AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

AIUB

DEPARTMENT OF COMPUTER SCIENCE

CSC3113: THEORY OF COMPUTATION

Lecture: # 7

Week: # 4

Semester: Spring 2022-2023

REGULAR EXPRESSIONS (RE)

Instructor: Shakila Rahman,

Department of Computer Science, Faculty of Science & Technology.

Shakila.Rahman@aiub.edu

Lecture Outline

- Formal Definition of Regular Expression (RE)
- Equivalence with Finite Automaton
- Conversion from NFA to RE
- Conversion from DFA to RE.
- Closure under regular operations.

LEARNING OBJECTIVE

- Mathematical model of Regular Expression (RE)
- Understand the uniformity of RE and FA.
- Conversion Techniques from NFA to RE.
- The strength of RE.
- Techniques to convert DFA to RE
- Closure under different regular operations.

Learning Outcome

All outcome are represented with examples

- Understand the mathematical interpretation of Regular Expression (RE)
- Learn the rules for equivalence of RE with Finite Automaton
- Apply the conversion rules from RE to NFA
- Apply the techniques to convert DFA to RE
- Identify the closure under different regular operations.

REGULAR EXPRESSION

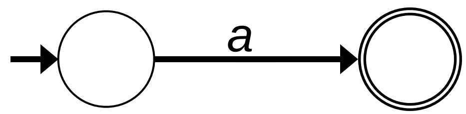
- Regular expression is used to describe languages.
- Regular expression is specific, standard textual syntax (combined with alphabets and regular operators) for representing patterns for matching strings.
- Regular expression can be built up using regular operations.
- Precedence order: $* \bullet \cup$
- Example:
  - $(0\cup1)0^* = (\{0\}\cup\{1\})\bullet\{0\}^* = \{0,1\}\bullet\{0\}^*$
    $A = \{w \mid$ string w starts with a 0 or a 1 followed by zero or more 0's$\}$
  - $(0\cup1)^* = (\{0\}\cup\{1\})^* = \{0,1\}^*$
    A = {all possible string with 0s and/or 1s}.

Formal Definition of Regular Expression

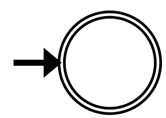
- ***R*** is a regular expression if ***R*** is –
  - a for some $a \in \Sigma$, represents the language $\{a\}$.
  - ε, represents the language $\{\varepsilon\}$ containing a single string, namely, the empty string.
  - ϕ, represents the empty language that doesn't contain any string. $L(\phi^*) = \{\varepsilon\}$.
  - $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
    - $R \cup \phi = R$, but $R \cup \varepsilon$ may not be equal to R.
  - $(R_1 \bullet R_2)$, where R_1 and R_2 are regular expressions,
    - $R \bullet \varepsilon = R$, but $R \bullet \phi$ may not be equal to R.
  - $(R_1{}^*)$, where R_1 is a regular expressions,

EQUIVALENCE WITH FINITE AUTOMATA

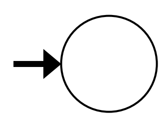
- Let convert regular language R into an NFA considering the six cases in the formal definition of regular language.
  - $R = a$, $a \in \Sigma$. Then $L(R)=\{a\}$, and the NFA that recognizes $L(R)$ is –



  - $R = \varepsilon$. Then $L(R)=\{\varepsilon\}$, and the NFA that recognizes $L(R)$ is –

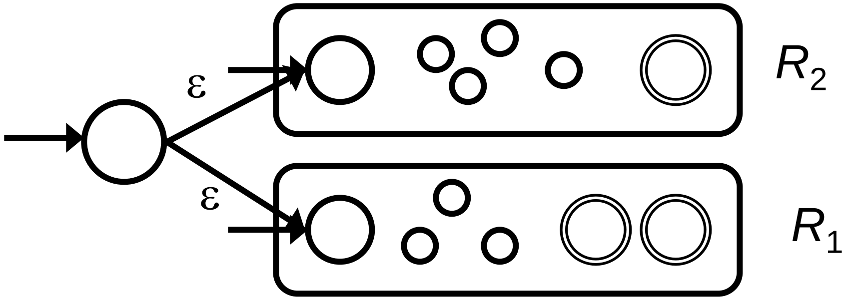


  - $R = \phi$. Then $L(R)= \phi$, and the NFA that recognizes $L(R)$ is –

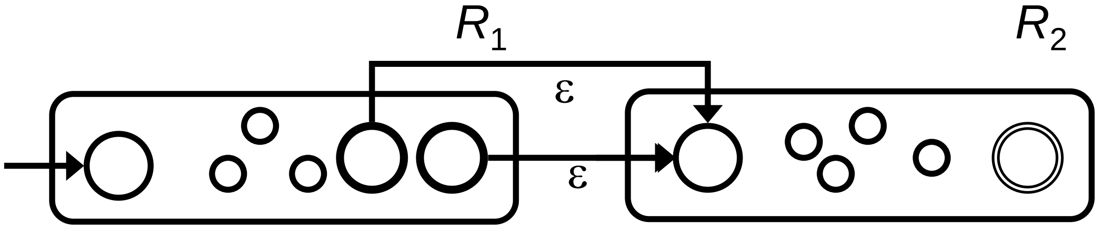


EQUIVALENCE WITH FINITE AUTOMATA

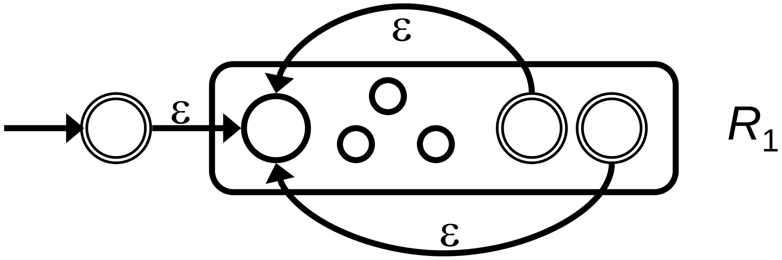
- $R = R_1 \cup R_2$. Then $L(R)=\{R_1,R_2\}$, and the NFA that recognizes $L(R)$ is –



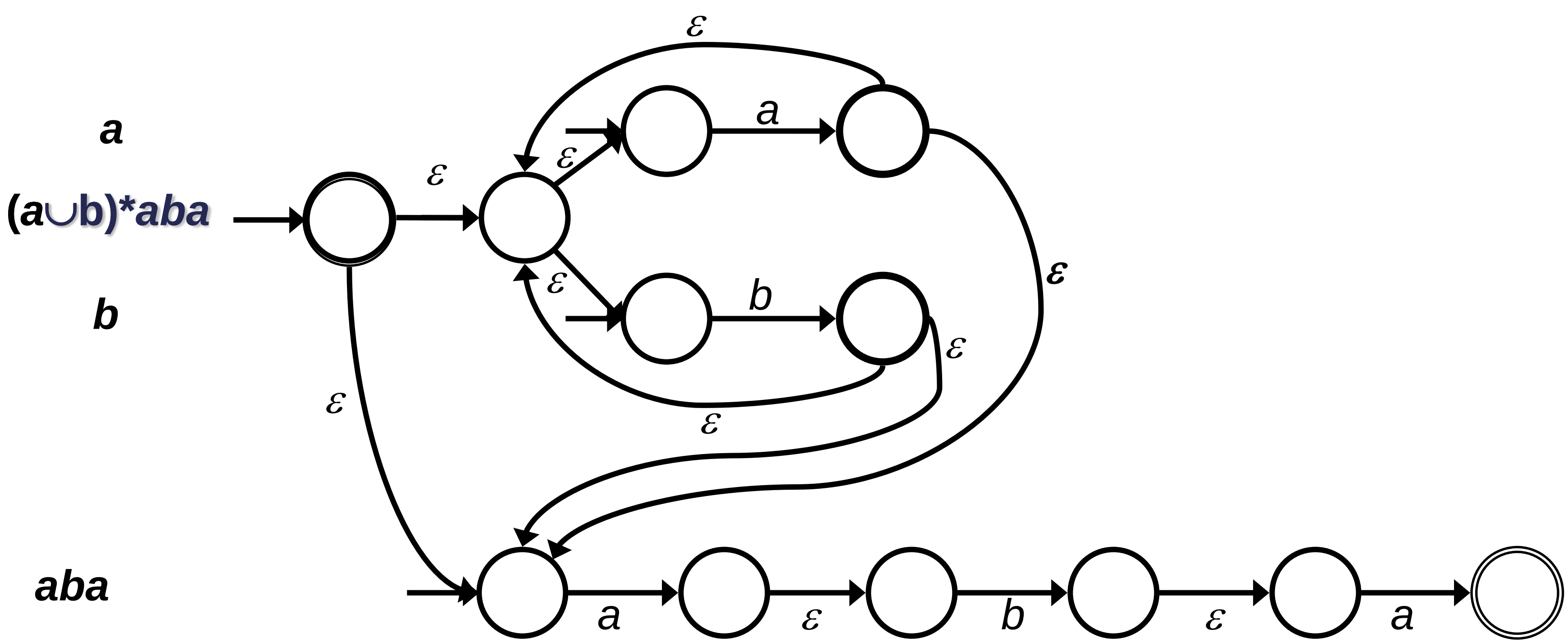
- $R = R_1 \bullet R_2$. Then $L(R)=\{R_1R_2\}$, and the NFA that recognizes $L(R)$ is –



- $R = R_1{}^*$. Then $L(R)=\{R_1\}^*$, and the NFA that recognizes $L(R)$ is –



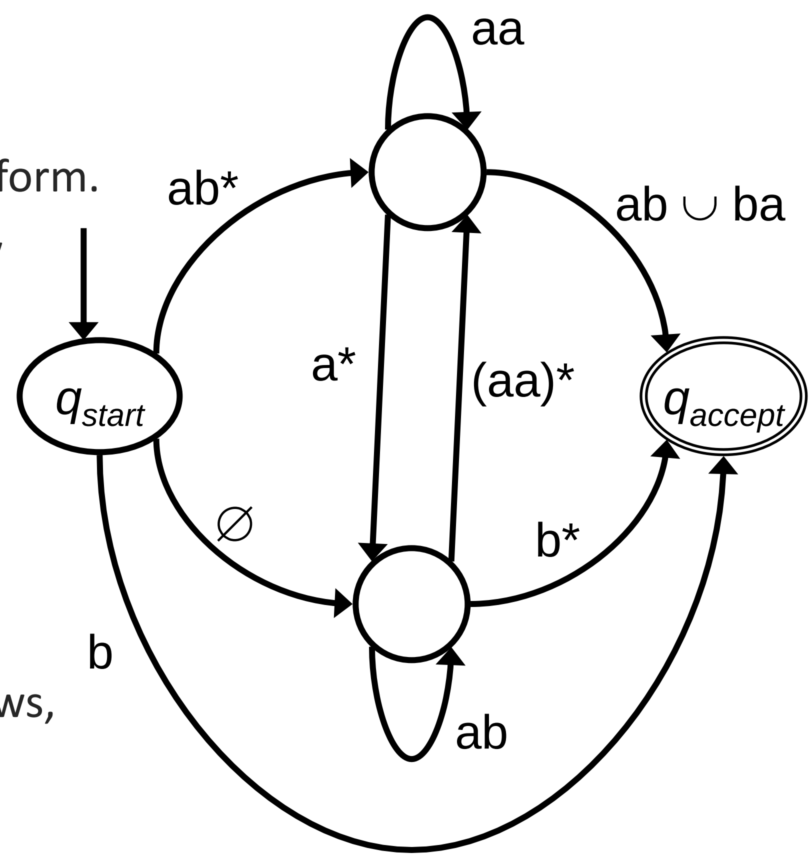
Converting a regular expression to an NFA



Building an NFA from regular expression: (*a*∪*b*)**aba*

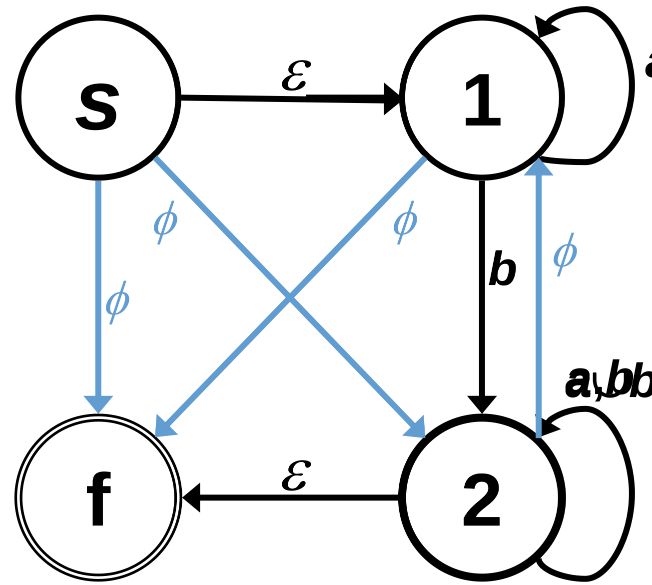
CONVERTING A DFA TO A REGULAR EXPRESSION

- This can be done in two parts. For this we introduce a new type of finite automata called ***generalized nondeterministic automaton***, GNFA.
  - First, we will convert a DFA to GNFA, and
  - then GNFA to regular expression.
- GNFA has the following special form –
  - Transition labels might be in regular expression form.
  - The start state doesn't have any incoming arrow from any other state.
  - There is only one accept state, and it doesn't have any outgoing arrow to any other state.
  - Start state is never the same as accept state.
  - There is only one outgoing arrow to any other state and to itself, except the start and accept states. We will consider ϕ labeled outgoing arrows, if no transition exists between any two states.



CONVERTING A DFA TO GNFA

- Add a new start state with an ε arrow to the old start state.
- Add new accept state with ε arrows from the old accept states.
- If any arrows have multiple labels, union the previous labels into one label.
- Add arrows with ϕ label between states where there are no arrows. This won't change the language as ϕ label arrows can never be used.
  - Even we might ignore adding such arrows, as these are arrows which can be assumed to be there with no use.



FORMAL DEFINITION OF GNFA

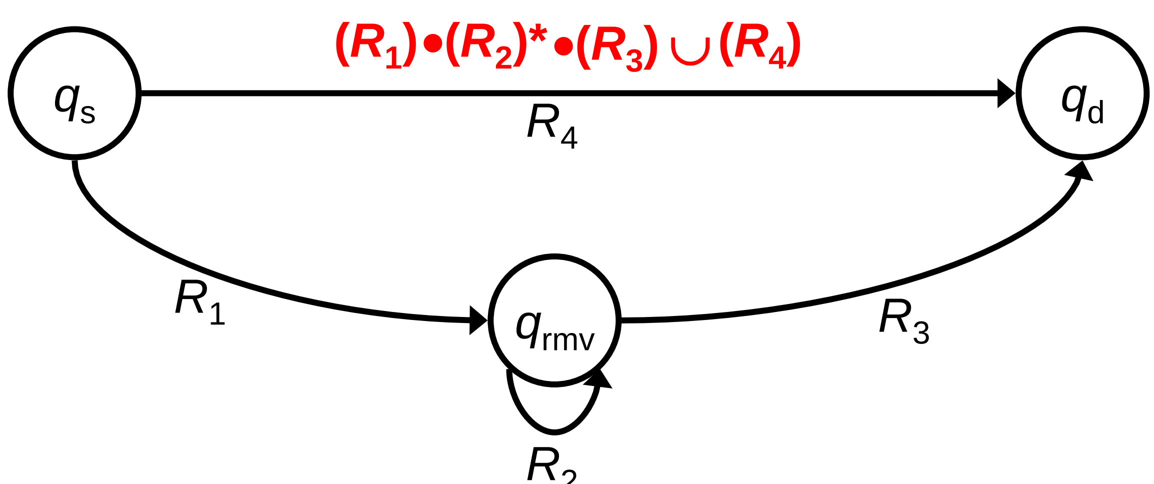
- A generalized nondeterministic finite automaton is a 5-tuple, $(Q, \Sigma, \delta, q_{start}, q_{accept})$ where –
  - Q is the finite set of states,
  - Σ is the input alphabet,
  - $\delta : (Q - \{q_{start}\}) \times (Q - \{q_{accept}\}) \rightarrow \mathcal{R}$ is the transition function,
  - q_{start} is the start state,
  - q_{accept} is the accept state.
- A GNFA accepts a string w in Σ^* if $w = w_1w_2\ldots w_k$, where each w_i is in Σ^* and a sequence of states $q_0, q_1, \ldots q_k$ exists such that –
  - $q_0=q_{start}$ is the start state,
  - $q_k=q_{accept}$ is the accept state, and
  - For each i, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$; i.e., R_i is the expression on the arrow from q_{i-1} to q_i.

CONVERTING A GNFA TO A REGULAR EXPRESSION

- Let consider the GNFA to be with k states.
- We will continuously remove one state from the GNFA until $k = 2$. These last two states are actually the start and the accept states.
- We do so by selecting a state, ripping it out of the machine, and ***repairing*** the remainder so that the same language is still recognized.
- Any state will do, provided that the state is not the start or the accept states.

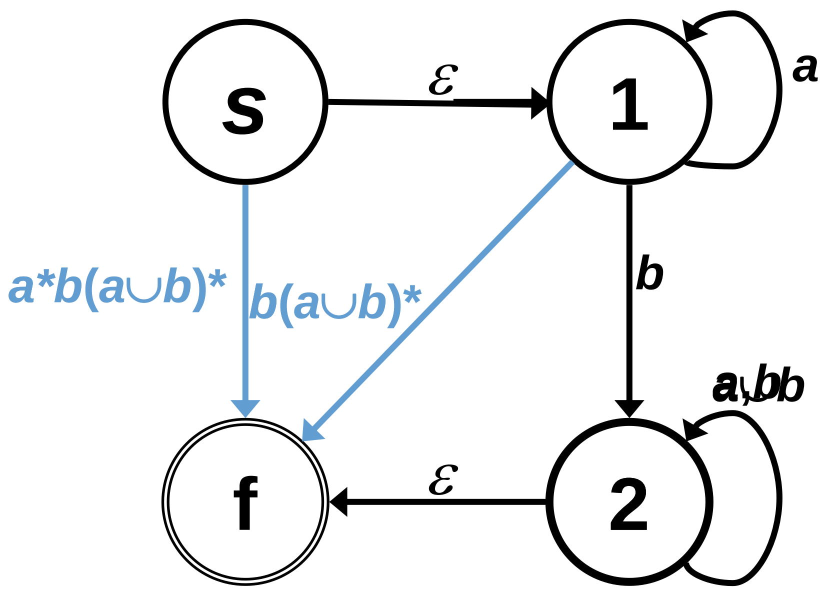
REPAIRING AFTER REMOVING A STATE

- Let us call the removed state q_{rmv}.
- Repair the machine by altering the regular expressions that label each of the remaining arrows. This change is done for each arrow going from any state q_s to q_d, including the case where $q_s = q_d$.
- The new labels compensate for the absence of q_{rmv} by adding back the lost computations. i.e., The new label going from a state q_s to state q_d is a regular expression that describes all strings that would take the machine from q_s to q_d either directly or via q_{rmv}.



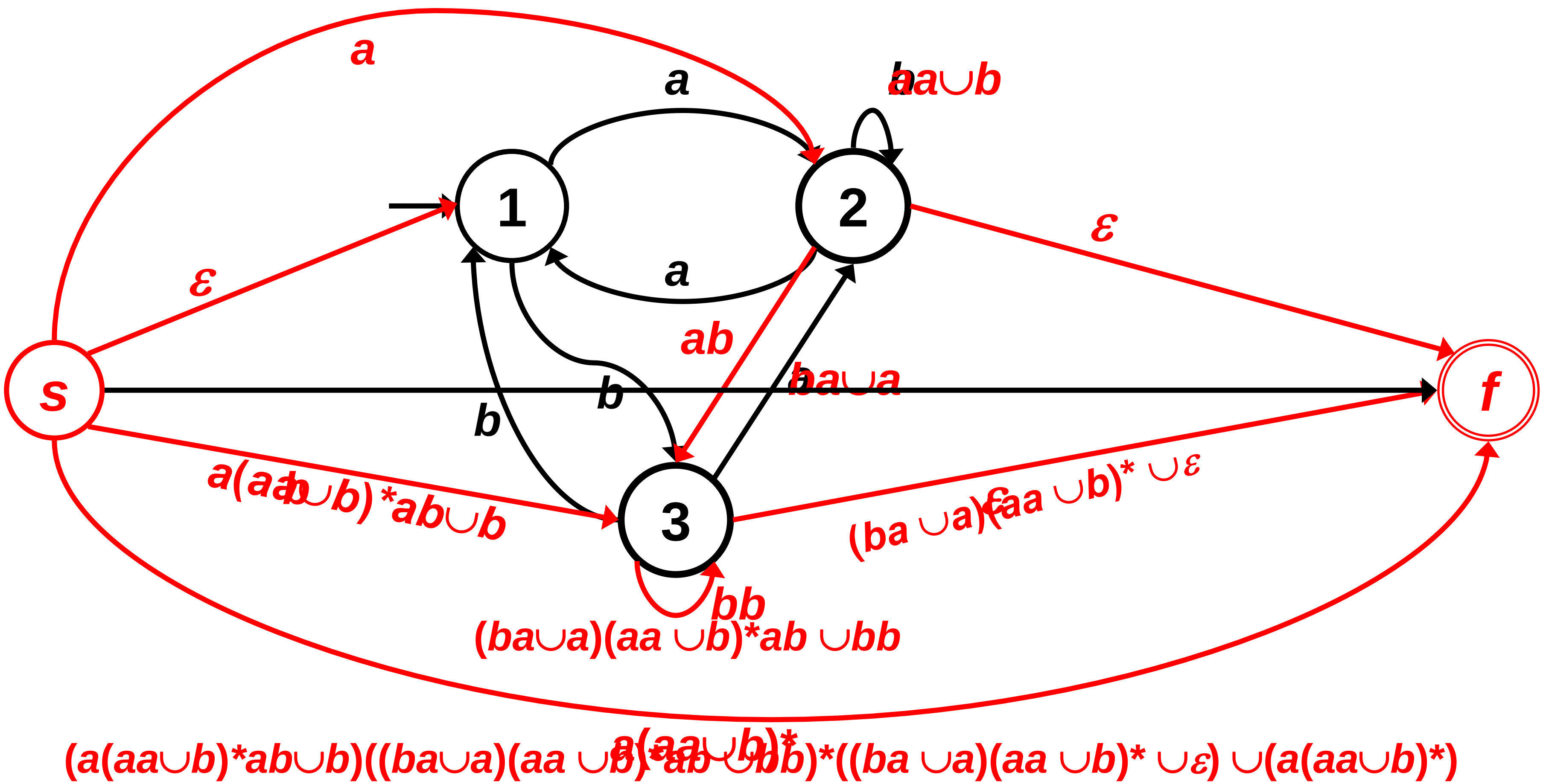
EXAMPLE:

CONVERTING A TWO STATE DFA TO AN EQUIVALENT REGULAR EXPRESSION



EXAMPLE:

CONVERTING A THREE STATE DFA TO AN EQUIVALENT REGULAR EXPRESSION



CLOSURE

- Regular Languages are Closed Under Regular Operations:
  - **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:** $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$
  - **Negation:** $\neg A = \{ w \mid w \notin A \}$
  - **Concatenation:** $A \cdot B = \{ vw \mid v \in A \text{ and } w \in B \}$
  - **Star:** $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

REFERENCES

- Introduction to Theory of Computation, Sipser, (3^{rd} ed), Regular Expression.