Expresiones Regulares

INAOE

Contenido

Expresiones Regulares

Es 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
- Ofrecen algo que los autómatas no: Manera declarativa de expresar las cadenas que queremos aceptar

Expresiones Regulares

- Ejemplos de sus usos
  - 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*), como *while*, números, signos ($+, -, <$, etc.)
  - Produce un DFA que reconoce el *token*

Expresiones Regulares

- 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 seguido de cualquier cantidad de 0's.
- Operaciones de los lenguajes:
  1. Unión: Si L y M son dos lenguajes, su unión se denota por $L \cup M$ (e.g., $L = \{11, 00\}$, $M = \{0, 1\}$, $LcuoM = \{0, 1, 00, 11\}$)
  2. Concatenación: La concatenación es: LM o $L.M$ (e.g., $LM = \{110, 111, 000, 001\}$)
  3. Cerradura (o cerradura de Kleene): Si L es un lenguaje su cerradura se denota por: L^* ($L^0 = \{\epsilon\}, L^1 = L, L^2 = LL$. Si $L = \{0, 11\}, L^2 = \{00, 011, 110, 1111\}$)

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: $L(a) = \{a\}$

Operandos

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: $L((E)) = L(E)$.

Precedencia

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

Ejemplos

- $L(001) = 001$.
- $L(0 + 10^*) = \{0, 1, 10, 100, 1000, \ldots\}$.
- $L((0(0 + 1))^*)$ = el conjunto de cadenas de 0's y 1's, de longitud par, de tal manera que cada posición impar tenga un 0.
- Expresión regular de cadenas que alterna 0's y 1's:
  1. $(01)^* + (10)^* + 0(10)^* + 1(01)^*$ (opción 1)
  2. $(\epsilon + 1)(01)^*(\epsilon + 0)$ (opción 2)

Ejemplos

1. Encuentra la expresión regular para el conjunto de cadenas sobre el alfabeto $\{a, b, c\}$ que tiene al menos una a y al menos una b
2. Encuentra la expresión regular para el conjunto de cadenas de 0's y 1's tal que cada par de 0's adyacentes aparece antes de cualquier par de 1's adyacentes

Soluciones

1. $c^*a(a+c)^*b(a+b+c)^* + c^*b(b+c)^*a(a+b+c)^*$
   Osea, cuando la primera *a* esta antes que la primera *b* o cuando la primera *b* está antes de la primera *a*
2. $(10+0)^*(\epsilon+1)(01+1)^*(\epsilon+1)$
   $(10+0)^*(\epsilon+1)$ es el conjunto de cadenas que no tienen dos 1's adyacentes. La segunda parte es el conjunto de cadenas que no tienen dos 0's adyacentes. De hecho $\epsilon+1$ lo podríamos eliminar porque se puede obtener el 1 de lo que sigue, por lo que podemos simplificarlo a: $(10+0)^*(01+1)^*(\epsilon+1)$

Equivalencia de Lenguajes de FA y Lenguajes RE

- 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.

De DFA's a Expresiones Regulares

Teorema 3.4: 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

De DFA's a Expresiones Regulares

- 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.

De DFA's a Expresiones Regulares

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 $R_{ij}^{(0)} = a_1 + a_2 + \ldots + a_k$.

De DFA's a Expresiones Regulares

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 $R_{ij}^{(0)} = \epsilon + a_1 + a_2 + \ldots + a_k$.

Expresiones Regulares
Operadores y Operandos
Equivalencia de Lenguajes de FA y Lenguajes RE
Leyes Algebraicas de las Expresiones Regulares

De DFA's a Expresiones Regulares

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 $R_{ij}^{(k-1)}$.
2. La ruta pasa por el estado k al menos una vez:
   - Se divide la ruta en varias partes, una división por cada vez 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.

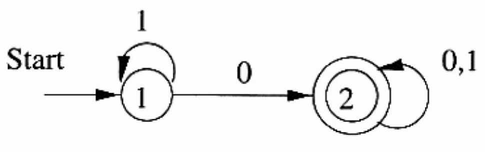
   Las etiquetas de estas rutas se representan con la RE:
   $R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^{*}R_{kj}^{(k-1)}$

De DFA's a Expresiones Regulares

- 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 $R_{ij}^{(n)}$.
- Suponemos 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

Un DFA que acepta todas las cadenas que tienen al menos un 0



Ejemplo

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

$R_{11}^{(0)} = \epsilon + 1$

$R_{12}^{(0)} = 0$

$R_{21}^{(0)} = \emptyset$

$R_{22}^{(0)} = (\epsilon + 0 + 1)$

Ejemplo

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

Ejemplo

Expresiones Regulares

Operadores y Operandos

Equivalencia de Lenguajes de FA y Lenguajes RE

Leyes Algebraicas de las Expresiones Regulares

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

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

Construcción de la RE final

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

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

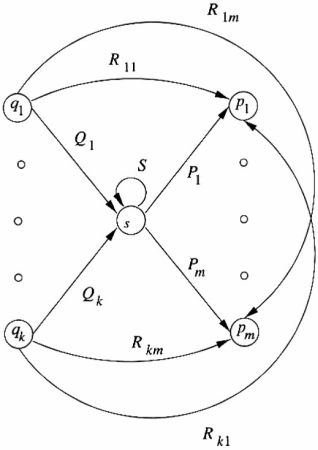
a) Dar todas las expresiones regulares para $R_{ij}^{(0)}$

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

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*".

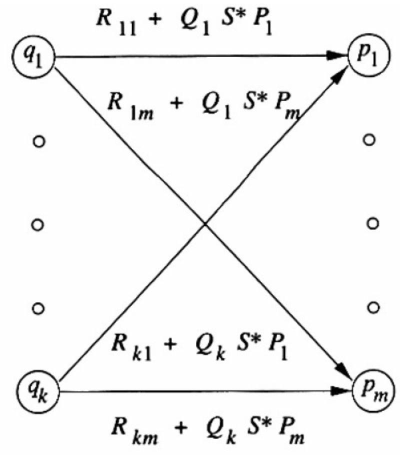
Conversión de un DFA a una RE por Eliminación de Estados



Conversión de un DFA a una RE por Eliminación de Estados

- 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:

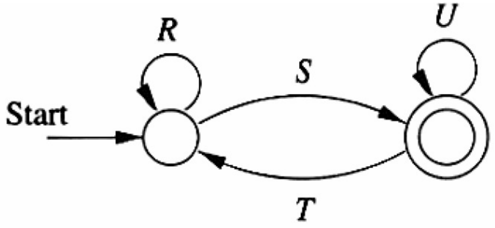
Conversión de un DFA a una RE por Eliminación de Estados



Estrategia para construir el autómata

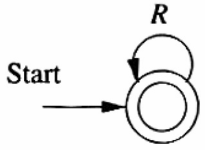
1. Para cada estado de aceptación q, aplicar el proceso de reducción para producir un autómata equivalente con RE como etiquetas en los arcos. Eliminar todos los estados excepto q y el estado inicial q_0.
2. Si $q \neq q_0$, se genera un autómata con 2 estados como el siguiente, una forma de describir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

Estrategia para construir el autómata



3. 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 el siguiente:

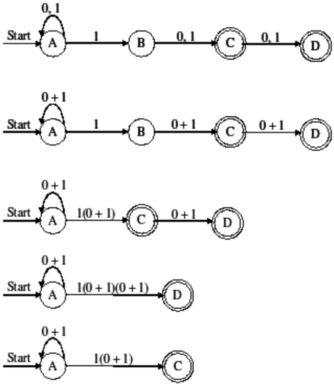
Estrategia para construir el autómata



4. 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.

Ejemplo

Para el siguiente NFA que acepta cadenas de 0's y 1's de manera que tienen un 1 dos o tres posiciones antes del final.



NFA Original

Autómata con RE como etiquetas

Eliminando B para evitar trabajo

RE eliminando C

RE eliminando D

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

RE final, suma de las RE anteriores que involucran el estado inicial y final

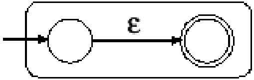
Convirtiendo una RE a un Autómata

Teorema 3.7: 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:

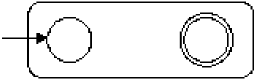
1. Exactamente un estado de aceptación
2. Sin arcos que lleguen al estado inicial
3. Sin arcos que salgan del estado de aceptación

Convirtiendo una RE a un Autómata

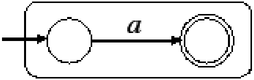
Base: cumpliendo las condiciones 1, 2, y 3.



El lenguaje es $\in$



El lenguaje es ϕ

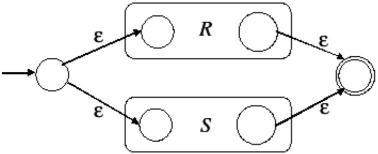


El lenguaje es la RE a, y sólo contiene la cadena a

Convirtiendo una RE a un Autómata

Inducción:

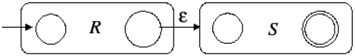
a) Para las expresiones $R + S$



Lenguaje: $L(R) \cup L(S)$

Convirtiendo una RE a un Autómata

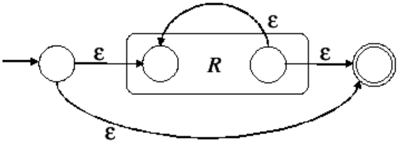
b) Para las expresiones *RS*



Lenguaje: $L(R)L(S)$

Convirtiendo una RE a un Autómata

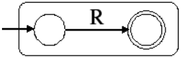
c) Para las expresiones R^*



El lenguaje es la R^*

Convirtiendo una RE a un Autómata

d) Para las expresiones (R)



El lenguaje es R ó (R)

Ejemplo

Convertir la RE $(0+1)^*1(0+1)$ a un ϵ-NFA.

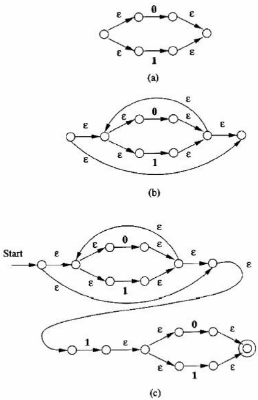


Figure 3.18: Automata constructed for Example 3.8

Ejemplo

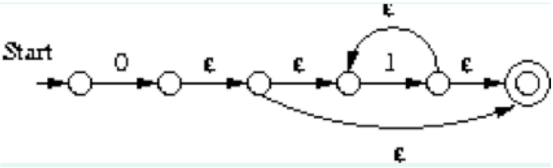
Dada la siguiente tabla de transición de un DFA:

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

- Dar las expresiones regulares para $R_{ij}^{(0)}$
- Dar las expresiones regulares para $R_{ij}^{(1)}$ y simplificar las expresiones
- Construir el diagrama de transición para el DFA y dar la expresión regular de su lenguaje eliminando el estado q_2

Ejemplo

Convierte la siguiente expresión regular a un NFA con transiciones ϵ: 01^*



Asociatividad y Conmutatividad

Existen un conjunto de leyes algebraícas 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 $LM \neq ML$

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.
- 0 es la identidad para la adición: $0 + x = x + 0 = x$.
- 1 es la identidad para la multiplicación: $1 \times x = x \times 1 = x$

Identidades y Aniquiladores

- 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

Identidades y Aniquiladores

- $\emptyset$ 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: $\emptyset L = L\emptyset = \emptyset$

NOTA: Estas leyes las utilizamos para hacer simplificaciones

Leyes Distributivas

- 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: $(M+N)L = ML + NL$

Ley de Idempotencia

- 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. Ley idempotente para la unión: $L + L = L$

Leyes que involucran la cerradura

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

Descubriendo leyes para RE

- Se puede proponer una variedad infinita de leyes para RE
- Se reduce a probar la igualdad de dos lenguajes específicos
- Ejemplo: probar que $(L + M)^* = (L^* M^*)^*$
- Para esto, probamos 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 $(L + M)^*$

Descubriendo leyes para RE

- 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: $(a^* b^*)^*$

Pasos de Prueba

Para probar una Ley Algebraica para una RE:
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: $L^* = L^*L^*$

Ejemplos

Probar que se cumple o no se sumple:

- $R + S = S + R$
- $(R^*)^* = R^*$
- $(R + S)^* = R^* + S^*$
- $S(RS + S)^* R = RR^* S(RR^* S)^*$

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