Ciencias computacionales

Propedeutico: Teoría de Autómatas y Lenguajes Formales

Expresiones regulares y lenguajes

Contents

1 Expresiones regulares

Las expresiones regulares son un equivalente algebraico para un autómata. Utilizado en muchos lugares como un lenguaje para describir patrones en texto que son sencillos pero muy útiles. Pueden definir exactamente los mismos lenguajes que los autómatas pueden describir: Lenguajes regulares. Además, ofrecen algo que los autómatas no: Manera declarativa de expresar las cadenas que queremos aceptar.

Sirven como lenguaje de entrada a muchos sistemas que procesan cadenas tales como:

- Comandos de búsqueda, e.g., grep de UNIX
- Sistemas de formateo de texto: Usan notación de tipo expresión regular para describir patrones
- Convierte la expresión regular a un DFA o un NFA y simula el autómata en el archivo de búsqueda
- Generadores de analizadores-léxicos. Como Lex o Flex.
- Los analizadores léxicos son parte de un compilador. Dividen el programa fuente en unidades lógicas (tokens). Tokens como while, números, signos ($+, -, <$, etc.)
- Produce un DFA que reconoce el token.

De forma más precisa, Las expresiones regulares denotan lenguajes. Por ejemplo, la expresión regular: $01^* + 10^*$ denota todas las cadenas que son o un 0 seguido de cualquier cantidad de 1's o un 1 seguida de cualquier cantidad de 0's.

1.1 Los operadores de las expresiones regulares

Comunmente existen tres operadores de las expresiones regulares: Unión, concatenación y cerradura. Si L y M son dos lenguajes, su unión se denota por $L \cup M$ e.g. $L = \{001,10,111\}$, $M = \{\epsilon,001\}$, entonces la unión será $L \cup M = \{\epsilon,10,001,111\}$.

La concatenación de lenguajes se denota como LM o $L.M$ e.g. $L = \{001,10,111\}$, $M = \{\epsilon,001\}$, entonces la concatenación será $LM = \{001,10,111,001001,10001,111001\}$.

Finalmente, la cerradura (o cerradura de Kleene) de un lenguaje L se denota como L^*. Representa el conjunto de cadenas que puede que pueden formarse tomando cualquier número de cadenas de L, posiblemente con repeticiones y concatenando todas ellas e.g. si $L = \{0, 1\}$, L^* son todas las cadenas con 0's y 1's. Si $L = \{0, 11\}$, entonces L^* son todas las cadenas de 0's y 1's tal que los 1's están en pareja. Para calcular L^* se debe calcular L^i para cada i y tomar la unión de todos estos lenguajes. L^i tiene 2^i elementos. Aunque cada L^i es finito, la unión del número de terminos de L^i es en general un conjunto infinito e.g. $\emptyset^* = \{\epsilon\}$ o $\emptyset^0 = \{\epsilon\}$. Generalizando, para todo i mayor o igual que uno, $\emptyset^i$ es el conjunto vacío, no se puede seleccionar ninguna cadena del conjunto vacío.

1.2 Construcción de expresiones regulares

Si E es una expresión regular, entonces $L(E)$ denota el lenguaje que define E. Las expresiones se construyen de la manera siguiente:

1. Las constantes ϵ y $\emptyset$ son expresiones regulares que representan a los lenguaje $L(\epsilon) = \{\epsilon\}$ y $L(\emptyset) = \emptyset$ respectivamente
2. Si a es un símbolo, entonces es una expresión regular que representan al lenguaje: $L(a) = \{a\}$

1. Si E y F son expresiones regulares, entonces $E + F$ también lo es denotando la unión de $L(E)$ y $L(F)$. $L(E + F) = L(E) \cup L(F)$.
2. Si E y F son expresiones regulares, entonces EF también lo es denotando la concatenación de $L(E)$ y $L(F)$. $L(EF) = L(E)L(F)$.

3. Si E es una expresión regular, entonces E^* también lo es y denota la cerradura de $L(E)$. Osea $L(E^*) = (L(E))^*$

4. Si E es una expresión regular, entonces (E) también lo es. Formalmente: $L((E)) = L(E)$.

Ejemplo 1 *Escriba la expresión regular para el conjunto de cadenas que consisten en 0's y 1's alternados.*

Si fuera una cadena sin repetición, el resultado fuera 01, sin embargo por cerradura se puede tener $010101\cdots01$. Una regla básica nos dice que 0 y 1 son expresiones que denotan los lenguajes $\{0\}$ y $\{1\}$, concatenando ambos lenguajes se obtiene entonces la expresión para el Ejemplo 1 que es $\{01\}$.

Ejemplo 2 *Escriba la expresión regular para todas las cadenas consisten de cero o más ocurrencias de 01.*

La solución del Ejemplo 2 es $(01)^*$, no se debe confundir con 01^* que es la expresión de todas las cadenas que comienzan con 0 y terminan con 1.

1.3 Precedencia de operaciones de las expresiones regulares

Las reglas básicas son las siguientes:

1. El asterisco de la cerradura tiene la mayor precedencia
2. Concatenación sigue en precedencia a la cerradura, el operador "dot". Concatenación es asociativa y se sugiere agrupar desde la izquierda (i.e. 012 se agrupa (01)2).
3. La unión (operador +) tiene la siguiente precedencia, también es asociativa.
4. Los paréntesis pueden ser utilizados para alterar el agrupamiento

1.4 Ejercicios

Ejercicio 1 *Explique las expresiones regulares que denotadas por el siguiente lenguaje:* $L(001)$

Ejercicio 2 *Explique las expresiones regulares que denotadas por el siguiente lenguaje:* $L(0 + 10^*)$

Solución: $L(0 + 10^*) = \{0, 1, 10, 100, 1000, \ldots\}$.

Ejercicio 3 *Explique las expresiones regulares que denotadas por el siguiente lenguaje:* $L((0(0 + 1))^*)$

Ejercicio 4 *Expresión regular de cadenas que alterna 0's y 1's.*

- Solución(1): $(01)^* + (10)^* + 0(10)^* + 1(01)^*$ (opción 1)

- Solucion(2):

Ejercicio 5 *Qué lenguaje está representado por la expresión regular* $L(((a^*a)b) \cup b)$

Ejercicio 6 *Escriba las expresiones regulares para los siguientes lenguajes*

- *El conjunto de cadenas de 0s y 1s con a lo más un par consecutivo de 1s.*
- *El conjunto de cadenas de 0s y 1s cuyo número de 0s es divisible entre cinco.*
- *El conjunto de cadenas de 0s y 1s cuyo número de 0s es divisible entre cinco y el número de 1s es par.*

2 Autómatas finitos y expresiones regulares

Si se necesita demostrar que para cada expresión regular, existe un autómata finito que acepta el mismo lenguaje. Se selecciona el autómata más apto $\epsilon - NFA$. Si por el contrario, se necesita demostrar que por cada autómata finito, hay una expresión regular definiendo su lenguaje se selecciona el autómata con mayores restricciones: DFA. Los lenguajes aceptados por DFA, NFA, ϵ-NFA, RE son llamados lenguajes regulares.

Para convertir expresiones regulares a ϵ-NFA se realizan pruebas por inducción en los diferentes operadores (+, concatenation, *) en la expresión regular. Siempre se construye un autómata de una forma especia.

Se mostrará que un NFA con transiciones-ϵ puede aceptar el lenguaje de una RE. Después, se mostrará que un RE puede describir el lenguaje de un DFA (la misma construcción funciona para un NFA). Los lenguajes aceptados por DFA, NFA, ϵ-NFA, RE son llamados lenguajes regulares.

Teorema 1 *Si $L = L(A)$ para algún DFA A, entonces existe una expresión regular R tal que $L = L(R)$.*

Prueba: Suponiendo que A tiene estados $\{1, 2, \ldots, n\}$, n finito. Tratemos de construir una colección de RE que describan progresivamente conjuntos de rutas del diagrama de transiciones de A

- $R_{ij}^{(k)}$ es el nombre de la RE cuyo lenguaje es el conjunto de cadenas w.
- w es la etiqueta de la ruta del estado i al estado j de A. Esta ruta no tiene estado intermedio mayor a k. Los estados inicial y terminal no son intermedios, i y/o j pueden ser igual o menores que k.
- Para construir $R_{ij}^{(k)}$ se utiliza una definición inductiva de $k = 0$ hasta $k = n$
- BASE: $k = 0$, implica que no hay estados intermedios. Sólo dos clases de rutas cumplen con esta condición:
  1. Un arco del nodo (estado) i al nodo j
  2. Una ruta de longitud 0 con un solo nodo i
- Si $i \neq j$, solo el caso 1 es posible.

Examinar el DFA A y encontrar los símbolos de entrada a tal que hay una transición del estado i al estado j con el símbolo a

- Si no hay símbolo a, entonces $R_{ij}^{(0)} = \emptyset$.
- Si hay sólo un símbolo a, entonces $R_{ij}^{(0)} = a$.
- Si hay varios símbolos $a_1, a_2, \ldots, a_k$, entonces $R_{ij}^{(0)} = a_1 + a_2 + \ldots + a_k$.

Si $i = j$, sólo se permiten rutas de longitud 0 y ciclos del estado i a él mismo.

- La ruta de longitud 0 se representa con ϵ
- Si no hay símbolo a, entonces $R_{ij}^{(0)} = \emptyset$.
- Si hay sólo un símbolo a, entonces $R_{ij}^{(0)} = \epsilon + a$.
- Si hay varios símbolos $a_1, a_2, \ldots, a_k$, entonces $R_{ij}^{(0)} = \epsilon + a_1 + a_2 + \ldots + a_k$.

INDUCCIÓN: Suponemos que hay una ruta del estado i al estado j que no pasa por ningún estado mayor que k.

Se consideran 2 casos.

1. La ruta no pasa por el estado k: La etiqueta de la ruta está en el lenguaje $R_{ij}^{(k-1)}$.

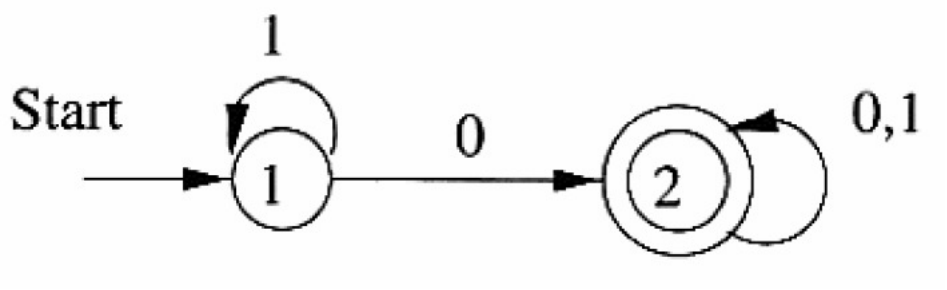

Figure 1: DFA que acepta todas las cadenas que tienen al menos un 0.

2. La ruta pasa por el estado k al menos una vez:

   - Se divide la ruta en varias partes, una división cada que se pasa por el estado k
   - Primero del estado i al estado k, después, varios pasos del estado k a sí mismo, finalmente, del estado k al estado j.

   Las etiquetas de estas rutas se representan con la RE: $R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$

- Si combinamos las expresiones de las rutas de los dos tipos: $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$ para todas las etiquetas de las rutas del estado i al j que no pasan por estados mayores que k.
- Eventualmente tendremos $R_{ij}^{(n)}$.
- Asumimos que 1 es el estado inicial. El estado de aceptación puede ser un conjunto de estados La expresión regular para el lenguaje del autómata es la suma (unión) de todas las expresiones $R_{1j}^{(n)}$ tal que j es un estado de aceptación.

Ejemplo 3 *Un DFA que acepta todas las cadenas que tienen al menos un 0. Ver Figura 1.*

Inicialmente sustituimos para la base: (i) $R_{ij}^{(0)} = \epsilon$, (ii) $R_{ij}^{(0)} = \epsilon + a$ y (iii) $R_{ij}^{(0)} = \epsilon + a_1 + a_2 + \ldots + a_k$

$R_{11}^{(0)} = \epsilon + 1$
$R_{12}^{(0)} = 0$
$R_{21}^{(0)} = \emptyset$
$R_{22}^{(0)} = (\epsilon + 0 + 1)$

Ahora para el paso de inducción: $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^* R_{1j}^{(0)}$

	Por sustitución directa	Simplificado
$R_{11}^{(1)} =$	$\epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1)$	1^*
$R_{12}^{(1)} =$	$0 + (\epsilon + 1)(\epsilon + 1)^* 0$	1^*0
$R_{21}^{(1)} =$	$\emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1)$	$\emptyset$
$R_{22}^{(1)} =$	$\epsilon + 0 + 1 + \emptyset(\epsilon + 1)^* 0$	$\epsilon + 0 + 1$

$R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)}(R_{22}^{(1)})^* R_{2j}^{(1)}$

	Por sustitución directa	Simplificado
$R_{11}^{(2)} =$	$1^* + 1^*0(\epsilon + 0 + 1)^*\emptyset$	1^*
$R_{12}^{(2)} =$	$1^*0 + 1^*0(\epsilon + 0 + 1)^*(\epsilon + 0 + 1)$	$1^*0(0 + 1)^*$
$R_{21}^{(2)} =$	$\emptyset + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*\emptyset$	$\emptyset$
$R_{22}^{(2)} =$	$\epsilon + 0 + 1 + (\epsilon + 0 + 1)(\epsilon + 0 + 1)^*$ $(\epsilon + 0 + 1)$	$(0 + 1)^*$

Para la construcción de un RE final se puede utilizar la unión de todas las expresiones donde el primer estado es el estado inicial y el segundo el estado de aceptación.

- Estado inicial: 1
- Estado final: 2
- Sólo necesitamos $R_{12}^{(2)}$
- $1^*0(0+1)^*$

Este método funciona también para NFA y ϵ-NFA pero su construcción es muy costosa, hasta en el orden de 4^n símbolos.

Ejemplo 4 *Dada la tabla de transición de un DFA:*

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

a) Dar todas las expresiones regulares para $R_{ij}^{(0)}$

b) Dar todas las expresiones regulares para $R_{ij}^{(1)}$

2.1 Conversión de un DFA a una RE por eliminación de estados

- Evita duplicar trabajo en algunos puntos del teorema anterior
- Ahora utilizaremos autómatas que podrán tener RE como etiquetas.
- El lenguaje del autómata es la unión de todas las rutas que van del estado inicial a un estado de aceptación.
  - Concatenando los lenguajes de las RE que van a través de la ruta.
  - En la siguiente figura se muestra un autómata al cual se va a eliminar el estado "s".

Se muestra un ejemplo gráfico en la Figura 2.

- Se eliminan todos los arcos que incluyen a "s"
- Se introduce, para cada predecesor q_i de s y cada sucesor p_j de s, una RE que representa todas las rutas que inician en q_i, van a s, quizás hacen un *loop* en s cero o más veces, y finalmente van a p_j.
- La expresión para estas rutas es $Q_i S^* P_j$.
- Esta expresión se suma (con el operador unión) al arco que va de q_i a p_j.
- Si este arco no existe, se añade primero uno con la RE $\emptyset$
- El autómata resultante después de la eliminación de "s" es el siguiente:

Se muestra un ejemplo gráfico en la Figura 3.
La estrategia para construir el autómata

1. Para cada estado de aceptación q, aplicar el proceso de reducción para producir un autómata equivalente con RE como etiquetas en los arcos. Eliminar todos los estados excepto q y el estado inicial q_0.
2. Si $q \neq q_0$, se genera un autómata con 2 estados como el siguiente, una forma de describir la RE de este autómata es $(R + SU^*T)^*SU^*$

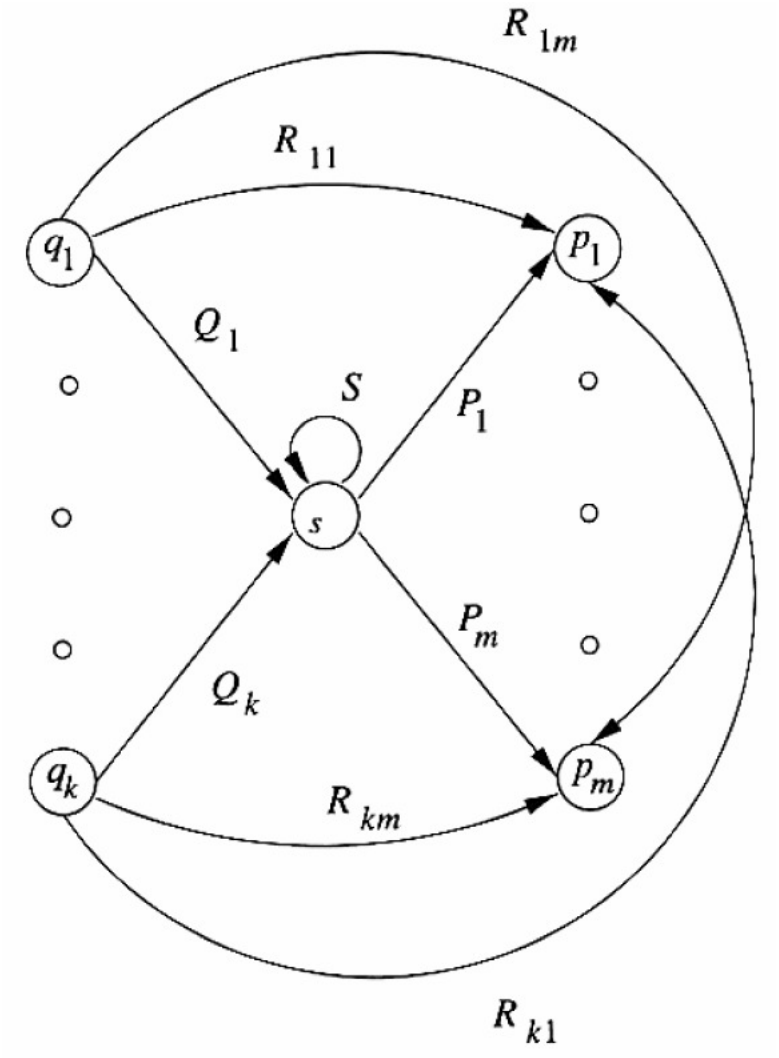

Figure 2: Ejemplo de DFA a RE por eliminación de estados.

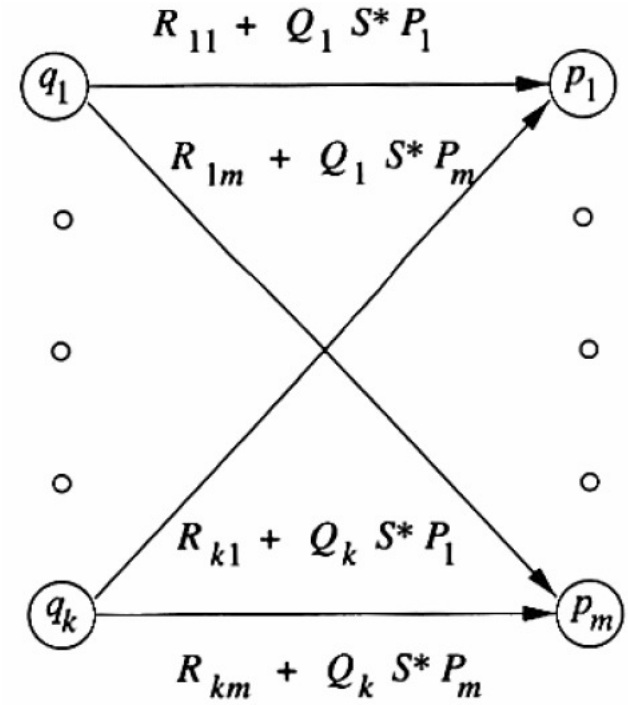

Figure 3: Eliminación de estados.

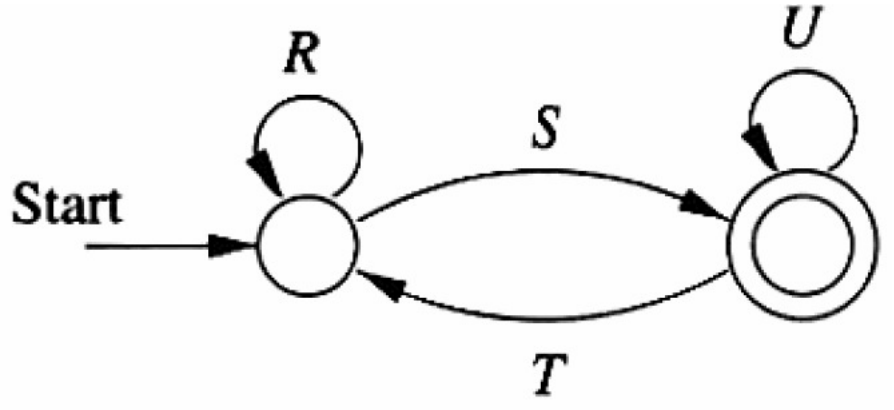

Figure 4: Construcción del autómata.

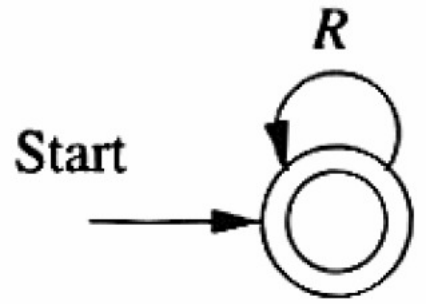

Figure 5: Construcción del autómata.

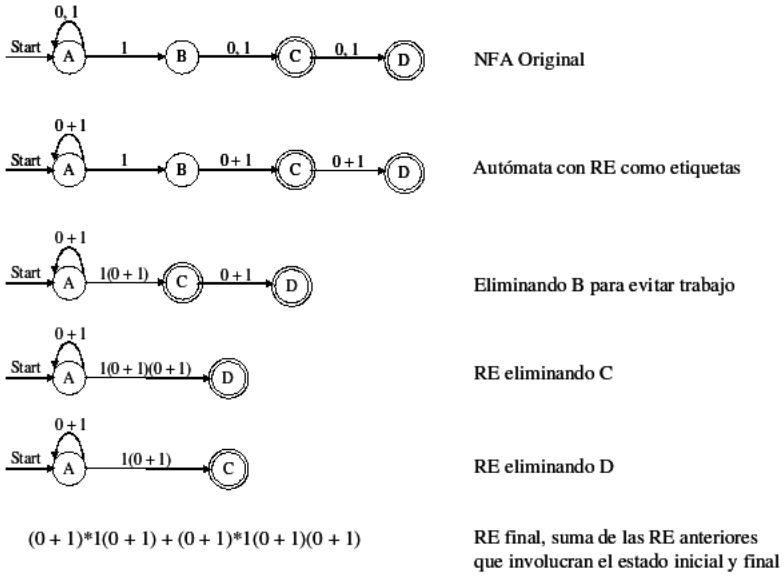

Figure 6: NFA.

Se muestra un ejemplo gráfico en la Figura 4.

Si el estado inicial también es un estado de aceptación, también se debe hacer una eliminación de estados del autómata original que elimine todos los estados menos el inicial y dejamos un autómata como se muestra en la Figura 5.

La RE final es la suma (unión) de todas las expresiones derivadas del autómata reducido para cada estado de aceptación por las reglas 2 y 3.

Ejercicio 7 *Construir el autómata para el NFA que se muestra en la Figura 6 el cual acepta cadenas de 0's y 1's de manera que tienen un 1 dos o tres posiciones antes del final.*

2.2 Convirtiendo una RE a un autómata

Teorema 2 *Todo lenguaje definido por una RE también esta definido por un autómata finito.*

Prueba: Suponemos $L = L(R)$ para la expresión regular R. Mostramos que $L = L(E)$ para algún ϵ-NFA E con:

1. Exactamente un estado de aceptación
2. Sin arcos que lleguen al estado inicial
3. Sin arcos que salgan del estado de aceptación

Base: cumpliendo las condiciones 1, 2, y 3. Ver Figura 7.
Inducción:

a)

b)

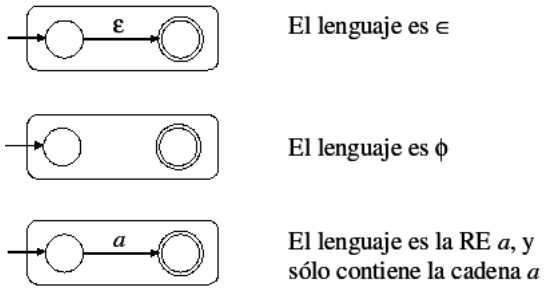

Figure 7: Caso base.

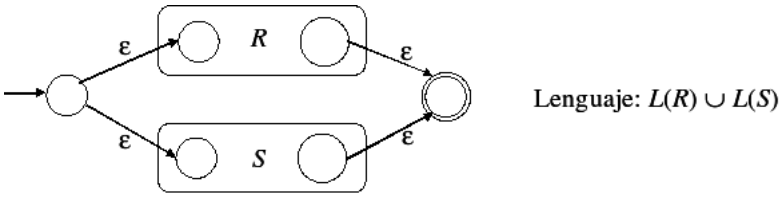

Figure 8: Induccion (a).

3 Aplicaciones de expresiones regulares

Entre las aplicaciones de expresiones regulares se encuentran:

- Comandos de búsqueda, e.g., grep de UNIX
- Sistemas de formateo de texto: Usan notación de tipo expresión regular para describir patrones
- Convierte la expresión regular a un DFA o un NFA y simula el autómata en el archivo de búsqueda
- Generadores de analizadores-léxicos. Como Lex o Flex.
- Los analizadores léxicos son parte de un compilador. Dividen el programa fuente en unidades lógicas (tokens). Tokens como while, números, signos ($+, -, <$, etc.)
- Produce un DFA que reconoce el token

4 Reglas algebraicas para expresiones regulares

Existen un conjunto de leyes algebraicas que se pueden utilizar para las expresiones regulares:

- Ley conmutativa para la unión: $L + M = M + L$
- Ley asociativa para la unión: $(L + M) + N = L + (M + N)$
- Ley asociativa para la concatenación: $(LM)N = L(MN)$

NOTA: La concatenación no es conmutativa, es decir $LM \neq ML$

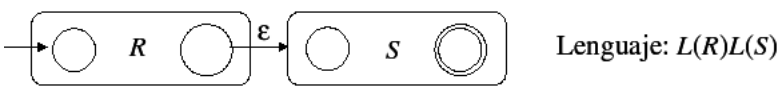

Figure 9: Induccion (b).

4.1 Identidades y "aniquiladores"

Una identidad para un operador es un valor tal que cuando el operador se aplica a la identidad y a algún otro valor, el resultado es el otro valor. Por ejemplo, 0 es la identidad de la suma, ya que $0+x = x+0 = x$, la identidad para la multiplicación es 1 debido a que $1x = x1 = x$. Un aniquilador para un operador es un valor tal que cuando el operador se aplica al aniquilador y algún otro valor, el resultado es el aniquilador. 0 es el aniquilador para la multiplicación: $0 \times x = x \times 0 = 0$. No hay aniquilador para la suma.

Estas leyes las utilizamos para hacer simplificaciones: $\emptyset$ es la identidad para la unión: $\emptyset + L = L + \emptyset = L$. ϵ es la identidad para la concatenación: $\epsilon L = L\epsilon = L$. $\emptyset$ es el aniquilador para la concatenación: $\emptyset L = L\emptyset = \emptyset$.

4.2 Ley distributiva

Como la concatenación no es conmutativa, tenemos dos formas de la ley distributiva para la concatenación:

- Ley Distributiva Izquierda para la concatenación sobre unión: $L(M + N) = LM + LN$
- Ley Distributiva Derecha para la concatenación sobre unión: $(M + N)L = ML + NL$

4.3 Ley de idem-potencia

Se dice que un operador es idempotente (*idempotent*) si el resultado de aplicarlo a dos argumentos con el mismo valor es el mismo valor. En general la suma no es idempotente: $x + x \neq x$ (aunque para algunos valores sí aplica como $0 + 0 = 0$). En general la multiplicación tampoco es idempotente: $x \times x \neq x$. La unión e intersección son ejemplos comunes de operadores idempotentes. La ley idempotente para la unión: $L + L = L$.

4.4 Leyes relacionadas con la propiedad de cerradura

Las leyes involucradas con la propiedad de cerradura:

- $(L^*)^* = L^*$ (Idempotencia para la cerradura)
- $\emptyset^* = \epsilon$
- $\epsilon^* = \epsilon$
- $L^+ = LL^* = L^*L, L^+$ se define como $L + LL + LLL + \ldots$
- $L^* = \epsilon + L + LL + LLL + \ldots$
- $LL^* = L\epsilon + LL + LLL + LLLL + \ldots$
- $L^* = L^+ + \epsilon$
- $L? = \epsilon + L$

4.5 Descubrir leyes para expresiones regulares

Se puede proponer una variedad infinita de leyes para expresiones regulares. El problema se reduce a probar la igualdad de dos lenguajes específicos:

Ejemplo 5 *Probar que* $(L + M)^* = (L^*M^*)^*$

Es necesario probar que las cadenas que están en $(L + M)^*$ también están en $(L^*M^*)^*$, y probamos que las cadenas que están en $(L^*M^*)^*$ también están en $(L + M)^*$.

Cualquier RE con variables se puede ver como una RE concreta sin variables, viendo cada variable como si fuera un símbolo diferente. La expresión $(L + M)^*$ se puede ver como $(a + b)^*$. Utilizamos esta forma como una guía para concluir sobre los lenguajes. Podemos analizar el lenguaje que nos describe: $(a + b)^*$ y analizar el lenguaje que nos describe: $(a^*b^*)^*$.

4.6 La prueba de una ley algebraica para expresiones regulares

Para probar una Ley Algebraica para una expresión regular, se siguen los siguientes pasos:

1. Convertir E y F a RE concretas C y D, respectivamente, reemplazando cada variable por un símbolo concreto.
2. Probar si $L(C) = L(D)$. Si es cierto, entonces $E = F$ es una ley verdadera y si no, la "ley" es falsa.

Ejemplo: $L^* = L^*L^*$

4.7 Ejercicios

Resolver los siguientes ejercicios:

Ejercicio 8 *Probar que se cumple o no se sumple:* $R + S = S + R$

Ejercicio 9 *Probar que se cumple o no se sumple:* $(R^*)^* = R^*$

Ejercicio 10 *Probar que se cumple o no se sumple:* $(R + S)^* = R^* + S^*$

Ejercicio 11 *Probar que se cumple o no se sumple:* $S(RS + S)^*R = RR^*S(RR^*S)^*$

Ejercicio 12 *Utilice las leyes distributivas para escribir dos expresiones más simples de la siguiente expresión.*

$$(0 + 1)^*1(0 + 1) + (0 + 1)^*1(0 + 1)(0 + 1)$$