18.404/6.840 Lecture 3

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

Today: (Sipser §1.4 – §2.1)
- Finite automata → regular expressions
- Proving languages aren't regular
- Context free grammars

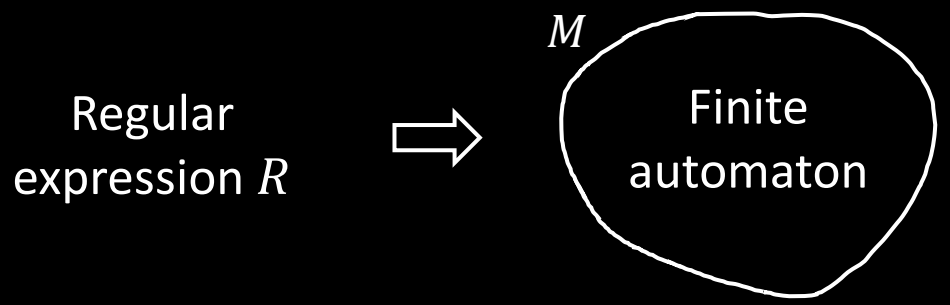
We start counting Check-ins today.
Review your email from Canvas.

Homework due Thursday.

DFAs → Regular Expressions

Recall Theorem: If R is a regular expressipn 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 example

It is the language of some finite automaton

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

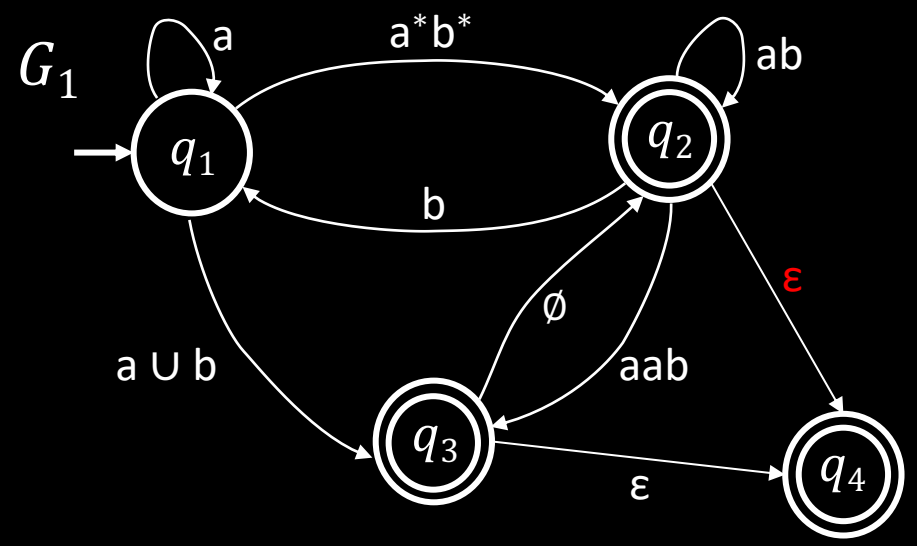
Proof: Give conversion DFA $M \rightarrow R$

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

Read a hole string one step and check the chunk of that read whether in the regular expression



We could easily modify the machine to achieve that

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 G has an equivalent regular expression R

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

a recursive proof that calls itself

Basis $(k = 2)$:

$G =$ 

Remember: G is in special form

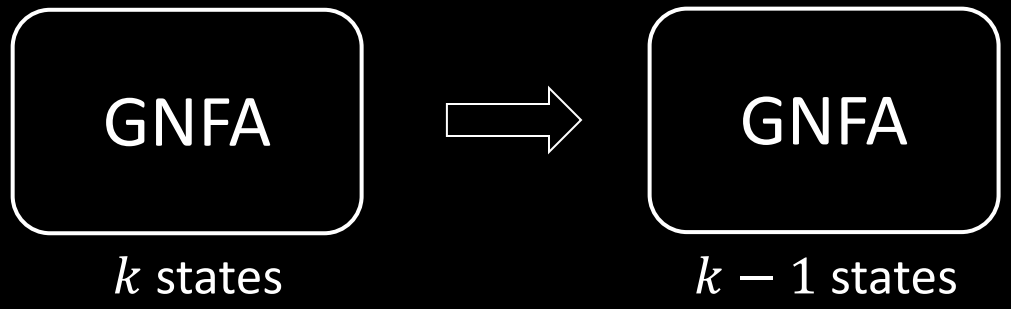
By definition: When k = 2, G must like this

Let $R = r$

The whole name of the game here is figuring out how to convert a GNFA that has k states into another one that has k-1 states that does the same language

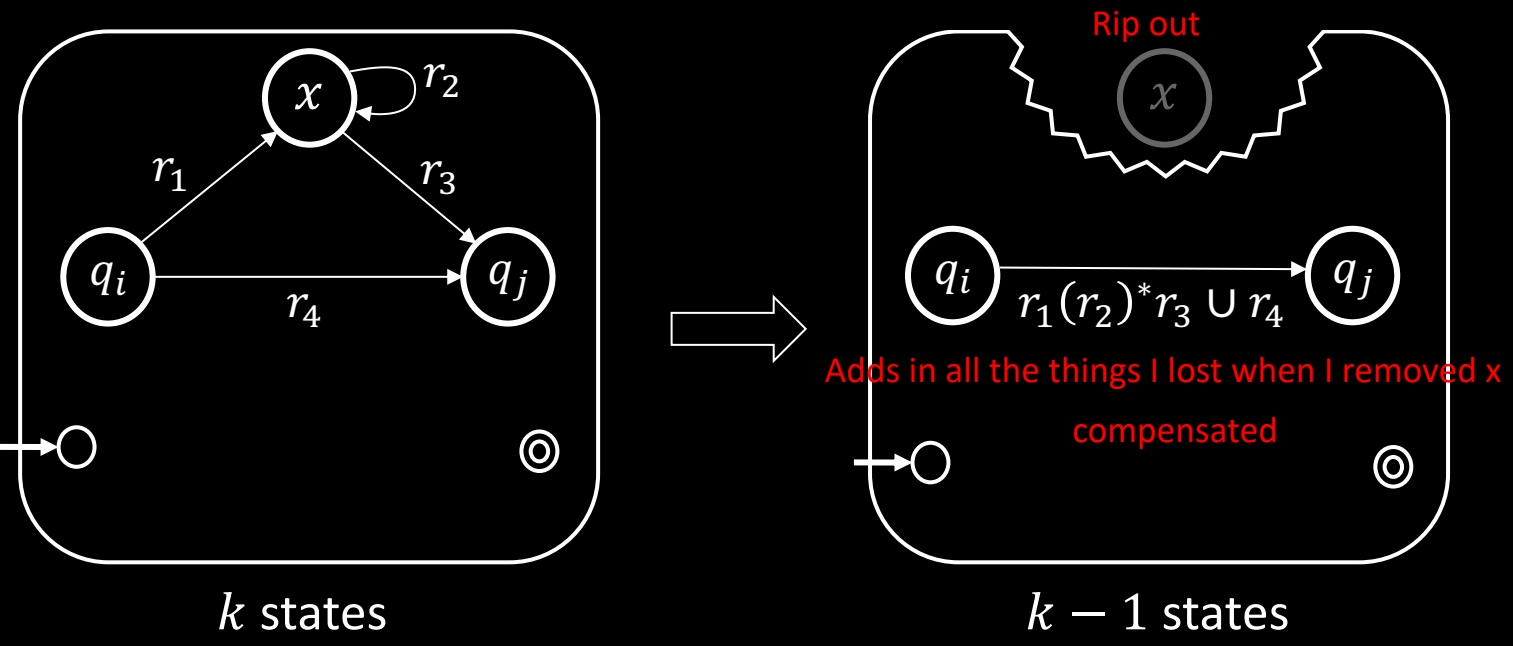
Induction step $(k > 2)$: Assume Lemma true for $k - 1$ states and prove for k states

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



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

Look at all the path that could go through x and make sure that they are still present even though I don't have x anymore



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 indicated change for each pair of states q_i, q_j .

Thus DFAs and regular expressions are equivalent.

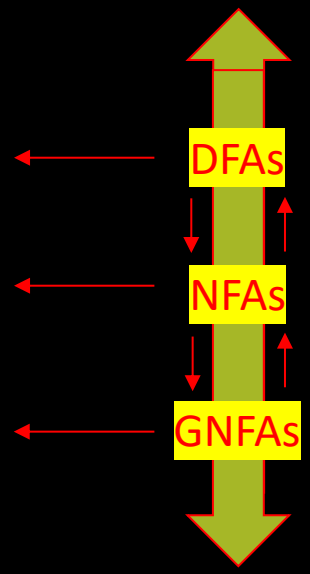
Conclusion

Regular Languages

Definition:
A language is regular if some deterministic finite automaton M recognizes it.

Proof by construction
If an NFA recognizes A then A is regular

Proof by introduction
K state GNFA -> (k – 1)state GNFA



Closure Properties

If Re is a regular expr and A = L(Re) then A is regular Language

Regular Expressions

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.

All DFAs are automatically GNFAs

We've already shown that regular languages can either come from regular expressions or DFAs

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.

DFAs are weak as a computation model

There is all sorts of things that they cannot do
Though there are some complicated things
that they can do, surprisingly enough

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

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

Intuition: B is not regular because DFAs cannot count unboundedly

2. Let $C = \{w | \ w$ has equal numbers of 01 and 10 substrings$\}$

Set of all strings that end and start with the same character (0 or 1)

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

Moral: You need to give a proof.

Method for Proving Non-regularity

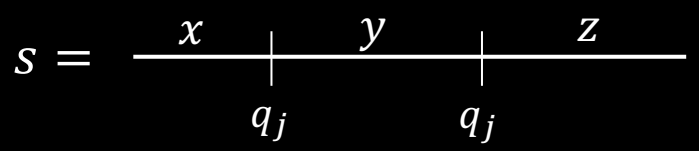
Pumped means I can cut the string into three pieces and repeat that middle piece as many times as I want

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

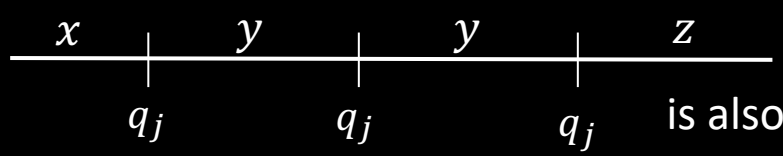
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

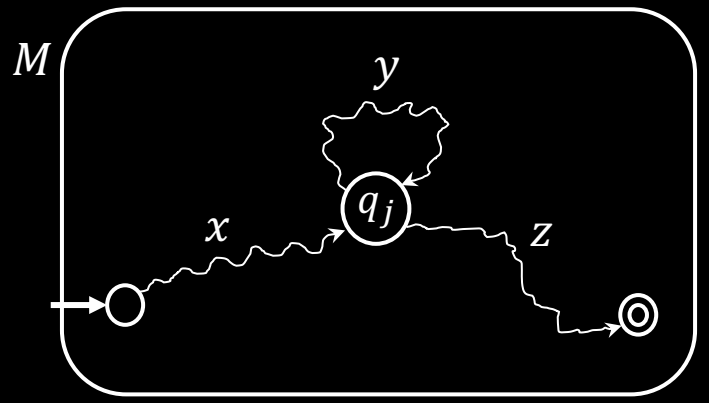
Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.



M will repeat a state q_j when reading s because s is so long.



is also accepted



The path that M follows when reading s.

Check-in 3.2

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

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

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 $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Let $D = \{0^k1^k \mid k \geq 0\}$

Show: D is not regular

Just need to find one incorrect p

Proof by Contradiction:

We do not know how large is k

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

The pumping lemma gives p as above. Let $s = 0^p1^p \in D$.

There will be lots of zeros followed by not so many ones

$$s = \underbrace{000\cdots000111\cdots111}_{x \mid y \mid z}$$

$\leftarrow \leq p \rightarrow$

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

But $xyyz$ has excess 0s 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) $y \neq \varepsilon$
3) $|xy| \leq p$

Let $F = \{ww | \, w \in \Sigma^*\}$. 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 $s = 0^p0^p \in F$.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

$$s = \underbrace{000\cdots000000\cdots000}_{\overbrace{x \quad y}^{\leftarrow \leq p \rightarrow} \quad z}$$

$y = 00$

$$s = \underbrace{000\cdots001000\cdots001}_{\overbrace{x \quad y}^{\leftarrow \leq p \rightarrow} \quad z}$$

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

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

Show: B is not regular

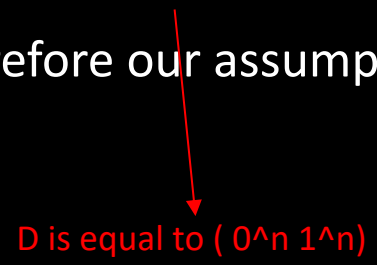
Proof by Contradiction:

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 $D = B \cap 0^*1^*$ and we already showed D is not regular. Contradiction!

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



Context Free Grammars

G_1

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

} (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

In G_1:
3 rules
R,S
0,1
S

G_2 $S \rightarrow RR$
$R \rightarrow 0R1$
$R \rightarrow \varepsilon$

Example of G_1 generating a string

Tree of substitutions

Resulting string

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. $L(G)$ is the language of all generated strings.

$\in L(G_1)$

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

Check-in 3.3

Check-in 3.3

Check <u>all</u> of the strings that are in $L(G_2)$:

(a) 001101

(b) 000111

(c) 1010

(d) ε

Ans:
A B D

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

18.404J Theory of Computation

Fall 2020