CMSC 141 Automata and Language Theory

Regular Languages

Mark Froilan B. Tandoc

September 3, 2014

Regular Expression Examples

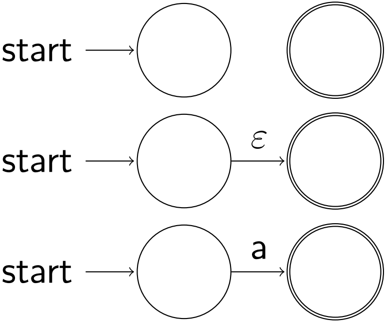
RegEx		**Regular Language**
0*10*	$\rightarrow$	$\{w \mid w$ contains a single 1$\}$
(0+1)*1(0+1)*	$\rightarrow$	$\{w \mid w$ contains at least one 1$\}$
0(0+1)*0+1(0+1)*1 +1+0	$\rightarrow$	$\{w \mid w$ begins and ends with the same symbol$\}$
((0+1)(0+1))*	$\rightarrow$	$\{w \mid w$ has even length$\}$

Regular Expression

- Like DFA and NFA, regular expressions can describe regular languages.
- Therefore, regular expressions can be converted into an equivalent DFA or NFA.

Converting RegEx to εNFA

First, the constants or the basic cases:



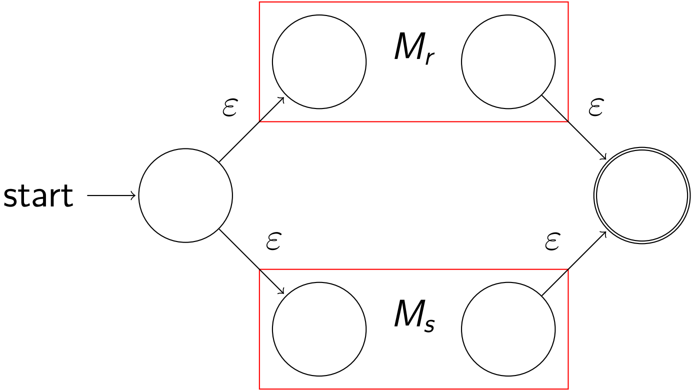
$\emptyset$

ε

a (for any $a \in \Sigma$)

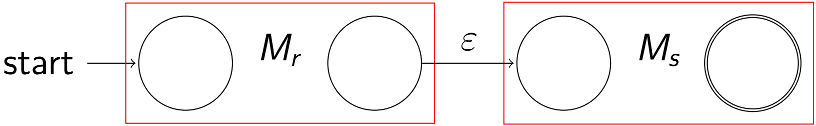
Regular Expression Operations

Let r, s be arbitrary regular expressions with NFAs M_r and M_s. The εNFA for **alternation (r+s)**:



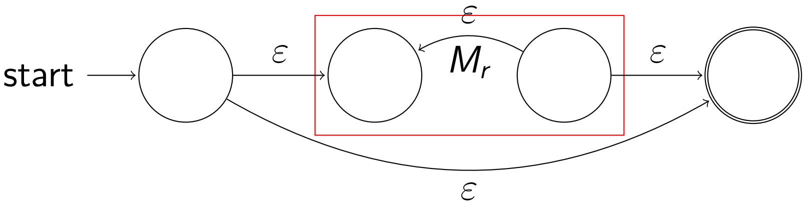
Regular Expression Operations

Let r, s be arbitrary regular expressions with NFAs M_r and M_s. The εNFA for **concatenation (rs)**:



Regular Expression Operations

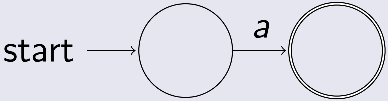
Let r be an arbitrary regular expression with NFA M_r. The εNFA for **Kleene star (r*)**:



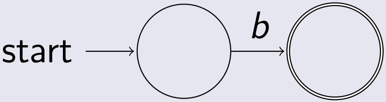
Example conversion of regex to εNFA

(ab+b)*

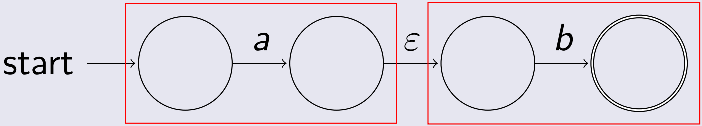
a



b



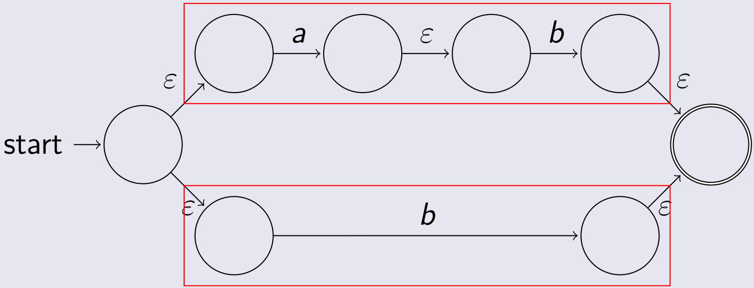
ab



Example conversion of regex to εNFA

(ab+b)*

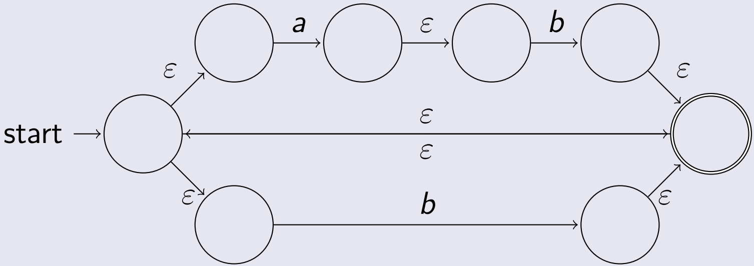
ab+b



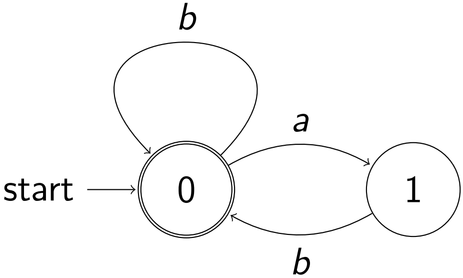
Example conversion of regex to εNFA

(ab+b)*

(ab+b)*



Minimal NFA for (ab+b)*



Describing Regular Languages

Regular Expression (regex)

$\Downarrow$

Nondeterministic Finite Automata with εmoves (εNFA)

$\Downarrow$

Nondeterministic Finite Automata (NFA)

$\Downarrow$

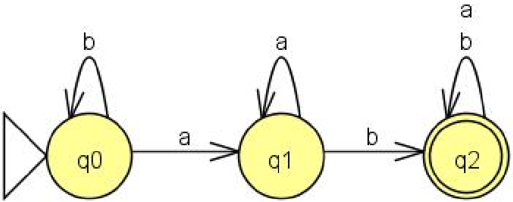
Deterministic Finite Automata (DFA)

Equivalence of all

To show that they are all essentially equivalent, we need to have DFA to regex conversion.

DFA → RegEx Conversion

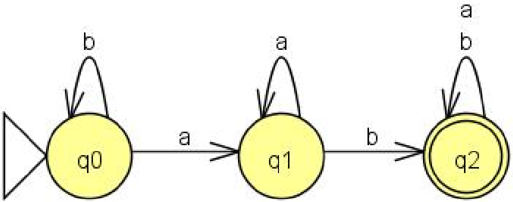
Using the state-elimination method



DFA $\rightarrow$ RegEx Conversion

Using the state-elimination method

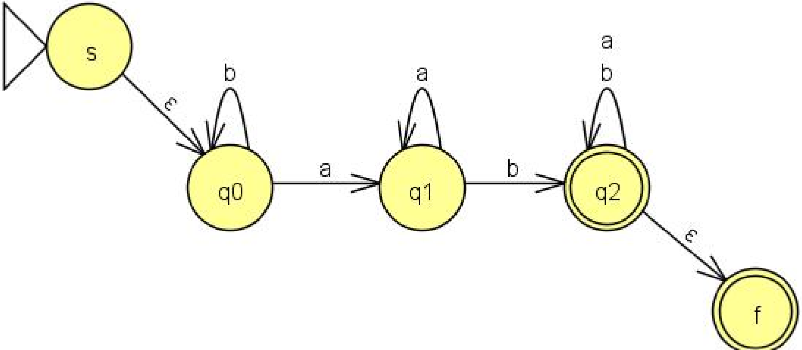
First step is to add new "super-state" (s) and (f), with ε-moves from (s) to the original start state, and from the original final states to (f)



DFA → RegEx Conversion

Using the state-elimination method

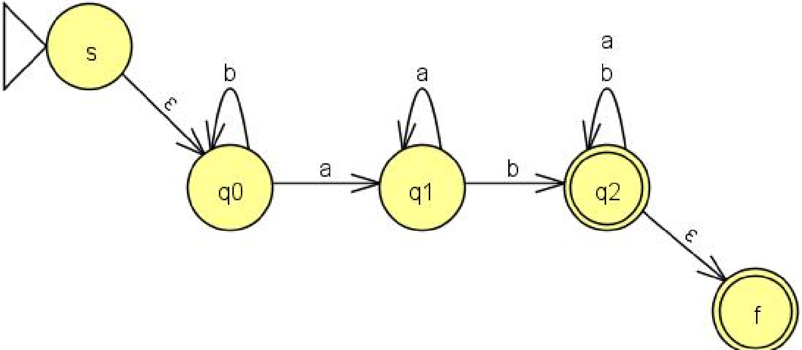
First step is to add new "super-state" (s) and (f), with ε-moves from (s) to the original start state, and from the original final states to (f)



DFA → RegEx Conversion

Using the state-elimination method

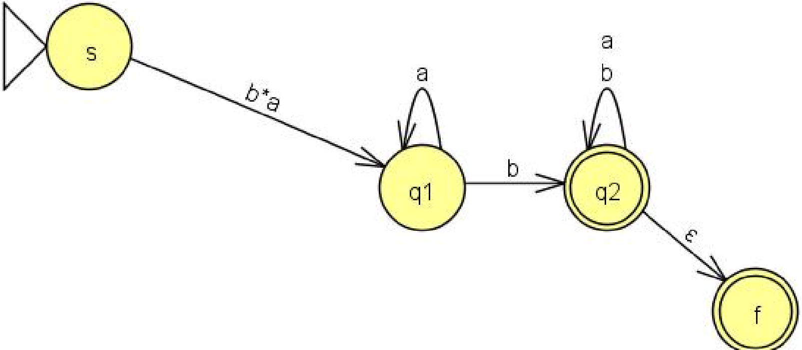
Then eliminate the original states one at a time, replacing transition labels with corresponding regular expressions



DFA → RegEx Conversion

Using the state-elimination method

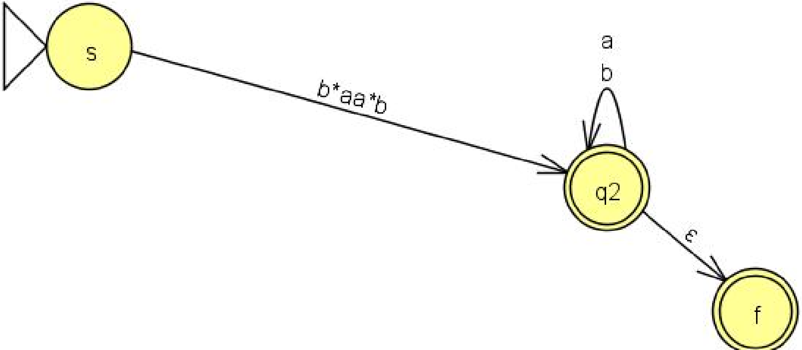
Then eliminate the original states one at a time, replacing transition labels with corresponding regular expressions



DFA → RegEx Conversion

Using the state-elimination method

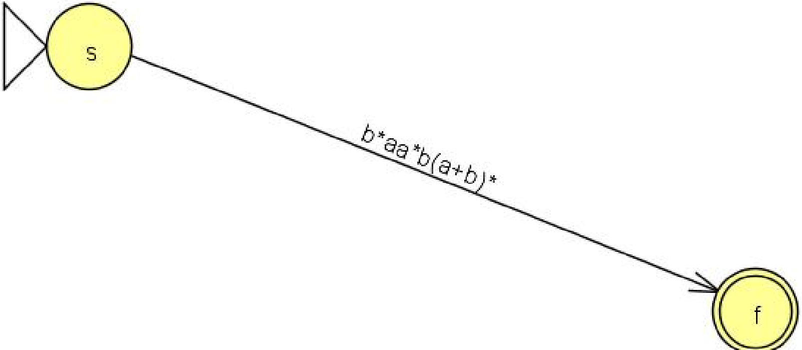
Then eliminate the original states one at a time, replacing transition labels with corresponding regular expressions



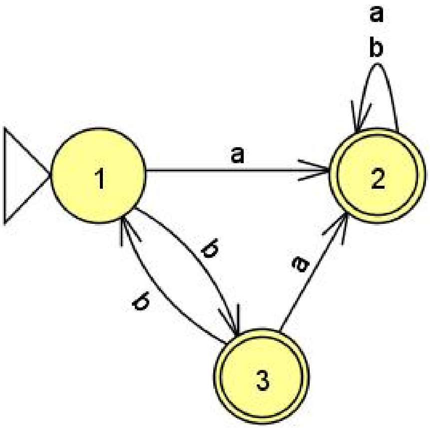
DFA → RegEx Conversion

Using the state-elimination method

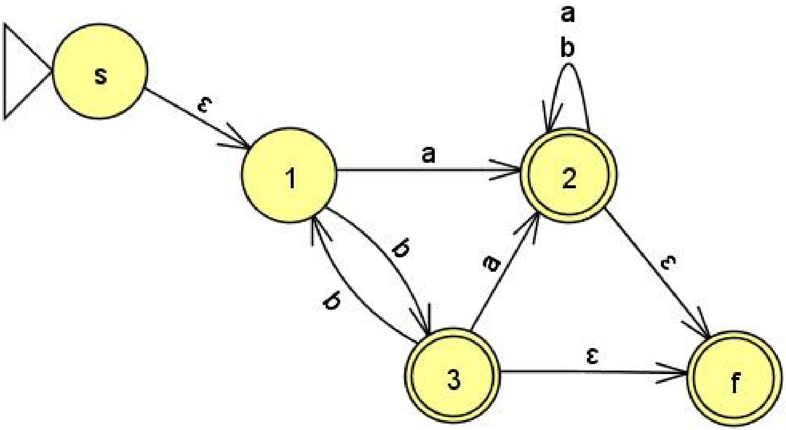
Then eliminate the original states one at a time, replacing transition labels with corresponding regular expressions



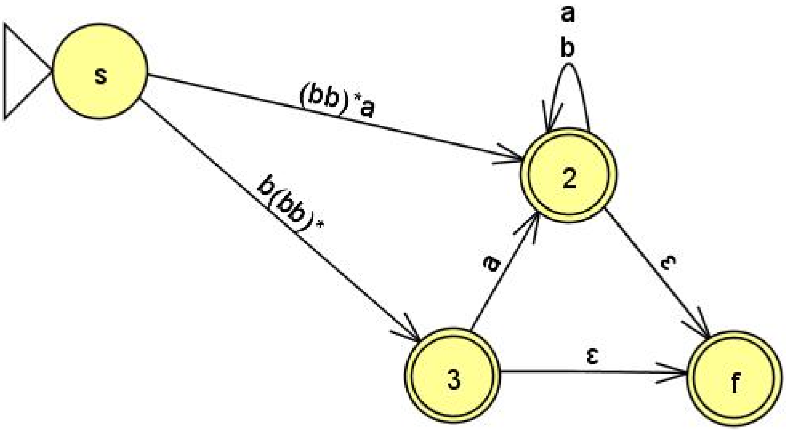
Another example



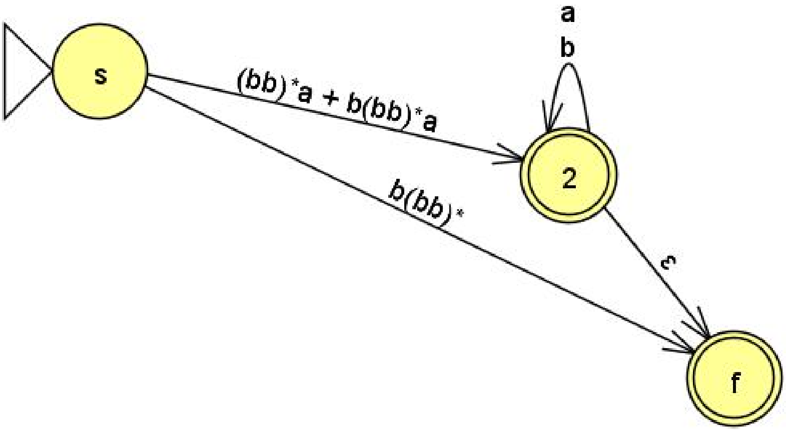
Another example



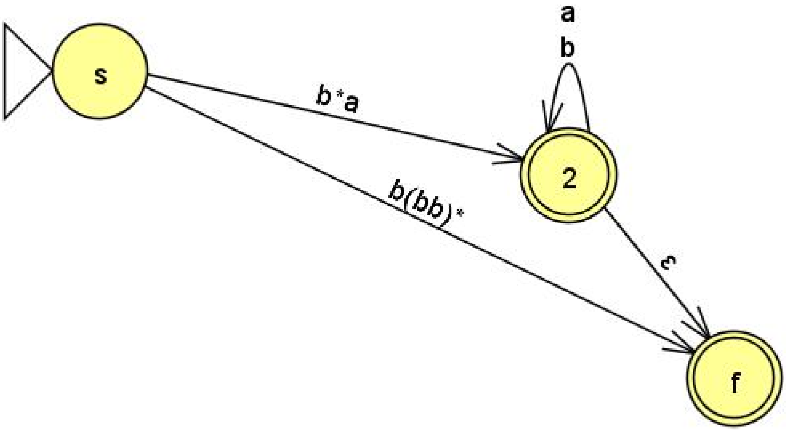
Another example



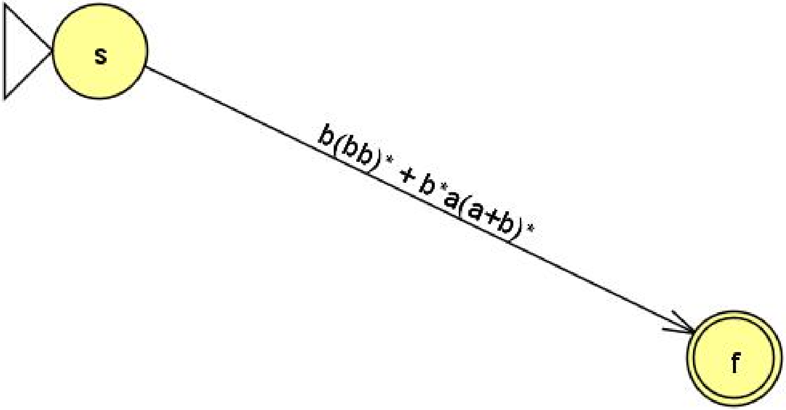
Another example



Another example

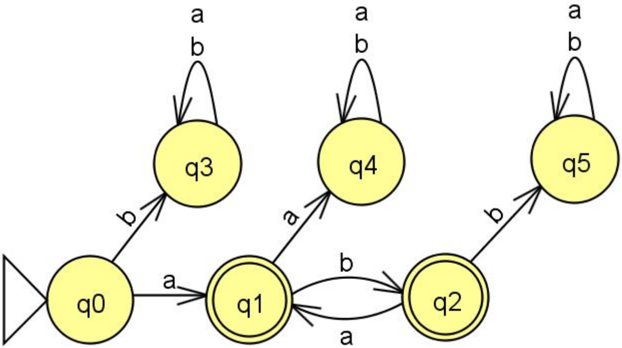


Another example



Exercise

- Does the sequence in which we eliminate states affect the resulting regular expression?
- Try finding a regular expression for the following DFA



References

- Previous slides on CMSC 141
- M. Sipser. Introduction to the Theory of Computation. Thomson, 2007.
- J.E. Hopcroft, R. Motwani and J.D. Ullman. Introduction to Automata Theory, Languages and Computation. 2nd ed, Addison-Wesley, 2001.
- E.A. Albacea. Automata, Formal Languages and Computations, UPLB Foundation, Inc. 2005
- JFLAP, www.jflap.org