Announcements:

- HW 1 — Review 5:30pm Thurs.
- HW 2 — due tonight.
- COVID

Today:

1. Using the Pumping Lemma to show <u>nonregularity</u>.

2. Context-Free Grammars — a new way to generate languages.

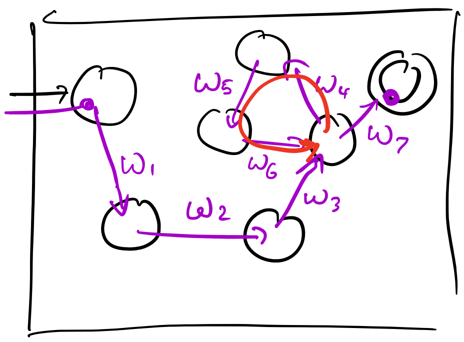
---

1. Using the Pumping Lemma.

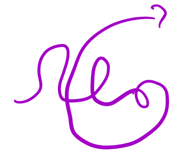
– Let L be a regular language, w/ DFA D_L, and D_L has the state set Q.

– If D_L accepts any string w with $|w| \geq |Q|$, our computation must touch some state twice and make a loop.

D_L



$w = w_1 \cdots w_7$



<u>Pumping Lemma</u>. For any regular language L, there exists some number p (the "pumping length") such that all strings $w \in L$ with $|w| \geq p$ can be divided into three parts $w = xyz$ such that

(1) $xy^iz \in L$ for all $i \geq 0$,
(2) $|y| > 0$
(3) $|xy| \leq p$.

$\underbrace{w_1 w_2 w_3}_{x} \underbrace{w_4 w_5 w_6}_{y} \underbrace{w_7}_{z}$

[skeleton proof]

Goal: Proving A is nonregular

1. Assume for contradiction A regular.
2. So, by assumption, A satisfies the PL
3. Find some long string $w \in A$ that <u>doesn't</u> meet the PL conditions
4. Contradiction $\longrightarrow$ A doesn't satisfy the PL, A not regular.

<u>Example</u>.

Show $B = \{0^n 1^n \mid n \geq 0\}$ is not regular.

1. Assume for contradiction that B is regular.

2. $\therefore$ B satisfies the PL, and has a pumping length p.

So, by assumption, any string $w \in B$, $|w| \geq p$, w can be broken into x, y, z with (1) $xy^iz \in B$ for all $i \geq 0$,

(2) $|y| > 0$

(3) $|xy| \le p$.

3. Pick a contradiction string.

$$w = 0^p 1^p, \quad \in B, \quad |w| \ge p,$$

$$w = \underbrace{000000 \;\cdots\; \overbrace{000}^{y=000}}_{p \text{ times}} 111 \cdots 111$$

x y z

- By (3), $|xy| \le p$, so any way of dividing w into x, y, z makes y all zeroes.
- By (2), $|y| > 0$, so y contains ≥ 1 zero.
- By (1), $xy^2z = xyyz \in B$

$$xyyz = 0^{p+|y|} 1^p \notin B.$$

4. This is a contradiction, so my assumption that B is regular is false. $\square$

Example 2. Show $E = \{0^i 1^j \mid i > j\}$ is not regular.

1. Assume E regular. $\therefore$ PL holds, so there exists a number p such that all $w \in E$ with $|w| \ge p$ can be broken into substrings x, y, z satisfying:

(1) $xy^iz \in E$ for all $i \ge 0$

(2) $|y| > 0$

(3) $|xy| \le p$.

(Q: why not just show we can't build a DFA?)

2. Pick my contradiction string.

$$w = 0^{p+1} 1^p. \quad \text{So:}$$

– $|xy| \le p$ by (3), so x, y are all zeroes

– $|y| > 0$, so y contains at least one zero

– $xyyz \in \mathcal{E}$. $xyyz = 0^{p+1+|y|}1^p \in \mathcal{E}$.

– instead, try $i=0$.

– (1) tells me $xy^0z = xz \in \mathcal{E}$.

$xz = 0^{p+1-|y|}1^p \notin \mathcal{E}$.

"pumping" ≈ "going around the loop"

"pumping" xyz means $xyz \to xyyz \to xyyyz$

"pumping down" means $xyz \to xz$

3. So: w can't be pumped down; so $\mathcal{E}$ does not satisfy the PL; $\mathcal{E}$ is not regular. □

Example 3.

Show $L = \{0^n1^n \mid n \ge 3\}$ is nonregular.

– Know: $A = \{0^n1^n \mid n \ge 0\}$ is non regular.

– Know: $B = \{\varepsilon, 01, 0011\}$ is regular.

Observe: $L \cup B = A$.

By closure of the regular languages under union $\cup$,

– If B, L are regular, A is regular.

– A is not regular $\Rightarrow$ either B or L is not regular

$\therefore$ L is nonregular.

5 mins — start at 1:59

$F := \{ww \mid w \in \{0,1\}^*\}$ // the language of repeated binary strings twice

(1) Suppose (for contradiction) F satisfies PL. Then there exists p s.t. for all $s \in F$ with $|s| \geq p$, s can be divided into xyz with

(1) $xy^iz \in F$ for all $i \geq 0$

(2) $|y| > 0$

(3) $|xy| \leq p$.

$0^p 1 0^p 1$ $\quad$ $000\cancel{1}0001$ $\quad$ $|s| \geq p$

- Come up w/ 3-5 candidate contradiction strings.
- Find one that violates at least one of (1)-(3) any way you split it.

(+) Is $G = \{\overbrace{0^n 1^m}^{\text{nonreg}} \mid n \geq m\} \overset{Y}{\cup} \{\overbrace{0^n 1^m}^{\text{nonreg}} \mid m \geq n\}$ Z regular? Show either way.

(++) Show $D = \{1^{n^2} \mid n \geq 0\}$ is nonregular. ⊛ end possibly

---

contradiction strings: $\quad 0^{2p} \quad 1^{2p} \quad (01)^{2p} \quad 0^p 1 0^p 1$

Choose $w = 0^p 1 0^p 1$.

By (3), $|xy| \leq p$, so x and y are all zeroes.

$\therefore xyyz = 0^{p+|y|} 1 0^p 1 \notin F$.

- $0^p 1^p 0^p 1^p$
- $\cancel{0^{2p}}$ $\quad 0^{2p+|y|}$

$G' := \{0^n 1^m \mid \underline{n \geq m \text{ or } m \geq n}\}$

$n, m \geq 0$.

regular. $\quad$ (equiv 0^*1^*)

Context-Free Grammar

A new, more powerful way of describing languages.

A CFG starts with a single start variable and repeatedly substitutes according to rules to generate a string.

Example.

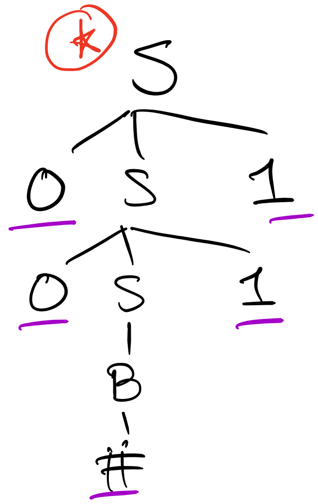
variables: usually capital letters

$$
\begin{aligned}
(1)\quad & S \longrightarrow 0S1 \\
(2)\quad & S \longrightarrow B \\
(3)\quad & B \longrightarrow \cancel{\#}\ \varepsilon
\end{aligned}
$$

terminals: can't be transformed, end with all terminals $\{0, 1, \#\}$.

Stop when all I have left are 0's, 1's, and #'s.

- $S \longrightarrow 0S1 \longrightarrow 00S11 \longrightarrow 00B11 \longrightarrow 00\#11$.
- $S \rightarrow B \rightarrow \#$.
- $S \rightarrow 0S1 \rightarrow 0B1 \rightarrow 0\#1$.



The language of our grammar is

$$\{0^n \# 1^n \mid n \geq 0\}.$$

Def. The language of all strings that can be derived from the start state is the language $L(G)$ of the grammar G.

(derived = obtained by applying rules until we have only terminals left.)

By our example: CFGs can represent some nonregular languages!

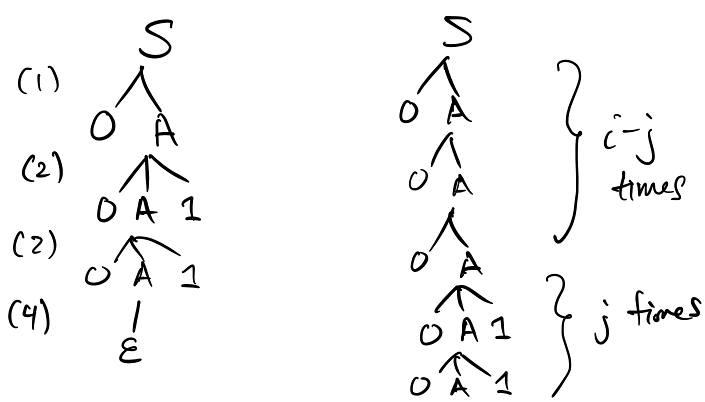
$$\mathcal{E} = \{0^i 1^j \mid i > j\}$$

(1) $S \longrightarrow 0A$

(2) $A \longrightarrow 0A1$

(3) $A \longrightarrow 0A$

(4) $A \longrightarrow \varepsilon$



$\Rightarrow 00011$

Claim: this CFG produces exactly the language $\mathcal{E}$.

$\Rightarrow$. make $0^i 1^j$ as follows: $(0^i 1^j \in \mathcal{E})$

$$0^i 1^j = 0^{i-j} 0^j 1^j$$

To make this string: use rules (1) and (3) a total of $(i-j)$ times, and rule (2) a total of j times.

$\Leftarrow$. Any string that comes out of my CFG is in $\mathcal{E}$.

(1) I must start with the rule $S \to 0A$.

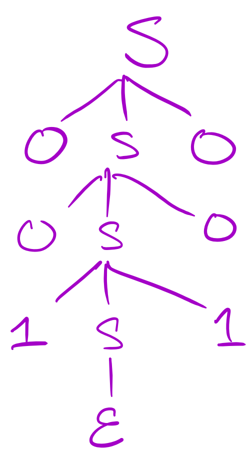
(2) Every subsequent rule keeps the total number of zeroes larger than the total number of ones.

Brief form: $S \to 0A$

$A \to 0A1 \mid 0A \mid \varepsilon$ "or"

$$H = \{ww^R \mid w \in \{0,1\}^*\}$$

$$S \to 0S0 \mid 1S1 \mid \varepsilon$$



Say $w = 001$

so $ww^R = \underline{001100} \in H$.

$S \Rightarrow 0S0 \Rightarrow 00S00$

$\Rightarrow 001S100 \Rightarrow 001100$.

"grammar"?

$S \longrightarrow N_p \; V_p$

$N_p \longrightarrow A \; N$

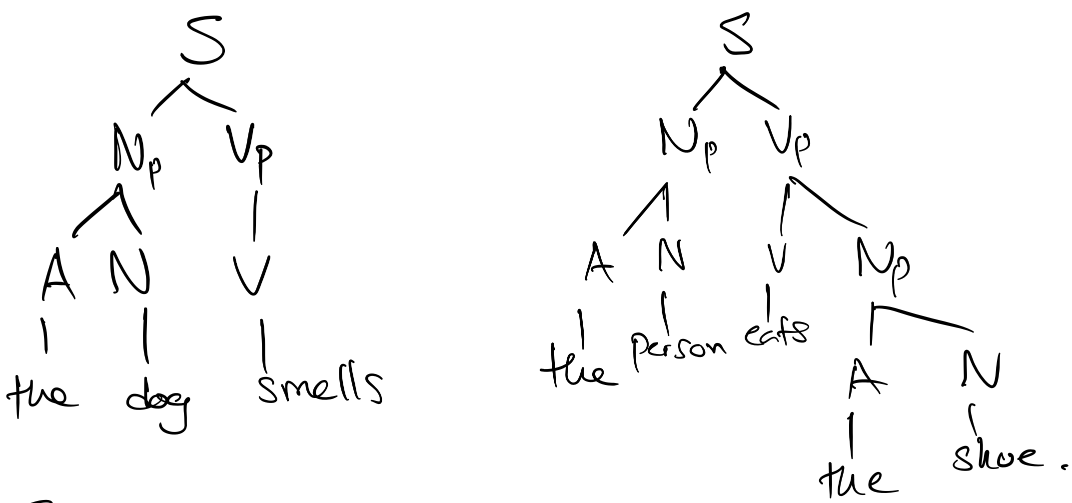
$V_p \longrightarrow V \mid V N_p$

$V \longrightarrow$ sees | smells | eats

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

$A \longrightarrow$ a | the

Adj



Ex puzzles:

- Derive "the cat sees"
- Add rule(s) that let you make adjectives, which modify nouns
- Add rule(s) that let you make adverbs, that modify verbs & adjectives.

slowly
almost
always
carefully
well

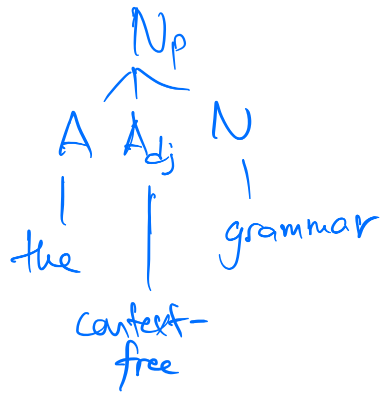
(+) "the student quickly learned context-free grammars"

(+) build a CFG for strings like "a+a", "(a x a)+a", "(a+a+a) x a", "(a x a)+(a x a)", etc.

$Np \rightarrow A\ Adj\ N$

$Adj \rightarrow Adj\ Adj$

$Adj \rightarrow$ context-free

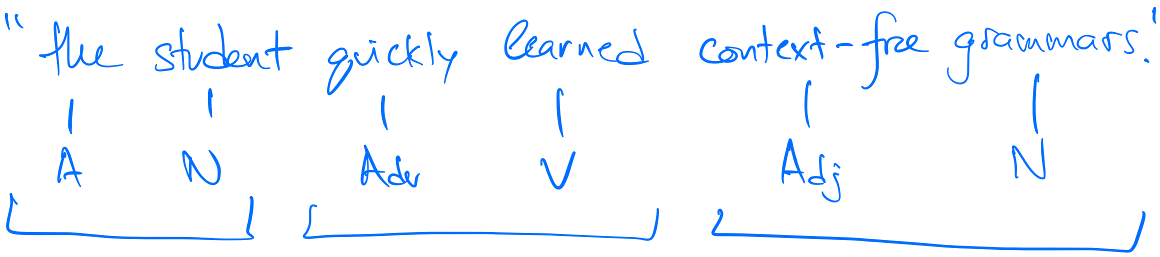


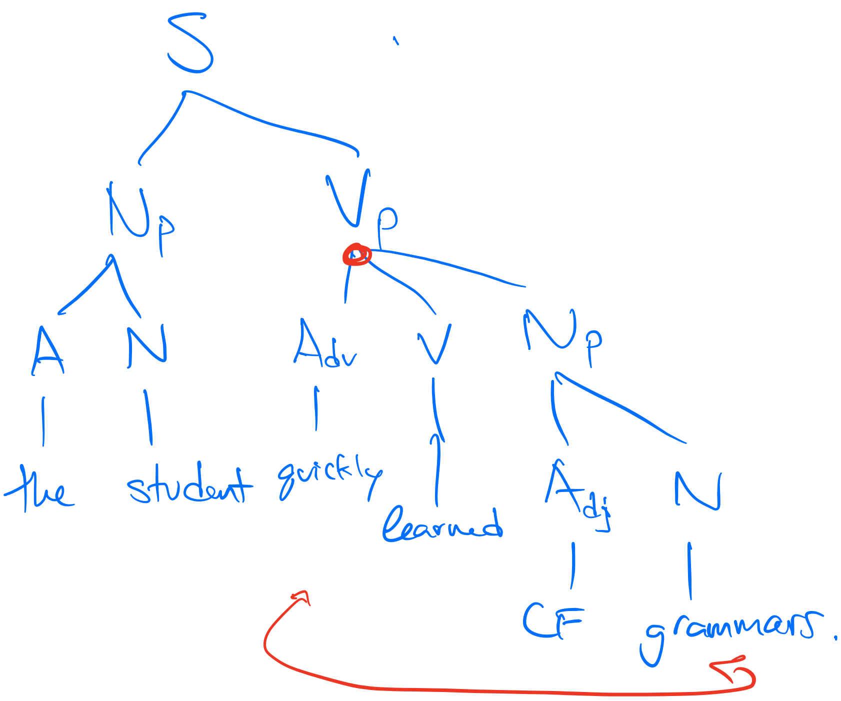
$Adj \rightarrow Adv\ Adj$

$Vp \rightarrow Adv\ V \mid Adv\ V\ Np$

$Adv \rightarrow Adv\ Adv$

$Adv \rightarrow$ quickly | slowly | well.





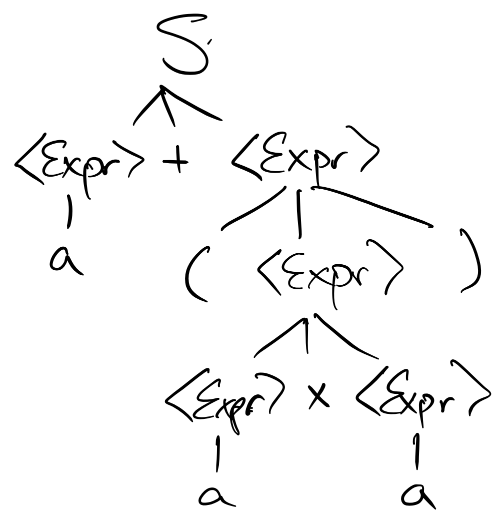
Math CFG:

$S \longrightarrow \langle Expr \rangle$

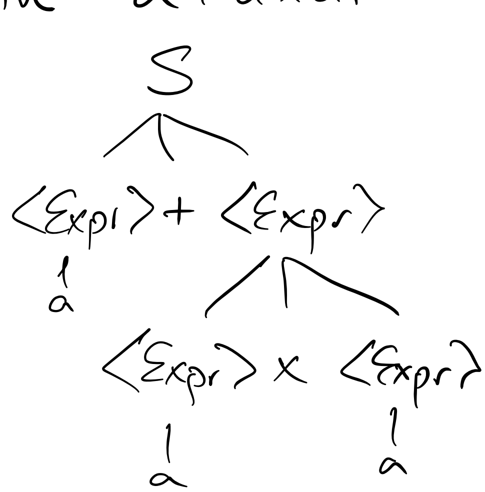
$\langle Expr \rangle \longrightarrow \langle Expr \rangle + \langle Expr \rangle \mid \langle Expr \rangle \times \langle Expr \rangle$

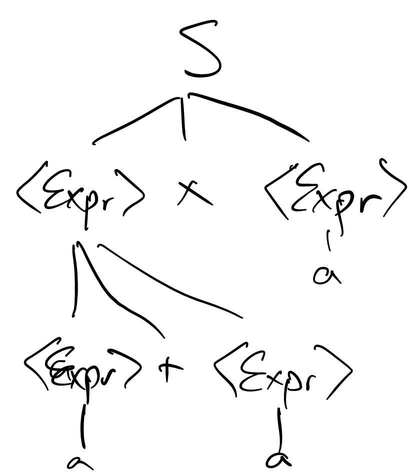
$\langle Expr \rangle \longrightarrow (\langle Expr \rangle) \mid a$

$a + (a \times a)$



Derive $a + a \times a$.





Takeaway: derivations are <u>not</u> always unique.

A grammar with multiple derivations for a string is ambiguous. (For more: Sipser pp. 108-111 for details.)

break: 5 minutes

back at 3:30.

Def (CFG, Formal.) A Context-Free Grammar is a 4-tuple (V, Σ, R, S), where:

V is a finite set of variable symbols,
Σ is a set of terminal symbols,
S is the start variable,
and R is a finite set of substitution rules, where each rule consists of one input variable $v \in V$, and an output string over $V \cup \Sigma$.

Short form:

start variable @ top left

V, set of variables

$$S \to 0A0$$
$$A \to 0A0 \mid 1A1$$
$$A \to B+ \mid B\times$$
$$B \to \varepsilon$$

Σ, the terminals, are all symbols that are not variables
$\{0, 1, +, \times\}$

"00AB+"

Also:
If A is a variable, and u, v, and w are strings over $V \cup \Sigma$, and $A \to w$ is a rule, we write

$$uAv \Rightarrow uwv \quad \text{"}uAv \text{ yields } uwv\text{"}$$

If u and v are strings over $V \cup \Sigma$, can write

$$u \overset{*}{\Rightarrow} v \qquad \text{"u derives v"}$$

if there is a sequence of rules that transforms u to v.

Finally: CFG tricks

1. CFG union trick. Given

$$G_1 = \{V_1, \Sigma_1, R_1, S_1\}$$
$$G_2 = \{V_2, \Sigma_2, R_2, S_2\}$$

$(V_1 \cap V_2 = \emptyset$
$\Sigma_1 \cap \Sigma_2 = \emptyset)$

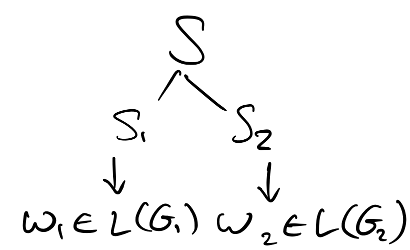
G_3. $\quad S \to S_1 \mid S_2$

$\vdots$

R_1, R_2.

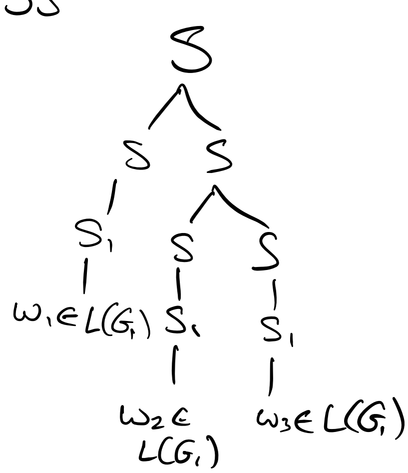
2. CFG concatenation trick.

$$S \to S_1 S_2$$



3. CFG star trick

(Grammar for $L(G_1)^*$) $\quad S \to \varepsilon \mid S_1 \mid SS$



Prop. CFGs can represent every regular language

Proof sketch: we'll show how to build a CFG for any regular expression.

Regular expressions come in 6 types:

$a \in \Sigma \longrightarrow S \to a,$

$\varepsilon \longrightarrow S \to \varepsilon,$

$\emptyset \longrightarrow S \to S,$ or "no rules," etc

any CFG that makes no valid strings.

$R_1 \cup R_2 \longrightarrow$ union trick

$R_1 R_2 \longrightarrow$ concat trick

$R_1^* \longrightarrow$ star trick. $\square$

Takeaway: CFGs can represent all regular languages, and some nonregular languages too!