Automata and Languages

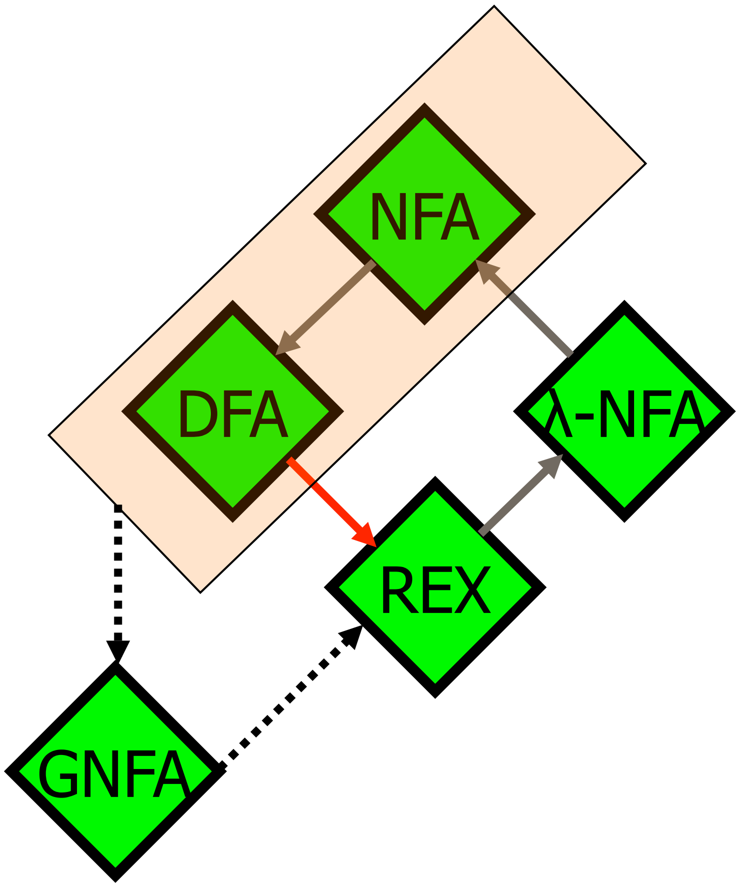
Prof. Mohamed Hamada

Software Engineering Lab.
The University of Aizu
Japan

Today's Topics

- **DFA to Regular Expression**
- **GFNA**
- **DFA→GNFA**
- **GNFA → RE**
- **DFA → RE**
- **Examples**

DFA → RE



GNFA

Definition

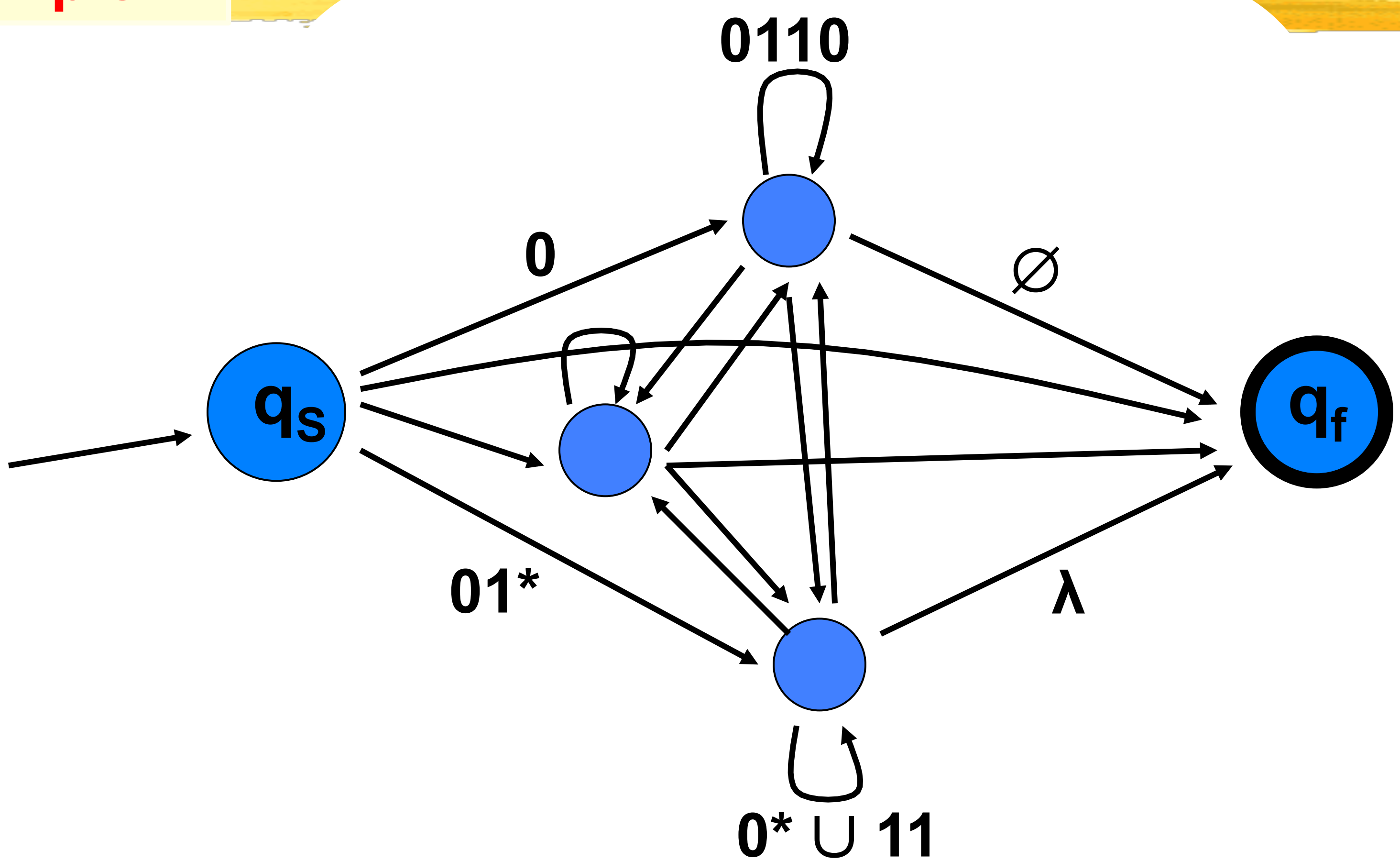
A ***generalized nondeterministic finite automaton*** (**GNFA**) is a graph whose edges are labeled by regular expressions, with a unique start state with in-degree 0, and a unique final state with out-degree 0.

A string *u* is said to ***label*** a path in a GNFA, if it is an element of the language generated by the regular expression which is gotten by concatenating all labels of edges traversed in the path.

The ***language accepted*** by a GNFA consists of all the accepted strings of the GNFA.

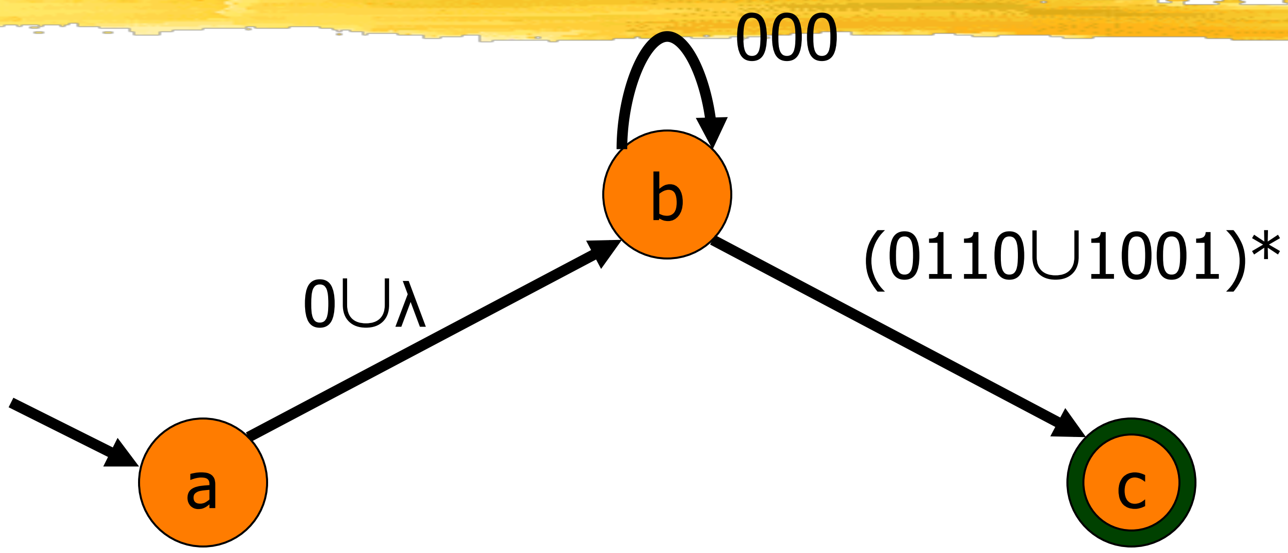
GNFA

Example 1



GNFA

Example 2

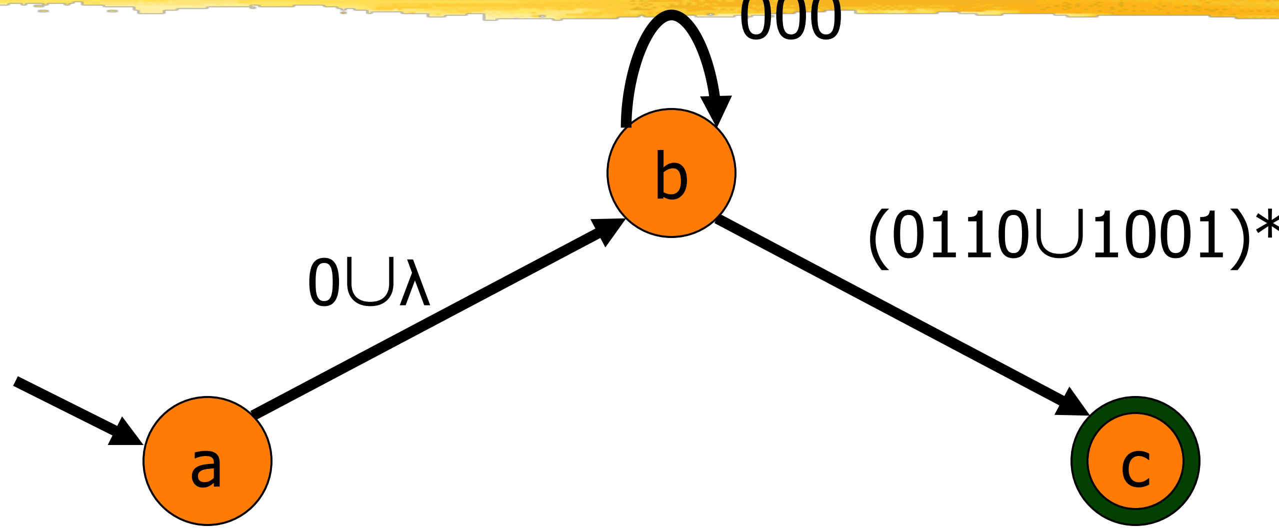


This is a GNFA because edges are labeled by REX's, start state has no in-edges, and the *unique final* state has no out-edges.

Q: Is 00000100101100110 accepted?

GNFA

Example 2



A: 00000010010110 0110 is accepted. The path given by

$$a \xrightarrow[\lambda]{} b \xrightarrow[000]{} b \xrightarrow[000]{} b \xrightarrow[100101100110]{} c$$

proves this fact.

(The last label results from

100101100110 ∈ (0110∪1001)*)

GNFA

Definition

A Generalized nondeterministic finite automaton (GNFA) is defined by $M=(Q, \Sigma, \delta, q_{start}, q_{final})$ with

- Q finite set of states
- Σ the input alphabet
- q_{start} the start state with 0 in-degree
- q_{final} the accept state with 0 out-degree

- $\delta:(Q\setminus\{q_{final}\})\times(Q\setminus\{q_{start}\}) \rightarrow R$ the transition function

- (R is the set of regular expressions over Σ)

DFA→GNFA

Theorem

For every DFA M, there exist an equivalent GNFA M'

DFA → GNFA

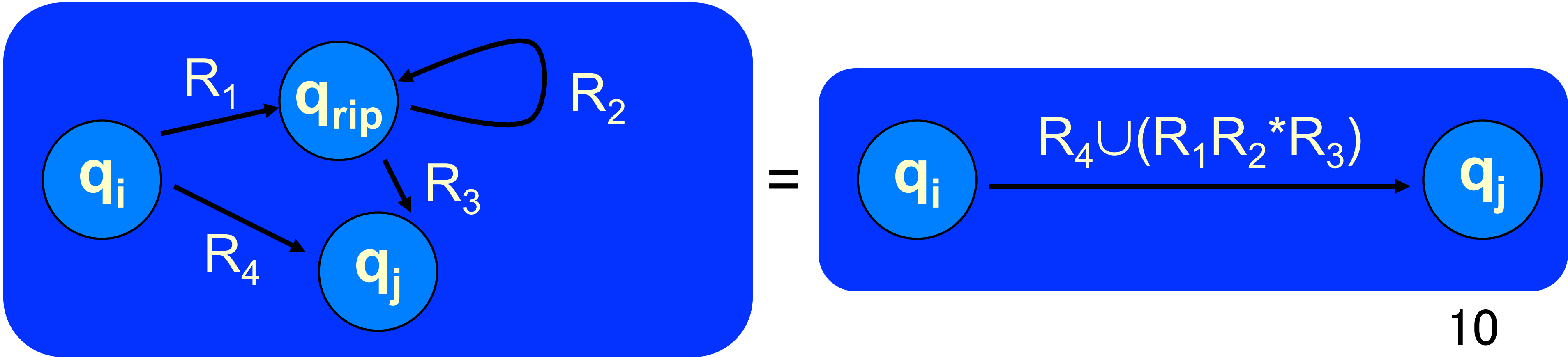
Remove Internal state of GNFA

If the GNFA M has more than 2 states, 'rip' internal q_{rip} to get equivalent GNFA M' by:

- Removing state q_{rip}: $Q' = Q \backslash \{q_{rip}\}$
- Changing the transition function δ by

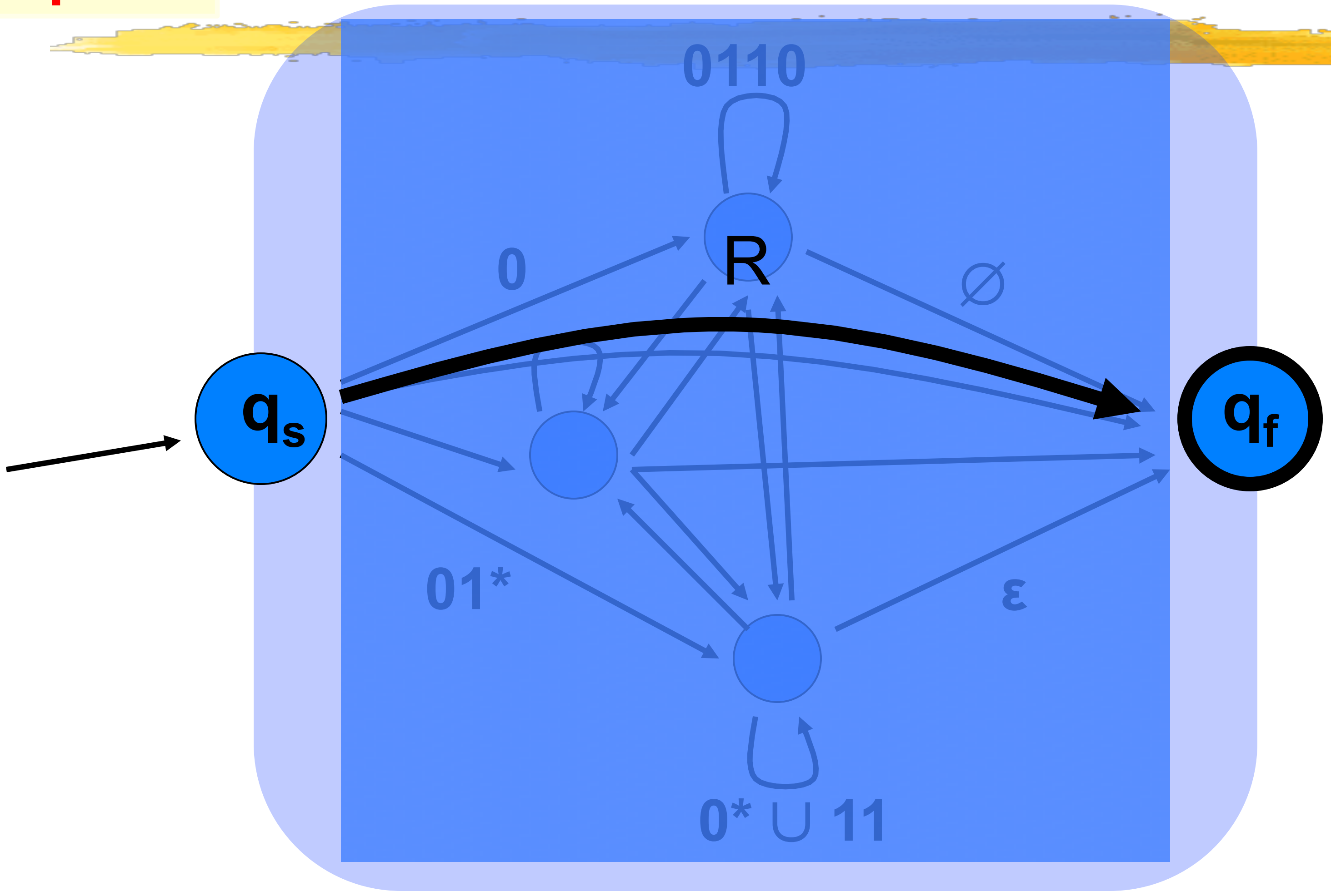
$$\delta'(q_i,q_j) = \delta(q_i,q_j) \cup (\delta(q_i,q_{rip})(\delta(q_{rip},q_{rip}))^*\delta(q_{rip},q_j))$$

for every $q_i \in Q' \backslash \{q_{finalt}\}$ and $q_j \in Q' \backslash \{q_{start}\}$



GNFA→RE

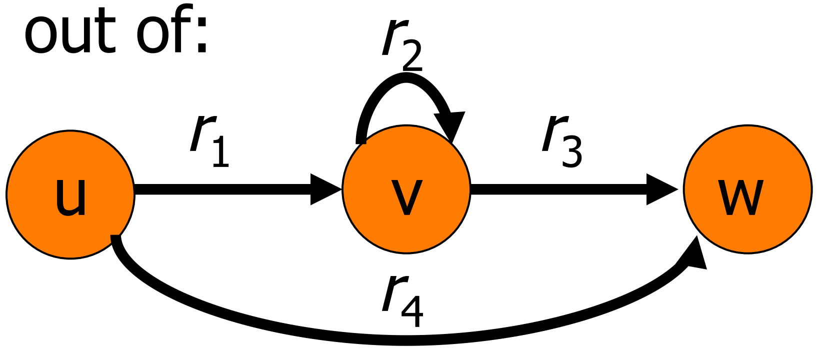
Example



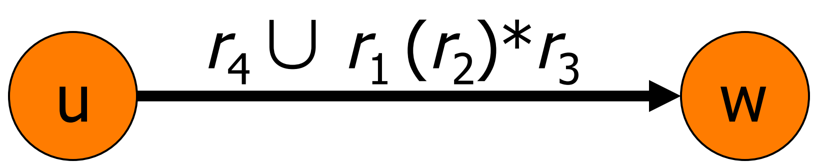
GNFA→RE

Ripping out

Ripping out is done as follows. If you want to rip the middle state *v* out of:



then you'll need to recreate all the lost possibilities from *u* to *w*. I.e., to the current REX label r_4 of the edge (*u*,*w*) you should add the concatenation of the (*u*,*v*) label r_1 followed by the (*v*,*v*)-loop label r_2 repeated arbitrarily, followed by the (*v*,*w*) label r_3. The new (*u*,*w*) substitute would therefore be:



DFA→RE

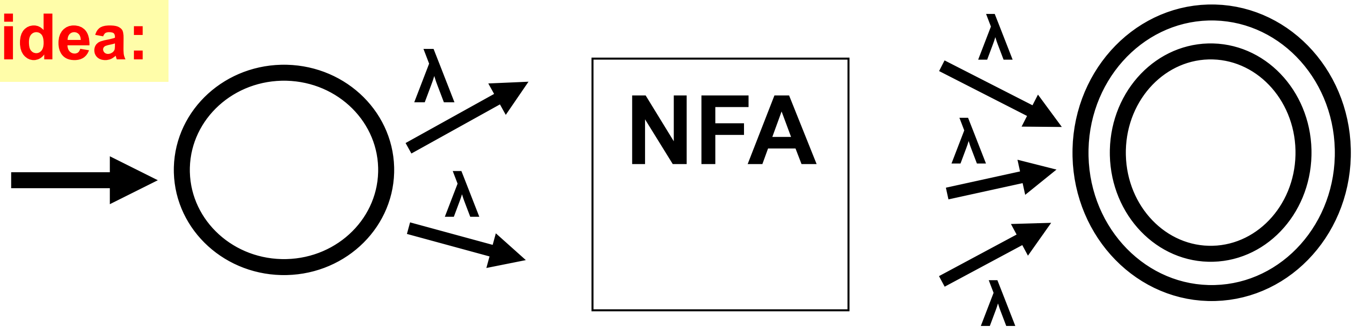
Theorem

For every (NFA) DFA *M*, there exist an equivalent RE *R*, with L(M)=L(R)

Proof idea:

1. **Transform an NFA into an equivalent GNFA**
2. **Transform the GNFA into a regular expression by removing states and re-labeling the arrows with regular expressions**

Proof idea:



1. NFA➔GNFA

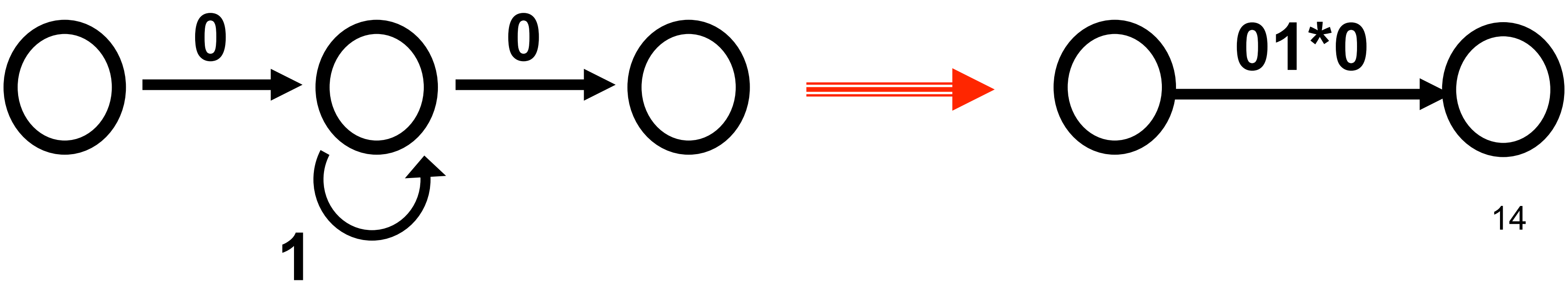
Add unique and distinct start state with 0 in-degree and a final state with 0 out-degree, then connect them to the NFA with λ

2. GNFA➔RE

While machine has more than 2 states:

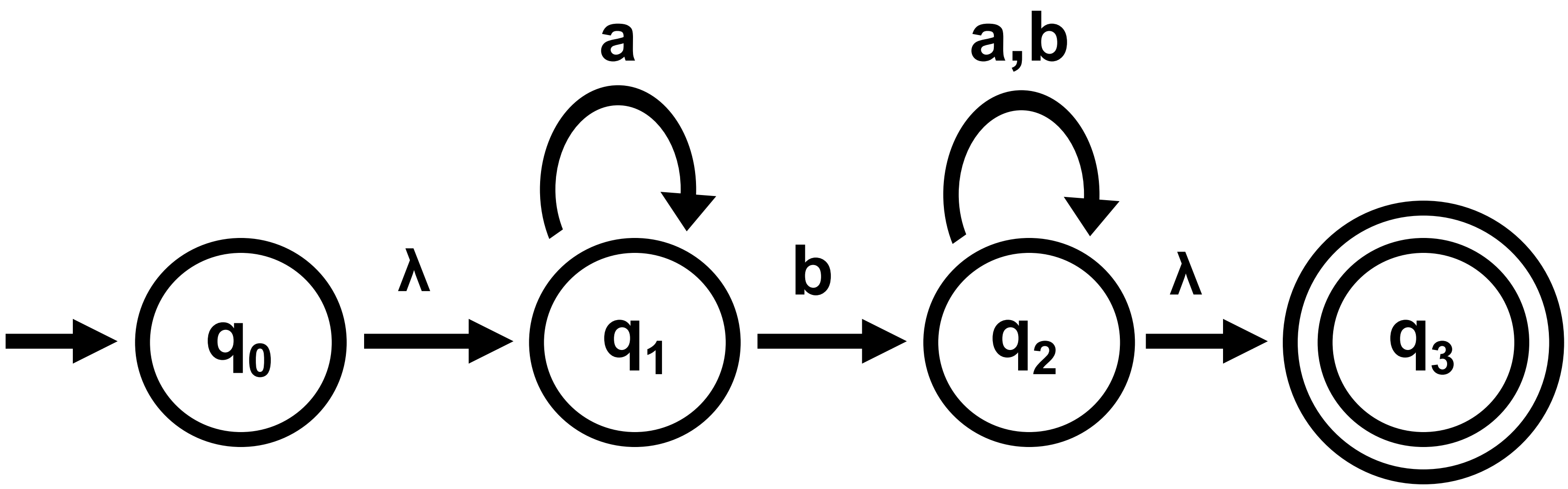
Pick an internal state, rip it out and re-label the arrows with regular expressions to account for the missing state

For Example:



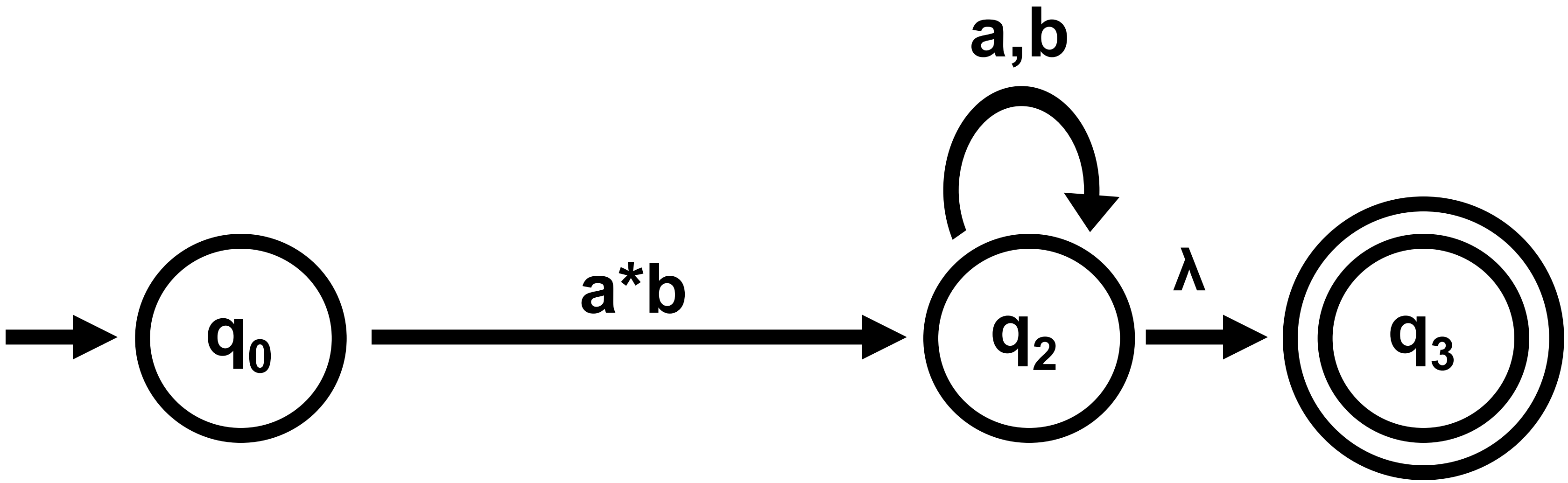
DFA→RE

Example 1



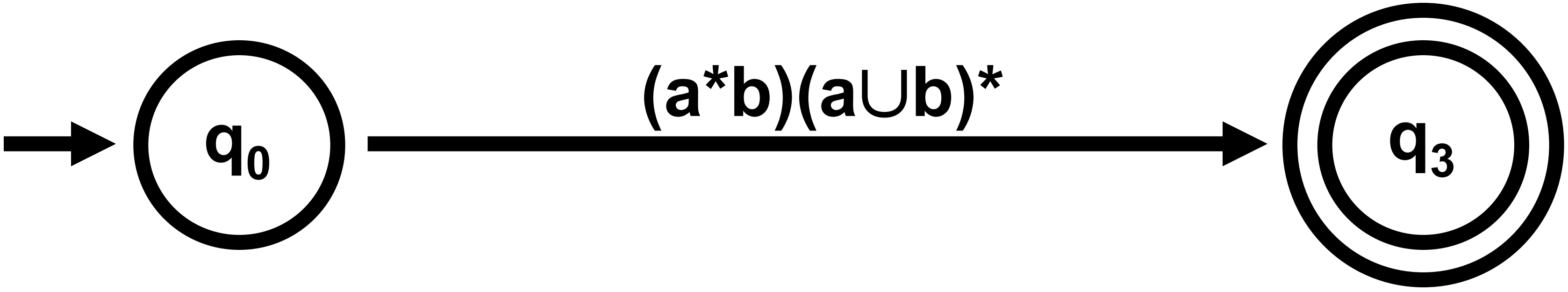
DFA→RE

Example 1



DFA → RE

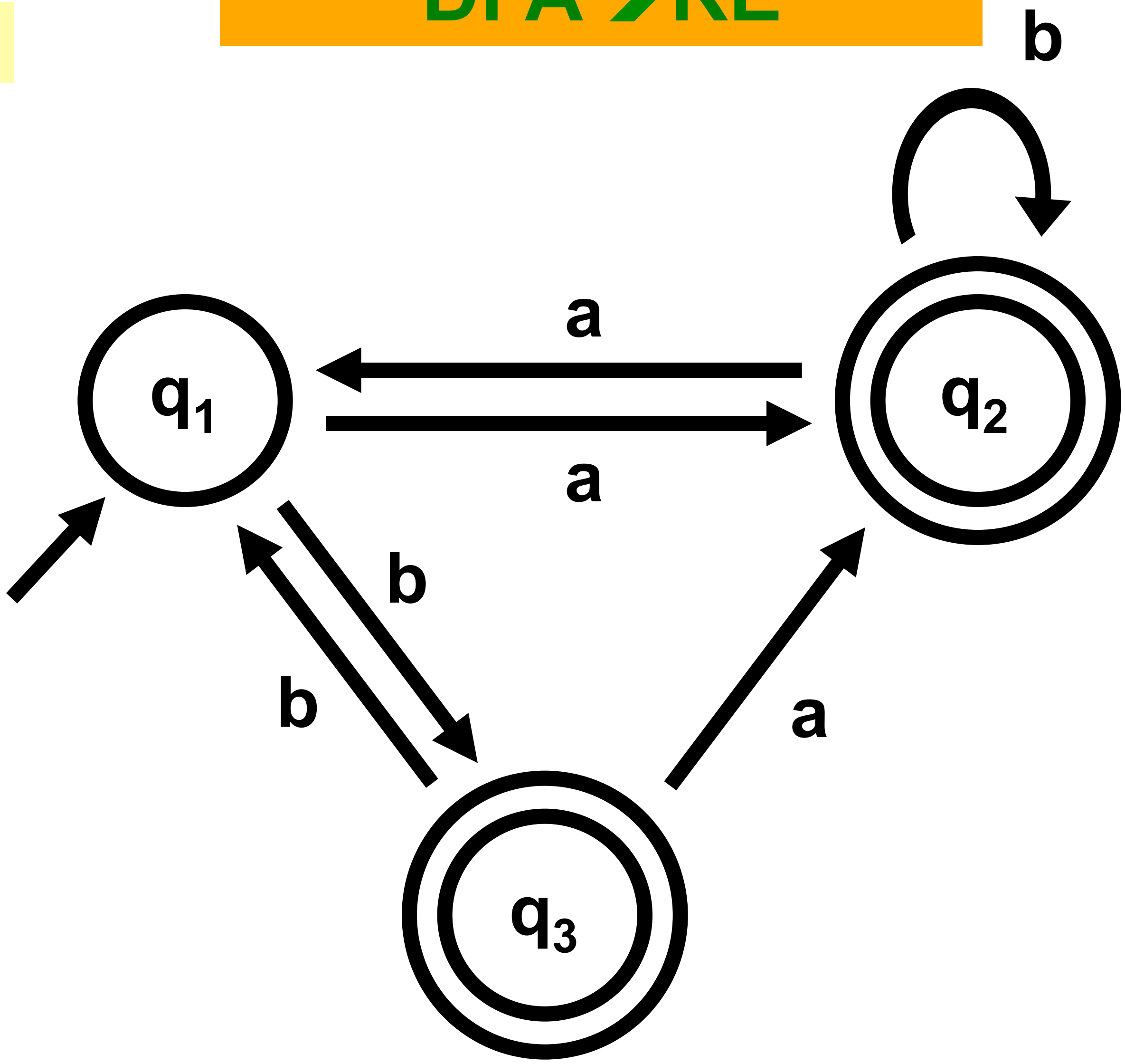
Example 1



$$R(q_0,q_3) = (a^*b)(a \cup b)^*$$

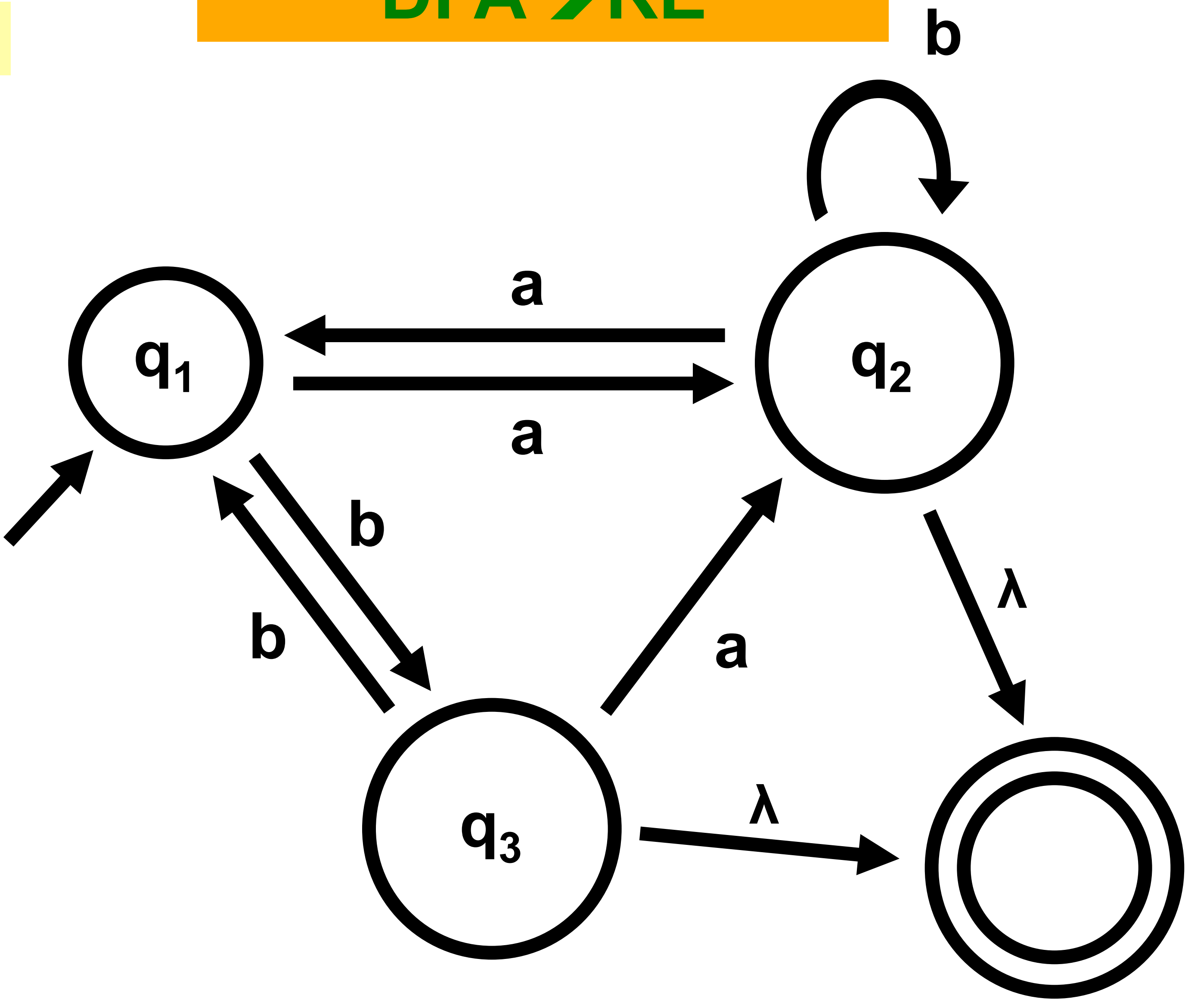
DFA→RE

Example 2



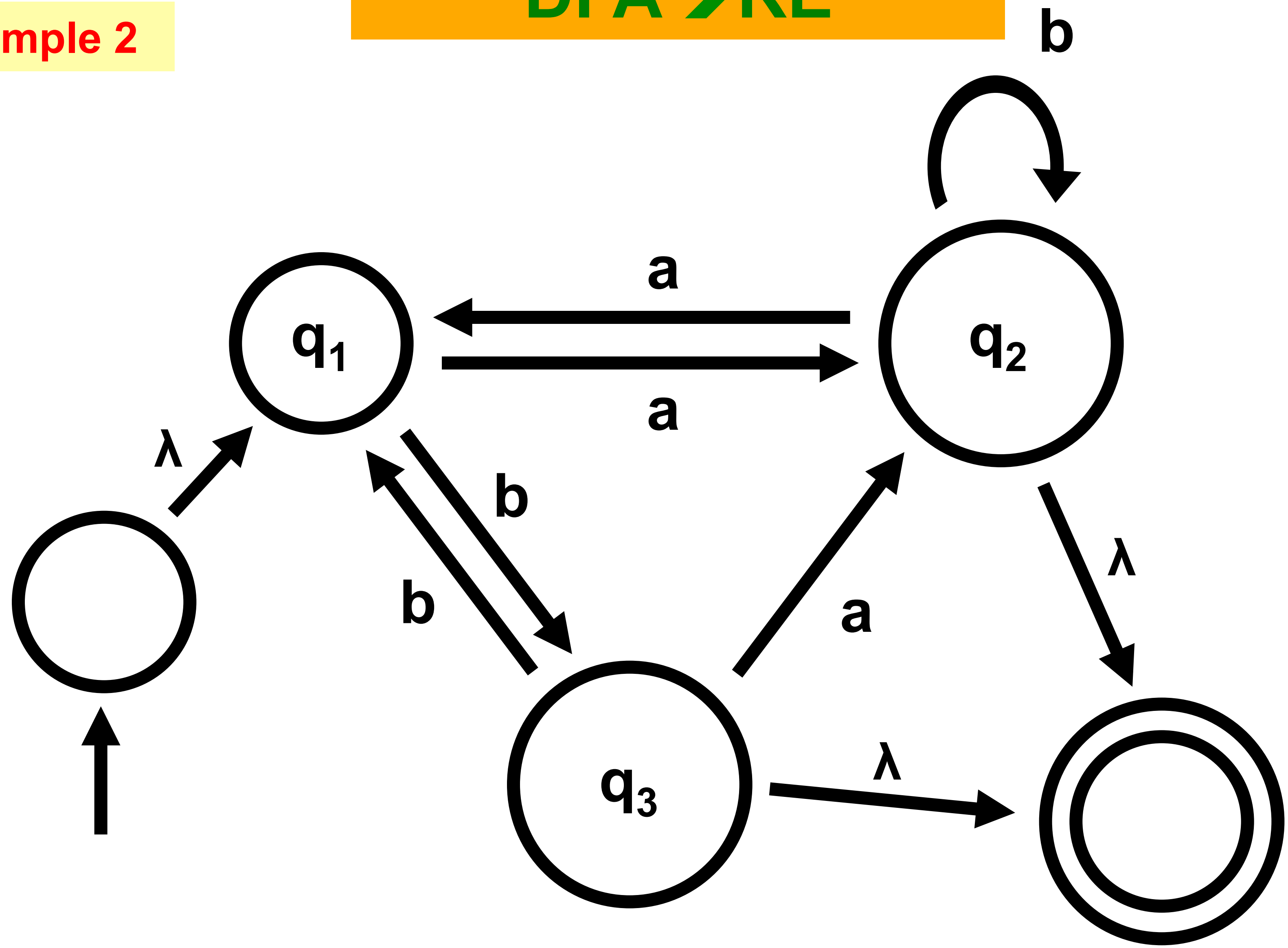
Example 2

DFA→RE



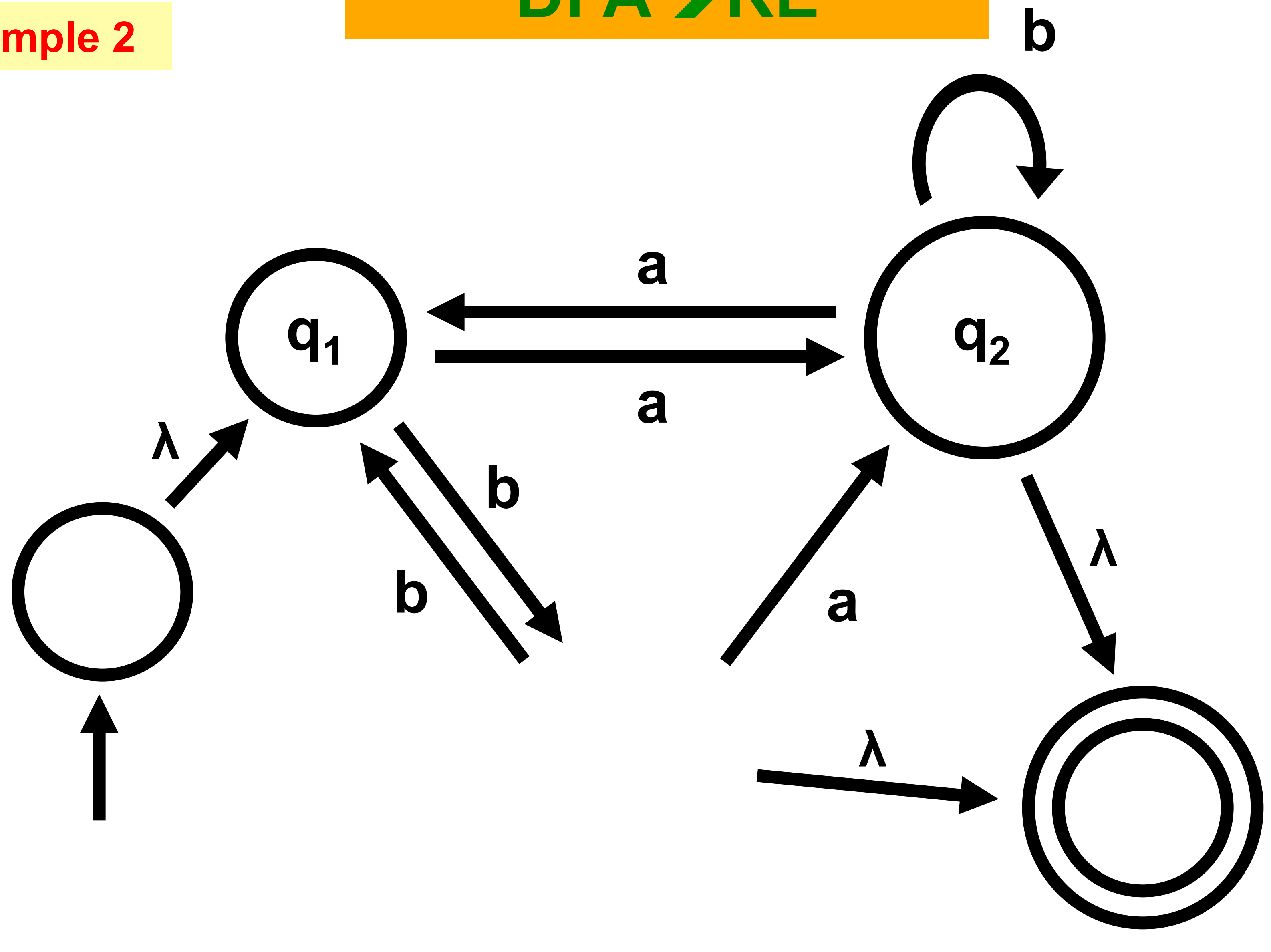
Example 2

DFA→RE



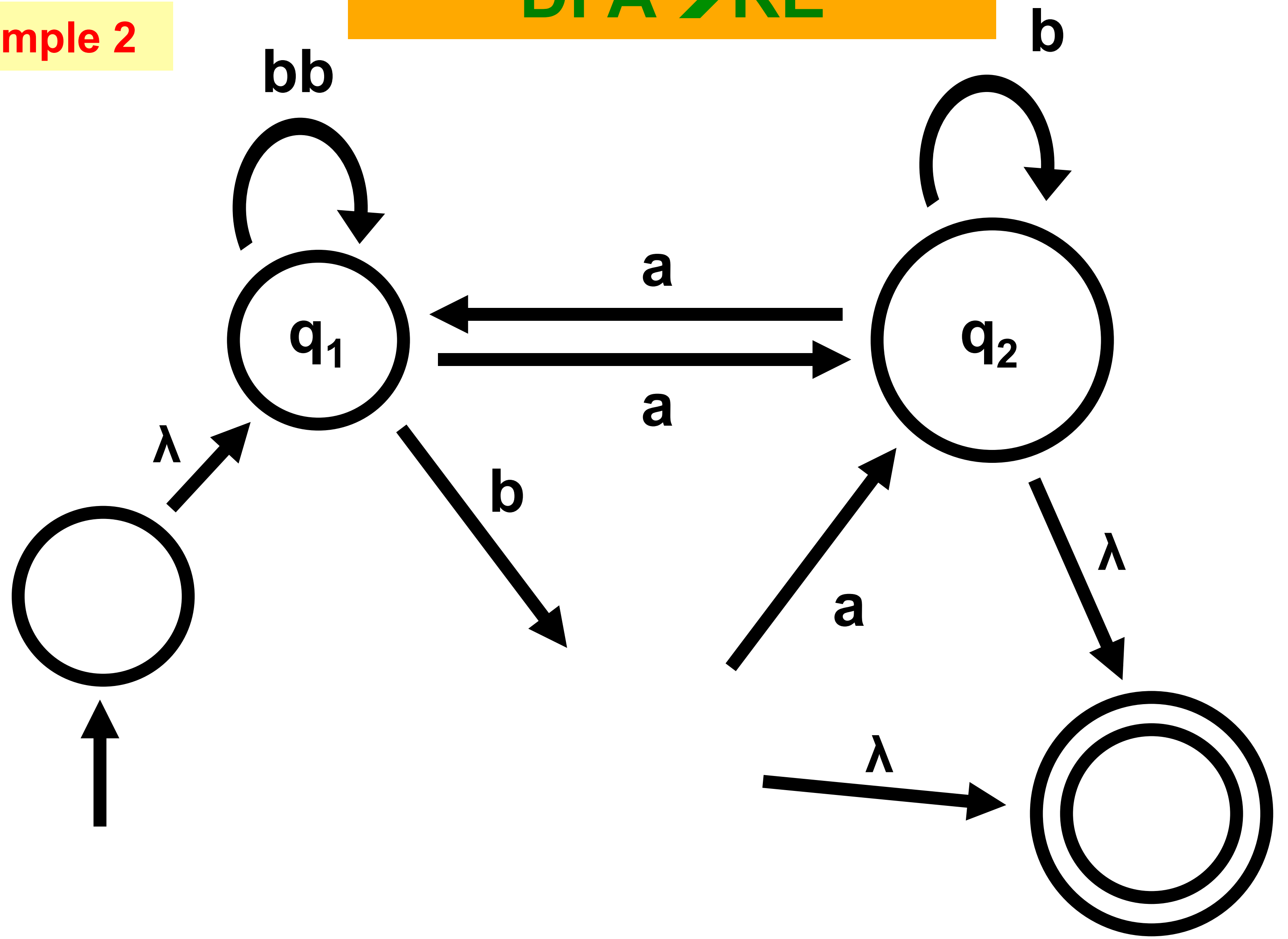
DFA➔RE

Example 2



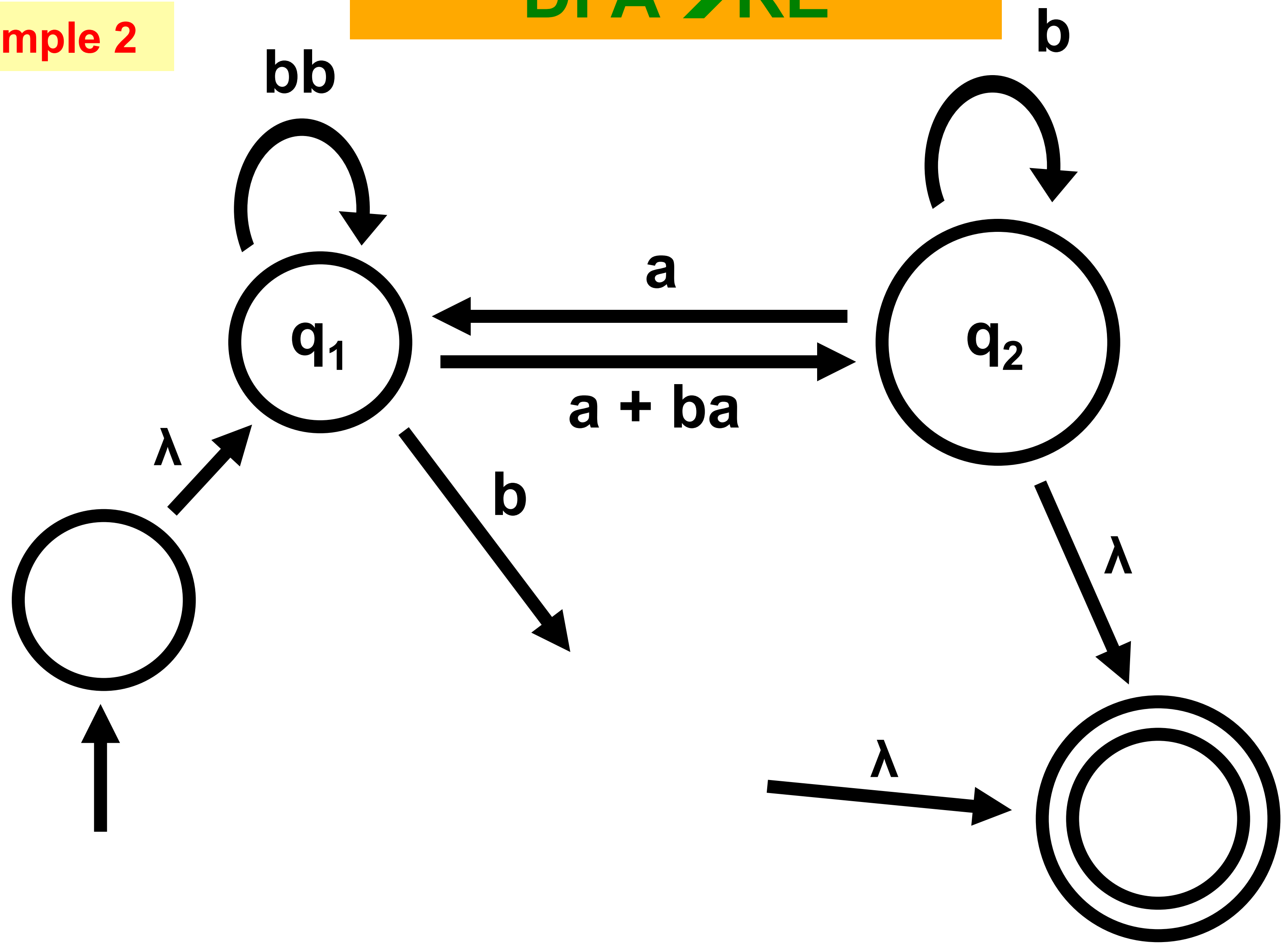
DFA→RE

Example 2



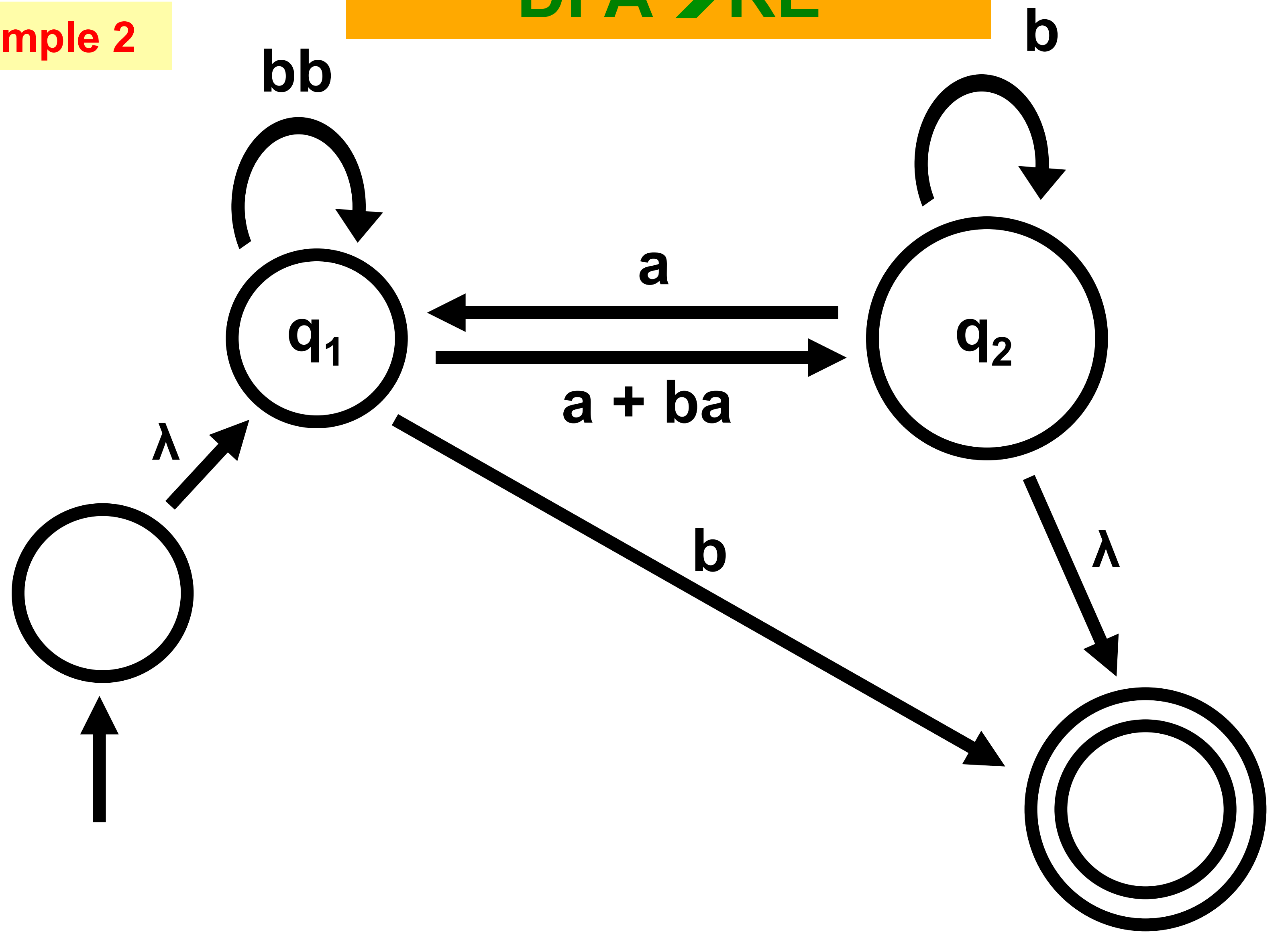
DFA→RE

Example 2



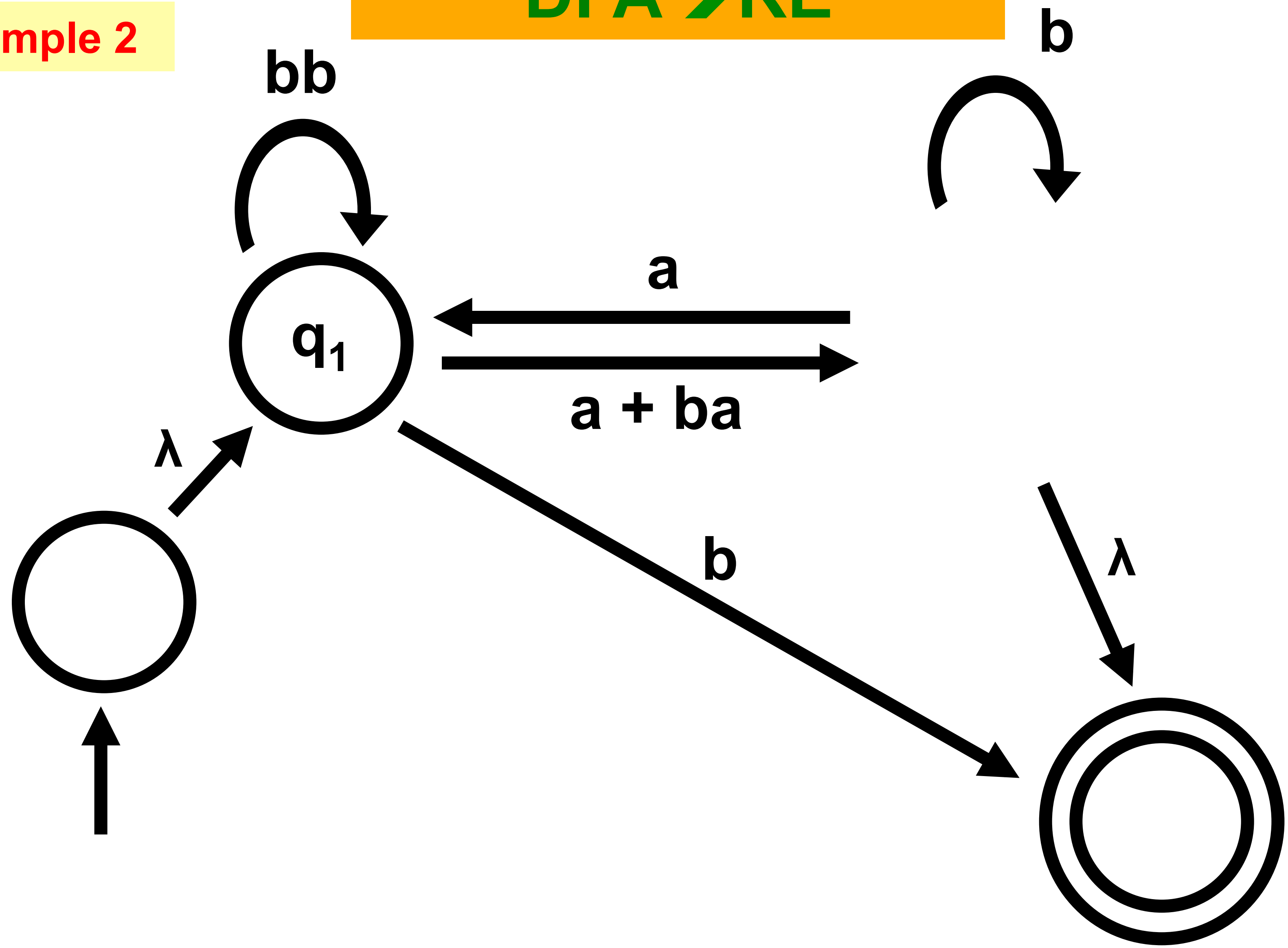
Example 2

DFA→RE



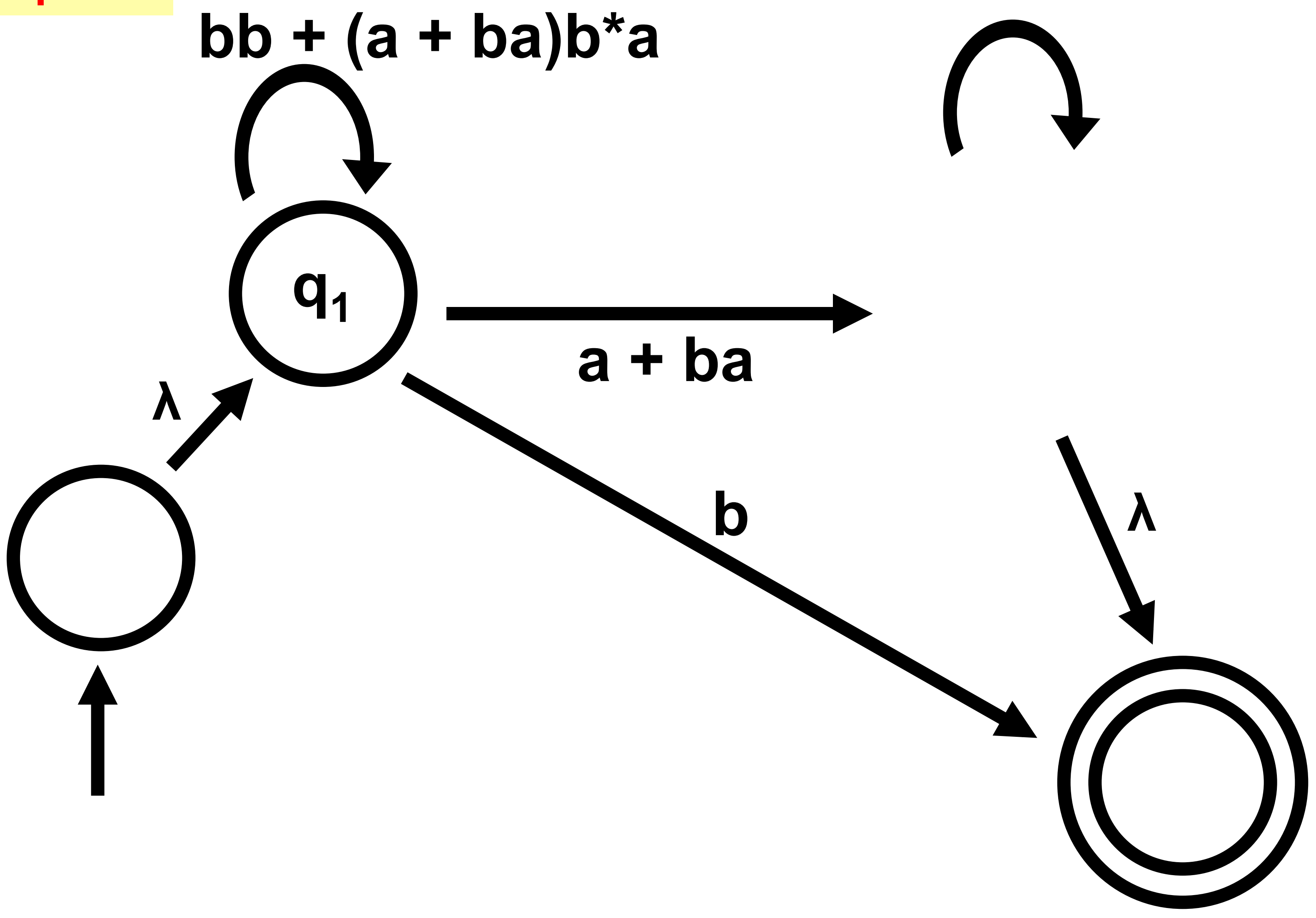
DFA→RE

Example 2



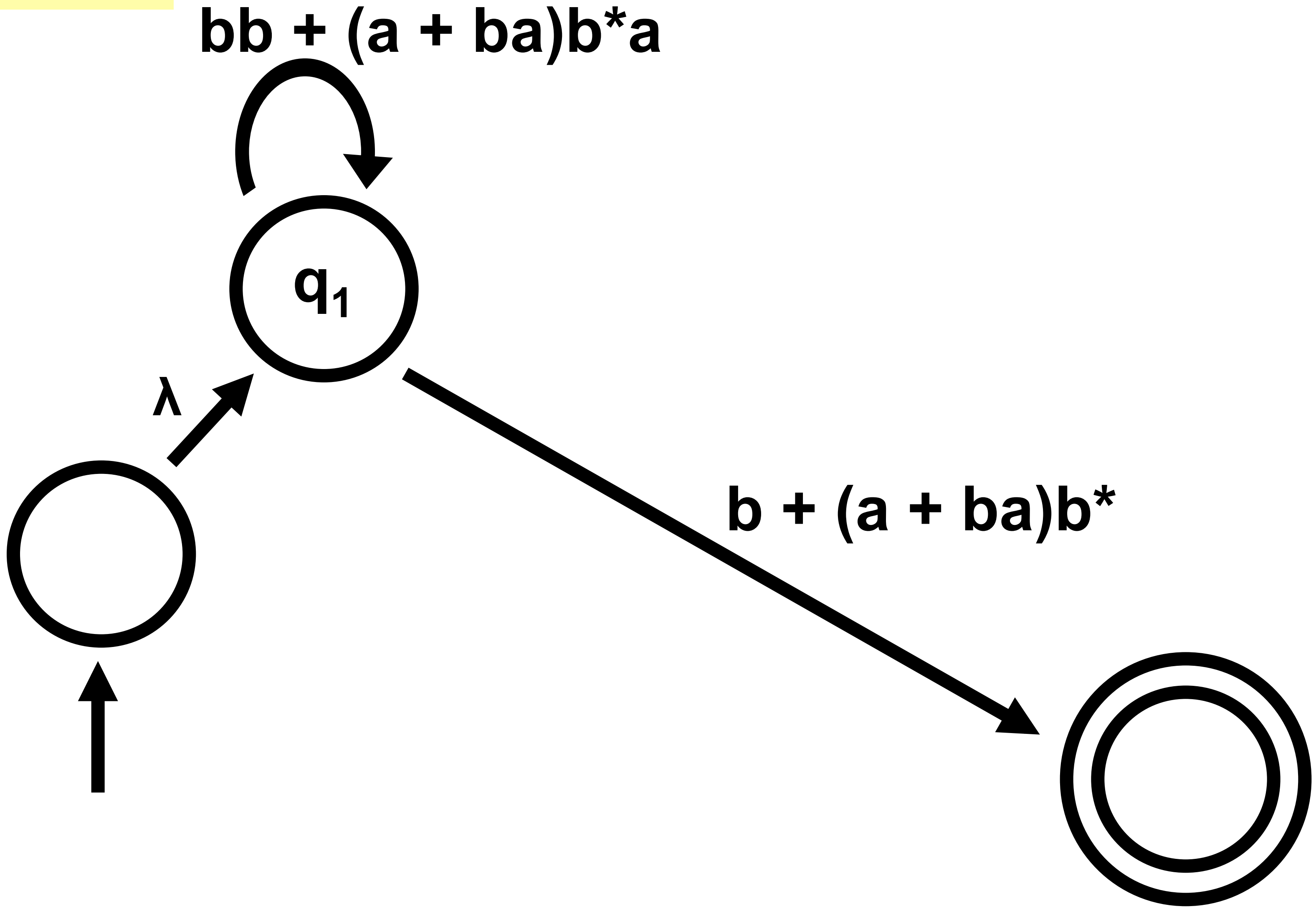
Example 2

DFA→RE



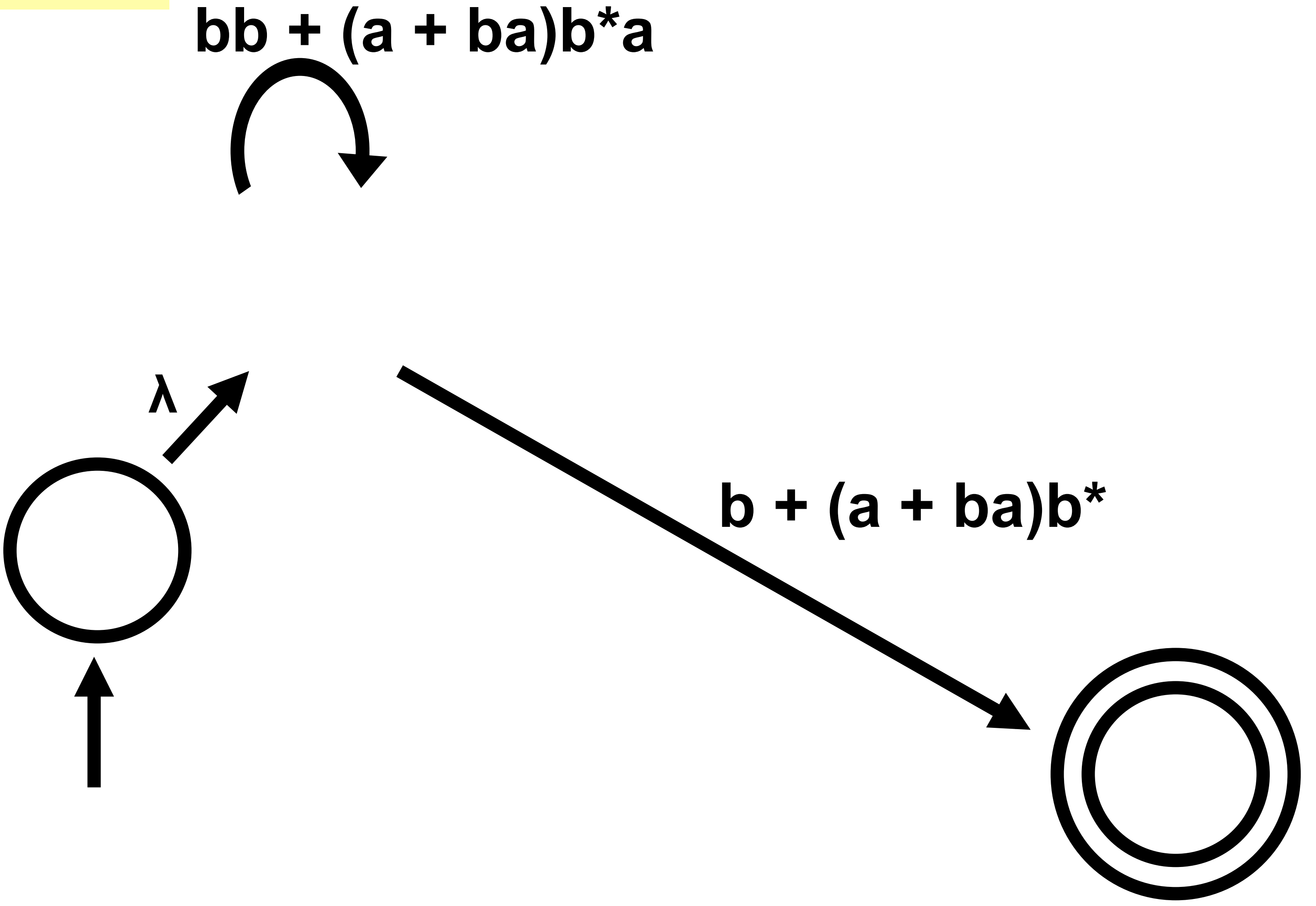
DFA→RE

Example 2



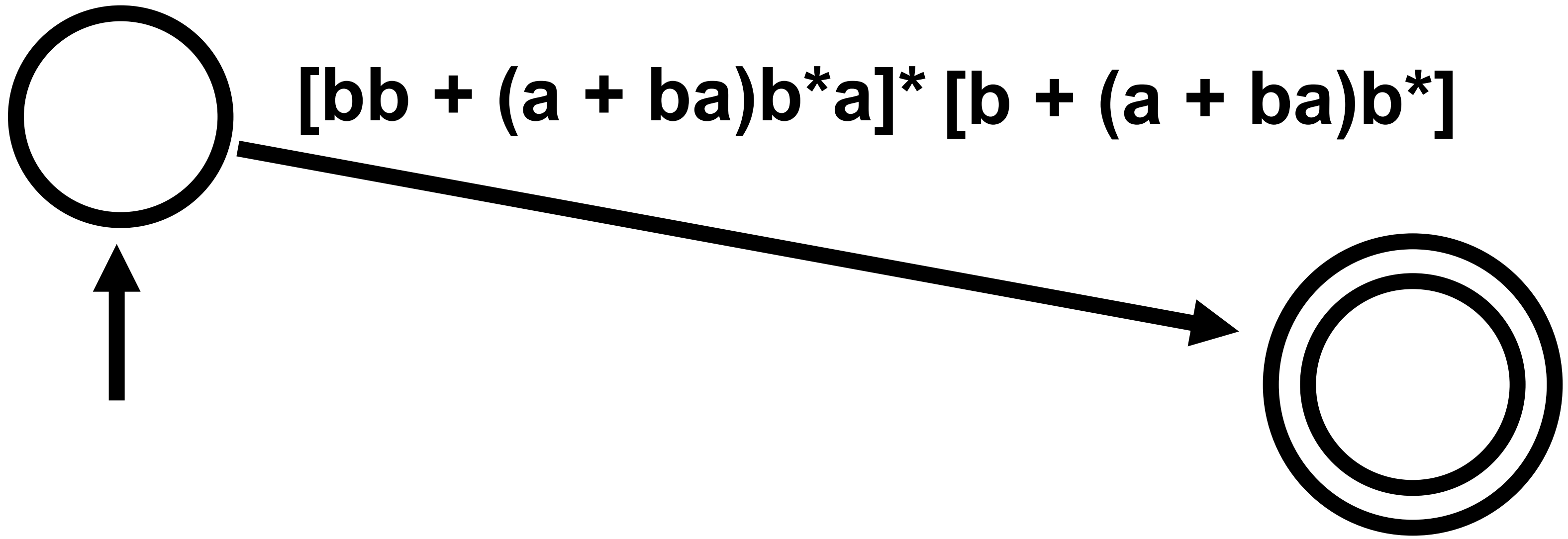
Example 2

DFA→RE



DFA→RE

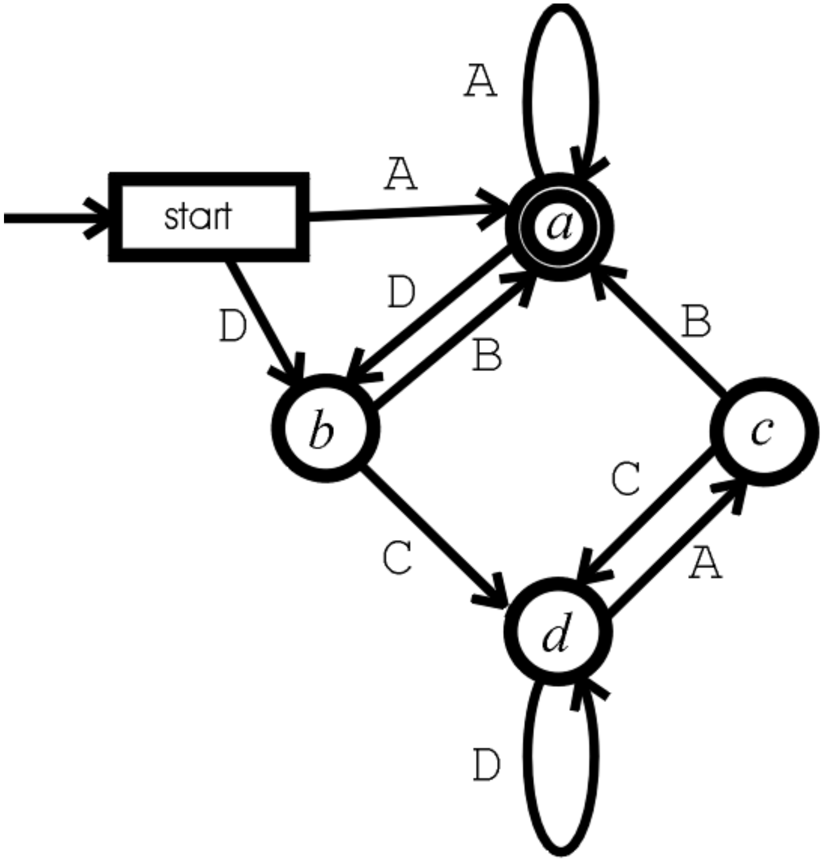
Example 2



DFA→RE

Example 3

Start with:



This is
not a GNFA because
final state, though
unique, has nonzero
out-degree.

First we need
to convert this into
a GNFA.

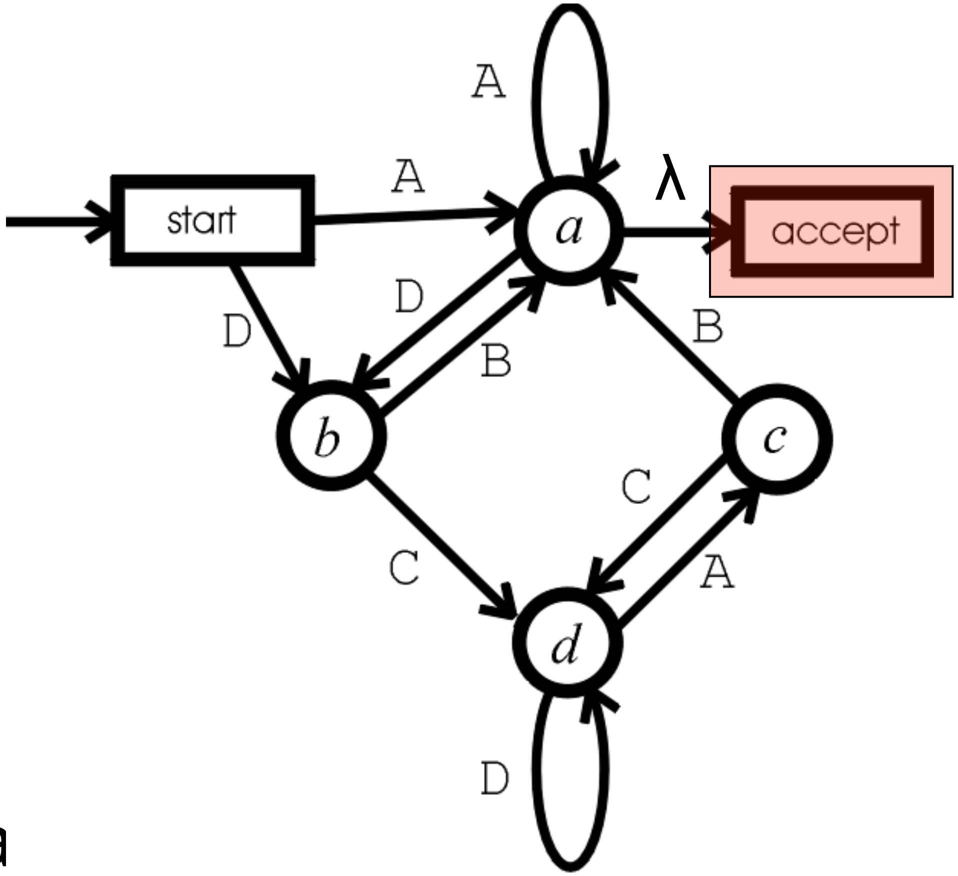
Start state has zero in-degree, so is okay.

Example 3

DFA→RE

Just added an accept state by connecting via λ from the old accept state.

Since the labels are single letters, they are automatically regular expressions so this is a GNFA And we are ready to start ripping out interior states.



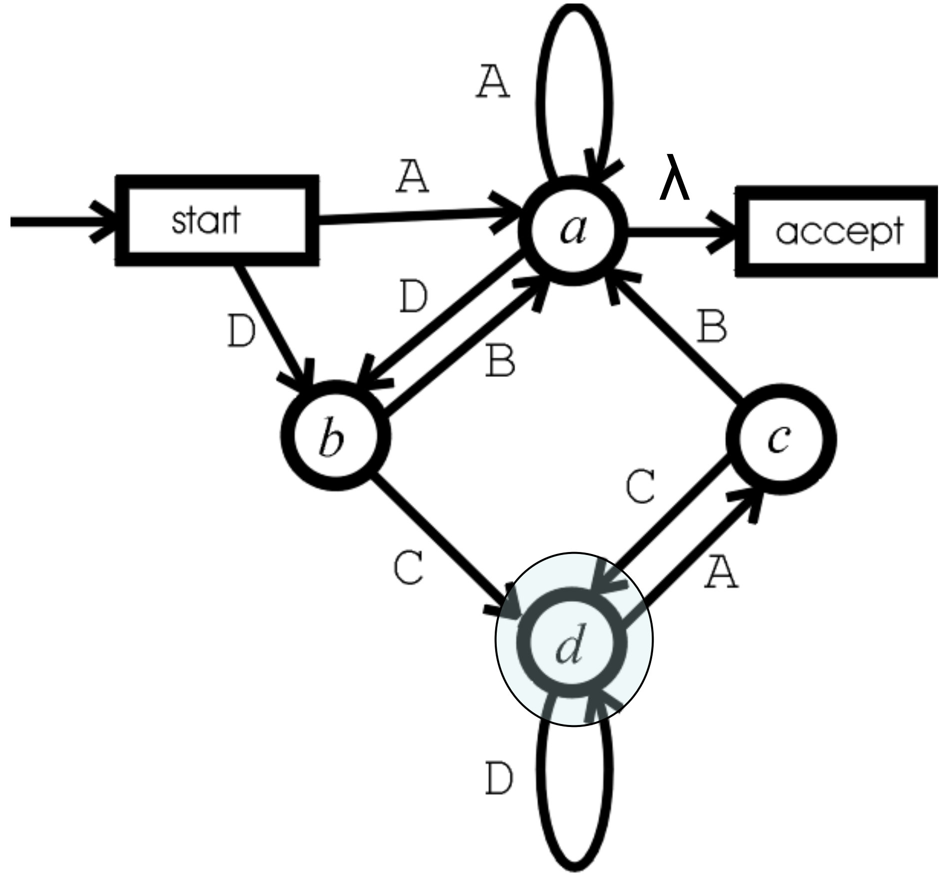
Example 3

DFA→RE

Now, let' s
rip out *d*.

Q1: What will be the label of (*b*,*c*)?

Q2: What will be the label of (*c*,*c*)?

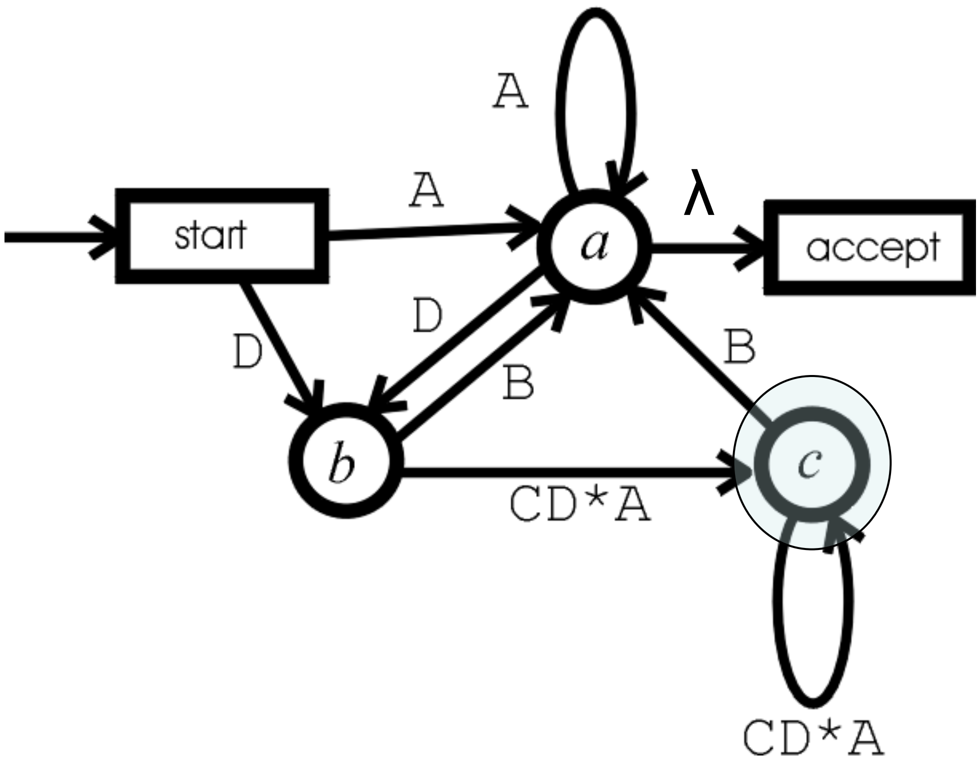


DFA➔RE

Example 3

A1: (*b*,*c*): **CD*A**

A2: (*c*,*c*): **CD*A**



Next, rip out *c*.

Q: What will be the label of (*b*,*a*)?

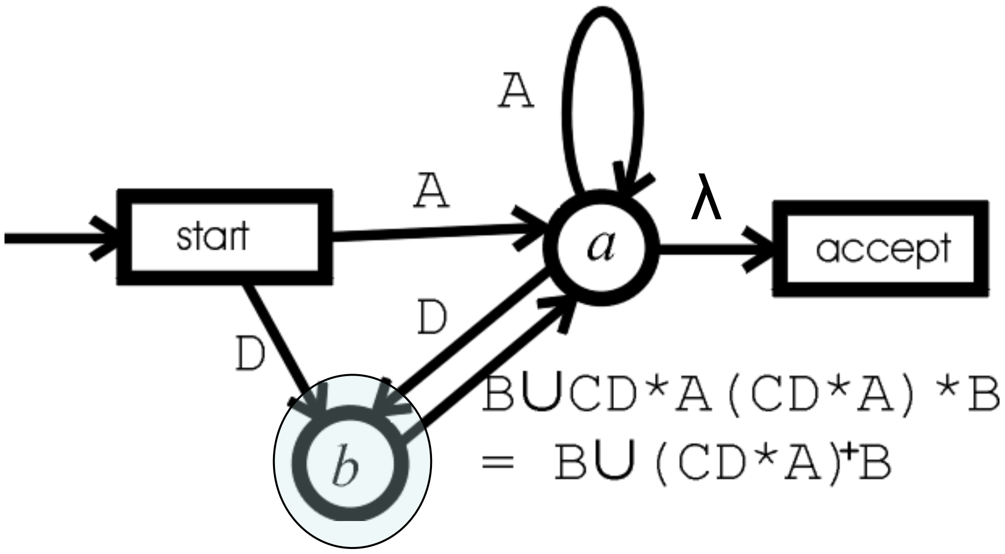
DFA→RE

A:

B∪(CD*A)(CD*A)*B

which simplifies to

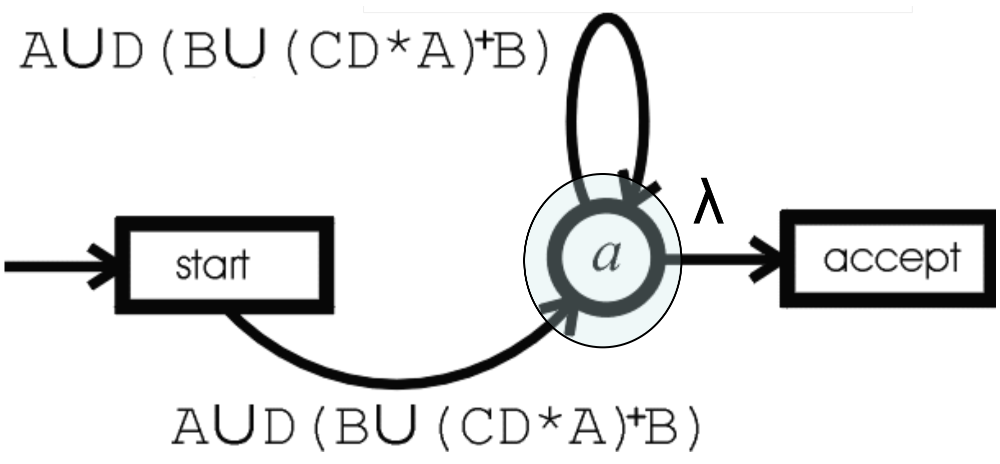
B∪(CD*A)⁺B



Next, rip out *b.*

Example 3

DFA➔RE



Finally, rip out *a* :

Example 3

DFA→RE

The resulting REX

Is the unique

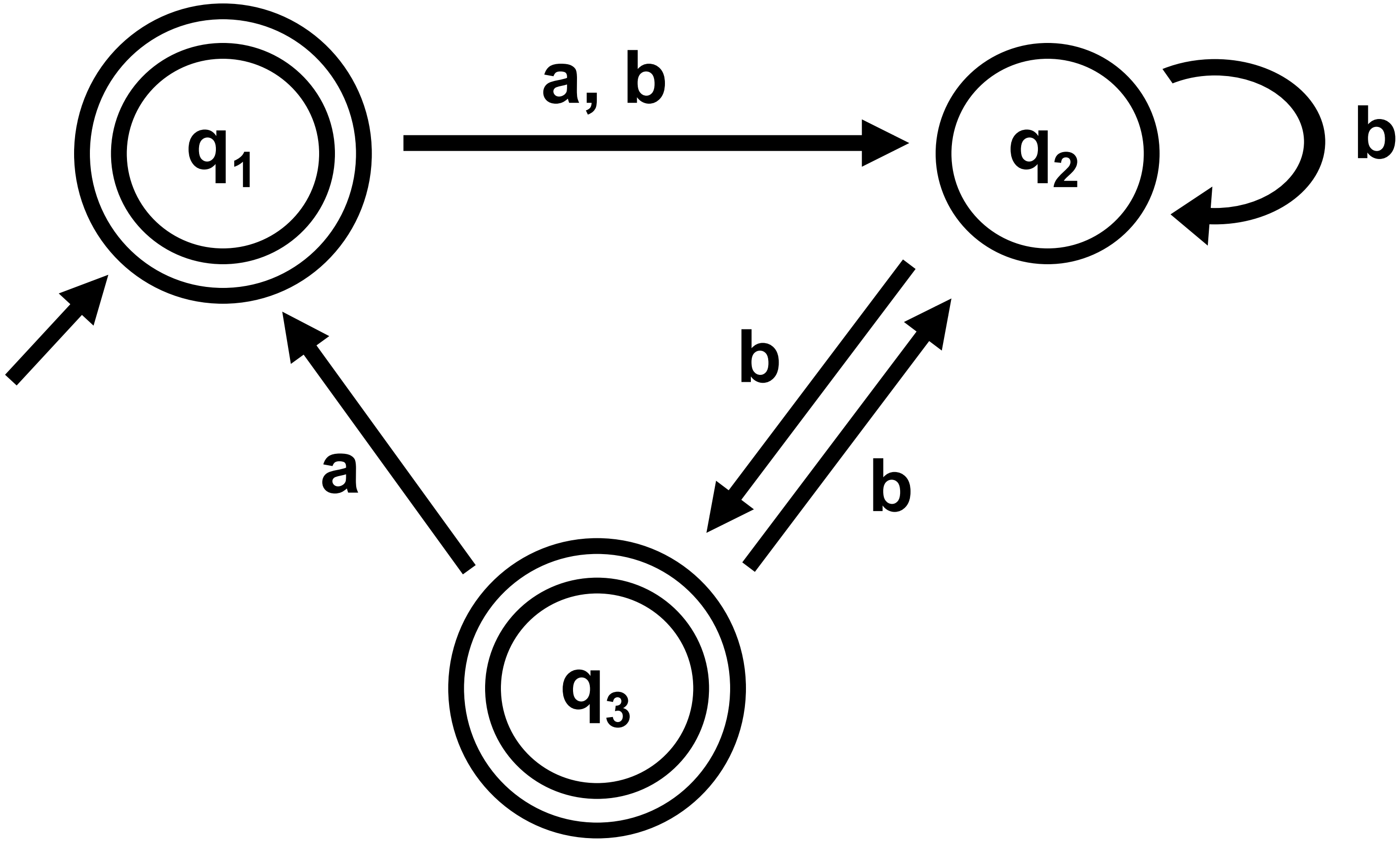
label

(A∪D(B∪(CD*A)$^+$B))$^+$

DFA→RE

Example 4

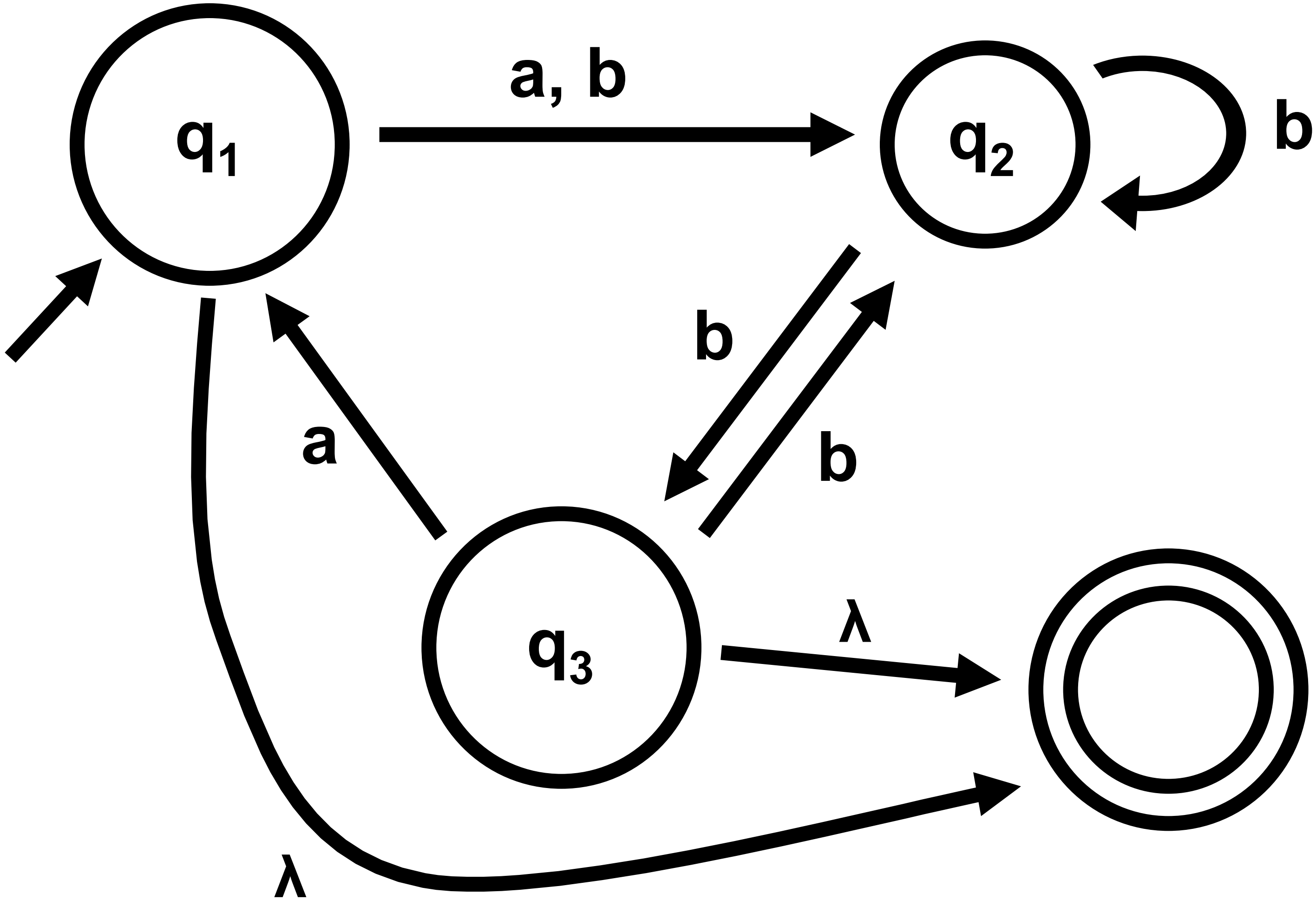
Convert the following NFA into a regular expression?



DFA→RE

Example 4

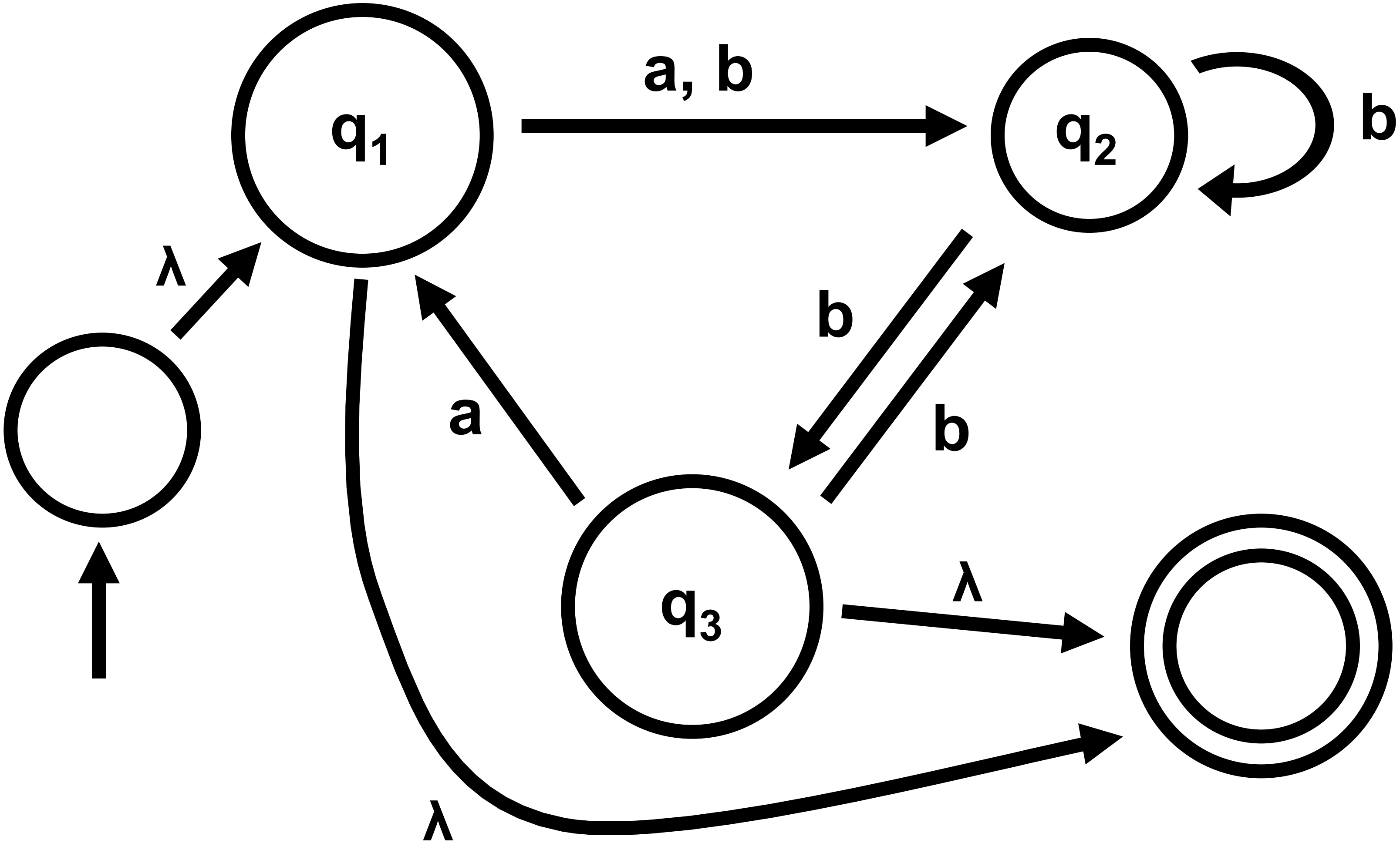
Convert the following NFA into a regular expression?



Example 4

DFA→RE

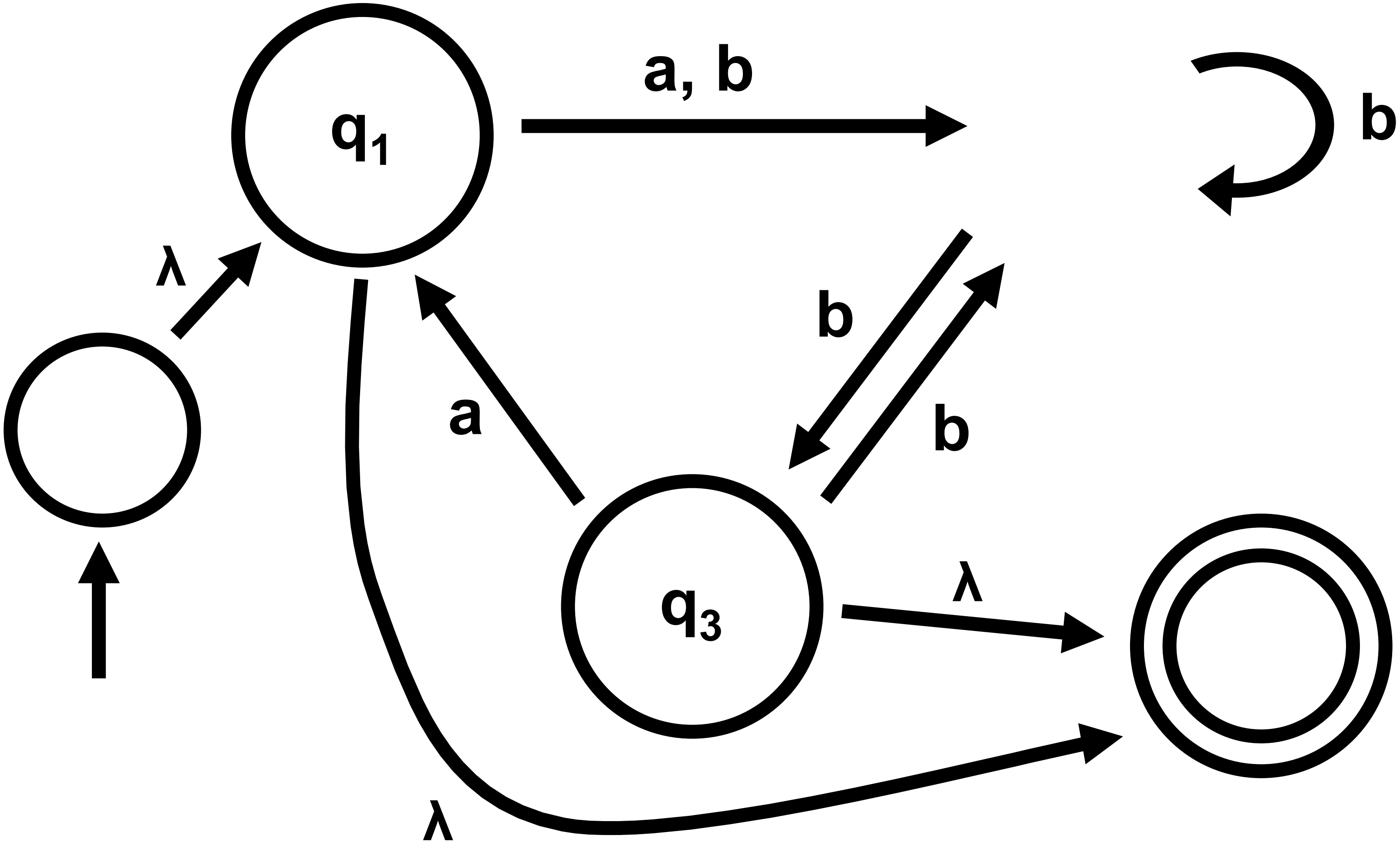
Convert the following NFA into a regular expression?



DFA→RE

Example 4

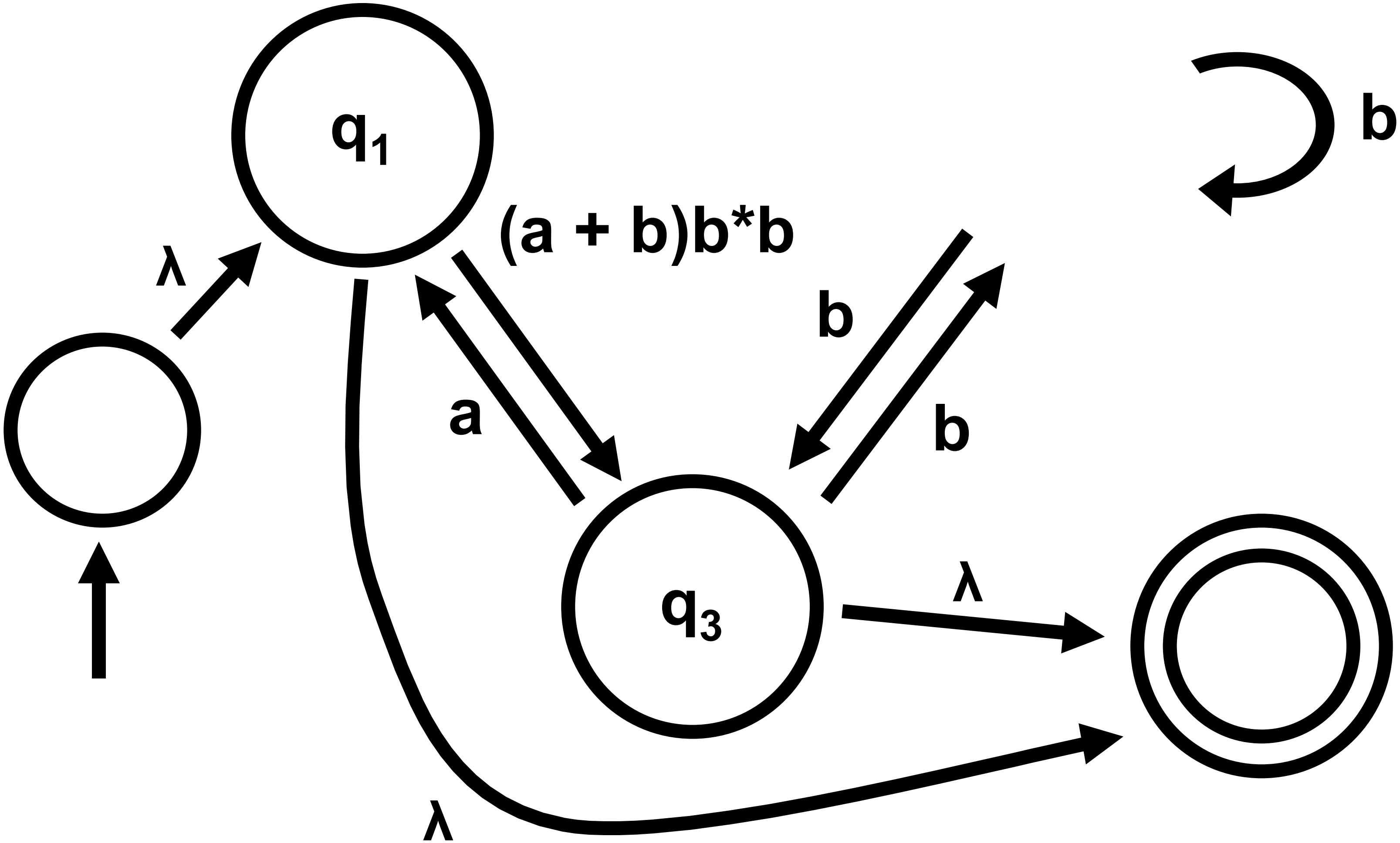
Convert the following NFA into a regular expression?



DFA→RE

Example 4

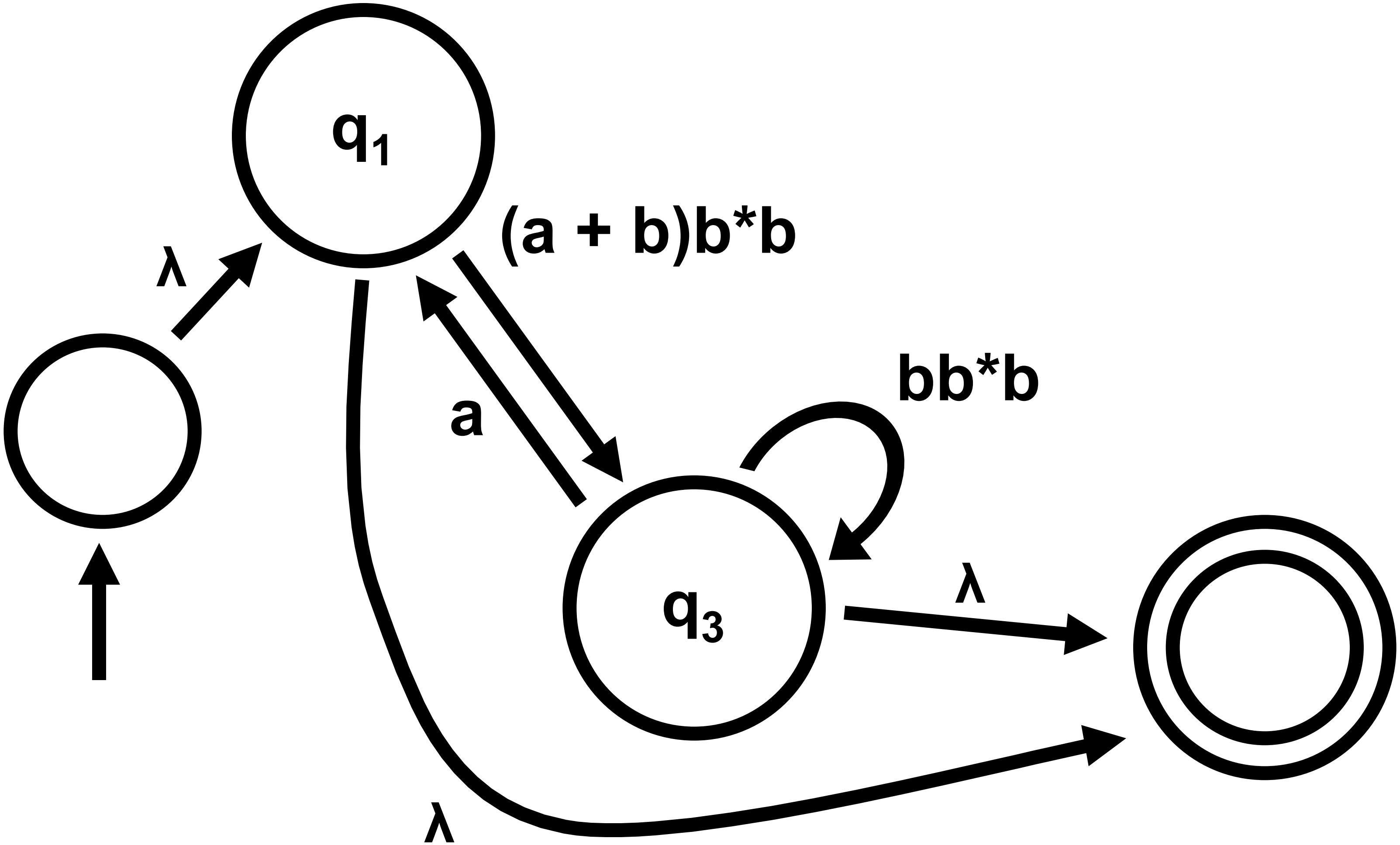
Convert the following NFA into a regular expression?



DFA→RE

Example 4

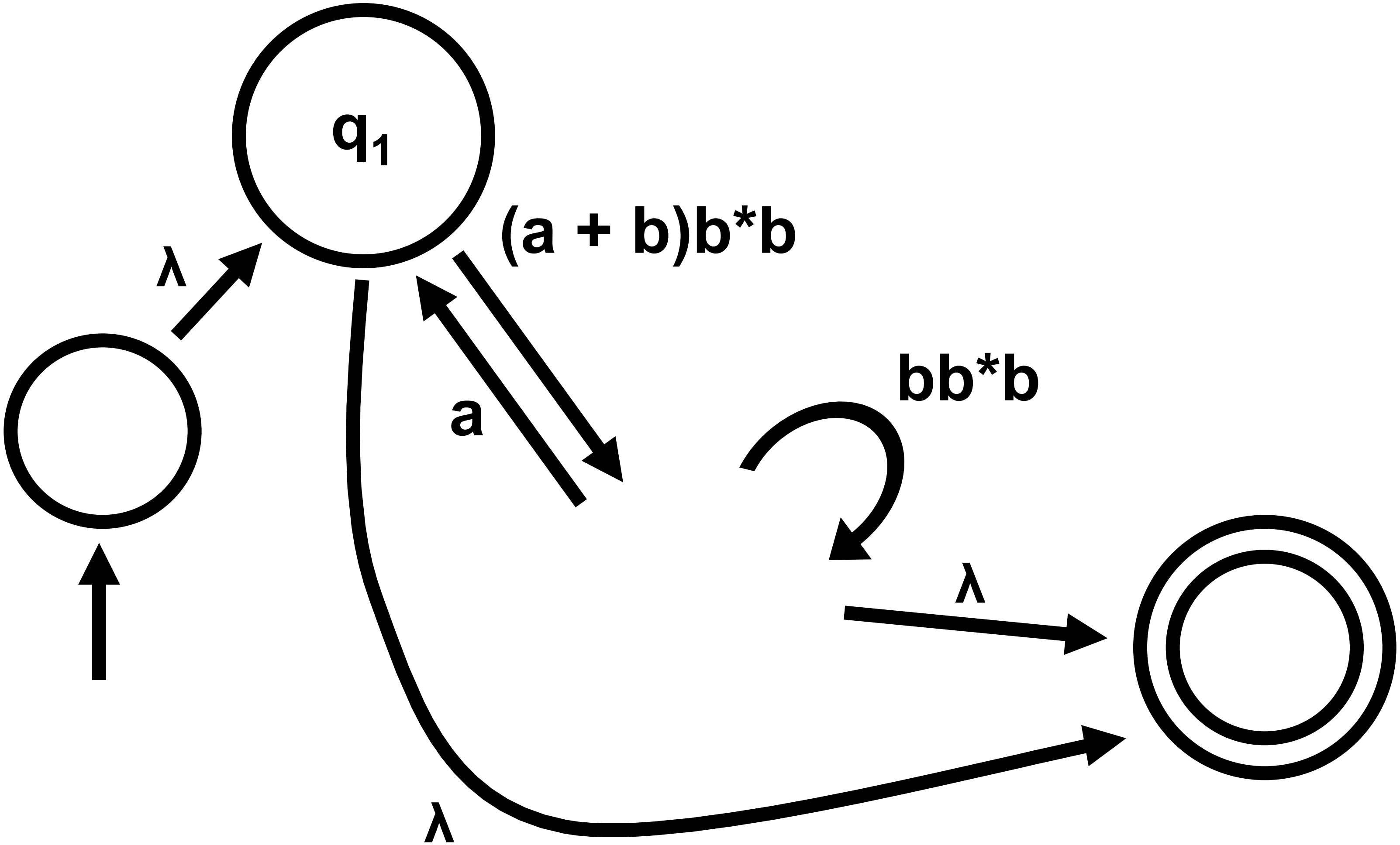
Convert the following NFA into a regular expression?



Example 4

DFA→RE

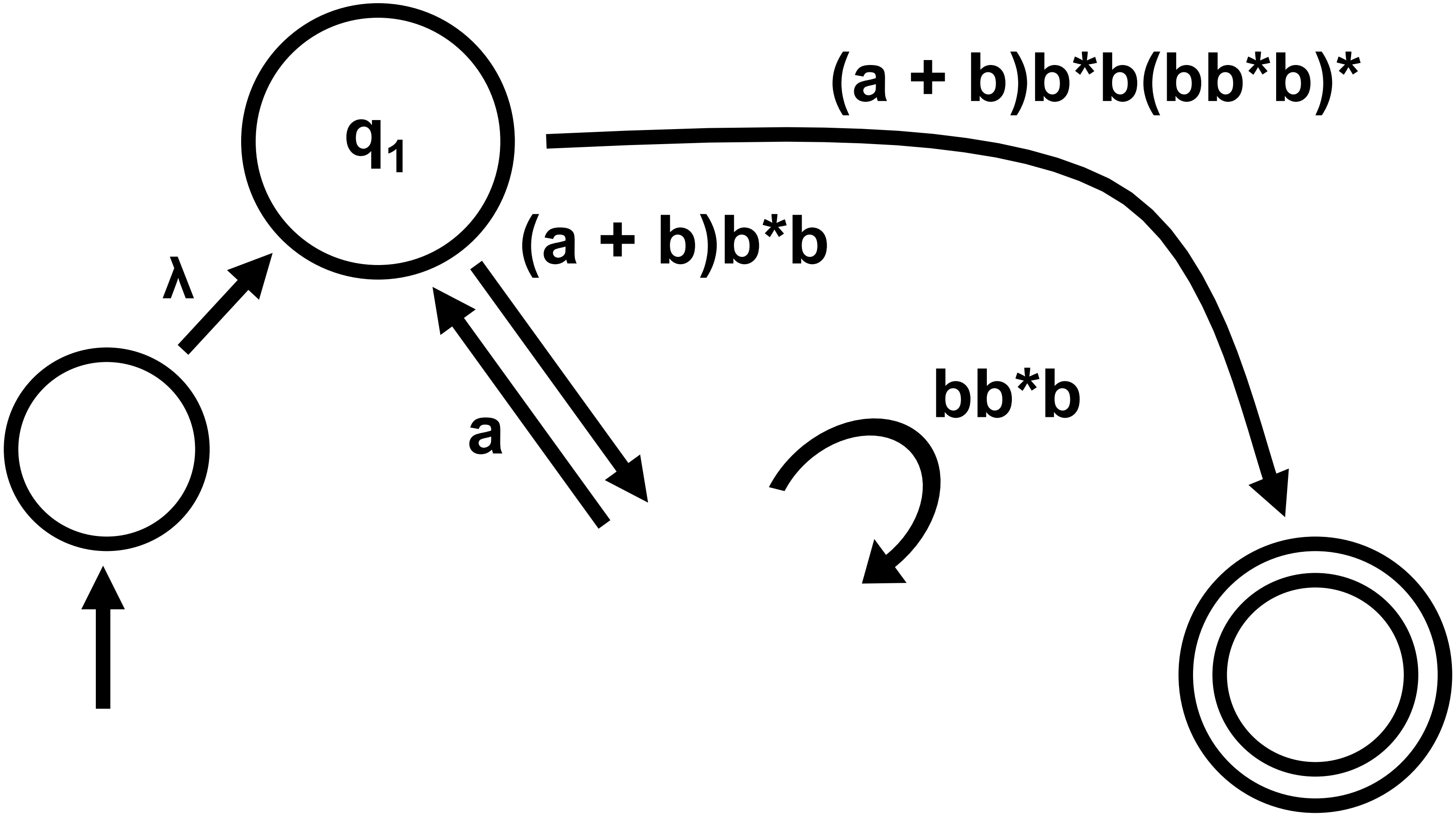
Convert the following NFA into a regular expression?



Example 4

DFA→RE

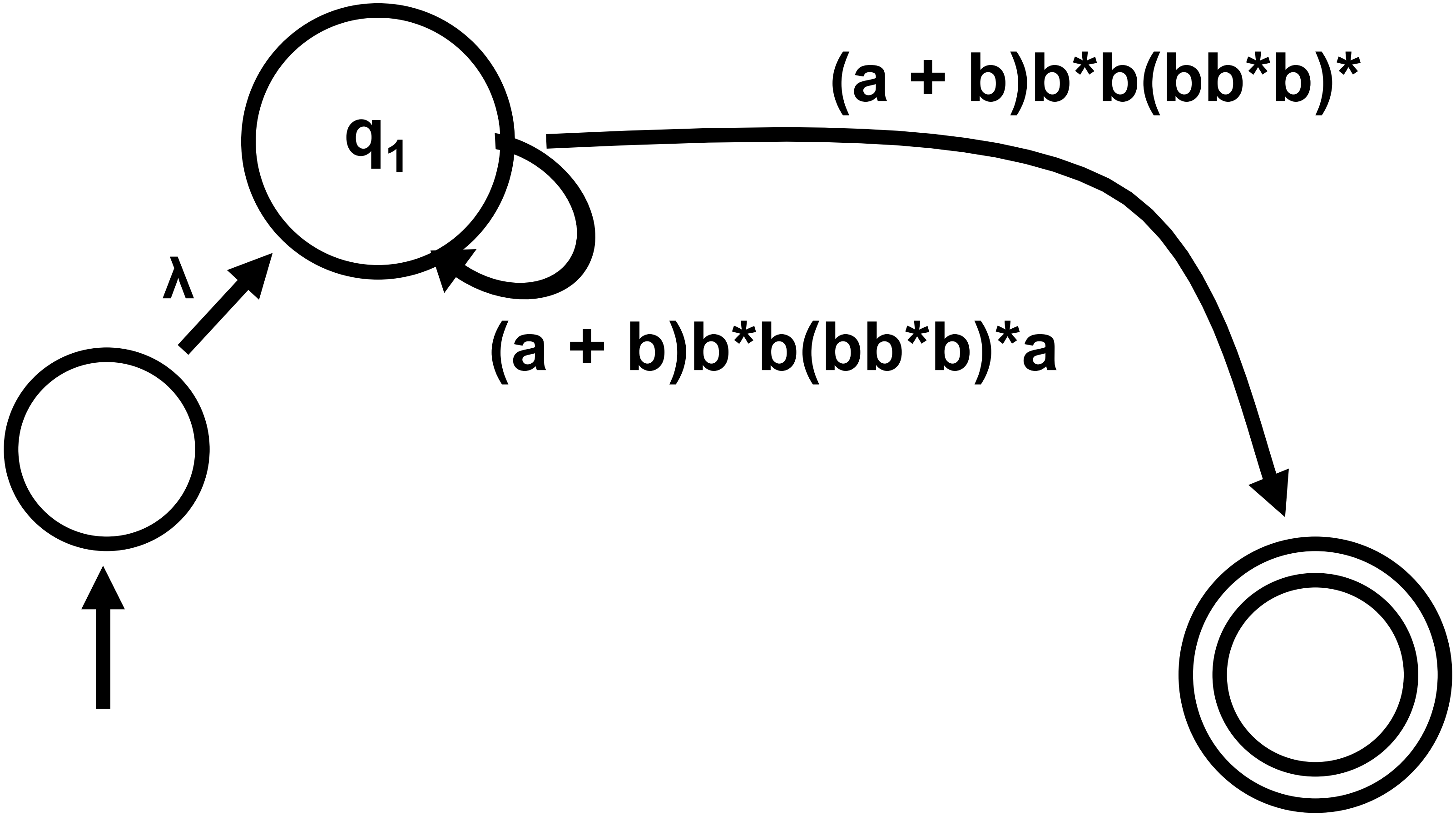
Convert the following NFA into a regular expression?



DFA→RE

Example 4

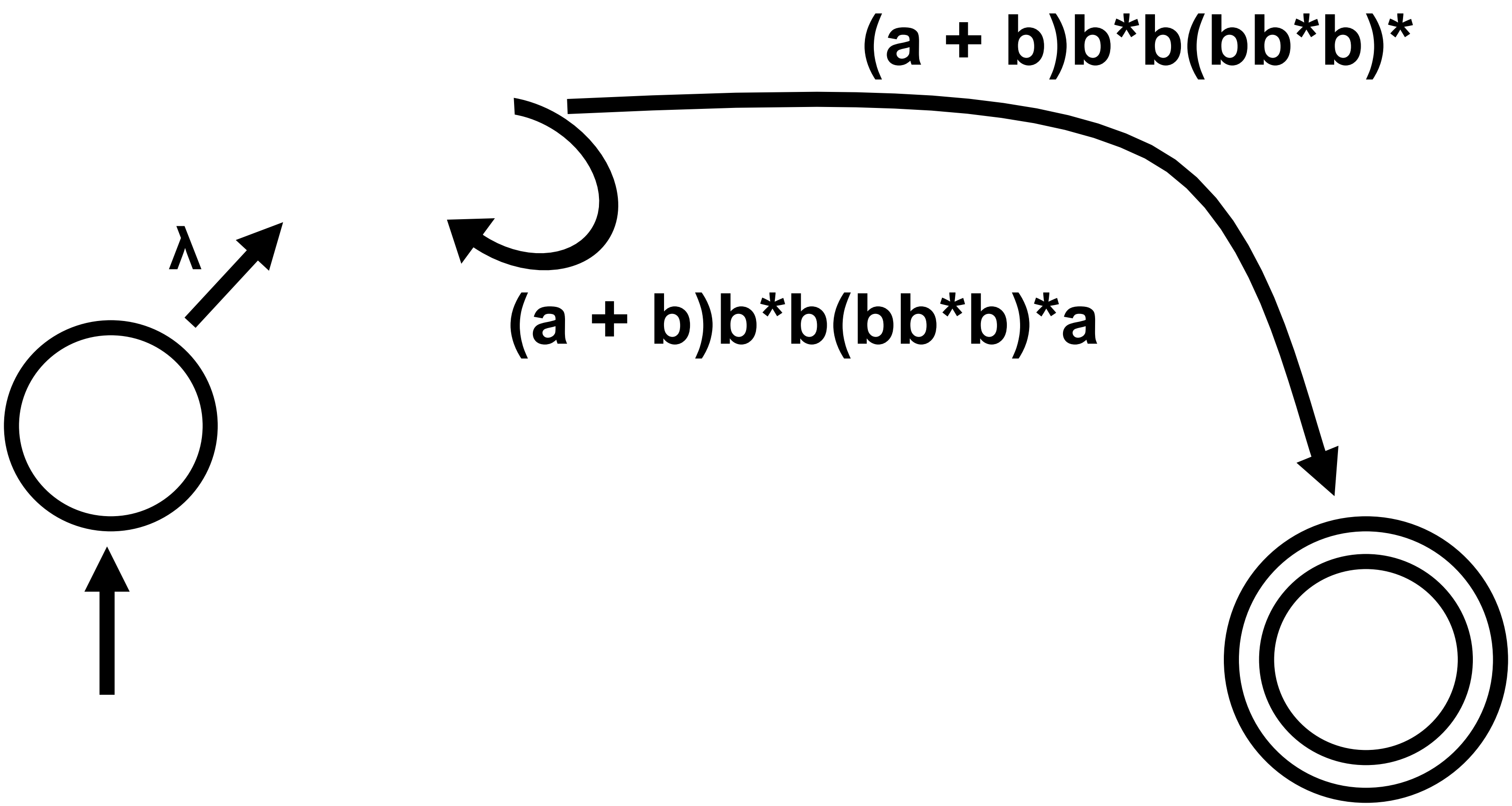
Convert the following NFA into a regular expression?



DFA→RE

Example 4

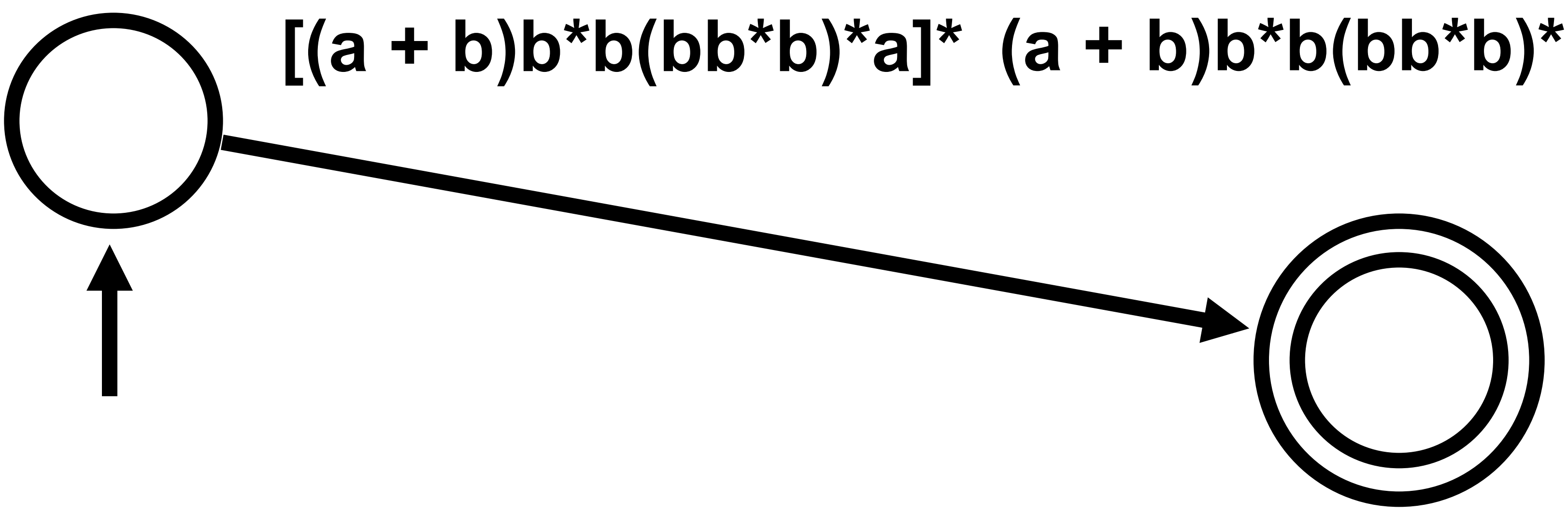
Convert the following NFA into a regular expression?



Example 4

DFA→RE

Convert the following NFA into a regular expression?



DFA→RE

Algorithm

Next we study the DFA→RE Algorithm

Algorithm

Add q_{start} and q_{final} to create G

Run CONVERT(G):

If #states = 2

return the expression on the arrow going from q_{start} to q_{final}

If #states > 2

select $q_{rip} \in Q$ different from q_{start} and q_{final}

define $Q' = Q - \{q_{rip}\}$

define R′ as:

$$R'(q_i,q_j) = R(q_i,q_{rip})R(q_{rip},q_{rip})^*R(q_{rip},q_j) \cup R(q_i,q_j)$$

return CONVERT(G′)

CONVERT(G) is equivalent to G

Proof by induction on k (number of states in G)

Base Case:

- ✓ **k = 2**

Inductive Step:

Assume claim is true for k-1 states

We first note that G and G′ are equivalent

But, by the induction hypothesis, G′ is equivalent to CONVERT(G′)

Exercise

DFA→RE