Intro to Theory of Computation

CS 332

LECTURE 4

Last time:

- Equivalence of NFAs and DFAs
- Closure properties of regular languages

Today:

- Finish closure properties
  Equivalence of NFAs, DFAs and regular expressions

Sofya Raskhodnikova

Operations on languages

Complement: $\neg A = \{ w \mid w \notin A \}$

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

Reverse: $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$

Concatenation: $A \circ B = \{ vw \mid v \in A \text{ and } w \in B \}$

Star: $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

Closure properties of the class of regular languages

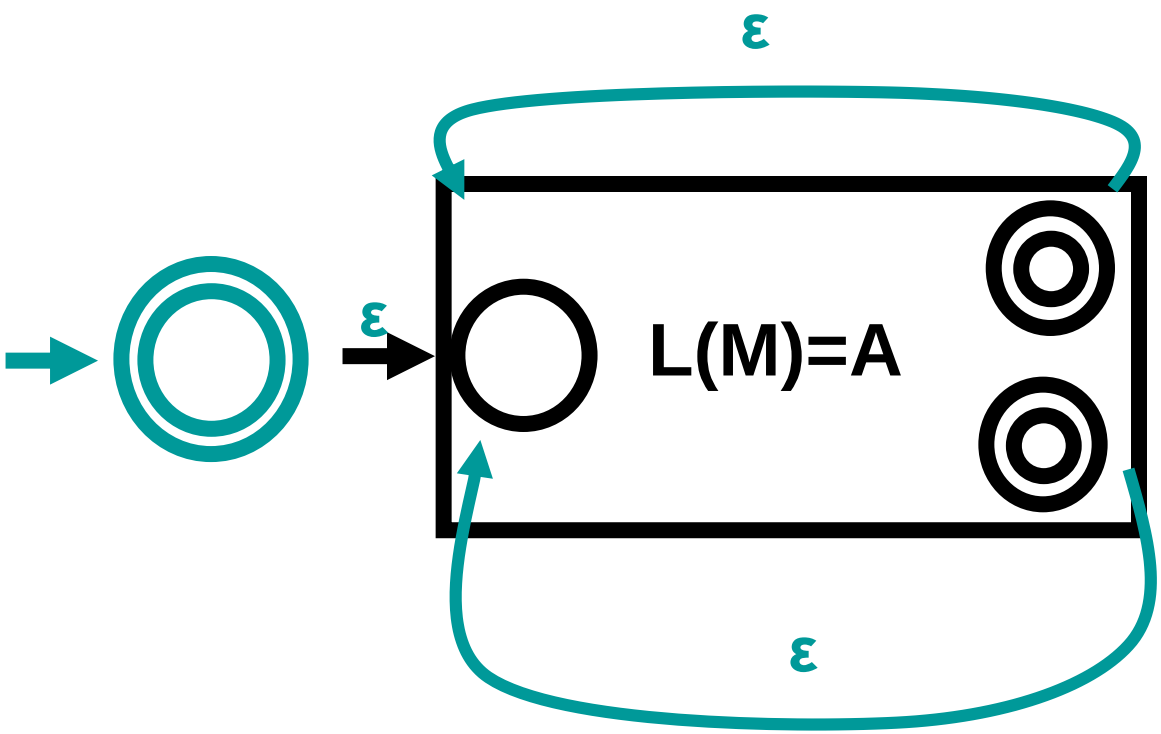
THEOREM. The class of regular languages is **closed** under all 6 operations.

If A and B are regular, applying any of these operation yields a regular language.

Star operation

Star: $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

Theorem. **If A is regular, A* is also regular.**



The class of regular languages is closed under

Regular operations

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

Concatenation: $A \circ B = \{ vw \mid v \in A \text{ and } w \in B \}$

Star: $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

Other operations

Complement: $\neg A = \{ w \mid w \notin A \}$

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

Reverse: $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$

Regular expressions

- **In a regular expression, we can use**
  - **Constants:** ε , $\emptyset$, a set Σ, members of Σ.
  - **Regular operations:** $*$, $\circ$, $\cup$
- **Examples**:
  - 0^*1^* = { w | w has a run of 0s followed by a run of 1s}
  - $(0 \cup 1)^*$ = the set of all strings over the alphabet $\Sigma=\{0,1\}$
  - $0^*1^*(\varepsilon \cup 0)$
- **L(R)** = the language regular expression R describes

Precedence

$*$ $\circ$ $\cup$

EXAMPLE

$$R_1 R_2^* \cup R_3 R_4 = (R_1(R_2^*)) \cup (R_3 R_4)$$

Regular expressions: examples

1) { w | w has exactly one character 1, any # of 0s}

$$0^*10^*$$

2) { w | w has length ≥ 3 and its 3rd symbol is 0 }

$$(0 \cup 1)(0 \cup 1)\ 0\ (0 \cup 1)^*$$

3) { w | every odd position of w is a 1 }

$$(1(0 \cup 1))^*\ (\varepsilon \cup 1)$$

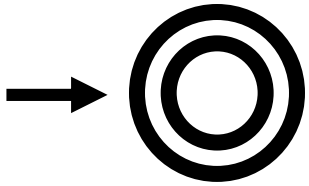
Regular expressions to NFAs

Theorem. Every regular expression has an equivalent NFA.

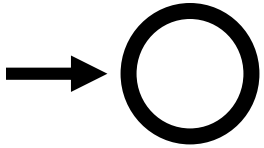
Proof: Induction on the length of regular expression R.

Base case: length 1

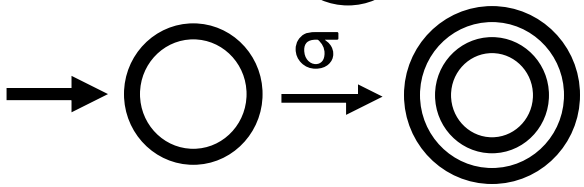
$R = \varepsilon$



$R = \varnothing$



$R = \sigma$



Inductive step: follows from closure of the class of regular languages under the regular operations.

Exercise

What should the induction hypothesis be?

A. Suppose some regular expression of length k can be converted an NFA, for some $k \in \mathbb{N}$.

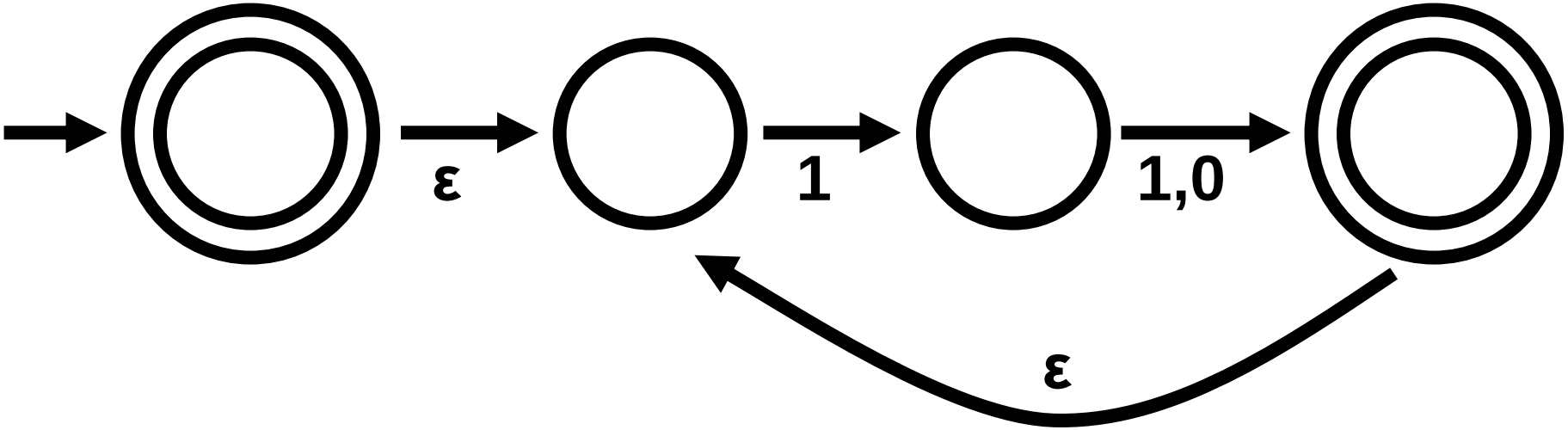
B. Suppose each regular expression of length k can be converted an NFA, for some $k \in \mathbb{N}$.

C. Suppose each regular expression of length at most k can be converted an NFA, for some $k \in \mathbb{N}$.

D. None of the above.

Regular expression to NFA

Transform (1(0 ∪ 1))* to an NFA

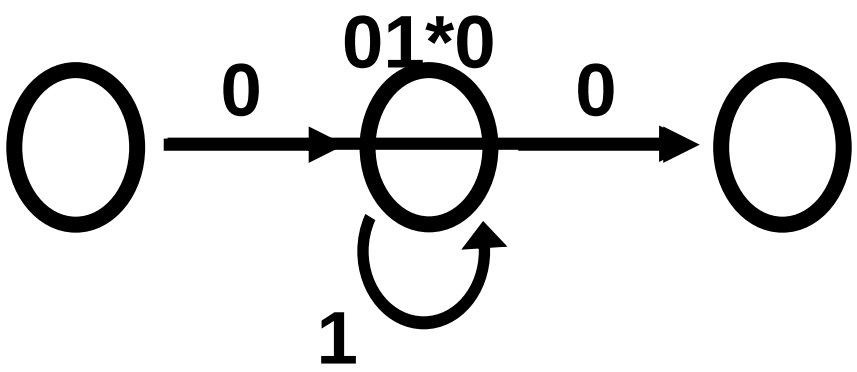


NFAs to regular expressions

Theorem. **Every NFA has an equivalent regular expression.**

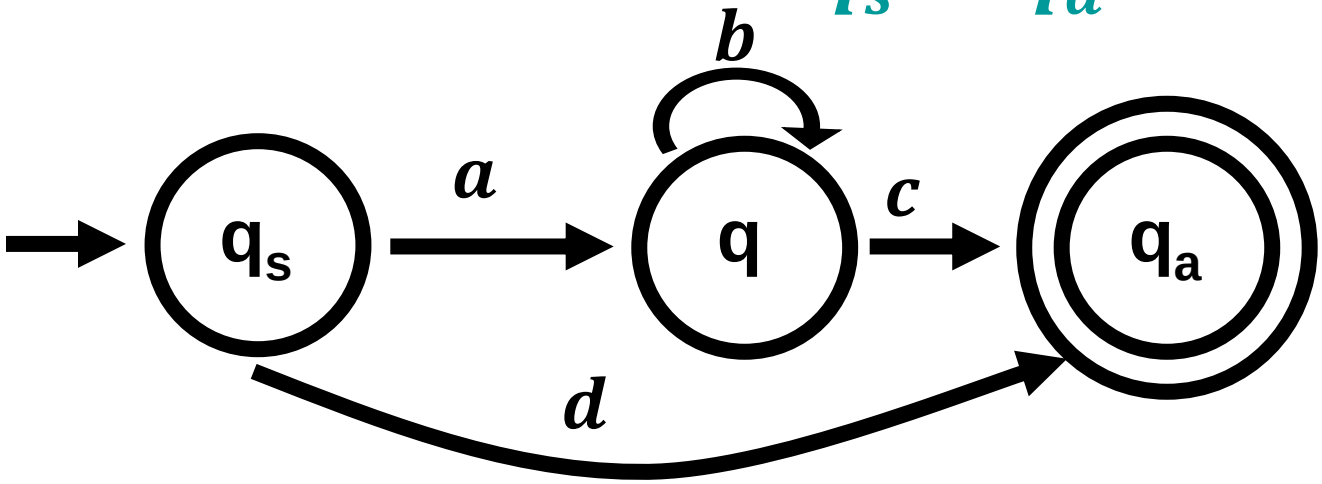
Proof idea:

Transform NFA to a regular expression by removing states and relabeling the arrows with regular expressions.



Exercise

What is the regular expression that generates all strings that take this machine from q_s to q_a?



A. $ab^*c \cup d$

B. $ab^*c \cap d$

C. $abc \cup d$

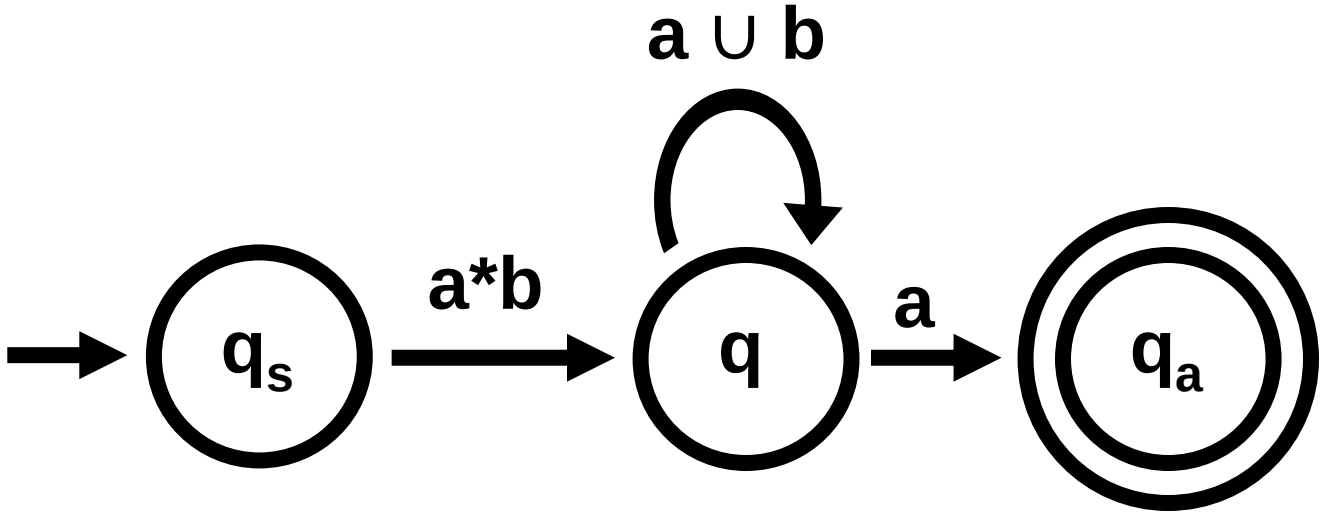
D. $(ab^*cd)^*$

E. None of the above.

Generalized NFAs

- **Each transition is labeled with a regular expression**
- Unique and distinct start and accept states
- No transitions **to** the start state
- No transitions **from** the accept state

Generalized NFAs



A GNFA accepts w if $\exists\, q_0, q_1, \ldots, q_k,\ w_1, \ldots, w_k$:

$R(q_s, q) = a^*b$

w_i is generated by $R(q_{i-1}, q_i)$

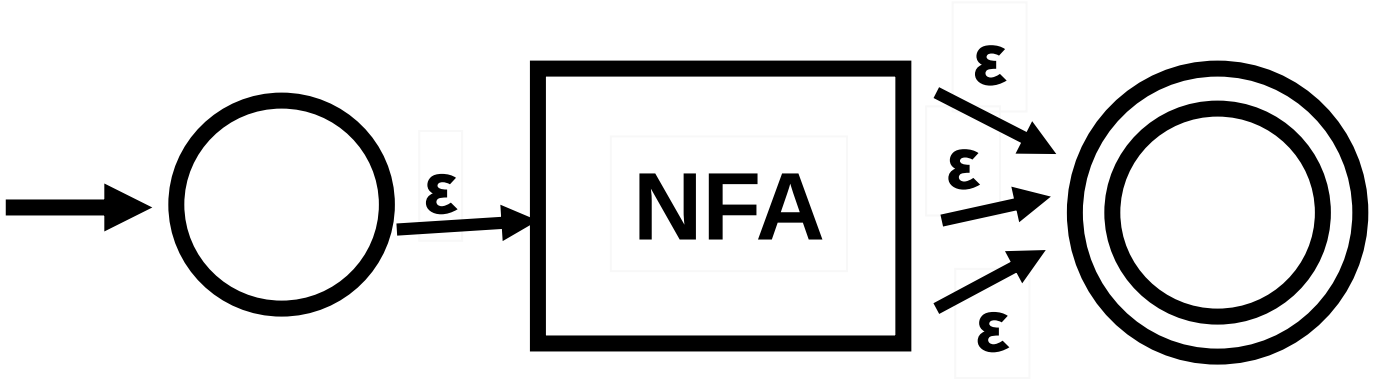
$R(q_a, q) = \emptyset$

$w = w_1 w_2 \ldots w_k$

$R(q, q_s) = \emptyset$

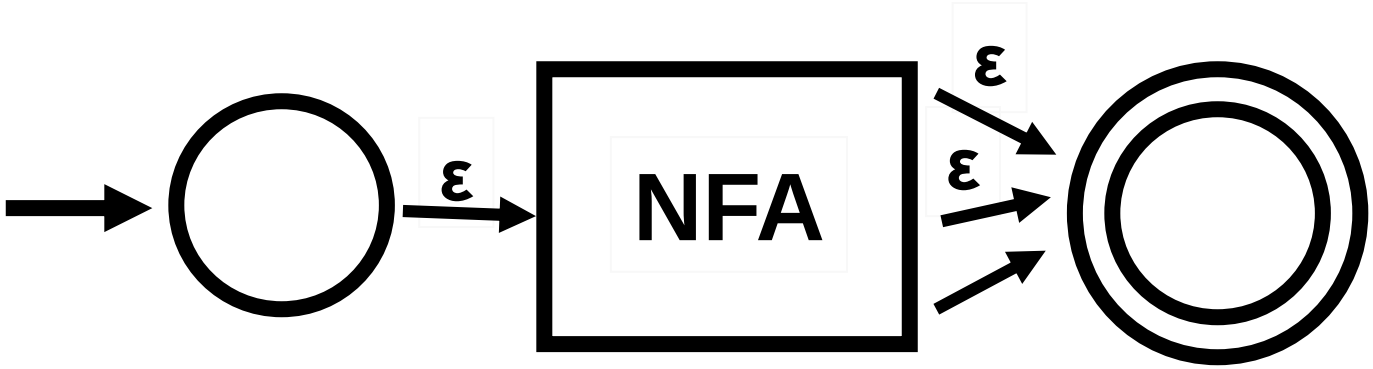
$q_0 = q_s,\ q_k = q_a$

NFA to GNFA



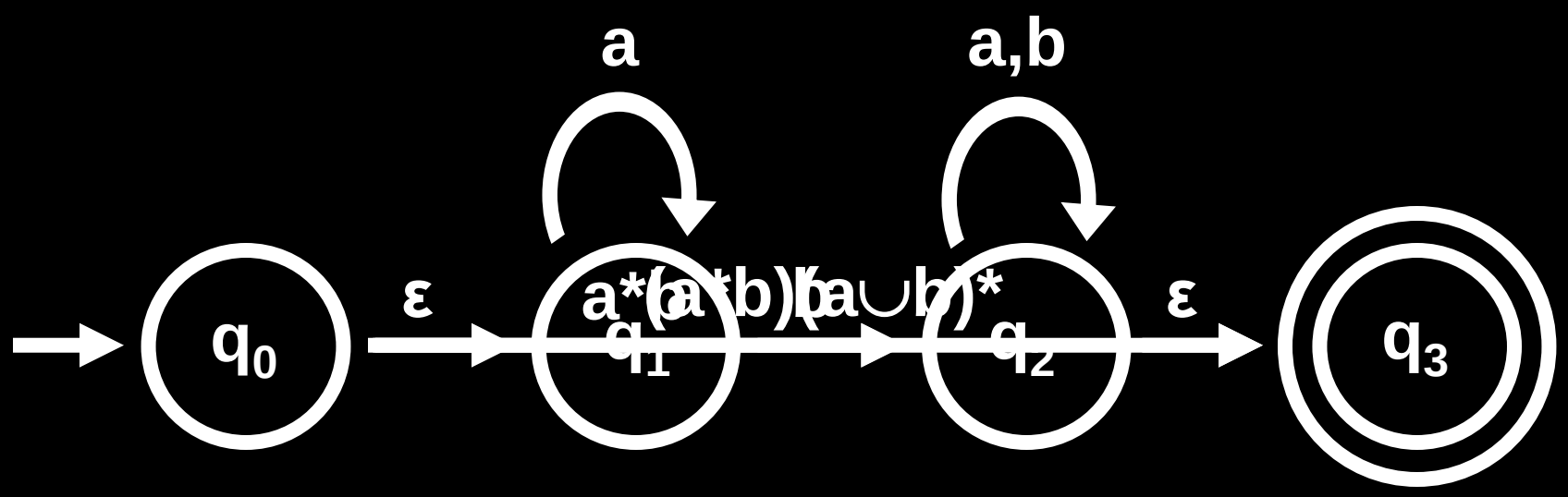
Add a new start state with no incoming arrows.
Make a unique accept state with no outgoing arrows.

GNF to regular expression

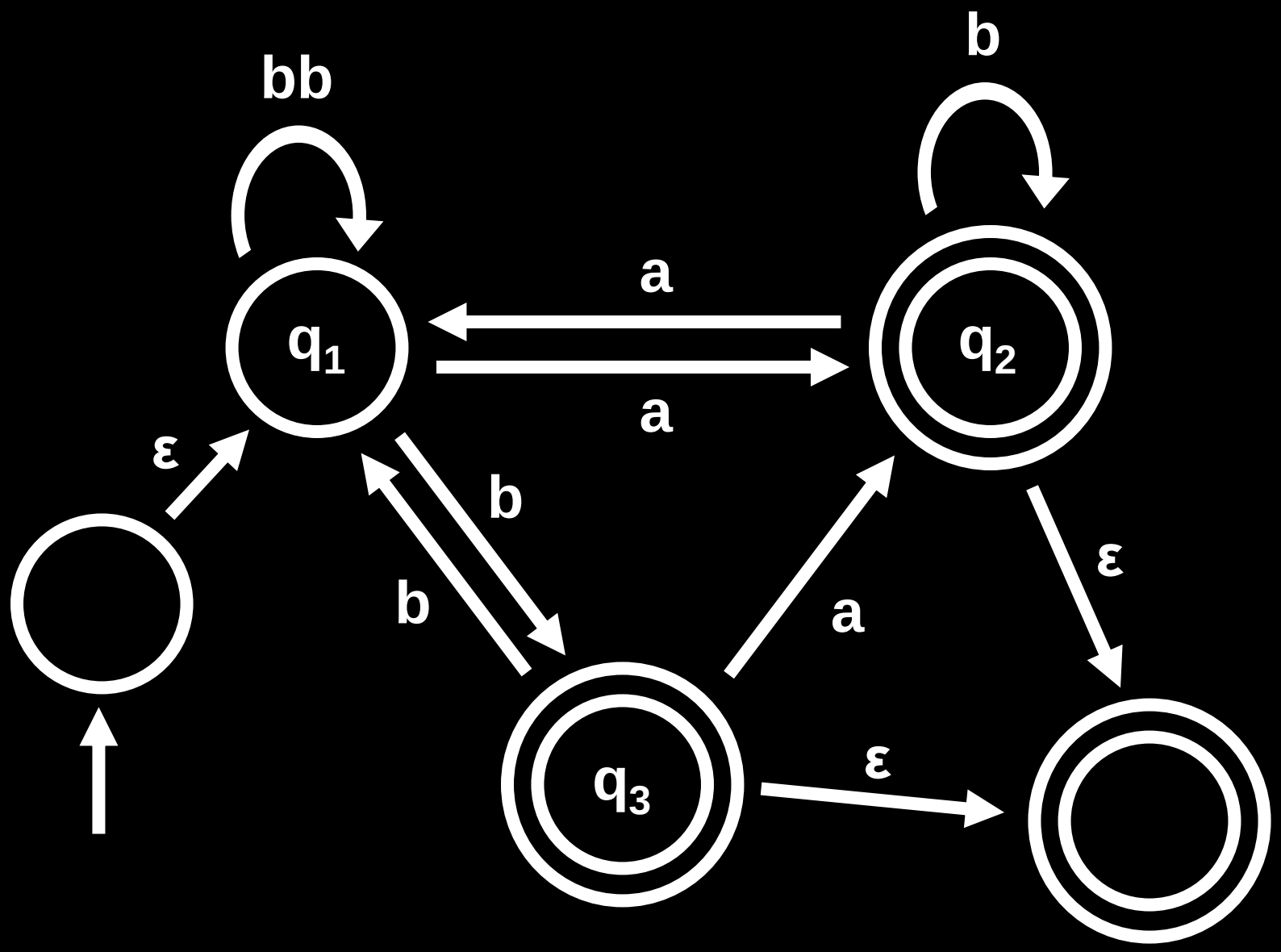


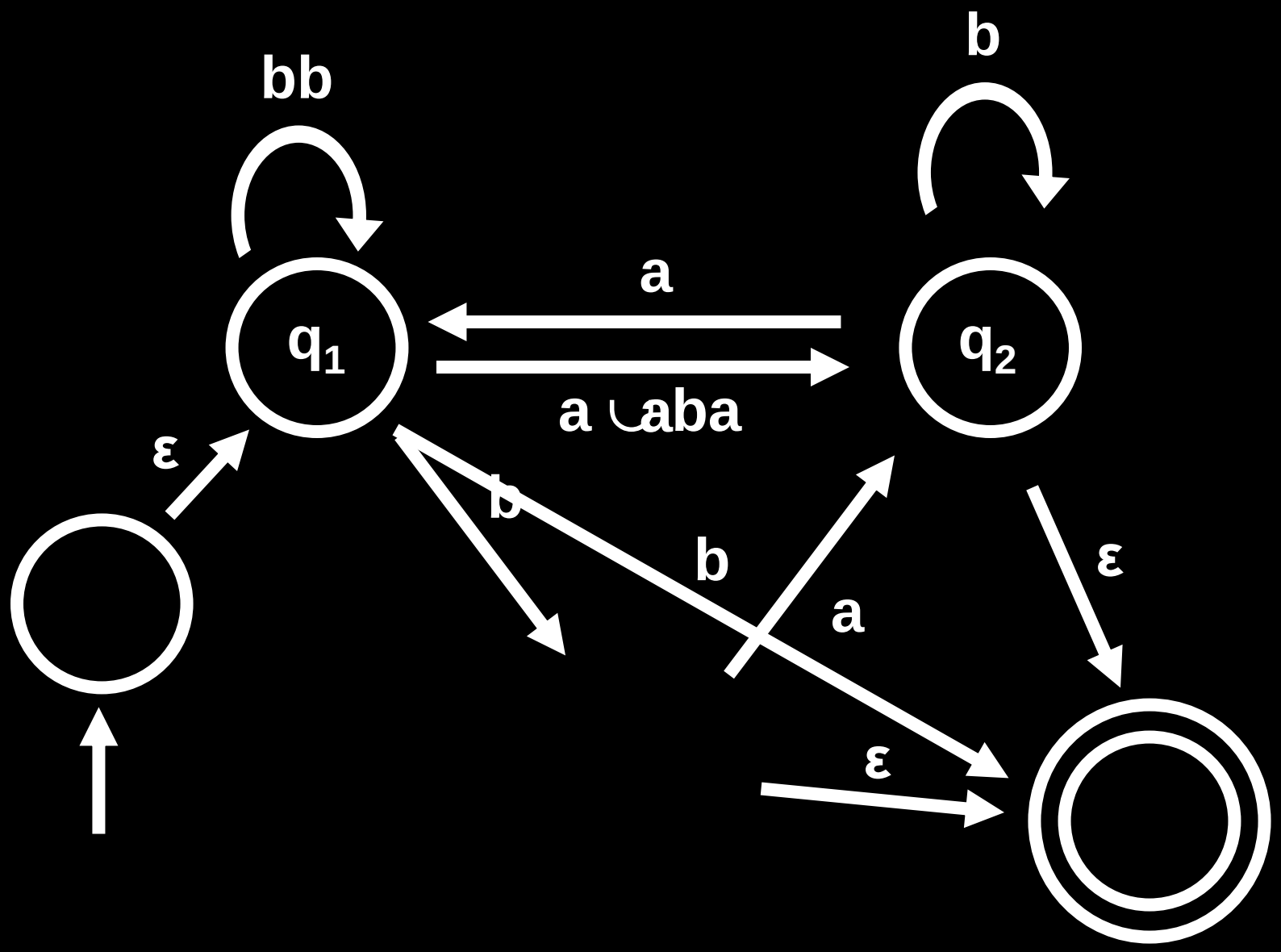
While machine has more than 2 states:

Pick an internal state, rip it out and relabel the arrows with regular expressions to account for the missing state.

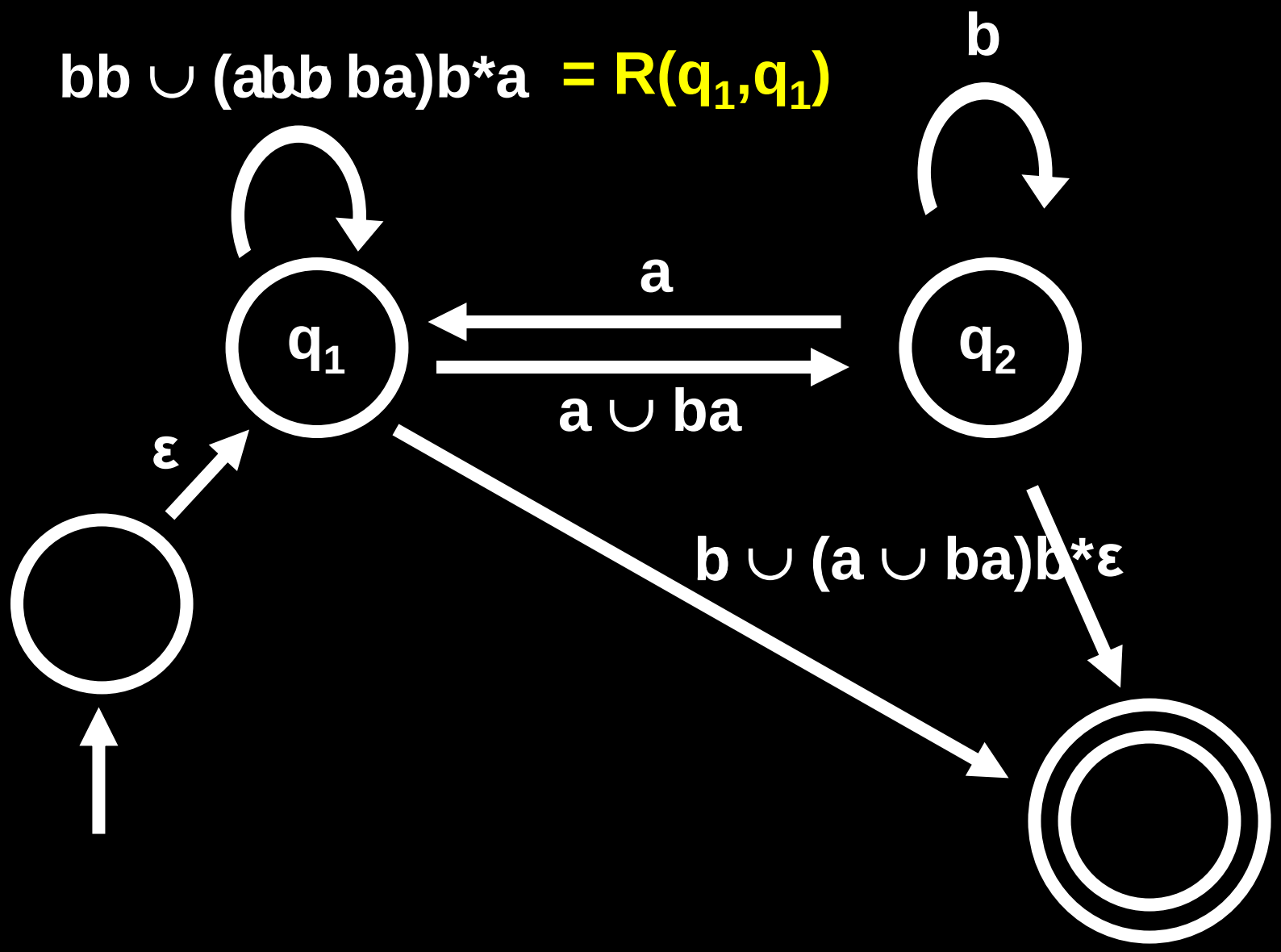


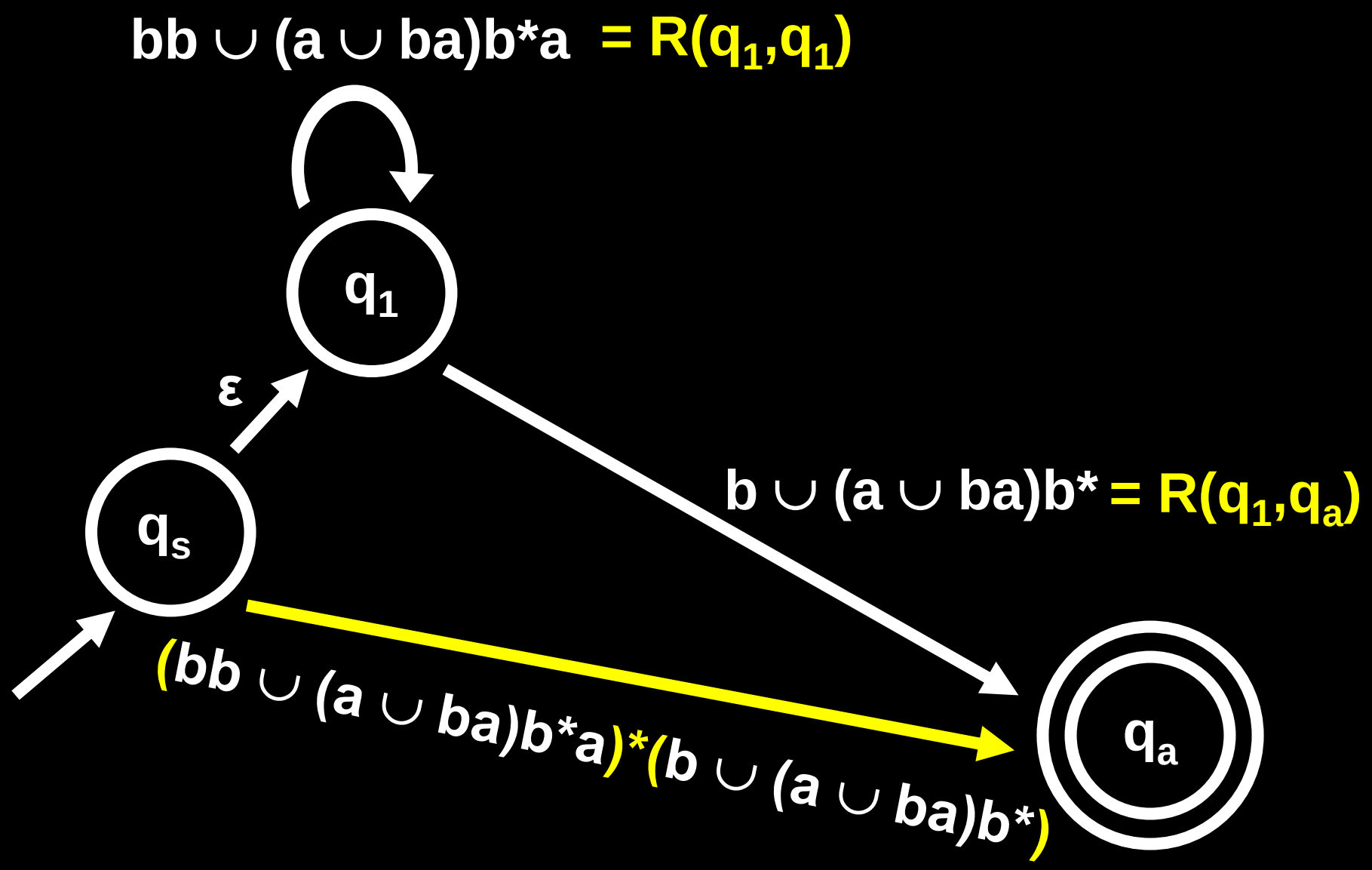
$R(q_0,q_3) = (a^*b)(a \cup b)^*$


bb
b
a
q1
q2
a
ε
b
b
ε
a
q3
ε


bb
b
a
q1
q2
a ∪ aba
ε
b
b
a
ε
ε

$bb \cup (abb\ ba)b^*a = R(q_1,q_1)$




bb ∪ (a ∪ ba)b*a = R(q1,q1)
q1
ε
qs
b ∪ (a ∪ ba)b* = R(q1,qa)
qa
(bb ∪ (a ∪ ba)b*a)*(b ∪ (a ∪ ba)b*)

Conversion procedures