Finite Automata 05

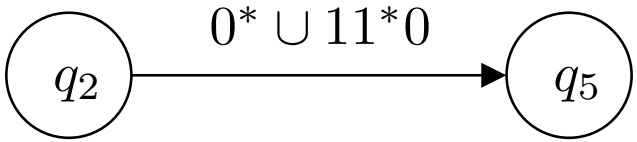
Thumrongsak Kosiyatrakul
tkosiyat@cs.pitt.edu

Regular Language

- **Question**: Given a regular language, can it be expressed by a regular expression?
  - There are infinite number of regular languages
  - Luckily, all of them have one thing in common
  - Each of them has some finite-state machines that recognize it
- So, to try to answer the above question, we need to show a way to convert a finite-state machine into a regular expression that expresses the language of the machine

Generalized Nondeterministic Finite Automaton

- A Generalized Nondeterministic Finite Automaton (GNFA) N of a DFA M is
  - a special NFA where $L(N) = L(M)$
  - N has exactly one accept state
  - all transitions of N are regular expressions
    - To transition from one state to another, you need a string in the language expressed by the regular expression instead of a symbol
  - Example:

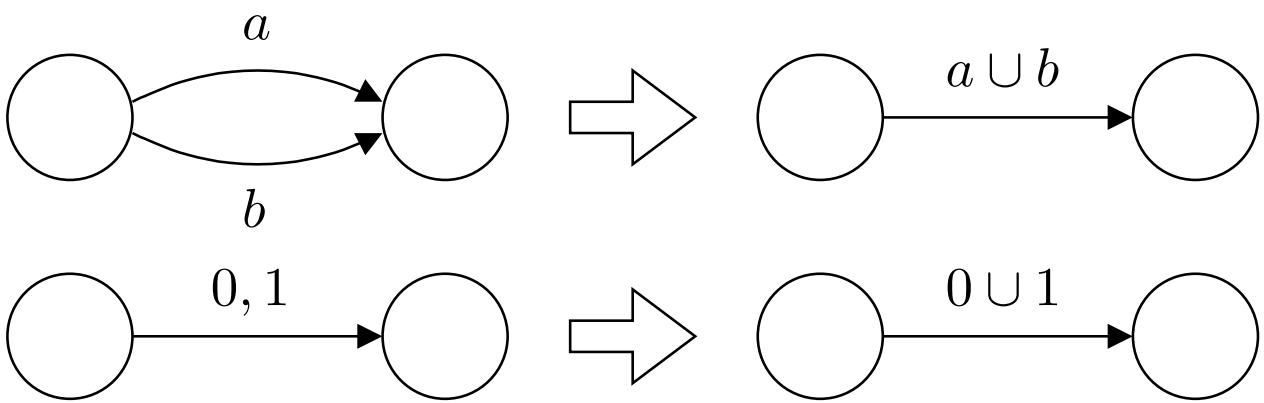


At q_2, if you encounter a string in the language $0^* \cup 11^*0$, move to state q_5

$$0^* \cup 11^*0 = \{\varepsilon, 0, 00, \dots\} \cup \{10, 110, 1110, \dots\}$$

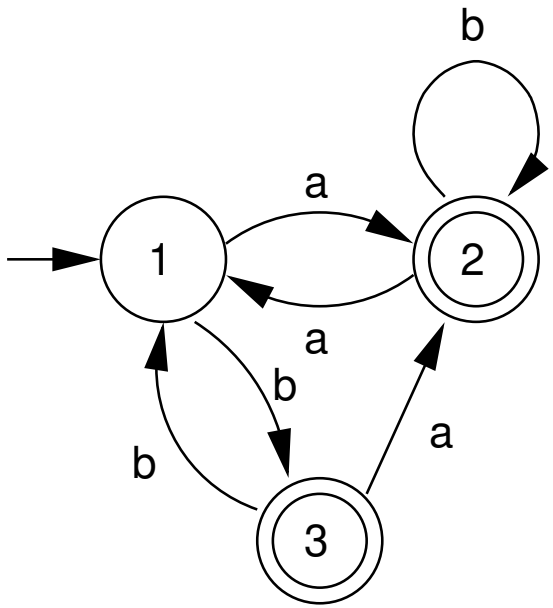
Converting a DFA to a GNFA

1. Start with a DFA M
2. Add a new start state s with an ε arrow to the original start state of M
3. Add a new accept state a
   - From every accept state of M, add an ε arrow to the new accept state
   - Change all original accept states of M to non-accept states
4. Turn transition labels to regular expressions (rule #1 of regular expression)
5. Add necessary transition arrows
   - Change multiple arrows or multiple labels to a single arrow while label is the union of the previous labels



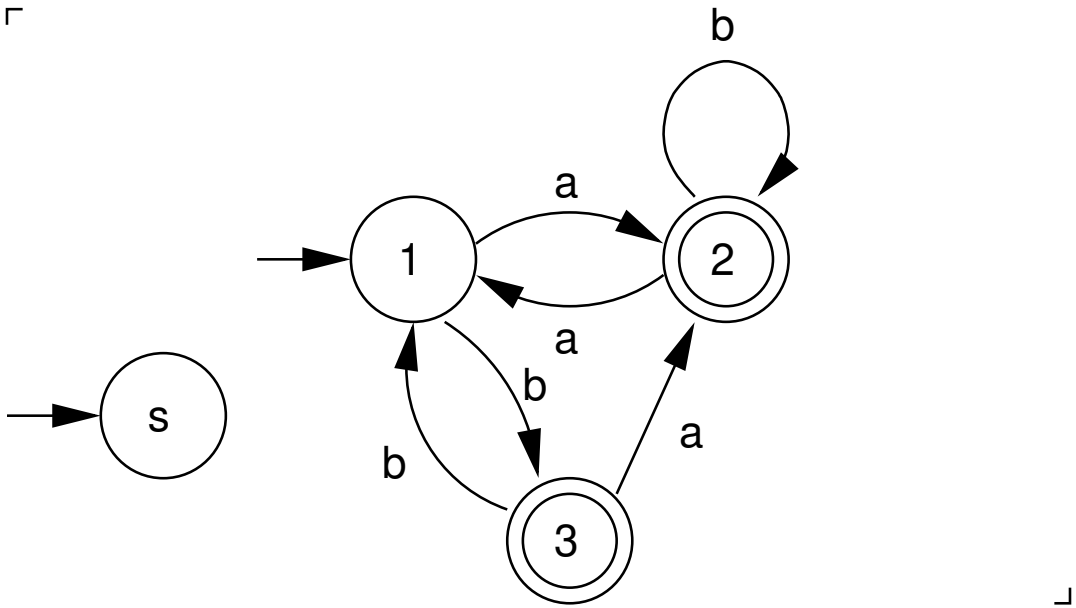
Converting a DFA to a GNFA (Example)

- A DFA



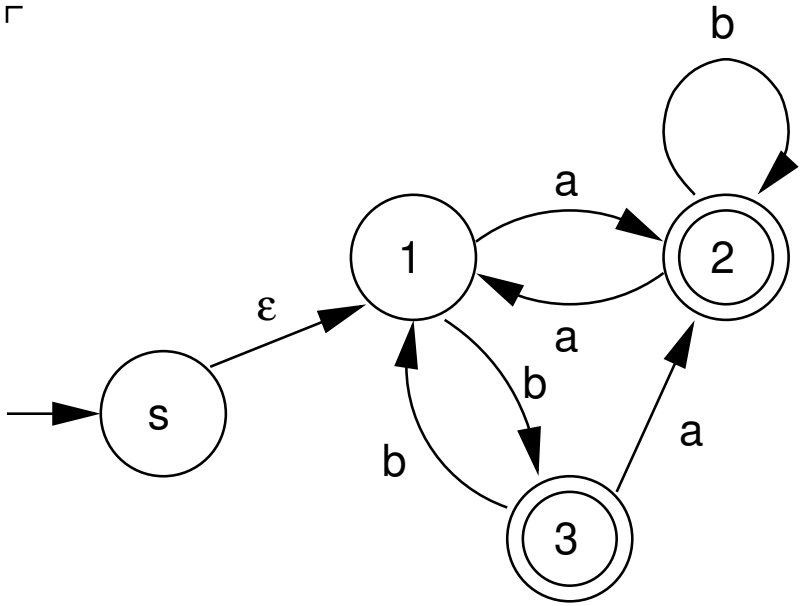
Converting a DFA to a GNFA (Example)

- Add a new start state s



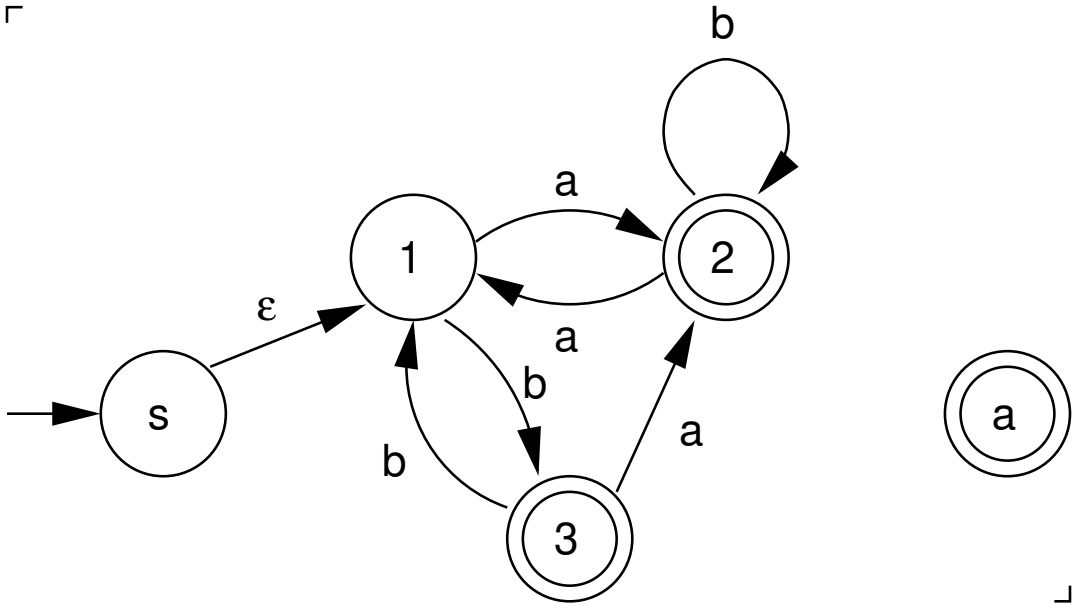
Converting a DFA to a GNFA (Example)

- ε from s to the original start state



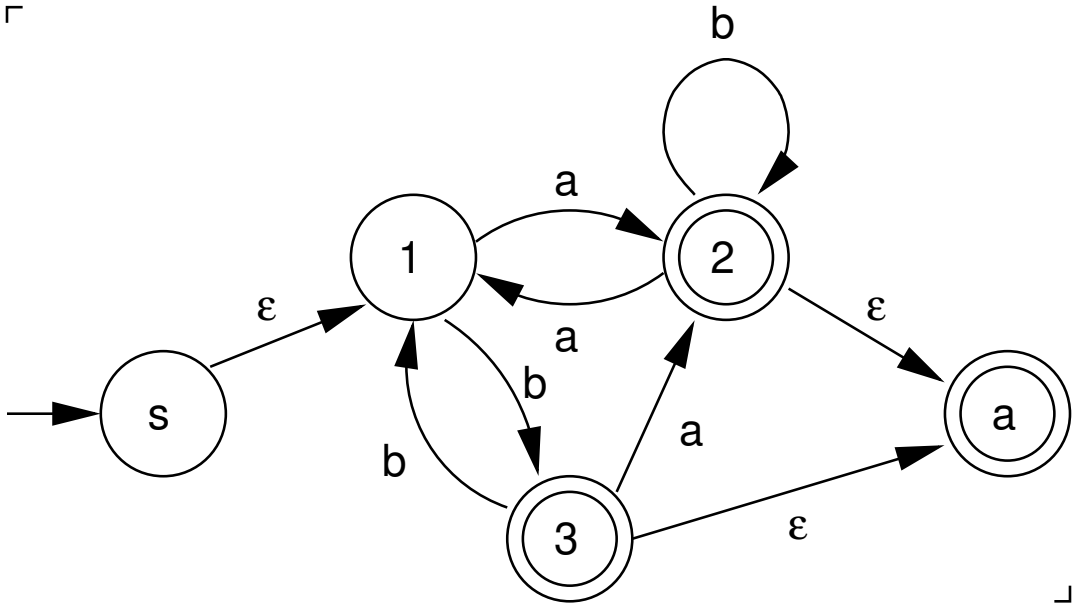
Converting a DFA to a GNFA (Example)

- Add a new accept state a



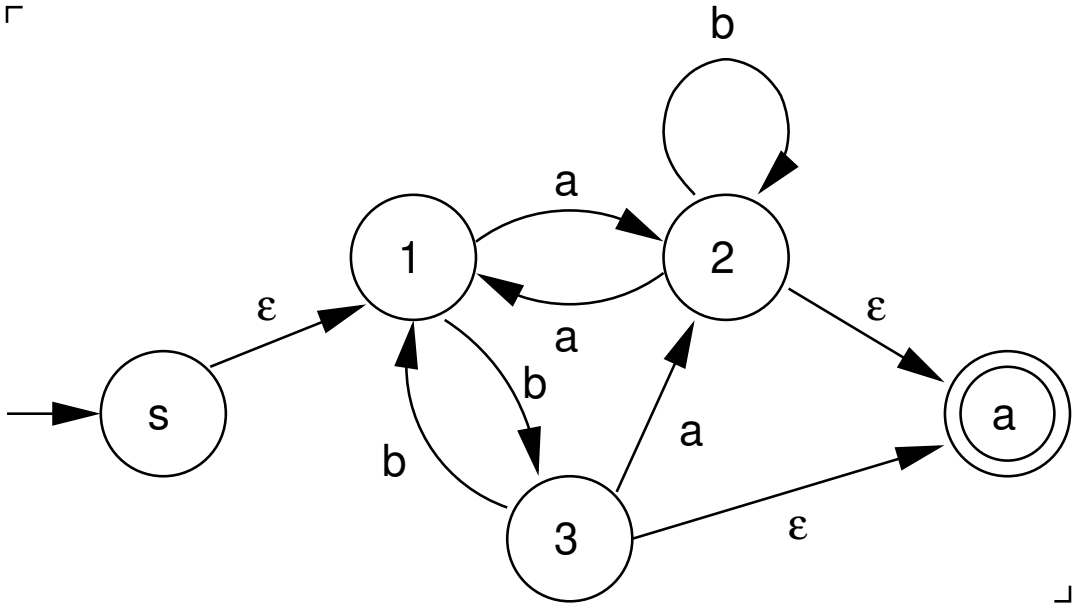
Converting a DFA to a GNFA (Example)

- ε from all accepts state to a



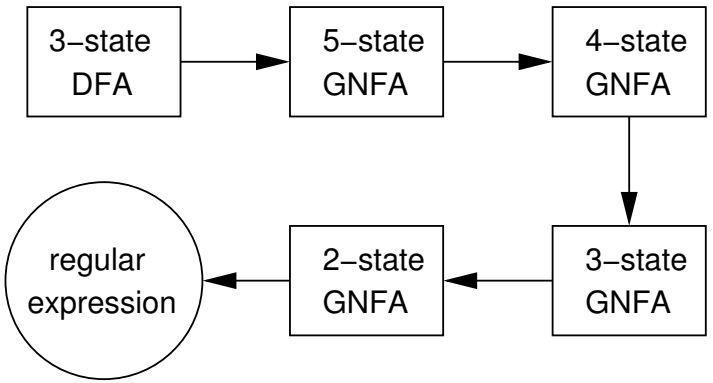
Converting a DFA to a GNFA (Example)

- All original accept states to non-accept state



GNFA to Regular Expression

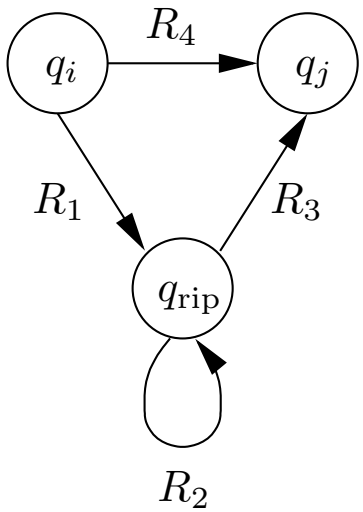
- Note that by converting a DFA and an equivalent GNFA, the language of the machine remain unchanged
- Next, we are going to remove original states of DFA one at a time until there are only two states left
  - When a state is removed, we have to make sure that the language of the machine is not changed
  - The new start state s and the new accept state a
- Example:



- The transition from s to a is a regular expression that expresses the language of the original DFA

Reducing Number of States of GNFA

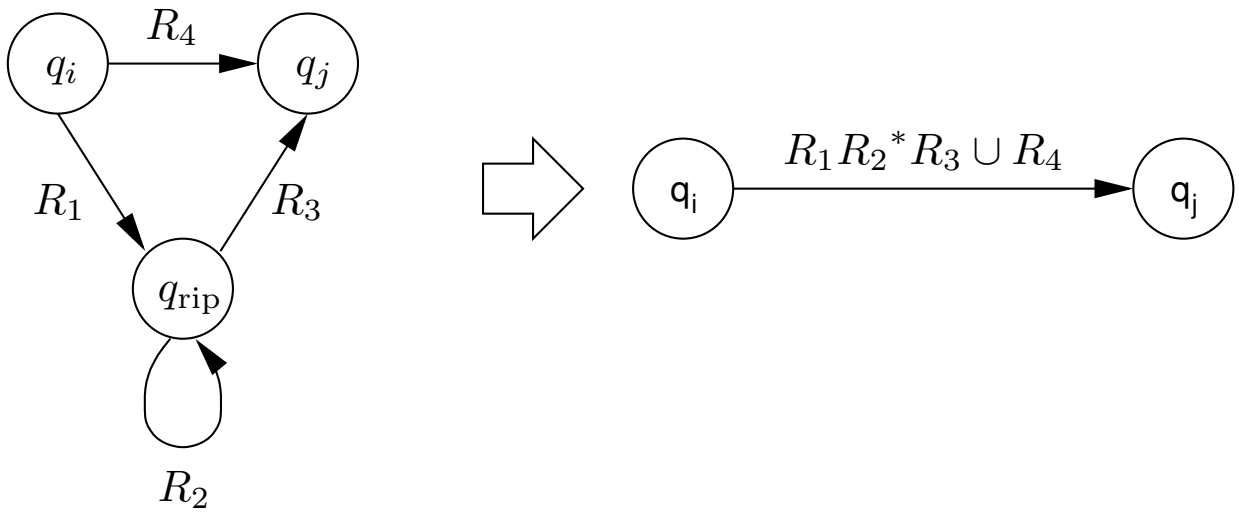
- Removing q_{rip}



- How to go from q_i to q_j?
  - $q_i \to q_j$ using a string in R_4
  - $q_i \to q_{\text{rip}} \to q_j$
    - $q_i \to q_{\text{rip}}$ using a string in R_1
    - $q_{\text{rip}} \to q_{\text{rip}}$ using a string in R_2 any number of times (R_2^*)
    - $q_{\text{rip}} \to q_j$ using a string in R_3
- From q_i to q_j by any strings in $R_1 R_2^* R_3 \cup R_4$

Reducing Number of States of GNFA

- Removing q_{rip}



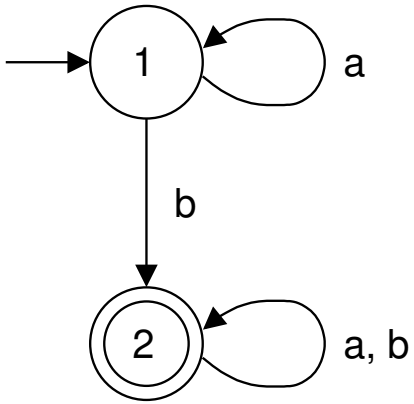
- To remove q_{rip}
  1. Search for all possible paths from a state q to q_{rip} and to a state r

$$q \rightarrow q_{\text{rip}} \rightarrow r$$

  2. Turn all paths from previous step to regular expressions
  3. Remove q_{rip} which results in $q \rightarrow r$
  4. Insert paths $q \rightarrow r$ back with their regular expressions

Example

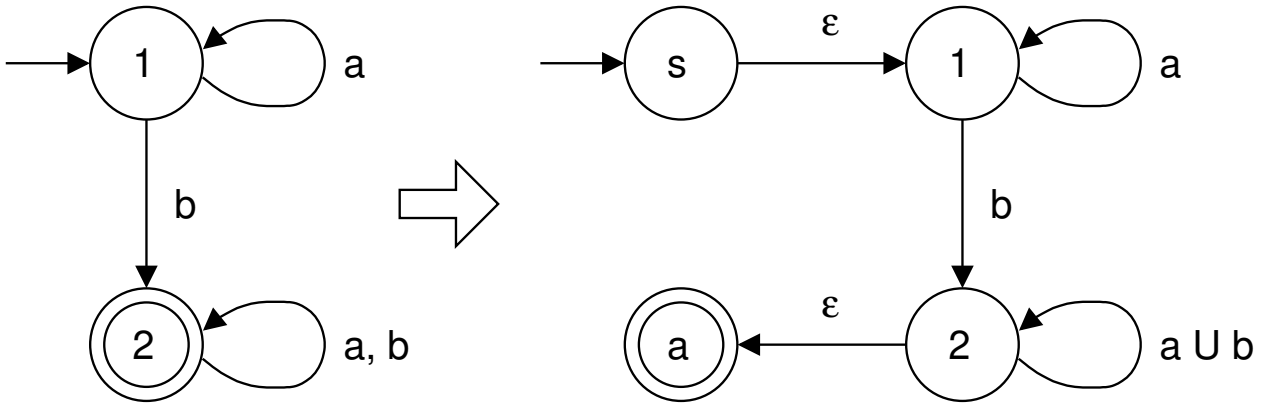
- DFA



$$\{w \mid w \text{ contains at least one } b\}$$

Example

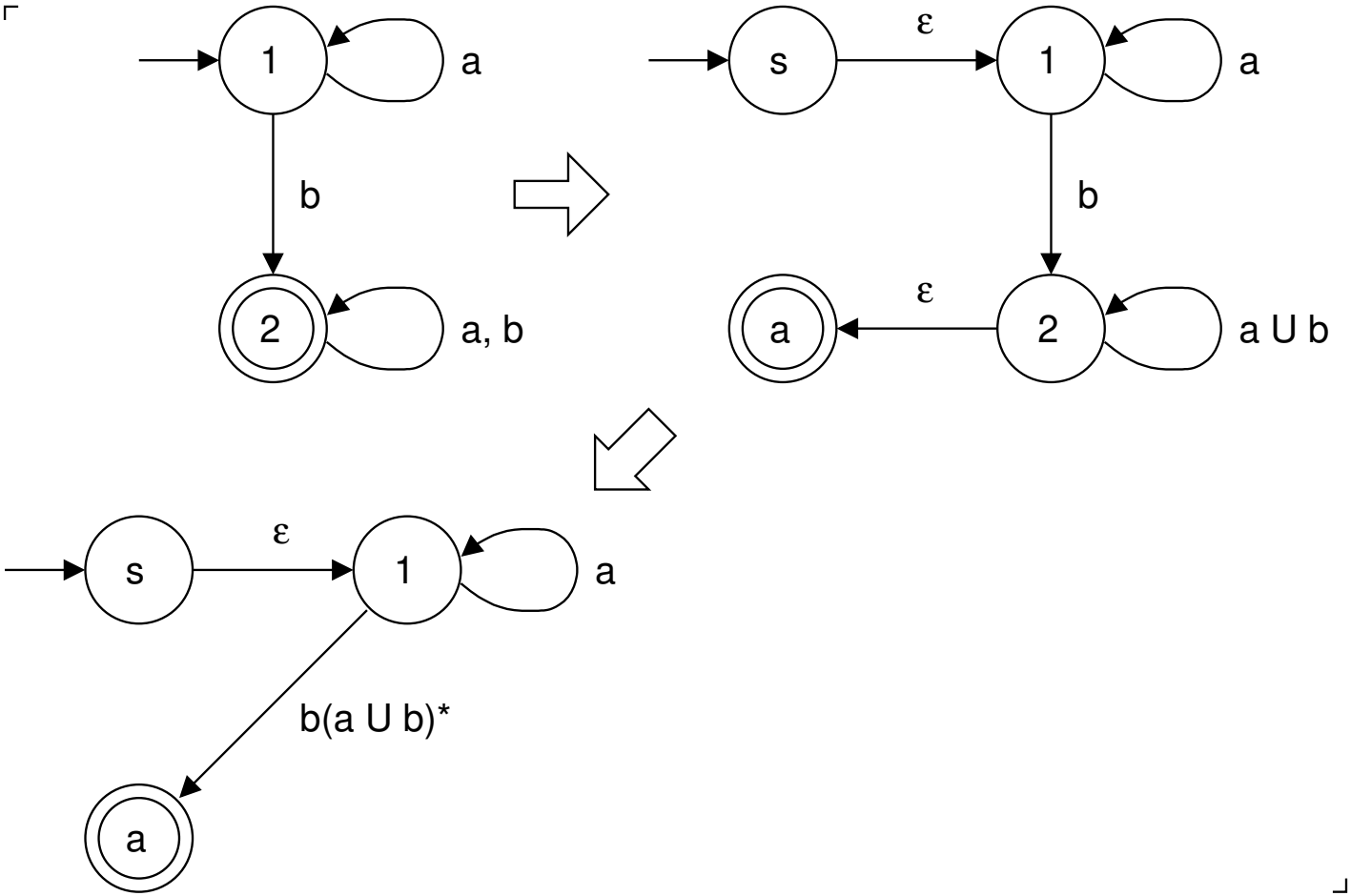
- DFA to GNFA



$\{w \mid w$ contains at least one $b\}$

Example

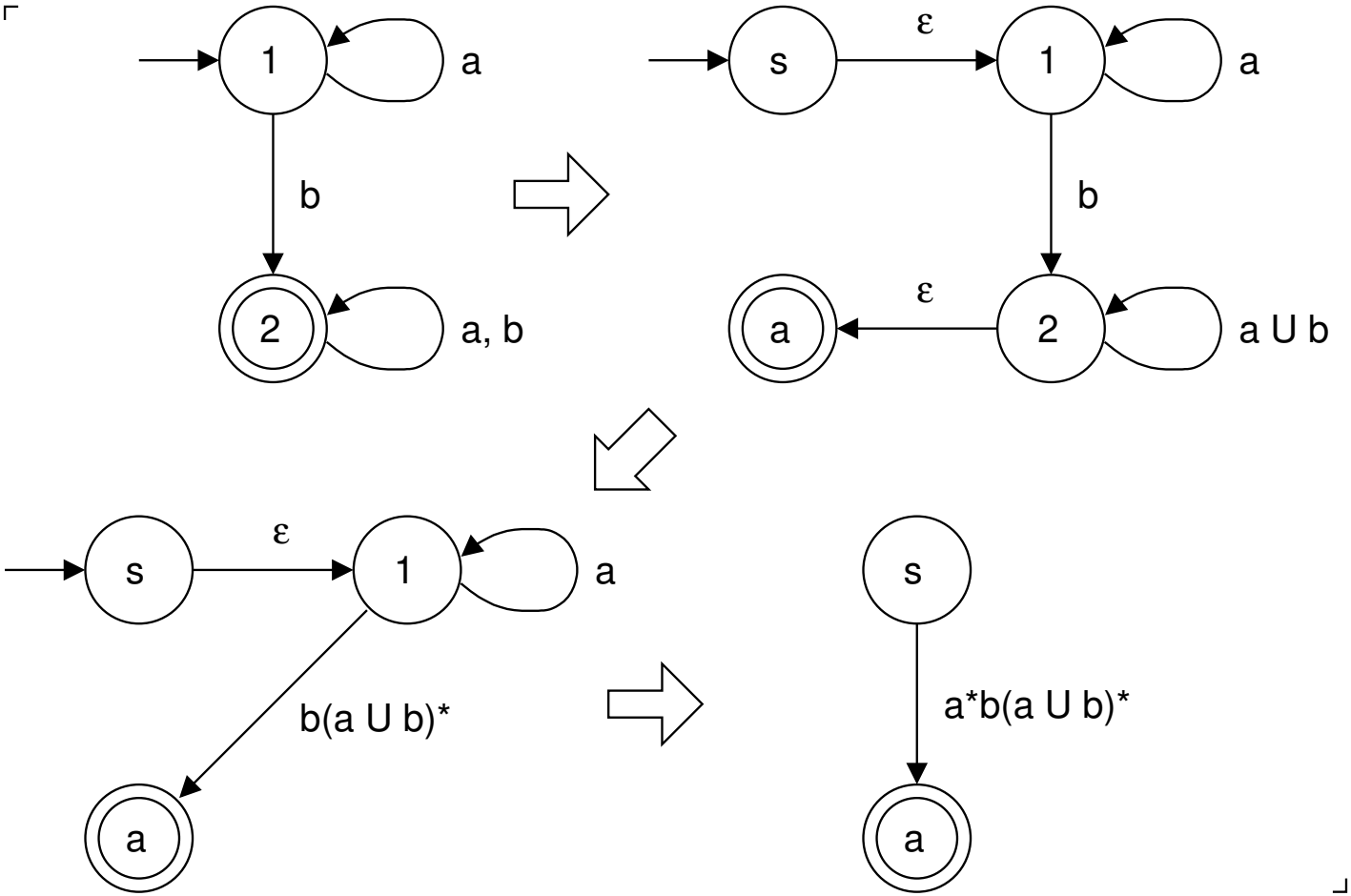
- Get rid of state 2



$\{w \mid w \text{ contains at least one } b\}$

Example

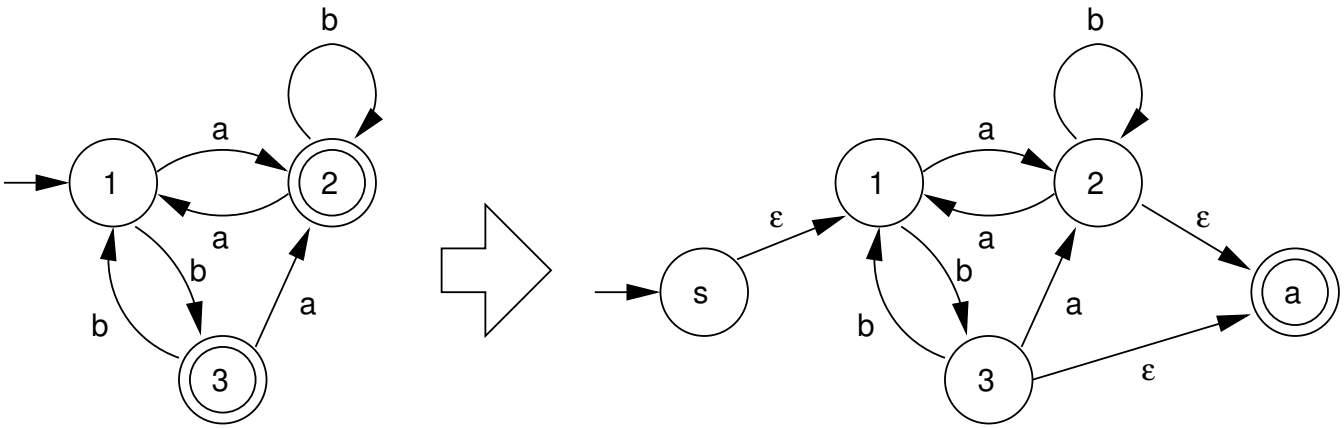
- Get rid of state 1



$\{w \mid w \text{ contains at least one } b\}$

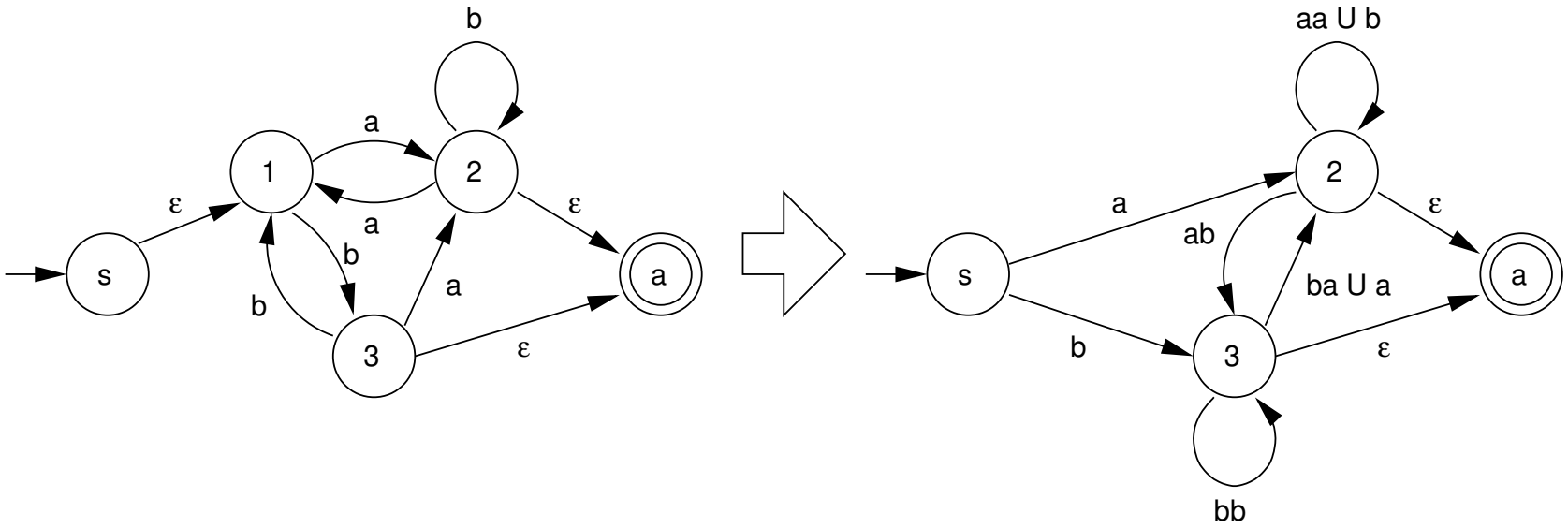
Example

- DFA to GNFA



Example

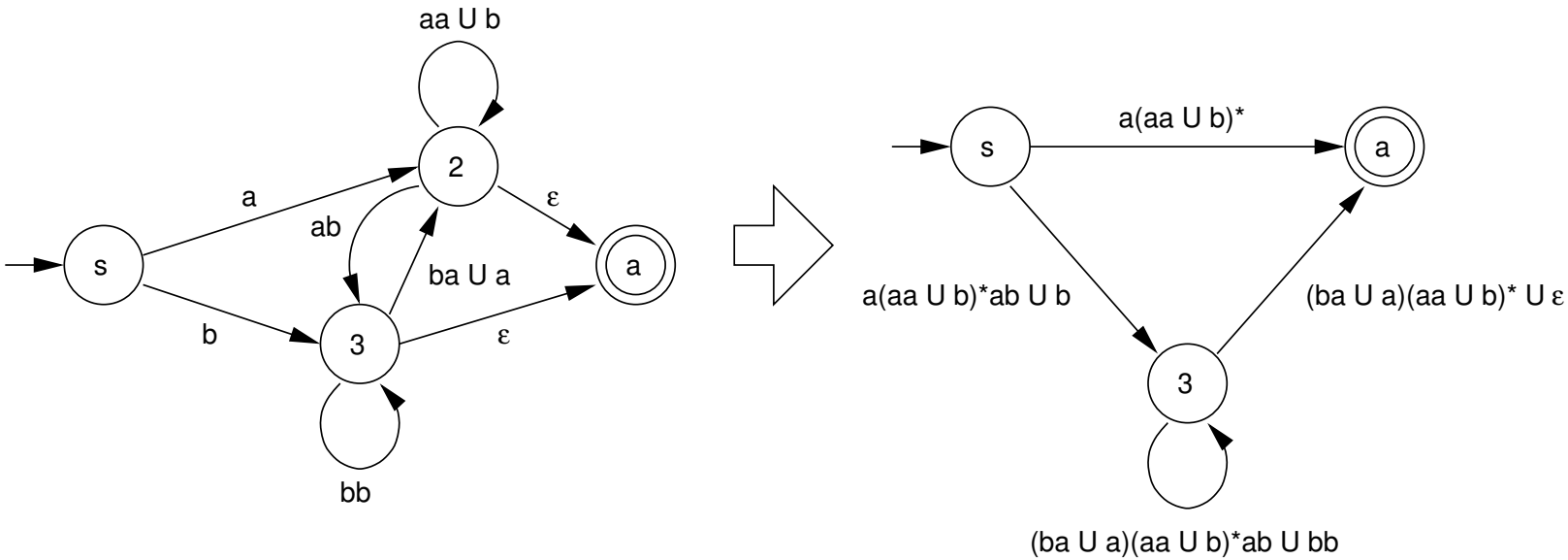
- Get rid of state 1



- Paths that go through state 1 $(q \to 1 \to r)$
  - $s \to 1 \to 2 \rightsquigarrow \varepsilon a = a$
  - $s \to 1 \to 3 \rightsquigarrow \varepsilon b = b$
  - $2 \to 1 \to 2 \rightsquigarrow aa$
  - $2 \to 1 \to 3 \rightsquigarrow ab$
  - $3 \to 1 \to 2 \rightsquigarrow ba$
  - $3 \to 1 \to 3 \rightsquigarrow bb$

Example

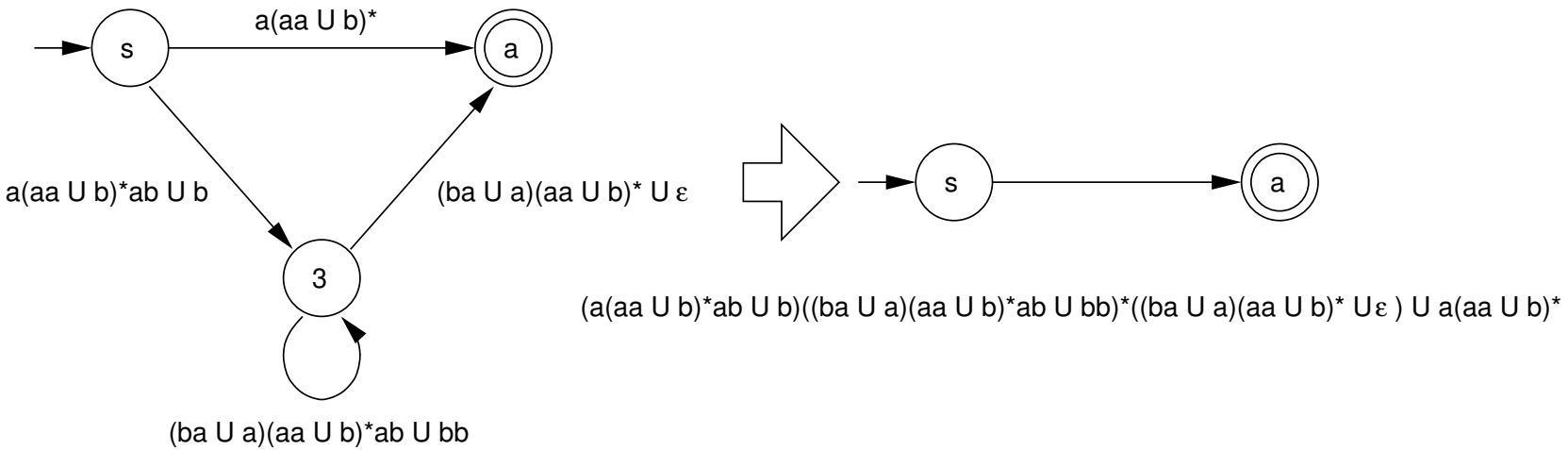
- Get rid of state 2



- Paths that go through state 2 $(q \to 2 \to r)$
  - $s \to 2 \to a \rightsquigarrow a(aa \cup b)^*\varepsilon = a(aa \cup b)^*$
  - $s \to 2 \to 3 \rightsquigarrow a(aa \cup b)^*ab$
  - $3 \to 2 \to a \rightsquigarrow (ba \cup a)(aa \cup b)^*\varepsilon = (ba \cup a)(aa \cup b)^*$
  - $3 \to 2 \to 3 \rightsquigarrow (ba \cup a)(aa \cup b)^*ab$

Example

- Get rid of state 3



- Paths that go through state 3 $(q \to 3 \to r)$
  - $s \to 3 \to a$

$$(a(aa \cup b)^* ab \cup b)((ba \cup a)(aa \cup b)^* ab \cup bb)^*((ba \cup a)(aa \cup b)^* \cup \varepsilon)$$

Theorem 1.54

- Earlier, we show that every regular expression expresses a regular language
- We just show that every state diagram of a DFA can be converted into an equivalent regular expression

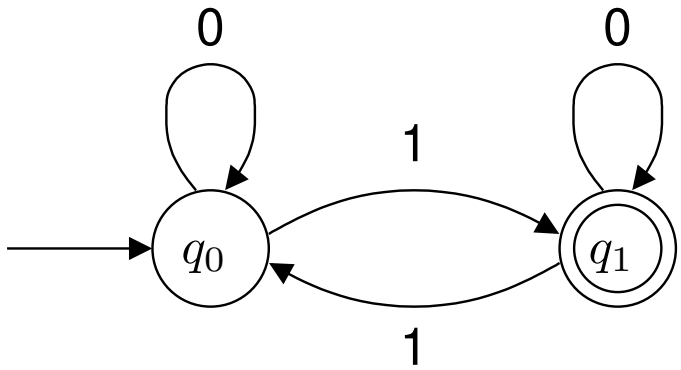
Theorem (1.54)

A language is regular if and only if some regular expression describes it.

- **Problem**: If you cannot express a language using a regular expression, it does not mean it is not regular
  - The language is so complicate that it needs a very long and complicate regular expression

Finite Automaton and Algorithm

- For every finite automaton, there is an equivalent algorithm
- For example, consider a machine where its language is the set of all strings over 0 and 1 that consist of an odd number of 1s.



- The above machine can be converted into a Java method that
  - Takes a string of 0s and 1s as an argument
  - Returns true (accept) of the input string consisting of an odd number of 1s. Otherwise, returns false (reject).

Finite Automata and Algorithm

```
public boolean isAccepted(String input)
{
    String state = "q0";
    int length = input.length();

    for(int i = 0; i < length; i++)
    {
        char c = input.charAt(i);

        if(state.equals("q0") && c == '0')
            state = "q0";
        else if(state.equals("q0") && c == '1')
            state = "q1";
        else if(state.equals("q1") && c == '0')
            state = "q1";
        else if(state.equals("q1") && c == '1')
            state = "q0";
    }

    if(state.equals("q1"))
        return true;
    else
        return false;
}
```

Limitation of Finite Automaton

- Finite Automaton only captures a small subset of algorithms
  - Problems that associated with Regular languages
- Not all language can be recognized by a finite state machine
  - Such languages is called **Nonregular Languages**
- Example: $B = \{0^n 1^n \mid n \geq 0\}$
  - $B = \{\varepsilon, 01, 0011, 000111, \dots\}$
  - The machine need to be able to remember how many 0s and how many 1s it sees so far
  - We need an **infinite** number of states
- If you cannot express a language using regular expression, it does not mean it is not a regular language
  - Too complicate and requires a huge number of states
- We need a way to determine whether a language is not regular.