definición, no sólo describimos lo que son las expresiones regulares válidas, sino que para cada expresión regular E, describimos el lenguaje que representa, al que denominaremos $L(E)$.

BASE. El caso básico consta de tres partes:

1. Las constantes ε y $\emptyset$ son expresiones regulares, que representan a los lenguajes $\{\varepsilon\}$ y $\emptyset$, respectivamente. Es decir, $L(\varepsilon) = \{\varepsilon\}$ y $L(\emptyset) = \emptyset$.

2. Si a es cualquier símbolo, entonces **a** es una expresión regular. Esta expresión representa el lenguaje $\{a\}$. Es decir, $L(\mathbf{a}) = \{a\}$. Observe que utilizamos la fuente en negrita para indicar la expresión correspondiente a un símbolo. La correspondencia, por ejemplo, que **a** hace referencia a a, es obvia.

3. Una variable, normalmente escrita en mayúsculas e itálicas, como L, representa cualquier lenguaje.

PASO INDUCTIVO. Existen cuatro partes en el paso de inducción, una para cada uno de los tres operadores y otra para la introducción de paréntesis.

1. Si E y F son expresiones regulares, entonces $E + F$ es una expresión regular que representa la unión de $L(E)$ y $L(F)$. Es decir, $L(E + F) = L(E) \cup L(F)$.

2. Si E y F son expresiones regulares, entonces EF es una expresión regular que representa la concatenación de $L(E)$ y $L(F)$. Es decir, $L(EF) = L(E)L(F)$.

   Observe que el punto puede utilizarse opcionalmente para explicitar el operador de concatenación, bien como una operación sobre lenguajes o como el operador en una expresión regular. Por ejemplo, **0.1** es una expresión regular que significa lo mismo que **01** y que representa el lenguaje $\{01\}$. Sin embargo, nosotros vamos a evitar el uso del punto en la concatenación de expresiones regulares.[2]

3. Si E es una expresión regular, entonces E^* es una expresión regular, que representa la clausura de $L(E)$. Es decir, $L(E^*) = \big(L(E)\big)^*$.

4. Si E es una expresión regular, entonces (E), una E encerrada entre paréntesis, es también una expresión regular, que representa el mismo lenguaje que E. Formalmente; $L\big((E)\big) = L(E)$.

EJEMPLO 3.2

Escribamos una expresión regular para el conjunto de cadenas que constan de 0s y 1s alternos. Primero, desarrollamos una expresión regular para el lenguaje formado por una sola cadena 01. Podemos luego emplear el operador asterisco para obtener una expresión para todas las cadenas de la forma $0101\cdots01$.

La regla básica de las expresiones regulares nos dice que **0** y **1** son expresiones que representan los lenguajes $\{0\}$ y $\{1\}$, respectivamente. Si concatenamos las dos expresiones, obtenemos una expresión regular para el lenguaje $\{01\}$; esta expresión es **01**. Como regla general, si queremos una expresión regular para el lenguaje formado por una sola cadena w, utilizaremos la propia cadena w como expresión regular. Observe que en la expresión regular, los símbolos de w normalmente se escribirán en negrita, pero este cambio de tipo de fuente es sólo una ayuda para diferenciar las expresiones de las cadenas y no debe considerarse significativo.

Ahora obtenemos todas las cadenas formadas por cero o más ocurrencias de 01, utilizando la expresión regular $(\mathbf{01})^*$. Observe que primero hemos colocado los paréntesis alrededor de **01**, con el fin de evitar

[2] *De hecho, la expresiones regulares de UNIX utilizan el punto para un propósito completamente diferente: para representar cualquier* carácter ASCII.

Expresiones y sus lenguajes

Estrictamente hablando, una expresión regular E es sólo una expresión, no un lenguaje. Deberíamos emplear $L(E)$ cuando deseemos hacer referencia al lenguaje que E representa. Sin embargo, es habitual emplear "E" cuando realmente lo que se quiere decir es "$L(E)$". Utilizaremos este convenio siempre y cuando esté claro que estamos hablando de un lenguaje y no de una expresión regular.

confusiones con la expresión $\mathbf{01}^*$, cuyo lenguaje son todas las cadenas que constan de un 0 y un número cualquiera de 1s. La razón de esta interpretación se explica en la Sección 3.1.3, pero podemos adelantar que el operador asterisco precede al punto y que por tanto el argumento del asterisco se selecciona antes de realizar cualquier concatenación.

Sin embargo, $L((\mathbf{01})^*)$ no es exactamente el lenguaje que deseamos. Sólo incluye aquellas cadenas formadas por 0s y 1s alternos que comienzan por 0 y terminan por 1. También necesitamos considerar la posibilidad de que exista un 1 al principio y/0 un 0 al final de las cadenas. Un método sería construir tres expresiones regulares más que manejasen estas tres otras posibilidades. Es decir, $(\mathbf{10})^*$ representa las cadenas alternas que comienzan por 1 y terminan por 0, mientras que $\mathbf{0}(\mathbf{10})^*$ se puede emplear para las cadenas que comienzan y terminan por 0 y $\mathbf{1}(\mathbf{01})^*$ para las cadenas que comienzan y terminan por 1. La expresión regular completa es

$$(\mathbf{01})^* + (\mathbf{10})^* + \mathbf{0}(\mathbf{10})^* + \mathbf{1}(\mathbf{01})^*$$

Observe que utilizamos el operador + para obtener la unión de los cuatro lenguajes que nos proporcionan todas las cadenas con ceros y unos alternos.

Sin embargo, existe otra forma de obtener una expresión regular algo más sucinta. Partimos de nuevo de la expresión $(\mathbf{01})^*$. Podemos añadir un 1 opcional al principio si concatenamos por la izquierda la expresión $\varepsilon + \mathbf{1}$. Del mismo modo, añadimos un 0 opcional al final con la expresión $\varepsilon + \mathbf{0}$. Por ejemplo, empleando la definición del operador +:

$$L(\varepsilon + \mathbf{1}) = L(\varepsilon) \cup L(\mathbf{1}) = \{\varepsilon\} \cup \{1\} = \{\varepsilon, 1\}$$

Si concatenamos este lenguaje con cualquier otro lenguaje L, la opción ε nos proporciona todas las cadenas de L, mientras que la opción 1 nos proporciona $1w$ para todas las cadenas w de L. Por tanto, otra expresión para el conjunto de cadenas formadas por ceros y unos alternos es:

$$(\varepsilon + \mathbf{1})(\mathbf{01})^*(\varepsilon + \mathbf{0})$$

Observe que es necesario encerrar entre paréntesis cada una de las expresiones añadidas, con el fin de garantizar que los operadores se agrupan correctamente. □

3.1.3 Precedencia de los operadores en las expresiones regulares

Como con otras álgebras, los operadores de las expresiones regulares tienen un orden de "precedencia" prefijado, lo que significa que se asocian con sus operandos en un determinado orden. Estamos familiarizados con el concepto de precedencia en las expresiones aritméticas ordinarias. Por ejemplo, sabemos que en $xy + z$ primero se realiza el producto xy y luego la suma, y esto es equivalente a escribir la expresión empleando paréntesis así $(xy) + z$ y no de la forma $x(y + z)$. De forma similar, en aritmética, dos operadores iguales se agrupan comenzando por la izquierda, por lo que $x - y - z$ es equivalente a $(x - y) - z$, y no a $x - (y - z)$. En las expresiones regulares, el orden de precedencia de los operadores es el siguiente:

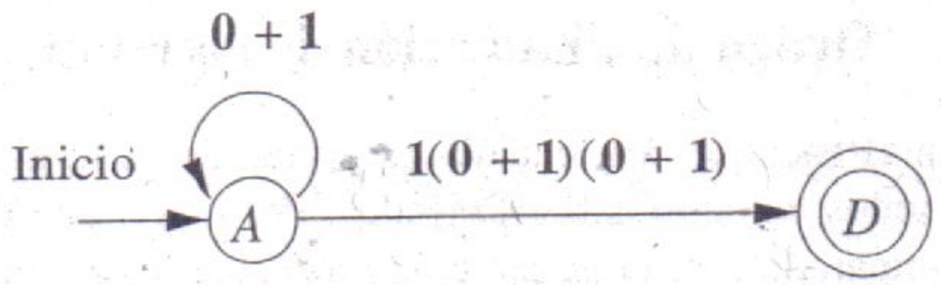


Figura 3.14. Autómata de dos estados con los estados A y D.

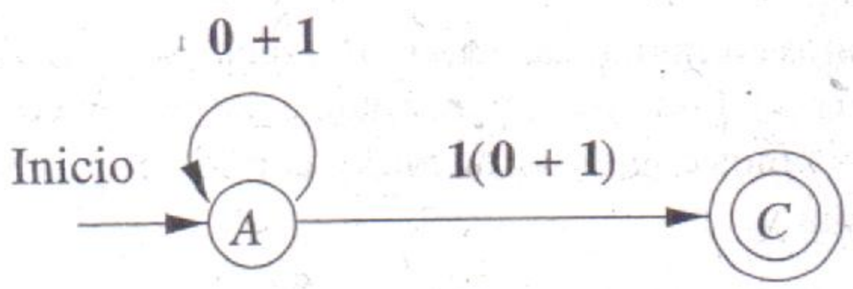


Figura 3.15. Autómata de dos estados resultado de la eliminación de D.

Ahora tenemos que volver de nuevo sobre la Figura 3.13 y eliminar el estado D en lugar del C. Puesto que D no tiene sucesores, una inspección de la Figura 3.7 nos indica que se producirán cambios en los arcos y que basta con eliminar el arco de C a D junto con el estado D. El autómata resultante de dos estados se muestra en la Figura 3.15.

Este autómata es muy parecido al de la Figura 3.14; sólo la etiqueta sobre el arco del estado inicial hasta el estado de aceptación es diferente. Por tanto, podemos aplicar la regla para el autómata de dos estados y simplificar la expresión para obtener $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$. Esta expresión representa el otro tipo de cadena que acepta el autómata: aquellas con un 1 en la segunda posición contando desde el final. Todo lo que queda es sumar las dos expresiones para obtener la expresión para el autómata completo de la Figura 3.11. Esta expresión es:

$$(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})+(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$$ □

3.2.3 Conversión de expresiones regulares en autómatas

Ahora vamos a completar el esquema de la Figura 3.1 demostrando que todo lenguaje L que es $L(R)$ para alguna expresión regular R, es también $L(E)$ para algún AFN-ε E. La demostración se realiza por inducción estructural sobre la expresión R. Comenzaremos mostrando cómo construir autómatas para las expresiones base: símbolos simples, ε y $\emptyset$. A continuación veremos cómo combinar estos autómatas en un autómata más grande que acepte la unión, la concatenación o la clausura del lenguaje aceptado por los autómatas más pequeños.

Todos los autómatas que vamos a construir son AFN-ε con un único estado de aceptación.

TEOREMA 3.7

Todo lenguaje definido mediante una expresión regular también puede definirse mediante un autómata finito.

DEMOSTRACIÓN. Suponga $L = L(R)$ para una expresión regular R. Vamos a demostrar que $L = L(E)$ para un AFN-ε E con:

1. Exactamente un estado de aceptación.
2. Ningún arco que entre en el estado inicial.
3. Ningún arco que salga del estado de aceptación.

Orden de eliminación de los estados

Como hemos observado en el Ejemplo 3.6, cuando un estado no es ni el estado inicial ni un estado de aceptación, se elimina en todos los autómatas derivados. Por tanto, una de las ventajas del proceso de eliminación de estados comparado con la generación mecánica de expresiones regulares como la que hemos descrito en la Sección 3.2.1 es que podemos comenzar eliminando todos los estados que no son ni el inicial ni uno de aceptación. Basta con duplicar el esfuerzo de reducción cuando necesitemos eliminar algunos estados de aceptación.

Incluso entonces podremos combinar esfuerzos. Por ejemplo, si existen tres estados de aceptación p, q y r, podemos eliminar p y luego bien q o r, generando el autómata con los estados de aceptación r y q, respectivamente. Después podemos comenzar de nuevo con los tres estados de aceptación y eliminar q y r para obtener el autómata para p.

La demostración se realiza por inducción estructural sobre R, siguiendo la definición recursiva de las expresiones regulares que hemos proporcionado en la Sección 3.1.2.

BASE. Hay tres partes en el caso base, como se muestra en la Figura 3.16. En la parte (a) vemos cómo se maneja la expresión ε. Puede verse fácilmente que el lenguaje del autómata es $\{\varepsilon\}$, ya que el único camino desde el estado inicial a un estado de aceptación está etiquetado con ε. La parte (b) muestra la construcción de $\emptyset$. Claramente, no existen caminos desde el estado inicial al de aceptación, por lo que $\emptyset$ es el lenguaje de este autómata. Por último, la parte (c) proporciona el autómata que reconoce una expresión regular **a**. Evidentemente, el lenguaje de este autómata consta de una cadena a, que es también $L(\mathbf{a})$. Es fácil comprobar que todos estos autómatas satisfacen las condiciones (1), (2) y (3) de la hipótesis inductiva.

PASO INDUCTIVO. Las tres partes del paso inductivo se muestran en la Figura 3.17. Suponemos que el enunciado del teorema es verdadero para las subexpresiones inmediatas de una expresión regular dada; es decir, los lenguajes de estas subexpresiones también son los lenguajes de los AFN-ε con un único estado de aceptación. Los cuatro casos son:

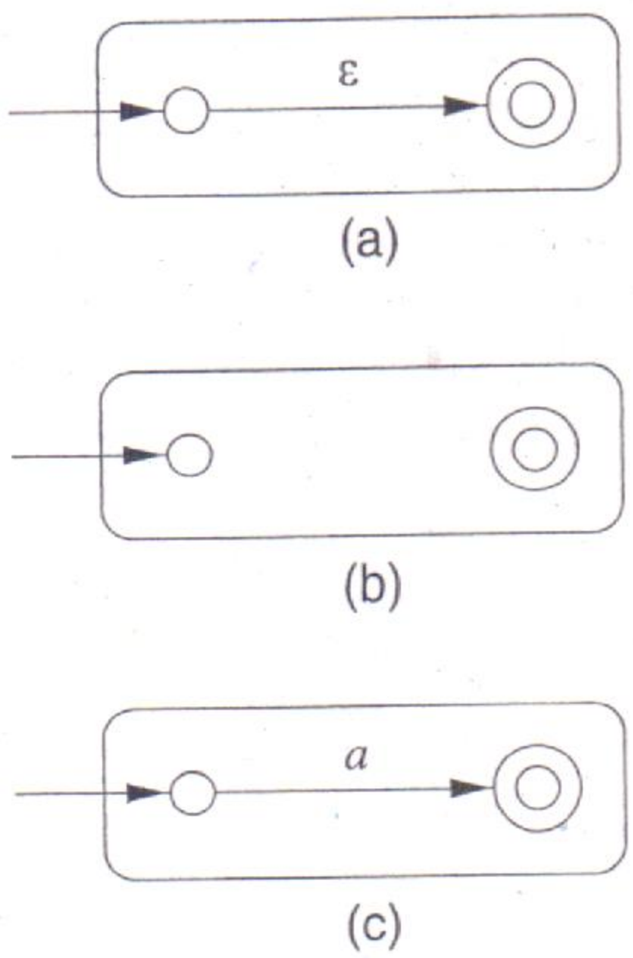
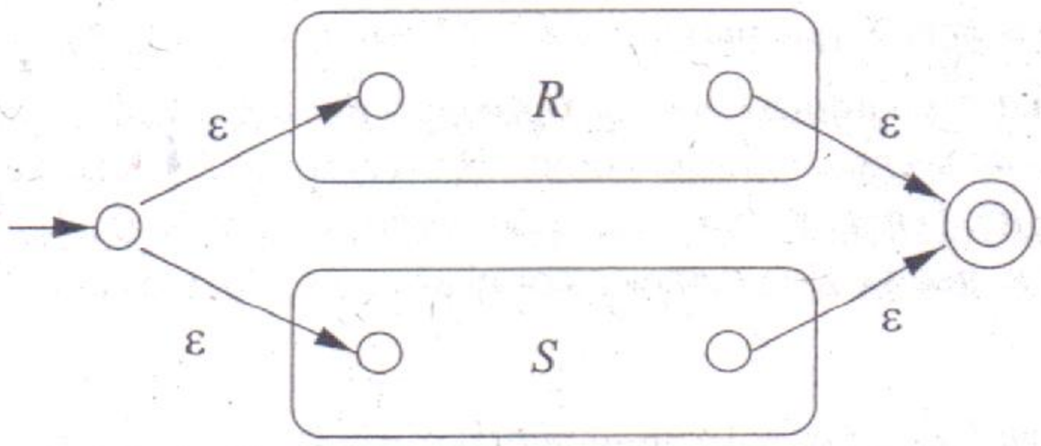
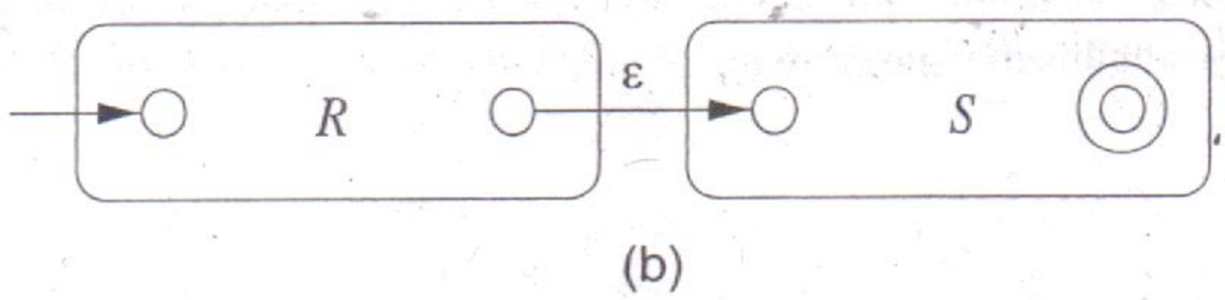


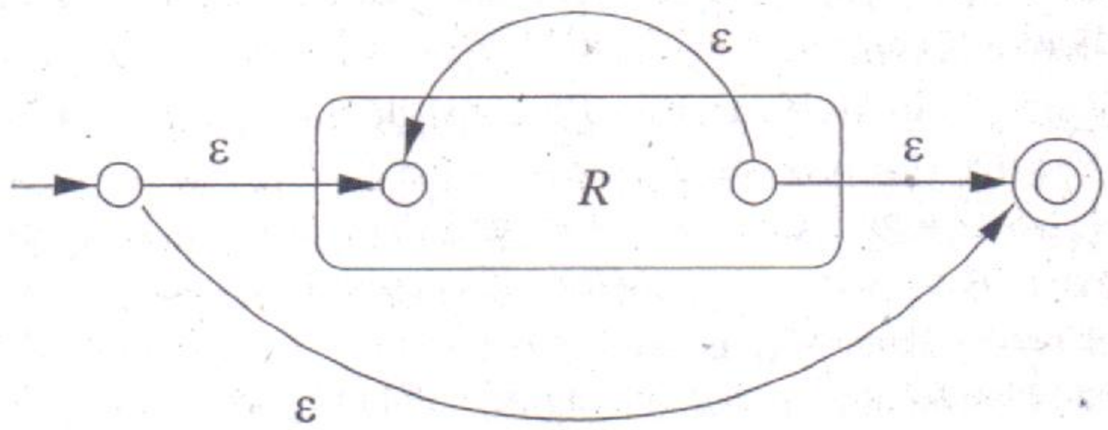
Figura 3.16. Casos básicos de la construcción de un autómata a partir de una expresión regular.



(a)



(b)



(c)

Figura 3.17. Paso inductivo en la construcción del AFN-ε a partir de una expresión regular.

1. La expresión es de la forma $R+S$ para dos expresiones R y S más pequeñas. Así podemos emplear el autómata de la Figura 3.17(a). Es decir, partiendo del nuevo estado inicial, podemos llegar al estado inicial del autómata correspondiente a R o del autómata correspondiente a S. A continuación alcanzaremos el estado de aceptación de uno de estos autómatas, siguiendo un camino etiquetado por alguna cadena de $L(R)$ o $L(S)$, respectivamente. Una vez alcanzado el estado de aceptación del autómata correspondiente a R o S, podemos seguir uno de los arcos-ε hasta el estado de aceptación del nuevo autómata. Por tanto, el lenguaje del autómata de la Figura 3.17(a) es $L(R) \cup L(S)$.
2. La expresión es de la forma RS para expresiones R y S más pequeñas. El autómata para la concatenación se muestra en la Figura 3.17(b). Observe que el estado inicial del primer autómata se convierte en el estado inicial del conjunto y que el estado de aceptación del segundo autómata se convierte en el estado de aceptación del conjunto. La idea es que los únicos caminos desde el estado inicial hasta el de aceptación pasan primero a través del autómata para R, en el que debe seguir un camino etiquetado con una cadena perteneciente a $L(R)$, y luego a través del autómata para S, donde sigue un camino etiquetado con una cadena perteneciente a $L(S)$. Por tanto, los caminos en el autómata de la Figura 3.17(b) son todos y sólo los etiquetados con cadenas pertenecientes a $L(R)L(S)$.
3. La expresión es de la forma R^* para una expresión R más pequeña. Utilizamos el autómata de la Figura 3.17(c). Dicho autómata nos permite ir:

a) Directamente desde el estado inicial al estado de aceptación siguiendo un camino etiquetado con ε. Dicho camino acepta ε, que pertenece a $L(R^*)$ sin importar qué expresión sea R.

b) Al estado inicial del autómata correspondiente a R, atravesando dicho autómata una o más veces, y luego al estado de aceptación. Este conjunto de caminos nos permite aceptar cadenas pertenecientes a $L(R)$, $L(R)L(R)$, $L(R)L(R)L(R)$, etc., cubriendo por tanto todas las cadenas pertenecientes a $L(R^*)$ excepto quizá ε, que ha sido cubierta por el arco directo al estado de aceptación mencionado en el apartado (3a).

4. La expresión es de la forma (R) para alguna expresión R más pequeña. El autómata correspondiente a R también sirve para reconocer el autómata correspondiente a (R), ya que los paréntesis no cambian el lenguaje definido por la expresión.

Observe que el autómata construido satisface las tres condiciones dadas en las hipótesis inductivas (un único estado de aceptación y ningún arco que entre en el estado inicial o que salga del estado de aceptación). □

EJEMPLO 3.8

Deseamos convertir la expresión regular $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$ en un AFN-ε. El primer paso consiste en construir un autómata para $\mathbf{0}+\mathbf{1}$. Utilizamos los dos autómatas construidos de acuerdo con la Figura 3.16(c), uno con la etiqueta **0** sobre el arco y otro con la etiqueta **1**. Estos dos autómatas se combinan entonces utilizando la construcción correspondiente a la unión de la Figura 3.17(a). El resultado se muestra en la Figura 3.18(a).

A continuación aplicamos a la Figura 3.18(a) la construcción inicial de la Figura 3.17(c). Este autómata se muestra en la Figura 3.18(b). Los dos últimos pasos implican aplicar la construcción de la concatenación de la Figura 3.17(b). En primer lugar, conectamos el autómata de la Figura 3.18(b) a otro autómata diseñado para aceptar sólo la cadena 1. Este autómata es otra aplicación de la construcción básica de la Figura 3.16(c) con la etiqueta **1** sobre el arco. Observe que tenemos que crear un autómata *nuevo* para reconocer el 1; no debemos emplear para esto el autómata que formaba parte de la Figura 3.18(a). El tercer autómata en el proceso de concatenación es otro autómata para $\mathbf{0}+\mathbf{1}$. De nuevo, tenemos que crear una copia del autómata de la Figura 3.18(a); no debemos emplear la misma copia que forma parte de la Figura 3.18(b). El autómata completo se muestra en la Figura 3.18(c). Fíjese en que este AFN-ε, cuando se eliminan las transiciones-ε, se parece al autómata mucho más simple de la Figura 3.15 que también acepta las cadenas que contienen un 1 en la penúltima posición. □

3.2.4 Ejercicios de la Sección 3.2

Ejercicio 3.2.1. He aquí una tabla de transiciones para un AFD:

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

* a) Obtenga todas las expresiones regulares $R_{ij}^{(0)}$. *Nota*: piense en el estado q_i como si fuera el estado asociado al número entero i.

* b) Obtenga todas las expresiones regulares $R_{ij}^{(1)}$. Intente simplificar las expresiones lo máximo posible.