CSC 6011: Theory of Computation - Lecture 3

Time: Mon & Wed, 4:00 - 5:20 PM

Location: Teaching A, 107

Instructor: Tao LIN (林涛)

OH: Thursday, 5 – 6 pm, Daoyuan (道远) 420c

Email: lintao@cuhk.edu.cn

New TA: Yefan GAO (高叶繁)

OH: Friday, 10 – 11 am, Zhixin (知新) 410

Email: 226040008@link.cuhk.edu.cn

Survey:

Which date is NOT a good time for you for Midterm Exam (4 pm – 5:20 pm)?

~~(a) Oct 19 (Mon)~~ ~~(b) Oct 21 (Wed)~~

~~(c) Oct 26 (Mon)~~ ~~(d) Oct 28 (Wed)~~

~~(e) Nov 2 (Mon)~~ ~~(f) Nov 4 (Wed)~~

(g) Nov 9 (Mon) (h) Nov 11 (Wed)

Lecture 3

Last time:

- Nondeterminism
- NFA = DFA
- Closure under $\circ$ and $*$
- Regular expressions → finite automata

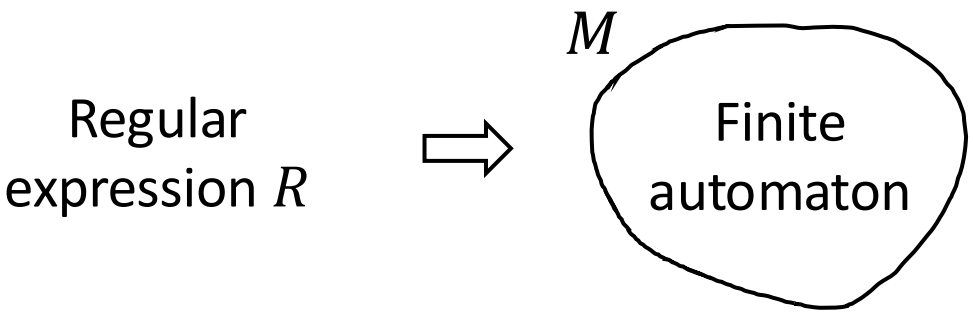
Today:

- Finite automata → regular expressions
- Proving languages aren't regular
  - Pumping lemma
- Context free grammars

DFAs → Regular Expressions

Recall Theorem: If R is a regular expression and $A = L(R)$, then A is regular

Proof: Conversion $R \rightarrow$ NFA $M \rightarrow$ DFA M'



Recall: we did $(\text{a} \cup \text{ab})^*$ as an exercise

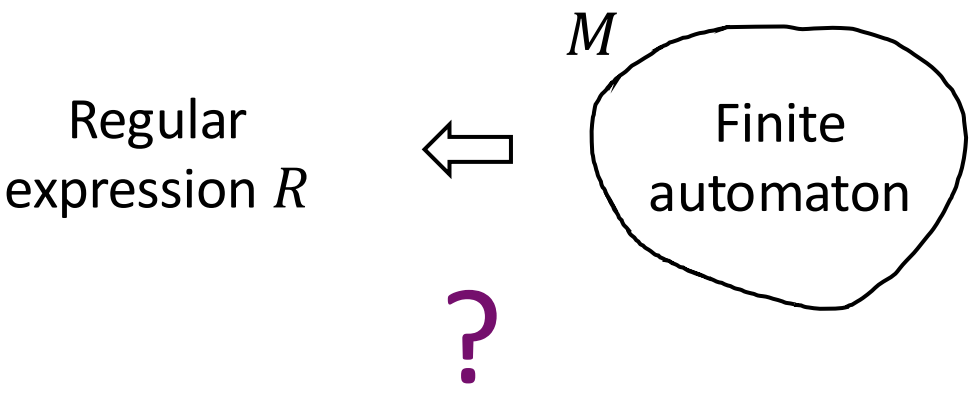
Today's Theorem: If A is regular, then $A = L(R)$ for some regular expression R

Proof: Give conversion DFA $M \rightarrow R$

DFAs → Regular Expressions

Recall Theorem: If R is a regular expression and $A = L(R)$, then A is regular

Proof: Conversion $R \rightarrow$ NFA $M \rightarrow$ DFA M'



Recall: we did $(\text{a} \cup \text{ab})^*$ as an exercise

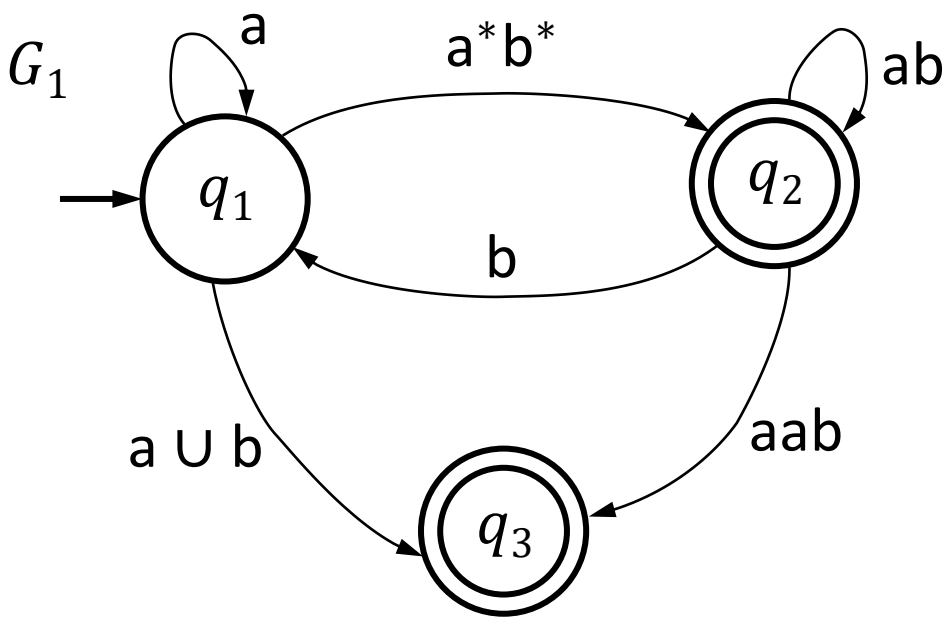
Today's Theorem: If A is regular, then $A = L(R)$ for some regular expression R

Proof: Give conversion DFA $M \rightarrow R$

Will introduce a new concept.

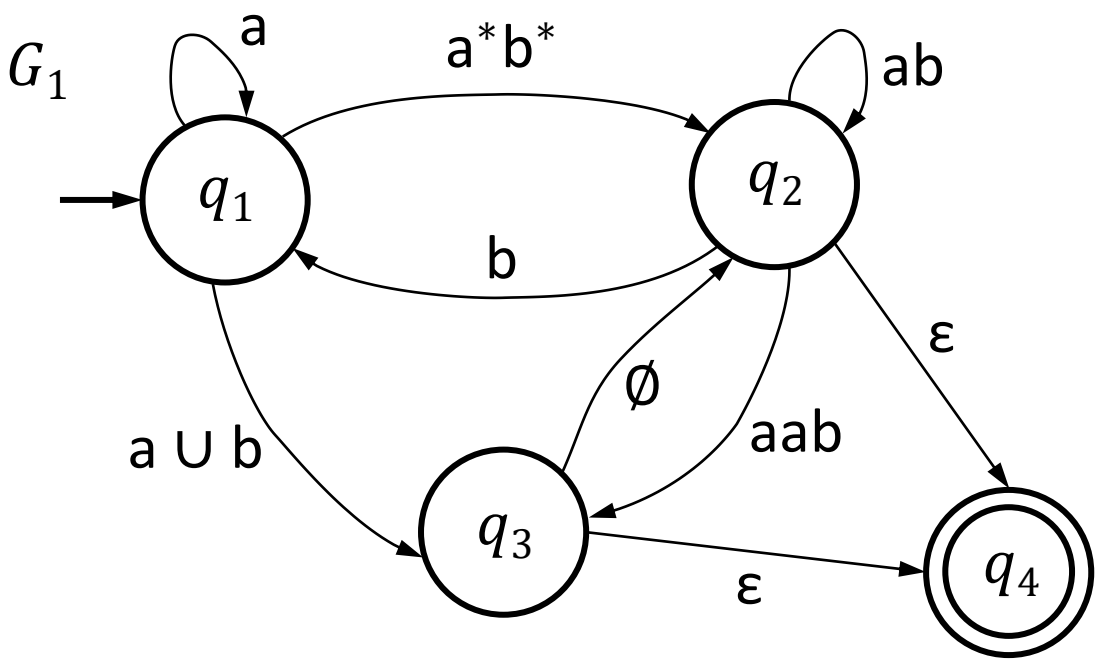
Generalized NFA

Defn: A Generalized Nondeterministic Finite Automaton (GNFA) is similar to an NFA, but allows regular expressions as transition labels



Generalized NFA

Defn: A Generalized Nondeterministic Finite Automaton (GNFA) is similar to an NFA, but allows regular expressions as transition labels



For convenience, we assume:

- One accept state, different from the start state
- One arrow from each state to each state, except
  a) only exiting the start state
  b) only entering the accept state

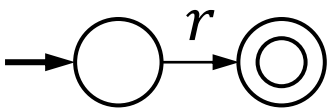
We can easily modify a GNFA to have this special form.

GNFA → Regular Expressions

Lemma: Every GNFA G has an equivalent regular expression R

Proof: By induction on the number of states k of G

Basis $(k = 2)$:

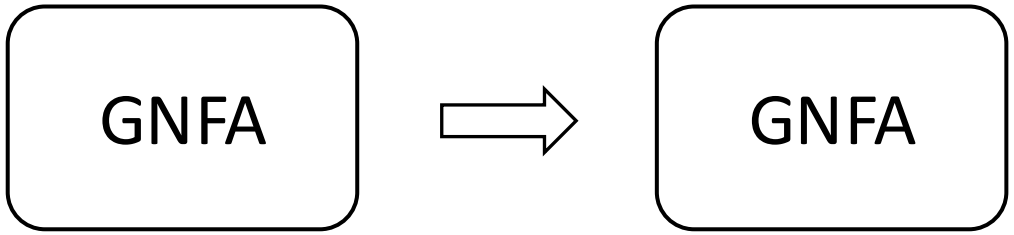
$G =$ 

Remember: G is in special form

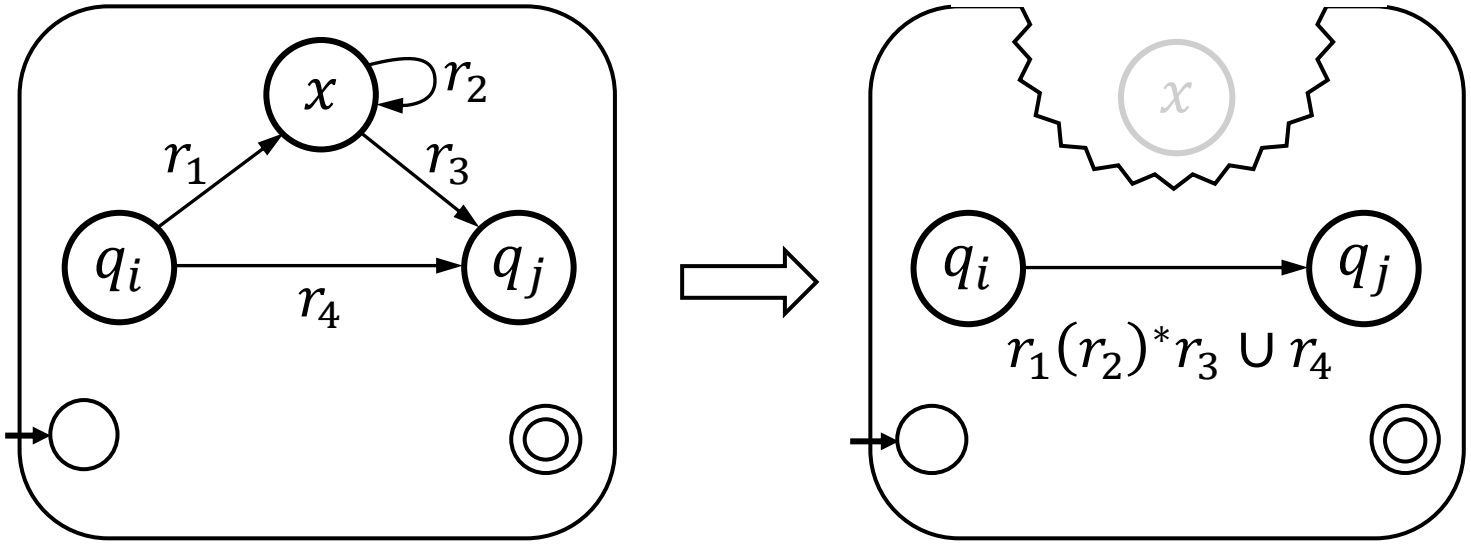
Let $R = r$

Induction step $(k > 2)$: Assume Lemma true for $k - 1$ states and prove for k states

IDEA: Convert k-state GNFA to equivalent $(k - 1)$-state GNFA



k-state GNFA $\rightarrow$ $(k-1)$-state GNFA



k states

$k-1$ states

1. Pick any state x except the start and accept states.
2. Remove x.
3. Repair the damage by recovering all paths that went through x.
4. Make the change for every pair of states q_i, q_j .

By induction, GNFAs can be converted to regular expressions.

k-state GNFA $\rightarrow$ $(k-1)$-state GNFA

Mini-Quiz 3.1

We just showed how to convert GNFAs to regular expressions, but our goal was to convert DFAs to regular expressions. How do we achieve our goal?

(a) Show how to convert DFAs to GNFAs

(b) Show how to convert GNFAs to DFAs

(c) We are already done. DFAs are a type of GNFAs.

1. Pick any state x except the start and accept states.
2. Remove x.
3. Repair the damage by recovering all paths that went through x.
4. Make the change for every pair of states q_i, q_j .

By induction, GNFAs can be converted to regular expressions.

Therefore, DFAs and regular expressions are equivalent (Kleene's Theorem).

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA (or a regular expression).
- To show a language is *not* regular, we must give a proof.
  - It is not enough to say that you couldn't find a DFA for it.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w \text{ has equal numbers of 01 and 10 substrings}\}$ $\quad 0\underline{10}1 \notin C \quad 01\underline{10} \in C$

Intuition: C is not regular because DFAs cannot count unboundedly.

However, C is regular! Sometimes intuition works, but sometimes it is wrong.

We need to give a proof.

Method for Proving Non-Regularity

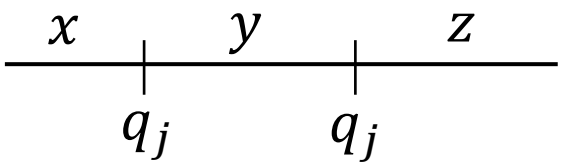
Pumping Lemma: For every regular language A, there is a number p (the "pumping length") such that: if $s \in A$ and $|s| \geq p$, then $s = xyz$ where

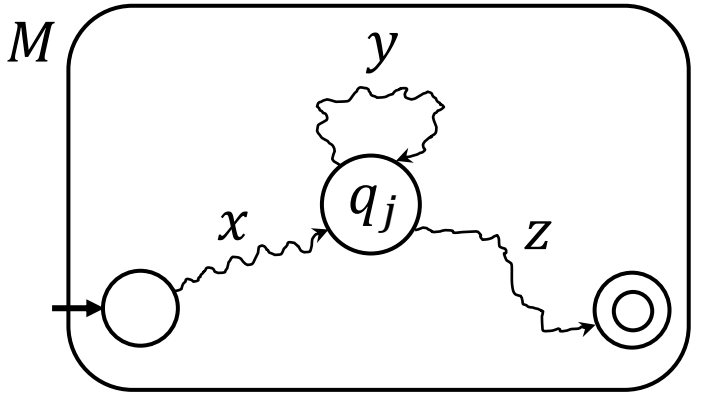
1) $xy^iz \in A$ for all $i \geq 0$
2) $y \neq \varepsilon$ (x and z can be ε)
3) $|xy| \leq p$

$y^i = \underbrace{yy \cdots y}_{i \text{ times}}$

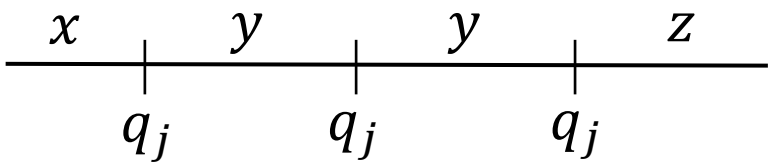
Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ 



When reading s, M will visit some state q_j at least twice, because s is too long. *"Pigeonhole Principle"*

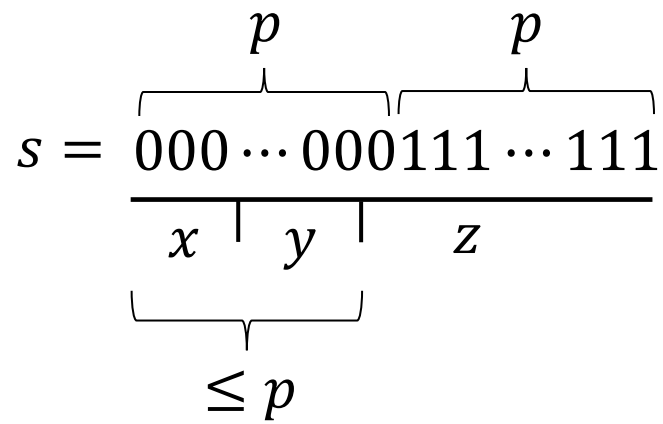


is also accepted

Example 1 of Proving Non-Regularity

Pumping Lemma: For every regular language A, there is a p such that: if $s \in A$ and $|s| \geq p$, then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$ (x and z can be ε)
3) $|xy| \leq p$



Let $D = \{0^k 1^k \mid k \geq 0\}$

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.

The pumping lemma gives p as above. Let $s = 0^p 1^p \in D$.

Pumping lemma says that we can divide $s = xyz$ satisfying the 3 conditions.

But $xyyz$ has more 0s than 1s, and thus $xyyz \notin D$ contradicting the pumping lemma.

Therefore, our assumption (D is regular) is false. We conclude that D is not regular.

Example 2 of Proving Non-Regularity

Pumping Lemma: For every regular language A, there is a p such that: if $s \in A$ and $|s| \geq p$, then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$ (x and z can be ε)
3) $|xy| \leq p$

$$s = \underbrace{000\cdots000}_{x}\underbrace{000\cdots000}_{y}\underbrace{000}_{z} \qquad y = 00$$

$$\leq p$$

$$s = \underbrace{000\cdots001}_{x}\underbrace{000\cdots001}_{y}\underbrace{000}_{z}$$

$$\leq p$$

Let $F = \{ww \mid w \in \Sigma^*\}$. Say $\Sigma^* = \{0,1\}^*$.

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

The pumping lemma gives p as above. Need to choose $s \in F$. Which s?

- Try $s = 0^p0^p \in F$. *This s can be pumped and stay inside F. Bad choice.*

- Try $s = 0^p10^p1 \in F$. We see that $s = xyz$ satisfies the 3 conditions, but $xyyz \notin F$.

Contradiction! Therefore, F is not regular.

Example 3 of Proving Non-Regularity

Another method: Combine closure properties with the Pumping Lemma.

Let $B = \{w | \ w$ has equal numbers of 0s and 1s$\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection).

But $B \cap 0^*1^* = \{0^k 1^k \big| \ k \geq 0\} = D$ and we already showed D is not regular.

Contradiction!

Therefore, our assumption is false, so B is not regular.

Context Free Grammars

G_1

S → 0S1
S → R
R → ε

} (Substitution) Rules

In G_1:

Rule: Variable → string of variables and terminals — 3 rules
Variables: Symbols appearing on left-hand side of rule — R,S
Terminals: Symbols appearing only on right-hand side — 0,1
Start Variable: Top left symbol — S

Grammars generate strings

1. Write down start variable
2. Replace any variable by the RHS according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Context Free Grammars

G_1

$S \rightarrow 0S1$
$S \rightarrow R$
$R \rightarrow \varepsilon$

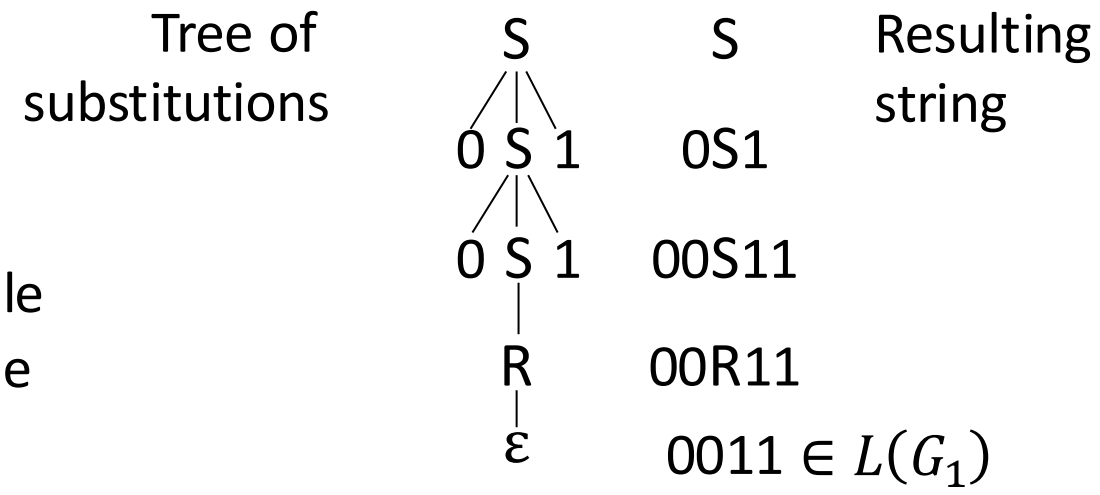
} (Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

Grammars generate strings

1. Write down start variable
2. Replace any variable by the RHS according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string



$$L(G_1) = \{0^k 1^k \mid k \geq 0\}$$

Context Free Grammars

G_1

S → 0S1
S → R
R → ε

} (Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

Grammars generate strings

1. Write down start variable
2. Replace any variable by the RHS according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Exercise 3.2

G_2

S → RR
R → 0R1
R → ε

Check all of the strings that are in $L(G_2)$:

(a) 001101
(b) 000111
(c) 1010
(d) ε

Quick review of today

1. Conversion of DFAs to regular expressions

   Summary: DFAs, NFAs, regular expressions are all equivalent

2. Proving languages not regular by using the pumping lemma and closure properties

3. Context Free Grammars

These slides are adapted from Michael Sipser, *18.404J Theory of Computation*, Fall 2020. Massachusetts Institute of Technology:

- Source: MIT OpenCourseWare https://ocw.mit.edu