplo $\{a^i b^i \mid i \geq 0\}$. El lenguaje l) es lo

que más nos podemos acercar a $\{a^i b^i \mid i \geq 0\}$, pero en este lenguaje el número de *aes* puede ser diferente del número de *bes*.

Definición

[KELLEY] Sea Σ un alfabeto, Los lenguajes regulares sobre Σ son los indicados por las siguientes reglas.

1. ϕ y $\{\varepsilon\}$ son lenguajes regulares
2. Para todo $a \in \Sigma$, {a} es un lenguaje regular.
3. Si A y B son lenguajes regulares sobre Σ, entonces, $A \cup B$ y $A \cdot B$ también son lenguajes regulares sobre Σ.
4. Si A es un lenguaje regular sobre Σ, entonces A^* también es un lenguaje regular sobre Σ
5. Ningún otro lenguaje es un lenguajes regular sobre Σ.

Demostrar que un lenguaje es regular

Para demostrar que un lenguaje dado es regular sobre un alfabeto dado tenemos dos procedimientos.

1. Podemos hacer la lista de los lenguajes regulares sobre el alfabeto hasta llegar al lenguaje dado.
2. Podemos expresar el lenguaje en términos de uniones, concatenaciones o cerraduras de estrellas de lenguajes unitarios.

Usaremos el método 2, por ser más práctico. Siguen dos ejemplos:

1) Demuestre que A = { ε, bb, a } sobre { a, b } es un lenguaje regular.

Solución:

$\{\varepsilon, bb, a\} = \{\varepsilon\} \cup \{bb\} \cup \{a\} = \{\varepsilon\} \cup \{b\}\{b\} \cup \{a\}$. Esta última expresión solo tiene operaciones permitidas: unión y concatenación.

Además, todos los lenguajes que componen esta expresión son unitarios. Por tanto, el lenguaje dado es un lenguaje regular.

2) Demuestre que $A = \{cd(ab)^i \mid i \geq 0\}$ es un lenguaje regular

Solución:

$\{cd(ab)^i | i \geq 0\} = \{cd(ab)^0, cd(ab)^1, cd(ab)^2, \dots\} =$

$\{cd\}\{(ab)^0, (ab)^1, d(ab)^2, \dots\} = \{c\}\{d\}\{ab\}^* = \{c\}\{d\}\{a\}\{b\}^*$ y de nuevo llegamos a una expresión conformada de conjuntos unitarios y de solo operaciones válidas.

Ejercicios

Demostrar que los siguientes lenguajes son regulares.

1) $A = \{(ab)^i a^j \mid i \geq 0, j \geq 0\}$

2) $A = \{a^{2i} \mid i \geq 2\}$

3) *A* = { *w* | toda *b* de *w* está precedida por una *a* } sobre { *a*, *b* }

4) *A* = { *w* | *w* comienza por *a* o por *b* } sobre { *a*, *b*, *c* }

5) *A*= { *w* | *w* comienza por *a*, *b* o *c* } sobre { *a*, *b*, *c*, *d* }. Sugerencia: adapte la respuesta del *f*)

6) { *w* | *w* es un identificador de un lenguaje de programación } sobre el alfabeto conformado por todos los caracteres de la tabla ASCII. Asuma que los identificadores de este lenguaje deben comenzar por letra o símbolo de subrayado. Luego siguen cero o más letras, dígitos o símbolos de subrayado.

7) *A* = { *w* | *w* es un entero sin signo }

Ejercicios con respuestas

8) Demuestre que los siguientes lenguajes son regulares, encontrando la expresión correspondiente:

a) $\{a^i \mid i \geq 4\}$

b) El lenguaje de todas las cadenas que tienen al menos dos ceros consecutivos sobre el alfabeto {*0,1*}

c) El conjunto de todos los identificadores del lenguaje de programación C. Suponga que un identificador C es una secuencia de 1 a 32 letras, números o caracteres de subrayado (_), pero comenzando siempre por letra o por el carácter de subrayado.

d) El lenguaje de todas las palabras conformadas por una o más *aes* seguidas de cero o más grupos *ab*.

e) El lenguaje de todas las palabras conformadas por una *a* seguida de *bes* o por una *b* seguida de *aes.*

f) El lenguaje sobre el alfabeto {*a*, *b*, *c*} de todas las palabras que no tienen *a*.

g) El lenguaje sobre el alfabeto {*a, b*} de todas las palabras que no contienen la subcadena *aa* ni la subcadena *bb*.

h) El lenguaje sobre el alfabeto {*a, b}* de todas las palabras que no contienen la subcadena *aaa*.

i) El lenguaje sobre el alfabeto {*a, b, c*} de todas las palabras que no tienen a *ab* ni a *aa* por subcadena.

j) El lenguaje sobre el alfabeto {*a*, *b*, *c*} de todas las palabras que no comienzan con *ab*

Respuestas

a) $\{a\}^4\{a\}^*$

b) $(\{0\} \cup \{1\})^* (\{0\}\{0\})(\{0\} \cup \{1\})^*$

c) $\{A, B, C, \ldots, Z, a, b, c, \ldots z, \_\}\{A, B, C, \ldots, Z, a, b, c, \ldots, z, 0,1,2,\ldots,9, \_, \varepsilon\}^{31}$

d) $\{a\}\{a\}^* (\{a\}\{b\})^*$

e) $\{a\}\{b\}^* \cup \{b\}\{a\}^*$

f) $(\{b\} \cup \{c\})^*$

g) $(\{\varepsilon\} \cup \{b\})(\{a\}\{b\})^* (\{\varepsilon\} \cup \{a\})$

h) $(\{\varepsilon\} \cup \{a\} \cup \{a\}\{a\})(\{b\} \cup \{b\}\{a\} \cup \{b\}\{a\}\{a\})^*$

i) $(\{b\} \cup \{c\} \cup \{a\}\{c\})*(\{\varepsilon\} \cup \{a\})$

j) $\{\varepsilon\} \cup \{a\} \cup \{a\}\{a,c\}\{a,b,c\}^* \cup \{b,c\}\{a,b,c\}*$

9) Sea A el lenguaje de las palabras que comienzan por a y terminan en b sobre el alfabeto { a, b }. Demuestre que A es un lenguaje regular.

Respuesta: A es regular ya que A=$\{a\}(\{a\} \cup \{b\})^*\{b\}$

10) Sea $A = \{(ab)^{2i} \mid i \geq 0\}$ un lenguaje sobre el alfabeto $\Sigma = \{a, b\}$. Demuestre que A es un lenguaje regular.

Respuesta: $A = \{abab\}^*$

11) Sea A el lenguaje de las palabras conformadas de una secuencia aes seguida de una secuencia de bes. Hay al menos dos aes y el número de bes es par. A es un lenguaje regular, porque:

A. $A = \{aa\}^*\{bb\}^*$

B. $A = \{a^i b^{2j} \mid i \geq 2,\ j \geq 0.\}$

C. $A = \{a\}^*\{a\}\{a\}\{bb\}^*$

D. $A = \{a\}^* \{b\} \{b\}^*$

E. $A = \{a\}^* \{bb\}^*$

Respuesta: C

12) Responda Verdadero o Falso a las siguientes igualdades de lenguajes sobre el alfabeto dado:

a) $\Sigma = \{a, b, c\}$

$\{abbb, bcab\} = (\{a\}\{b\} \cup \{b\}\{c\})(\{b\} \cup \{a\})\{b\}$ _____

b) $\Sigma = \{a\}$

$\{a^{3i} \mid i \geq 0\} = \{a\}^* \{a\}^* \{a\}^*$ _____

c) $\Sigma = \{a, b, c\}$

$\{w \mid w \ \ NO \ \ comienza \ \ por \ \ a\} =$

$= (\{b\} \cup \{c\})(\{a\} \cup \{b\} \cup \{c\})^*$ _____

d) $\Sigma = \{a, b\}$

$\{w \mid w \ \ NO \ \ comienza \ \ por \ \ aa\} =$

$\{\varepsilon\} \cup \{a\} \cup \{b\}(\{a\} \cup \{b\})^* \cup \{a\}\{b\}(\{a\} \cup \{b\})^*$ _____

e) $\Sigma = \{a, b, c\}$

$\{w \mid aab \ \ es \ \ una \ \ subcadena \ \ de \ \ w\} =$

$(\{a\} \cup \{b\} \cup \{c\})^* \{a\}\{a\}\{b\}(\{a\} \cup \{b\} \cup \{c\})^*$ _____

f) $\Sigma = \{a, b\}$

$\{w \mid w \ \ tiene \ \ al \ \ menos \ \ una \ \ a\} =$

$\{b\}^* \{a\}(\{a\} \cup \{b\})^*$ _____

g) $\Sigma = \{a, b, c\}$

$\{w \mid ba \quad NO \quad es \quad una \quad subcadena \quad de \quad w\} =$

$\{a\}^*(\{b\} \cup \{c\}\{a\}^*)^*$ _____

h) $\Sigma = \{a, b\}$

$\{w \mid w \quad comienza \quad por \quad ab \quad y \quad termina \quad en \quad aa\} =$

$(\{a\} \cup \{b\})(\{a\} \cup \{b\})^*(\{a\} \cup \{a\})$ _____

Respuestas: a) F, b) F, c) F, d) V, e) V, f) V, g) V, h) F

3.2 Expresión Regular

En la sección anterior, vimos una forma de representar un lenguaje es regular. Lo representábamos en términos de lenguajes unitarios y solamente con las operaciones unión, intersección y cerradura de estrella.

Una manera más sencilla de representar a los lenguajes regulares, consiste en suprimir las llaves. Consideremos por ejemplo el lenguaje $(\{a\} \cup \{b\})(\{a\} \cup \{b\} \cup \{c\})^*$. Si suprimimos los corchetes a esta expresión queda $(a \cup b)(a \cup b \cup c)^*$. Esta última expresión es una expresión regular y es una forma más corta y ágil de representar los lenguajes regulares.

Definición

[KELLEY] Sea Σ un alfabeto. Las expresiones regulares sobre Σ están definidas por las siguientes reglas.

1. ϕ y ε son expresiones regulares sobre Σ.
2. Para todo $a \in \Sigma$, a es una expresión regular sobre Σ

3. Si r y s son expresiones regulares sobre Σ, entonces, $r \cup s$ y $r \cdot s$ también lo son.
4. Si r es una expresión regular sobre Σ, entonces, r^* también lo es.
5. Ninguna otra secuencia de símbolos es una expresión regular sobre Σ.

Para denotar el lenguaje representado por una expresión regular *r*, usaremos *L(r)*, por ejemplo: $L(a^*b) = \{a\}^*\{b\}$.

Con unas convenciones adicionales, ya tenemos una forma breve de representar los tokens válidos en un lenguaje de programación. La convenciones son las siguientes:

$$L = A \cup B \cup C \cup \ldots \cup Z \cup a \cup b \cup c \cup \ldots \cup z \cup \acute{a} \cup \acute{e} \cup \acute{i} \cup \acute{o} \cup \acute{u}$$

$$D = 0 \cup 1 \cup 2 \cup \ldots \cup 9 \ .$$

Así, los enteros en un lenguaje de programación pueden ser representados por la sencilla expresión DD^*, como se puede apreciar en el siguiente razonamiento.

El lenguaje de los números enteros es el siguiente:

$$(\{0\} \cup \{1\} \cup \{2\} \cup \ldots \cup \{9\})(\{0\} \cup \{1\} \cup \{2\} \cup \ldots \cup \{9\})^*$$

Suprimiendo las llaves queda:

$$(0 \cup 1 \cup 2 \cup \ldots \cup 9)(0 \cup 1 \cup 2 \cup \ldots \cup 9)^*$$

Y, finalmente, aplicando la convención $D = 0 \cup 1 \cup 2 \cup 3 \cup \ldots \cup 9$ tenemos:

$$DD^*$$

Ejercicios

Especifique los siguientes lenguajes.

1) El lenguaje de los enteros con signo de un lenguaje de programación.
2) El lenguaje de los identificadores de algún lenguaje de programación. Suponga que los identificadores de este lenguaje se conforman de solo letras y tienen al menos una letra.
3) El lenguaje de los identificadores de algún lenguaje de programación. Los identificadores son los nombres de las variables, funciones, métodos y clases de los lenguajes de programación. Suponga que los identificadores del lenguaje se conforman de letras, dígitos, signo de pesos y rayas bajas, pero comenzando por letra, signo de pesos o raya baja.
4) El lenguaje de los identificadores de algún lenguaje de programación. Suponga que los identificadores de este lenguaje se conforman de un signo de pesos, seguido de una letra, seguida de cero o más letras o dígitos.
5) Los identificadores de algún lenguaje de programación están especificados por la expresión regular LDD^*. De tres ejemplos de identificadores de este lenguaje.

Ejercicios con respuestas

6) Los identificadores de cierto lenguaje de programación están dados por la siguiente expresión regular:

   $\$L(LD \cup \$)^*$

   Nota: L representa cualquier letra mayúscula o minúscula. *D* representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

   Responda V o F.

$AB5C$ es un identificador válido _____

$A es un identificador válido _____

$$ es un identificador válido _____

bA5 es un identificador válido _____

A_$ es un identificador válido _____

Respuesta: F V F V F

7) Los identificadores de cierto lenguaje de programación están dados por la siguiente expresión regular:

$$L(L \cup DL)^*$$

Nota: L representa cualquier letra mayúscula o minúscula. D representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Responda V o F.

A5BC5BC es un identificador válido _____

A5B es un identificador válido _____

A7 es un identificador válido _____

A es un identificador válido _____

5B es un identificador válido _____

Respuesta: V V F V F

8) Los identificadores de cierto lenguaje de programación están dados por la siguiente expresión regular:

$$LLLDD^*$$

Nota: L representa cualquier letra mayúscula o minúscula. D representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Responda V o F.

ABC125 es un identificador válido _____

AB es un identificador válido _____

ABC es un identificador válido _____

ABC1 es un identificador válido _____

5ABC es un identificador válido _____

Respuesta: V F F V F

9) Los identificadores de cierto lenguaje de programación están dados por la siguiente expresión regular:

$$LLDD(LL \cup DD)^*$$

Nota: L representa cualquier letra mayúscula o minúscula. D representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Responda V o F.

A es un identificador válido _____

A_2 es un identificador válido _____

AA23A es un identificador válido _____

AA24CC es un identificador válido _____

AA2325AA26 es un identificador válido _____

Respuesta: F F F V V

10) Los identificadores de cierto lenguaje de programación están dados por la siguiente expresión regular:

$$\$LD(L \cup DL)^*$$

Nota: L representa cualquier letra mayúscula o minúscula. D representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Responda V o F.

$A5 es un identificador válido _____

$A es un identificador válido _____

$A5B es un identificador válido _____

$A55AA5C es un identificador válido _____

$A33 es un identificador válido _____

Respuesta: V F V V F

Propiedades de las expresiones regulares

Muchas de las siguientes propiedades son propiedades son propiedades sencillas de los conjuntos y de los lenguajes, llevadas a la notación de las expresiones regulares.

Sean r, s y t expresiones regulares, entonces:

1. $r \cup s = s \cup r$
2. $r \cup \phi = r$
3. $\phi \cup r = r$
4. $r \cup r = r$
5. $(r \cup s) \cup t = r \cup (s \cup t)$
6. $r\varepsilon = r$
7. $\varepsilon r = r$
8. $r\phi = \phi$
9. $\phi r = \phi$
10. $(rs)t = r(st)$
11. $r(s \cup t) = rs \cup rt$
12. $(r \cup s)t = rt \cup st$
13. $r^* = r^{**}$
14. $r^* = r^* r^*$
15. $r^* = (\varepsilon \cup r)^*$
16. $r^* = r^*(r \cup \varepsilon)$
17. $r^* = (r \cup \varepsilon) r^*$
18. $r^* = \varepsilon \cup r r^*$

19. $(r \cup s)^* = (r^* \cup s^*)^*$

20. $(r \cup s)^* = (r^* s^*)^*$

21. $(r \cup s)^* = (r^* s)^* r^*$

22. $(r \cup s)^* = r^* (s r^*)^*$

23. $r(sr)^* = (rs)^* r$

24. $(r^* s)^* = \varepsilon \cup (r \cup s)^* s$

25. $(r s^*)^* = \varepsilon \cup r(r \cup s)^*$

26. $s r^* = s(r \cup \varepsilon)^* (r \cup \varepsilon) \cup s$

27. $r r^* = r^* r$

28. $\varepsilon^* = \varepsilon$

29. $\phi^* = \varepsilon$

30. $\varepsilon \cup r^* = r^*$

31. $r^* \cup \varepsilon = r^*$

Las anteriores propiedades se pueden usar para demostrar igualdades entre expresiones regulares. Como es típico en matemáticas, para realizar este tipo de demostración, se parte de un lado cualquiera de la igualdad y, mediante pasos rigurosos, se debe llegar al otro lado. Sigue un ejemplo.

Demostrar la siguiente igualdad:

$$(a \cup b)^* (a \cup b) a \cup a = (a \cup b)^* a$$

Demostración:

Por lo general es conveniente partir del lado más complejo.

$(a \cup b)^* (a \cup b) a \cup a =$

$[(a \cup b)^*(a \cup b) \cup \varepsilon]a =$ Propiedad 12

$[\varepsilon \cup (a \cup b)^*(a \cup b)]a =$ Propiedad 1

$[\varepsilon \cup (a \cup b)(a \cup b)^*]a =$ Propiedad 27

$(a \cup b)^*a$ Propiedad 18

Ejercicios con respuestas

11) Responda Verdadero o falso

a) $(a \cup \varepsilon)(ab \cup c)^* \cup (a \cup \varepsilon)(ab \cup c)^*(ab \cup c) =$
$(a \cup \varepsilon)(c \cup ab)^*[\varepsilon \cup (ab \cup c)]$ _____

b) $c(\varepsilon \cup aa)^* \cup (\varepsilon \cup aa)^*b = (\varepsilon \cup aa)^*(c \cup b)$ _____

c) $(a \cup bb \cup ab)^*(a \cup bb \cup ab)^* = (a \cup ab \cup bb)^*$ _____

d) $(a \cup bb)^*(a \cup bb) \cup \varepsilon = (a \cup bb)^*$ _____

e) $\varepsilon \cup (a \cup ba)(a \cup ab)^* = (a \cup ba)^*$ _____

f) $(ab^*a \cup c)[bc^*a(ab^*a \cup c)]^* =$

$[(ab^*a \cup c)bc^*a]^*(ab^*a \cup c)$ _____

g) $(aca^* \cup bc^*)^*(aca^* \cup bc^*) =$

$(aca^* \cup bc^*)(aca^* \cup bc^*)^*$ _____

h) $(ab^* \cup c^*)^* = (ab \cup c)^*$ _____

Respuestas a) V, b) F, c) V, d) V, e) F, f) V, g) V, h) F

12) Una de las siguientes expresiones regulares es igual a $(a \cup bc)^*$. ¿Cuál es esta expresión?

A. $a^* \cup (bc)^*$

B. $a^* \cup b^*c^*$

C. $\varepsilon \cup a \cup bc(a \cup bc)^*$

D. $(a \cup bc)^*(a \cup bc \cup \varepsilon)$

E. $a^*(bc)^*$

Respuesta: D

13) Una de las siguientes expresiones regulares es igual a $a(ab^* \cup c)^*$. ¿Cuál es esta expresión?

A. $(aab^* \cup ac)^*$

B. $a(ab \cup c)$

C. $a(b^*a \cup c)^*$

D. $a(ab^* \cup c)^* \cup \varepsilon$

E. $a(ab^*)^*[c(ab^*)^*]^*$

Respuesta: E

14) Una de las siguientes expresiones regulares es igual a $[a^*(bc)^*]^*$. ¿Cuál es esta expresión?

A. $\left(a^*\right)^*\left((bc)^*\right)^*$

B. $(abc)^*$

C. $(a \cup bc)^*$

D. $(a \cup cb)^*$

E. $[(ab)^*c^*]^*$

Respuesta: C

15) Una de las siguientes expresiones regulares es igual a $(c \cup ab^*)^*(c \cup ab^*)bb$. ¿Cuál es esta expresión?

A. $bb(c \cup ab^*)^*(c \cup ab^*)$

B. $(c \cup ab^*)(c \cup ab^*)^* bb$

C. $(c \cup ab^*)^*(c \cup ab^*)$

D. $(c \cup ab^*)^*(c \cup ab^* bb)$

E. $(c \cup ab^*)(c \cup ab^*)bb$

Respuesta: B

16) Una de las siguientes expresiones regulares es igual a $[(\varepsilon \cup aa^*)b^*]^*$. ¿Cuál es esta expresión?

A. $(ab)^*$

B. $(a \cup b)^*$

C. $(ba)^*$

D. $a^* \cup b^*$

E. $(aab)^*$

Respuesta: B

17) Una de las siguientes expresiones regulares es igual a $[(ab)(cd)]^*(ab)$. ¿Cuál es esta expresión?

A. $[(ab)(cd)(ab)]^*$

B. $\varepsilon \cup abcd(abcd)^*$

C. $(ab)^*(cd)^*(ab)^*$

D. $(ab)[(cd)(ab)]^*$

E. $\varepsilon \cup [(ab)(cd)]^*(ab)$

Respuesta: D

18) Una de las siguientes expresiones regulares es igual a $a \cup (ab)(ab)^* a$. ¿Cuál es esta expresión?

A. $a[\varepsilon \cup (ab)(ab)^*]$

B. $a \cup (ba)(ba)^* a$

C. $a \cup (ab)a^* b^* a$

D. $a(ba)^*$

E. $(ab)(ab)^* \cup aa$

Respuesta: D

19) Una de las siguientes expresiones regulares es igual a $[a^* (\varepsilon \cup b)^*]^*$. ¿Cuál es esta expresión?

A. $a^* b^*$

B. $(a \cup b)^*$

C. $(ab)^*$

D. $a^* \cup b^*$

E. $a^{**} (\varepsilon \cup b)^{**}$

Respuesta: B

Ejercicios

Demuestre las igualdades de los siguientes ejercicios.

20) $(ab)(ab)^* = (ab)^* ab(ab \cup \varepsilon)$

21) $[(\varepsilon \cup a^*)^* a^*]^* = a^*$

22) $a \cup (ab^*)(ab^*)^* a = a(b^* a)^*$

23) $[(a^*b^*)^*(b^*a^*)^*]^* = (a \cup b)^*$

24) $b(ab \cup ac) = (ba \cup ba)(b \cup c)$

25) $(aa)^* a = a(aa)^*$

26) $(a \cup b)(\varepsilon \cup aa)^*(\varepsilon \cup aa) \cup (a \cup b) = (a \cup b)(aa)^*$

27) $\phi^* \cup a^* \cup b^* \cup (a \cup b)^* = b^* \cup (a \cup b)^*$

28) $[b(ab)^*]^* = (b \cup ba)^* b \cup \varepsilon$

29) $a \cup a^* = a^*$

30) $[a(ba)^*]^*(ab)^* a \cup \varepsilon = (a(ba)^*)^*$

Deduzca la siguiente propiedad de las propiedades 1 a 29.

31) $\varepsilon \cup r^* = r^*$

Errores típicos

A continuación se presenta una lista de los errores más frecuentes en el proceso de demostración de igualdad de expresiones regulares.

1. $ab \cup cb = b(a \cup c)$ es un error. Según la propiedad 12, la parte común debe quedar al lado derecho, lo correcto es $ab \cup cb = (a \cup c)b$
2. $ab \cup bc = b(a \cup c)$ es un error. Para aplicar la propiedad 11 o la 12, se requiere que la parte que se va a 'factorizar' esté al mismo lado en ambos 'términos'.
3. $ab \cup cd = ac \cup bd$ es un error. Para aplicar la propiedad 1, se deben conmutar las dos expresiones que se están uniendo. Lo correcto es: $ab \cup cd = cd \cup ab$

4. $\varepsilon \cup aa^*b = a^*b$ es un error. En cambio, es correcto: $(\varepsilon \cup aa^*)b = a^*b$. Lo que está entre paréntesis sí tiene la forma general de la propiedad 18.

5. $ab^*c = acb^*$ es un error. No existe la propiedad conmutativa de la concatenación.

6. $a^*b = ab^*$ es un error, es una aplicación incorrecta de $r^*r = rr^*$. Sí es correcto $a^*a = aa^*$

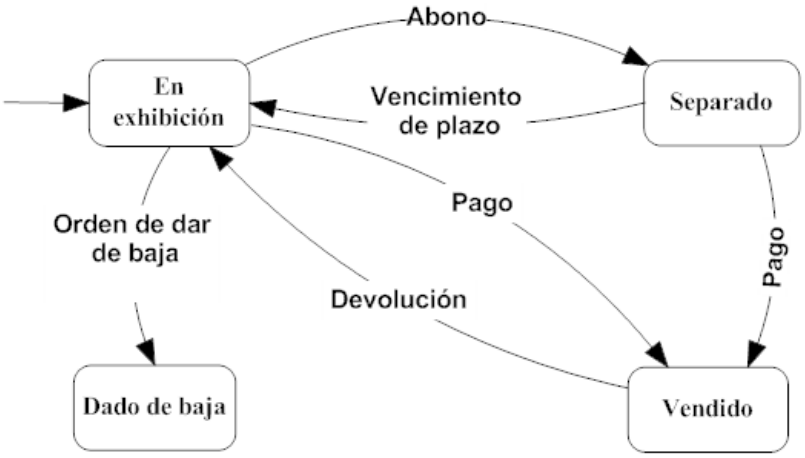
4. Autómatas finitos

Los autómatas finitos también se denominan máquinas de estados finitos. Los autómatas finitos son otra forma de representar a los lenguajes regulares. La primera forma ya tratada en este texto son las expresiones regulares. En consecuencia, los autómatas finitos también se usan para representar los tokens de un lenguaje de programación.

Los autómatas finitos son una forma de modelar en la que consideramos a los objetos como máquinas que pueden pasar por varios estados. Siguen ejemplos.

1. Podemos ver un semáforo como una máquina que puede pasar por los estados rojo, verde y amarillo.
2. Podemos ver una silla como una máquina que pueden pasar por los estados desocupada, ocupada, destruida.
3. Podemos ver a una persona como una máquina que puede pasar por los estados soltero, casado, viudo, divorciado.
4. Un artículo en un almacén puede pasar por los estados: en exhibición, separado, vendido, dado de baja.

La máquina en cuestión puede ser representada gráficamente por un diagrama denominado diagrama de transición, como el del siguiente ejemplo.



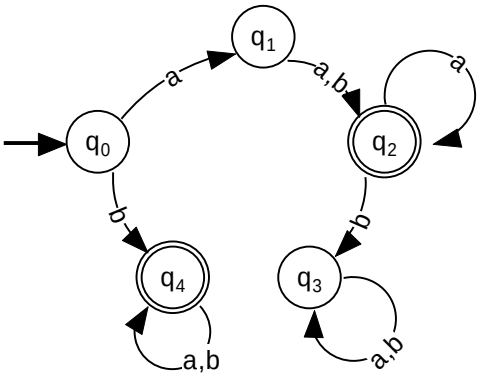
Cada nodo representa un estado: *En exhibición, Separado, Vendido, Dado de baja*.

Cada arista representa una transición, lo que se necesita para el cambio de estado. En la figura, las transiciones son: *Abono, Vencimiento de plazo, Pago, Pago, Devolución, Orden de dar de baja*.

El estado *En exhibición* en este autómata es el estado inicial. El estado inicial va señalado con una flecha.

4.1 Autómata finito determinista

En Teoría de Autómatas y Lenguajes Formales, podemos usar un autómata finito como una máquina que acepta o rechaza palabras, por ejemplo:



En el autómata de la figura tenemos lo siguiente:

- Los estados son q_0, q_1, q_2, q_3 y q_4.
- El estado inicial el q_0.
- Los estados finales o de aceptación son q_2 y q_4.
- Hay diez transiciones.

Para simplificar el diagrama, se usa una sola flecha con las dos letras en lugar de dos flechas con una letra cada una.

Un autómata finito determinista, abreviadamente AFD, acepta las palabras que correspondan a una ruta que comience en el estado inicial y termine en un estado final. El autómata rechaza cualquier otra palabra. Así, en el autómata del ejemplo, tenemos lo siguiente:

Palabra	**Ruta correspondiente a la palabra**	**¿La palabra es aceptada?**
ab	$q_0 \rightarrow q_1 \rightarrow q_2$	Sí
bab	$q_0 \rightarrow q_4 \rightarrow q_4 \rightarrow q_4$	Sí
aabb	$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_3$	No
a	$q_0 \rightarrow q_1$	No

Ejercicio

1) Complete la siguiente tabla, de acuerdo con el autómata del ejemplo anterior.

Palabra	Llega a:	¿Estado final?	¿Aceptada?
bb			
aba			
aab			
a^{100}			
$a^3ba^5b^5$			

Restricción de los autómatas finitos deterministas.

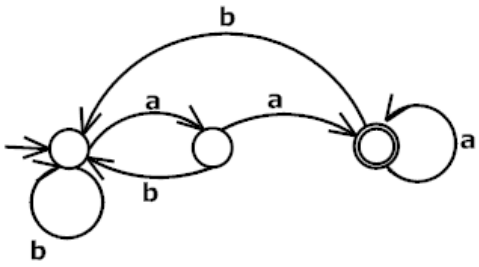
Los autómatas finitos deterministas cumplen con la restricción de que de cada estado sale una transición por cada símbolo del alfabeto. En el ejemplo, el alfabeto es $\Sigma = \{a, b\}$ y de cada estado sale una transición con a y una transición con b. No importa el número de transiciones que lleguen al estado, la restricción solo se refiere las transiciones que salen.

Lenguaje aceptado por un autómata finito determinista

El lenguaje aceptado por un autómata finito determinista es el lenguaje de todas las palabras aceptadas por el autómata. Formalmente:

> Sea *M* un autómata finito determinista con alfabeto de entrada Σ, se define el lenguaje aceptado por *M*, notado como *L(M)*, como $\{w \in \Sigma^* | w \ es \ una \ palabra \ aceptada \ por \ M\}$

Por ejemplo, el siguiente autómata acepta el lenguaje $(a \cup b)^* aa$ sobre {a, b}:



Ejercicios

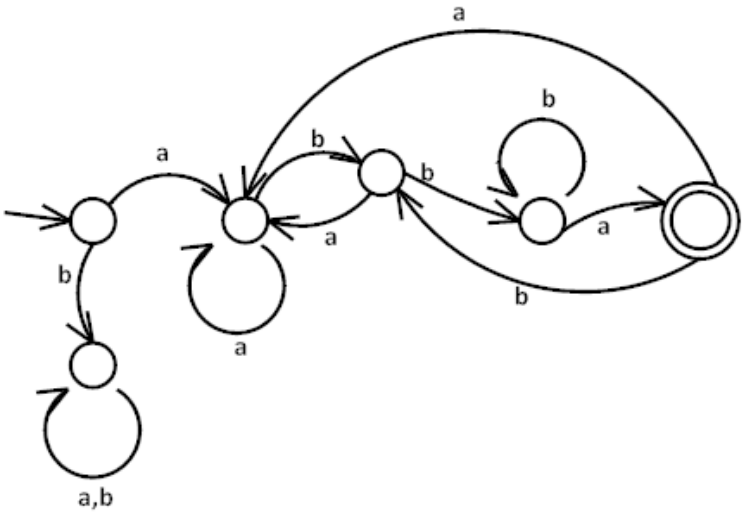
1) Elabore diagramas de transición de AFD que acepten los siguientes lenguajes.
   a) $(a \cup b)^* aa$ sobre {a,b}
   b) $(ab)^+ a$ sobre {a,b}
   c) $a^* bba^*$ sobre {a,b}
   d) $(aba)^*$ sobre {a,b}
   e) { w | w tiene al menos dos aes } sobre { a, b }
   f) { w | w tiene a abc por subcadena } sobre { a, b, c }
   g) { w | w tiene un número par de símbolos } sobre {a, b, c }
   h) { w | w no tiene a ab por subcadena } sobre { a, b }
   i) $a^* b^*$

Ejercicios con respuestas

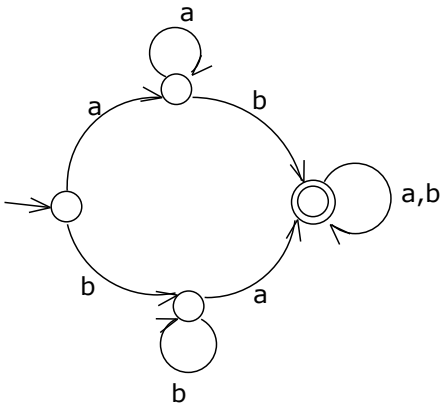
2) Elabore un diagrama de transición de un AFD para el siguiente lenguaje.

   { *w* | *w* comienza por *a* y termina en *bba* } sobre { *a*, *b* }

   Respuesta:



3) ¿Cuál es el lenguaje aceptado por el siguiente Autómata finito determinista?.



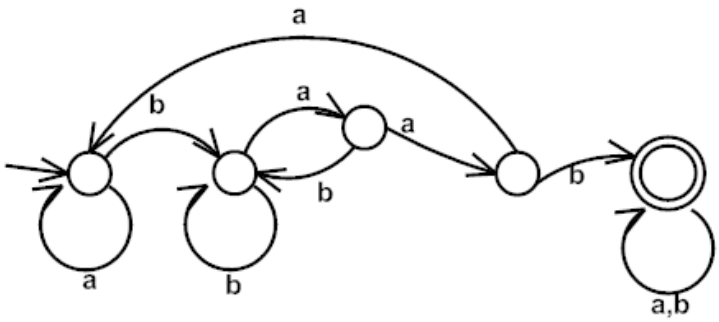
A. { *w* | *w* comienza por *ab* } sobre { *a*, *b* }
B. { *w* | *w* comienza por *a* o comienza por *b* } sobre { *a*, *b* }
C. { *w* | *w* no comienza por *a* } sobre { *a*, *b* }
D. { *w* | *w* tiene a *ab* por subcadena } sobre { *a*, *b* }
E. { *w* | *w* tiene al menos una *a* y tiene al menos una *b*} sobre {*a*, *b*}

Respuesta: E

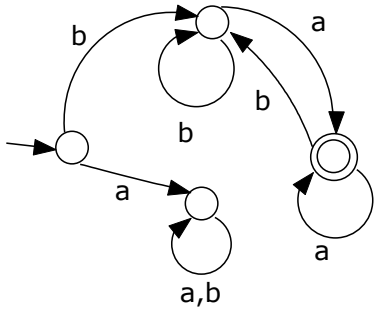
4) Elabore un diagrama de transición de un AFD para el siguiente lenguaje.

{ *w* | *w* tiene a *baab* por subcadena } sobre { *a*, *b* }

Respuesta:



5) ¿Cuál es el lenguaje aceptado por el siguiente Autómata finito determinista?



A. { *w* | *w* comienza por *b* y termina en *a* } sobre { *a*, *b* }
B. { *w* | *w* comienza por *b* y tiene al menos una a } sobre { *a*, *b* }
C. { *w* | *w* comienza por *ba* } sobre { *a*, *b* }
D. { *w* | *w* no comienza por *a* } sobre { *a*, *b* }
E. $\{b\}\{b\}^*\{a\}\{a\}^*$ sobre { *a*, *b* }

Respuesta: A

6) Elabore el diagrama de transición de un AFD para los siguientes lenguajes sobre el alfabeto {a,b}

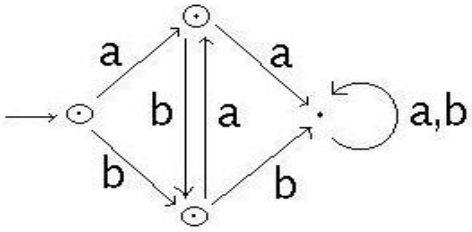
a) L={w | aa NO es una subcadena de W y bb NO es una subcadena de W}

b) L={w | ab NO es una subcadena de W y ba NO es una subcadena de W}

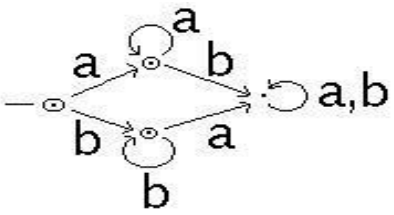
c) El lenguaje de las palabras que tienen a *abb* o a *bba* por subcadena.

Respuestas

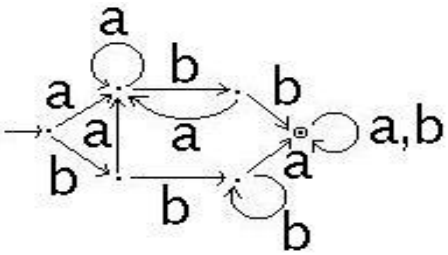
a)



b)

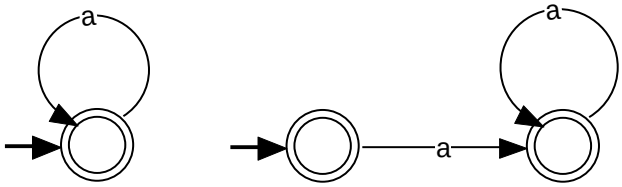


c)



4.1.1 Autómatas equivalentes

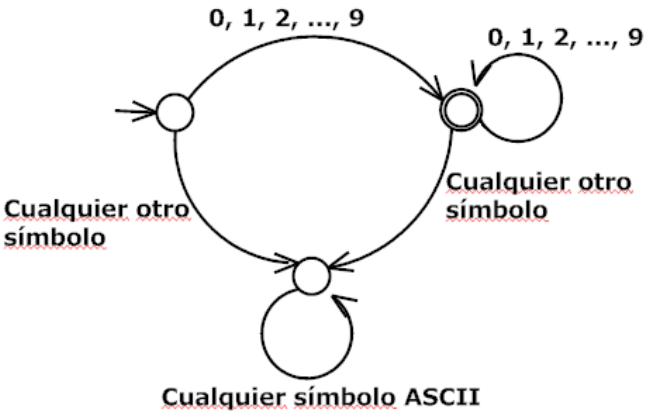
Diferentes autómatas pueden representar el mismo lenguaje. En este caso, decimos que los autómatas son equivalentes. Por ejemplo, los autómatas siguientes son equivalentes, ya que ambos representan el lenguaje a^*.



4.1.2 Aplicaciones de los autómatas finitos deterministas

Mediante autómatas finitos deterministas, podemos representar las reglas para construir los tokens de un lenguaje de programación. En otras palabras, podemos especificar los aspectos léxicos del lenguaje.

Por ejemplo, el siguiente autómata cuyo lenguaje de entrada es la tabla de caracteres ASCII, representa los números enteros de un lenguaje de programación.

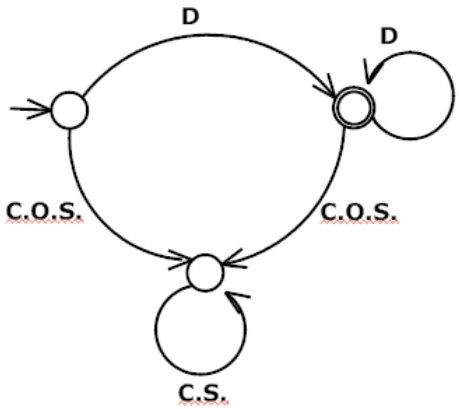


Note por ejemplo que la ruta correspondiente al entero 79245 finaliza en un estado de aceptación.

Por sencillez, se acostumbra utilizar unas convenciones muy naturales:

Forma abreviada	Lista de símbolos del alfabeto
D	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
L	a, b, c, …, z, A, B, C, …, Z, á, é, í, ó, ú, Á, É, Í, Ó, Ú
C.S.	Cualquiera de los 256 símbolos de la tabla ASCII
C.O.S.	Cualquiera de los 256 símbolos de la tabla ASCII que aparezcan en otras transiciones que salgan del estado en cuestión

Utilizando las convenciones anteriores, el diagrama de transición del anterior autómata queda más sencillo:



Ejercicios

Elabore AFD para los siguientes lenguajes.

1) Los identificadores de un lenguaje de programación. Suponga que los identificadores deben comenzar por letra o símbolo de

subrayado, después de lo cual puede seguir cero o más letras, dígitos o símbolos de subrayado.

2) Los operadores relacionales
3) Los números reales en notación normal
4) Un lenguaje que incluye a la vez los números enteros y los números reales en notación normal.
5) Las cadenas de caracteres Java. Tenga en cuenta que después de backslash solo puede ir por ejemplo: t, r, n, " y \.
6) A la vez, los enteros, los identificadores y los operadores aritméticos. Puede obviar dibujar el pozo de rechazo.
7) A la vez, los operadores relacionales; los operadores aritméticos +, -, * y /; y el operador de asignación =.

Ejercicios con respuestas

8) Mediante un autómata finito determinista podemos representar las reglas para construir uno de los siguientes elementos de un lenguaje de programación. Señale el elemento (Señale solo uno).

A. Una sentencia for
B. Una sentencia if
C. Una operador lógico
D. Una condición o expresión booleana.
E. Una expresión algebraica

Respuesta: C

9) Mediante un autómata finito determinista podemos representar las reglas para construir uno de los siguientes elementos de un lenguaje de programación.

A. Una sentencia while
B. Una sentencia if
C. Una sentencia for

D. Un operador relacional
E. Una expresión algebraica

Respuesta: D

10) Mediante un autómata finito determinista podemos representar las reglas para construir uno de los siguientes elementos de un lenguaje de programación. Señale el elemento (Señale solo uno).

A. Una sentencia for
B. Una sentencia if
C. Una sentencia switch - case
D. Una condición o expresión booleana.
E. Un identificador

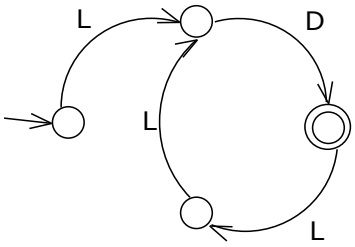
Respuesta: E

11) Mediante un autómata finito determinista podemos representar las reglas para construir uno de los siguientes elementos de un lenguaje de programación.

A. Un bucle while de Java
B. Una instrucción if
C. Un número real
D. La definición de una clase
E. Una sentencia de asignación

Respuesta: C

12) El autómata de la figura especifica los identificadores de un supuesto lenguaje de programación.



Nota: L representa cualquier letra mayúscula o minúscula. D representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Responda V o F.

A5AB6AB7 es un identificador válido _____

A es un identificador válido _____

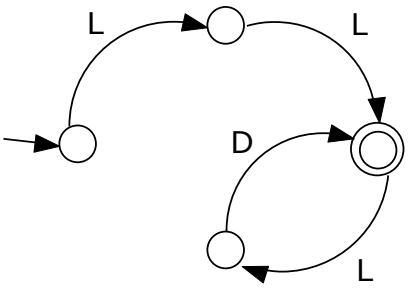
A6 es un identificador válido _____

A65 es un identificador válido _____

A7FF8 es un identificador válido _____

Respuesta: V F V F V

13) El autómata de la figura especifica los identificadores de un supuesto lenguaje de programación.



Nota: L representa cualquier letra mayúscula o minúscula. D representa un dígito: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Responda V o F.

ABC5C5 es un identificador válido _____

AB5 es un identificador válido _____

AB5C es un identificador válido _____

A es un identificador válido _____

AB es un identificador válido _____

Respuesta: V F F F V

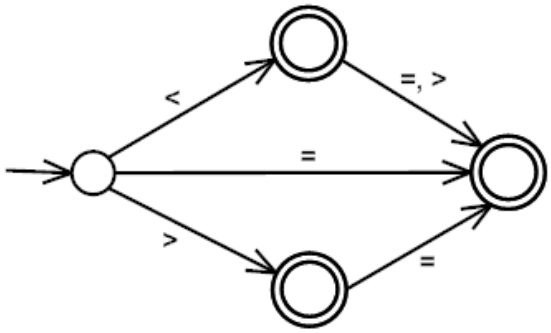
14) Los operadores relacionales de cierto lenguaje de programación son los siguientes:

    <
    >
    =
    >=
    <=
    <>

Elabore un Autómata finito determinista que represente estos operadores.

Nota: Por simplicidad, puede omitir en el diagrama el sumidero de rechazo.

Respuesta:



4.2 Compiladores

Un tratado más o menos completo sobre el tema de los compiladores no está dentro del alcance de este texto. Sin embargo, de incluyen unos conceptos mínimos para ir comprendiendo cómo se aplica la Teoría de Autómatas y Lenguajes Formales al desarrollo de compiladores.

4.2.1 Fases de un compilador

[Lemone] El trabajo de un compilador se divide en las siguiente fases.

- Análisis léxico, análisis lexicográfico o rastreo.
- Análisis sintáctico o análisis gramatical.
- Análisis semántico
- Optimización
- Preparación para la generación de código.
- Generación de código.

Seguidamente se incluyen explicaciones sobre la fase de análisis léxico. No está dentro del alcance de este texto la explicación de las restantes fases.

4.2.2 Analizador léxico

Podemos considerar un programa como una secuencia de sentencias, las sentencias como una secuencia de tokens y los tokens como una secuencia de caracteres ASCII. De acuerdo con esto, algunos tipos de tokens son los números enteros, los números reales, las variables y los operadores. Algunos ejemplos de sentencias son las sentencias de asignación, los ciclos y las sentencias de decisión.

El analizador léxico confirma que los tokens estén bien construidos, no se preocupa de si las sentencias están bien construidas o no. De esto se encargan fases posteriores del compilador.

Un analizador léxico no se preocupará de errores como falta operador entre dos variables, falta la expresión a la derecha de una expresión de asignación, no hay balance de paréntesis etc.

Más concretamente, un analizador léxico es un programa que recibe como entrada una secuencia de caracteres y produce una secuencia de tokens. La secuencia de caracteres de entrada consiste en un programa en algún lenguaje de computador, el cual puede tener errores de compilación. La salida es básicamente la lista de los tokens que conforman el programa con su correspondiente tipo.

Ejemplo

Supongamos que vamos a realizarle el análisis léxico a un programa en un extraño lenguaje de computador, que usa palabras reservadas en español y con operador de asignación representado por el signo :=.

La entrada podría ser un archivo de tipo texto con el siguiente contenido.

```
xyz:=  xyz + 1 + @
si y > 57
     a :=  (35 + b1 b2
fin si
```

Como se puede observar, el programa presenta tres errores de compilación.

La salida del analizador léxico es básicamente la siguiente.

xyz	identificador
:=	operador de asignación
xyz	identificador
+	operador aditivo
1	entero
+	operador aditivo
@	**No reconocido**
si	identificador
y	identificador
>	operador relacional
57	entero
a	identificador
:=	operador de asignación
(	Paréntesis de abrir
35	entero
+	operador aditivo
b1	identificador
b2	identificador
fin	identificador
si	identificador

Los identificadores son los nombres de las variables, de las funciones, de las clases y de los métodos, permitidos en el lenguaje de programación que se está analizando.

Note en el ejemplo que el analizador léxico solo detectó el error del símbolo @, que se supone que no tiene sentido en este lenguaje. El analizador léxico no detectó el error de la falta de balance de los paréntesis, ni el error de dos variables separadas por un espacio.

Estos errores son responsabilidad del analizador gramatical, no del léxico.

No confunda el lenguaje con el que se construye el analizador léxico con el lenguaje del código de entrada al analizador. Por ejemplo, es posible escribir en lenguaje Java un analizador léxico que reciba como entrada código en el extraño lenguaje del ejemplo.

4.2.3 Método extraerSiguienteToken()

En la implementación de un analizador léxico, es muy natural incluir un método que se encargue de extraer el token de la posición actual y a la vez calcular la posición del siguiente. El método se denomina en este texto extraerSiguienteToken(). El analizador léxico obtiene la lista de los tokens de un programa de computador, ejecutando repetidamente este método, cuyas características son las siguientes:

Parámetros de entrada:

- Una cadena de caracteres *a.*
- Una posición *i,* en la cadena de caracteres anterior, a partir de la cual se va a extraer un token.

Valores que retorna:

- El token que se encuentra en la cadena *a* a partir de la posición *i*
- El tipo de este token
- La posición inicial del token que sigue al token extraído

Ejemplo

Si el método extraerSiguienteToken() recibe:

Cadena: *nota=50*

Iniciar la búsqueda en: 0

El método devuelve:

Token: *nota*

Tipo de token: identificador

Posición del siguiente token: 4

Se supuso que las posiciones de la cadena están numeradas desde cero.

La posición hallada, 4, se reingresa al método, en una segunda llamada, para extraer el segundo token, es decir, el método recibe;

Cadena: *nota=50*

Iniciar la búsqueda en: 4

y devuelve

Token: *=*

Tipo de token: operador de asignación

Posición del siguiente token: 5.

De nuevo, reingresamos la posición hallada al método, 5, en una tercera llamada, para hallar el tercer y último token. Es decir, el método recibe:

Cadena: *nota=50*

Iniciar la búsqueda en: 5

y devuelve

Token: *50*

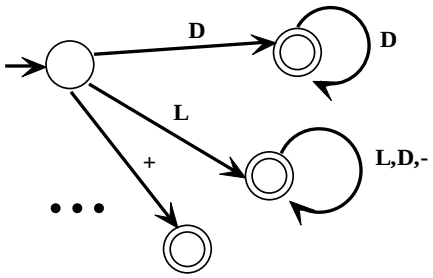
Tipo de token: entero

Posición del siguiente token: 7

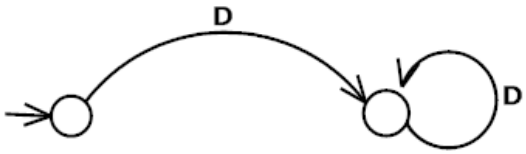
AFD base del analizador léxico.

Lo usual es que el método *extraerSiguienteToken()* llame a otros métodos que se encarguen de extraer un tipo concreto de token, por ejemplo a los métodos *extraerEntero()*, *extraerReal()*, *extraerIdentificador()*, etc.

La algoritmia de estos últimos métodos se basa en un AFD que representa todos los tokens válidos en el lenguaje de los programas de computador que se van a analizar. Observe en la figura una parte de este AFD. Para simplificar el diagrama, se acostumbra omitir los sumideros de rechazo, a excepción de esto, es recomendable que el autómata finito sea determinista.



Por ejemplo, la algoritmia del método *extraerEntero()* se basa en la siguiente parte del AFD anterior:



A continuación se muestra el código Java del método *extraerEntero().* Se extrae un token *entero* del código *codigo* a partir de la posición i*ndice*.

Método Java extraerEntero()

```
public Token extraerEntero
   ( String codigo, int indice)
{
   Token token = null;
   if( esDigito( codigo.charAt(indice)) )
   {
     // Salva la posición inicial del entero
     int indiceInicial=indice;

     // Incrementa el índice para
     // apuntarle al siguiente símbolo del
     // código que se está analizando
     indice++;

     // recorre los demás dígitos del entero
     While
      ( índice < codigo.length( ) &&
      esDigito( codigo.charAt(indice)))
     {
      indice++;
     }

     // Copia el entero en una cadena temporal
     String cadena = codigo.substring
        ( indiceInicial, indice);
     // construye el token para retornarlo.
     // El token se compone de
     // una cadena de caracteres,
     // el tipo y
     // la posición del siguiente token.
     token = new Token
        ( cadena, Token.ENTERO, indice );
   }
   return token;
}
```

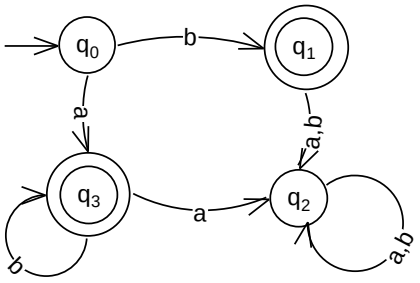
Se puede observar que hay una concordancia entre el autómata y el código Java. Cada una de las transiciones que no son bucles corresponden a un *si … entonces.* Las transiciones que son bucles corresponden a un bucle *mientras*.

4.3 Función de transición, δ

Las transiciones de un autómata finito determinista pueden representarse por medio de una tabla, la tabla de la función de transición δ, en lugar de por un diagrama de transición.

Sea q_i un estado de un AFD y *s* un símbolo del alfabeto de entrada, se define $\delta(q_i, s)$ como el estado q_j al que se llega partiendo de q_i y consumiendo el símbolo *s*.

En el siguiente ejemplo, las transiciones del diagrama quedan representadas exactamente en la tabla de la función δ.



δ	a	b
q_0	q_3	q_1
q_1	q_2	q_2
q_2	q_2	q_2
q_3	q_2	q_3

Cada transición queda representada en la tabla por el cruce de una fila y una columna. Por ejemplo, la transición que parte de q_3, tiene una *a* y llega a q_2 está representada en la tabla por el cruce de la fila q_3 con la columna *a*, donde aparece q_2.

El conjunto de estados finales o de aceptación se representa por *F*, en el ejemplo $F = \{q_1, q_3\}$.

Se puede determinar si una palabra *w* es aceptada por un autómata finito determinista, utilizando la tabla de la función de transición δ. Se aplica esta función a la palabra, símbolo por símbolo, y, si el resultado es un estado de aceptación, la palabra es aceptada. Si el resultado no es un estado de aceptación, la palabra no se acepta. Siguen 3 ejemplos.

El autómata anterior acepta *abb* ya que $\delta(q_0, abb) = \delta(q_3, bb) = \delta(q_3, b) = q_3$, el cual es un estado de aceptación. Observe que en cada paso se halló el resultado de la función aplicado al estado actual y al primer símbolo de la palabra. Se tomó como estado actual en el primer paso a q_0, como era de esperarse.

El autómata no acepta b^3, ya que $\delta(q_0, bbb) = \delta(q_1, bb) = \delta(q_2, b) = q_2$, el cual no es un estado de aceptación.

El autómata tampoco acepta ε, ya que $\delta(q_0, \varepsilon) = q_0$, el cual no es un estado de aceptación.

Definición.

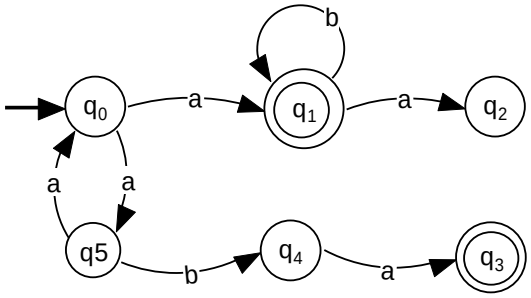
[KELLEY] Un autómata finito M es una tupla de cinco elementos, $M = (Q, \Sigma, s, F, \delta)$, donde:

- Q es una colección finita de estados.
- Σ es un alfabeto de entrada.
- s es el estado inicial, $s \in Q$
- F es la colección de estados finales o de aceptación.
- $\delta : Q \times \Sigma \to Q$ es la función de transición

4.4 Autómatas finitos no deterministas

Los autómatas finitos no deterministas, AFN, no están obligados a cumplir la restricción de los deterministas. No es necesario que de cada estado salga exactamente una transición por cada símbolo del alfabeto.

Ejemplo:



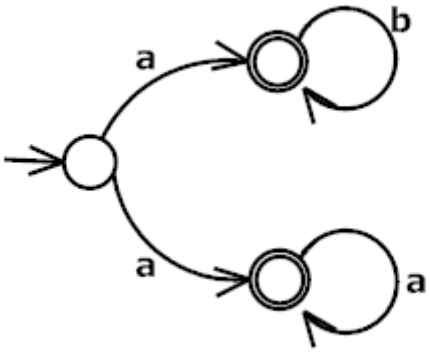
Note que del estado q_2 no sale ninguna transición y del estado q_4 no parte la transición con el símbolo *b*.

Un AFN acepta una palabra cuando existe al menos una ruta que corresponda a la palabra y termine en un estado final o de aceptación. Siguen ejemplos de palabras aceptadas y palabras no aceptadas por el autómata:

Palabra	Ruta que conduce a un estado final	¿La palabra es aceptada?
ab	$q_0 \rightarrow q_1 \rightarrow q_1$	Sí
aa	No hay	No
aba	$q_0 \rightarrow q_5 \rightarrow q_4 \rightarrow q_3$	Sí
abb	$q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1$	Sí
abab	No hay	No

Los AFD son un subconjunto de los AFN, que cumple una restricción especial. Podría pensarse erróneamente que los AFD y los AFN son dos conjuntos disyuntos.

En general, es más fácil elaborar un AFN que un AFD, por ejemplo el lenguaje ab^* ∪ aa^* queda representado por el siguiente autómata.



La operación unión con frecuencia queda representada por una bifurcación en el AFN.

Ejercicios

1) Obtenga un AFN para cada uno de los siguientes lenguajes

   a) $b(aa \cup b)^* \cup b(ba)^*$

   b) $a(bba \cup ba)^*$

   c) $a(ab)^* b \cup ba^*$

   d) $(aa \cup ab)^* a$

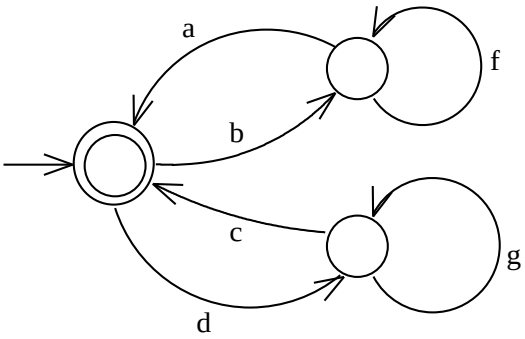
   e) $(aaa \cup aab \cup b)^*$

   f) $a^* \cup b^*$

   g) $(ab)^* \cup b^*$

Ejercicios con respuestas

2)

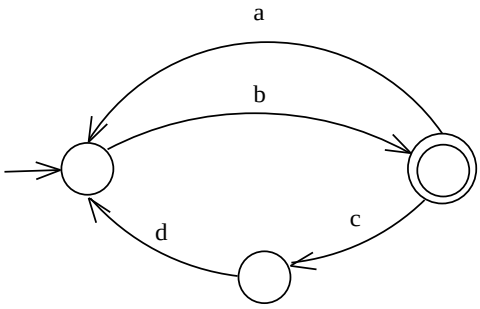


La expresión regular que representa el lenguaje aceptado por el anterior Autómata finito no determinista es (complete):

_____

Respuesta: $(bf^*a \cup dg^*c)^*$

3)

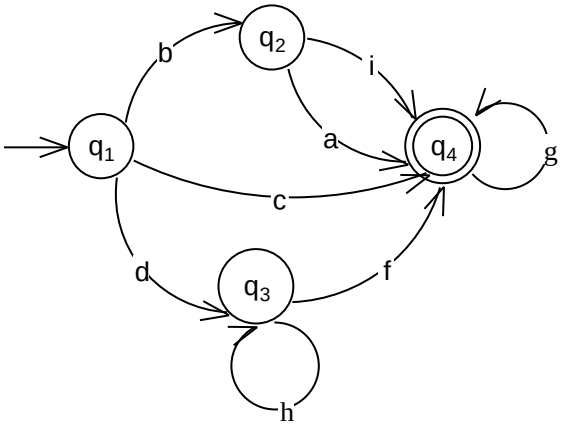


La expresión regular que representa el lenguaje aceptado por el anterior Autómata finito no determinista es (complete):

_____

Respuesta: $b(ab \cup cdb)^*$

4) ¿Cuál es el lenguaje aceptado por el siguiente AFN? (Señale solo uno)



A. $b(a \cup i) \cup c \cup dh^* fg^*$

B. $(ba \cup i \cup c \cup dh^* f)g^*$

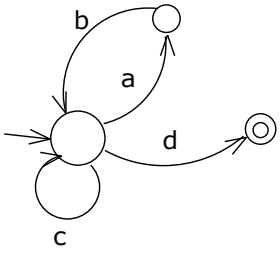
C. $[b(a \cup i) \cup c \cup dh^* fg]^*$

D. $[b(a \cup i) \cup c \cup dh^* f]g^*$

E. $[b(a \cup i)cdh^* f]g^*$

Respuesta: D

5) ¿Cuál es el lenguaje aceptado por el siguiente Autómata finito?



A. $d(ab \cup c)^*$

B. $(ab \cup c)^* d$

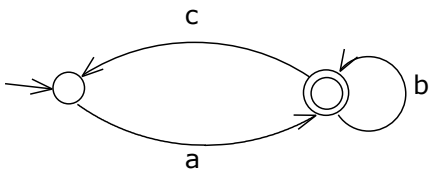
C. $c^*(ab)^* d$

D. $(ab)^* \cup c^* \cup d$

E. $(ab)^* c^* d$

Respuesta: B

6) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. ab^*ca

B. $a(b \cup ca)^*$

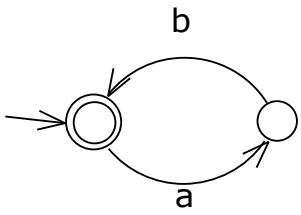
C. $(abc)^*$

D. $a(bca)^*$

E. ab^*c

Respuesta: B

7) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. $(a \cup b)^*$

B. $(ba)^*$

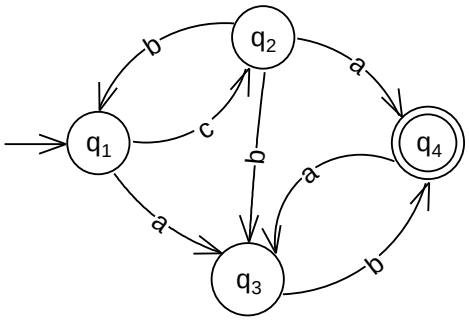
C. $a^* b^*$

D. $ab(ab)^*$

E. $(ab)^*$

Respuesta: E

8) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. $c(a(ab)^* \cup b(ba)^* b) \cup (ab)^* a(ba)^* b$

B. $(cb)^* c(a \cup b(ba)^* b) \cup (ab)^* a(ba)^* b$

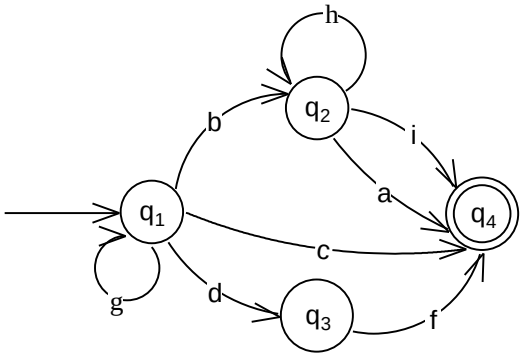
C. $(cb)^* c(a(ab)^* \cup bb) \cup (ab)^* a(ba)^* b$

D. $(cb)^* c(a(ab)^* \cup b(ba)^* b) \cup (ab)^* a(ba)^* b$

E. $(cb)^* c(a(ab)^* \cup b(ba)^* b) \cup a(ba)^* b$

Respuesta: D

9) ¿Cuál es el lenguaje aceptado por el siguiente AFN? (Señale solo uno)



A. $g^*[bh^*(i \cup a) \cup c \cup df]$

B. $g^*bh^*(i \cup a) \cup c \cup df$

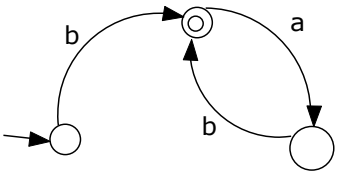
C. $g^*[bh^*i \cup a \cup c \cup df]$

D. $g^*bh^*i \cup a \cup c \cup df$

E. $[bh^*(i \cup a) \cup c \cup df]g^*$

Respuesta: A

10) ¿Cuál es el lenguaje aceptado por el siguiente Autómata finito?



A. $b(a \cup b)^*$

B. ba^*b^*

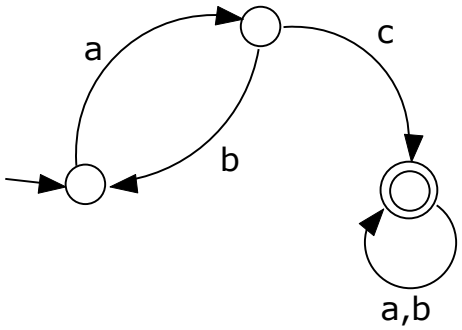
C. $b(ba)^*$

D. $b(a \cup b)$

E. $b(ab)^*$

Respuesta: E

11) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. $a(ab)^* c(a \cup b)^*$

B. $a(ba)^* ca^* b^*$

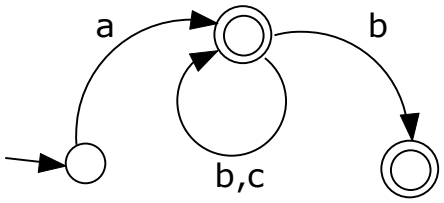
C. $a(ba)^* c(a \cup b)^*$

D. $a(ba)^* \cup c(a \cup b)^*$

E. $a(ba)^* c(ab)^*$

Respuesta: C

12) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. $a(b \cup c)^* b$

B. $a(b \cup c)^* b^*$

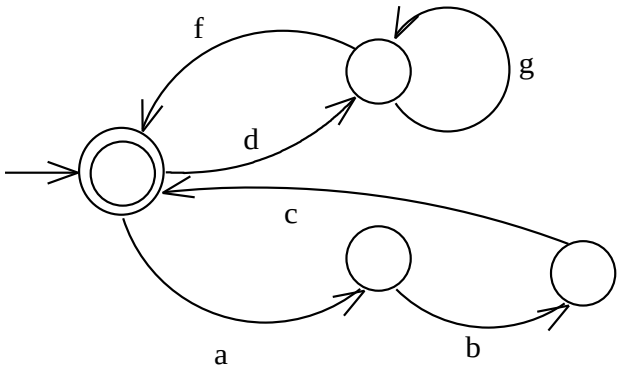
C. ab^*c^*b

D. $a(b \cup c)^* \cup a(b \cup c)^* b$

E. $ab^*c^* \cup ab^*c^*b$

Respuesta: D

13)

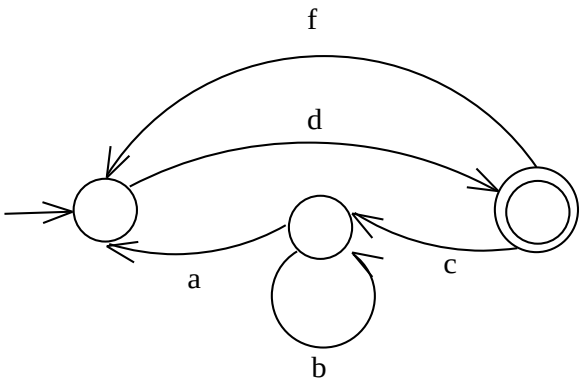


La expresión regular que representa el lenguaje aceptado por el anterior Autómata finito es (complete):

_____

Respuesta: $(dg^*f \cup abc)^*$

14) Escriba la expresión regular que representa el lenguaje aceptado por el siguiente Autómata finito.



Respuesta: $d(fd \cup cb^*ad)^*$

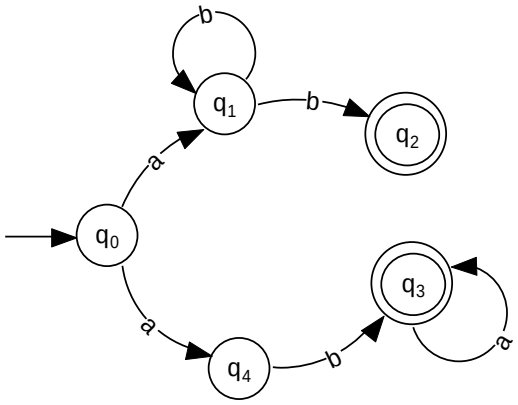
4.4.1 Relación de transición Δ

Es posible representar las transiciones de un AFN mediante una tabla, la tabla de la relación de transición Δ, en lugar de representarlas mediante un diagrama.

Sea q_i un estado de un AFN y *s* un símbolo del alfabeto de entrada, se define $\Delta(q_i, s)$ como el conjunto de todos los estados a los que se puede llegar partiendo de q_i y consumiendo el símbolo *s*.

La relación Δ también se puede aplicar a un conjunto de estados Sea Q un conjuntos de estados de un AFN y *s* un símbolo del alfabeto de entrada, se define $\Delta(Q, s)$ como el conjunto de todos los estados a los que se puede llegar partiendo cualquier estado $q_i \in Q$ y consumiendo el símbolo *s*.

En el ejemplo que sigue, se puede observar que las transiciones del diagrama quedan representadas exactamente en la relación de transición Δ.



Δ	a	b
q_0	$\{q_1, q_4\}$	ϕ
q_1	ϕ	$\{q_1, q_2\}$
q_2	ϕ	ϕ
q_3	$\{q_3\}$	ϕ
q_4	ϕ	$\{q_3\}$

El conjunto de estados finales es $F = \{q_2, q_3\}$

El posible determinar analíticamente si una palabra es aceptada o no por un autómata, con base en la relación de transición Δ. Para ello se aplica la relación de transición al estado inicial y a la palabra, símbolo por símbolo. Si alguno de los estados del resultado final es un estado de aceptación, la palabra es aceptada, de lo contrario, no es aceptada. Siguen varios ejemplos, en los que se determina si una palabra dada es aceptada o no por el AFN anterior.

Ejemplo 1: *ab*

Solución: $\Delta(q_0, ab) = \Delta(\{q_1, q_4\}, b) = \{q_1,\ q_2, q_3\}$. Como $q_2 \in F$ se concluye que la palabra *ab* es aceptada por el autómata.

Ejemplo 2: *a*.

Solución: $\Delta(q_0, a) = \{q_1, q_4\}$. Como $q_1, q_4 \notin F$ se concluye que la palabra *a* no es aceptada por el autómata.

Ejemplo 3: *aab*.

Solución: $\Delta(q_0, abb) = \Delta(\{q_1, q_4\}, bb) = \Delta(\{q_1, q_2, q_3\}, b) = \{q_1, q_2\}$. Como $q_2 \in F$ se concluye que la palabra *ab* es aceptada por el autómata.

Ejemplo 4: ε.

Solución: $\Delta(q_0, \varepsilon) = \{q_0\}$. Como $q_0 \notin F$ se concluye que la palabra ε no es aceptada por el autómata.

4.4.2 Definición

[KELLY] Un autómata finito no determinista es una tupla de cinco elementos $(Q, \Sigma, s, F, \Delta)$, donde:

Q es un conjunto finito de estados

Σ es el alfabeto de entrada

s es uno de los estados de Q, designado como el estado de partida.

F es una colección de estados de aceptación o finales, $F \subseteq Q$

Δ es una relación sobre $(Q \times \Sigma) \times Q$ y se llama relación de transición.

4.4.3 Conversión de un AFN en AFD

Sea M = $(Q, \Sigma, s, F, \Delta)$ un AFN. Es posible obtener un AFD M' = $(P, \Sigma, s', F', \delta)$ equivalente a M, mediante los siguientes pasos.

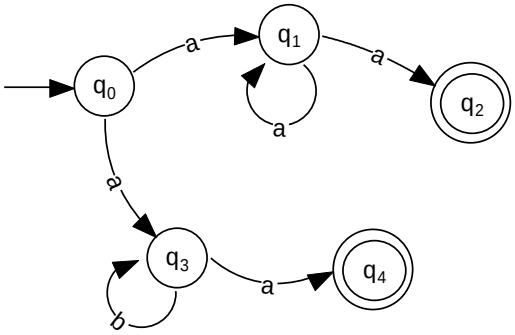
1. Elabore la tabla de la relación de transición, Δ.
2. Obtenga la tabla de la función de transición del AFD, δ, haciendo lo siguiente.

a. Escriba el alfabeto en la parte superior de la tabla. El alfabeto del AFD es igual al alfabeto del AFN.
b. Escriba $\{q_0\}$ en la tabla, como el estado inicial del AFD. Escríbalo así, con llaves y todo.
c. Halle los resultados faltantes de la tabla, con $\delta(p_i, \sigma) = \Delta(p_i, \sigma)$
d. Escriba un nuevo estado igual a cada resultado nuevo de la tabla.
e. Repita los pasos c) y d) hasta que no surjan más estados.

3. Por facilidad, renombre los estados del AFD, como $p_1, p_2, ..., p_n$
4. Los estados finales del AFD serán aquellos a los que pertenezca al menos un estado final del AFN.
5. Dibuje el diagrama de transición del nuevo AFD.

Ejemplo

Obtenga un AFD equivalente al siguiente AFN.



Solución:

i) La tabla de la relación Δ del AFN es la siguiente

Δ	a	b
q_0	$\{q_1, q_3\}$	$\emptyset$
q_1	$\{q_1, q_2\}$	$\emptyset$
q_2	$\emptyset$	$\emptyset$
q_3	$\{q_4\}$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$

ii) La tabla de la función δ del AFD equivalente es la siguiente.

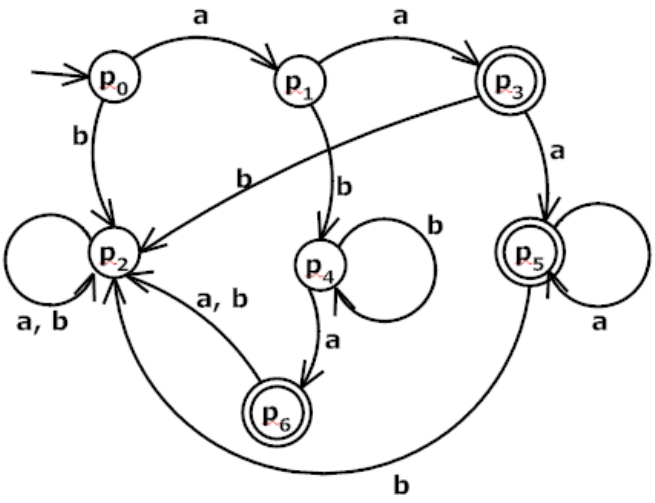
δ	a	B
$\{q_0\}$	$\{q_1, q_3\}$	$\emptyset$
$\{q_1, q_3\}$	$\{q_1, q_2, q_4\}$	$\{q_3\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_1, q_2, q_4\}$	$\{q_1, q_2\}$	$\emptyset$
$\{q_3\}$	$\{q_4\}$	$\{q_3\}$
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\emptyset$
$\{q_4\}$	$\emptyset$	$\emptyset$

iii) La siguiente es la tabla de la función δ, con una notación simplificada.

δ	a	B
p_0	p_1	p_2
p_1	p_3	p_4
p_2	p_2	p_2
p_3	p_5	p_2
p_4	p_6	p_4
p_5	p_5	p_2
p_6	p_2	p_2

iv) El conjunto de estados finales es $F' = \{p_3,\ p_5,\ p_6\}$

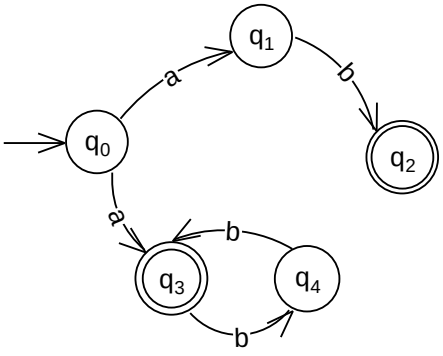
v) El diagrama del AFD equivalente al AFN inicial es el siguiente.



Note que cualquier palabra que acepta este AFD, como *aa*, también es aceptada por el AFN inicial. Igualmente, cualquier palabra que no acepte el AFD, como por ejemplo *aaab*, tampoco es aceptada por el AFN inicial.

Ejercicio con respuesta

1) Aplique el algoritmo de conversión de AFN en AFD al siguiente AFN.



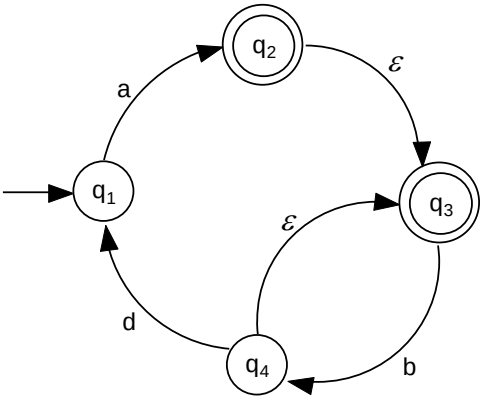
Respuesta: La función de transición del AFD es la siguiente:

δ	a	B
P1	P2	P3
P2	P3	P4
P3	P3	P3
P4	P3	P5
P5	P3	P6
P6	P3	P5

4.5 ε-transiciones

Son transiciones que no consumen ningún símbolo de la palabra. Se etiquetan con el símbolo ε.

Ejemplo de un AFN con ε-transiciones.

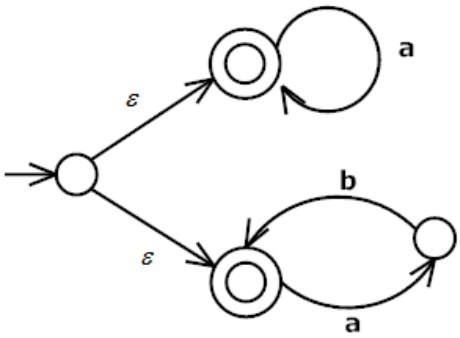


El autómata acepta una palabra si existe al menos una ruta que, utilizando cualquier número de ε-transiciones, comience en el estado inicial, consuma exactamente la palabra y finalice en algún estado final. La ruta puede incluir cualquier cantidad de ε - transiciones, en cualquier parte de ella. Podemos usar ε - transiciones al comienzo de la ruta, en el intermedio, al final o en todos estos sitios a la vez.

En la siguiente tabla se incluyen algunas palabras aceptadas y no aceptadas por el autómata del ejemplo.

Palabra	**Ruta que finaliza en un estado final**	**¿La palabra es aceptada?**
abb	$q_1 \to q_2 \to q_3 \to q_4 \to q_3 \to q_4 \to q_3$	Sí
abda	$q_1 \to q_2 \to q_3 \to q_4 \to q_1 \to q_2$	Sí
abd	No hay	No
ad	No hay	
$abdab^{50}$	$q_1 \to q_2 \to q_3 \to q_4 \to q_1 \to q_2 \to q_3$ $q_4 \to q_3 \to q_4 \to q_3 \to \cdots q_4 \to q_3$	Sí

Las ε-transiciones facilitan aún más la elaboración de los autómatas. Por ejemplo, el lenguaje $a^* \cup (ab)^*$ es representado de una manera muy natural por el siguiente autómata.



Ejercicios

1) Elabore el diagrama de transición de un AFN que acepte cada uno de los siguientes lenguajes.
   a) $a^*ba^* \cup b^*b$
   b) $(ab)^*a^*$

c) $(ab^* \cup ac^*)^*$

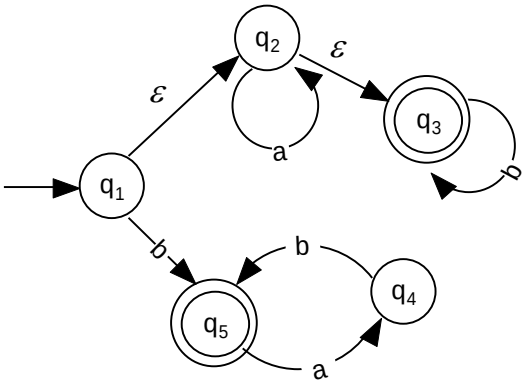
d) $a^*(ba)^* \cup ca^*$

e) $[a^*(bb)^* \cup c^*a]^*$

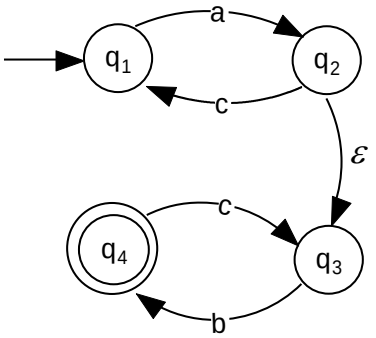
f) $c^*(ac^* \cup b)^*$

2) ¿Cuál el lenguaje aceptado por los siguientes AFN?

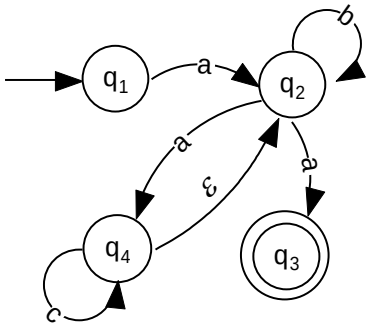
a)



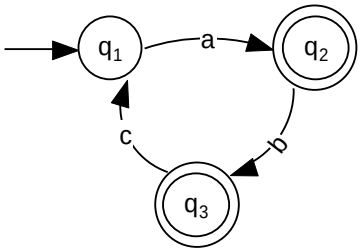
b)



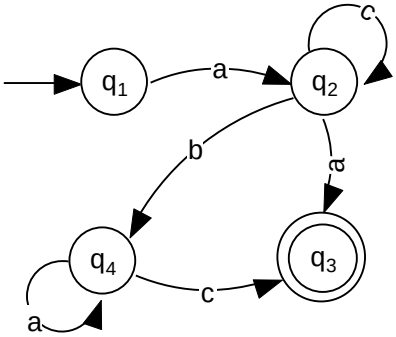
c)



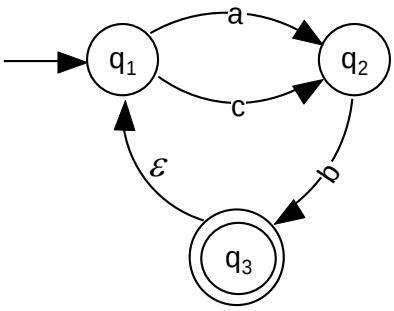
d)



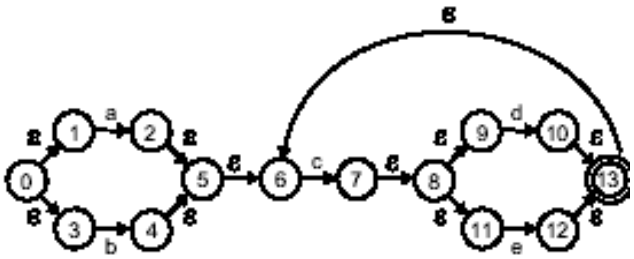
e)



f)



3) El siguiente autómata finito [1]



reconoce un lenguaje determinado. Identifique, cuál de las siguientes expresiones regulares describe ese mismo lenguaje

A) $(a \cup b)c(d \cup e)$

B) $(a \cup b)c(d \cup e)^*$

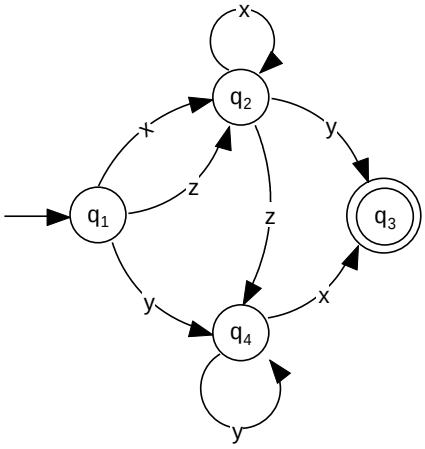
C) $(a \cup b)(c(d \cup e))^*$

D) $(a \cup b)(c(d \cup e))^+$

E) $(a \cup b)\,\varepsilon\, c(d \cup e)$

[1] Ejercicio tomado del examen ECAES del año 2003

4) Dado el autómata[2] M



Una expresión regular que define el lenguaje L(M) es

A) $(((x \cup z)(x^* y \cup z y^* x)) \cup y y^* x)$

B) $(y y^* x) \cup ((x \cup z) x^* ((z y^* x) \cup y))$

C) $(y^* x) \cup ((x \cup z) x^* ((z y x) \cup y))$

D) $(y y^* x)((x \cup z) x^* ((z y^* x) \cup y))$

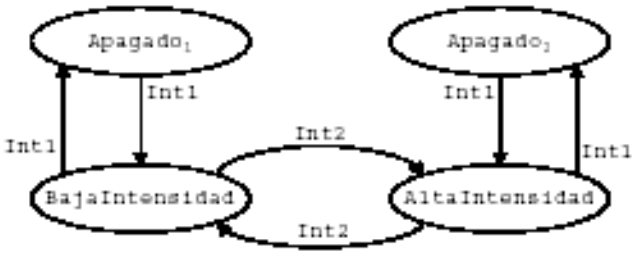
E) $(y y^* x) \cup (x z x^* ((z y^* x) \cup y))$

5) Una lámpara[3] enciende en dos intensidades y tiene dos interruptores, int1, int2. El interruptor int1 se usa para prender y apagar y el interruptor int2 para cambiar de intensidad.

La condición lámpara apagada se representa en los estados $Apagado_1$ y $Apagado_2$, La condición de intensidad baja (alta), se representa en el estado BajaIntensidad (AltaIntensidad).

El siguiente autómata describe el comportamiento de la lámpara

[2] Ejercicio tomado del examen ECAES del año 2003
[3] Ejercicio tomado del examen ECAES del año 2003

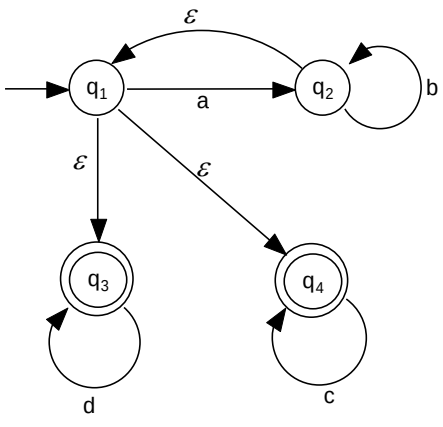


A partir de este autómata se deduce que

A) Cuando se enciende la lámpara siempre queda en AltaIntensidad.
B) Si se apaga la lámpara cuando está en BajaIntensidad, al encenderla nuevamente queda en AltaIntensidad.
C) Si se apaga la lámpara cuando está en BajaIntensidad al encenderla nuevamente queda en BajaIntensidad
D) Si se apaga la lámpara cuando está en BajaIntensidad ya no se puede volver a encender
E) Si al encender la lámpara queda en BajaIntensidad, sólo llega a AltaIntensidad pasando antes por un estado de apagado

Ejercicios con respuestas

6) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. $(ab^*)^*(d^* \cup c^*)$

B. $ab^* \cup d^* \cup c^*$

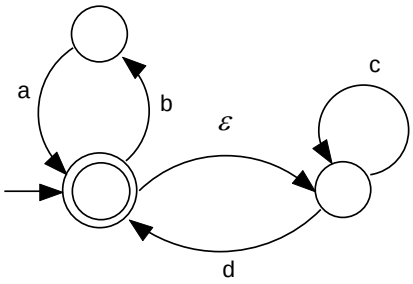
C. $(ab^* \cup d \cup c)^*$

D. $(ab^*)^*(d \cup c)^*$

E. $ab^* d^* c^*$

Respuesta: A

7) ¿Cuál es el lenguaje aceptado por el siguiente AFN?



A. $(ba \cup c^* d)^*$

B. $(ba)^*(c^* d)^*$

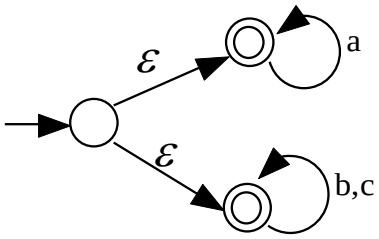
C. $(ba)^* \cup (c^* d)^*$

D. $(ab \cup c^* d)^*$

E. $(b \cup a)^* c^* d$

Respuesta: A

8)



¿Cuál es el lenguaje aceptado por el AFN de la figura?

A. $a^*(b \cup c)^*$
B. $a^* b^* c^*$
C. $a^* \cup b^* \cup c^*$
D. $a^* \cup b^* c^*$
E. $a^* \cup (b \cup c)^*$

Respuesta: E

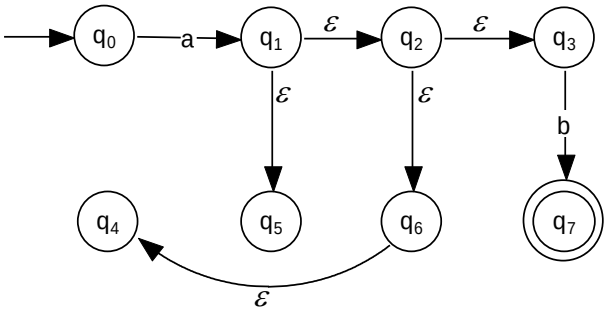
4.6 ε-cerradura

La ε-cerradura se emplea en el algoritmo para convertir un AFN con $\varepsilon - transiciones$ en un AFN sin $\varepsilon - transiciones$. Este algoritmo será tratado más adelante.

La ε-cerradura de un estado q_i es el conjunto de estados a los que se puede llegar partiendo de q_i y utilizando únicamente ε-transiciones. Se puede utilizar cualquier cantidad de ε-transiciones.

Ejemplo

Halle la ε- cerradura de los estados q_1, q_6 y q_3 del siguiente AFN.



Solución:

$\varepsilon - c(q_1) = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\varepsilon - c(q_6) = \{q_6, q_4\}$

$\varepsilon - c(q_3) = \{q_3\}$

4.7 Algoritmo para eliminación de ε-transiciones

Sea M = $(Q, \Sigma, s, F, \Delta)$ un AFN con ε-transiciones. Puede obtener un AFD M' = $(Q, \Sigma, s, F', \Delta')$ equivalente a M, mediante los siguientes pasos.

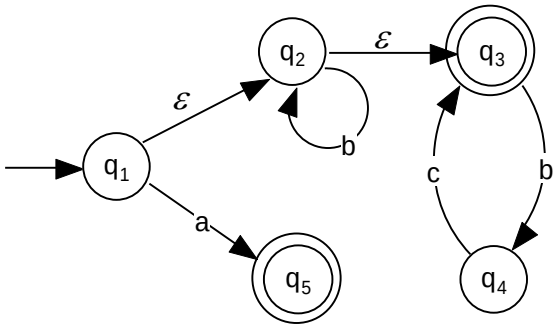
1. Halle las ε-cerraduras de todos los estados de M.
2. Los estados finales del nuevo AFN serán aquellos cuya ε-cerradura tiene al menos un estado final.
3. Elabore la tabla de la relación de transición Δ' del nuevo AFN. En $\Delta'(q_i, \sigma)$ va el conjunto de todos los estados a los que se puede llegar desde q_i, consumiendo el símbolo σ y cualquier cantidad de ε-transiciones. La ε-transiciones pueden ir en cualquier

lugar de la ruta: al comienzo, al final, en el intermedio o en varios lugares de estos a la vez.

4. Dibuje el diagrama de transición del nuevo AFN.

Ejemplo

Dibuje el diagrama de transición de un AFN sin ε -transiciones equivalente al siguiente AFN.



Solución:

i) Las ε-cerraduras de los estados del autómata son las siguientes

$\varepsilon - c(q_1) = \{q_1, q_2, q_3\}$

$\varepsilon - c(q_2) = \{\ q_2, q_3\}$

$\varepsilon - c(q_3) = \{\ q_3\}$

$\varepsilon - c(q_4) = \{\ q_4\}$

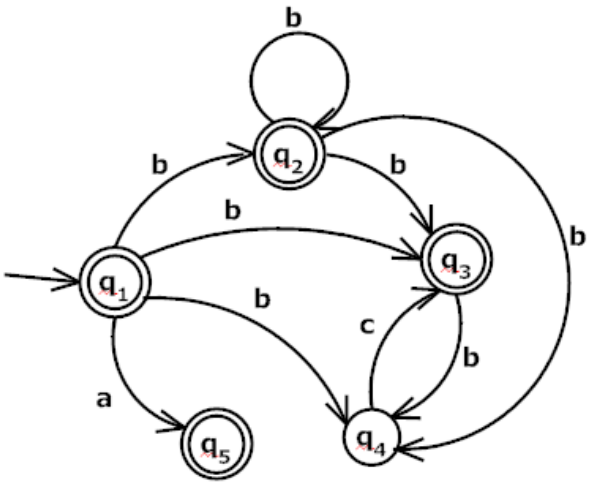
$\varepsilon - c(q_5) = \{\ q_5\}$

ii) El conjunto de estados finales del nuevo autómata es $F' = \{q_1, q_2, q_3, q_5\}$

iii) La tabla de la relación de transición del nuevo autómata es la siguiente:

Δ'	a	b	c
q_1	$\{q_5\}$	$\{q_2, q_3, q_4\}$	$\emptyset$
q_2	$\emptyset$	$\{q_2, q_3, q_4\}$	$\emptyset$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$
q_5	$\emptyset$	$\emptyset$	$\emptyset$

iv) El diagrama del AFN sin ε-transiciones equivalente al AFN inicial es el siguiente:



Note que cualquier palabra que acepte el nuevo autómata, como por ejemplo *bbc*, también es aceptada por el autómata original. Igualmente, cualquier palabra que no acepte el nuevo autómata, como por ejemplo *bbcb*, tampoco es aceptada por el autómata inicial.

4.8 Teorema de Kleene

Un lenguaje es regular si y solo si es aceptado por un autómata finito.

El teorema de Kleene expresa que si tenemos una una expresión regular, por extraña que sea, siempre será posible obtener un autómata que acepte el lenguaje representado por la expresión regular. Igualmente, si tenemos un autómata, siempre es posible obtener una expresión regular que represente el lenguaje aceptado por el autómata.

La demostración de este teorema es más bien extensa y compleja. No se incluye en este texto, pero puede consultarse en la bibliografía, [KELLY].

4.9 Proyecto de análisis léxico

En esta sección se propone como proyecto la implementación del analizador léxico de un lenguaje de programación diseñado por los estudiantes. El analizador se puede desarrollar en cualquier lenguaje de programación conocido por los estudiantes. En este proyecto, el estudiante aplicará los conceptos aprendidos durante el curso, en especial lo relacionado con expresiones regulares y con autómatas finitos deterministas.

El procedimiento propuesto para desarrollar el proyecto es seguir las instrucciones de la sección 4.9.1 y diligenciar la plantilla de la sección 4.9.2.

4.9.1 Instrucciones

Objetivos

Este proyecto gira alrededor de un lenguaje de programación supuesto. El lenguaje tiene 16 tipos de token, 8 de los cuales ya

están especificados en la plantilla de la siguiente sección. El estudiante debe especificar los 8 restantes.

Los objetivos de proyecto son los siguientes:

a) Terminar de especificar los token del lenguaje de programación, de manera de que no queden muy similares a los token de los lenguajes conocidos.

b) Elaborar el diagrama de transición de un autómata finito determinista que acepte todos los token del lenguaje. En este diagrama, se puede omitir el sumidero de rechazo, por simplicidad.

c) Desarrollar el analizador léxico del lenguaje, basándose en el autómata elaborado en el objetivo b).

   Nota: En las secciones 0, 0, 0, 0 y 0 se incluye respectivamente:

   - Un prototipo de la interfaz del software
   - El caso de uso que se debe implementar
   - Requerimientos
   - Consideraciones a tener en cuenta en la construcción del analizador léxico
   - Procedimiento para diligenciar la plantilla

d) Implementar al menos un método de prueba *JUnit* para cada método de extraer un tipo de token. Por ejemplo, se podría implementar el método *testExtraerOperadorLogico()*, para verificar el correcto funcionamiento del método *extraerOperadorLogico()*.

Nota. Los entregables correspondientes a los 4 objetivos son el software desarrollado y la plantilla de la sección 4.9.2 diligenciada.

Según estudiantes que han desarrollado este proyecto en grupos de máximo 3 estudiantes, el tiempo promedio requerido por estudiante es de 18 horas 8 minutos.

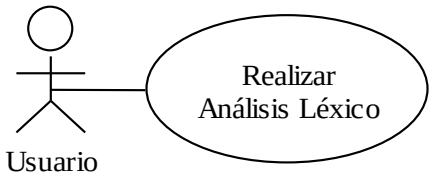
Prototipo de la interfaz del analizador léxico

Archivo	Análisis Léxico	
AB:=cde; X:=”hola”; ¿”hola		
AB	Identificador	
:=	Operador de asignación	
cde	Identificador	
;	Separador de sentencias	
X	Identificador	
:=	Operador de asignación	
“hola”	Cadena de caracteres	
;	Separador de sentencias	
¿	No reconocido	
“hola	Cadena mal delimitada	

En el ejemplo de la interfaz, se analiza código en lenguaje Pascal, pero en el proyecto debe analizarse código en lenguaje que diseñen los estudiantes.

Pueden variar la presentación de la interfaz. La dada es solo para facilitar la comprensión de lo que se requiere.

Caso de uso Realizar análisis léxico



- El usuario selecciona la opción *análisis léxico.*

- El sistema descompone en tokens el código que se encuentra en el área de edición

- El sistema imprime la lista de tokens, cada uno con su tipo correspondiente y en el mismo orden en el que se encuentran en el área de edición.

Notas:

- No deben aparecer en la interfaz espacios en blanco o saltos de línea clasificados como token no reconocidos
- Las cadenas que no se puedan clasificar de acuerdo con el lenguaje diseñado deben aparecer en la interfaz como *ni reconocidas*.

Requerimientos del analizador léxico

- Deben incluir el archivo de extensión jar.

- El archivo de extensión jar debe poderse ejecutar sin importar en que carpeta esté copiado.

- Pueden implementar el analizador en el lenguaje de programación de su preferencia. En caso de que usen un lenguaje diferente a Java, deben realizar lo equivalente a pruebas JUnit en ese lenguaje.

- La implementación debe incluir varios métodos para extraer token determinados, como *extraerOperadorDeAsignación()*, *extraerOperadorAditivo()* etc.

- Para la construcción de los métodos que extraen los tokens, deben basarse en el diagrama del autómata finito elaborado en el objetivo b).

- El profesor debe poder ejecutar el programa sin necesidad de leer manuales, archivos *léame*, ni de llamar a los desarrolladores.

- La presentación de la pantalla debe ser clara y ordenada. No debe haber largas animaciones al comienzo de la ejecución que no puedan ser interrumpidas.

- Deben usar estándares de programación apropiados, como por ejemplo:

    - Uso de comentarios
    - Estructuración del programa en varias clases
    - Clases diferentes para representar el mundo y para representar la interfaz
    - No abreviar los nombres de los atributos ni de los métodos.
    - No sobrecargar de responsabilidades los métodos ni las clases

Recuerde que

Un analizador léxico solo tiene la responsabilidad de detectar errores léxicos, no tiene la responsabilidad de detectar errores gramaticales ni semánticos. Por ejemplo, debe detectar el error de cadena mal delimitada y el error de token no reconocido. No tiene la responsabilidad de detectar los siguientes errores:

- Dos signos de multiplicación seguidos
- No hay balance de paréntesis
- Falta la variable de la izquierda del operador de asignación
- Variable no declarada
- Tipo de variable inválido

Procedimiento para diligenciar la plantilla

Para diligenciar la plantilla realice lo siguiente:

- Reemplace lo que esté entre paréntesis angulares, <>, por lo que corresponda y elimine estos paréntesis.
- Agregue el contenido de cada sección, según las instrucciones que se encuentran entre corchetes, [].

- Borre lo que se encuentre entre corchetes, [], cuando ya no lo necesite, y elimine también los corchetes.

4.9.2 Plantilla

Tokens del lenguaje

[Complete en esta sección las especificaciones de los token que faltan, de manera que no sean muy similares a los de los lenguajes más conocidos

Para especificar los token se utilizarán las siguientes convenciones.

- L: Letra mayúscula o minúscula.
- M: Letra mayúscula
- m: Letra minúscula
- D: Dígito; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- _: Guión bajo
- H: Dígito hexadecimal; 0,1,2,3, 4, 5, 6, 7, 8, 9, 0, A, B, C, D, E, F, a, b, c, d, e, f
- S: Cualquier símbolo excepto los delimitadores de cadena.]

A continuación se listan los tipos de token del lenguaje.

Entero corto: $DD^* \_ c$

Entero largo: <Expresión regular>

Hexadecimales: <Expresión regular>

Dinero: $\$((DDD,\cup DD,\cup D,\cup \varepsilon)(DDD,)^* DDD \cup DD \cup D)$

Hora: DD:DD

Reales en notación normal: <Expresión regular>

Reales en notación científica: <Expresión regular>

Identificadores: $(L \cup \_)(L \cup DL \cup \_\, L)^*$

Operadores aditivos: $+ \cup -$

Operadores multiplicativos: $* \cup / \cup \%$

Operadores relacionales: <Expresión regular para 6 operadores relacionales, al menos 3 de ellos deben ser de más de un carácter>

Operadores lógicos: <Expresión regular para los operadores lógicos y, o y no. Al menos uno de estos operadores deben ser de al menos dos caracteres>

Paréntesis de abrir: (

Paréntesis de cerrar:)

Operadores de asignación: <Expresión regular para 3 operadores de asignación, dos de ellos con al menos dos símbolos>

Cadenas de caracteres: <Expresión regular, la cadena debe estar delimitada por al menos dos caracteres a cada extremo. Por ejemplo, en Python, ""Hola"" es una cadena válida >

Autómata finito determinista

[Incluya aquí el diagrama de transición de un autómata finito determinista que acepte todos los tokens del lenguaje. Por sencillez, puede omitir el sumidero de rechazo.]

Foto de la GUI después de realizar un análisis léxico

[Incluya aquí una imagen del software inmediatamente después de realizar un análisis léxico]

Foto de las pruebas JUnit

[Incluya aquí una imagen de la interfaz de Eclipse inmediatamente después de realizar las pruebas JUnit. Debe haber al menos un método de prueba por cada método de extraer un tipo de token. Por ejemplo, podrían elaborar el método de prueba testExtraerReal(), para probar el método extraerReal().

5. Lenguajes independientes del contexto

En este capítulo, se trata la teoría necesaria para la especificación de aspectos gramaticales de un lenguaje de programación. Estos aspectos se relacionan con las reglas que deben cumplir las sentencias de un lenguaje de programación, para que estén bien construidas.

Para representar los aspectos gramaticales de un lenguaje de programación se utilizan gramáticas independientes del contexto, gramáticas BNF y autómatas de pila no deterministas. Se aclara que además de los aspectos gramáticas, estos elementos pueden representar los aspectos léxicos.

Se recuerda que los autómatas finitos y las expresiones regulares se utilizan para especificar aspectos léxicos de un lenguaje de programación, no permiten representar los aspectos gramaticales. Es decir, las gramáticas independientes del contexto, las gramáticas BNF y los autómatas de pila no deterministas pueden representar lo que representan las expresiones regulares y los autómatas y mucho más.

La parte léxica de un lenguaje también puede representarse mediante gramáticas regulares, que se tratarán más adelante.

Los lenguajes independientes del contexto, LIC, son un conjunto más grande que los lenguajes regulares y los incluyen. En realidad, Los lenguajes regulares son solo una pequeña parte de los lenguajes independientes del contexto. Aun así, los lenguajes independientes del contexto no son todos los lenguajes posibles, aún quedan lenguajes por fuera de este conjunto.

Un ejemplo de un lenguaje regular y por lo tanto independiente del contexto es $\{a^i \mid i \geq 0\}$.

Un ejemplo de un lenguaje independiente del contexto que no es regular es $\{a^i b^i \mid i \geq 0\}$.

Un ejemplo de un lenguaje que es no es ni regular, ni independiente del contexto, es $\{a^i b^i c^i \mid i \geq 0\}$

5.1 Gramática independiente del contexto

Un Lenguaje Independiente del Contexto se define como un que es generado por una Gramática Independiente del Contexto o GIC [KELLY]. Por lo tanto se explicará primero en qué consiste una gramática de este tipo.

Con frecuencia usamos reglas para precisar que es correcto sintácticamente o no en un lenguaje, por ejemplo las siguientes reglas.

- Una forma de conformar una oración en castellano es con un sujeto, un verbo, un complemento directo y un complemento circunstancial. Abreviadamente:
  O → S V D C

- Una forma de conformar una sentencia de asignación es con una variable, un operador de asignación y una expresión, en este orden. Abreviadamente,
  A → V O X

Las gramáticas independientes del contexto o GIC constan de especificaciones similares a las anteriores, denominadas producciones. El siguiente es un ejemplo de una GIC.

$S \rightarrow AB$
$A \rightarrow aA \mid Ab \mid \varepsilon$
$B \rightarrow aB \mid a$

Los símbolos se leen de la siguiente manera:

$\rightarrow$	puede ser, se compone de
|	O

A continuación se explica la terminología asociada con las gramáticas independientes del contexto.

Producción

Las producciones son reglas que permiten derivar en un paso una palabra a partir de un símbolo. Por ejemplo y de acuerdo con la GIC anterior, se puede derivar *Ab* a partir de *A*, gracias a la producción $A \rightarrow Ab$

La gramática anterior consta de 6 producciones.

- $S \rightarrow AB$
- $A \rightarrow aA$
- $A \rightarrow Ab$
- $A \rightarrow \varepsilon$
- $B \rightarrow aB$
- $B \rightarrow a$

No terminal

Los no terminales son los símbolos que pueden derivar palabras. En el ejemplo, los no terminales son S, A, B. Por convención, se representan los no terminales con letras mayúsculas.

Símbolo inicial

El símbolo inicial es el no terminal a partir del cual se comienza un proceso de derivación en varios pasos. En el ejemplo, el símbolo inicial es *S*. Por convención se usa S para representar el no terminal inicial.

Terminal

Los terminales son símbolos que no pueden derivar palabras. En el ejemplo, los terminales son *a* y *b*. Por convención representaremos los terminales con letras minúsculas.

Derivación

La derivación es un proceso mediante el cual se deriva una palabra en varios pasos comenzando por el símbolo inicial de la GIC. La siguiente es la derivación de la palabra bba utilizando la gramática del ejemplo.

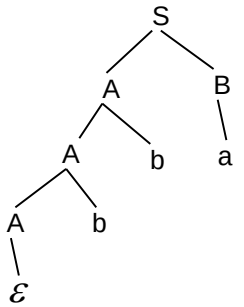
$$S \Rightarrow \mathbf{A}B \Rightarrow \mathbf{Ab}B \Rightarrow AbbB \Rightarrow bbB \Rightarrow bba$$

El símbolo $\Rightarrow$ se lee produce, deriva, genera.

Partimos de S y mediante cinco pasos llegamos a *bba*. Solo podemos realizar los pasos permitidos por la gramática. Por ejemplo en el segundo paso se reemplazó A por Ab, porque esto lo permite la producción $A \rightarrow Ab$.

Árbol de derivación o análisis

La derivación de una palabra mediante una GIC también puede ser representada por un árbol denominado árbol de derivación o de análisis. Por ejemplo, la derivación anterior puede ser representada por el siguiente árbol de derivación:



Leyendo las hojas de izquierda a derecha obtenemos la palabra derivada *bba*.

Lenguaje generado por una gramática

Sea *G* una gramática, el lenguaje generado por la gramática, *L(G)*, es el lenguaje de todas las palabras que la gramática *G* permite derivar. Por ejemplo el lenguaje generado por la gramática:

$$S \rightarrow aS|Sb|\varepsilon$$

es el lenguaje $\{a^i b^j | i \geq 0, j \geq 0\}$.

Se puede observar que la anterior gramática deriva exactamente las palabras de lenguaje, ni una menos, ni una más. El lector puede comprobar fácilmente que por ejemplo deriva las palabras $a^5 b^2$, a, ε, ab, a^7.

Ejercicios con respuestas

1)

$$S \rightarrow aS \mid bS \mid c$$

De acuerdo con la GIC anterior responda V o F

a. La GIC deriva la palabra *ca* _____

b. La GIC deriva la palabra *c* _____

c. La GIC deriva la palabra *cb* _____

d. La GIC deriva la palabra *abababbac* _____

e. La GIC deriva la palabra *a* _____

Respuesta: F V F V F

2)

$$S \rightarrow aSb \mid c$$

De acuerdo con la GIC anterior responda V o F

a. La GIC deriva la palabra *ac* _____

b. La GIC deriva la palabra *acbb* _____

c. La GIC deriva la palabra *abc* _____

d. La GIC deriva la palabra aa*aacbbbb* _____

e. La GIC deriva la palabra *c* _____

Respuesta: F F F V V

Ejercicios

3) Derive las palabras *aba* y *bb*, si es posible, a partir de la siguiente GIC:

$S \rightarrow AB$
$A \rightarrow aA \mid Ab \mid \varepsilon$
$B \rightarrow aB \mid a$

En los ejercicios 4) a 13), obtenga una GIC que derive cada uno de los lenguajes dados

4) $\{a^i b \mid i \geq 0\}$
5) $\{a^{2i} \mid i \geq 0\}$
6) $\{a^{2i} b^j \mid i \geq 0, j \geq 0\}$
7) $\{a^i b^i \mid i \geq 0\}$
8) $\{(ab)^i (ba)^i \mid i \geq 0\}$
9) $\{a^i b^i a^{2j} \mid i \geq 0, j \geq 0\}$
10) $\{a^i a b^i a^{2j} b^j \mid i \geq 0, j \geq 0\}$
11) $\{w \mid w \text{ tiene un número par de símbolos}\}$ sobre {a, b }
12) $\{a^{2i+1} b^j c^{j+1} a^i \mid i \geq 0, j \geq 1\}$
13) { w | w se lee igual de derecha a izquierda que de izquierda a derecha } sobre {a, b, c}

Cuál es el lenguaje generado por las GIC de los numerales 12) a 15)

14) $S \rightarrow bbbS \mid \varepsilon$
15) $S \rightarrow aSb \mid ab$
16) $S \rightarrow aaS \mid aaaa$
17) $S \rightarrow aAb$
$A \rightarrow bAa \mid a$

Ejercicios con respuestas

18) ¿Cuál es el lenguaje derivado por la siguiente GIC?

$$S \to aSd \mid A$$
$$A \to bAc \mid \varepsilon$$

A. $\{a^i b^j c^j d^i \mid i \geq 0,\ j \geq 0\}$

B. $\{a^i b^j c^k d^m \mid i \geq 0,\ j \geq 0, k \geq 0, m \geq 0\}$

C. $\{(abcd)^i \mid i \geq 0\}$

D. $\{a^i b^j c^i d^j \mid i \geq 0,\ j \geq 0\}$

E. $a^* b^* c^* d^*$

Respuesta: A

19) ¿Cuál es el lenguaje derivado por la siguiente GIC?

$$S \to aSb \mid A$$
$$A \to cA \mid Ad \mid \varepsilon$$

Respuesta: $\left\{a^i c^j d^k b^i \middle| i \geq 0, j \geq 0, k \geq 0\right\}$

20) ¿Cuál es el lenguaje derivado por la siguiente GIC?

$$S \to cS \mid Sd \mid A$$
$$A \to aAb \mid \varepsilon$$

Respuesta: $\{c^i a^k b^k d^j | i \geq 0, j \geq 0, k \geq 0\}$

21) ¿Cuál es el lenguaje derivado por la siguiente GIC?

$$S \rightarrow abS \mid Sc \mid \varepsilon$$

A. $\{(ab)^i c^j \mid i \geq 1,\ j \geq 1\}$
B. $\{(ab)^i c^i \mid i \geq 0, i \geq 0\}$
C. $\{(ab)^i c^j \mid i \geq 0,\ j \geq 0\}$
D. $\{a^i b^j c^k \mid i \geq 0,\ j \geq 0, k \geq 0\}$
E. $\{(abc)^i \mid i \geq 0\}$

Respuesta: C

22)

$S \rightarrow aS \mid b$

¿Cuál es el lenguaje derivado por la anterior GIC?

A. $\{ba^i \mid i \geq 0\}$
B. $\{a^i b^j \mid i \geq 0,\ j \geq 0\}$
C. $\{a^i b \mid i \geq 1\}$
D. $\{a^i b \mid i \geq 0\}$
E. $\{(ab)^i \mid i \geq 0\}$

Respuesta: D

23) ¿Cuál de las GIC deriva el siguiente lenguaje?

$$(a \cup b)^* c^* d^*$$

A. S→ aS | bS | Sd | A
   A→ cA | ε

B. S→ aS | bS | Sc | Sd | ε

C. S→ AB

A→ aA | Ab | ε
B→ cB | Bd | ε

D. S→ aS | bS | cS | dS | ε

E. S→ aS | bS | A
A→cA | dA | ε

Respuesta: A

24) Obtenga una gramática cuyo lenguaje derivado sea

$$\{(ab)^i(cd)^{i+1} \mid i \geq 0\}$$

Respuesta: $S \rightarrow abScd|cd$

25) Obtenga una gramática cuyo lenguaje derivado sea

$$\{(ac)^{i+1}(bd)^i \mid i \geq 0\}$$

Respuesta: $S \rightarrow acSbd|ac$

26) Complete: La gramática

$$S \rightarrow \_\_\_\_\_ \mid bc$$

deriva el lenguaje

$$\{bca^i \mid i \geq 0\}$$

Respuesta: Sa

27) Obtenga gramáticas independientes del contexto para los siguientes lenguajes:

a) { w ∈ {a , b }* | w tiene el doble de aes que de bes }

b) { $a^n ab^n b^m$ | n ≥ 0, m ≥ 1 }

c) $\{ (ab)^n(cd)^m(dc)^m(ac)^n \mid n \geq 0, m \geq 0 \}$

d) $\{ a^m b^n c^p a^p b^n c^m \mid n \geq 0, p \geq 1, m \geq 0 \}$

e) $\{ a^n b^m \mid n \geq 0, m \geq 0 \text{ y } (n \neq 0 \text{ o } m \neq 0) \}$

f) $\{ a^n b^{3n} \mid n \geq 0 \}$

g) $\{ w \in \{a,b\}^* \mid w = w^I \}$

h) $\{ a^{2n+1} \mid n \geq 0 \}$

i) $\{ w \in \{a,b\}^* \mid$ w tiene un número par de bes $\}$

j) $\{(ab)^n(cd)^m(dc)^m(ba)^n \mid n \geq 1, m \geq 1\}$

k) $\{w \in \{a,b\}^* \mid w \text{ tiene la misma cantidad de } aes \text{ que de } bes\}$. La gramática debe generar por ejemplo la palabra: abbaabbaab

l) $\{a^i b^j c^k \mid (i = j \text{ o } j = k), i \geq 0, j \geq 0, k \geq 0\}$

Respuestas:

a) $S \rightarrow SaSaSbS \mid SaSbSaS \mid SbSaSaS \mid \varepsilon$

b) $S \rightarrow AB$

$S \rightarrow aAb \mid a$

$S \rightarrow bB \mid b$

c) $S \rightarrow abSac \mid A$

$A \rightarrow cdAdc \mid \varepsilon$

d) $S \rightarrow aSc \mid A$

$A \rightarrow bAb \mid B$

$B \rightarrow cBa \mid ca$

e) $S \rightarrow aA \mid Ab$

$A \rightarrow aA \mid Ab \mid \varepsilon$

f) $S \rightarrow aSbbb \mid \varepsilon$

g) $S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$

h) $S \rightarrow aaS \mid a$

i) $S \rightarrow aS \mid Sa \mid bSb \mid \varepsilon$

j) $S \rightarrow abSba \mid abAba$

$A \rightarrow cdAdc \mid cddc$

k) $S \rightarrow SaSbS \mid SbSaS \mid \varepsilon$

l) $S \rightarrow A \mid B$

$A \rightarrow Ac \mid C$

$C \rightarrow aCb \mid \varepsilon$

$B \rightarrow aB \mid D$

$D \rightarrow bDc \mid \varepsilon$

28) ¿Cuál es el lenguaje representado por las siguientes GIC?

a) $S \rightarrow AB$

$A \rightarrow aaA \mid b$

$B \rightarrow bBb \mid a$

b) $S \rightarrow aSab \mid \varepsilon$

c) $S \rightarrow aAbB$

$A \rightarrow aaA \mid \varepsilon$

$B \rightarrow bBb \mid b$

d) $S \rightarrow A \mid B \mid C$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bbBa \mid \varepsilon$

$C \rightarrow bbaC \mid bba$

e) $S \rightarrow SS \mid aa$

Respuestas

a) $\{ (aa)^n bb^m ab^m \mid m \geq 0, n \geq 0 \}$

b) $\{ a^n (ab)^n \mid n \geq 0 \}$

c) $\{ a(aa)^n bb^m bb^m \mid n \geq 0, m \geq 0 \}$

d) $\{ a^i \mid i \geq 0 \} \cup \{ (bb)^i a^i \mid i \geq 0 \} \cup \{ (bba)^j \mid j \geq 1 \}$

e) $\{ a^{2n} \mid n \geq 1 \}$

Definición

[KELLY] Una gramática independiente del contexto es una 4-tupla $G = (N, \Sigma, S, P)$, donde tenemos lo siguiente.

- N es una colección finita de no terminales
- Σ es un alfabeto, también conocido como conjunto de terminales.
- *S* es un no terminal determinado que se llama símbolo inicial
- $P \subseteq N \times (N \cup \Sigma)^*$ es un conjunto de producciones.

El lenguaje generado por la GIC *G* se denota por *L(G)* y se denomina un lenguaje independiente del contexto, LIC.

5.2 Aplicaciones

A continuación de dan ejemplos de gramáticas independientes del contexto que representan elementos de los lenguajes de programación de computadores. Se aclara que en cada ejemplo no se vuelven a repetir producciones que ya están escritas en ejemplos previos.

1. GIC para los enteros sin signo

   E $\rightarrow$ ED | D

   D $\rightarrow$ 0 | 1 | 2 | … | 9

Es fácil comprobar que esta gramática deriva por ejemplo el entero 789.

$$E \Rightarrow ED \Rightarrow EDD \Rightarrow DDD \Rightarrow 7DD \Rightarrow 78D \Rightarrow 789$$

2. GIC para los enteros con signo

   C $\rightarrow$ SE

   S $\rightarrow$ + | - | ε

3. GIC para los identificadores del lenguaje de programación Java

   I $\rightarrow$ IL | ID | I$ | I_ | L | $ | _

   L $\rightarrow$ A | B | ... | Z | a | b |...| z

4. GIC para algunas expresiones algebraicas.

   X $\rightarrow$ X + T | T

   T $\rightarrow$ T * F | F

   F $\rightarrow$ I | (X)

El lector puede comprobar fácilmente que, por ejemplo, la última gramática deriva la expresión $(a + b) * c$.

5.3 Notación BNF

La notación BNF se usa para especificar las reglas de sintaxis de un lenguaje de programación. Es una notación alternativa a la que se ha venido usando en el texto. Como se verá, esta notación es muy fácil de interpretar.

BNF son las iniciales de Backus Naur Form. John Backus y Peter Naur introdujeron esta notación para describir la sintaxis del lenguaje de programación ALGOL, alrededor de 1960.

Los metasímbolos de la notación se presentan en la siguiente tabla:

Metasímbolos de la notación BNF

::=	se define como
|	O
< >	Para delimitar categorías, es decir, no terminales
[ ]	Ítem opcional
" "	Para delimitar terminales de un solo carácter

La notación BNF es un lenguaje para especificar lenguajes de programación de computadores, de ahí que sus símbolos de denominen más precisamente metasímbolos.

Ejemplo

La siguiente gramática, en notación BNF, especifica la sintaxis de un lenguaje de programación hipotético, muy limitado por cierto. El lenguaje tiene palabras reservadas en español y utiliza como operador de asignación :=, como lo hace el lenguaje Pascal.

1. <Dígito> ::= "0" | "1" | "2" | "3" | … | "9"

Note la correspondencia con la notación que se ha venido usando en el texto: $D \rightarrow 0|1|2|3|\ldots|9$. Al no terminal le corresponde una categoría que va entre paréntesis angulares. Al signo *se define como* (::=) le corresponde el signo *se compone de* (→). Los terminales de

un solo carácter van entre comillas en la notación BNF. El símbolo *o* (|) es igual en ambas notaciones.

2. <Entero> ::=

   <Digito> <Entero> |

   <Dígito>

Note la correspondencia con la notación anterior:

$$E \rightarrow DE \mid D$$

3. <Signo> ::= “+” | “-“

4. <EnteroConSigno> ::=

   [<Signo>]<EnteroSinSigno>

La notación BNF no tiene un símbolo para la cadena vacía ε. En su lugar se usan los corchetes que indican que lo contenido en ellos es opcional, es decir, que no es obligatorio usarlo en la derivación. La anterior definición corresponde en la anterior notación a:

$$C \rightarrow SE$$

$$S \rightarrow +| - |\varepsilon$$

5. <Letra> ::=
   “A” | “B” | … | “Z” | “a” | “b” | … | “z”

6. <Identificador> ::=

   <Identificador><Letra>

   <Identificador><Digito> |

   <Letra>

7. <Sentencia> ::=

    <SentenciaAsignación> |

    <SentenciaSi> |

    <SentenciaMientras> |

    <SentenciaPara> |

    <LlamadaFunción> |

    <SentenciaAsignación> |

8. <ListaSentencias> ::=

    <ListaSentencias>

    <Sentencia> |

    <Sentencia>

9. <SentenciaMientras> ::=

    mientras <Condición> hacer

        <ListaSentencias>

    fin mientras

A partir de esta definición se puede derivar una sentencia como:

```
mientras b>a hacer
    a := a -1
    b := b +1
fin mientras
```

10.<SentenciaSi> ::=

si <Condición> entonces

<ListaSentencias>

[sino

<ListaSentencias>]

fin si

Observe que la parte 'de lo contrario' de la sentencia *si* es opcional, como es usual en muchos lenguajes de programación de computadores.

11.<SentenciaHacerMientras> ::=

hacer

<ListaSentencias>

mientras <Condición>

12.<SentenciaPara> ::=

para <Identificador>:= <Expresión>

hasta <Expresión> hacer

<ListaSentencias>

fin para

13.<Factor> ::=

"(" <Expresión> ")" |

<Identificador>

14.<Término> ::=

<Término> "*" <Factor> |

<Factor>

15.<Expresión> ::=

<Expresión> “+” <Término> |

<Término>

16.<SentenciaAsignación> ::=

<Identificador> := <Expresión>

El lector puede comprobar que a partir de esta definición se puede derivar una sentencia como:

a = (b+c)*d

17.<CondiciónSimple> ::=

[no] <CondiciónSimple> |

<Expresión> “<” <Expresión> |

“(“ <Condición> “)” |

verdadero |

falso

18.<Condición> ::=

<Condición> “y” <CondiciónSimple> |

<CondiciónSimple>

19.<ListaParámetros> ::=

<ListaParámetros> ”,” <Identificador> |

<Identificador>

20.<DefiniciónFunción> ::=

función <Identificador>

“(“ [<ListaParámetros>] “)”

<ListaSentencias>

fin función

21.<ListaExpresiones> ::=

<ListaExpresiones> “,” <Expresión> |

<Expresión>

22.<LlamadaFunción> ::=

<Identificador> “(“[<ListaExpresiones>]”)”

23.<ListaDefinicionesFunción> ::=

<ListaDefinicionesFunción>

<DefiniciónFunción> |

<DefiniciónFunción>

24.<Programa> ::=

<ListaSentencias>

[<ListaDefinicionesFuncion>]

Variaciones de BNF

Los libros realizan variaciones a la notación BNF clásica, para mejorar la legibilidad. Por ejemplo la BNF de un si…entonces podría presentarse de la siguiente forma.

SentenciaSi
si Condición **entonces**
ListaSentencias
[**sino**
ListaSentencias]
fin si

Note que los terminales se diferencian de los no terminales porque aparecen en negrilla.

Ejercicios

1) Los identificadores de un supuesto lenguaje de programación solo se componen de letras, por ejemplo los identificadores *SueldoA* o *notaB*. Escriba la gramática BFN para especificarlos.

2) Los identificadores de cierto lenguaje de programación deben iniciar con un signo pesos, $. Luego debe seguir una letra y después de esta letra pueden seguir cualquier número de letras o dígitos en cualquier orden. Por ejemplo el identificador *$nota13b*. Escriba la gramática BNF para especificarlos.

3) Los identificadores de cierto lenguaje de programación están especificados por la siguiente gramática.

   <Identificador> ::=
   $[<IdentificadorB>]$

   < IdentificadorB > ::=
   <IdentificadorB><Letra> | <Letra>

   a) De dos ejemplos de identificadores según la BNF anterior.
   b) ¿Cuál es el identificador de menor longitud de este lenguaje?

4) Elabore la definición BNF para una instrucción for como la de los siguientes ejemplos. La definición BNF debe permitir que el *for* repita cualquier cantidad de instrucciones, de cualquier tipo, cualquier cantidad de veces.

Ejemplo 1:

```
For  i = 0 To 10 Step 1
     k = k + i
     m = m * i
Next i
```

Ejemplo 2:

```
For j = a To b+1 Step 5+a
      k = k + j
      m = m * i
      c = k + m
Next i
```

Asuma ya definidas las siguientes categorías sintácticas: <Expresión>, <Lista de Sentencias>, <Identificador>.

5) Un lenguaje de programación tiene categorías sintácticas definidas así:[4]

<dígito> → 0 | 1 | 2 | ... | 9

<letra> → <minúscula> | <mayúscula>

<minúscula> → a | b | c | ...| z

<mayúscula> → A | B | C |... | Z

Si <vares> es una categoría sintáctica para describir palabras reservadas y se quiere que éstas se escriban en mayúsculas, una producción que cumple este propósito es

A) <vares> → <letra> | <vares> <mayúscula>

B) <vares> → <mayúscula> | <vares> <letra>

C) <vares> → <mayúscula> <minúscula> |
<vares> <mayúscula>

D) <vares> → <mayúscula> |
<vares> <mayúscula>

E) <vares> → <mayúscula> |
<vares> <minúscula>

[4] Ese ejemplo es tomado del examen ECAES del año 2003.

Ejercicios con respuestas

6)

<Identificador> ::= <Identificador> <Minúscula> |
<Mayúscula>
<Minúscula> ::= ”a”| ”b”| ”c”|…| ”z”
<Mayúscula> ::= ”A”| ”B”| ”C”|…| ”Z”

Según la anterior BNF, responda V o F a las siguientes afirmaciones

a. *Abcabccba* es un identificador válido _____

b. AB es un identificador válido _____

c. ABa es un identificador válido _____

d. A es un identificador válido _____

e. aA es un identificador válido _____

Respuesta: V F F V F

7) ¿Cuál de los identificadores se puede derivar la siguiente BNF?

<Identificador> ::= <Identificador> <Dígito> |
“$”<Mayúscula><Minúscula>

<Dígito> ::= “1” | “2” | … | “9”
<Minúscula> ::= ”a”| ”b”| ”c”|…| ”z”
<Mayúscula> ::= ”A”| ”B”| ”C”|…| ”Z”

A. $Ab314
B. A210
C. Ab1
D. $Abc
E. $AbA1

Respuesta: A

8) De un ejemplo de una sentencia de asignación de acuerdo con la siguiente gramática.

<Variable> ::=
a01 | b02 | c03
<Oper. de Asignación > ::=
<<< | <--
<Operador Aritmético> ::=
"+" | "-" | "*" | "/"
<Función> ::=
raiz | abs | exp
<Expresión > ::=
("<Variable>")"<Funcion>"
("<Variable>")"[<Funcion>]<Operador Aritmético>
<Sentencia de Asignación> ::=
<Variable> <Operador de Asignación> <Expresión>

Respuesta: a01<<< (b02) raíz(c03) +

9) De un ejemplo de una sentencia de asignación de acuerdo con la siguiente gramática.

<Variable> ::=
p01 | q02 | r03
<Oper. de Asignación > ::=
:= | "#"
<Operador Aritmético> ::=
"+" | "-" | "*" | "/"
<Función> ::=
cuadrado | doble | cubo
<Expresión > ::=
<Operador Aritmético> <Funcion> "("<Variable>")"
[<Funcion>] " ("<Variable>")"

<Sentencia de Asignación> ::=
<Variable> <Operador de Asignación> <Expresión>

Respuesta: p01:= + cubo (q02) (r03)

10) ¿Cuál de las sentencias se puede derivar de la siguiente BNF?

<SentenciaEjecute> ::= ejecute <identif> <entero> veces
<Identif> ::= “a” | “b” | “c”
<Entero> ::= 1 | 2 | 3

A. SentenciaEjecute b 3 veces
B. ejecute abc 123 veces
C. ejecute b 3 veces
D. ejecute bbb 3 veces
E. SentenciaEjecute ::= ejecute b 3 veces

Respuesta: C

11) ¿Cuál de las sentencias se puede derivar de la siguiente BNF?

<SentenciaSi> ::= si <Condición> vaya a <Etiqueta>
<Condición> ::= <Identif> <OperadorRelacional><Identif>
<Etiqueta> ::= “$”<Identif>
<Identif> ::= “a” | “b” | “c”
<operadorRelacional> ::= _MQ_ | _DF_

A. si a _DF_ b vaya a $c
B. si a5 _DF_ b vaya a $c
C. si 6 _MQ_ 7 vaya a $c
D. si a _DF_ b vaya a c
E. si a > b vaya a $c

Respuesta: A

12) Se tiene definida para un lenguaje la siguiente categoría sintáctica.

<minúscula> ::= ”a”| ”b”| ”c”|…| ”z”

Las variables del lenguaje se conforman de la letra v, seguida de un signo de subrayado, seguido de una o más letras minúsculas. ¿Cuál de las siguientes BNF define bien las variables de este lenguaje?

A. <variable> ::= <variable> <minúscula> | v
B. <variable> ::= <variable> <minúscula> | v_<minúscula>
C. <variable> ::= <minúscula> | v_<minúscula>
D. <variable> ::= <variable> | v_<minúscula>
E. <variable> ::= <variable> <minúscula> | v_

Respuesta: B

13) Un supuesto lenguaje de programación tiene un ciclo denominado ciclo perform. El ciclo permite repetir 0 o más sentencias un número determinado de veces, dado por una expresión. Los siguientes son ejemplos de este ciclo.

Ejemplo 1:

```
perform
        begin
                imprimir x
                y = y * 2
        end
25 times
```

Ejemplo 2:

```
perform
        begin
                a = b + a
                imprimir x
                y = y * 2
                imprimir y
```

Escriba la definición BNF para el ciclo perform descrito previamente. Suponga ya definidas previamente las categorías <Expresión> y <Lista de Sentencias>.

Respuesta:

```
<CicloPerform>::=
        perform
                begin
                        <ListaDeSentencias>
                end
        <Expresión> times
```

5.4 Gramáticas ambiguas y no ambiguas

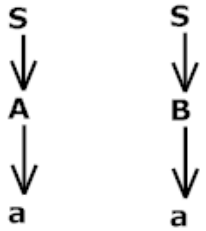
Las gramáticas se pueden clasificar en ambiguas y en no ambiguas. Una gramática es ambigua si y solo si existen dos o más árboles distintos que derivan la misma palabra [KELLY].

Las gramáticas ambiguas no son apropiadas para ser el modelo para la construcción de compiladores. Por ejemplo, no manejan bien la precedencia de los operadores.

La siguiente GIC es ambigua

S → A | B
A → aA | a
B → bB | a

Los árboles diferentes que derivan la misma palabra son:



Ejercicios

Demostrar que son ambiguas las siguientes gramáticas.

1) S → A | B
   A → aA | b
   B → Bb | a

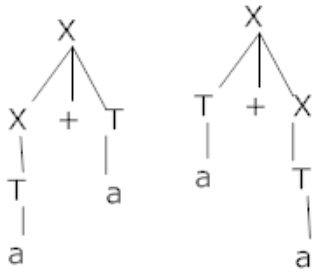
2) S → A | B
   A → aaA | aa
   B → aaaB | aaa

3) E → E + E | E * E | I
   I → a | b | c

Ejercicio con respuesta

4) Demuestre que la siguiente gramática es ambigua
   $X \to X+T \mid T+X \mid T$
   $T \to a \mid b$

Respuesta:



5.5 Recursividad por la izquierda

Otra condición que no es recomendable, si vamos a utilizar una gramática como base de un compilador, es la recursividad por la izquierda.

Una producción de la forma A → Aw, donde A es un no terminal y w es una palabra, se dice que es recursiva por la izquierda. Por ejemplo, la siguiente GIC tiene tres producciones recursivas por la izquierda.

S → Sa | AB

A → AbbS | Ab | ε

B → c

La recursividad por la izquierda es una característica no deseable de las producciones, porque dificulta la construcción del correspondiente compilador.

Podemos eliminar la recursividad por la izquierda con el siguiente procedimiento.

[KELLY] Sea *G* una GIC y sea *A* un no terminal de *G* que tiene producciones recursivas por la izquierda. Sean las siguientes todas las producciones de A.

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$$

Las primeras *n* producciones son todas las producciones recursivas por la izquierda de A. Las restantes son todas las producciones de *A* que no son recursivas por la izquierda.

Podemos reemplazar las anteriores producciones por las siguientes que no son recursivas por la izquierda.

$$A \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m \mid \beta_1 Z \mid \beta_2 Z \mid \ldots \mid \beta_m Z$$

$$Z \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n \mid \alpha_1 Z \mid \alpha_2 Z \mid \ldots \mid \alpha_n Z$$

Como se puede observar, se ha introducido un no terminal nuevo, *Z*.

Ninguna de las nuevas producciones es recursiva por la izquierda.

Como ejemplo, apliquemos las fórmulas a la gramática presentada al inicio de esta sección:

S → Sa | AB

A → AbbS | Ab | ε

B → c

De acuerdo con las fórmulas, la gramática equivalente y sin producciones recursivas por la izquierda queda:

$S \rightarrow AB|ABZ$

$Z \rightarrow a|aZ$

$A \rightarrow \varepsilon|Z'$

$Z' \rightarrow bbS|b|bbsZ'|bZ'$

$B \rightarrow c$

Ejercicios con respuestas

1) Aplique el procedimiento de eliminar recursividad por la izquierda a la siguiente GIC.

S → Sa | Sbcd | ab | c

¿Cuál es la GIC resultante?

A. S→ ab | c | abZ | cZ
   Z→ a | bcd | aZ | bcdZ

B. S→ a | bcd | aZ | bcdZ
   Z→ ab | c | abZ | cZ

C. S→ ab | c | Z
   Z→ a | bcd | aZ | bcdZ

D. S→ ab | c | abZ | cZ
   Z→ Sa | Sbcd | SaZ | SbcdZ

E. S→ a | bcd | ab | c

Respuesta: A

2) Aplique el procedimiento de eliminar la recursividad por la izquierda a la siguiente gramática.

$$X \rightarrow X++ \mid X-- \mid X+a \mid -X \mid b$$

Respuesta:

$$X \rightarrow -X \mid b \mid -XZ \mid bZ$$
$$Z \rightarrow ++ \mid -- \mid +a \mid ++Z \mid --Z \mid aZ$$

3) Aplique el procedimiento de eliminar la recursividad por la izquierda a la siguiente gramática.

E → E * a | b

¿Cuál es la gramática resultante?

Respuesta: E → b | bZ

Z→ *a | *aZ

Ejercicios

Obtenga GIC no recursivas por la izquierda equivalentes a las siguientes, utilizando el procedimiento descrito en esta sección.

4) S → Sa | Sb | cA

   A → Aa | a | ε

5) S → SaAb | Sb | aa | b

   A → Abbb | Ab | Aa | a | b | ε

6) E → E + E | (E) | V

   V → a | b

5.6 Factorización a izquierdas

A veces sucede que en dos producciones de una gramática para el mismo no terminal, $A \rightarrow w|x$, se tiene que las palabras w y x inician con los mismos símbolos. Por ejemplo, las siguientes producciones tienen esa característica:

$$A \rightarrow Bbca|BbcBC$$

Esta característica podría hacer más compleja la elaboración de un futuro compilador basado en esta gramática, específicamente en lo relacionado con la selección por parte del compilador de que producción aplicar.

Por fortuna es fácil transformar las producciones en otras equivalentes sin esta condición. Por ejemplo, las producciones del

ejemplo anterior son equivalentes a las que se presentan a continuación:

$$A \rightarrow BbcD$$

$$D \rightarrow a|BC$$

[LEMONE] En general, dos producciones de la forma $A \rightarrow \beta\alpha_1|\beta\alpha_2$ son equivalentes a las producciones:

$$A \rightarrow \beta B$$

$$B \rightarrow \alpha_1|\alpha_2$$

Ejercicios con respuestas

1) Aplique el procedimiento de factorización a izquierdas a la siguiente gramática

$$S \rightarrow abSA|abSBba$$
$$A \rightarrow cccSa|cccBbb$$
$$B \rightarrow a|\varepsilon$$

Respuesta:

$$S \rightarrow abSD$$
$$D \rightarrow A|Bba$$
$$A \rightarrow cccE$$
$$E \rightarrow Sa|Bbb$$
$$B \rightarrow a|\varepsilon$$

2) Aplique el procedimiento de factorización a izquierdas a la siguiente gramática

<SentenciaRepetir> ::=

Repita <Entero> veces

<ListaSentencias> |

Repita mientras <Condición>

<ListaSentencias>

Respuesta:

<SentenciaRepetir> ::=

Repita <SegundaParteSentenciaRepetir>

<SegundaParteSentenciaRepetir> ::=

<Entero> veces

<ListaSentencias> |

mientras <Condición>

<ListaSentencias>

5.7 Gramática regular

Hay dos tipos de gramáticas regulares: gramáticas regulares por la izquierda y gramáticas regulares por la derecha.

Definición

[KELLY] Una gramática independiente del contexto es una gramática regular por la izquierda si y solo si todas sus producciones son de la forma $A \rightarrow w$ donde se cumple lo siguiente:

- *w* tiene un no terminal como máximo.
- si *w* tiene un no terminal este será el símbolo de la izquierda de *w*.

Sigue un ejemplo:

S → Sabab | A | aab
A → Aaa | Ab | b | ε

Definición

[KELLY] Una gramática independiente del contexto es una gramática regular por la derecha si y solo si todas sus producciones son de la forma *A* → *w* donde se cumple lo siguiente:

- *w* tiene un no terminal como máximo.
- si *w* tiene un no terminal este será el símbolo de la derecha de *w*.

Sigue un ejemplo:

S → ababS | A | bba
A → aaA | bA | ε

Poder de representación

Las gramáticas regulares tienen un poder de representación igual al de los autómatas finitos y al de las expresiones regulares. Es decir, pueden representar lenguajes regulares únicamente. En el campo de los compiladores, las gramáticas regulares pueden representar los aspectos léxicos de un lenguaje de programación de computadores, pero no los aspectos gramaticales. Es decir, pueden representar tokens, más no expresiones algebraicas ni instrucciones

Ejercicios

Elabore gramáticas regulares por la izquierda y por la derecha para los siguientes lenguajes

1) a^*

2) $(ab)^* b^*$

3) $a(a \cup b)^*$

4) $ab^*(ab)^*$

5.8 Forma normal de Chomsky

Definición

[KELLY] Una GIC se encuentra en forma normal de Chomsky, si y solo si, todas sus producciones *A* → *w* cumplen una de las dos siguientes condiciones.

- *w* se conforma exactamente de dos no terminales.
- *w* se conforma exactamente de un terminal.

Sigue un ejemplo de GIC en forma normal de Chomsky:

S → AA | AB
A → a
B → AS | b

Los árboles de derivación de una gramática en forma normal de Chomsky son binarios.

Ejemplo:

Elabore una GIC en forma normal de Chomsky para el lenguaje $\{a^{2i+1} | i \geq 0\}$

Solución:

Elaboremos como punto de partida un GIC común y corriente:

$$S \rightarrow aSa|a$$

Reemplacemos los terminales de las producciones que no están en forma normal de Chomsky, por no terminales. En este caso, reemplazamos las aes de la primera producción, por *A*. La gramática queda:

$$S \rightarrow ASA|a$$

$$A \rightarrow a$$

En las producciones que tengan más de 2 no terminales, reemplazamos grupos de 2 no terminales por un no terminal nuevo. En este caso, reemplazamos por ejemplo AS por B y la gramática ya queda en forma normal de Chomsky:

$$S \rightarrow BA|a$$

$$B \rightarrow AS$$

$$A \rightarrow a$$

Ejercicio

1) Elabore GIC en forma normal de Chomsky para el lenguaje $\{(ab)^i a^i \mid i \geq 1\}$

5.9 Simplificación de gramáticas

En esta sección se tratarán procedimientos para eliminar producciones inútiles de las gramáticas.

5.9.1 No terminales que no derivan cadenas de solo terminales

Consideremos la siguiente gramática

S → aS | aA | b
A → aA | bA

Se observa que no se puede derivar una cadena de solo terminales a partir de A.

$A \Rightarrow aA \Rightarrow abA \ldots$

Lo anterior ocasiona que tres producciones de la gramática no aporten nada. Estas producciones se pueden omitir. La gramática simplificada queda:

S → aS | b

El siguiente algoritmo permite eliminar este tipo de producciones.

Algoritmo

i. Seleccione las producciones x → w, tales que $w \in \Sigma^*$.

ii. Inicialice el conjunto N' con los no terminales de la izquierda del signo → de las producciones seleccionadas en el paso i.

iii. Seleccione las producciones x → w no seleccionadas anteriormente, tales que $w \in (N' \cup \Sigma)^*$

iv. Agregue al conjunto N' los no terminales de la izquierda del signo → de las producciones seleccionadas en el paso iii.

v. Repita los pasos iii y iv hasta que no se pueden seleccionar nuevas producciones o agregar nuevos no terminales a N'.

vi. Escriba la gramática simplificada, la cual consta de las producciones seleccionadas en los pasos i y iii.

Ejemplo

Aplique el algoritmo que elimina los no terminales que no derivan cadenas de solo terminales a la siguiente gramática.

$S \rightarrow BS|aC|A$
$A \rightarrow$ abE
$B \rightarrow$ DaDC | aCB | bCF
$C \rightarrow$ DDb | abG
$D \rightarrow aba$
$E \rightarrow$ aA
$F \rightarrow \varepsilon$

Solución:

i) $D \rightarrow aba$

$F \rightarrow \varepsilon$

ii) $N' = \{D, F\}$

iii) $C \rightarrow$ DDb

iv) $N' = \{D, F, C\}$

iii) $S \rightarrow aC$

$B \rightarrow$ DaDC

$B \rightarrow$ bCF

iv) $N' = \{D, F, C, S, B\}$

iii) $S \rightarrow BS$

$B \rightarrow$ aCB

iv) N'no cambia. Esta es la condición de parada del algoritmo.

vi. La gramática simplificada se conforma de las producciones seleccionadas:

$S \rightarrow BS|aC$

$B \rightarrow \text{DaDC} \mid \text{aCB} \mid \text{bCF}$

$C \rightarrow \text{DDb}$

$D \rightarrow aba$

$F \rightarrow \varepsilon$

5.9.2 Producciones imposibles de usar

Consideremos la siguiente gramática

```
S → aSa | b
B → Ca | a
C → Ba | a
```

Se observa que es imposible utilizar las 4 últimas producciones en una derivación. Estas producciones pueden omitirse. La gramática simplificada queda:

```
S → aSa | b
```

El siguiente algoritmo elimina este tipo de producciones.

Algoritmo

i. Inicialice el conjunto N' = {S}
ii. Selecciones las producciones x → w tales que $x \in N'$
iii. Agregue a N' los no terminales a la derecha del signo → de las producciones seleccionadas anteriormente y que no se encuentren previamente en N'.
iv. Selecciones las producciones x → w tales que $x \in N'$ y que no hayan sido seleccionadas previamente.

v. Repita los pasos iii y iv hasta que en el paso iii no se agreguen nuevos no terminales o en el paso iv no se selecciones nuevas producciones.

vi. Escriba la gramática simplificada. Esta gramática de conforma de todas las producciones seleccionadas en los anteriores pasos.

Ejemplo

Aplique el algoritmo de suprimir producciones que no se pueden usar a la siguiente gramática.

$S \rightarrow \text{aE} \mid \text{bF}$

$A \rightarrow \text{aA} \mid \text{C}$

$B \rightarrow \text{aB} \mid \text{a}$

$C \rightarrow \text{aD} \mid \text{b}$

$D \rightarrow \text{aA} \mid \text{aC}$

$E \rightarrow \text{aE} \mid \text{bE}$

$F \rightarrow \text{FB} \mid \text{b}$

Solución:

i. $N' = \{S\}$

ii. $S \rightarrow \text{aE} \mid \text{bF}$

iii. $N' = \{S, E, F\}$

iv. $E \rightarrow \text{aE} \mid \text{bE}$

$F \rightarrow \text{FB} \mid \text{b}$

iii. $N' = \{S, E, F, B\}$

iv. $B \rightarrow \text{aB} \mid \text{a}$

iii. N' no cambia. Esta es la condición de parada del algoritmo.

vi. La gramática simplificada queda:

$S \rightarrow$ aE | bF

$B \rightarrow$ aB | a

$E \rightarrow$ aE | bE

$F \rightarrow$ FB | b

5.9.3 No terminales anulables

Antes de pasar al algoritmo de eliminación de producciones épsilon, es necesario estudiar el algoritmo que detecta los no terminales anulables.

Sea A un no terminal, A es anulable, si y solo si, A deriva ε en uno o más pasos.

Por ejemplo, en la gramática

S → ABa | b
A → ε
B → A

A y B son anulables, ya que

A $\Rightarrow \varepsilon$
B $\Rightarrow$ A $\Rightarrow$ ε

S no es un no terminal anulable.

El siguiente algoritmo determina los no terminales anulables.

Algoritmo

i. Se seleccionan las producciones de la forma x → ε
ii. Se inicializa el conjunto N con los no terminales de la izquierda del signo → de las producciones seleccionadas en el paso i.
iii. Se seleccionan la producciones x → w no seleccionadas antes tales que w ∈ N^*

iv. Se agregan al conjunto N los no terminales no agregados antes que estén a la izquierda del signo → de las producciones seleccionadas en el paso 3.

v. Se repiten los pasos iii y iv, hasta que en el paso iii no se seleccionen nuevas producciones o, en el paso iv no se agreguen nuevos no terminales.

En el conjunto N se encuentran todos los no terminales anulables.

Ejemplo

Aplique el algoritmo para encontrar los no terminales anulables a la siguiente gramática.

$S \rightarrow \text{aS} \mid \text{FEB}$

$A \rightarrow \text{aA} \mid \text{ab} \mid \text{C}$

$B \rightarrow \text{bB} \mid \varepsilon$

$C \rightarrow \text{aC} \mid \text{b}$

$D \rightarrow \text{A} \mid \text{SSB}$

$E \rightarrow \text{BF}$

$F \rightarrow \varepsilon$

Solución:

i. $B \rightarrow \varepsilon$

$F \rightarrow \varepsilon$

ii. $\text{N} = \{B, F\}$

iii. $E \rightarrow \text{BF}$

iv. $\text{N} = \{B, F, E\}$

iii. $S \rightarrow \text{FEB}$

iv. $\text{N} = \{B, F, E, S\}$

iii. $D \rightarrow \text{SSB}$

iv. $\text{N} = \{B, F, E, S, D\}$

iii. No se seleccionan nuevas producciones. Esta es la condición de parada del algoritmo.

El conjunto de los no terminales anulables de la gramática dada es $\text{N} = \{B, F, E, S, D\}$

5.9.4 Eliminación de producciones épsilon, excepto $S \rightarrow \varepsilon$

La gramática

S → aS | aA
A → b | ε

es equivalente a la gramática

S → aS | ab | a

Se observa que la segunda gramática no tiene producciones épsilon.

La gramática

S → aS | A
A → b | ε

es equivalente a la gramática

S → aS | b | ε

Vemos que la segunda gramática no tiene producciones épsilon, excepto S → ε , la cual no se puede suprimir.

El siguiente algoritmo elimina las producciones épsilon, excepto $S \rightarrow \varepsilon$.

Algoritmo

a) Halle el conjunto de los no terminales anulables, aplicando el algoritmo de la sección anterior.
b) Suprima las producciones épsilon de la gramática
c) Agregue todas las producciones que resulten de eliminar cualquier combinación de no terminales anulables a las producciones iniciales y que no sean producciones épsilon.
d) Agregar $s \to \varepsilon$, si la gramática inicial deriva ε

Ejemplo

Aplique el algoritmo de eliminación de producciones épsilon a la siguiente gramática

$S \to \text{aS} \mid \text{bABE} \mid \text{D}$

$A \to \text{BEDa}$

$B \to \text{ED}$

$E \to \text{b} \mid \varepsilon$

$D \to \text{a} \mid \varepsilon$

Solución:

a) Hallamos el conjunto de no terminales anulables

i. $E \to \varepsilon$

$D \to \varepsilon$

ii. $\text{N} = \{E, D\}$

iii. $S \to \text{D}$

$B \to \text{ED}$

iv. $\text{N} = \{E, D, S, B\}$

iii. <Fin>

El conjunto de no terminales anulables de la gramática es $\mathrm{N} = \{E, D, S, B\}$

b) Se suprimen las producciones épsilon

$S \rightarrow$ aS | bABE | D

$A \rightarrow$ BEDa

$B \rightarrow$ ED

$E \rightarrow$ b

$D \rightarrow$ a

c) Se agregan nuevas producciones según lo indicado por el algoritmo:

$S \rightarrow$ aS | bABE |D| a |bAE | bAB |bA

$A \rightarrow$ BEDa | EDa |BDa |BEa |Da | Ea |Ba | a

$B \rightarrow$ ED | D |E

$E \rightarrow$ b

$D \rightarrow$ a

c) Se agrega $S \rightarrow \varepsilon$, debido a que la gramática deriva ε:

$S \rightarrow$ aS | bABE |D| a |bAE | bAB |bA | ε

$A \rightarrow$ BEDa | EDa |BDa |BEa |Da | Ea |Ba | a

$B \rightarrow$ ED | D |E

$E \rightarrow$ b

$D \rightarrow$ a

5.10 Autómatas de pila no deterministas

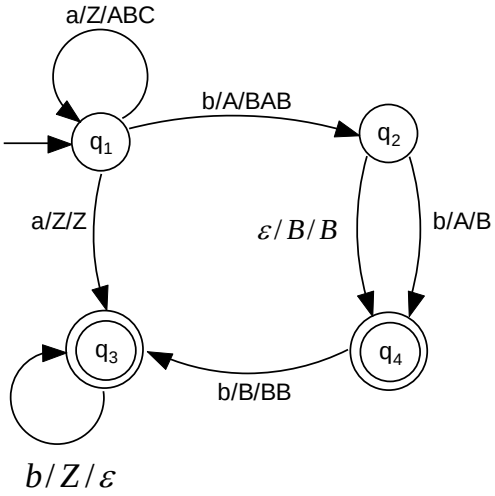
Los ADPND tienen el mismo poder de representación de las GIC. Es decir pueden representar a los lenguajes independientes de contexto

y por lo tanto también a los lenguajes regulares. En el contexto de los compiladores, los ADPND pueden representar tantos tokens, como expresiones algebraicas e instrucciones

Los autómatas de pila no deterministas, ADPND, son autómatas finitos no deterministas a los cuales se les ha agregado una estructura de datos pila.

La pila tiene un símbolo inicial, que en los ejemplos representaremos por Z.

Ejemplo del diagrama de transición de un ADPND



Cada transición tiene la forma general $\sigma/A/w$.

σ puede ser un símbolo del alfabeto o la palabra vacía. Si es un símbolo del alfabeto, el ADPND al usar la transición consume este símbolo de la palabra que se está analizando. Si σ es la palabra vacía, el ADPND al usar la transición no consume ningún símbolo.

A es un símbolo del alfabeto. Al usar la transición, el autómata extrae este símbolo de la pila. Por tanto, la pila debe tener en su tope el símbolo *A*, o no podrá ser utilizada la transición.

w es una palabra. Si *w* es una palabra de uno o más símbolos, el autómata al usar la transición introduce estos símbolos en la pila, en orden de derecha a izquierda. Si *w* es la palabra vacía, el autómata al usar la transición no introduce símbolos en la pila.

Un ADPND acepta una palabra cuando existe al menos una ruta que tenga las siguientes características.

- La ruta debe partir del estado inicial
- La ruta debe consumir exactamente los símbolos de la palabra, en el orden en el que aparecen en ella.
- La ruta debe terminar en un estado final.
- Debe ser posible que el ADPND realice todas las operaciones sobre la pila indicada por la ruta. No interesa si al final la pila queda vacía o no.

Se aclara que no está unificado el concepto de que si la pila debe quedar vacía al final o no, para que la palabra sea aceptada. Según Dean Kelly no se requiere que la pila quede vacía [KELLY]. Según Ramón Brena, la pila tiene que quedar vacía al final[BRENA].

Ejemplo

¿Cuáles de las siguientes palabras acepta el ADPND de la figura?.

a) abb
b) bb

Solución.

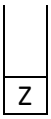
a) La ruta $q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_4 \rightarrow q_3$, usando la transición $\varepsilon/B/B$ para ir de q_2 a q_4, cumple con todas las condiciones requeridas, por lo que la palabra *abb* es aceptada por el autómata.

La ruta cumple las condiciones requeridas ya que:

- Parte del estado inicial

- Consume exactamente los símbolos *abb* y en este orden
- La ruta termina en un estado de aceptación, q_3
- Se pueden realizar todas las operaciones sobre la pila indicadas por las 4 transiciones utilizadas, como se muestra a continuación.

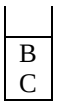
Inicialmente la pila tiene una Z:



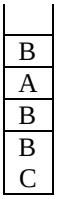
Debido a la transición $a/Z/ABC$, el autómata retira la *Z* de la pila

Debido a la misma transición, el autómata introduce los símbolos de la palabra *ABC* de derecha a izquierda en la pila:

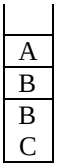
Ahora el autómata retira *A* en la pila, ya que aplica la transición $b/A/BAB$:



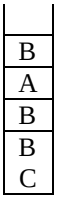
Según indica la misma transición, el ADPND agrega a la pila los símbolos B, A, B



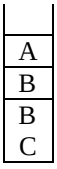
Seguidamente se utiliza la transición $\varepsilon/B/B$, por lo que se retira *B* del tope de la pila.



La anterior transición ocasiona que en este momento se agregue *B* a la pila.



La última transición de la ruta es $b/B/BB$, por lo el autómata retira B del tope de la pila.



Finalmente, la última transición también ocasiona que se agregue *BB* al tope de la pila.

Así, fue posible realizar en la pila todas las operaciones establecidas por las 4 transiciones de la ruta y se cumplió la última condición de la ruta. Como se explicó, no es necesario que la pila quede vacía al final.

b) *bb*

La palabra *bb* no es aceptada por el ADPND del ejemplo, ya que no existe ninguna ruta que cumpla con los requisitos.

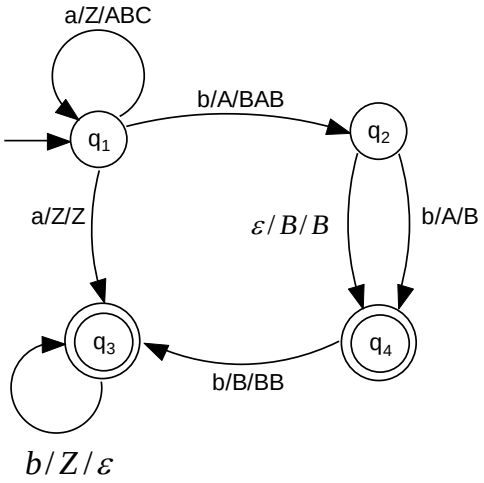
Por ejemplo la ruta $q_1 \rightarrow q_2 \rightarrow q_4$, utilizando la transición $b/A/B$ par ir de q_2 a q_4, falla porque no se puede retirar A del tope de la pila, en el momento de aplicar la primera transición $b/A/BAB$.

Regla de transición

Las transiciones de un ADPND pueden representarse analíticamente utilizando la regla de transición Δ. Mediante esta regla, una transición $\sigma/A/w$ que vaya del estado q_i al estado q_j, se representa por:

$$\Delta(q_i, \sigma, A) = \{(q_j . w)\}$$

Como ejemplo, se incluye el diagrama de transición del autómata anterior y la correspondiente representación analítica.



$\Delta(q_1, a, Z) = \{(q_1, ABC), (q_3, Z)\}$

$\Delta(q_1, b, A) = \{(q_2, BAB)\}$

$\Delta(q_2, \varepsilon, B) = \{(q_4, B)\}$

$\Delta(q_2, b, A) = \{(q_4, B)\}$

$\Delta(q_4, b, B) = \{(q_3, BB)\}$

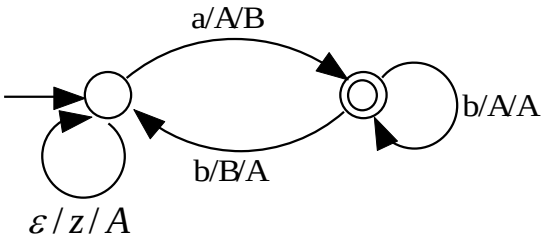
$\Delta(q_3, b, Z) = \{(q_3, \varepsilon)\}$

El conjunto de los estados de aceptación se representa por $F = \{q_3, q_4\}$

Note que por brevedad se agruparon dos transiciones en un solo reglón. Se puede realizar esta agrupación cuando la parte de la izquierda de 2 o más transiciones es igual en todas ellas.

Ejercicio con respuesta

1) ¿Cuál de las palabras es aceptada por el siguiente el ADPND?

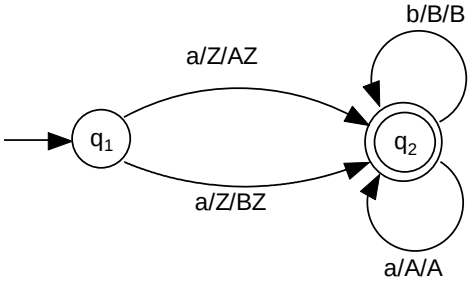


A. ababa
B. ab
C. abab
D. abb
E. aa

Respuesta: A

Ejercicios

2) En el siguiente ADPND, ¿Cuáles de las palabras dadas acepta el ADPND?



a) ba
b) aba

c) a

d) abb

Lenguaje aceptado por un ADPND

Sea *M* un ADPND, se define el lenguaje aceptado por el autómata, L(*M*), como el lenguaje de todas las palabras que acepta M. Por ejemplo el lenguaje aceptado por el autómata siguiente es $\{a^i b^j | i > j, j > 0\}$.

$\Delta(q_1, a, z) = \{(q_1, bz)\}$

$\Delta(q_1, a, b) = \{(q_1, bb)\}$

$\Delta(q_1, b, b) = \{(q_2, \varepsilon)\}$

$\Delta(q_2, b, b) = \{(q_2, \varepsilon)\}$

$\Delta(q_2, \varepsilon, b) = \{(q_3, b)\}$

$F=\{q_3\}$

Ejercicios

¿Cuál es el lenguaje aceptado por los siguientes ADPND?

3) $\Delta(q_1, a, Z) = \{(q_1, AZ)\}$
$\Delta(q_1, b, A) = \{(q_2, B)\}$
$\Delta(q_2, b, B) = \{(q_2, B)\}$
$F = \{q_2\}$

4) $\Delta(q_1, \varepsilon, Z) = \{(q_1, A)\}$
$\Delta(q_1, a, A) = \{(q_2, B)\}$
$\Delta(q_2, b, B) = \{(q_2, D)\}$
$\Delta(q_2, a, D) = \{(q_1, A)\}$
$F = \{q_1\}$

5) $\Delta(q_1, a, Z) = \{(q_1, A)\}$

$\Delta(q_1, a, A) = \{(q_1, AA)\}$

$\Delta(q_1, \varepsilon, A) = \{(q_2, A)\}$

$\Delta(q_2, b, A) = \{(q_2, \varepsilon)\}$

$F = \{q_2\}$

Ejercicios con respuesta

6) ¿Cuál es el lenguaje aceptado por el siguiente ADPND?

$\Delta(q_1, a, Z) = \{(q_1, A)\}$

$\Delta(q_1, b, A) = \{(q_1, B)\}$

$\Delta(q_1, c, B) = \{(q_1, C)\}$

$\Delta(q_1, d, C) = \{(q_2, C)\}$

$\Delta(q_2, d, C) = \{(q_2, C)\}$

$F = \{q_2\}$

Respuesta: $abcdd^*$

7) ¿Cuál es el lenguaje aceptado por el siguiente ADPND?

$\Delta(q_1, a, z) = \{(q_1, bz)\}$

$\Delta(q_1, a, b) = \{(q_1, bb)\}$

$\Delta(q_1, b, b) = \{(q2, \varepsilon)\}$

$\Delta(q_2, b, b) = \{(q_2, \varepsilon)\}$

$\Delta(q_2, \varepsilon, b) = \{(q_3, \varepsilon)\}$

$\Delta(q_3, \varepsilon, z) = \{(q_4, z)\}$

$F=\{q_4\}$

Respuesta

$\{ a^{i+1}b^i \mid i \geq 1 \}$

Ejercicio

Obtenga ADPND para los siguientes lenguajes

8) $\{a^i b^i \mid i \geq 0\}$

9) $\{a^i b^{2i} \mid i \geq 0\}$

10) $\{a^i b^j c^{i+j} \mid i \geq 1,\ j \geq 1\}$

Ejercicios con respuestas

11) Obtenga un ADPND para los siguientes lenguajes:

a) $\{ a^m b^n \mid m \geq 0, n \geq 0 \text{ y } m \neq n \}$

b) $\{ ww^{|} \mid w \in \{a,b\}^+ \}$

Respuestas

a) $\Delta(q_1, a, z) = \{ (q_1, bz) \}$

$\Delta(q_1, a, b) = \{ (q_1, bb) \}$

$\Delta(q_1, b, b) = \{ (q_2, \varepsilon) \}$

$\Delta(q_2, b, b) = \{ (q_2, \varepsilon) \}$

$\Delta(q_2, \varepsilon, b) = \{ (q_3, \varepsilon) \}$

$\Delta(q_2, b, z) = \{ (q_2, az) \}$

$\Delta(q_2, b, a) = \{ (q_2, aa) \}$

$\Delta(q_1, \varepsilon, b) = \{ (q_3, b) \}$

$\Delta(q_1, b, z) = \{ (q_2, az) \}$

$\Delta(q_2, \varepsilon, a) = \{(q_3, a)\}$

F={q3}

b) $\Delta(q_1, a, z) = \{(q_1, az)\}$

$\Delta(q_1, a, a) = \{(q_1, aa), (q2, \varepsilon)\}$

$\Delta(q_1, b, a) = \{(q_1, ba)\}$

$\Delta(q_1, b, b) = \{(q_1, bb), (q_2, \varepsilon\}$

$\Delta(q_2, a, a) = \{(q_2, \varepsilon)\}$

$\Delta(q_2, \varepsilon, z) = \{(q_3, z)\}$

$\Delta(q_1, b, z) = \{(q_1, bz)\}$

$\Delta(q_1, a, b) = \{(q_1, ab)\}$

$\Delta(q_2, b, b) = \{(q_2, \varepsilon)\}$

F={q3}

Método Construcción de un ADPND a partir de una GIC

Con el método que se explicará a continuación, se puede construir mecánicamente un ADPND partiendo de una GIC.

El autómata que resulta de este método siempre queda con tres estados: q_1, q_2 y q_3, sin importar de que lenguaje se trate. De los tres estados, solo q_3 es estado final.

Para construir el ADPND aplique los siguientes pasos.

i. Cree una GIC para el lenguaje cuyo símbolo inicial sea *S*.

ii. Agregue las siguientes dos transiciones.

$\Delta(q_1, \varepsilon, Z) = \{(q_2, SZ)\}$

$\Delta(q_2, \varepsilon, Z) = \{(q_3, \varepsilon)\}$

iii. Por cada producción x → w agregue la transición:

$$\Delta(q_2, \varepsilon, x) = \{(q_2, w)\}$$

iv. Por cada terminal *a* agregue la transición:

$$\Delta(q_2, \mathrm{a}, \mathrm{a}) = \{(q_2, \varepsilon)\}$$

Ejemplo

Obtenga una GIC para el lenguaje dado y, a partir de ella, construya un ADPND, aplicando el método explicado en la presente sección.

$$\{a^i b a^i b^j a^j \mid i \geq 0,\ j \geq 0\}$$

Solución:

i. Primero se crea la GIC para el lenguaje dado

$$S \rightarrow AB$$

$$A \rightarrow aAa|b$$

$$B \rightarrow bBa|\ \varepsilon$$

ii. Agregamos las dos primeras transiciones

$$\Delta(q_1, \varepsilon, Z) = \{(q_2, SZ)\}$$

$$\Delta(q_2, \varepsilon, Z) = \{(q_3, \varepsilon)\}$$

iii. Agregamos las transiciones correspondientes a cada producción

$$\Delta(q_1, \varepsilon, Z) = \{(q_2, SZ)\}$$

$$\Delta(q_2, \varepsilon, Z) = \{(q_3, \varepsilon)\}$$

$$\Delta(q_2, \varepsilon, S) = \{(q_2, AB)\}$$

$$\Delta(q_2, \varepsilon, A) = \{(q_2, aAa), (q_2, b)\}$$

$$\Delta(q_2, \varepsilon, B) = \{(q_2, bBa), (q_2, \varepsilon)\}$$

v. Agregamos las transiciones correspondientes a cada símbolo del alfabeto.

$$\Delta(q_1, \varepsilon, Z) = \{(q_2, SZ)\}$$

$$\Delta(q_2, \varepsilon, Z) = \{(q_3, \varepsilon)\}$$

$$\Delta(q_2, \varepsilon, S) = \{(q_2, AB)\}$$

$$\Delta(q_2, \varepsilon, A) = \{(q_2, aAa), (q_2, b)\}$$

$$\Delta(q_2, \varepsilon, B) = \{(q_2, bBa), (q_2, \varepsilon)\}$$

$$\Delta(q_2, \mathrm{a}, \mathrm{a}) = \{(q_2, \varepsilon)\}$$

$$\Delta(q_2, b, b) = \{(q_2, \varepsilon)\}$$

Ejercicio

12) Construya el ADPND equivalente a la siguiente gramática, aplicando el método explicado en la presente sección.

X → X+T | T
T → T*F | F
F → a | (X)

5.11 Proyecto de elaboración de una gramática BNF

La notación BNF se utiliza para especificar lenguajes de programación. Es así, como el presente proyecto consiste en diseñar parte de un lenguaje de programación, algunos elementos ya están definidos en el texto y se solicita definir otros.

Se deben aprovechar las definiciones previas a la que se esté realizando. En otras palabras, no deben redefinirse categorías que ya se encuentren definidas en puntos anteriores.

El procedimiento sugerido para realizar el proyecto es realizar lo solicitado para cada una de los siguientes puntos.

1) Categorías básicas

<Mayuscula> ::= “A” | “B” | “C” | … | “Z”

<Minuscula> ::= “a” | “b” | “c” | … | “z”

<Letra> ::= <Mayuscula> | <Minuscula>

<Digito> ::= “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”

2) Entero corto

<Entero> ::= <Dígito><Entero> | <Digito>”c”

3) Identificadores

Los identificadores están dados por la expresión regular $(L \cup \_)(L \cup D \cup \_ \cup \$\$)^*$

Escriba la definición BNF de los identificadores.

4) Operadores aditivos

<OperadorAditivo> :: = “+” | "-"

5) Operadores multiplicativos

Los operadores multiplicativos están dados por la expresión regular $* \cup / \cup \%$.

Escriba la definición BNF de los operadores multiplicativos.

6) Operadores relacionales

Escriba la definición BNF de los operadores relacionales, diseñados por usted en el proyecto de la sección 4.9.

7) Operadores lógicos

Escriba la definición BNF de los operadores lógicos, diseñados por usted en el proyecto de la sección 4.9

8) Operadores de asignación

Escriba la definición BNF de los operadores de asignación diseñados por usted en el proyecto de la sección 4.9.

9) Factor, término y expresión

<Factor> ::= <Identificador> | <Entero> | <Real> | "(" <Expresion> ")"

<Termino> ::= <Termino> <OperadorMultiplicativo> <Factor> | <Factor>

<Expresion > ::= <Expresion> <OperadorAditivo> <Termino> | <Termino>

10) Instrucción de asignación

Escriba la definición BNF de las instrucciones de asignación. Estas instrucciones deben permitir evaluar una expresión y guardar el resultado en una variable.

11) Sentencia escriba

A continuación se dan tres ejemplos de la sentencia *escriba*.

i) escriba fila 5 posicion 10 resultado de x
ii) escriba fila n+5 posicion m+10 resultado de x+y

iii) escriba fila 5+18*25 posicion w +m + (q-z) resultado de x-y/z

Escriba la definición BNF de esta sentencia. Debe permitir derivar los ejemplos dados y cualquier otro de gramática similar.

12) Sentencia

Una sentencia puede ser una instrucción de asignación o una instrucción escriba.

Escriba la definición BNF de la categoría sintáctica *sentencia*

13) Lista de Sentencias

<ListaDeSentencias> ::=
<ListaDeSentencias>
<Sentencia> |
<Sentencia>

14) Sentencia realice … iteraciones: … fin realice

Esta sentencia debe permitir repetir 0 o más sentencias de cualquier tipo, un número de veces dado. Ejemplos:

i) realice 100 iteraciones:
a := a +1
imprima en fila 10 columna 10 resultado de a
fin realice

ii) realice (i+j)-w/20 iteraciones:
a := a +1
imprima en fila 10 columna 10 resultado de a
fin realice

Escriba la definición BNF de la sentencia *realice … iteraciones: … fin realice*

15) Condición simple

Nota: se supuso que el operador lógico NOT es *@NO*

<CondicionSimple> ::=
<Expresion> <OperadorRelacional> <Expresion> |
[@NO] "(" <Condicion> ")"

16) Condición

Nota: se supuso que los operadores lógicos AND y OR son @Y y @O

<Condición> ::=
<Condicion> @O <CondicionSimple> |
<Condicion> @Y <CondicionSimple> |
<CondicionSimple>

17) Ciclo con test al principio

Esta sentencia permite repetir cero o más veces cero o más sentencias. Un ejemplo de ciclo con test al principio es el ciclo *while d*el lenguaje Java.

a) Escriba un ejemplo de un ciclo con test al principio diseñado por Usted. La sentencia no debe ser demasiado similar a las conocidas, ni a los ejemplos dados en clase, y debe ser fácil de interpretar.

b) Escriba un segundo ejemplo del ciclo diseñado

c) Escriba la definición BNF del ciclo diseñado

18) Bifurcación en dos ramas

La bifurcación en dos ramas permite que un programa elija para ejecutar de entre dos listas de sentencias. Un ejemplo de una bifurcación en dos ramas es la sentencia if del lenguaje Java.

a) Escriba un ejemplo de la bifurcación en dos ramas diseñada por usted. La sentencia no debe ser demasiado similar a las conocidas, ni a la explicada en clase, y debe ser fácil de interpretar.

b) Escriba un segundo ejemplo de la bifurcación en dos ramas diseñada

c) Escriba la definición BNF de la bifurcación en dos ramas diseñada

19) Sentencia para abrir una página en Internet

a) Escriba un ejemplo de una sentencia diseñada por usted que abra una página en Internet, dada la URL de esta página.

b) Escriba un segundo ejemplo de la sentencia diseñada

c) Escriba la definición BNF de la sentencia diseñada

20) Sentencia

La categoría sintáctica *sentencia*, definida previamente, solo incluye dos tipos de sentencias.

Escriba de nuevo la definición BNF de la categoría sintáctica sentencia, pero que incluya los siguientes tipos de sentencias, definidos a lo largo de este proyecto:

Una instrucción de asignación
Una instrucción escriba
Una Sentencia realice … iteraciones: … fin realice
Una bifurcación en dos ramas
Un ciclo con test al principio
La sentencia de abrir una página Web

21) Reales

Los reales están dados por la expresión regular D^*, DDD^*

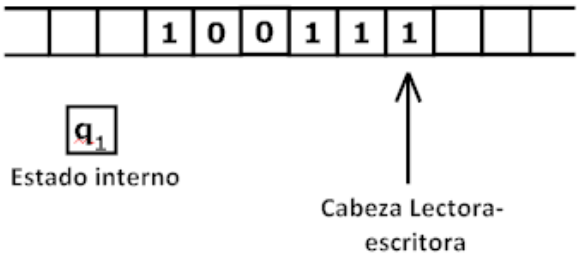
Escriba la definición BNF de los reales

6. Máquinas de Turing

La máquina de Turing es un modelo matemático simple de una computadora. Fue inventada por Turing en 1936.

Alan Mathison Turing, matemático, nació en Londres en 1912. Fue un verdadero pionero en el campo del desarrollo de las computadoras. Turing fue un entusiasta del Ajedrez. Junto con Champernowne inventó el primer programa jugador de Ajedrez.

La máquina de Turing es una máquina de estados. Existen muchas clases de máquinas de Turing, pero en esta sección sólo se tratará a manera de introducción un tipo sencillo. Un ejemplo de este tipo se puede apreciar en el diagrama que sigue. La máquina usa una cinta infinita a ambos lados y un cabeza lectora-escritora.



Para representar la configuración de la máquina en un momento dado, se utilizan descripciones instantáneas. Por ejemplo, la descripción instantánea de la configuración ilustrada en la figura es la siguiente:

$$(q_1, 100111)$$

Esta notación muestra el estado en el que se encuentra la máquina, lo almacenado en su cinta, y la posición de la cabeza lectora-

escritora. Esta última se indica subrayando la correspondiente posición.

La máquina de Turing, como modelo de computadora que es, puede ejecutar un programa, el cual se denomina función de transición. Como ejemplo, vamos a aplicar la siguiente función de transición a la máquina de la ilustración.

$$\delta(q_1,1) = (q_1,0,L)$$
$$\delta(q_1,0) = (q_2,1,R)$$
$$\delta(q_1,B) = (q_2,1,R)$$
$$\delta(q_2,0) = (q_2,0,R)$$
$$\delta(q_2,B) = (q_3,B,L)$$

Las transiciones son de la forma $\delta(q_i, x) = (q_j, y, direccion)$, donde tenemos lo siguiente.

- q_i es el estado de la máquina antes de utilizar la transición. La máquina de Turing debe estar en este estado, o no se puede aplicar la transición.
- *x* es el símbolo al que apunta la cabeza lectora-escritora. La máquina de Turing lee este símbolo. *x* también puede ser un espacio en blanco, representado por B, el cual será leído por la máquina. La cabeza tiene que estar señalando el símbolo *x,* o no se podrá utilizar esta transición.
- q_j es el estado de la máquina después de aplicarse la transición.
- *y* es el símbolo que sobre-escribe la máquina. El símbolo *y* también puede ser un espacio en blanco, representado por *B*, el cual sobre-escribirá la máquina.
- *dirección* es L o R. L indica que la máquina moverá la cabeza lectora-escritora hacia la izquierda y R indica que la máquina moverá la cabeza hacia la derecha.

La máquina funciona ejecutando el siguiente ciclo.

1. Busca que transición puede utilizar. El estado inicial de la transición debe concordar con el estado actual de la máquina y el símbolo a leer de la transición debe concordar con el símbolo señalado por la cabeza lectora-escritora. En todo momento la máquina puede utilizar exactamente una transición, o si no quiere decir que la función de transición está mal construida. No sería una función.

   Si no se puede utilizar ninguna transición, la máquina se detiene.

2. Se aplica la transición, según lo explicado anteriormente.
3. Retrocede al paso 1.

A continuación se muestran los sucesivos cambios que ocurren en la máquina de Turing de la ilustración, al aplicarle su respectiva función de transición. La configuración de la máquina en cada momento será representada con una descripción instantánea:

Primero se aplica la transición $\Delta(q_1, 1) = \{(q_1, 0.L)\}$ ya que la máquina se encuentra en estado q_1 y la cabeza señala 1. La transición hace que la máquina pase al estado q_1, sobre-escriba un cero y mueva la cabeza a la izquierda. Formalmente, la máquina cambia su configuración de la siguiente manera:

$$(q_1, 100111) \vdash (q_1, 100110)$$

Los restantes cambios en la configuración de la máquina son los siguientes:

$(q_1, 100110) \vdash (q_1, 100100) \vdash (q_1, 100000) \vdash (q_2, 101000) \vdash (q_2, 101000) \vdash (q_2, 101000) \vdash (q_2, 101000\_) \vdash (q_3, 101000)$

No hay ninguna transición que comience por $\delta(q_3, 0)$, por lo que la máquina se detiene.

Puede observarse que lo que ha realizado la máquina es sumar 1 en binario.

Ejercicio con respuesta

1) Aplique la siguiente función de transición de una máquina de Turing a la configuración dada.

Función de transición

δ (q_1 , a) = (q_1 , a , R)
δ (q_1 , b) = (q_2 , a , R)
δ (q_2 , a) = (q_2 , a , R)
δ (q_2 , b) = (q_2 , b , R)
δ (q_2 , B) = (q_3 , B , L)

Configuración

(q_1 , <u>a</u>babb)

Nota: B representa un espacio en blanco

¿En cuál de las siguientes configuraciones para la máquina?

A. (q_3 , aaab<u>b</u>)

B. (q_3 , aaaa<u>a</u>)

C. (q_3 , <u>a</u>aabb)

D. (q_3 , aaabb_)

E. (q_3 , <u>a</u>aaaa)

Respuesta: A

Ejercicios

2) Aplique la siguiente función de transición de una máquina de Turing a la descripción instantánea dada.

   Función de transición:

$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_1, b, R)$$
$$\delta(q_1, B) = (q_2, B, L)$$

   Descripción instantánea: (q_1 ,abbab)

3) Aplique la siguiente función de transición de una máquina de Turing a la descripción instantánea dada.

Función de transición:

$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_1, b, L)$$

   Descripción instantánea: (q_1 ,abbab)

4) Construya una máquina de Turing que analice una cadena sobre {a,b} y cambie todas sus bes por aes. La cabeza inicia sobre el primer símbolo de la izquierda y debe terminar sobre este símbolo. Por ejemplo, si la descripción instantánea inicial es

   (q_1 , ababba)

   la descripción instantánea en el momento de la parada debe ser, por ejemplo:

   (q_3 , aaaaaa).

5) Construya una máquina de Turing que junte dos cadenas de unos separadas por un espacio. Por ejemplo, si la descripción instantánea inicial es

(q_1 , 1111B111111)

la final podría ser algo como

(q_4 , 1111111111)

7. Bibliografía

[DE CASTRO] DE CASTRO, Rodrigo. Teoría de la computación. Lenguajes, autómatas, gramáticas. Bogotá: Universidad Nacional de Colombia, sede Bogotá, 2004.

[BRENA] BRENA, Ramón. Autómatas y Lenguajes. Tecnológico Monterrey. 2003. Disponible en Internet: <URL: http://lizt.mty.itesm.mx/~rbrena/AyL.html >

[CASES] CASES MUÑOZ, Rafael; MARQUEZ VILLADRE, Lluís. Lenguajes, gramáticas y autómatas: curso básico. México: Alfaomega, 2002

[JOHNSONBAUGH] JOHNSONBAUGH, Richard. Matemáticas Discretas. Méjico: Pearson Prentice Hall, 1999.

[ISASI] ISASI VIÑUELA, Pedro; MARTINEZ FERNANDEZ, Paloma y BORRAJO MILLAN, Daniel. Lenguajes, Gramáticas y Autómatas, un enfoque práctico. Universidad Carlos III, Madrid. España: Addison-Wesley. 1997.

[LEMON] LEMON, Karen A. Fundamentos de compiladores, cómo traducir al lenguaje de computadora. CECSA. Méjico, 1996.

[KELLY] KELLEY, Dean. Teoría de Autómatas y Lenguajes Formales. Prentice Hall, 1995.

[HOPCROFT] HOPCROFT Y ULLMAN. Introducción a la Teoría de Autómatas, Lenguajes y Computación. Editorial Cecsa. 1993.

[GLENN] BROOKSHEAR, J. Glenn. Teoría de la computación: Lenguajes formales, autómatas y complejidad. Addison-Wesley. 1993.

[VISO] VISO GUROVICH, Elisa. Introducción a la Teoría de la Computación (Autómatas y Lenguajes Formales). Universidad

Nacional Autónoma de Méjico, 2008. Disponible en Internet: <URL: http://books.google.es/books?id=NXQE8NJw9d4C&source=gbs_navlinks_s>

[NEPOMUCENO] NEPOMUCENO FERNÁNDEZ, Ángel; QUESADA MORENO, José Francisco; SALGUERO LAMILLAR , Francisco José. Información: Tratamiento y Representación. Universidad de Sevilla, 2001. Disponible en Internet: <URL: http://books.google.es/books?id=Q1BSlLveu7wC&source=gbs_navlinks_s>

[GOMEZ] GOMEZ DE SILVA GARZA, Andrés; ANIA BRISENO, Ignacio de Jesús. Introducción a la Computación. Méjico: Cengage Learning Editores, 2008. Disponible en Internet: <URL: http://books.google.es/books?id=ov3E_De2p6MC&source=gbs_navlinks_s>

[PÉREZ] PÉREZ, Iván. Lenguaje y compiladores. Caracas: Universidad Católica Andrés Bello, 2005. Disponible en Internet: <URL: http://books.google.es/books?id=X4-MGtEw5TAC&source=gbs_navlinks_s>

[CHOMSKY] CHOMSKY, Noam. Syntactic Structures. Berlín: Walter de Gruyter GmbH & Co. KGm 10785, 2002. Disponible en Internet: <URL: http://books.google.es/books?id=SNeHkMXHcd8C&dq=inauthor:chomsky&source=gbs_navlinks_s>

[LOUDEN] LOUDEN, Kenneth C. Lenguajes de Programación: Principios y práctica. Méjico: International Thomson Editores S.A., 2004. Disponible en Internet: <URL: http://books.google.es/books?id=MnrVc_GVKbMC&source=gbs_navlinks_s>