# 18.404/6.840 Lecture 3

**Last time:**

- Nondeterminism
- NFA → DFA
- Closure under  and
- Regular expressions → finite automata

**Today:**

- Finite automata → regular expressions
- Proving languages aren’t regular
- Context free grammars

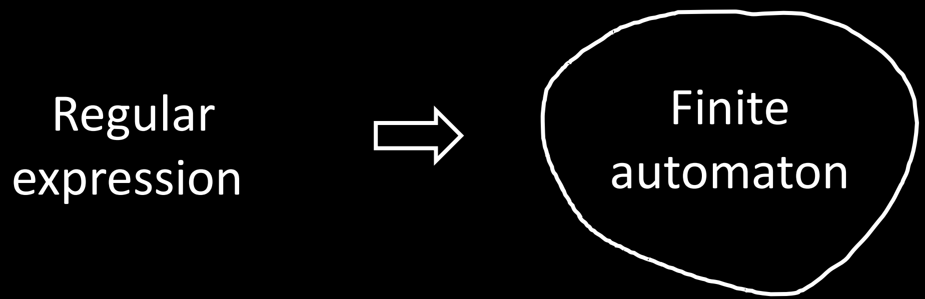
We start counting Check-ins today.
Review your email from Canvas.

Homework due Thursday, posted on homepage.

# DFAs Regular Expressions

**Recall Theorem:** If  is a regular expressipn and  then  is regular

**Proof:** Conversion  NFA  DFA



Recall: we did  as an example

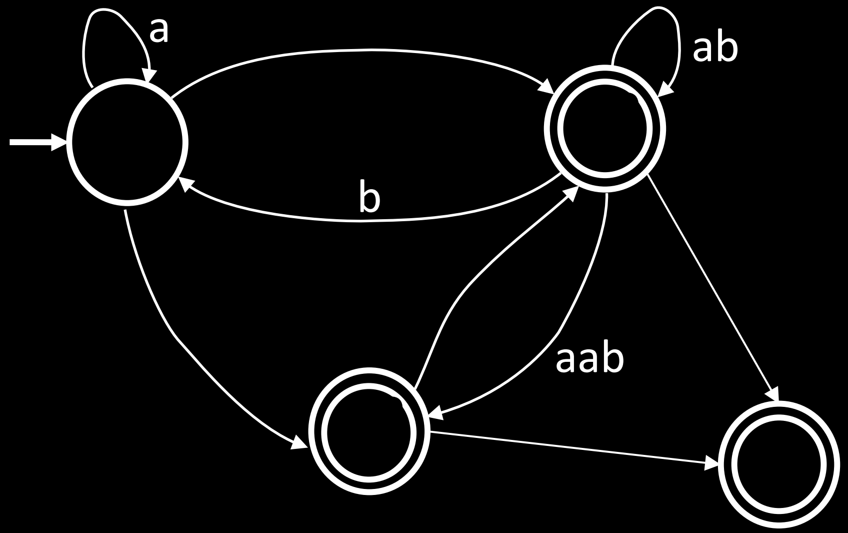
**Today's Theorem:** If  is regular then  for some regular expr

**Proof:** Give conversion  DFA

WAIT! Need new concept first.

# Generalized NFA

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



**For convenience we will assume:**

- One accept state, separate 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

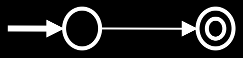
We can easily modify a GNFA to have this special form.

# GNFA Regular Expressions

**Lemma:** Every GNFA has an equivalent regular expression

**Proof:** By induction on the number of states of

*Basis* :

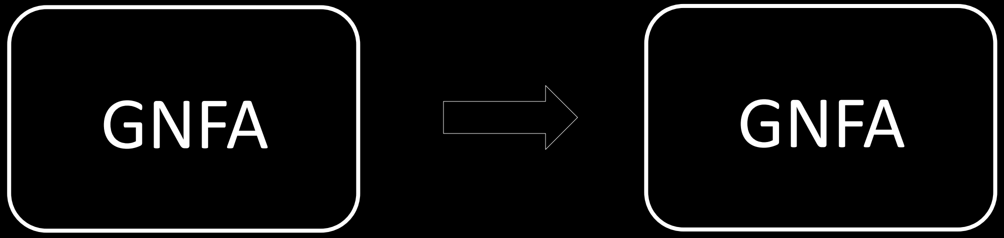
¿ 

Remember: is in special form

Let

*Induction step* : Assume Lemma true for states and prove for states

IDEA: Convert-state GNFA to equivalent -state GNFA



GNFA

GNFA

states

states

# -state GNFA    (—1)-state GNFA

Check-in 3.1

We just showed how to convert GNFAs to regular expressions but our goal was to show that how to convert DFAs to regular expressions. How do we finish 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.

Thus DFAs and regular expressions are equivalent.

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

👍 Check-in 3.1

# Non-Regular Languages

## How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- 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, therefore the language isn't regular.

**Two examples:** Here .

1. Let has equal numbers of 0s and 1s

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

2. Let has equal numbers of 01 and 10 substrings

*Intuition:* is not regular because DFAs cannot count unboundedly.
However is regular! S

## Moral: You need to give a proof.

Coffee
Break

# Method for Proving Non-regularity

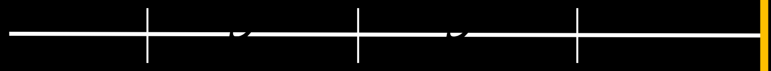
**Pumping Lemma:** For every regular language ,
there is a number (the “pumping length”) such that
if and then where

1) for all
2)
3)

Informally: is regular → every long str

**Proof:** Let DFA recognize . Let be th

will repeat a state when reading
because is so long.



is als

## Check-in 3.2

The Pumping Lemma depends on the fact that
if has states and it runs for more than steps then
will enter some state at least twice.
We call that fact:

(a) The Pigeonhole Principle
(b) Burnside's Counting Theorem
(c) The Coronavirus Calculation

Check-in 3.2

# Example 1 of Proving Non-regularity

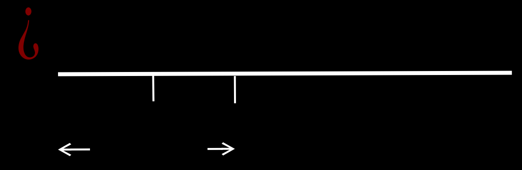
**Pumping Lemma:** For every regular language , there is a such that if and then where

1) for all
2)
3)

## Let

**Show:** is not regular

**Proof by Contradiction:**
Assume (to get a contradiction) that is regular.
The pumping lemma gives as above. Let .
Pumping lemma says that can divide satisfying the 3 conditions.



But has excess 0s and thus contradicting the pumping lemma.
Therefore our assumption ( is regular) is false. We conclude that is not regular.

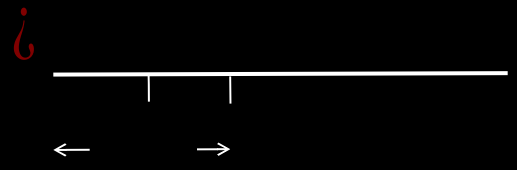
# Example 2 of Proving Non-regularity

**Pumping Lemma:** For every regular language , there is a such that if and then where

1) for all
2)
3)

Let Say .

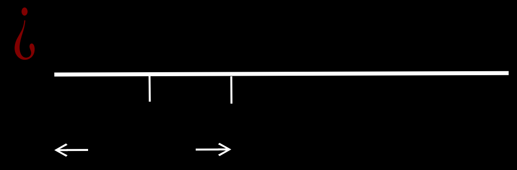
**Show:** is not regular



**Proof by Contradiction:**

Assume (for contradiction) that is regular.

The pumping lemma gives as above. Need to choose . Which ?



Try . But that can k

Try . Show cannot be pumped satisfying the 3 conditions.

Contradiction! Therefore is not regular.

# Example 3 of Proving Non-regularity

**Variant:** Combine closure properties with the Pumping Lemma.

Let has equal numbers of 0s and 1s

**Show:** is not regular

**Proof by Contradiction:**

Assume (for contradiction) that is regular.

We know that is regular so is regular (closure under intersection).

But and we already showed is not regular. Contradiction!

Therefore our assumption is false, so is not regular.

# Context Free Grammars

S 0S1<br>
S R<br>
R

(Substitution) Rules

**Rule:** Variable string of variables and terminals<br>
**Variables:** Symbols appearing on left-hand side of rule<br>
**Terminals:** Symbols appearing only on right-hand s<br>
**Start Variable:** Top left symbol

## Grammars generate strings

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

Check-in 3.3

S RR<br>
R 0R1<br>
R

Check all of the strings that are in

(a) 001101<br>
(b) 000111<br>
(c) 1010<br>
(d)

Check-in 3.3

# 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