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.

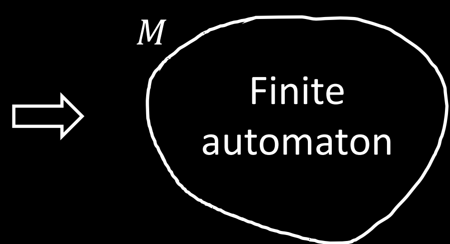
Homework due Thursday.

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'

Regular
expression R



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

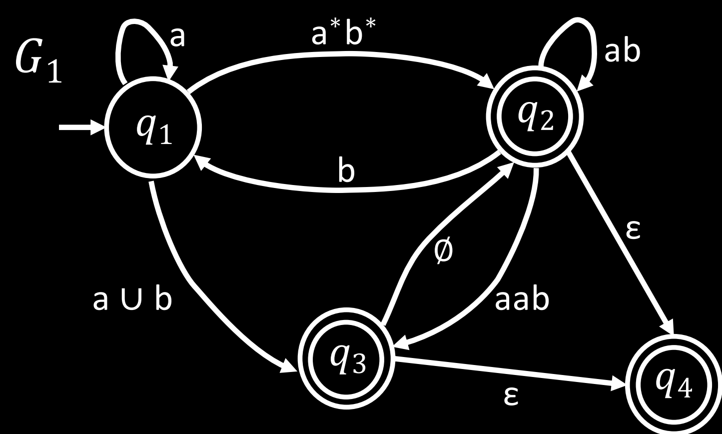
Today's Theorem: If A is regular then $A = L(R)$ for some **regular expr** R

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



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

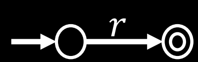
We can easily modify a GNFA to have this special form.

GNFA → Regular Expressions

Lemma: Every GNFA G has an equivalent regular expression R

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

<u>*Basis*</u> $(k = 2)$:

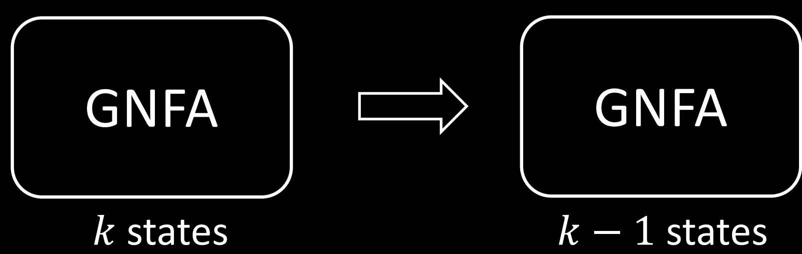
$G =$ 

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

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 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.

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 $\Sigma = \{0,1\}$.

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$\}$

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

Moral: You 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 $i \geq 0$ $\quad y^i = \underbrace{yy \cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long s

Proof: Let DFA M recognize A. Let p

$$s = \underset{}{\overset{x}{\rule{2em}{0.4pt}}}\underset{q_j}{|}\overset{y}{\rule{3em}{0.4pt}}\underset{q_j}{|}\overset{z}{\rule{3em}{0.4pt}}$$

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

$$\overset{x}{\rule{2em}{0.4pt}}\underset{q_j}{|}\overset{y}{\rule{3em}{0.4pt}}\underset{q_j}{|}\overset{y}{\rule{3em}{0.4pt}}\underset{q_j}{|}\overset{z}{\rule{3em}{0.4pt}}$$ is als

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

Check-in 3.2

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

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

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.
The pumping lemma gives p as above. Let $s = 0^p1^p \in D$.
Pumping lemma says that can divide $s = xyz$ satisfying the 3 conditions.

$$s = 000\cdots000111\cdots111$$

$$x \quad y \qquad\qquad z$$

$$\leftarrow \leq p \rightarrow$$

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

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 $s = 0^p0^p \in F$.

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.

$s = 000\cdots000000\cdots000$

$x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

$y = 00$

$s = 000\cdots001000\cdots001$

$x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

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

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 → 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 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.

Check-in 3.3

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

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

18.404J / 18.4041J / 6.840J Theory of Computation
Fall 2020