جامعة الملك سعود
King Saud University

Theory of Computation
CSC 339 – Spring 2021

Chapter-1: part3
Regular Languages

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Outline

- Recap
- Introduction
- Regular Expressions

Recap

- **Non-deterministic Finite Automata (NFA)**
  - **Given a word w, the automaton can end up in different state each time.**
  - **NFA provides us with an easier ways to design finite automata.. But, we can always convert it to equivalent DFA.**

Regular Expressions: Introduction

- **In arithmetic, we often use operations + and x to build up expressions such as:**
  - **(5+3) x 4**

Regular Expressions: Introduction

- **In arithmetic, we often use operations + and x to build up expressions such as:**
  - **(5+3) x 4**
- **In a similar fashion, we can use regular operations to build up expressions describing languages.**
  - **(0∪1)0***

Regular Expressions: Introduction

- Regular expressions are useful in many text-based applications
  - Compilers, search engines, log processing, information retrieval, etc
- In arithmetic, we know that x has precedence over +
- In regular expressions, order of operations:
  - Star
  - Concatenation
  - Union

Regular Expressions: Definition 1.52

- **Say that R is a regular expression if R is**
  - **a for some a in the alphabet Σ,**
  - **ε,**
  - **∅,**
  - **(R1 ∪ R2), where R1 and R2 are regular expressions,**
  - **(R1 ∘ R2), where R1 and R2 are regular expressions, or**
  - **(R1∗), where R1 is a regular expression.**

Regular Expressions: Definition 1.52

- **Say that R is a regular expression if R is**
  - **a for some a in the alphabet Σ,**
  - **ε,**
  - ∅,
  - **(R1 ∪ R2), where R1 and R2 are regular expressions,**
  - **(R1 ∘ R2), where R1 and R2 are regular expressions, or**
  - **(R1∗), where R1 is a regular expression.**

Represent the languages {a} and {ε}

Regular Expressions: Definition 1.52

- **Say that R is a regular expression if R is**
  - **a for some a in the alphabet Σ,**
  - **ε,**
  - **∅,**
  - **(R1 ∪ R2), where R1 and R2 are regular expressions,**
  - **(R1 ∘ R2), where R1 and R2 are regular expressions, or**
  - **(R1∗), where R1 is a regular expression.**

Represent the empty language

Regular Expressions: Examples

- **R^***
  - **All strings that are <u>zero or more</u> concatenations of strings from R**
  - **$R^+ \cup \varepsilon = R^*$**
- **R^+**
  - **All strings that are <u>one or more</u> concatenations of strings from R**
  - **Shorthand for RR^***

Regular Expressions: Examples

- 0^*10^* = {w| w contains a single 1}
- $\Sigma^*1\Sigma^*$ = {w| w has at least one 1}
- $\Sigma^*001\Sigma^*$ = {w| w contains the string 001 as a substring}
- $(\Sigma\,\Sigma)^*$ = {w| w is a string of even length}
- $(\Sigma\,\Sigma\,\Sigma)^*$ = {w| the length of w is a multiple of 3}
- $(0 \cup \varepsilon)(1 \cup \varepsilon)$ = { ε , 0, 1, 01}

Regular Expressions: Identities

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$
- What if we flip the $\emptyset$ and ε?

Regular Expressions: Use-cases

- **Regular expressions have numerous text-based applications in computer science.**

Regular Expressions: Use-cases

- **Regular expressions have numerous text-based applications in computer science.**
  - **Compilers for programming languages**

Regular Expressions: Use-cases

- **Regular expressions have numerous text-based applications in computer science.**
  - **Compilers for programming languages**

Example for defining the format of numerical constants with a fractional part and/or a sign

$$(+ \cup - \cup \varepsilon)\,(D^{+} \cup D^{+} . D^{*} \cup D^{*} . D^{+})$$

Regular Expressions: Use-cases

- **Regular expressions have numerous text-based applications in computer science.**
  - **Compilers for programming languages**

Example for defining the format of numerical constants with a fractional part and/or a sign

$$(+ \cup - \cup \varepsilon)\,(D^{+} \cup D^{+} . D^{*} \cup D^{*} . D^{+})$$

Examples of accepted expressions: **72**, **3.14159**, **+7.**, **-0.1**

Regular Expressions: Use-cases

- **Regular expressions have numerous text-based applications in computer science.**
  - **Compilers for programming languages**
  - **Log analysis and debugging large systems**
  - **Natural language processing (NLP)**

Regular Expressions: equivalence w/ finite automata

- **Regular expressions are equivalent with finite automata.**

- **Any regular expression can be converted into a finite automaton that recognizes the language it describes.**

Regular Expressions: theorem 1.54

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

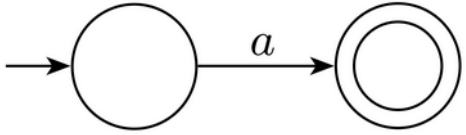
Regular Expressions: theorem 1.54

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

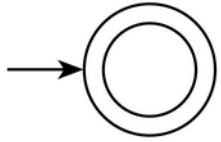
- **Two-direction theorem (iff)**
- **First direction:**
  - **If we can convert a regular expression R into NFA, then we can prove that the language R describes is regular.**
- **Back to definition of a regular expression.**

Regular Expressions: theorem 1.54

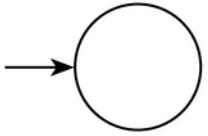
- R = a for some a ∈ Σ. Then L(R) = {a} and the following NFA recognizes L(R).



- R = ε. Then L(R) = { ε }



- R = Φ. Then L(R) = { Φ }



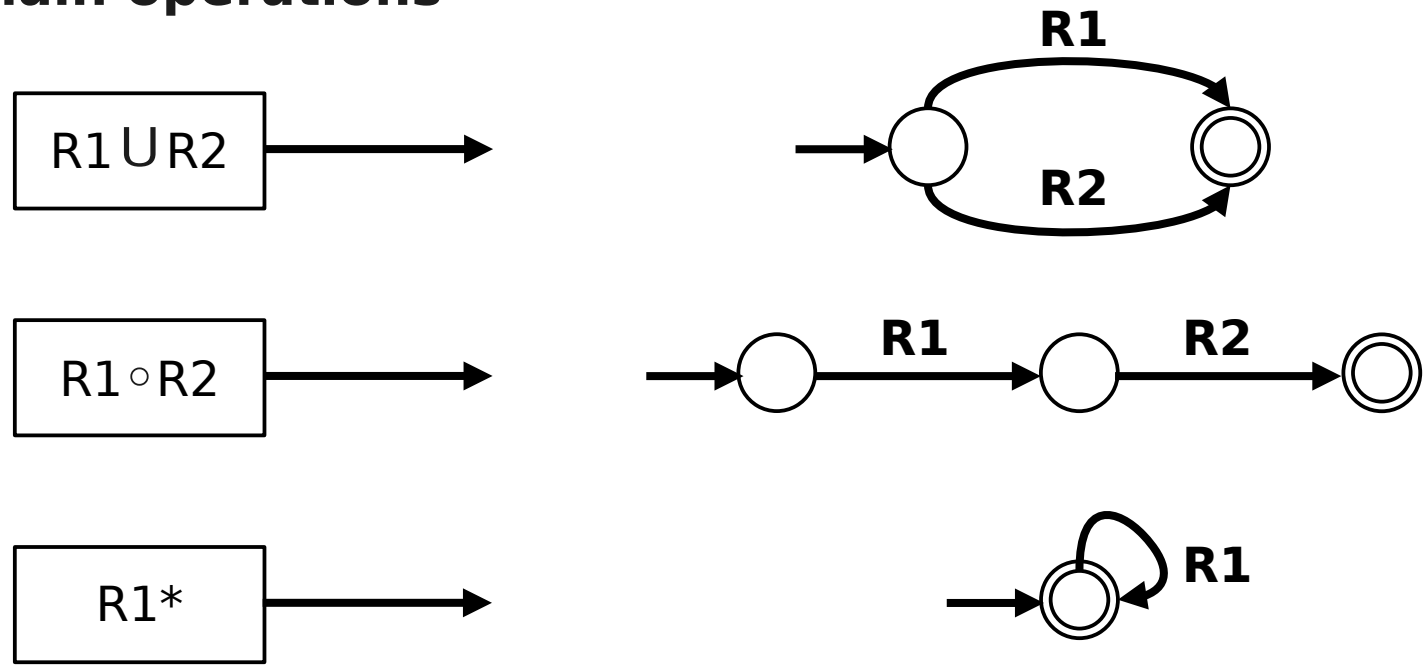
Regular Expressions: theorem 1.54

- **(R1 ∪ R2), where R1 and R2 are regular expressions,**
- **(R1 ∘ R2), where R1 and R2 are regular expressions, or**
- **(R1∗), where R1 is a regular expression.**

- **For these three cases, we can use constructions given in previous proofs that the class of regular languages is closed under the regular operations.**

Regular Expressions: Regex to NFA

- **Three main operations**



Regular Expressions: Regex to NFA - Examples

- **Three main operations**

  1.ab*a

  2.(a U b)a

  3.a(bb)*

  4.(ba)*

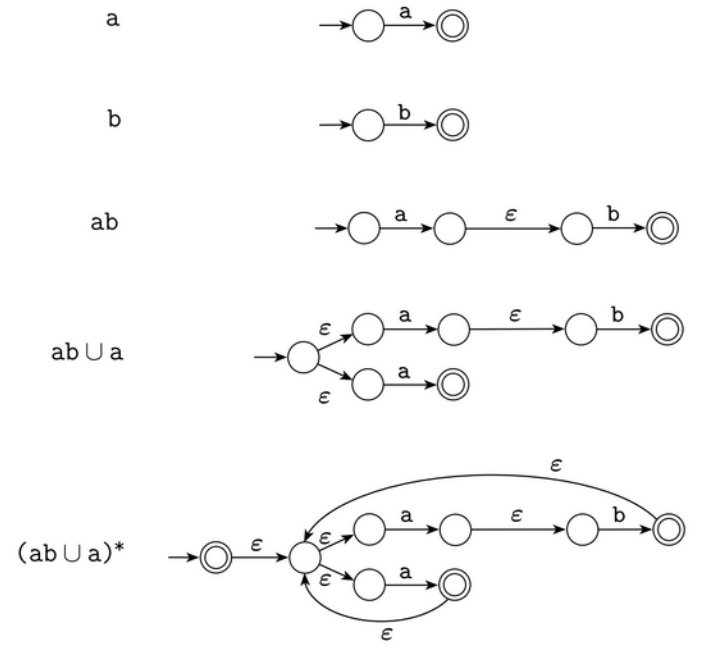
  5.$(ba)^{+}$

Regular Expressions: Example 1.56

- Let’s convert the regular expression (ab ∪ a)∗ into an NFA.

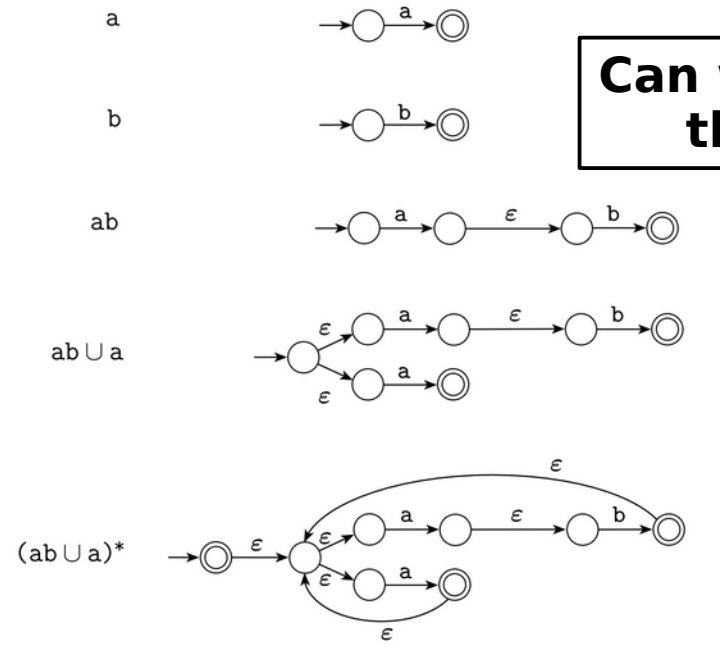
Regular Expressions: Example 1.56

- **Let’s convert the regular expression (ab ∪ a)∗ into an NFA.**



Regular Expressions: Example 1.56

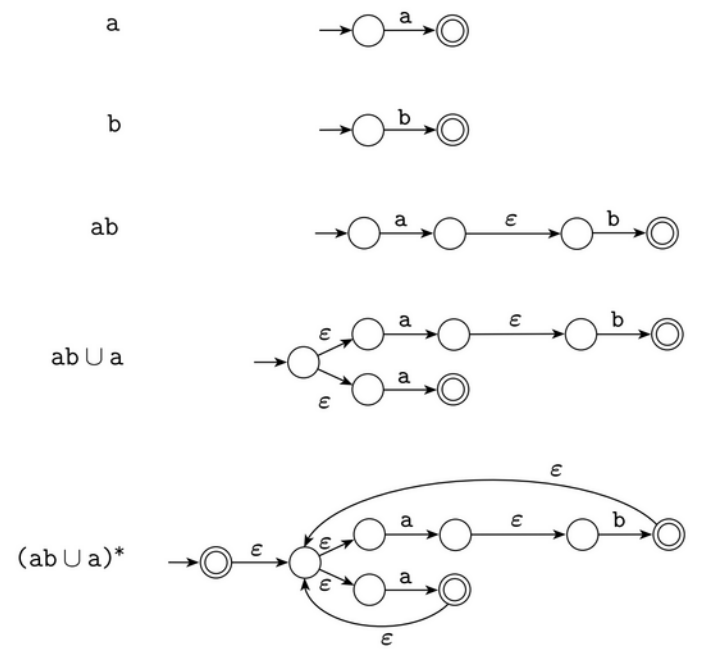
- Let's convert the regular expression (ab ∪ a)∗ into an NFA.



Can we simplify this NFA?

Regular Expressions: Example 1.56

- **Let’s convert the regular expression (ab ∪ a)∗ into an NFA.**



Also see example 1.58

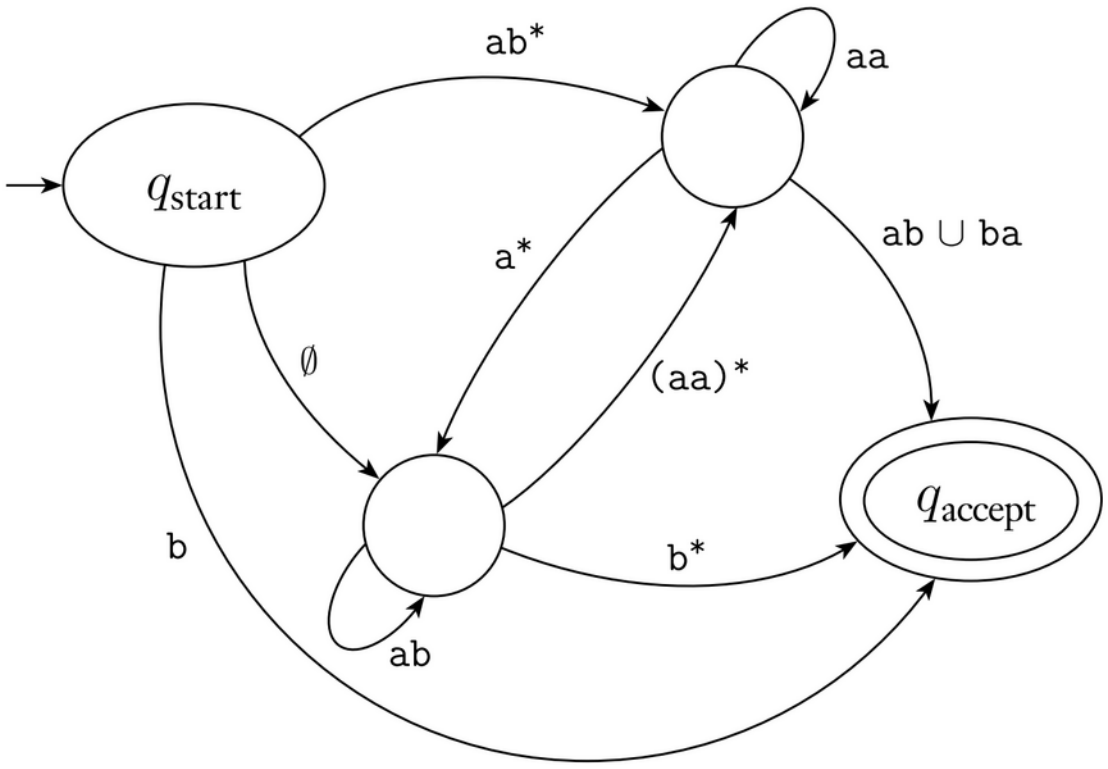
Regular Expressions: GNFA

- **The other direction of theorem 1.54**
  - *"If a language is regular, then it is described by a regular expression"*
- **Converting an NFA into a regular expression**
- **Generalized nondeterministic finite automata (GNFA)**
  - **Transition arrows may have regular expressions as labels**
  - **GNFA reads blocks of symbols from the input** – as opposed to reading symbol by symbol

Regular Expressions: GNFA

- **Special conditions**
  - **Start state has outgoing transition arrows to every other state (but, no incoming arrows).**
  - **Only single accept state that has arrows from every other state.**
  - **Except for start and accept states, one arrow goes from every state to every other state, and from each state to itself.**

Regular Expressions: GNFA



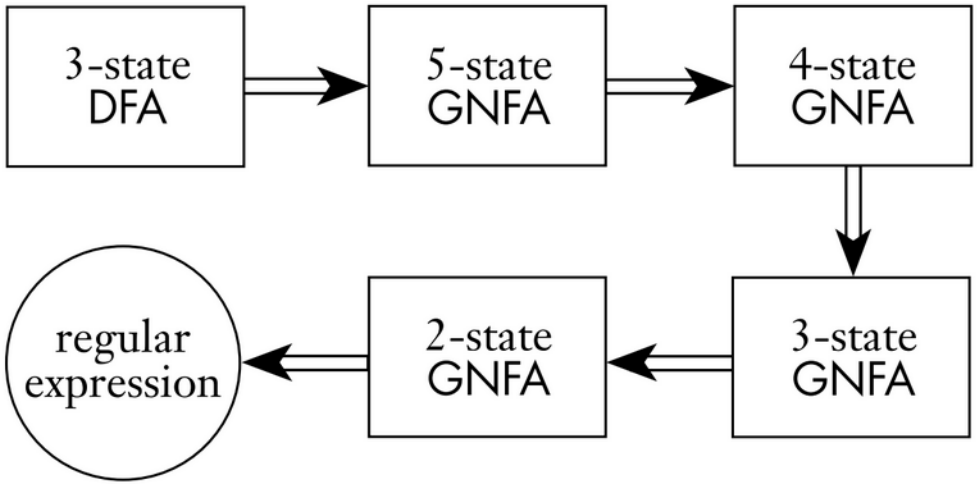
Regular Expressions: GNFA

- **Converting DFA into GNFA**
  - **Add new start state with an ε arrow to the old start state.**
  - **Add new accept state with ε arrows from old accept states.**

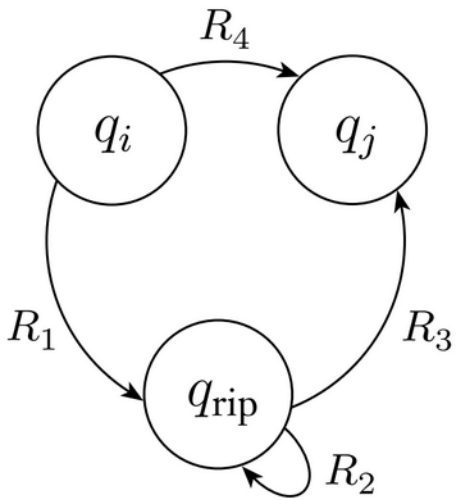
Regular Expressions: DFA to Regex

- We will follow a state elimination technique
- Convert the DFA into a GNFA, and then, start eliminating states by converting them into corresponding regular expressions.

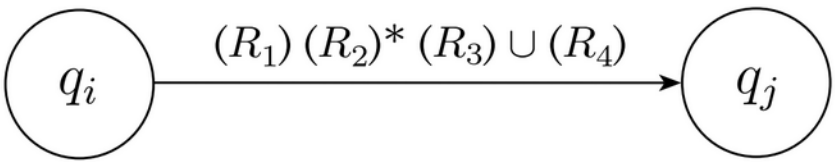
Regular Expressions: GNFA



Regular Expressions: GNFA

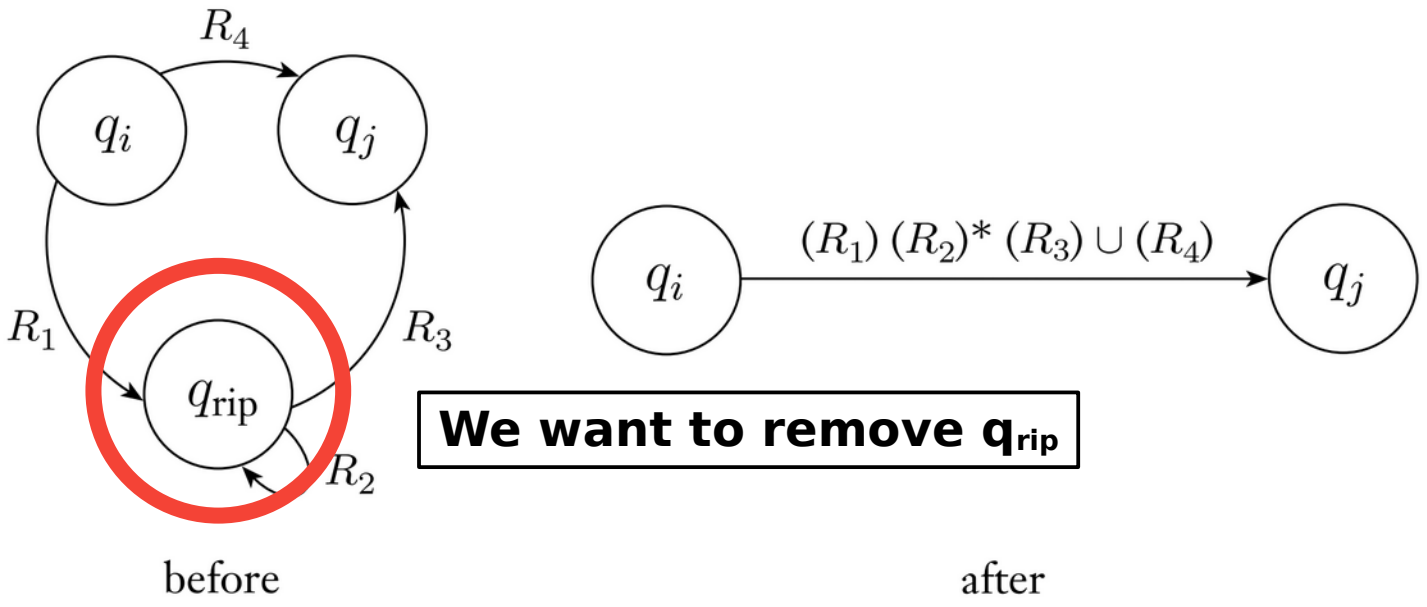


before



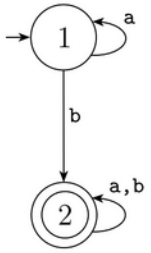
after

Regular Expressions: GNFA



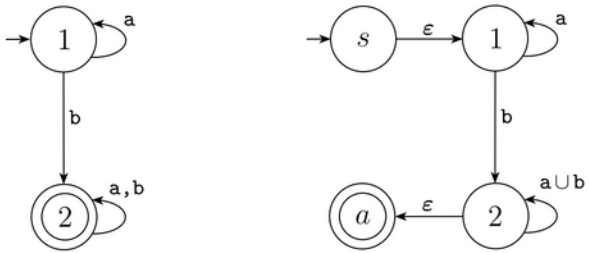
We want to remove q_{rip}

Regular Expressions: GNFA



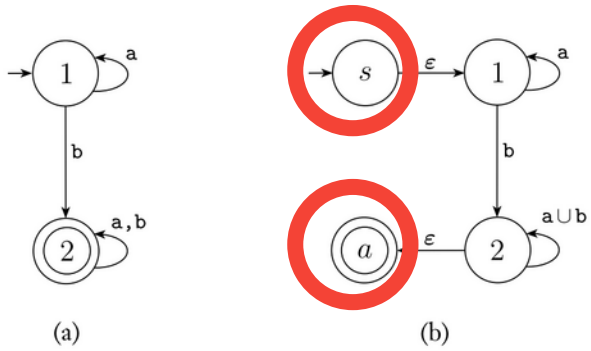
(a)

Regular Expressions: GNFA

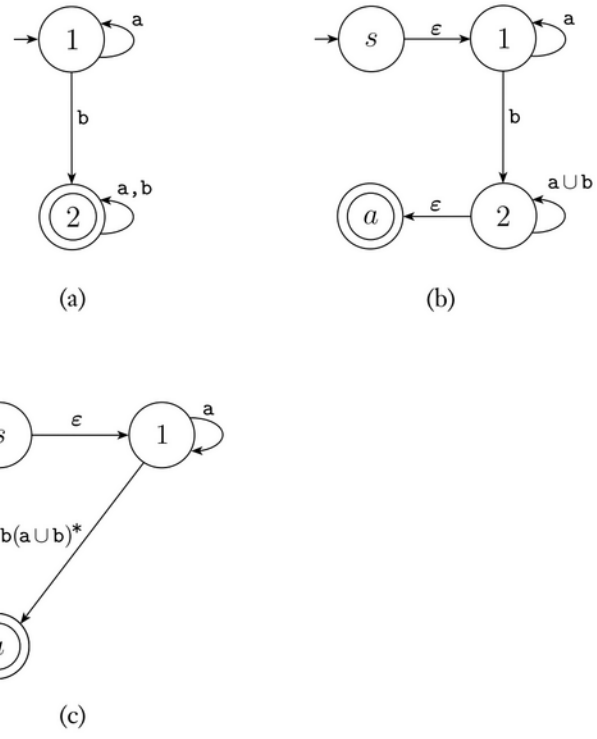


Regular Expressions: GNFA

Add new start and accept states

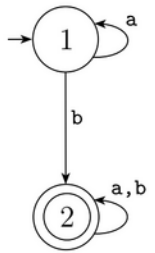


Regular Expressions: GNFA

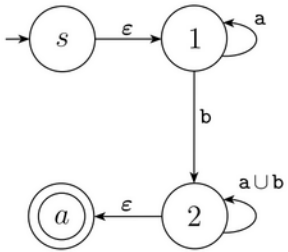


Regular Expressions: GNFA

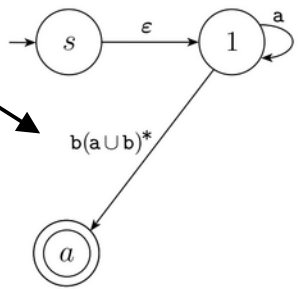
We've eliminated state(2) and replaced it with a regular expression that describes its transitions



(a)

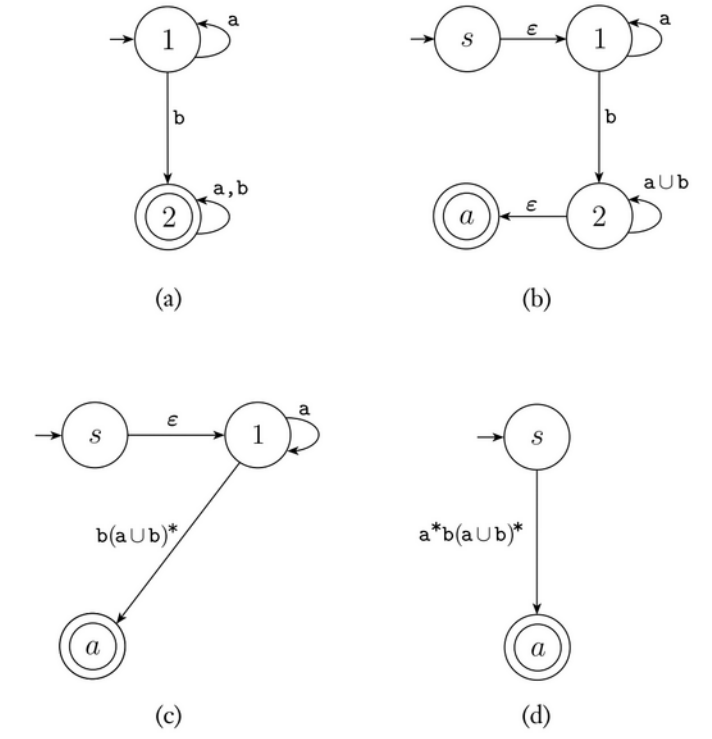


(b)

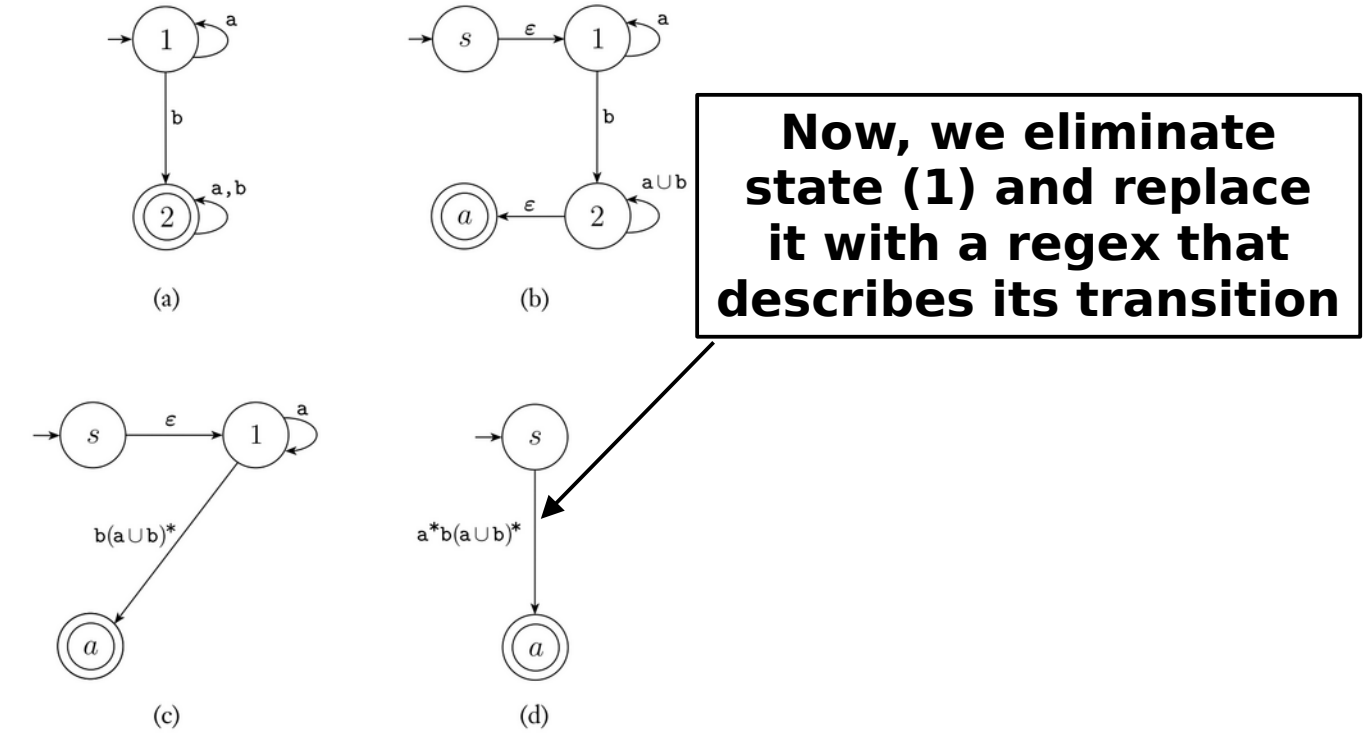


(c)

Regular Expressions: GNFA

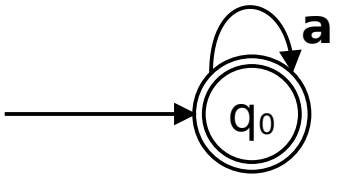


Regular Expressions: GNFA



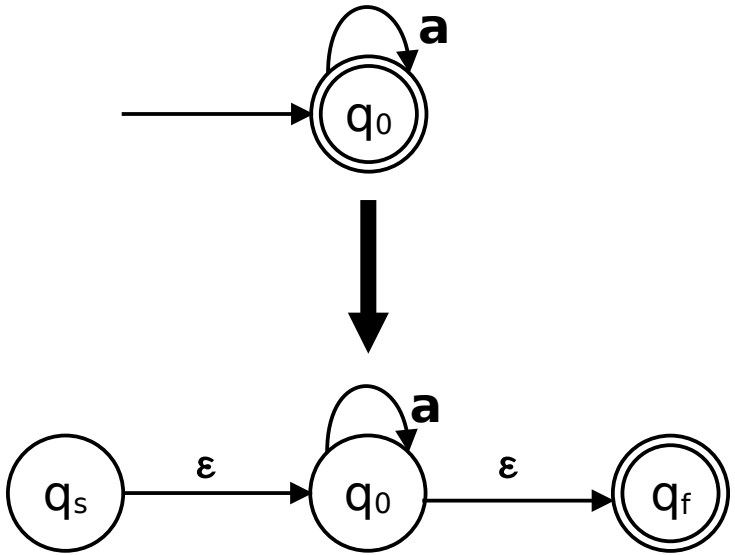
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



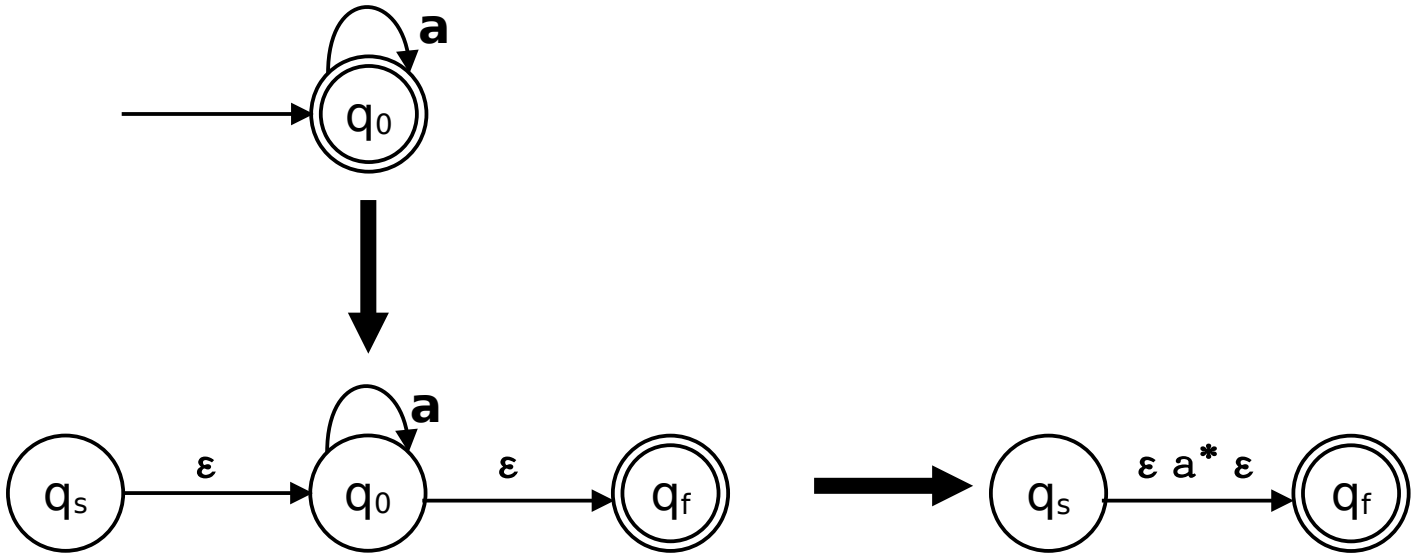
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



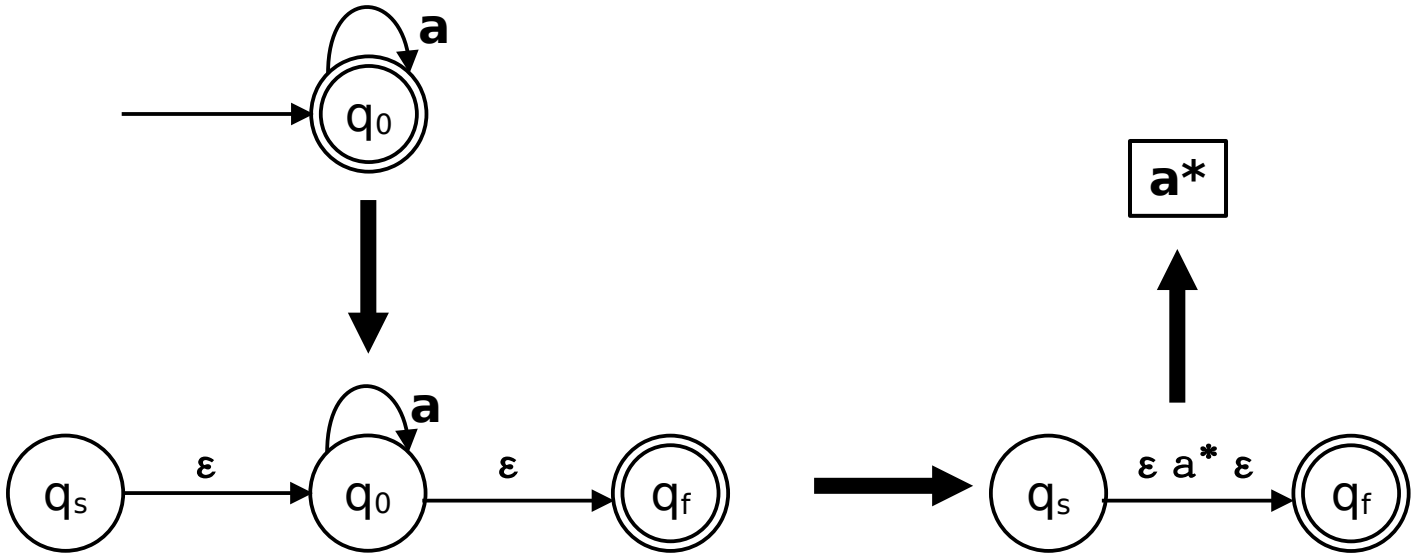
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



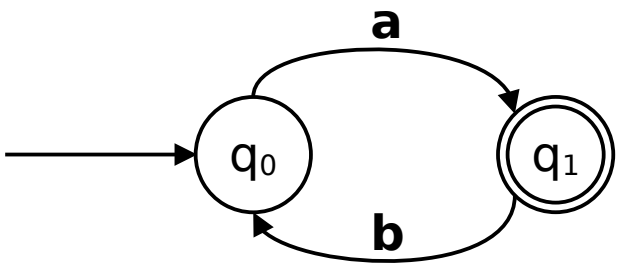
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



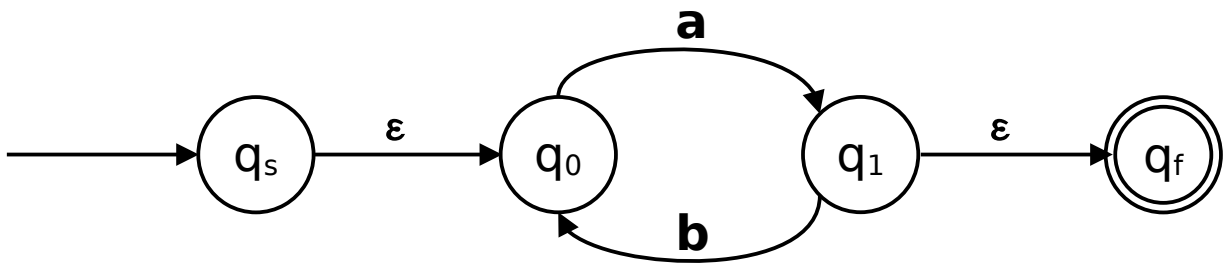
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



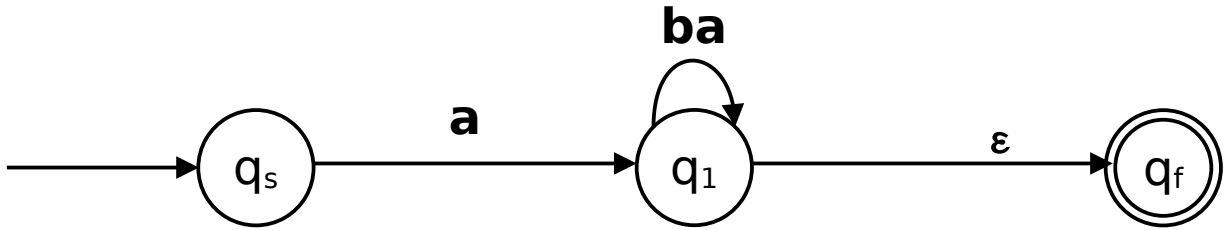
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



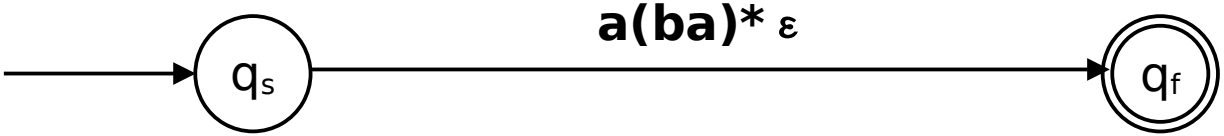
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



Regular Expressions: DFA to Regex

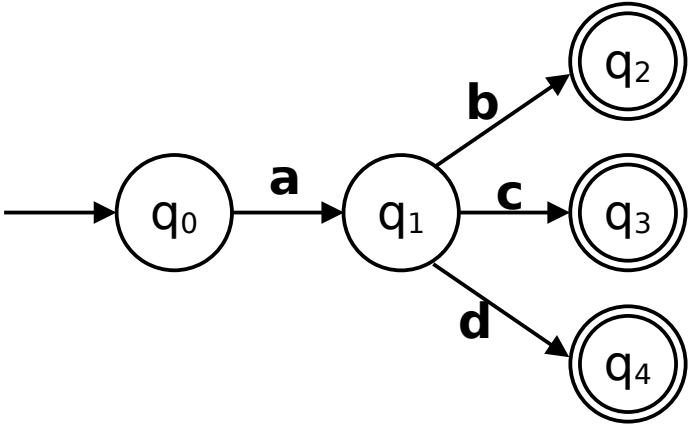
- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



a(ba)*

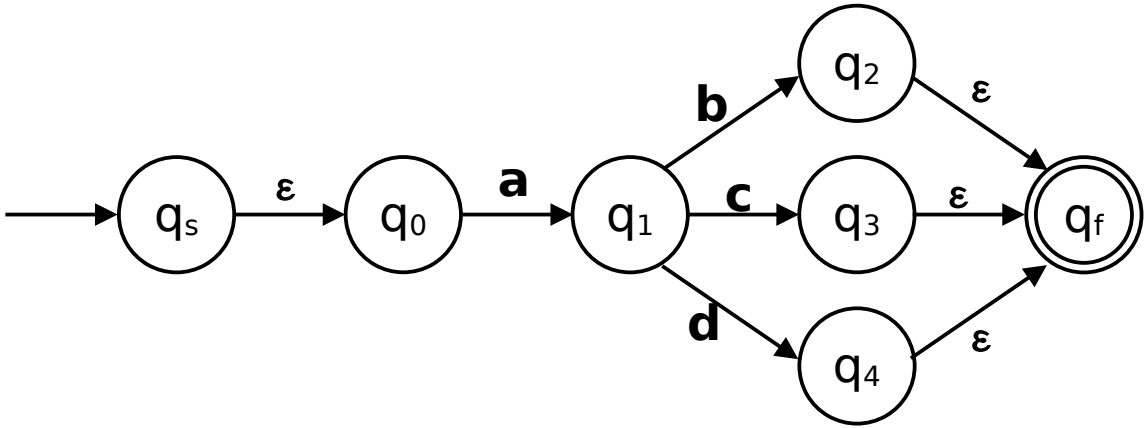
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



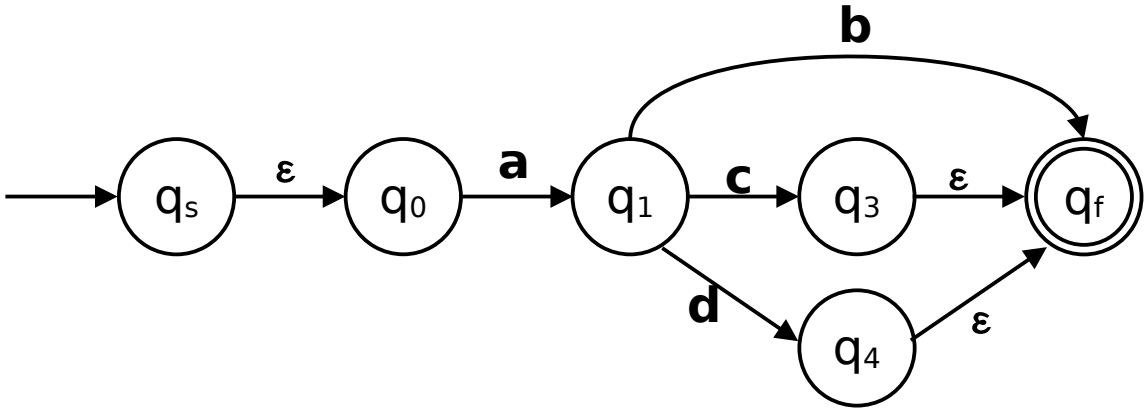
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



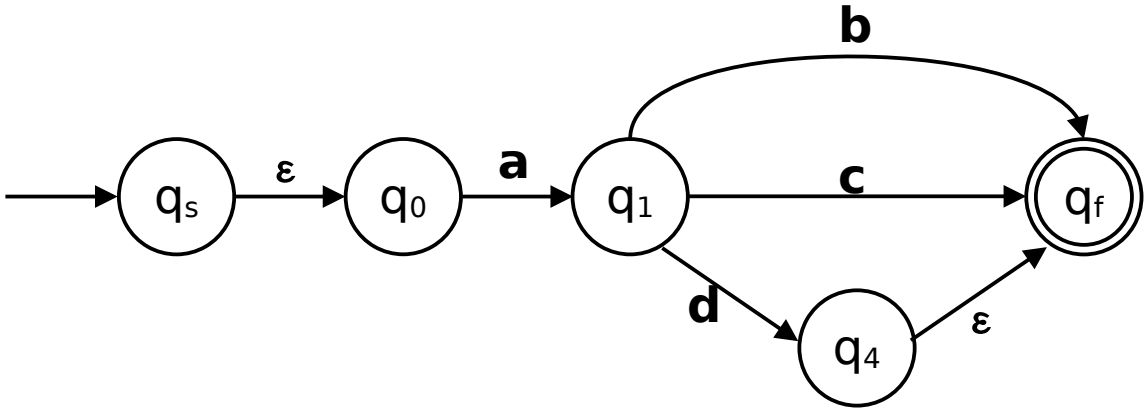
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



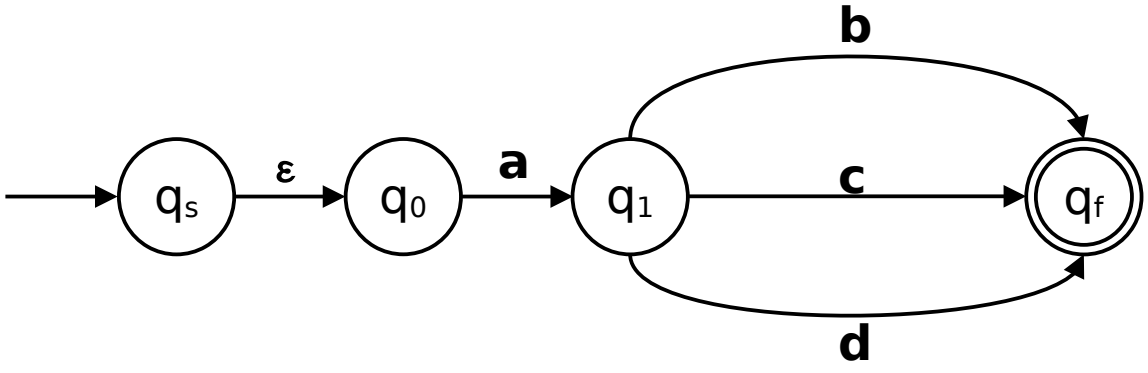
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



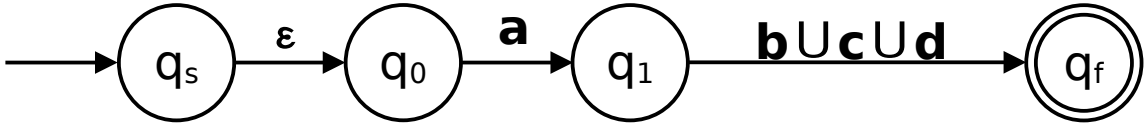
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



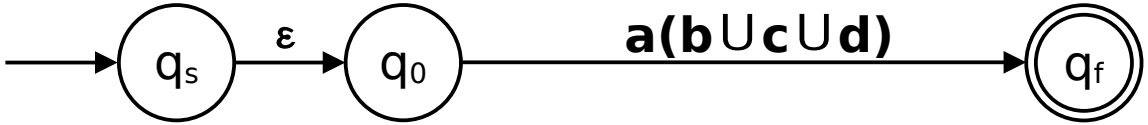
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



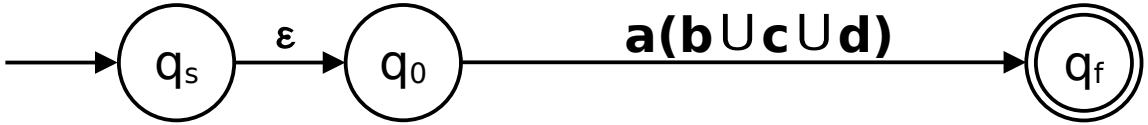
Regular Expressions: DFA to Regex

- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



Regular Expressions: DFA to Regex

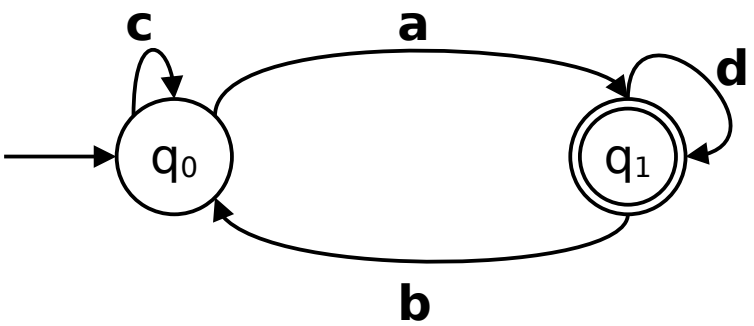
- **Convert the following DFA into regular expressions**
- **We want to reserve the same language.**



a(b∪c∪d)

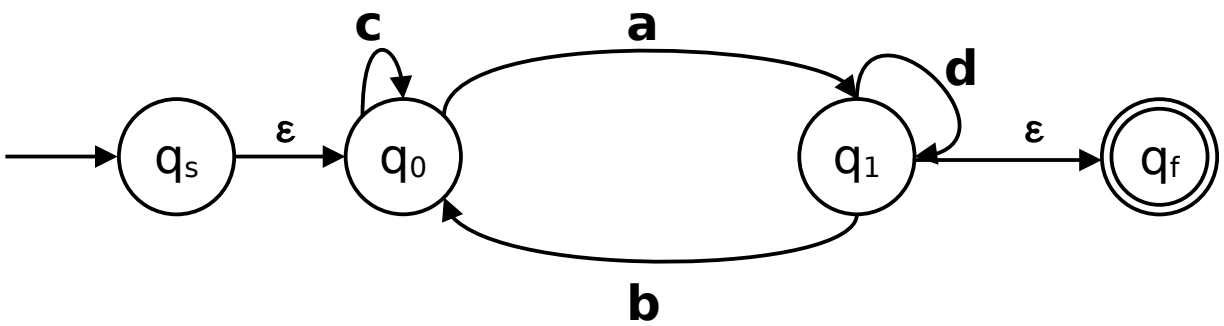
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



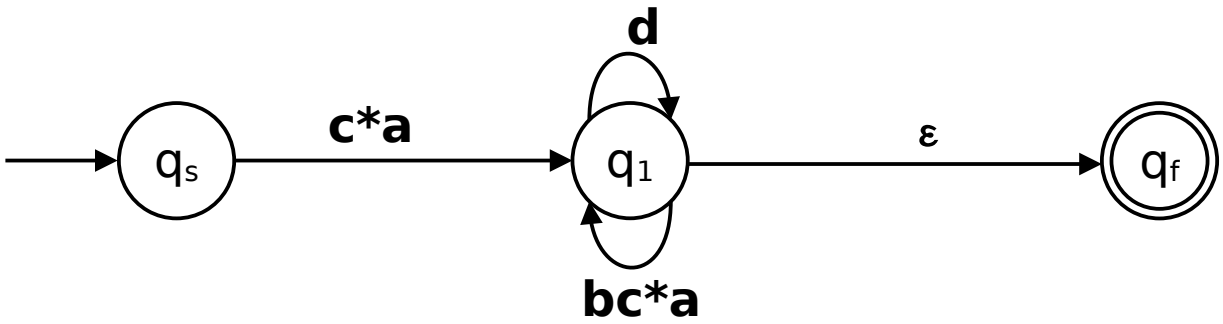
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



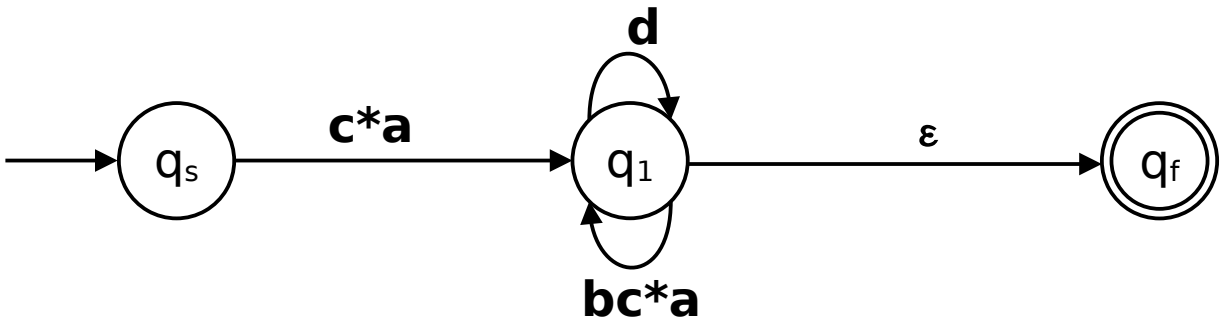
Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



Regular Expressions: DFA to Regex

- Convert the following DFA into regular expressions
- We want to reserve the same language.



c*a(d ∪ bc*a)*

Regular Expressions: GNFA

Read example **1.68**

Homework

- **Reading**
  - **1.4**