# COMS 3261 — Lecture 5:

Pumping Lemma examples; Context-Free Languages.

**Teaser:**

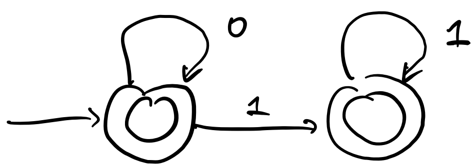
$$A = \overbrace{\{0^n 1^n \mid n \geq 0\}}^{\text{nonregular}} \cup \underbrace{\{0^n 1^m \mid n \geq 0, m \geq 0\}}_{\text{regular}}$$

Is this language regular?

Observe that

$$\{0^n 1^n \mid n \geq 0\} \subseteq \{0^n 1^m \mid n \geq 0, m \geq 0\}$$

$$A = \{0^n 1^m \mid n \geq 0, m \geq 0\}$$



accept:
$\varepsilon$
0000
00011
1111

reject: 00110 ✗

---

**Announcements:** HW #1 solutions on website
HW #3 due Monday, 7/19/21 @ 11:59 PM

**Today:**

1. Review
2. Pumping Lemma examples
3. Context-Free Grammars
4. Parse trees, derivations, ambiguity.

1. Review:

- A language is regular $\longleftrightarrow$ DFA recognizes (by definition)

$\longleftrightarrow$ NFA recognizes

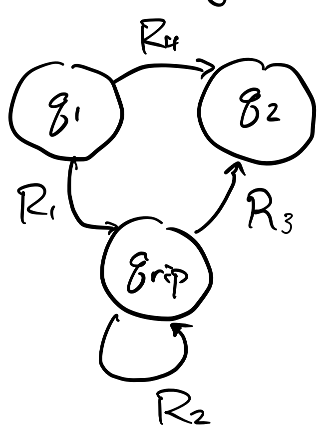
$\longleftrightarrow$ some regular expression evaluates to it.

(reg. ex $\longrightarrow$ NFA, using regular operations)

(DFA $\rightarrow$ GNFA $\longrightarrow$ regular expression)

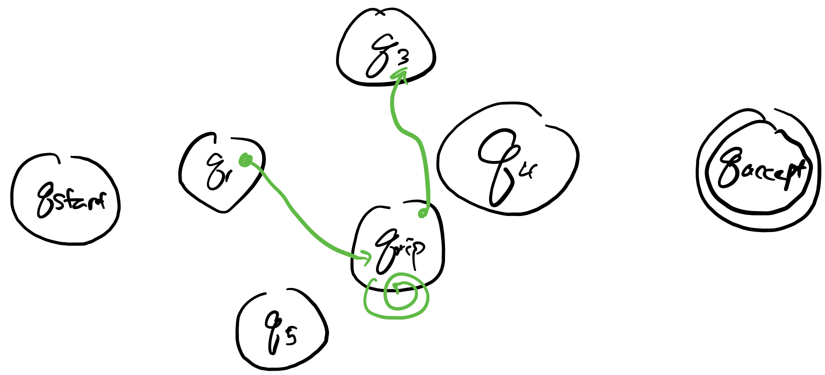
- GNFA rules:
  - Exactly one start / accept state.
  - Regular expressions labeling transition edges between every pair of states $(Q - \{q_{accept}\}) \times (Q - \{q_{start}\})$
- DFA $\longrightarrow$ GNFA

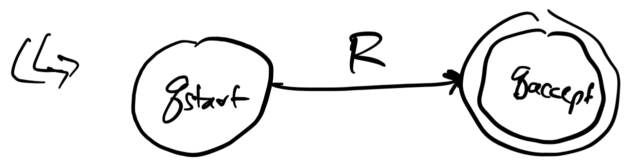
  (add $q_{start}$, $q_{accept}$, $\phi$-edges)
  
  $\hookrightarrow \{\}$
- GNFA $\rightarrow$ Reg. Ex.





repeat for every possible pair $(q_1, q_2) \in (Q - \{q_{accept}\}) \times (Q - \{q_{start}\})$



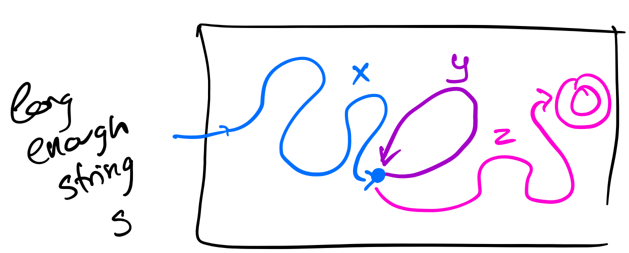




We introduced the pumping lemma, which says "all regular languages have property X."

Language doesn't satisfy X → nonregular.

<u>Pumping Lemma</u>: If $A$ is a regular language, there exists a "pumping length" $p$ such that for every string $s \in A$, $|s| \geq p$, $s$ can be divided into $x, y$, and $z$ such that:

- For each $i \geq 0$, $xy^iz \in A$
- $|y| > 0$,
- $|xy| \leq p$.



Strategy: (proving languages nonregular):

1. Assume for contradiction that PL holds. (assuming our language is regular.)

2. If our language is regular, $\exists$ some pumping length $p$ such that for $s$ in our language, $|s| \geq p$, $s$ can be divided into $xyz$ such that $|y| > 0$, $|xy| \leq p$,
$$xy^iz \in L \text{ for all } i \geq 0.$$

3. Pick some string $s \in L$ with $|s| > p$ and show that $s$ cannot be pumped — no matter how we divide $s$ into substrings $x$, $y$, and $z$ with $|y| > 0$, $|xy| \leq p$, there exists some $i$ such that $xy^iz \notin L$.

Example. Show $B = \{0^n 1^n \mid n \geq 0\}$ is not regular using the pumping lemma.

1. Assume $B$ regular, and thus PL holds.

2. Therefore there exists a pumping length $p$ such that for $|s| \geq p$, $s \in B$, $s$ can be divided into $x, y,$ and $z$, with $|y| > 0$, $|xy| \leq p$, and for all $i \geq 0$, $xy^iz \in B$.

3. Pick $s = 0^p 1^p$. $|s| = 2p \geq p$. ✓

$$s = 0000\underbrace{\cdots\cdots}_{y}0000\,111\cdots\cdots111$$

Case 1. What if $y$ is all zeroes?
Then: $xy^2z = xyyz = 0^{p+|y|}1^p \notin B$, because $|y| > 0$

Case 2. What if $y$ is all ones? Same argument.

Case 3. What if $y$ has at least one 0 and at least one 1?

$$S = \; 000 \cdots \underbrace{000111}_{y} \cdots 11$$

$$xyyz \rightarrow \; 00000\overbrace{0011}^{y}\overbrace{0011}^{y}11111.$$

$xyyz$ is not in $B$ because there exist more than one substring of 0's and 1's.

In conclusion: there is no way to divide $S$ into $x, y,$ and $z$ and satisfy the conditions of PL. Therefore $s$ cannot be pumped, our assumption that $B$ is regular leads to a contradiction, and thus $B$ is nonregular. ■

<u>Example:</u> Show that $F = \{ww \mid w \in \{0,1\}^*\}$ is nonregular.

<u>Proof:</u>

1. Assume $F$ is regular for contradiction. Thus $F$ satisfies PL.

2. Therefore, $\exists$ some pumping length $p$ such that for all strings $s \in F$, $|s| \geq p$, $s$ can be divided into $x, y,$ and $z$ such that $|y| > 0$, $|xy| \leq p$, for all $i \geq 0$, $xy^iz \in F$.

3. Pick a string $S = 0^p10^p1$

$(s = 0101$ ? what if $|0101| < p$?$)$

$(s = 0^p1^p$ ? $s \notin F)$

$0^p1^p0^p1^p$ — $s \in F$, $|s| \geq p$

$(01)^p$ $\quad 010101 \cdots 01$ $p$ even

$010|101$ if $p$ odd.

$0^p10^p1 \in F$, $|0^p10^p1| = 2p+2$.

By assumption, we can split s into x, y, z such that $|xy| \le p$.

This means xy is a substring of 0's.

$$\overbrace{\underbrace{000 \cdots 000}_{p}}^{xy} 1\, 000 \cdots 000\, 1$$

Now: $\underbrace{x}_{\text{all 0's}} \underbrace{yy}_{\text{0's}} z = 0^{p+|y|} 1 0^{p} 1 \notin F$.

Thus s cannot be pumped, so F fails the conditions of the PL. This contradicts our assumption that F is regular $\longrightarrow$ F is nonregular.

Warning: Some strings might be pumpable.

$0^p 0^p$ — not a good candidate for s.

Why? $|0^p 0^p| \ge p$.

$0^p 0^p \in F$.

$$\underbrace{0000}_{x}\overbrace{00}^{y}\underbrace{000000}_{z}$$

$xy^0z = 00000000$

$xy^2z = 0000\ 00\ 00\ 0000$

---

Example. Proving nonregularity using closure properties.

Show $\{0^n 1^n \mid n \ge 3\}$ is nonregular.

(Similar to $\{0^n 1^n \mid n \ge 0\}$.)

Know: $\{0^n 1^n \mid n \ge 0\}$ is nonregular.

Know: $\{0^n 1^n \mid n < 3\} = \{\varepsilon, 01, 0011\}$ is regular.

Observe that $\{0^n 1^n \mid n \ge 3\} \cup \{0^n 1^n \mid n < 3\}$

$= \{0^n 1^n \mid n \geq 0\}$

If $\{0^n 1^n \mid n \geq 3\}$ were regular, then $\{0^n 1^n \mid n \geq 0\}$ would also be regular by closure under union.

Because $\{0^n 1^n \mid n \geq 0\}$ is not regular, $\{0^n 1^n \mid n \geq 3\}$ must not be regular. ∎

To Prove $L$ nonregular.

If $A$ regular, $A \circ L$ is regular if $L$ is regular

$\therefore$ $A \circ L$ nonregular $\Longrightarrow$ $L$ nonregular.

Break: back at 11:27 AM.

Next up: Context-Free Grammars.

## 2. Context-Free Grammars

Idea: describe language using substitution rules.

Example.
- $A \to 0A1$
- $A \to B$
- $B \to \#$

$A \to 0A1 \to 00A11 \to 00B11 \to 00\#11.$

Variables can be substituted for other strings

(Example: $A, B$ variables. Usually written as capital letters.)

Terminals are symbols that end up in the final string because they cannot be substituted.

How to generate strings:

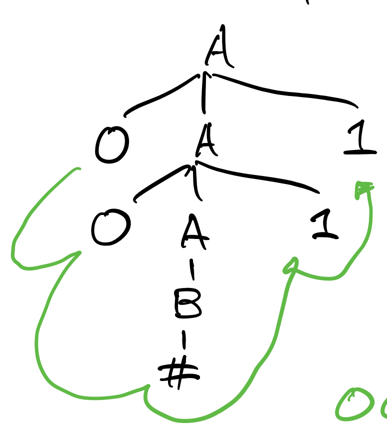
1. Writing down start variable (variable at the top left).
2. Replace any variable according to any rule.
3. Repeat until no variables remain.

$A \longrightarrow B \longrightarrow \#.$

$A \longrightarrow 0A1 \longrightarrow 0B1 \longrightarrow 0\#1.$

Def. A sequence of substitutions that creates a string of terminals from the start variable is a derivation.

We can also represent a derivation pictorally, as a parse tree.




- substitute symbols using rules, moving downward.
- done when every leaf is a terminal
- read parse trees by considering the leaves left to right.

00#11.

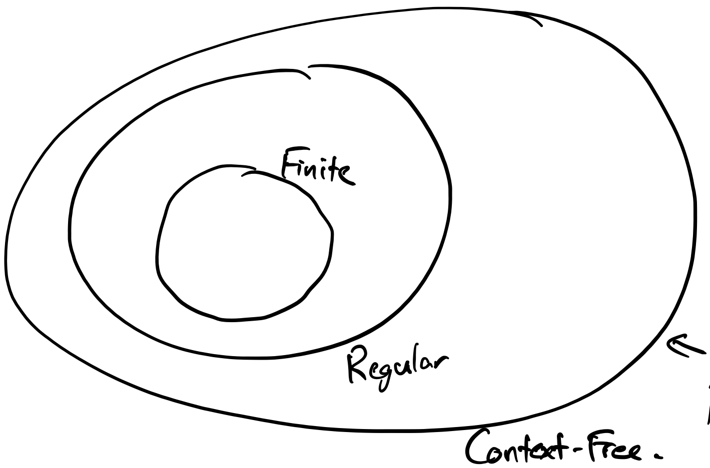
Def. The language $L(G)$ of a grammar $G$ is the set of all strings that can be produced by derivation.

$$G: \begin{array}{l} A \longrightarrow 0A1 \\ A \longrightarrow B \\ B \longrightarrow \# \end{array} \qquad L(G) = \{0^n \# 1^n \mid n \geq 0\}.$$

$$G_2: \quad \begin{array}{l} A \to 0A1 \\ A \to \varepsilon. \end{array} \qquad L(G_2) = \{0^n 1^n \mid n \geq 0\}.$$

(Note: CFGs can produce non-regular languages!)

<u>Def.</u> A language is called <u>context-free</u> if it is the language of some context-free grammar.




← not necessarily true
Regular ⊆ Context-Free.

<u>Definition (Context-Free Grammar, Formally.)</u> A context-free grammar is a 4-tuple $(V, \Sigma, R, S)$, where

$V$ is a finite set called the variables,

$\Sigma$ is a finite set called the terminals (disjoint from $V$)

$R$ is a set of substitution rules that map variables to strings of variables and terminals,

$S$ is the start variable.

For any strings of variables and terminals $u$, $v$, and $w$, we say that if $A \to w$ is a rule, $uAv \Rightarrow uwv$ (where $\Rightarrow$ indicates "yields"!)

We say $u$ derives $v$, written $u \overset{*}{\Rightarrow} v$, if $u = v$, or if there exists a sequence $u_1, u_2, \cdots, u_k$ such that

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v.$$

The language of a grammar $G$, $L(G) = \{ w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w \}$.

## Example of a CFG:

$$S \longrightarrow \langle NP \rangle \langle VP \rangle$$

// using $\langle\ \rangle$ to indicate one variable

$$\langle NP \rangle \longrightarrow AN$$

$$\langle VP \rangle \longrightarrow V \mid V \langle NP \rangle$$

// the bar '|' abbreviates two rules as one. $A \to Y, A \to X$

$A \to Y \mid X$

$V \longrightarrow$ is | eats | sees | smells

$N \longrightarrow$ dog | cat | car | shoe | person

$A \longrightarrow$ a | the

(Variables: $S, \langle NP \rangle, \langle VP \rangle, A, N, V$)
(Terminals: $\Sigma = \{a, b, c \cdots, z\}$)

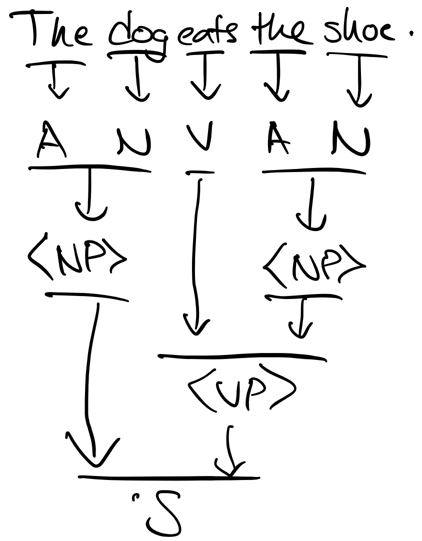
$$S \longrightarrow \langle NP \rangle \langle VP \rangle \longrightarrow AN \langle VP \rangle \longrightarrow ANV$$

$\longrightarrow$ A dog V $\longrightarrow$ the dog V $\longrightarrow$ <u>the dog eats</u>.

$$S \longrightarrow \langle NP \rangle \langle VP \rangle \longrightarrow \langle NP \rangle V \longrightarrow \langle NP \rangle V \langle NP \rangle$$

$$\longrightarrow ANV \langle NP \rangle \longrightarrow ANVAN \longrightarrow \cdots$$

"the cat sees the car"




Backwards?

Techniques.

- Easier to construct a CFG for a language if I can break it into smaller parts.

CFG for $\{0^n 1^n \mid n \geq 0\}$? –

$$S_1 \to 0 S_1 1 \mid \varepsilon$$

for $\{1^n 0^n \mid n \geq 0\}$? –

$$S_2 \to 1 S_2 0 \mid \varepsilon$$

Now I can easily create a grammar for

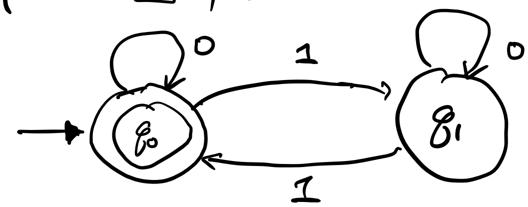
$$\{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}:$$

Add a new rule:

$$S \to S_1 \mid S_2$$
$$S_1 \to 0 S_1 1 \mid \varepsilon$$
$$S_2 \to 1 S_2 0 \mid \varepsilon$$

Technique 2: Convert any DFA to a CFG.

$\{w \mid w \in \Sigma^*, w \text{ has an even number of 1's.}\}$




To convert a DFA to a CFG:

1. Make a variable $R_i$ for each state $q_i$ of our DFA.
2. For each transition $\delta(q_i, a) = q_j$, add the rule $R_i \to aR_j$
3. Add $R_i \to \varepsilon$ for each accept state $q_i$.
4. $R_0$ is the start variable.

($\Sigma$ is the same.) ←

$V = \{R_0, R_1\}$

$R =$ $R_0 \to 0R_0 \mid 1R_1$

$R_1 \to 0R_1 \mid 1R_0$

$R_0 \to \varepsilon$

0101. In DFA: $q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_0$ ✓

In CFG: $R_0 \to 0R_0 \to 01R_1 \to 010R_1 \to 0101R_0$.

$\hookrightarrow 0101$.

(Informally proved.)

<u>Fact.</u> Any regular language is generated by some CFG, equivalently, regular languages are context-free.

---

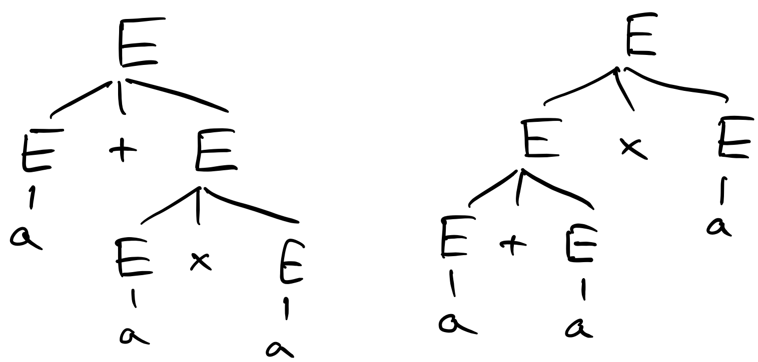
<u>Idea:</u> If a grammar generates the same string in ways corresponding to different parse trees, it's "ambiguous."

Example. $G_5 = (V, \Sigma, R, E)$,

where $V = \{E\}$, $\Sigma = (+, \times, (, ), a)$, "times symbol"

$$R: \quad E \longrightarrow E \times E \mid E + E \mid (E) \mid a$$

Derivation of $a + a \times a$?



two parse trees!

(Note: we don't care about the order in which we replace symbols — $E \rightarrow E + E \rightarrow E + a \rightarrow a + a$. (Same parse tree.)

Ⓛ $E \rightarrow E + E \rightarrow a + E \rightarrow a + a$.

Def: A leftmost derivation is one in which we always replace the leftmost variable first. A string is derived ambiguously if it has at least two leftmost derivations.

A grammar is ambiguous if some string can be derived ambiguously.

Next: More stuff about CFLs.

New automaton!

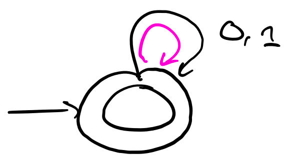
Reminder — HW #3 due Monday

HW #1 solutions are now on the website.

Reading — PL: Sipser 1.4, CFGs: Sipser 2.1.

$B = \{ w \mid w = 1^{n^2}, n \geq 0 \}$

1. Assume B regular
2. $\therefore$ B satisfies the PL.




DFA has 1 state — so $p \leq 1$.

$\underbrace{01}_{x}\underbrace{101}_{y}\underbrace{01}_{z} \in L$

$xyyz \in L$ $\quad$ $xz \in L$.

$xyyyz \in L$