Section: Regular Languages

Regular Expressions

Method to represent strings in a language

$+$ **union (or)**
$\circ$ **concatenation (AND) (can omit)**
$*$ **star-closure (repeat 0 or more times)**

Example:

$(a + b)^* \circ a \circ (a + b)^*$

$(a+b)^* a (a+b)^*$

strings w/ at least one a

Example:

$(aa)^*$

even number of a's

Definition Given Σ,

1. $\emptyset$, λ, $a \in \Sigma$ **are R.E.**
2. **If r and s are R.E. then**
   - **r+s is R.E.**
   - **rs is R.E.**
   - **(r) is a R.E.**
   - **r* is R.E.**
3. **r is a R.E. iff it can be derived from (1) with a finite number of applications of (2).**

Definition: L(r) = language denoted by R.E. r.

1. $\emptyset$, $\{\lambda\}$, $\{a\}$ **are L denoted by a R.E.**
2. **if r and s are R.E. then**
   (a) $\mathbf{L(r+s) = L(r) \cup L(s)}$
   (b) $\mathbf{L(rs) = L(r) \circ L(s)}$
   (c) $\mathbf{L((r)) = L(r)}$
   (d) $\mathbf{L((r)^*) = (L(r)^*)}$

Precedence Rules

$*$ **highest**

$\circ$

$+$

Example:

$ab^* + c =$ $(a(b^*)) + c$

OR $b^*(ab^* + ab^*ab^* + \lambda)ab$

Examples:

1. $\Sigma = \{a, b\}$, $\{w \in \Sigma^* \mid w$ **has an odd number of** a**'s followed by an even number of** b**'s**$\}$.

$a(aa)^*(bb)^*$

2. $\Sigma = \{a, b\}$, $\{w \in \Sigma^* \mid w$ **has no more than 3** a**'s and must end in** $ab\}$.

$(b^* + b^*ab^* + b^*ab^*ab^*)ab$

3. Regular expression for all integers (including negative)

$0 + (- + \lambda)(1+2+\dots+9)(0+1+\dots+9)^*$

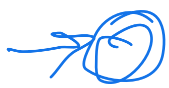
Section 3.2 Equivalence of DFA and R.E.

Theorem Let r be a R.E. Then $\exists$ NFA M s.t. L(M) = L(r).

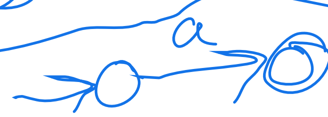
- **Proof:**

  $\emptyset$

  $\{\lambda\}$

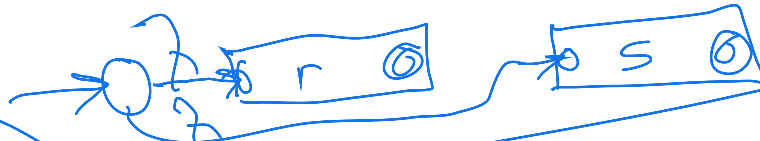


  $\{a\}$



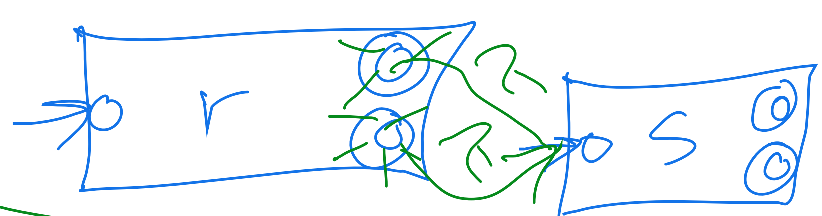
  Suppose r and s are R.E.

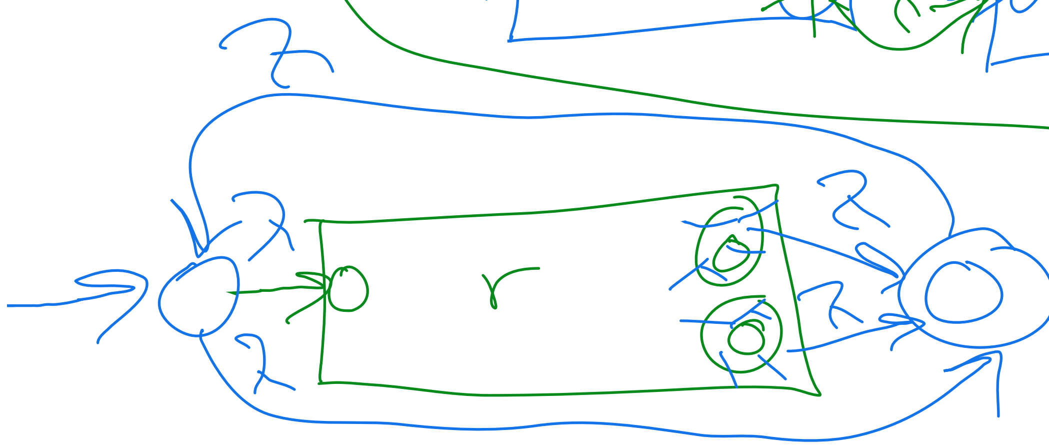
  1. r+s



  2. r∘s

  3. r*



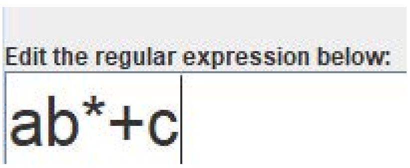


STOPPED!

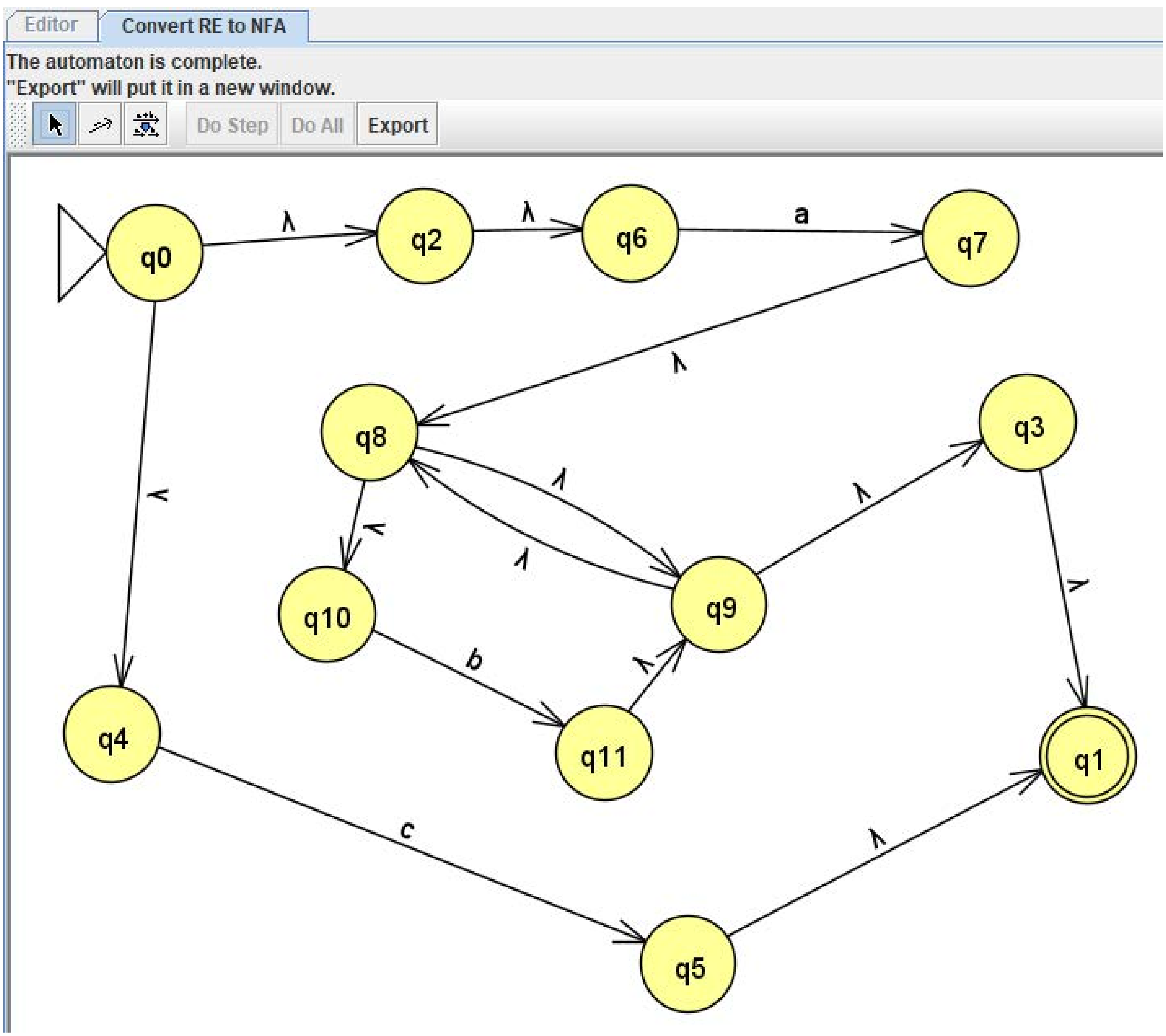
Example

$ab^* + c$

In JFLAP:



Convert to NFA:



Theorem Let L be regular. Then $\exists$ R.E. r s.t. L=L(r).

Proof Idea: remove states sucessively until two states left

- **Proof:**

  L is regular

  $\Rightarrow \exists$ NFA M s.t. L=L(M)

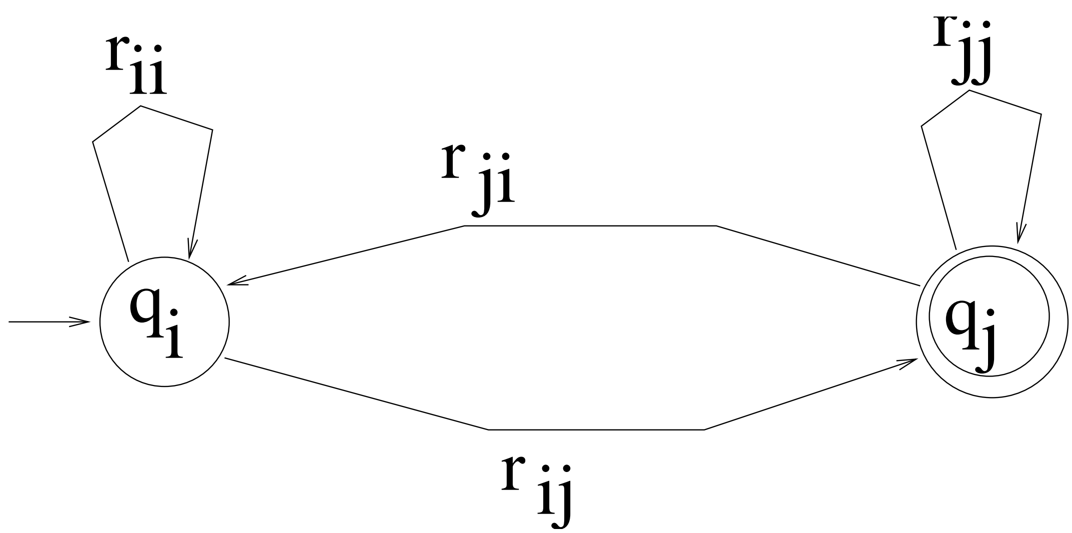
  1. Assume M has one final state and $q_0 \notin F$

  2. Convert to a generalized transition graph (GTG), all possible edges are present. (regular expr on arcs)

     If no edge, label with $\emptyset$

     Let r_{ij} stand for label of the edge from q_i to q_j

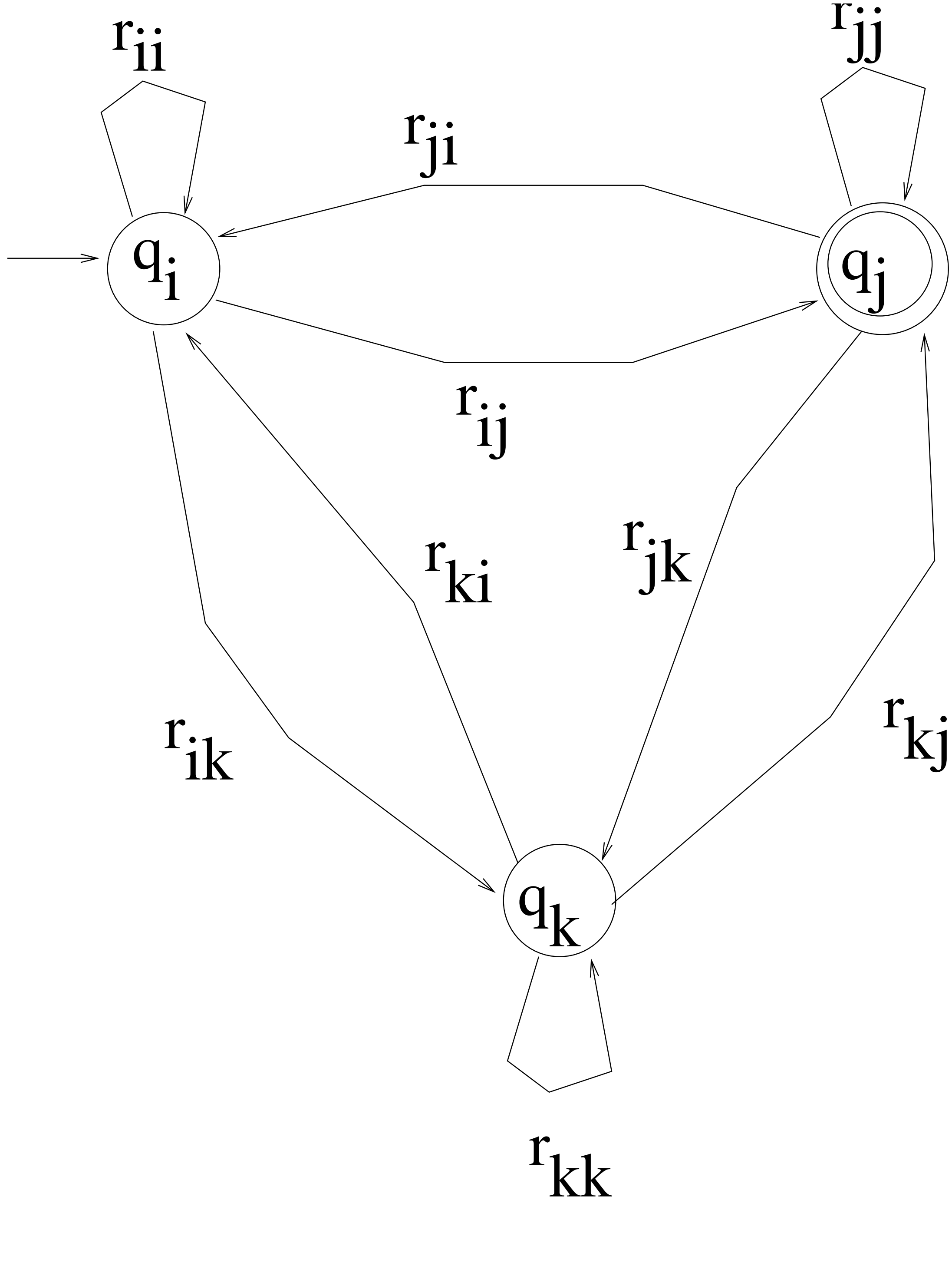
3. If the GTG has only two states, then it has the following form:



In this case the regular expression is:

$r = (r_{ii}^* r_{ij} r_{jj}^* r_{ji})^* r_{ii}^* r_{ij} r_{jj}^*$

4. If the GTG has three states then it must have the following form:



REPLACE	**WITH**
r_{ii}	$r_{ii} + r_{ik} r_{kk}^{*} r_{ki}$
r_{jj}	$r_{jj} + r_{jk} r_{kk}^{*} r_{kj}$
r_{ij}	$r_{ij} + r_{ik} r_{kk}^{*} r_{kj}$
r_{ji}	$r_{ji} + r_{jk} r_{kk}^{*} r_{ki}$

remove state q_k

5. If the GTG has four or more states, pick a state q_k to be removed (not initial or final state).

For all $o \neq k, p \neq k$ use the rule r_{op} replaced with $r_{op} + r_{ok} r_{kk}^{*} r_{kp}$ with different values of o and p.

When done, remove q_k and all its edges. Continue eliminating states until only two states are left. Finish with step 3.

6. In each step, simplify the regular expressions r and s with:

$r + r = r$

$s + r^* s = r^* s$

$r + \emptyset = r$

$r\emptyset = \emptyset$

$\emptyset^* = \{\lambda\}$

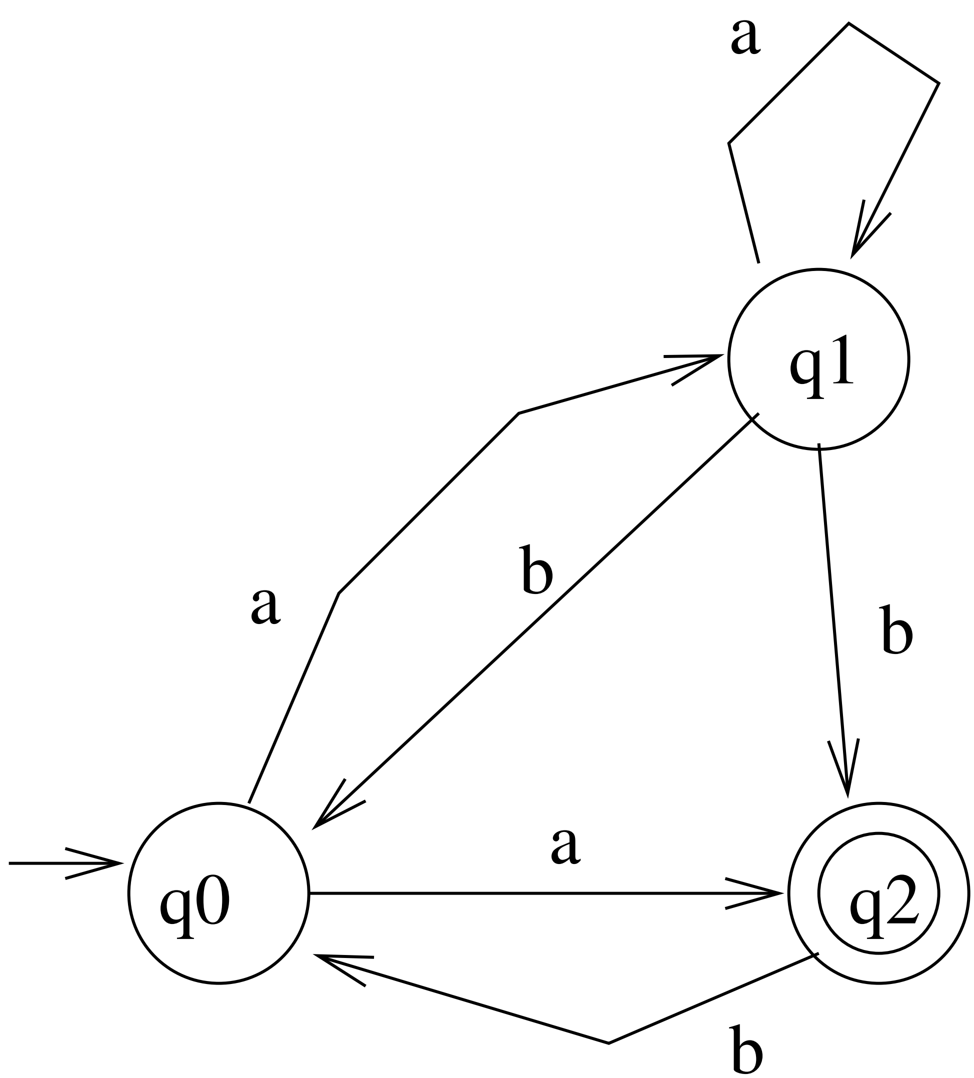
$r\lambda = r$

$(\lambda + r)^* = r^*$

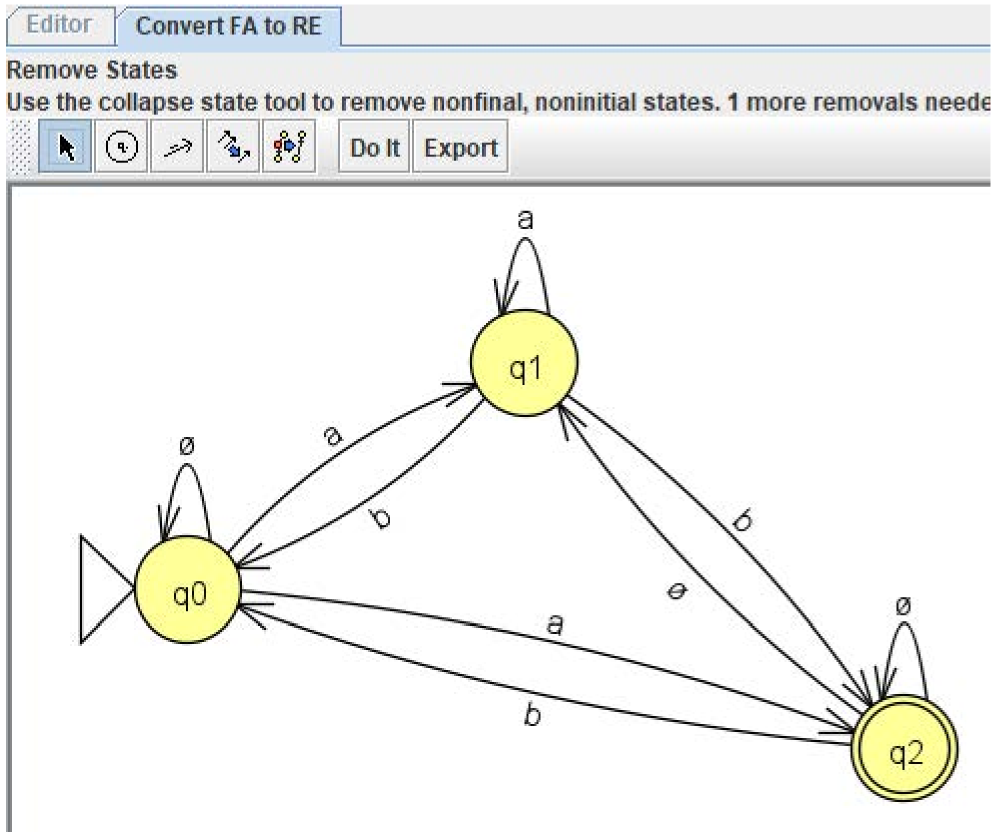
$(\lambda + r)r^* = r^*$

and similar rules.

Example:



Using JFLAP, first convert to GTG: add all missing edges:



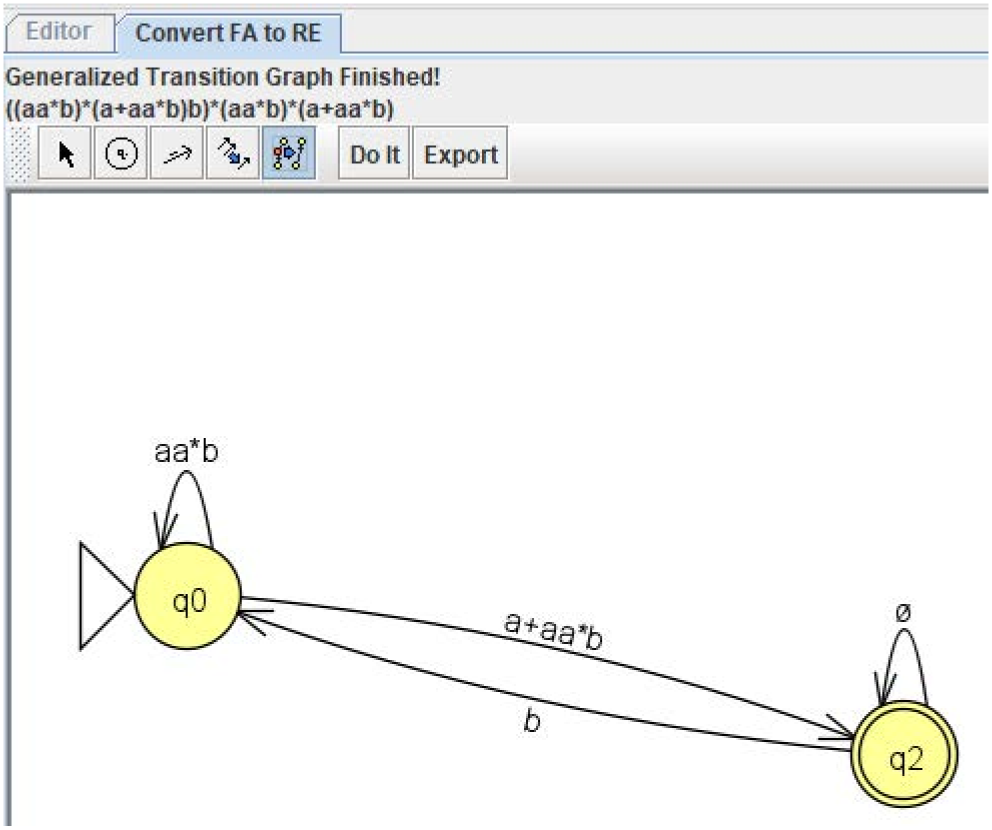
Then click on state q1 to remove it. Here are the replacement regular expressions:

Transitions

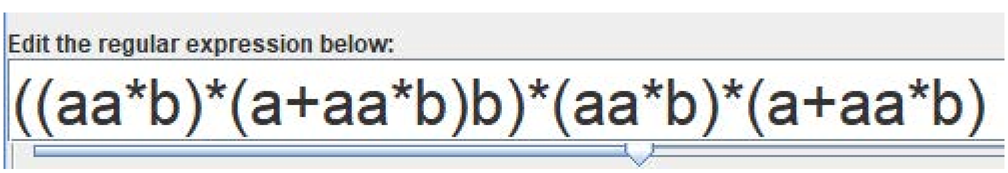
Select to see what transitions were co...

From	To	Label
0	0	aa*b
0	2	a+aa*b
2	0	b
2	2	ø

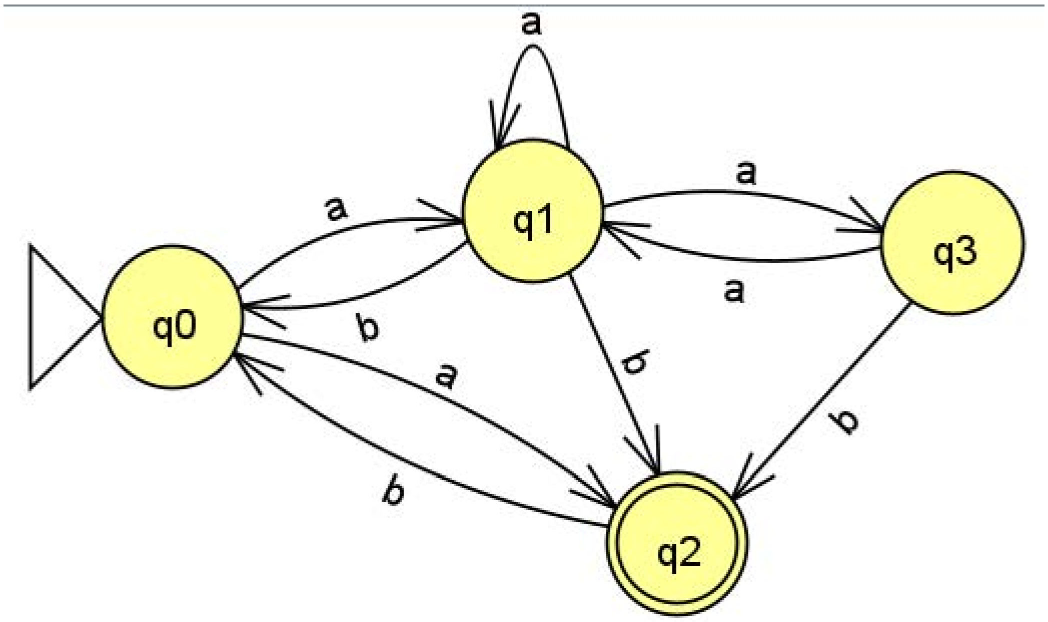
Then finalize and the resulting GTG is:



The resulting Regular Expression is listed above and also here:



We added one more state so we had this DFA:



Then we added missing arcs, and deleted q1 and then q3.

You can try that in JFLAP.

Grammar G=(V,T,S,P)

V variables (nonterminals)
T terminals
S start symbol
P productions

Right-linear grammar:

all productions of form

$$\mathbf{A \rightarrow xB}$$
$$\mathbf{A \rightarrow x}$$

where $\mathbf{A,B \in V,\ x \in T^*}$

Left-linear grammar:

all productions of form

$$\mathbf{A \rightarrow Bx}$$

$$\mathbf{A \rightarrow x}$$

where $\mathbf{A,B \in V,\ x \in T^*}$

Definition:

A regular grammar is a right-linear or left-linear grammar.

Example 1:

$$\mathbf{G=(\{S\},\{a,b\},S,P),\ P=}$$
$$\mathbf{S \rightarrow abS}$$ ← right-lin
$$\mathbf{S \rightarrow \lambda}$$
$$\mathbf{S \rightarrow Sab}$$ ← left-lin

Not regular

Example 2:

$$G=(\{S,B\},\{a,b\},S,P),\ P=$$
$$S \rightarrow aB \mid bS \mid \lambda$$
$$B \rightarrow aS \mid bB$$

regular grammar!

L = { strings with an even number of a's }

$\Sigma = \{a, b\}$

Theorem: L is a regular language iff $\exists$ regular grammar G s.t. L=L(G).

Outline of proof:

($\Longleftarrow$) **Given a regular grammar G**
Construct NFA M
Show L(G)=L(M)

($\Longrightarrow$) **Given a regular language**
$\exists$ DFA M s.t. L=L(M)
Construct reg. grammar G
Show L(G) = L(M)

Proof of Theorem:

($\Longleftarrow$) **Given a regular grammar G**
G=(V,T,S,P)

  V=$\{V_0, V_1, \ldots, V_y\}$
  T=$\{v_o, v_1, \ldots, v_z\}$
  S=V_0

Assume G is right-linear
  (see book for left-linear case).
Construct NFA M s.t. L(G)=L(M)
If w∈L(G), w=$v_1 v_2 \ldots v_k$

derivation

$$V_0 \Rightarrow v_1 V_i \Rightarrow v_1 v_2 V_j \stackrel{*}{\Rightarrow} v_1 v_2 v_3 \ldots v_{k-1} V_n \Rightarrow v_1 v_2 v_3 \ldots v_{k-1} v_k$$

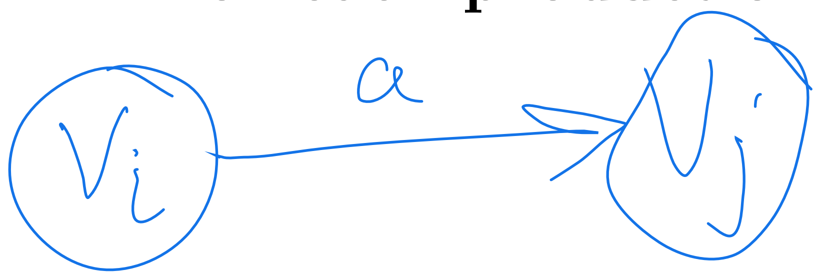
in NFA

NFA M

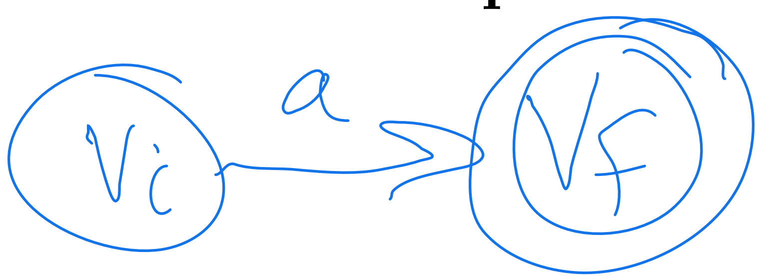
M=(V$\cup\{V_f\}$,T,δ,V_0,$\{V_f\}$)

V_0 **is the start (initial) state**

For each production, $V_i \rightarrow aV_j$**,**



For each production, $V_i \rightarrow a$**,**



Show L(G)=L(M)

Thus, given R.G. G,

L(G) is regular

(⟹) Given a regular language L

∃ DFA M s.t. L=L(M)

M=(Q,Σ**,**δ**,**q_0**, F)**

Q=$\{q_0, q_1, \ldots, q_n\}$

$\Sigma = \{a_1, a_2, \ldots, a_m\}$

Construct R.G. G s.t. L(G) = L(M

G=(Q,Σ**,**q_0**,P)**

if $\delta(q_i, a_j)$**=**q_k **then**

$q_i \to a_j q_k$

if q_k ∈**F then**

$q_k \to \lambda$

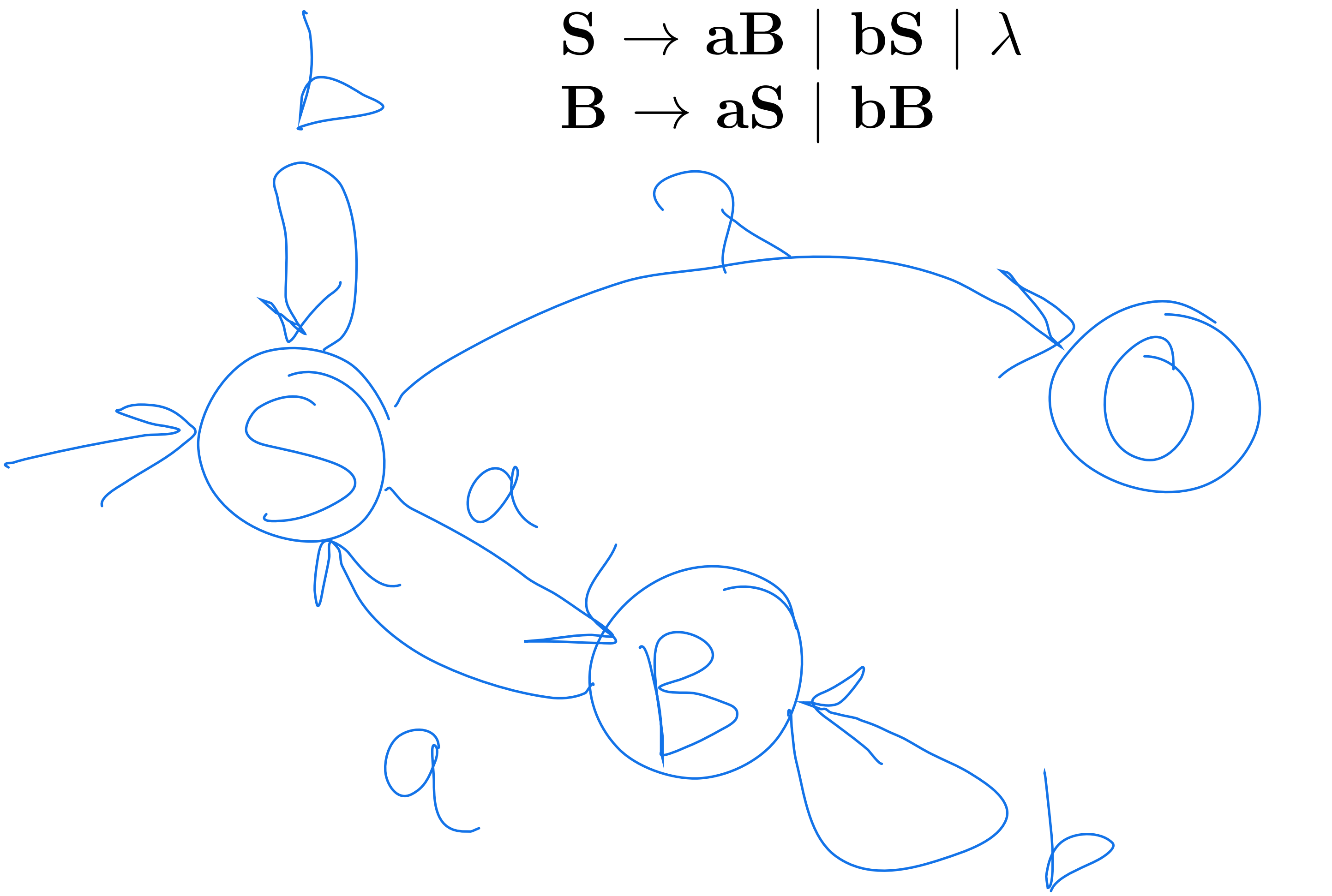
Show w ∈**L(M)** ⟺ **w** ∈ **L(G)**

Thus, L(G)=L(M).

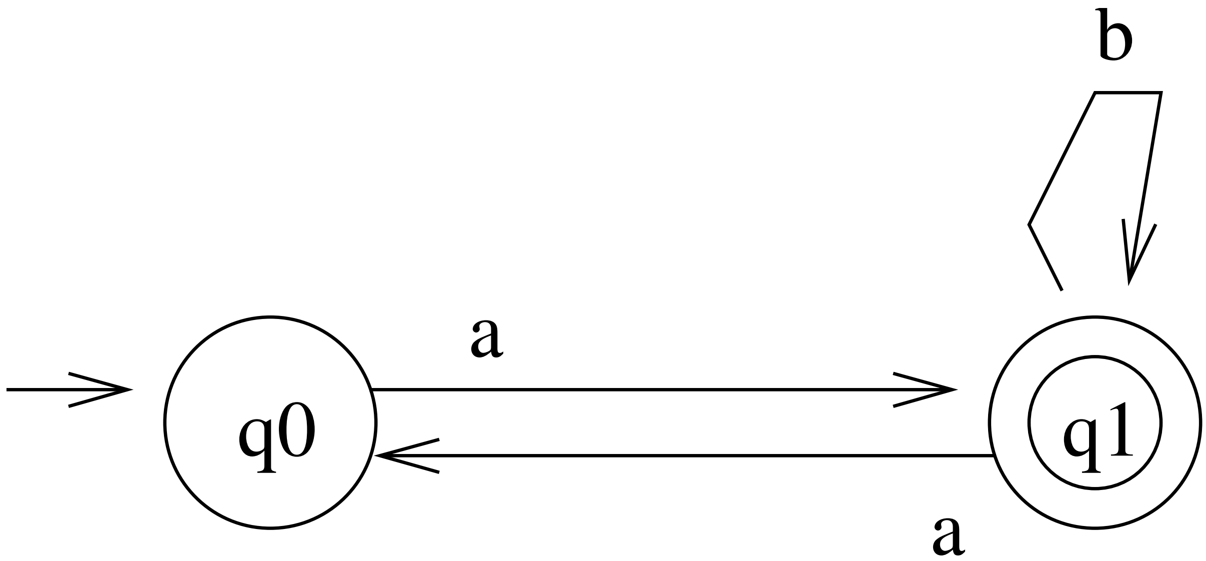
QED.

Example Construct NFA

$$G = (\{S, B\}, \{a, b\}, S, P),\ P =$$
$$S \rightarrow aB \mid bS \mid \lambda$$
$$B \rightarrow aS \mid bB$$



Example: Convert to reg grm



$q_0 \rightarrow a q_1$

$q_1 \rightarrow b q_1 \mid a q_0 \mid \lambda$