SCS 2112 : Automata Theory

Dinuni Fernando, PhD

Regular Language I

Languages

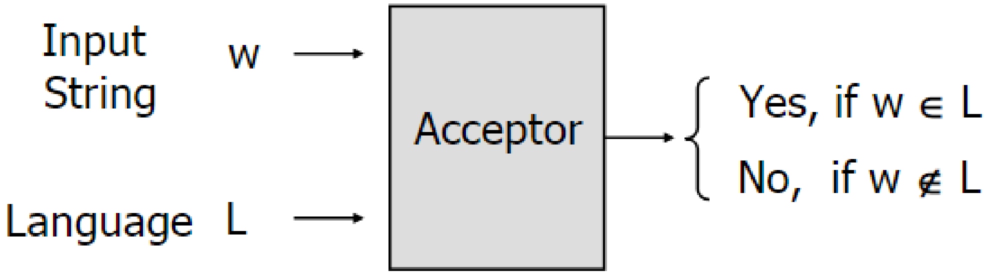
- A language is a set of words (strings) : Notation: L
- A language can be finite or infinite.
- The language with all words over some alphabet Σ is denoted Σ^*.
- A language with zero words called an Empty Language and denoted by ϕ.
- The empty string is denoted with ε or sometimes Λ or λ.

Regular Languages

Definition : A language is regular if there exist a finite acceptor for it.

Every regular language can be described by some NFA or DFA.

- Acceptor = determines if an input string belongs to a language L



Regular Expressions

- One way of describing regular languages is via the notation of regular expressions.
- Regular Expressions involves a combination of strings of symbols from some alphabet Σ, parentheses, and the operators +, . and *.
- The simplest case is the language {a}, which will be denoted by the regular expression a.

UCSC

Regular Expressions

- Slightly more complicated is the language {a, b, c},
  - for which, using the + to denote union, we have the regular expression a+b+c.
- We use · for concatenation and * for star-closure in a similar way . Consider following regular expression (a + (b·c))*

What is the strings that accepted by above regular Language ?

UCSC

Regular Expressions

- Slightly more complicated is the language {a, b, c},
    - for which, using the + to denote union, we have the regular expression a+b+c.
- We use · for concatenation and * for star-closure in a similar way . Consider following regular expression (a + (b·c))*

Accepting regular Language is {λ, a, bc, aa, abc, bca, bcbc, aaa, aabc,...}.

- We construct regular expressions from primitive constituents by repeatedly applying certain recursive rules.

Definition of Regular Expression

Let Σ be a given alphabet. Then,

1. ϕ, ε (λ) and a ∈ Σ are all regular expressions. These are called primitive regular expressions.

   Eg: L(ϕ) = {}   L(ε) = {ε}= {“”}   L(a) = {a} For every a ∈ Σ.

2. If r1 and r2 are regular expressions, so are r1+ r2,r1.r2 and (r1)*.
3. A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

UCSC

Regular Expression

- If r1 and r2 are regular expressions,

  1. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
  2. $L(r_1 . r_2) = L(r_1) . L(r_2)$
  3. $L((r)) = L(r)$
  4. $L(r^*) = (L(r))^*$

- Generally symbol "." is omitted in regular expressions.
- Any language that can be obtained by applying of the above rules in a **regular language** over Σ.
- In some literature "|" meta-symbol has been used instead of "+".
- Theorem: A language is regular if and only if some regular expression describes it.

Algebraic properties of regular expressions

For regular expressions

$L + M = M + L$	(commutative law for union)
$(L+M) + N = L + (M+N)$	(Associative law for union)
$(L.M).N = L.(M.N)$	(Associative law for concatenation)
$L.(M + N) = LM + LN$	(Distributive Laws)
$(M + N).L = ML + NL$	
$L + L = L$	(Idempotent Law)
$(L^*)^* = L^*$	$L? = \varepsilon + L$
$(L^*M^*)^* = (L + M)^*$	$LL^* = L^+ = L^*L$
$L^*.L^* = L^*$	$L\varepsilon = L = \varepsilon L$

Abbreviations

- [qb01] = q|b|0|1
- [0-9] = [0123456789]
- [a-zA-Z] – all uppercase and lowercase English letters.
- [0-9]* - zero or more occurrences of digits 0 -9
- s^+ - one or more occurrences of s
- s? – zero or one occurrence of s (s|∈)

Example :

[0-9][0-9]*

Abbreviations

- [qb01] = q|b|0|1
- [0-9] = [0123456789]
- [a-zA-Z] – all uppercase and lowercase English letters.
- [0-9]* - zero or more occurrences of digits 0 -9
- s^+ - one or more occurrences of s
- s? – zero or one occurrence of s (s|∈)

Example :

$$[0-9][0-9]^* = [0-9]^+$$

Example 1, 2

Example 1:

For $\Sigma = \{a, b, c\}$, the string

$(a+b+c)^* .(c + \phi)$ is a regular expression

Example 2:

For $\Sigma = \{a, b, c\}$, the string

$(a+b+)^* .(c + \phi)$ is not a regular expression

Example 3

Let $\Sigma = \{a,b\}$, What are the elements in $L(a^*.(a+b))$?

Example 3

Let Σ = {a,b}, What are the elements in $L(a^*.(a+b))$?

$$
\begin{aligned}
L(a^*.(a+b)) &= L(a^*).L(a+b) \\
&= (L(a))^*.(L(a) \cup L(b)) \\
&= \{\varepsilon, a, aa, aaa, \ldots.\}.\{a,b\} \\
&= \{a, aa, aaa, \ldots, b, ab, aab, \ldots.\}
\end{aligned}
$$

UCSC

Example 4

1. Let $\Sigma = \{a, b\}$, the expression $r = (a+b)^*(a+bb)$. What are the elements in L(r)?

2. Let $\Sigma = \{a, b\}$, the expression $r = (aa)^* (bb)^* b$. What are the elements in L(r)?

UCSC

Example 4

1. Let Σ = {a, b}, the expression $r = (a+b)^*(a+bb)$. What are the elements in L(r)?

   $L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}$.

2. Let Σ = {a, b}, the expression $r = (aa)^* (bb)^* b$. What are the elements in L(r)?

   $L(r) = \{a^{2n}b^{2m+1} : n \geq 0, m \geq 0\}$

Example 5:

- $a+b+c = ?$
- $abc = ?$
- $ab^* = ?$
- $(ab)^* = ?$
- $(a+b)^* = ?$
- $a+b^* = ?$

UCSC

Example 5 :

- $a+b+c = \{a, b, c\}$
- $abc = \{abc\}$
- $ab^* = \{a, ab, abb, abbb, \ldots\}$
- $(ab)^* = \{\in, ab, abab, ababab, \ldots\}$
- $(a+b)^* = \{\in, a, b, aa, ab, ba, bb, aaa, \ldots\}$
- $a+b^* = \{a, \in, b, bb, bbb, \ldots\}$

UCSC

Example 6

Let $\Sigma = \{0,1\}$, r be a regular expression and L(r) $= \{w \in \Sigma^* \mid w$ has at least one pair of consecutive zeros$\}$. What is r?

Example 6

Let $\Sigma = \{0,1\}$, r be a regular expression and $L(r) = \{w \in \Sigma^* \mid w$ has at least one pair of consecutive zeros$\}$.

What is r?

Every string in $L(r)$ must contain 00 somewhere, but what comes before and what goes after is completely arbitrary. An arbitrary string on {0,1} can be denoted by $(0+1)^*$.

$$r = (0+1)^*00(0+1)^*$$

Example 7

What is the language L(a.b+c)?

Two interpretations

- L(a.(b+c)) ? Or
- L((a.b) +c)) ?

Set of precedence rules are being used to solve this problem.

- Star-Closure highest precedence
- Concatenation
- Union (+)

UCSC

Example 8

1. Find all strings in $L((a + b)\ b\ (a + ab)^*)$ of length less than four.
2. Find all strings in $L((a + b)^*b(a + ab)^*)$ of length less than four.
3. Find a regular expression for the set $\{a^n b^m$: n≥3, m is even$\}$.
4. Find a regular expression for the set $\{a^n b^m:(n + m)$ is even$\}$.

Example 9

Give regular expressions for the following languages

- $L1 = \{a^n b^m : n \geq 4, m \leq 3\}$.
- $L2 = \{a^n b^m : n < 4, m \leq 3\}$.

Example 9

Give regular expressions for the following languages

- $L1 = \{a^n b^m : n \geq 4, m \leq 3\}$.

  (aaaa)a*.(∈+b+bb+bbb)

- $L2 = \{a^n b^m : n < 4, m \leq 3\}$.

  (∈+a+aa+aaa).(∈+b+bb+bbb)

Equivalence in Regular Expressions

- Two regular expressions are said to be equivalent if they define the same language.
  - Two regular expressions r_1 and r_2 are equivalent if $L(r_1) = L(r_2)$.
- Generally, there are an unlimited number of regular expressions for any given language.

Relationship between Regular Expressions and Regular Languages

- For every regular language a regular expression can be constructed and
- For every regular expression there is a regular language.

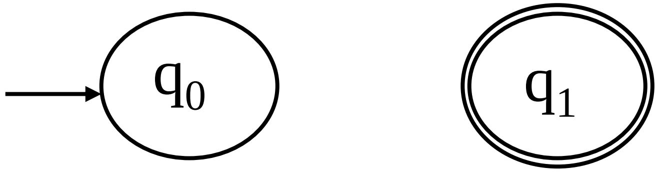
Theorem

Let r be a regular expression. Then there exists some nondeterministic finite accepter (NFA) that accepts L(r). Consequently, L(r) is a regular language.

Proofs

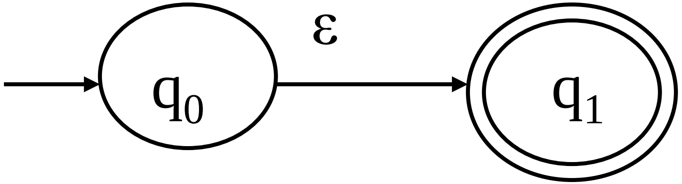
- We begin with automata that accept the languages for the simple regular expressions (primitive regular expressions): ϕ, ε, a

a) NFA accepts ϕ

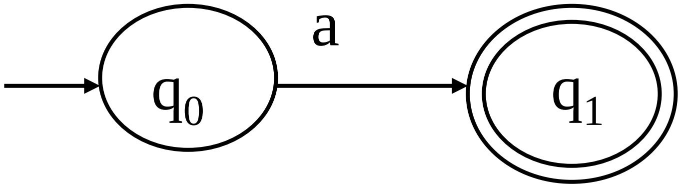


Proofs continued.

b) NFA accepts {ε}

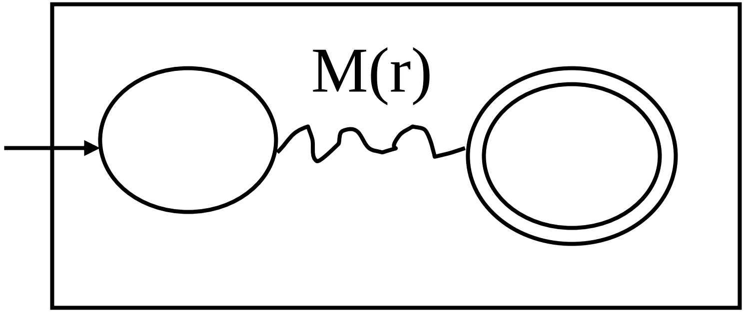


c) NFA accepts {a}



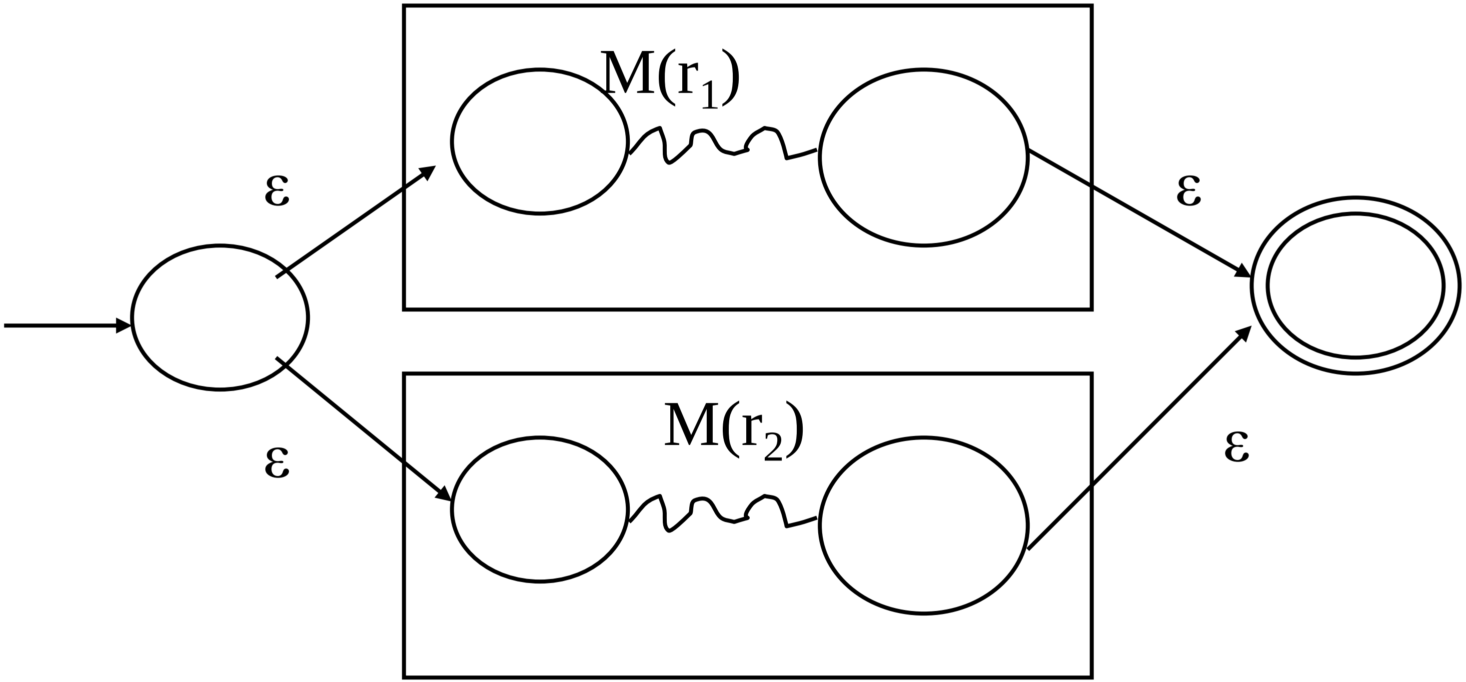
Schematic representation of an nfa accepting L(r).

Assume now that we have automata $M\,(r_1)$ and $M\,(r_2)$ that accept languages denoted by regular expressions r_1 and r_2, respectively.



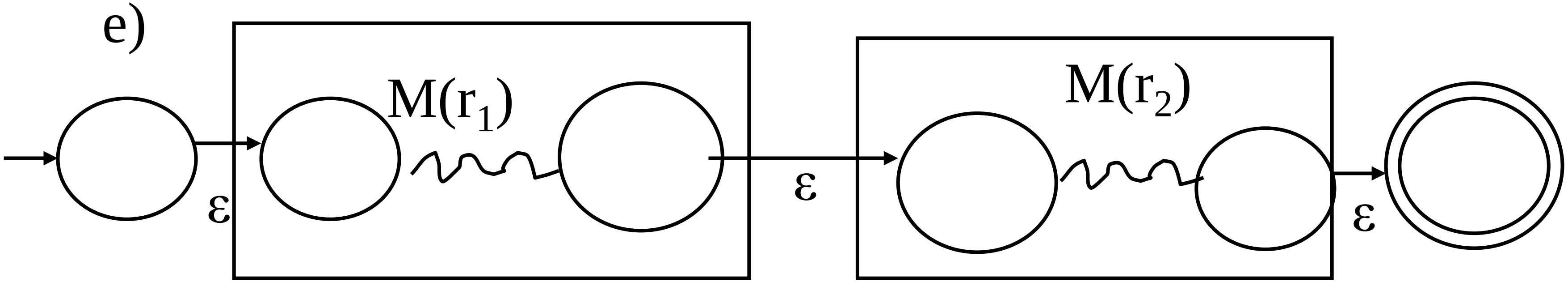
Proofs continued.

d) Automata for $L(r_1 + r_2)$



Proofs continued.

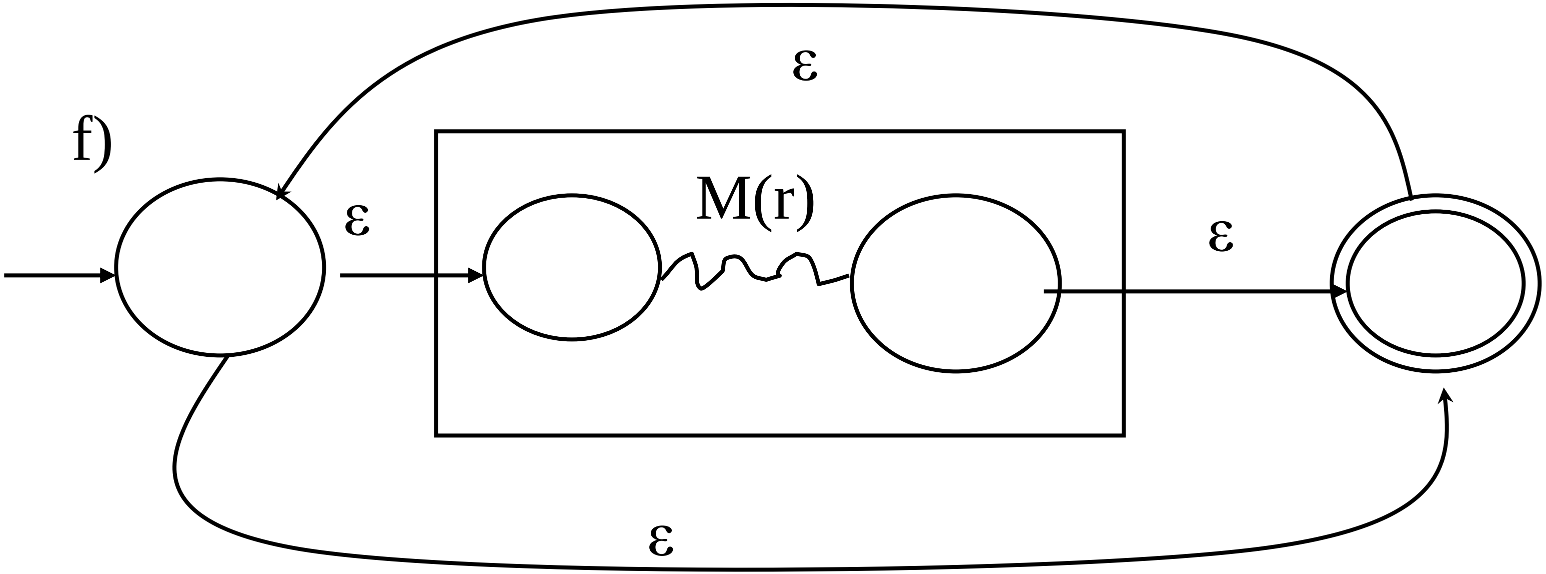
Automaton for $L(r_1 r_2)$



Proofs continued.

Automation for L(r*)

f)



Example 10

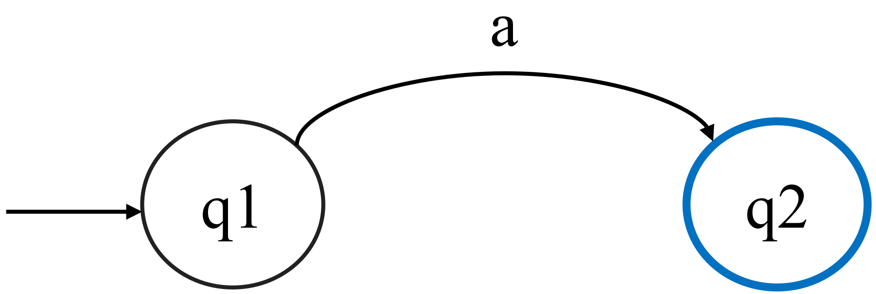
Build an automaton to accept the language defined by the regular expression

$$R = (ab + a)^*$$

UCSC

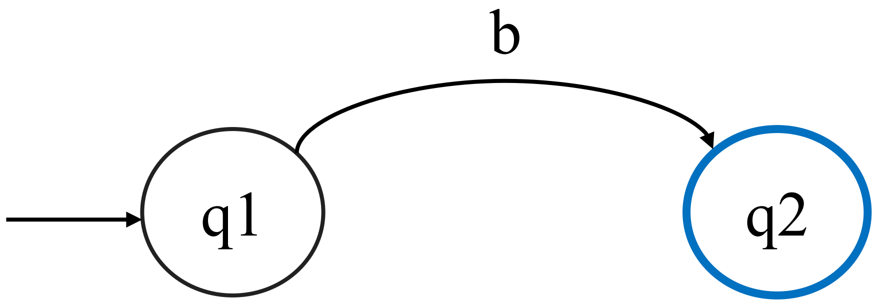
Example 10

R = a



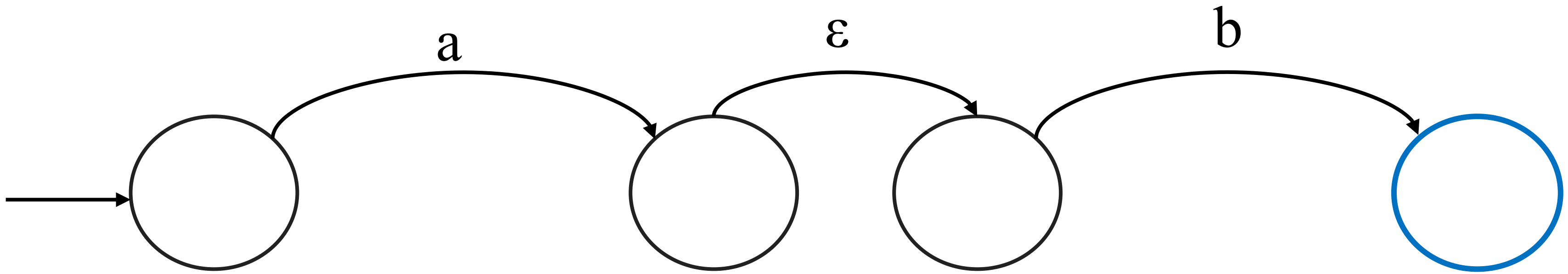
Example 10 – cont'd

R = b



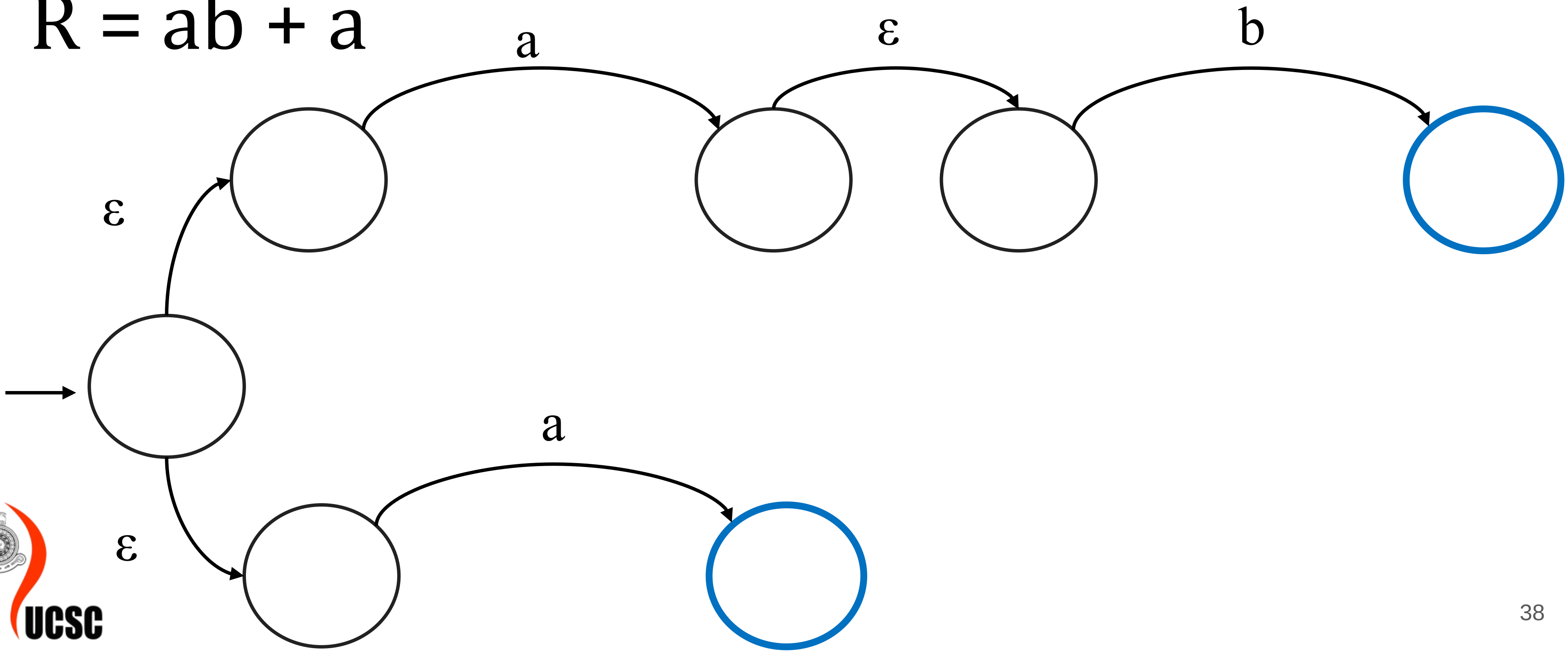
Example 10 – cont'd

R = ab



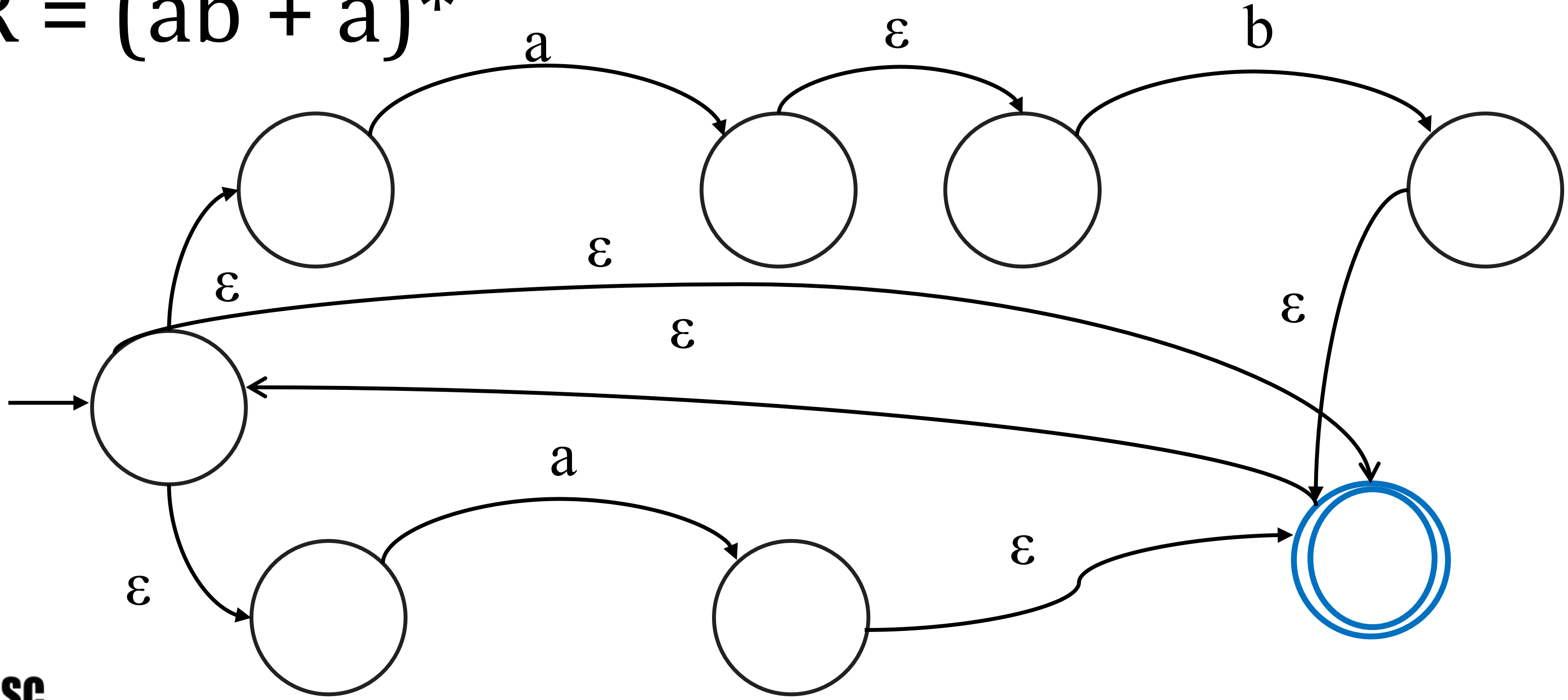
Example 10 – cont'd

R = ab + a



Example 10 – cont'd

R = (ab + a)*



Exercises

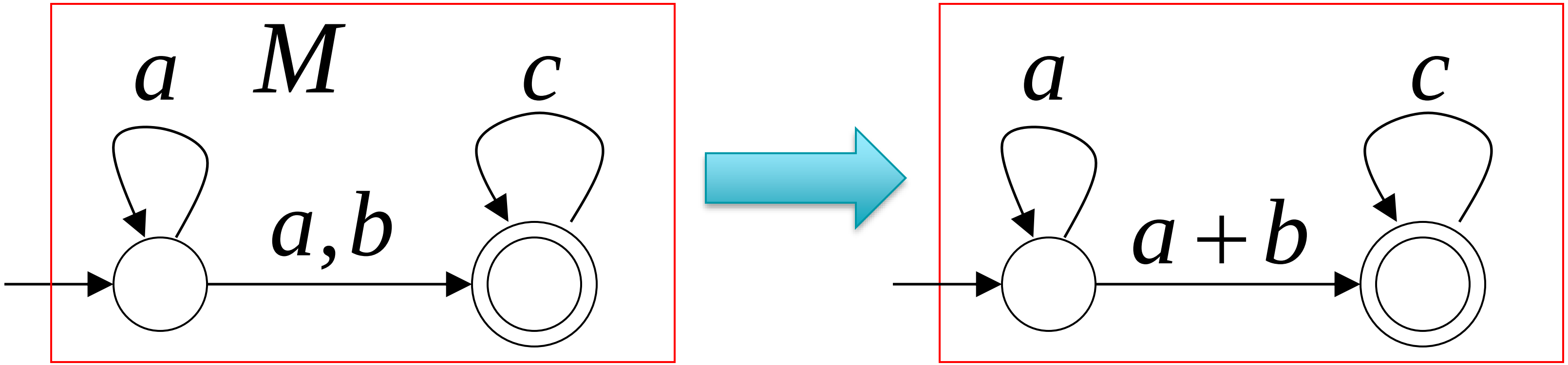
- Find an nfa that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \varepsilon)$
- Find an nfa that accepts $L(r)$, where r = (ε|a*b)
- Find an nfa that accepts the language L(ab*aa + bba*b*)

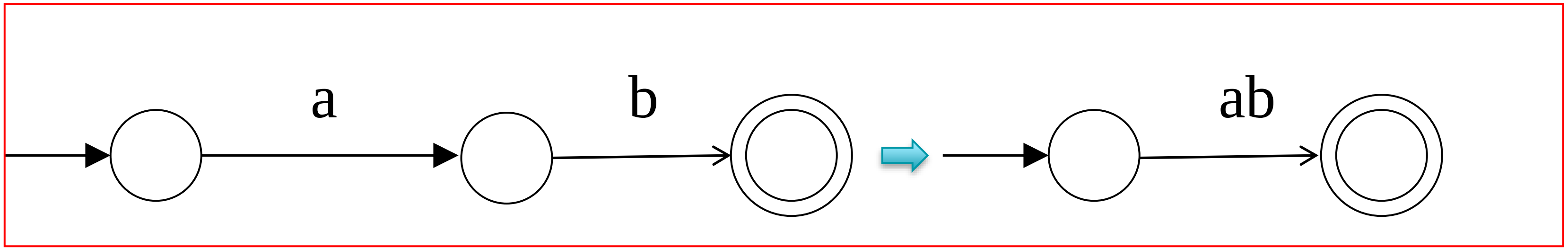
Regular Expressions for Regular Languages

- For every regular language, there should exist a corresponding regular expression
- Representation: **Generalized Transition Graphs (GTG)**
- A generalized transition graph is a transition graph whose edges are labeled with regular expressions; otherwise it is the same as the usual transition graph.
- The label of any walk from the initial state to a final state is the concatenation of several regular expressions, and hence itself a regular expression.

UCSC

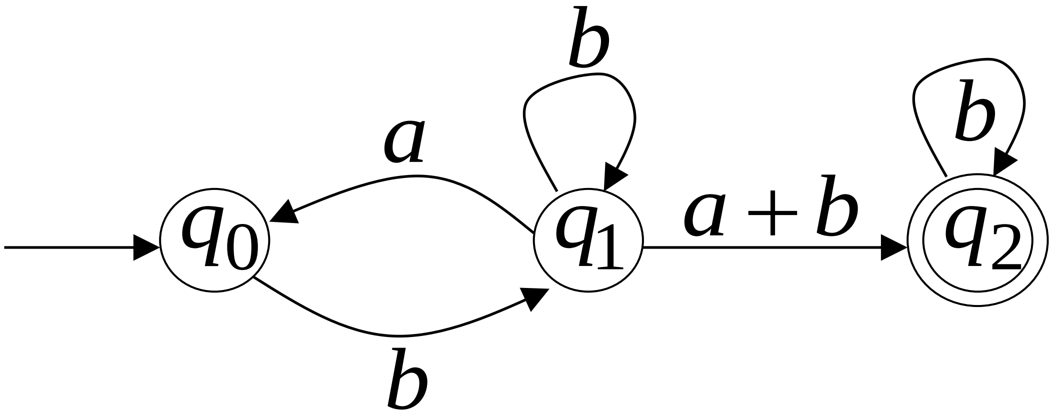
Example



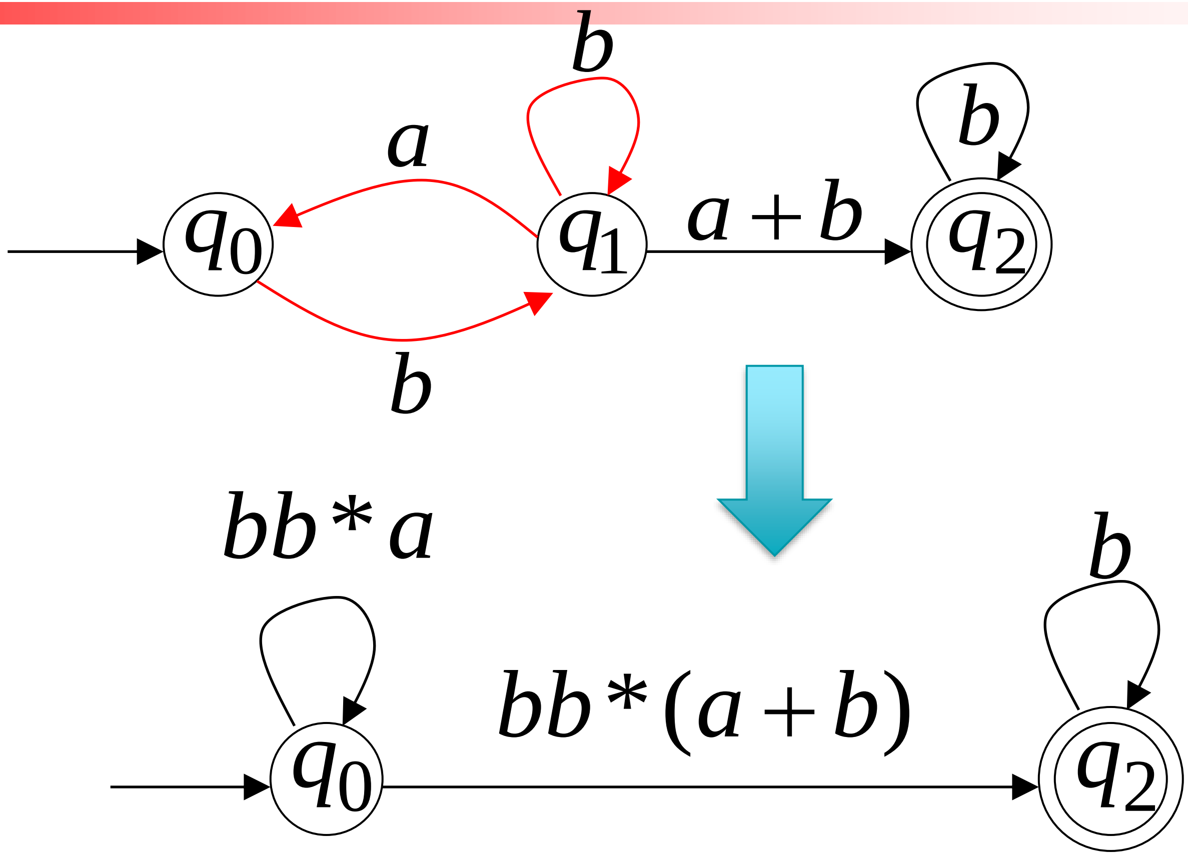


How to deduce the regular expression represented by a generalized transition graph ?

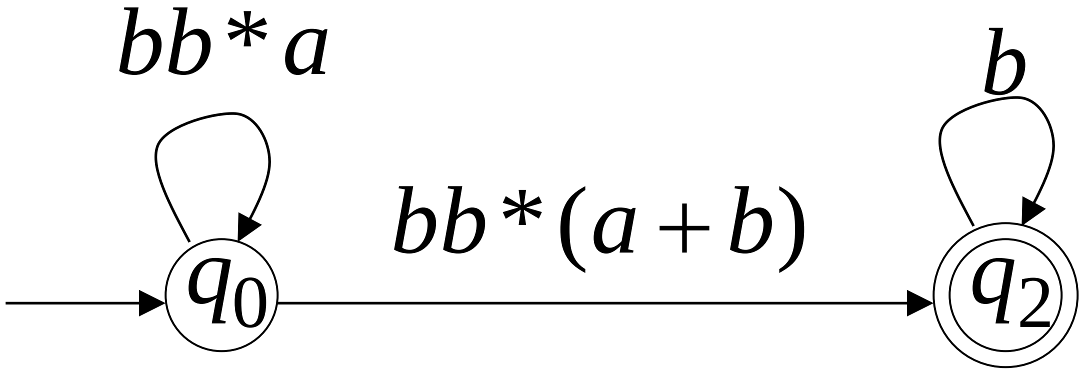
Reducing the number of states



Reducing the number of states

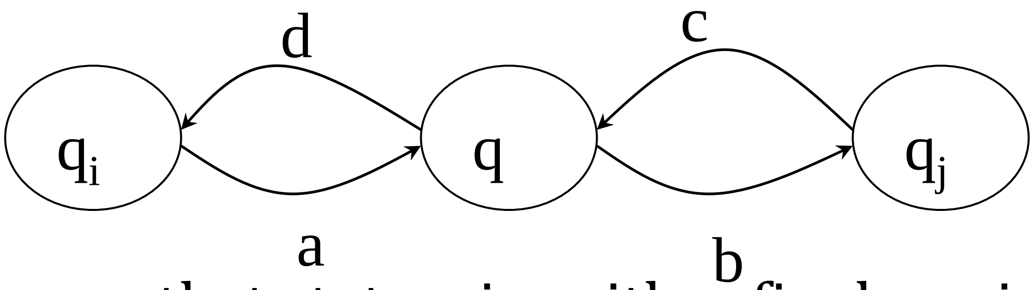


Resulting Regular Expression



$$r = (bb^*a)^*bb^*(a+b)b^*$$

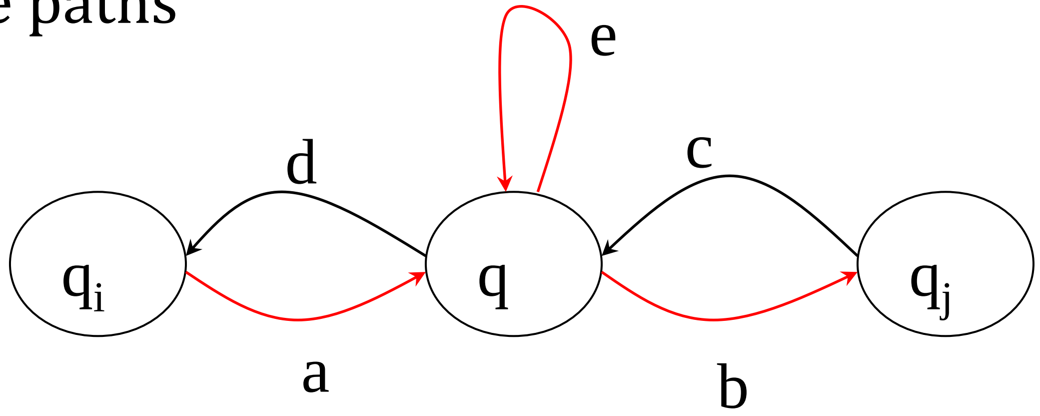
- The regular expression represented by a generalized transition graph can be deduced much easily by reducing the states of the graph.



- Let's assume that state q is neither final nor initial state.

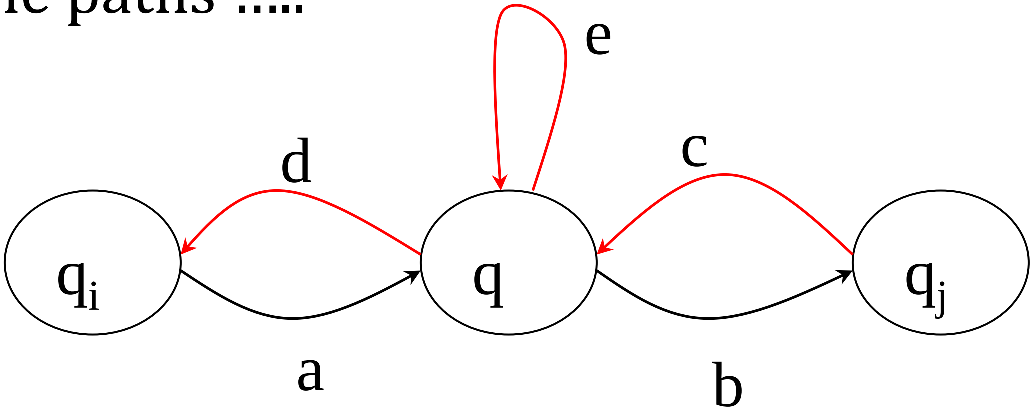
Reducing a state (say q) from the transition graph.

Possible paths



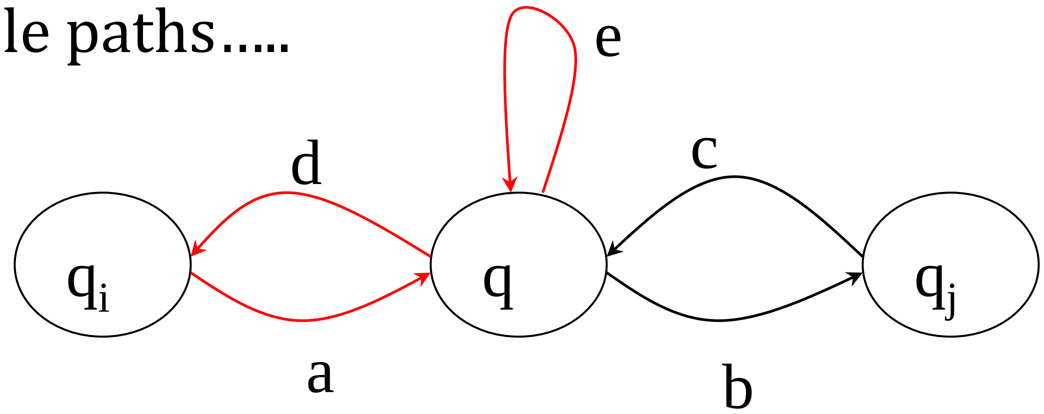
Regular expression generated ae*b

- Possible paths .....



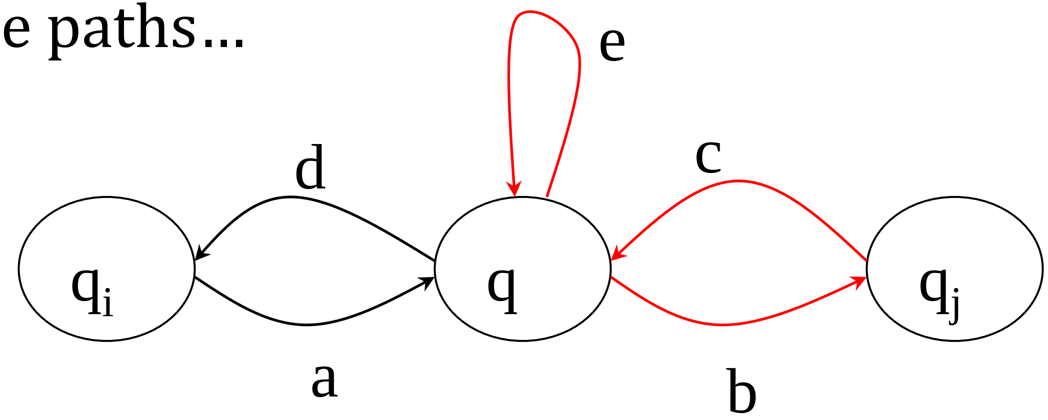
Regular expression generated ce*d

- Possible paths.....



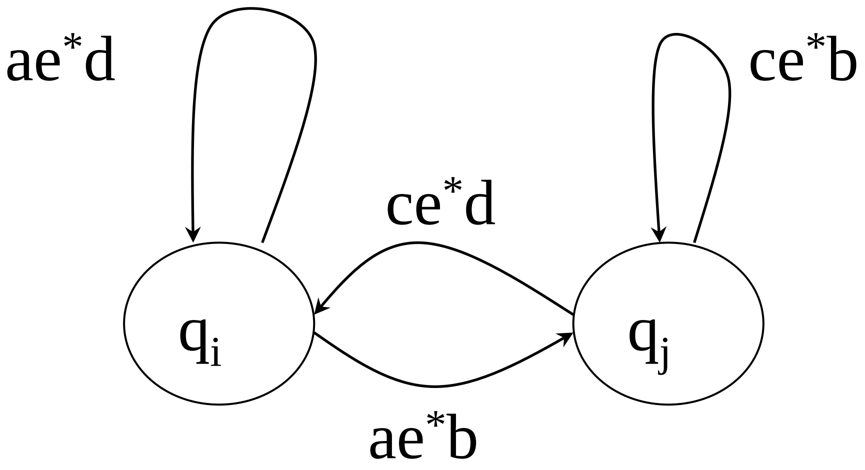
Regular expression generated ae*d

- Possible paths…



Regular expression generated ce*b

Final generalized graph



- Reducing a state of a transition graph should not reduce any of its possible paths.

Converting DFA's to Regular Expressions (by Eliminating states)

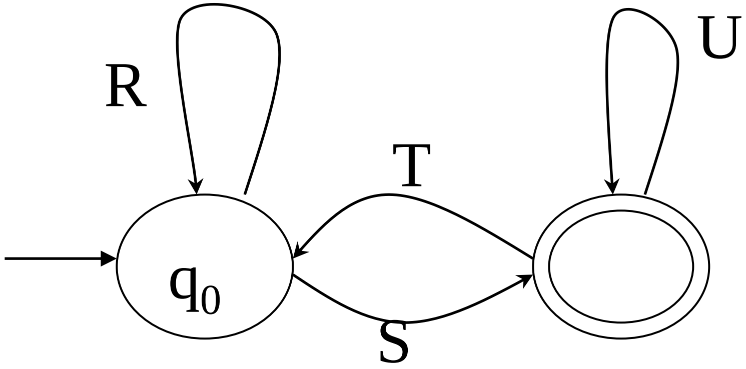
State reduction algorithm

1. Replace edges of the automata with equivalent regular expressions.

2. Eliminate all states except the initial and final states.

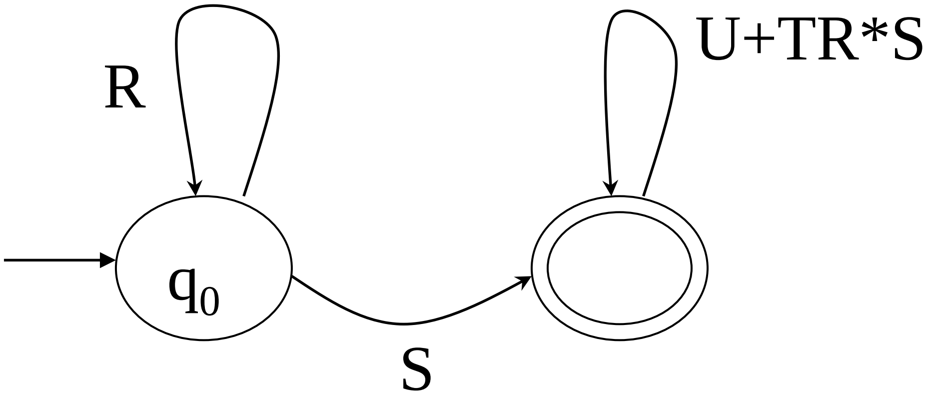
3. If initial state is not a final state the final automata must consist of two nodes as below

Example



- What is the Regular expression representing the above GTG ?

R*S(U+TR*S)*

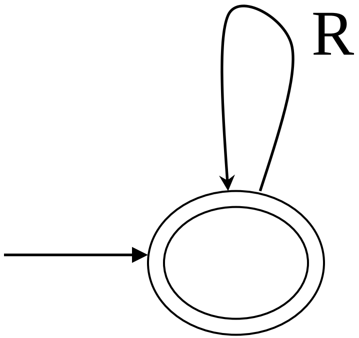


Regular expression representing the above RTG is

R*S(U+TR*S)*

State reduction algorithm [cont'd]

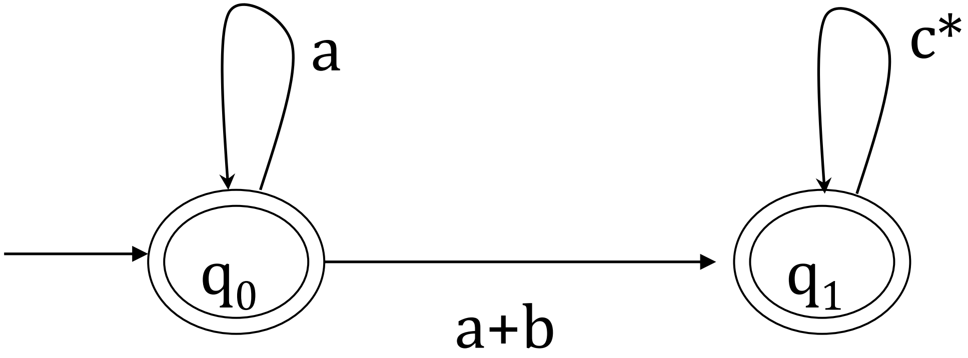
4. If the start state is also an accepting state after reduction we may end up in the following form.



The regular expression denoting this machine is R*

Exercise

- What is the language (Regular Expression) accepted by the following GTG.



$$L = (a^* + a^* (a + b)\, c^*)$$