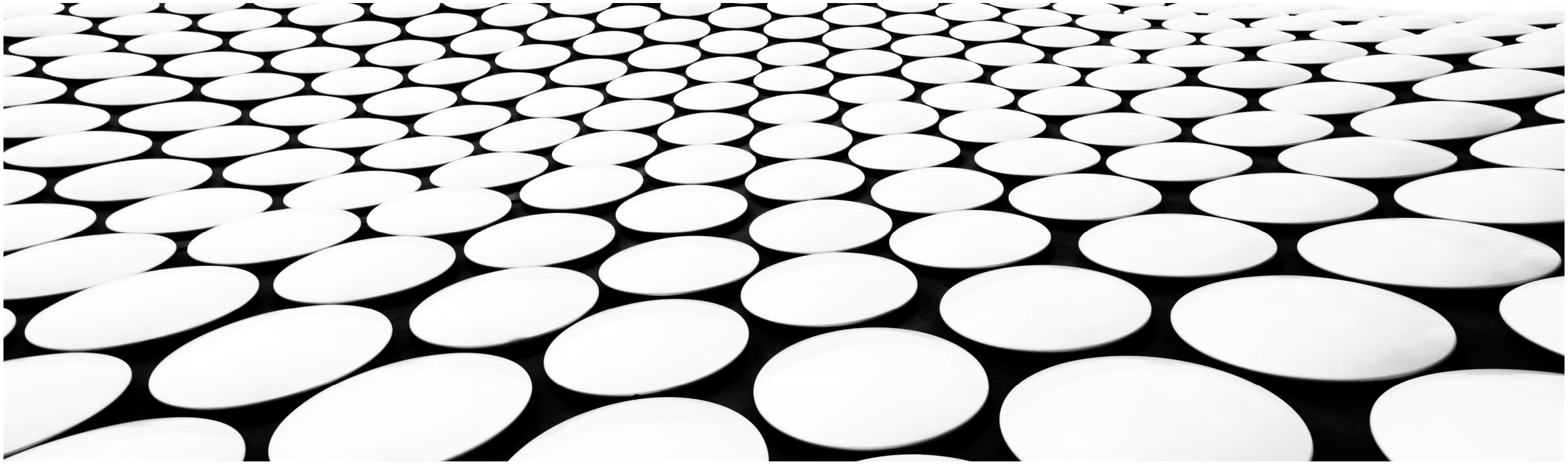
REGULAR EXPRESSIONS AND CONVERSIONS

IF 2124 TEORI BAHASA FORMAL OTOMATA

Judhi S.



Regular expressions

A FA (NFA or DFA) is a "blueprint" for contructing a machine recognizing a regular language.

A *regular expression* is a "user-friendly," declarative way of describing a regular language.

Example: $\mathbf{01^* + 10^*}$

Regular expressions are used in e.g.

1. UNIX `grep` command

2. UNIX Lex (Lexical analyzer generator) and Flex (Fast Lex) tools.

Operations on languages

Union:

$L \cup M = \{w : w \in L \text{ or } w \in M\}$

Concatenation:

$L.M = \{w : w = xy, x \in L, y \in M\}$

Powers:

$L^0 = \{\epsilon\},\ L^1 = L,\ L^{k+1} = L.L^k$

Kleene Closure:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Question: What are $\emptyset^0$, $\emptyset^i$, and $\emptyset^*$

Building regex's

Inductive definition of regex's:

Basis: $\boldsymbol{\epsilon}$ is a regex and $\boldsymbol{\emptyset}$ is a regex.
$L(\boldsymbol{\epsilon}) = \{\epsilon\}$, and $L(\boldsymbol{\emptyset}) = \emptyset$.

If $a \in \Sigma$, then $\boldsymbol{a}$ is a regex.
$L(\boldsymbol{a}) = \{a\}$.

Induction:

If E is a regex's, then (E) is a regex.
$L((E)) = L(E)$.

If E and F are regex's, then $E + F$ is a regex.
$L(E + F) = L(E) \cup L(F)$.

If E and F are regex's, then $E.F$ is a regex.
$L(E.F) = L(E).L(F)$.

If E is a regex's, then $E^{\star}$ is a regex.
$L(E^{\star}) = (L(E))^{*}$.

Example: Regex for

$L = \{w \in \{0,1\}^* :$ 0 and 1 alternate in $w\}$

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

or, equivalently,

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

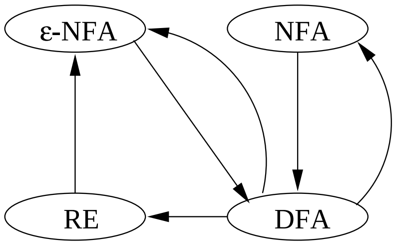
Order of precedence for operators:

1. Star
2. Dot
3. Plus

Example: $\mathbf{01^* + 1}$ is grouped $\mathbf{(0(1)^*) + 1}$

Equivalence of FA's and regex's

We have already shown that DFA's, NFA's, and ϵ-NFA's all are equivalent.



To show FA's equivalent to regex's we need to establish that

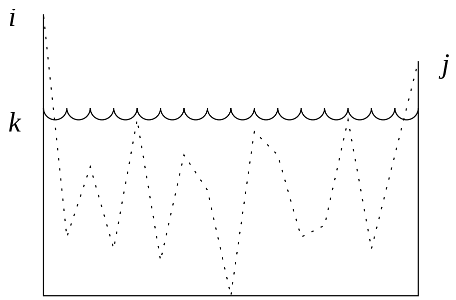
1. For every DFA A we can find (construct, in this case) a regex R, s.t. $L(R) = L(A)$.

2. For every regex R there is a ϵ-NFA A, s.t. $L(A) = L(R)$.

Theorem 3.4: For every DFA $A = (Q, \Sigma, \delta, q_0, F)$ there is a regex R, s.t. $L(R) = L(A)$.

Proof: Let the states of A be $\{1, 2, \ldots, n\}$, with 1 being the start state.

• Let $R_{ij}^{(k)}$ be a regex describing the set of labels of all paths in A from state i to state j going through intermediate states $\{1, \ldots, k\}$ only.



$R_{ij}^{(k)}$ will be defined inductively. Note that

$$L\left(\bigoplus_{j\in F} {R_{1j}}^{(n)}\right) = L(A)$$

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

- *Case 1:* $i \neq j$

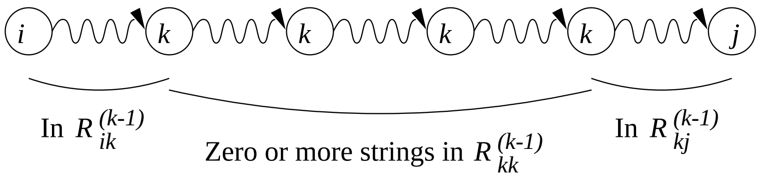
$$R_{ij}^{(0)} = \bigoplus_{\{a\in\Sigma:\delta(i,a)=j\}} \boldsymbol{a}$$

- *Case 2:* $i = j$

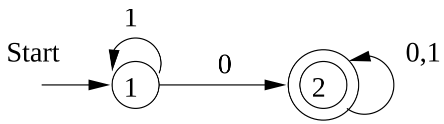
$$R_{ii}^{(0)} = \left(\bigoplus_{\{a\in\Sigma:\delta(i,a)=i\}} \boldsymbol{a}\right) + \boldsymbol{\epsilon}$$

Induction:

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



Example: Let's find R for A, where
$L(A) = \{x0y : x \in \{1\}^* \text{ and } y \in \{0,1\}^*\}$



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

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)

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

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

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

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

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

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

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

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

The final regex for A is

$$R_{12}^{(2)} = \mathbf{1^*0(0 + 1)^*}$$

Observations

There are n^3 expressions $R_{ij}^{(k)}$

Each inductive step grows the expression 4-fold

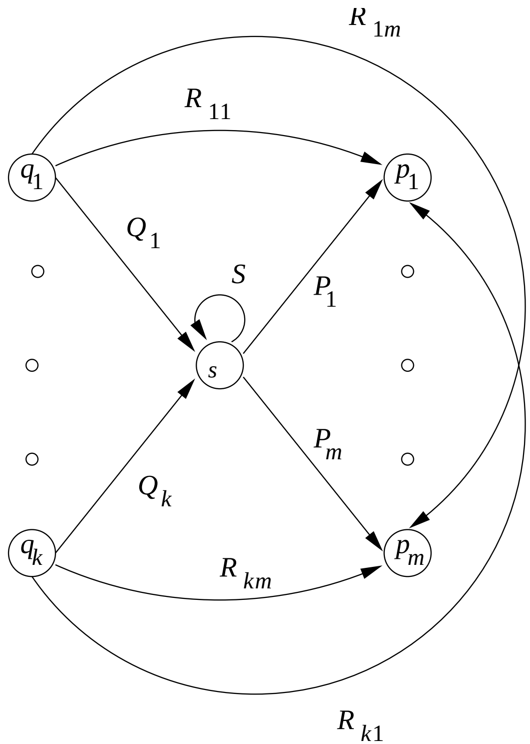
$R_{ij}^{(n)}$ could have size 4^n

For all $\{i, j\} \subseteq \{1, \ldots, n\}$, $R_{ij}^{(k)}$ uses $R_{kk}^{(k-1)}$
so we have to write n^2 times the regex $R_{kk}^{(k-1)}$

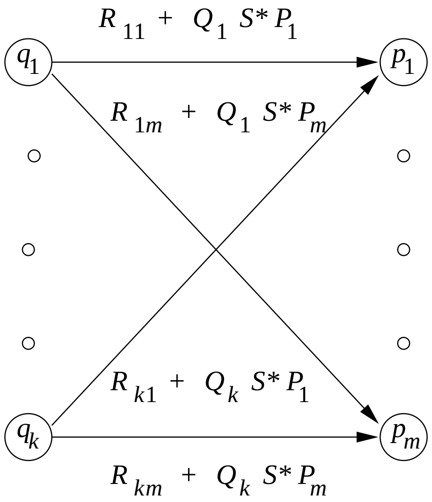
We need a more efficient approach:
the state elimination technique

The state elimination technique

Let's label the edges with regex's instead of symbols

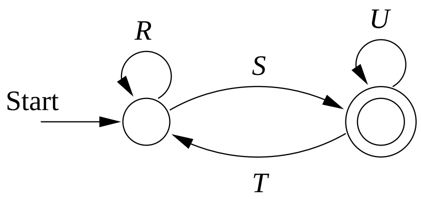


Now, let's eliminate state s.



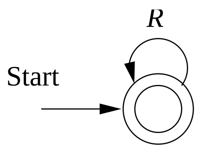
For each accepting state q eliminate from the original automaton all states exept q_0 and q.

For each $q \in F$ we'll be left with an A_q that looks like



that corresponds to the regex $E_q = (R{+}SU^*T)^*SU^*$

or with A_q looking like

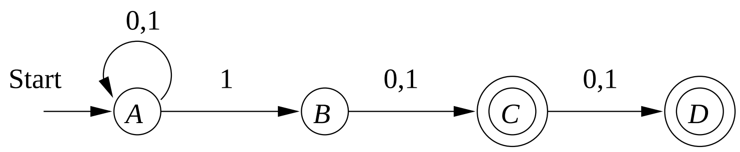


corresponding to the regex $E_q = R^*$

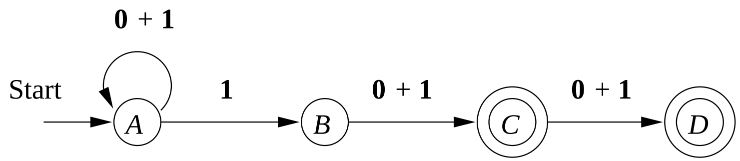
- The final expression is

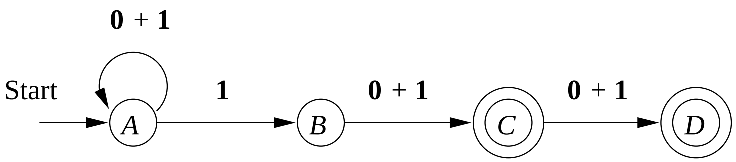
$$\bigoplus_{q \in F} E_q$$

Example: $\mathcal{A}$, where $L(\mathcal{A}) = \{W : w = x1b, \text{ or } w = x1bc, \ \ x \in \{0,1\}^*, \{b,c\} \subseteq \{0,1\}\}$

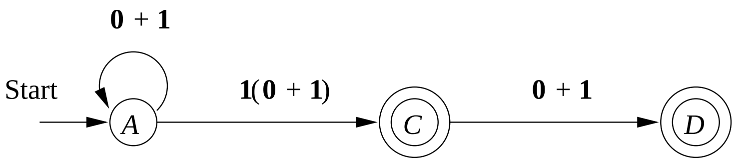


We turn this into an automaton with regex labels

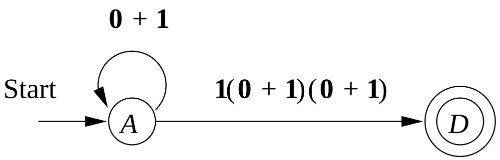




Let's eliminate state B

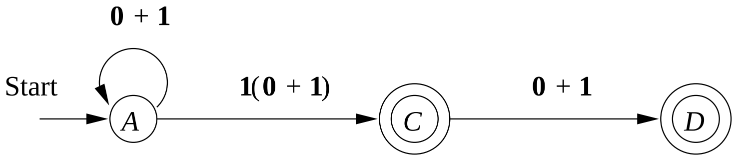


Then we eliminate state C and obtain $\mathcal{A}_D$

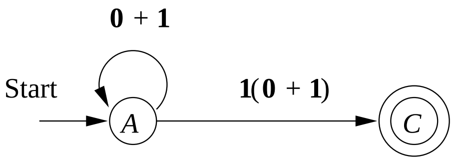


with regex $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$

From



we can eliminate D to obtain $\mathcal{A}_C$



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

- The final expression is the sum of the previous two regex's:

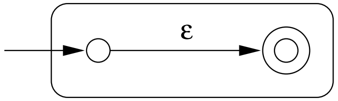
$$(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})+(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$$

From regex's to ϵ-NFA's

Theorem 3.7: For every regex R we can construct and ϵ-NFA A, s.t. $L(A) = L(R)$.

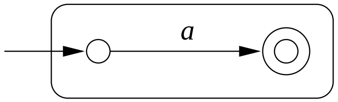
Proof: By structural induction:

Basis: Automata for $\boldsymbol{\epsilon}$, $\emptyset$, and $\boldsymbol{a}$.



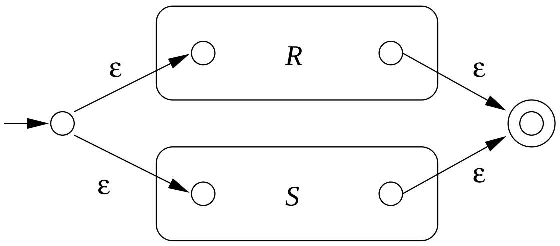
(a)

(b)

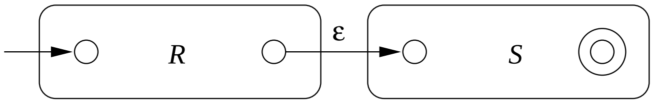


(c)

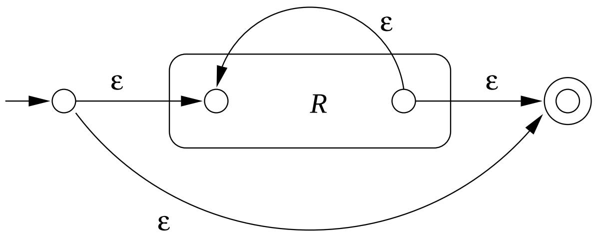
Induction: Automata for $R + S$, RS, and R^*



(a)

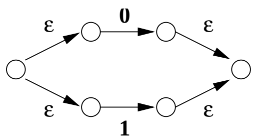


(b)

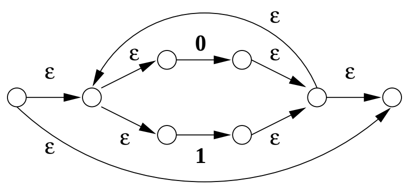


(c)

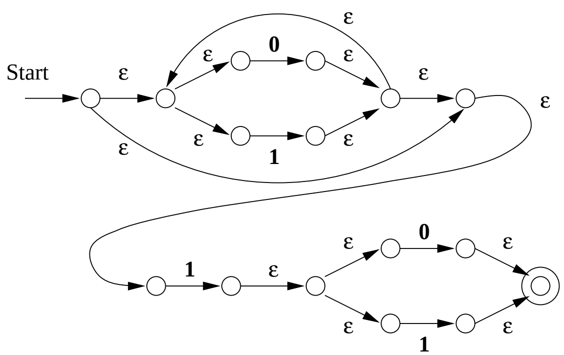
Example: We convert $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$



(a)



(b)



(c)