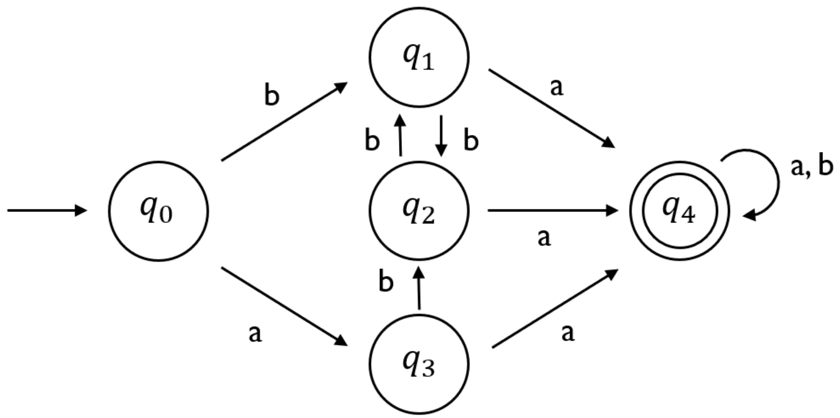
Please check your attendance using Blackboard!

(Revisit) Reduction of the number of states

- **Practice**

	0	1	2	3	4
0	-				
1	?	-			
2	?	?	-		
3	?	?	?	-	
4	X	X	X	X	-



Lecture 3

Regular Languages and Regular Grammars

COSE215: Theory of Computation

Seunghoon Woo

Fall 2023

Contents

- **Regular Languages**
- **Regular Expressions**
- **Regular Grammars**

Regular Expressions and Regular Grammars

- **It is difficult to understand natural languages in automata**
- **Formal language can be understandable!**
  - A formal language is an artificial language that generalizes/abstracts the characteristics of language and formalizes it mathematically
  - E.g., Regular language, context-free language, …
- **A language is regular if there exists a finite automaton for it**

Regular Expressions and Regular Grammars

- **We need more concise ways of describing regular languages**
  1. Regular expressions
  2. Regular grammars

Regular Expressions

- **A compact notation to describe finite-automaton patterns**
- **Definition**
  1. $\emptyset, \lambda,$ and $a \in \Sigma$ are all regular expressions
     - ❖ Called **primitive regular expressions**

Regular Expressions

- **A compact notation to describe finite-automaton patterns**
- **Definition**
  1. $\emptyset, \lambda,$ and $a \in \Sigma$ are all regular expressions
     - Called **primitive regular expressions**
  2. If r_1 and r_2 are regular expressions, so are $r_1 + r_2, \; r_1 \cdot r_2, \; r_1^*,$ and (r_1)

Regular Expressions

- **A compact notation to describe finite-automaton patterns**
- **Definition**

1. $\emptyset, \lambda,$ and $a \in \Sigma$ are all regular expressions
   - Called **primitive regular expressions**
2. If r_1 and r_2 are regular expressions, so are $r_1 + r_2$, $r_1 \cdot r_2$, r_1^*, and (r_1)
3. A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rule 2

Regular Expressions

- **A compact notation to describe finite-automaton patterns**
- **Definition**
  1. $\emptyset, \lambda$, and $a \in \Sigma$ are all regular expressions
     - Called **primitive regular expressions**
  2. If r_1 and r_2 are regular expressions, so are $r_1 + r_2$, $r_1 \cdot r_2$, r_1^*, and (r_1)
  3. A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rule 2
- **Example**
  - For $\Sigma = \{a, b, c\}$, the string $(a + b \cdot c)^* \cdot (c + \emptyset)$ is a regular expression

Regular Expressions

- **Many important applications in computer science**
  - Security and data verification
  - Language processing
  - Text processing and search
  - Data extraction and conversion
  - …

```python
import re
text = "Error 1122: Reference Error\n Error 1023: Argument Error."
regex = re.compile("Error\s\d+")
res = regex.findall(text)
print (res)
```

```
['Error 1122', 'Error 1023']
```

Regular Expressions

- **A regular expression can describe a language**
- **Definition**
  - The language $L(r)$ denoted by any regular expression r is defined:
    1. $\emptyset$ is a regular expression denoting the **empty set**
    2. λ is a regular expression denoting **{$\boldsymbol{\lambda}$}** ($L(\lambda) = \{\lambda\}$)
    3. For every $a \in \Sigma$, a is a regular expression denoting **{$\boldsymbol{a}$}** ($L(a) = \{a\}$)

Regular Expressions

- **A regular expression can describe a language**
- **Definition**
  - The language $L(r)$ denoted by any regular expression r is defined:

1. $\emptyset$ is a regular expression denoting the **empty set**
2. λ is a regular expression denoting **{λ}** ($L(\lambda) = \{\lambda\}$)
3. For every $a \in \Sigma$, a is a regular expression denoting **{a}** ($L(a) = \{a\}$)

If r_1 and r_2 are regular expressions, then

4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$ .... UNION
5. $L(r_1 \cdot r_2) = L(r_1)L(r_2)$ .... CONCATENATION
6. $L((r_1)) = L(r_1)$
7. $L(r_1^*) = \left(L(r_1)\right)^*$ .... STAR

(the last four rules are used to reduce $L(r)$ to simpler components recursively)

Regular Expressions

- **Example**
  - Exhibit the language $L(a^* \cdot (a+b))$ in set notation
    - $L(a^* \cdot (a+b))$
      $= L(a^*)L(a+b)$
      $= \left(L(a)\right)^* L(a) \cup L(b)$
      $= \{\lambda, a, aa, aaa, \dots\}\{a, b\}$
      $= \{a, aa, aaa, \dots, b, ab, aab, \dots\}$

Regular Expressions

- **Precedence and associativity rules**
  - E.g., $L(a \cdot b + c)$: which one is correct?
    - ❖ $L\ (a \cdot b) \cup L(c) = \{ab, c\}$
    - ❖ $L\ (a)\ \big(L(b) \cup L(c)\big) = \{ab, ac\}$
- **Order of precedence**
  - Star > concatenation > union
  - $01^* = 0(1^*)$
- **Left associativity of union and concatenation**
  - $0 \cdot 1 \cdot 0 = (0 \cdot 1) \cdot 0$

Regular Expressions

- **Regular expression can denote a language**
  - E.g., For $\Sigma = \{a, b\}$, the expression $r = (a + b)^*(a + bb)$ is regular and denotes

Regular Expressions

- **Regular expression can denote a language**
  - E.g., For $\Sigma = \{a, b\}$, the expression $r = (a + b)^*(a + bb)$ is regular and denotes

$$L(r) = \{a, bb, aa, abb, ba, bbb, \dots\}$$

❖ **All strings terminated by either an a or a bb**

Regular Expressions

- **Regular expression can denote a language**
  - E.g., For $\Sigma = \{0, 1\}$, give a regular expression r such that

$$L(r) = \{w \in \Sigma^*: w \text{ has at least one pair of consecutive zeros}\}$$

Regular Expressions

- **Regular expression can denote a language**
  - E.g., For $\Sigma = \{0, 1\}$, give a regular expression r such that

$$L(r) = \{w \in \Sigma^*: w \text{ has at least one pair of consecutive zeros}\}$$

❖ Every string in L(r) must contain '00' somewhere

- $r = (0+1)^*00(0+1)^*$

Regular Expressions

- **Practice**
  - E.g., For $\Sigma = \{0, 1\}$, give a regular expression r such that

$$L(r) = \{w \in \Sigma^*: w \text{ contains at least two 0's}\}$$

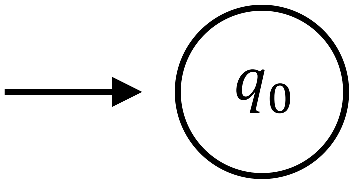
$$L(r) = \{w \in \Sigma^*: w \text{ contains an even number of 0's}\}$$

Regular Expressions and regular languages

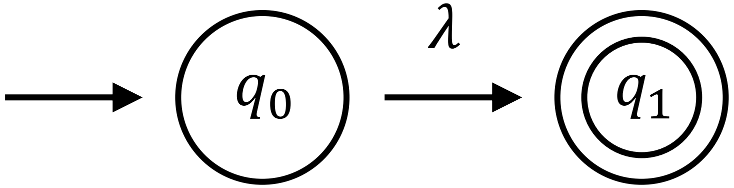
- **If r is a regular expression, then $L(r)$ is a regular language**
  - A language is regular if it is accepted by a DFA
  - We can construct an NFA that accepts $L(r)$ for any regular expression r
    - NFA => DFA (equivalence)

Regular Expressions and regular languages

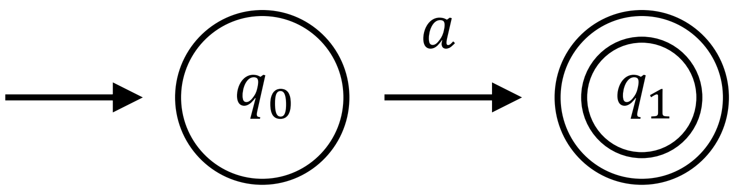
- **If r is a regular expression, then $L(r)$ is a regular language**
  1. Begin with automata that accept the languages for the simple REs



NFA accepts $\emptyset$



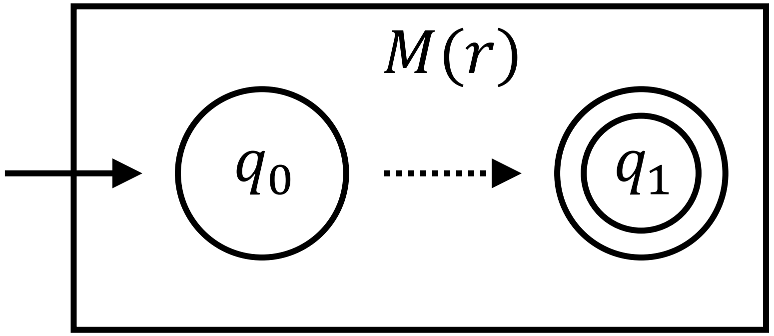
NFA accepts λ



NFA accepts a ($a \in \Sigma$)

Regular Expressions and regular languages

- **If r is a regular expression, then $L(r)$ is a regular language**
    2. Suppose automata $M(r_1)$ and $M(r_2)$ accept languages denoted by r_1 and r_2

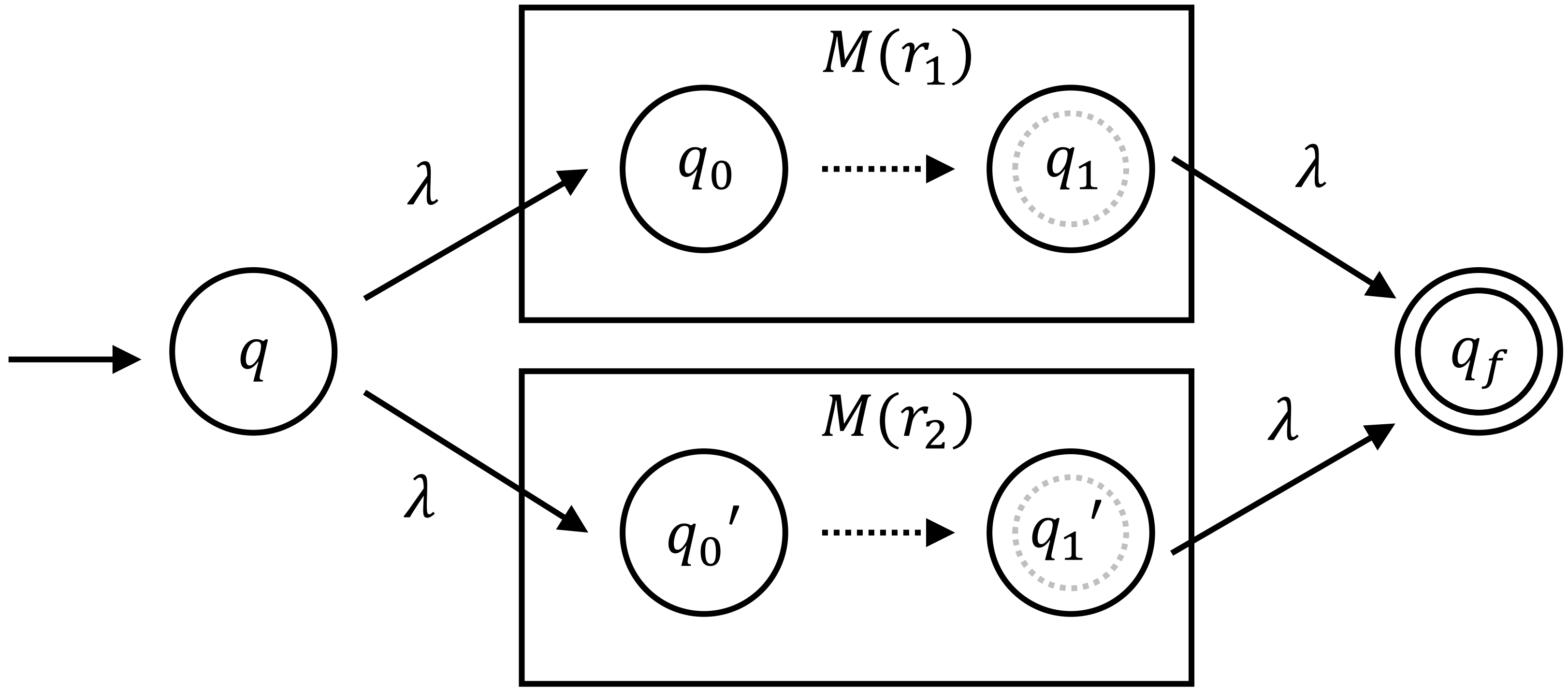


Regular Expressions and regular languages

- **If r is a regular expression, then $L(r)$ is a regular language**

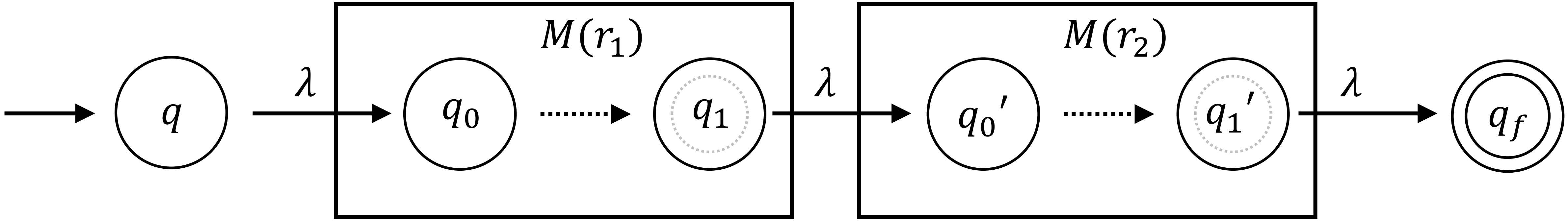
3. Then we can construct automata for the REs $r_1 + r_2, r_1 \cdot r_2$, and r_1^*

❖ $r_1 + r_2$



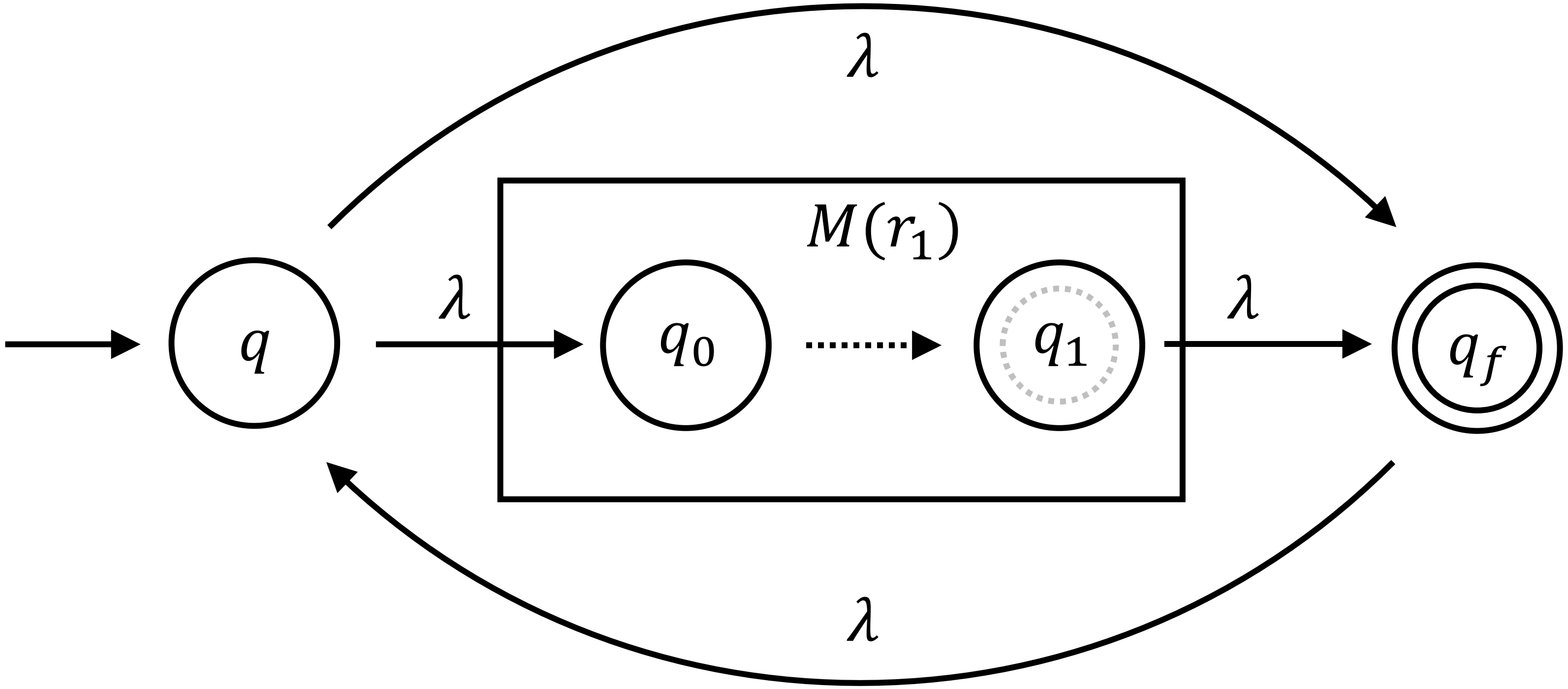
Regular Expressions and regular languages

- **If r is a regular expression, then $L(r)$ is a regular language**
  3. Then we can construct automata for the REs $r_1 + r_2, r_1 \cdot r_2$, and r_1^*
     - ❖ $r_1 \cdot r_2$



Regular Expressions and regular languages

- **If r is a regular expression, then $L(r)$ is a regular language**
  3. Then we can construct automata for the REs $r_1 + r_2, r_1 \cdot r_2$, and r_1^*
     - r_1^*

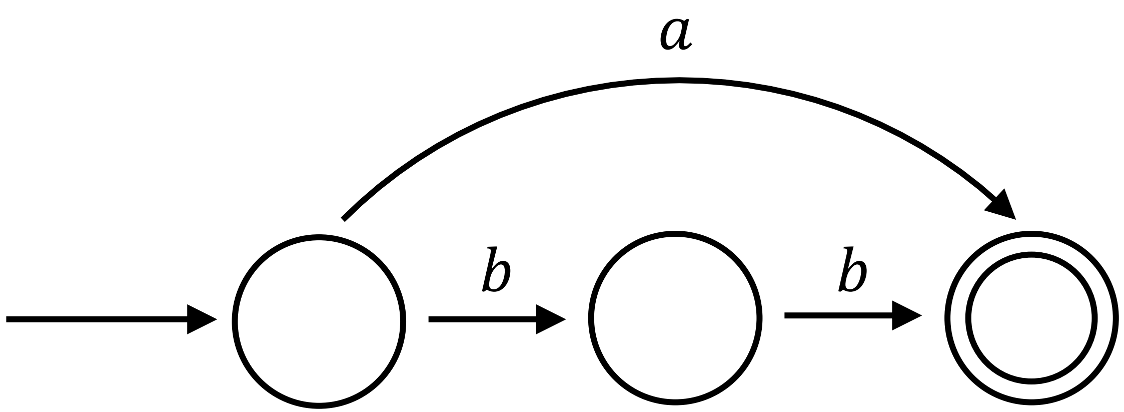


Regular Expressions and regular languages

- **Example**
  - Construct an NFA M that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \lambda)$

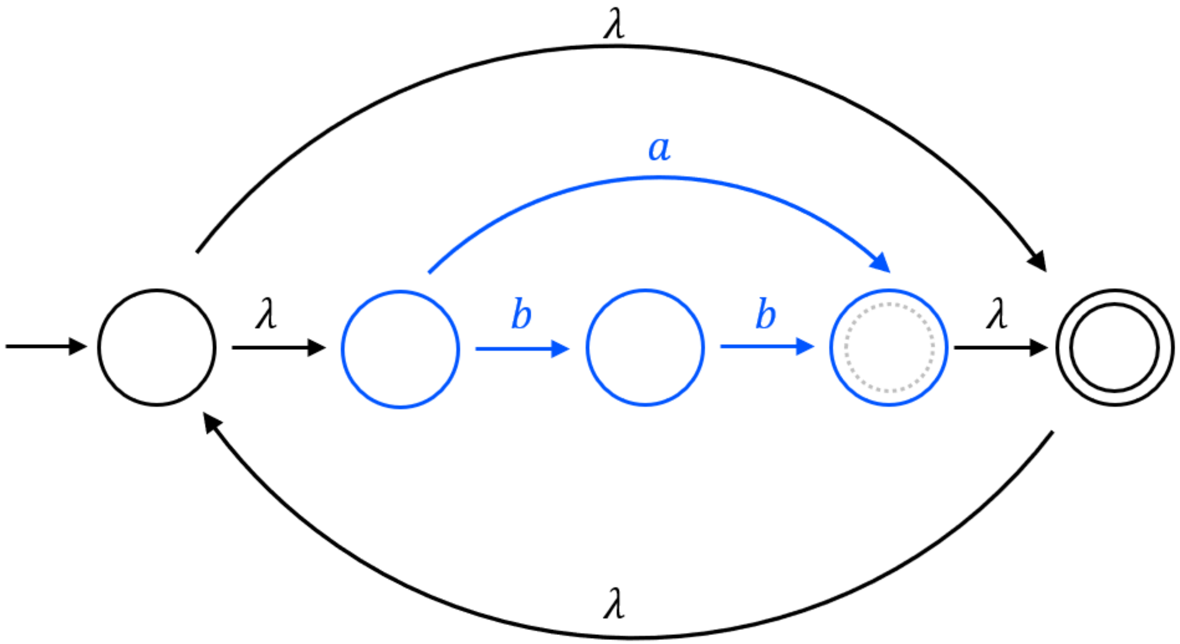
Regular Expressions and regular languages

- **Example**
  - Construct an NFA M that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \lambda)$
    - ❖ M_1 for $(a + bb)$



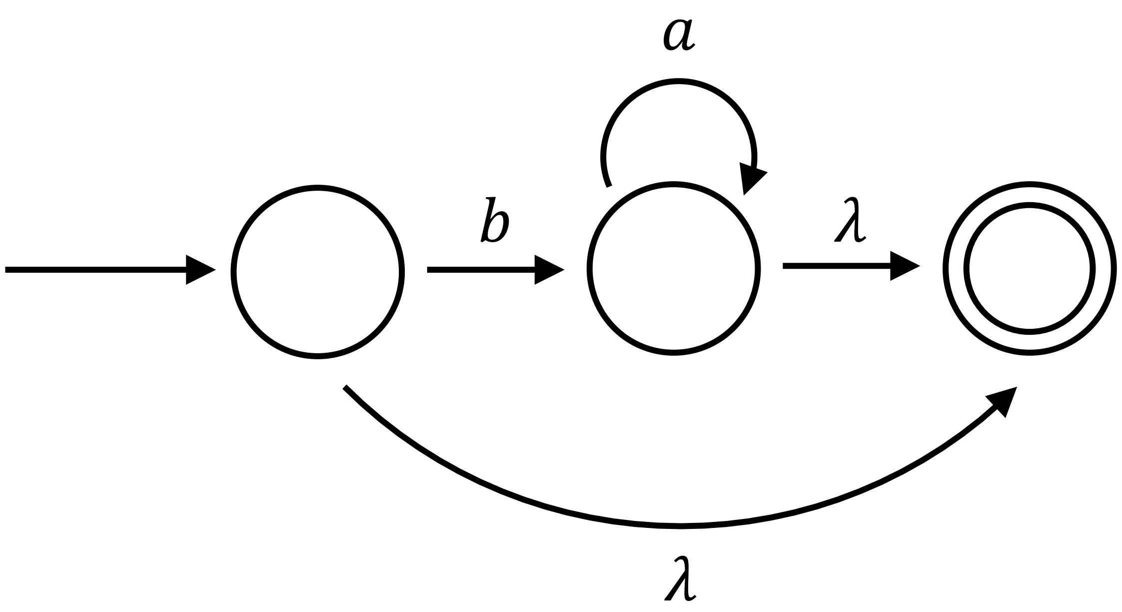
Regular Expressions and regular languages

- **Example**
  - Construct an NFA M that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \lambda)$
    - $(a + bb)^*$



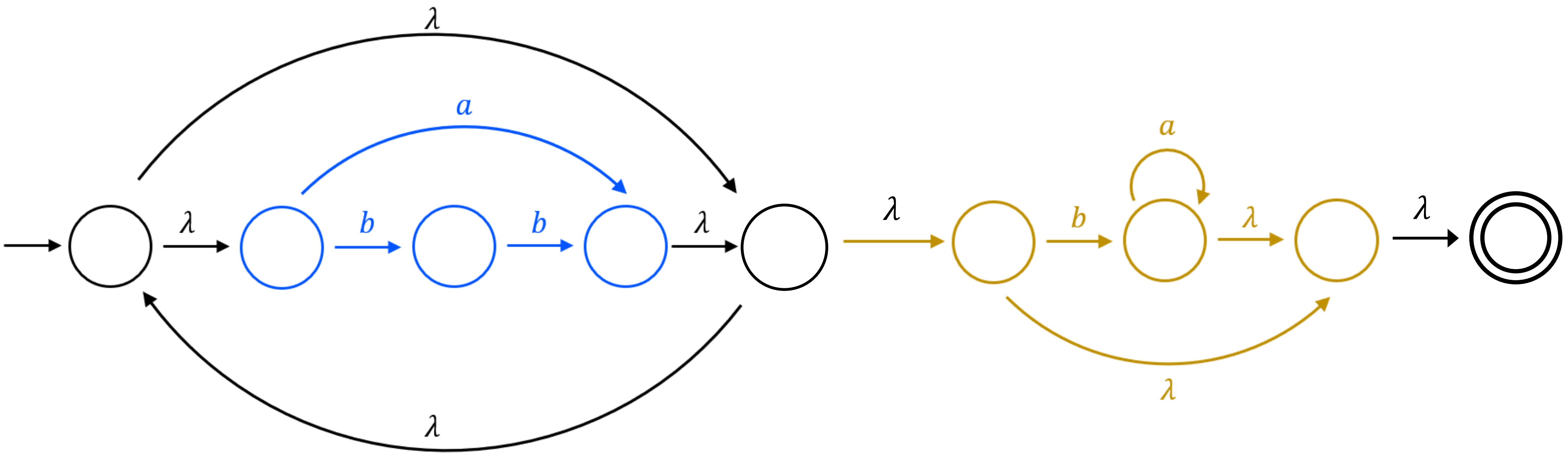
Regular Expressions and regular languages

- **Example**
  - Construct an NFA M that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \lambda)$
    - ❖ M_2 for $(ba^* + \lambda)$



Regular Expressions and regular languages

- **Example**
  - Construct an NFA M that accepts $L(r)$, where $r = (a + bb)^*(ba^* + \lambda)$
    - $L((a + bb)^*(ba^* + \lambda))$



Regular Expressions and regular languages

- **Practice**
  - Construct an NFA M that accepts $(0+1)^*00$

Next Lecture

- **DFA to regular expressions**
- **Regular grammars**