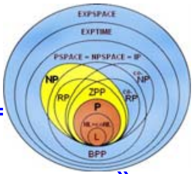
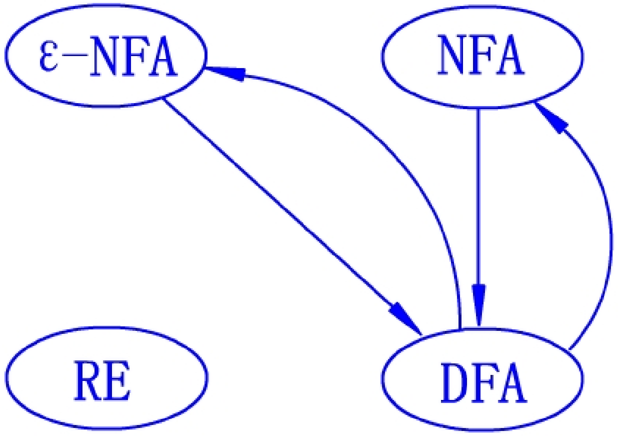


Session 4

- Finite Automata and Regular Expressions
- Algebraic Lows for Regular Expressions

Finite Automata and Regular Expressions

We have already shown that DFA's, NFA's and ϵ-NFA's all are equivalent. That is, they accept the same class of languages. In fact, finite-automaton and regular-expression represent *exactly* the same set of languages – "regular languages".


$$\begin{tikzcd}
\varepsilon\text{-NFA} \arrow[dr] & \text{NFA} \arrow[d] \\
\text{RE} & \text{DFA} \arrow[ul, bend right] \arrow[u, bend right]
\end{tikzcd}$$

To show FA's is equivalent RE's we need establish that

1. For every DFA A we can find a regular expression R such that $L(R) = L(A)$.
2. For every regular expression R there is a ϵ-NFA A such that $L(A) = L(R)$.

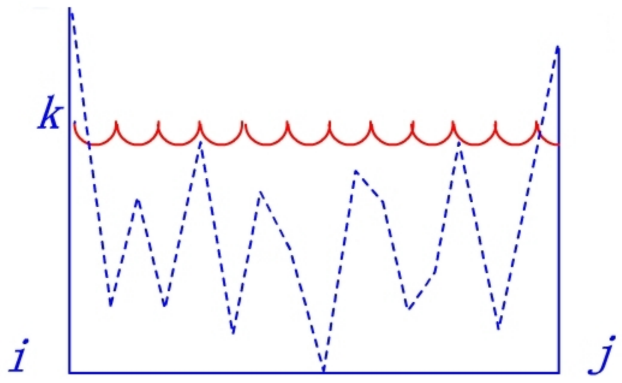
$$\begin{tikzcd}
\varepsilon\text{-NFA} \arrow[dr] & \text{NFA} \arrow[d] \\
\text{RE} \arrow[u] & \text{DFA} \arrow[l] \arrow[ul, bend right] \arrow[u, bend right]
\end{tikzcd}$$

From DFA's to RE's

Theorem 3.1 *For every DFA $A = (Q, \Sigma, \delta, q_0, F)$ there is a regular expression R such that $L(R) = L(A)$.*

Proof Let the states of A be $\{1, 2, \cdots, n\}$ for some integer n, with 1 being the start state. Let $R_{ij}^{(k)}$ be a regular expression describing the set of labels of all path in A from state i to state j going through intermediate states $\{1, 2, \cdots, k\}$ only.

Here is the figure suggested the requirement on the paths represented by $R_{ij}^{(k)}$.



$R_{ij}^{(k)}$ will be defined inductively. Note that $L\left(\sum_{j\in F} R_{1j}^{(n)}\right) = L(A)$.

Basis step: $k = 0$, i.e. no intermediate states.

- Case 1: $i \neq j$

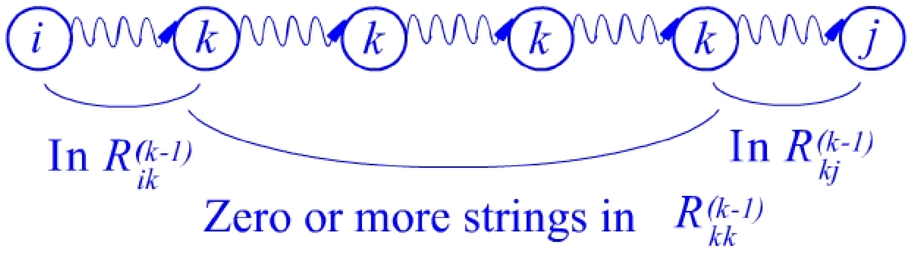
$$R_{ij}^{(0)} = \sum_{\{a \in \Sigma \mid \delta(i,a)=j\}} \mathbf{a}$$

- Case 2: $i = j$

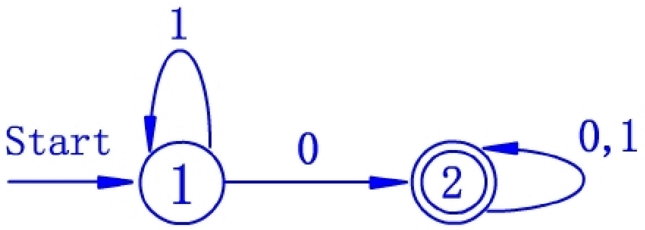
$$R_{ii}^{(0)} = \left(\sum_{\{a \in \Sigma \mid \delta(i,a)=i\}} \mathbf{a} \right) + \epsilon$$

Inductive step:

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$



Example Let's find regular expression R for DFA A where



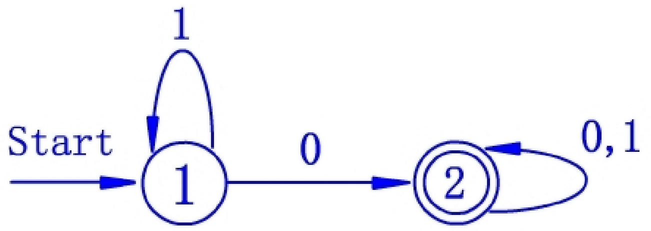
Clearly,

$$L(A) = \{x0y \mid x \in \{1\}^* \text{ and } y \in \{0, 1\}^*\}$$

Now, we use Theorem 3.1 to construct regular expression R.

The basis expression in the construction is

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



For building the expression in induction part, we will need the following simplification rules.

- $(\epsilon + R)^* = R^*$
- $R + RS^* = RS^*$
- $\emptyset R = R\emptyset = \emptyset$ (Annihilation)
- $\emptyset + R = R + \emptyset = R$ (Identity)

Computing expression $R_{ij}^{(1)}$

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

$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)} \left(R_{11}^{(0)}\right)^* R_{1j}^{(0)}$$

	By directive substitution	Simplified
$R_{11}^{(1)}$	$\epsilon + \mathbf{1} + (\epsilon + \mathbf{1})(\epsilon + \mathbf{1})^*(\epsilon + \mathbf{1})$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0} + (\epsilon + \mathbf{1})(\epsilon + \mathbf{1})^*\mathbf{0}$	$\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\epsilon + \mathbf{1})^*(\epsilon + \mathbf{1})$	$\emptyset$
$R_{22}^{(1)}$	$\epsilon + \mathbf{0} + \mathbf{1} + \emptyset(\epsilon + \mathbf{1})^*\mathbf{0}$	$\epsilon + \mathbf{0} + \mathbf{1}$

Computing expression $R_{ij}^{(2)}$

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

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

	By directive substitution	Simplified
$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^*\mathbf{0}(\epsilon + \mathbf{0} + \mathbf{1})^*\emptyset$	$\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0} + \mathbf{1}^*\mathbf{0}(\epsilon + \mathbf{0} + \mathbf{1})^*(\epsilon + \mathbf{0} + \mathbf{1})$	$\mathbf{1}^*\mathbf{0}(\mathbf{0} + \mathbf{1})^*$
$R_{21}^{(2)}$	$\emptyset + (\epsilon + \mathbf{0} + \mathbf{1})(\epsilon + \mathbf{0} + \mathbf{1})^*\emptyset$	$\emptyset$
$R_{22}^{(2)}$	$\epsilon + \mathbf{0} + \mathbf{1} + (\epsilon + \mathbf{0} + \mathbf{1})(\epsilon + \mathbf{0} + \mathbf{1})^*(\epsilon + \mathbf{0} + \mathbf{1})$	$(\mathbf{0} + \mathbf{1})^*$

The final regular expression for A is $R = R_{12}^{(2)} = \mathbf{1}^*\mathbf{0}(\mathbf{0} + \mathbf{1})^*$.

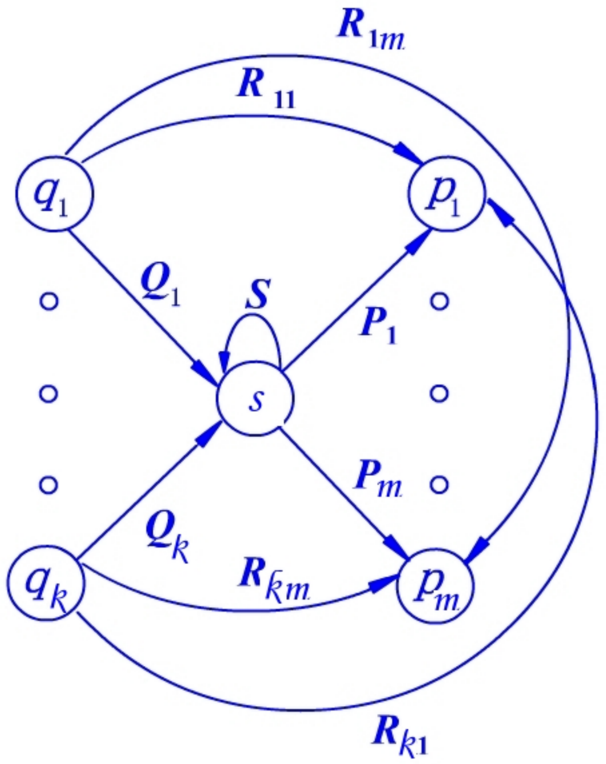
The State Elimination Technique

The method of last section for converting a DFA to a regular expression always works. But such construction is too expensive!

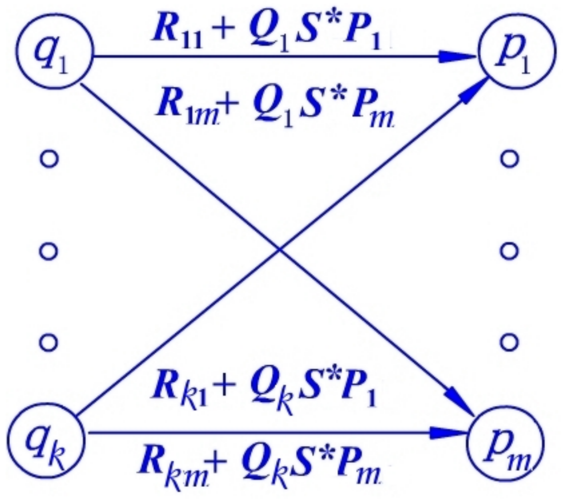
- There are n^3 expression $R_{ij}^{(k)}$.
- Each inductive step grows the expression 4-fold, $R_{ij}^{(n)}$ could have size 4^n.
- For all $\{i, j\} \subset \{1, 2, \cdots, n\}$, $R_{ij}^{(k)}$ uses $R_{kk}^{(k-1)}$, so we have to write n^2 times $R_{kk}^{(k-1)}$.

We need a more efficient approach.

Let's label the edges with regular expression instead symbols.



Now, let's eliminate state *s*.

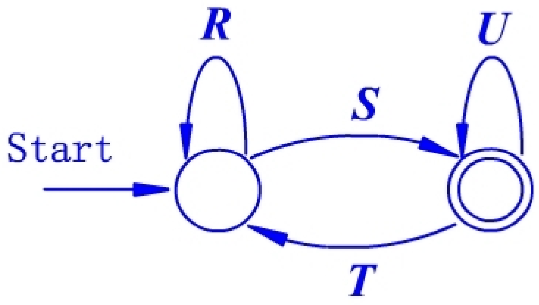


Such approach is called the state elimination technique.

The strategy for constructing regular expression from FA is as follows:

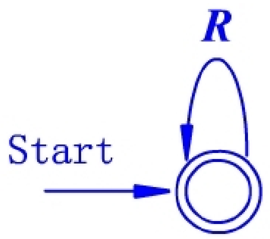
For each accepting state q eliminate from the original automaton all states except q_0 and q. For each $q \in F$, we will be left with an A_q

- If A_q looks like



then the corresponding regular expression is $E_q = (R + SU^*T)^*SU^*$.

- If A_q looks like

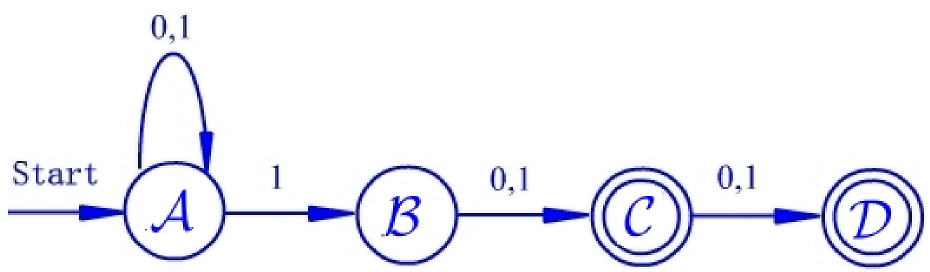


then the corresponding regular expression is $E_q = R^*$.

The final regular expression is

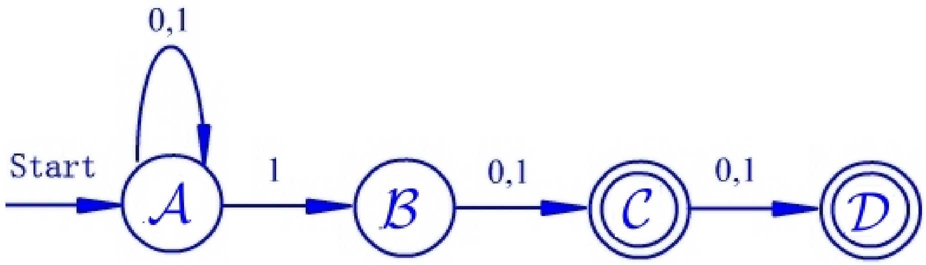
$$\sum_{q \in F} E_q.$$

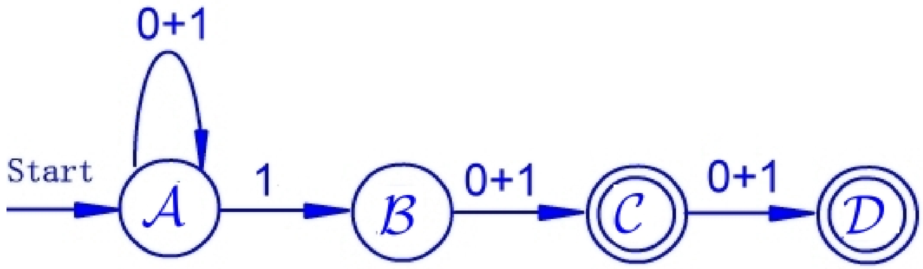
Example Convert NFA A to regular expression by the state elimination technique.



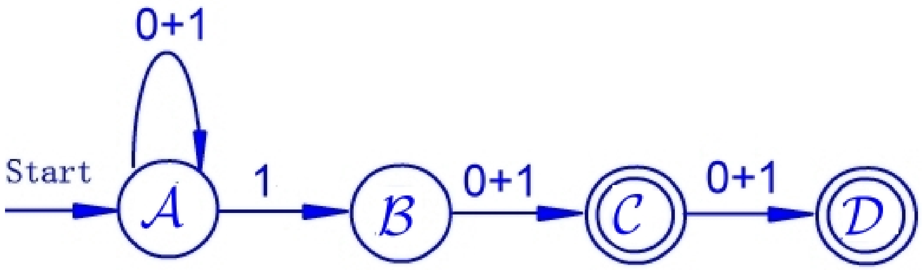
$$L(A) = \{w \mid w = x1b \text{ or } w = x1bc,\ \ x \in \{0,1\}^*,\ \ b, c \in \{0,1\}\}$$

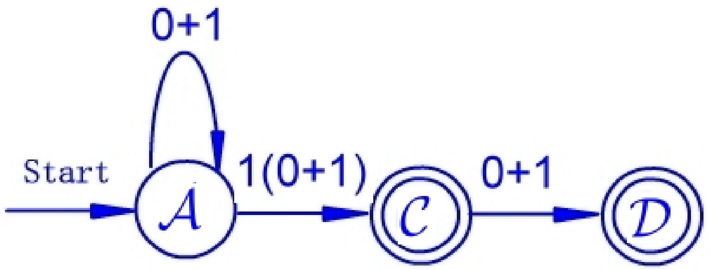
We turn this into an automaton with regular expression labels.



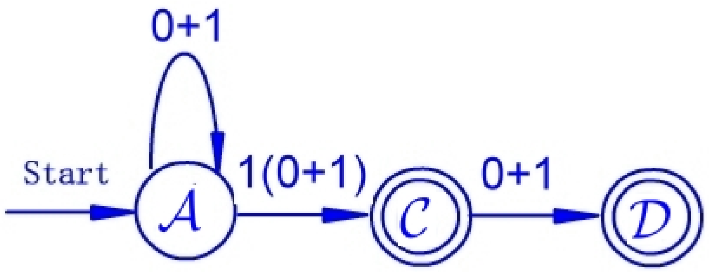


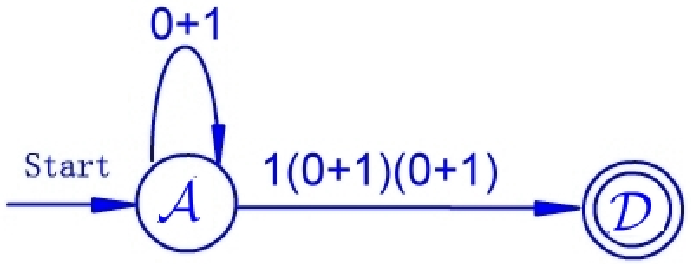
Let’s eliminate state *B*.





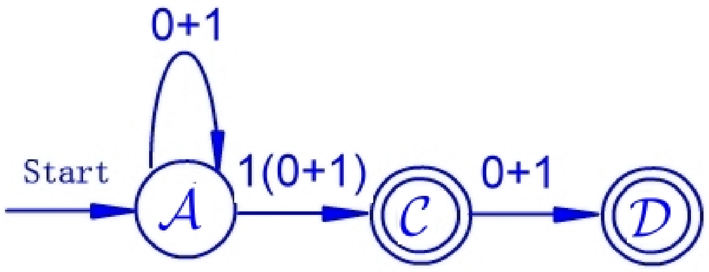
Then we eliminate state C and obtain A_D.

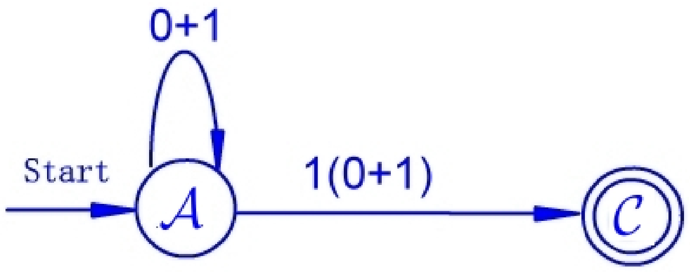




with regular expression $(\mathbf{0} + \mathbf{1})^*\mathbf{1}(\mathbf{0} + \mathbf{1})(\mathbf{0} + \mathbf{1})$.

Also we can eliminate state D and obtain A_C.

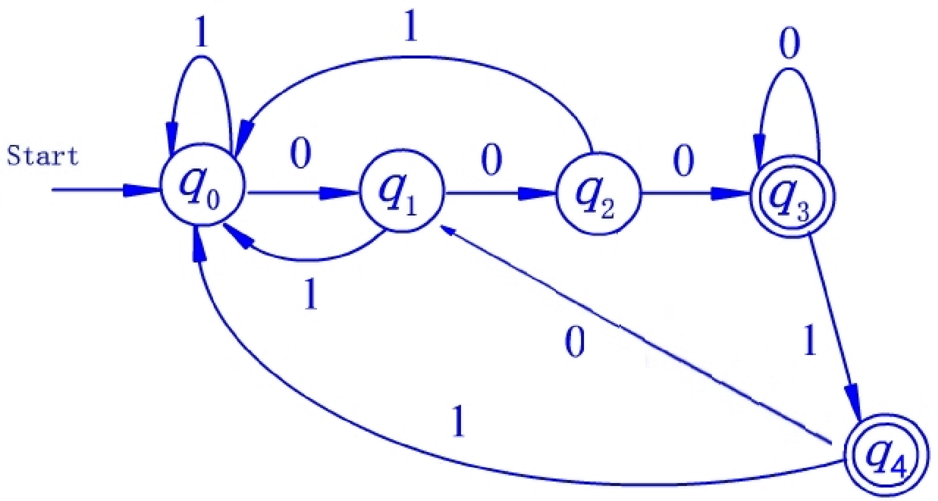




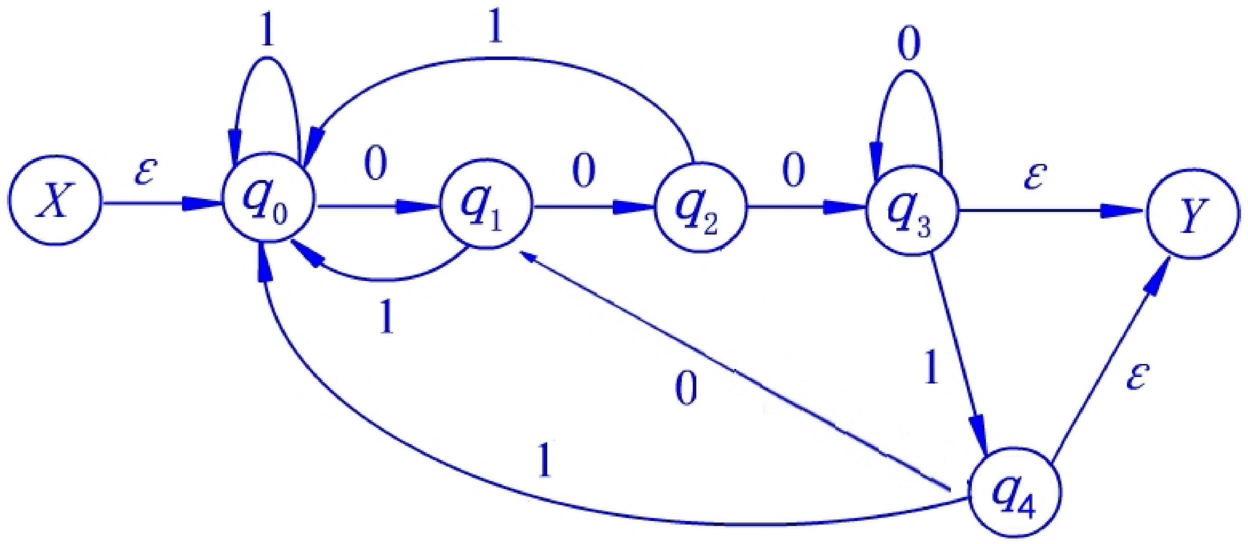
with regular expression $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$.

The final expression is $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})+(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$.

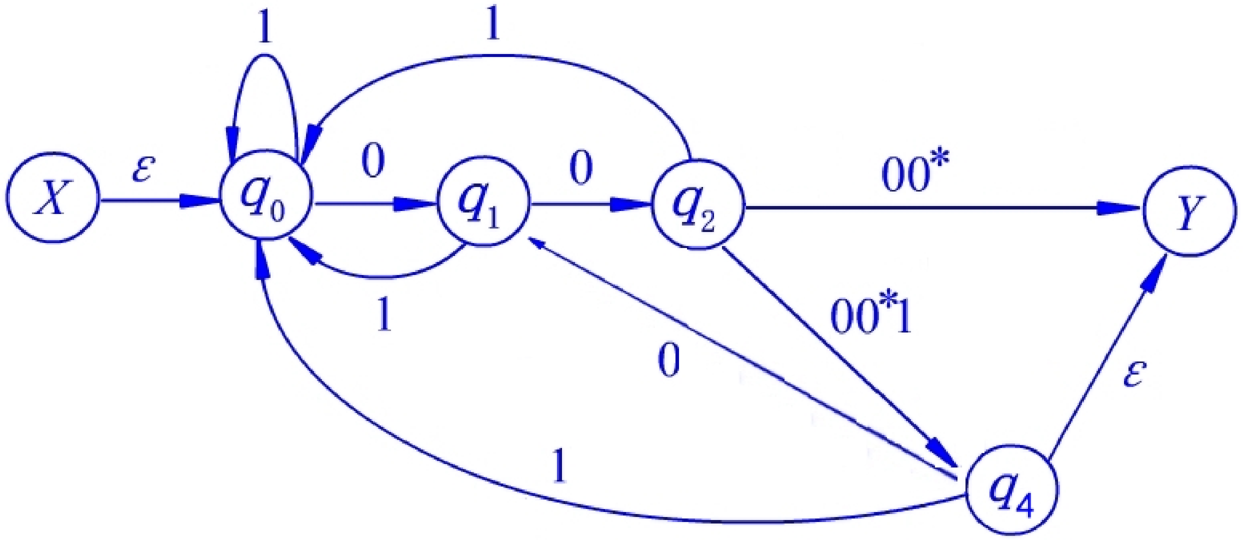
Example Construct regular expression from DFA A.



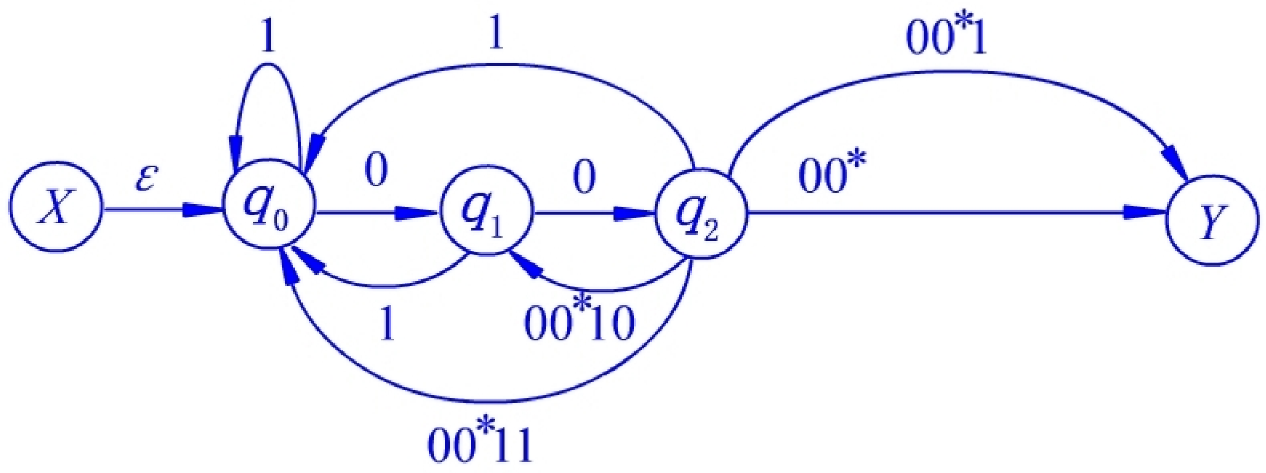
Label X and Y states as start and final states, respectively.



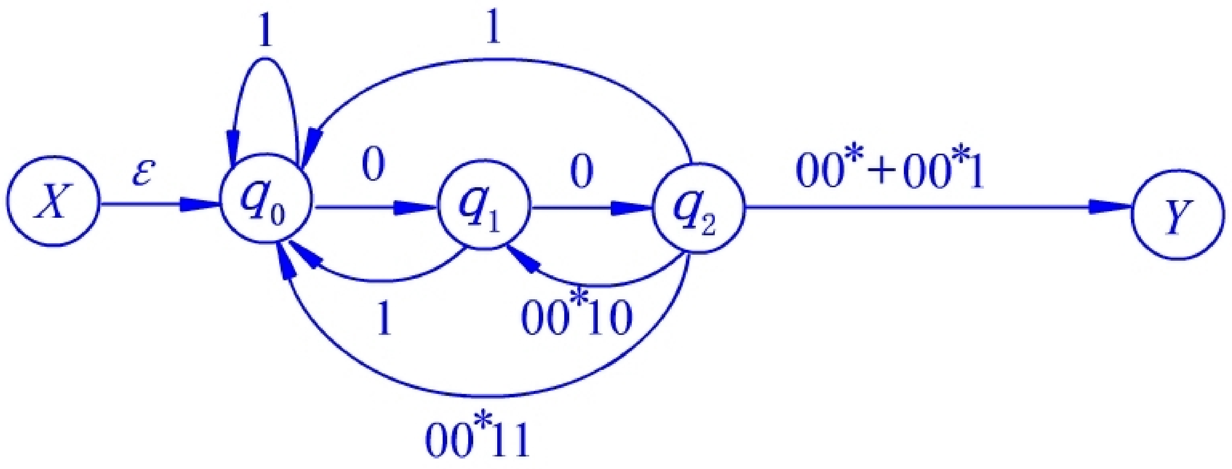
Eliminate state q_3.



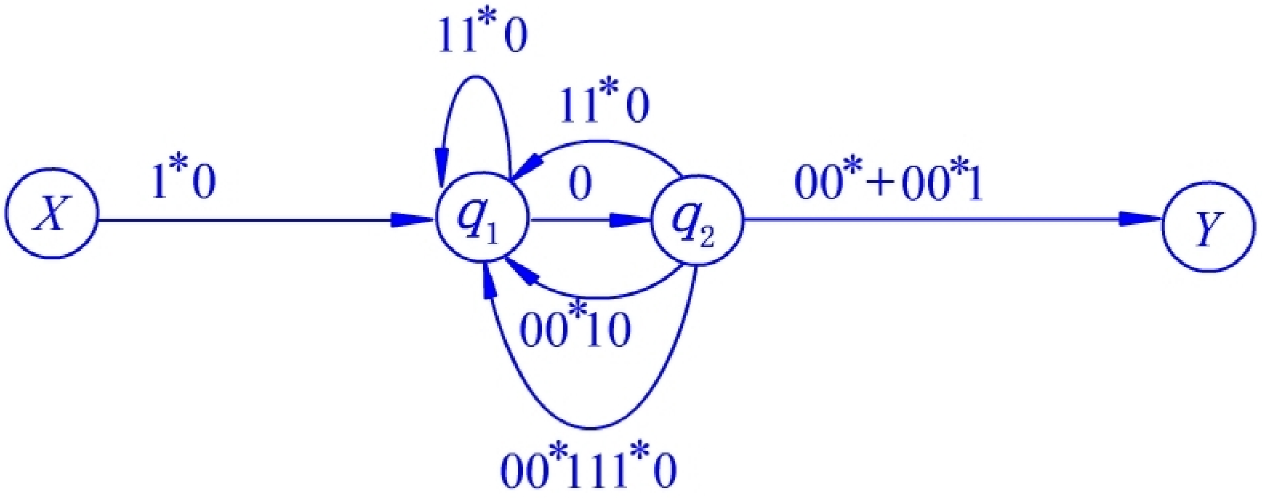
Eliminate state q_4.



Combine.



Eliminate state q_0.



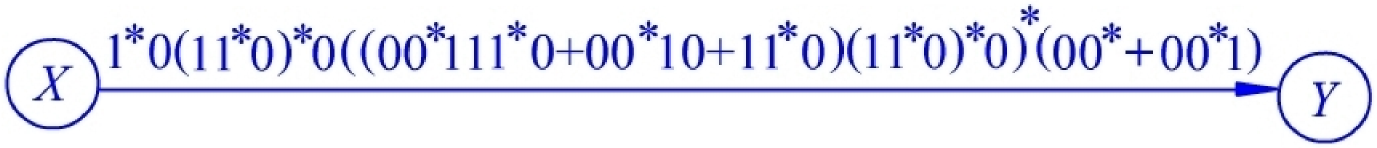
Combine.

$$\begin{tikzcd}[column sep=huge]
X \arrow[r, "1^*0"] & q_1 \arrow[loop above, "11^*0"] \arrow[r, "0"] & q_2 \arrow[l, bend left=60, "00^*111^*0+00^*10+11^*0"] \arrow[r, "00^*+00^*1"] & Y
\end{tikzcd}$$

Eliminate state q_1.

$$\begin{tikzcd}[column sep=huge]
X \arrow[r, "1^*0(11^*0)^*0"] & q_2 \arrow[loop above, "(00^*111^*0+00^*10+11^*0)(11^*0)^*0"] \arrow[r, "00^*+00^*1"] & Y
\end{tikzcd}$$

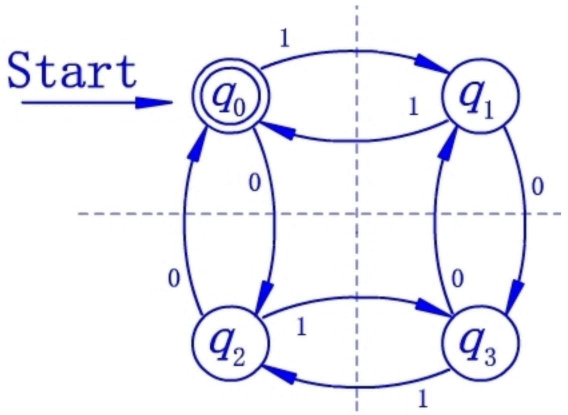
Eliminate state q_2.



The result is

$$\mathbf{1^*0(11^*0)^*0[(00^*111^*0 + 00^*10 + 11^*0)(11^*0)^*0]^*(00^* + 00^*1)}$$

Question Find a regular expression for the language accepted by DFA



The final expression is long and complicated, but the way to get it is relatively straightforward.

There are an unlimited number of regular expressions for any given language!

From RE's to ϵ-NFA's

Theorem 3.2 *For every regular expression R we can construct an ϵ-NFA A such that $L(R) = L(A)$.*

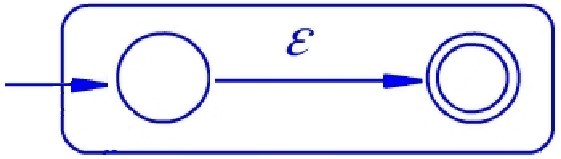
Proof The proof is by structural induction on R. We will construct ϵ-NFA A with:

1. Exactly one accepting state;

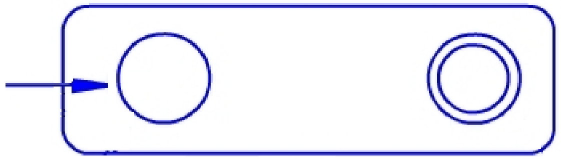
2. No arcs into the initial state;

3. No arcs out of the accepting state.

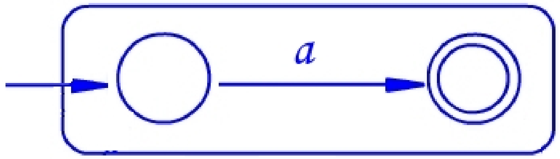
Basis step: Automata for ϵ, $\emptyset$, $\mathbf{a}$.



(a)

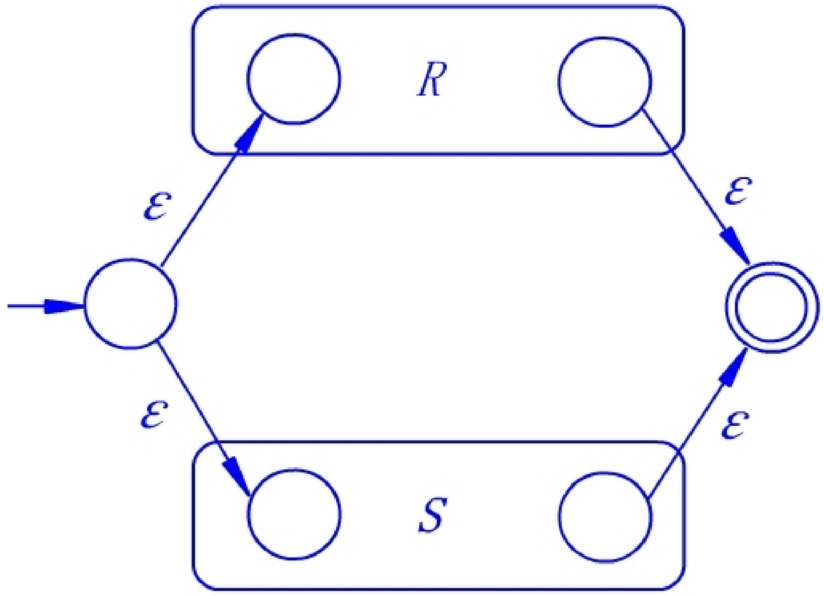


(b)

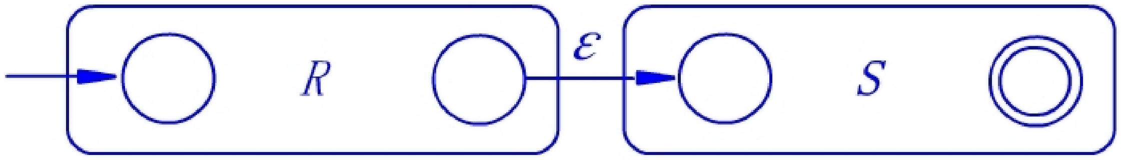


(c)

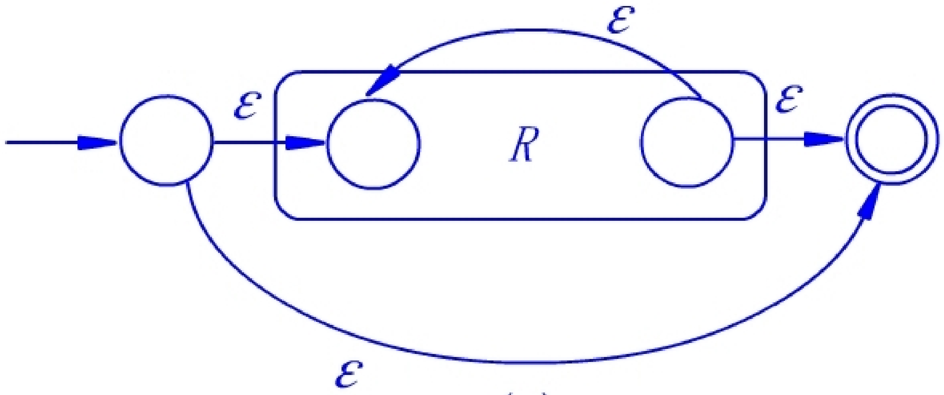
Inductive step: Automata for $R+S$, $R.S$, R^*.



(a)

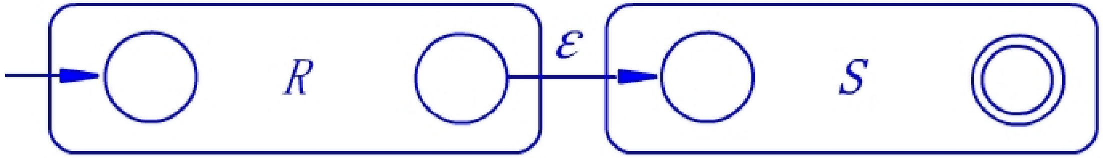


(b)



(c)

A formal proof for inductive step (b):

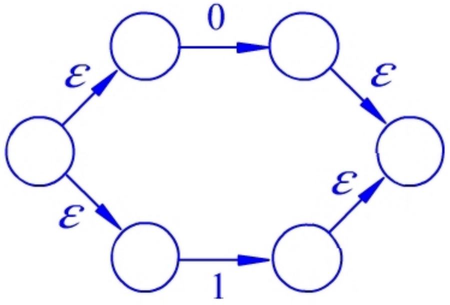


Let $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $L(R)$ and $L(S)$, respectively. Construct $A = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$ recognize $L(R.S)$, where

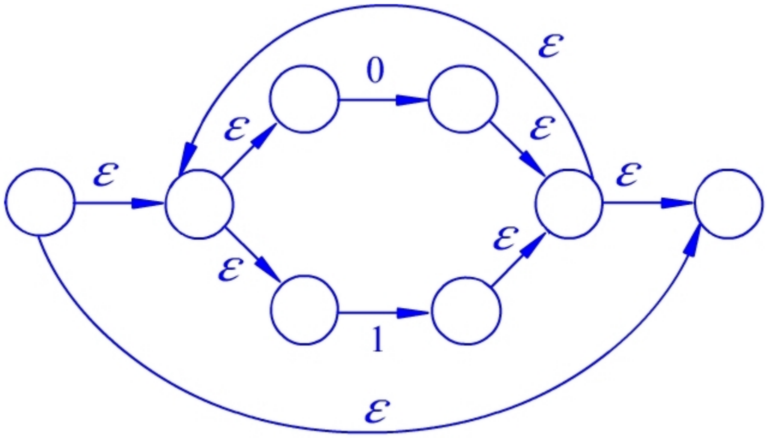
$$\delta(q,a) = \begin{cases} \delta_1(q,a), & q \in Q_1, q \notin F_1 \\ \delta_1(q,a), & q \in F_1, a \neq \epsilon \\ \delta_1(q,a) \cup q_2, & q \in F_1, a = \epsilon \\ \delta_2(q,a), & q \in Q_2. \end{cases}$$

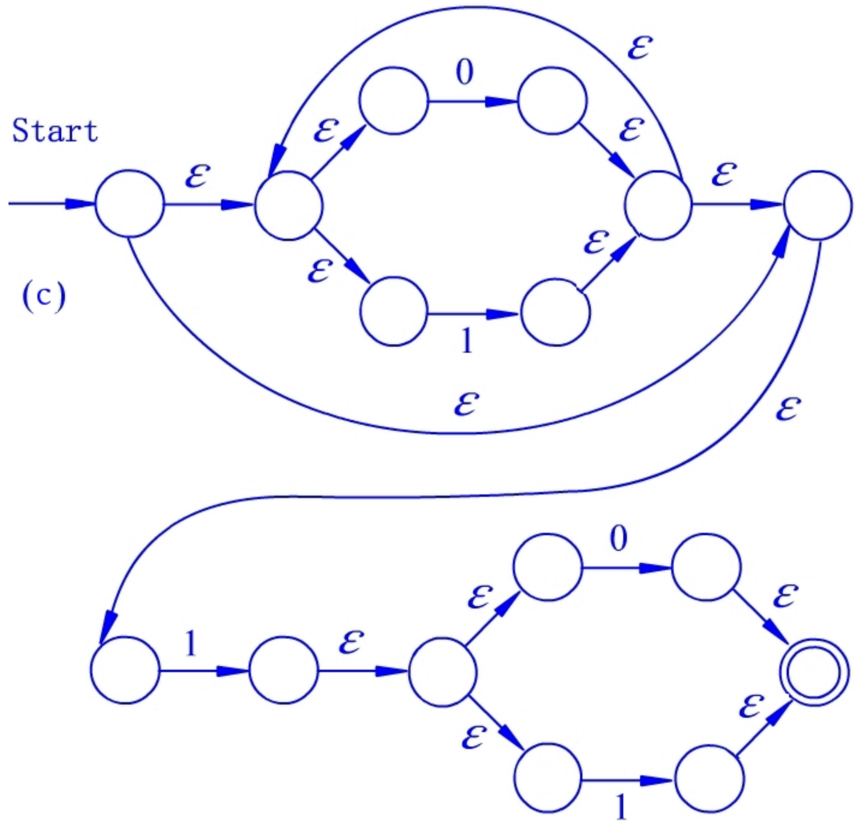
Example Convert $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$ to an ϵ-NFA.

(a)



(b)





Start
ε
ε
0
ε
ε
ε
ε
ε
1
(c)
ε
ε
1
ε
ε
0
ε
ε
ε
1

Break for 15 minutes

Algebraic Lows for Regular Expressions

Algebraic Lows for Regular Expressions

Regular expressions obey many of the algebraic laws of arithmetic. For example, we have the following identities:

- $R + S = S + R, \quad (R + S) + T = R + (S + T), \quad (RS)T = R(ST),$
- $R(S + T) = RS + RT, \quad (R + S)T = RT + ST, \quad \cdots,$
- $(R^*)^* = R^*, \quad (\epsilon + R)^* = R^*, \quad (R^*S^*)^* = (R + S)^*.$

That is, two regular expression denote the *same* language!

The Test for a Regular-Expressions Algebraic Law

Evidently, we have

$$L((\mathbf{0}+\mathbf{1})\mathbf{1}) = L(\mathbf{01}+\mathbf{11}),$$

and also, we have

$$L((\mathbf{00}+\mathbf{101})\mathbf{11}) = L(\mathbf{0011}+\mathbf{10111}).$$

In fact, for any concrete regular expression E, F, G, we can test (automatically)

$$L((E+F)G) = L(EG+FG).$$

In conclusion, we obtain a *general* identity

$$(\mathscr{E} + \mathscr{F})\mathscr{G} = \mathscr{E}\mathscr{G} + \mathscr{F}\mathscr{G}$$

for any regular expression $\mathscr{E}$, $\mathscr{F}$ and $\mathscr{G}$.

How do we verify that a general identity like above is true? Our method is

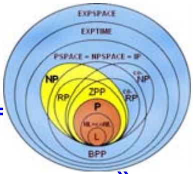
- "Freeze" $\mathscr{E}$, $\mathscr{F}$ and $\mathscr{G}$ to **a**, **b** and **c**.
- Test if the frozen identity is true, e.g. if $(\mathbf{a} + \mathbf{b})\mathbf{c} = \mathbf{ac} + \mathbf{bc}$, i.e. $L((\mathbf{a} + \mathbf{b})\mathbf{c}) = L(\mathbf{ac} + \mathbf{bc})$.

Example To prove $(\mathscr{E} + \mathscr{F})^* = (\mathscr{E}^*\mathscr{F}^*)^*$ it is enough to determine if $(\mathbf{a} + \mathbf{b})^*$ is equivalent to $(\mathbf{a}^*\mathbf{b}^*)^*$.

It is easy to check that both expressions $(\mathbf{a} + \mathbf{b})^*$ and $(\mathbf{a}^*\mathbf{b}^*)^*$ denote the same language with all strings of a's and b's. That is $L((\mathbf{a} + \mathbf{b})^*) = L((\mathbf{a}^*\mathbf{b}^*)^*)$, and the law holds.

Question: Does this always work?

Answer: Yes, as long as the identity use only **plus, dot** and **star**.



Let's denote a generalized regular expression, such as $(\mathscr{E} + \mathscr{F})^*$, by $\mathbf{E}(\mathscr{E}, \mathscr{F})$. We can for instance make the substitution $\mathbb{S} = \{\mathscr{E}/\mathbf{0}, \mathscr{F}/\mathbf{1}\}$ to obtain

$$\mathbb{S}(\mathbf{E}(\mathscr{E}, \mathscr{F})) = (\mathbf{0} + \mathbf{1})^*.$$

Theorem 3.3 *Fix a "freezing" substitution* $\mathbb{S} = \{\mathscr{E}_1/\mathbf{a}_1, \mathscr{E}_2/\mathbf{a}_2, \cdots, \mathscr{E}_m/\mathbf{a}_m\}$. *Let* $\mathbf{E}(\mathscr{E}_1, \mathscr{E}_2, \cdots, \mathscr{E}_m)$ *be a generalized regular expression. Then for any regular expression* $E_1, E_2, \cdots, E_m$, $w \in L(\mathbf{E}(E_1, E_2, \cdots, E_m))$ *if and only if there are strings* $w_i \in L(E_i)$ *s.t.* $w = w_{j_1} w_{j_2} \cdots w_{j_k}$ and $x = a_{j_1} a_{j_2} \cdots a_{j_k} \in L(\mathbf{E}(\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m))$.

For example, let $\mathbf{E}(\mathscr{E}_1, \mathscr{E}_2)$ be $(\mathscr{E}_1 + \mathscr{E}_2)\mathscr{E}_1$, then $\mathbb{S}(\mathbf{E}(\mathscr{E}_1, \mathscr{E}_2)) = (\mathbf{a}_1 + \mathbf{a}_2)\mathbf{a}_1$.

Let $E_1 = \mathbf{01}$, $E_2 = \mathbf{1}^*\mathbf{0}$. The Theorem 3.3 tells us

$$w \in L(\mathbf{E}(E_1, E_2)) = L((\mathbf{01} + \mathbf{1}^*\mathbf{0})\mathbf{01}) = (\{01\} \cup \{1^*0\})\{01\} = \{0101, 1^*001\}$$

iff $\exists w_1 \in L(E_1) = \{01\}$, $\exists w_2 \in L(E_2) = \{1^*0\}$ such that $w = w_{j_1}w_{j_2}$ and $x = a_{j_1}a_{j_2} \in L(\mathbf{E}(\mathbf{a}_1, \mathbf{a}_2)) = \{a_1a_1, a_2a_1\}$. However

$$x = a_{j_1}a_{j_2} \in \{a_1a_1, a_2a_1\} \quad \text{iff} \quad j_1 = j_2 = 1; \; j_1 = 2, j_2 = 1.$$

That means $w = w_1w_1 = 0101$ or $w = w_2w_1 = 1^*001$.

☞ Less formally, for any regular expression $E_1, E_2, \cdots E_m$, we can construct $L(\mathbf{E}(E_1, E_2, \cdots, E_m))$ by starting with each string x in $L(\mathbf{E}(\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m))$, say, $x = a_{j_1}a_{j_2}\cdots a_{j_k}$, and substituting for each the a_{j_i}'s any string from the corresponding language $L(E_{j_i})$.

Proof For proving the Theorem 3.3, we do a structural induction on **E**.

Basis step:

- If $\mathbf{E} = \epsilon$, the frozen expression is also ϵ.
- If $\mathbf{E} = \emptyset$, the frozen expression is also $\emptyset$.
- If $\mathbf{E} = \mathbf{a}$, the frozen expression is also $\mathbf{a}$.

  Now $w \in L(\mathbf{E})$ if an only if there is $u \in L(\mathbf{a})$, such that $w = u$ and u is in language of the frozen expression, i.e. $u \in \{a\}$.

Inductive step:

- Suppose $\mathbf{E}(\mathscr{E}_1, \cdots, \mathscr{E}_m) = \mathbf{F}(\mathscr{E}_1, \cdots, \mathscr{E}_m) + \mathbf{G}(\mathscr{E}_1, \cdots, \mathscr{E}_m)$, we have

$$\mathbb{S}(\mathbf{E}(\mathscr{E}_1, \cdots, \mathscr{E}_m)) = \mathbb{S}(\mathbf{F}(\mathscr{E}_1, \cdots, \mathscr{E}_m)) + \mathbb{S}(\mathbf{G}(\mathscr{E}_1, \cdots, \mathscr{E}_m)).$$

For any regular expression $E_1, E_2, \cdots, E_m$,

$$w \in L(\mathbf{E}(E_1, \cdots, E_m)) = L(\mathbf{F}(E_1, \cdots, E_m)) \cup L(\mathbf{G}(E_1, \cdots, E_m))$$

iff $\quad w \in L(\mathbf{F}(E_1, \cdots, E_m))$ or $w \in L(\mathbf{G}(E_1, \cdots, E_m))$

iff $\quad \exists w_i \in L(E_i)$, s.t. $w = w_{j_1} \cdots w_{j_k}$ and $x = a_{j_1} \cdots a_{j_k} \in L(\mathbf{F}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$ or $x = a_{j_1} \cdots a_{j_k} \in L(\mathbf{G}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$, i.e. $x \in L(\mathbf{E}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$.

- Suppose $\mathbf{E}(\mathscr{E}_1,\cdots,\mathscr{E}_m)=\mathbf{F}(\mathscr{E}_1,\cdots,\mathscr{E}_m)\mathbf{G}(\mathscr{E}_1,\cdots,\mathscr{E}_m)$, we have

$$\mathbb{S}(\mathbf{E}(\mathscr{E}_1,\cdots,\mathscr{E}_m))=\mathbb{S}(\mathbf{F}(\mathscr{E}_1,\cdots,\mathscr{E}_m))\mathbb{S}(\mathbf{G}(\mathscr{E}_1,\cdots,\mathscr{E}_m)).$$

For any regular expression $E_1,E_2,\cdots,E_m$,

$$w\in L(\mathbf{E}(E_1,\cdots,E_m))=L(\mathbf{F}(E_1,\cdots,E_m)).L(\mathbf{G}(E_1,\cdots,E_m))$$

iff $\quad w=uv,\ \ u\in L(\mathbf{F}(E_1,\cdots,E_m)),\ \ v\in L(\mathbf{G}(E_1,\cdots,E_m))$

iff $\quad \exists\, u_i,v_i\in L(E_i)$, s.t. $u=u_{j_1}\cdots u_{j_k}$, $v=v_{j_1}\cdots v_{j_l}$, and $y=a_{j_1}\cdots a_{j_k}\in L(\mathbf{F}(\mathbf{a}_1,\cdots,\mathbf{a}_m))$, and $z=b_{j_1}\cdots b_{j_l}\in L(\mathbf{G}(\mathbf{a}_1,\cdots,\mathbf{a}_m))$

iff $\quad x=yz\in L(\mathbf{E}(\mathbf{a}_1,\cdots,\mathbf{a}_m))$.

- Suppose $\mathbf{E}(\mathscr{E}_1, \cdots, \mathscr{E}_m) = \mathbf{F}(\mathscr{E}_1, \cdots, \mathscr{E}_m)^*$, we have

$$\mathbb{S}(\mathbf{E}(\mathscr{E}_1, \cdots, \mathscr{E}_m)) = \mathbb{S}(\mathbf{F}(\mathscr{E}_1, \cdots, \mathscr{E}_m))^*.$$

For any regular expression $E_1, E_2, \cdots, E_m$,

$$w \in L(\mathbf{E}(E_1, \cdots, E_m)) = L(\mathbf{F}(E_1, \cdots, E_m))^*$$

iff $\quad w = u \cdots v, \;\; u, v \in L(\mathbf{F}(E_1, \cdots, E_m))$

iff $\quad \exists\, u_i, v_i \in L(E_i)$, s.t. $u = u_{j_1} \cdots u_{j_k}$, $v = v_{j_1} \cdots v_{j_l}$, and $y = a_{j_1} \cdots a_{j_k} \in L(\mathbf{F}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$, and $z = b_{j_1} \cdots b_{j_l} \in L(\mathbf{F}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$

iff $\quad x = y \cdots z \in L(\mathbf{E}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$. ◀

Theorem 3.4 *Fix a "freezing" substitution* $\mathbb{S} = \{\mathscr{E}_1/\mathbf{a}_1, \mathscr{E}_2/\mathbf{a}_2, \cdots, \mathscr{E}_m/\mathbf{a}_m\}$,

$$\mathbf{E}(\mathscr{E}_1, \mathscr{E}_2, \cdots, \mathscr{E}_m) = \mathbf{F}(\mathscr{E}_1, \mathscr{E}_2, \cdots, \mathscr{E}_m)$$

iff $$L(\mathbf{E}(\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m)) = L(\mathbf{F}(\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m)).$$

Proof (Only if)

$\mathbf{E}(\mathscr{E}_1, \cdots, \mathscr{E}_m) = \mathbf{F}(\mathscr{E}_1, \cdots, \mathscr{E}_m)$ means $L(\mathbf{E}(E_1, \cdots, E_m)) = L(\mathbf{F}(E_1, \cdots, E_m))$ for any concrete regular expression $E_1, E_2, \cdots, E_m$.

In particularly then $L(\mathbf{E}(\mathbf{a}_1, \cdots, \mathbf{a}_m)) = L(\mathbf{F}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$.

(If) Let $E_1, E_2, \cdots, E_m$ be concrete regular expressions, we want to show

$$L(\mathbf{E}(E_1, \cdots, E_m)) = L(\mathbf{F}(E_1, \cdots, E_m)).$$

By theorem 3.3,

$$w \in L(\mathbf{E}(E_1, \cdots, E_m))$$

iff $\exists\, w_i \in L(E_i)$, s.t. $w = w_{j_1} \cdots w_{j_k}$ and $x = a_{j_1} \cdots a_{j_k} \in L(\mathbf{E}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$

iff $\exists\, w_i \in L(E_i)$, s.t. $w = w_{j_1} \cdots w_{j_k}$ and $x = a_{j_1} \cdots a_{j_k} \in L(\mathbf{F}(\mathbf{a}_1, \cdots, \mathbf{a}_m))$

iff $w \in L(\mathbf{F}(E_1, \cdots, E_m)).$ ◄

Example Since $L(\mathbf{a})^* = L(\mathbf{a})^*.L(\mathbf{a})^*$, so for any general regular expression $\mathscr{E}$, we have

$$\mathscr{E}^* = \mathscr{E}^*\mathscr{E}^*.$$

Question Does

$$\mathscr{E} + \mathscr{F}\mathscr{E} = (\mathscr{E} + \mathscr{F})\mathscr{E}$$

hold? How about the following statement?

$$(\mathscr{R}\mathscr{S} + \mathscr{R})^*\mathscr{R}\mathscr{S} = (\mathscr{R}\mathscr{R}^*\mathscr{S})^*$$

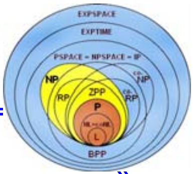
Replace $\mathscr{R}$ by **a** and $\mathscr{S}$ by **b**. We need to prove or disprove

$$L((\mathbf{ab}+\mathbf{a})^*\mathbf{ab}) = L((\mathbf{aa}^*\mathbf{b})^*).$$

In fact, the language of right side consists of all strings composed of zero or more occurrences of strings of the form $a\cdots ab$, that is, one or more a's ended by one b.

However, every string in the language of the left side has to end in ab. Thus, for instance, ϵ is in the language on the right, but not on the left.

So the answer is "Not the same."


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
co-NP
RP
ZPP
co-RP
P
L
BPP

Thank you!