Comp 330 - Lecture 13 - October 17th

IE: Ogni morte di papa
↳ Every death of a pope
Once in a blue moon

Midterm : Median 85
Mode 100

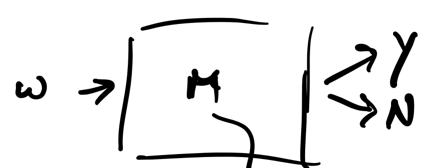
Come to OHs (MC 110)

Midterm evaluation

Prakash has agreed to give a guest lecture
Intro comp. theory => Halting Problem
Nov 2th/7th → Evening lecture

Introduction to grammars

Automata theory → DFA / NFA / NFA + ε
↓
Reg. Language recognizes/acceptors



→ Reg Exp M = user-friendly
Limitation: Finite memory

This limitation prevents FA accepting
$\{a^n b^n : n \in \mathbb{N}\}$

However, there are still comp ~~formalisms to~~ represent / model non-reg languages. One of those is known as the <u>grammar</u>:

G generates strings in a language

$\boxed{G} \rightarrow \{w_1, w_2, w_3, \ldots\} = L(G)$

$\downarrow$

language generated by G

Why do we need grammars?

Grammars provide a way to simply / concisely model non-reg languages. How?

Grammars are able to capture the recursive nature underlying several non-reg languages.

Intuition: $\underbrace{\text{FA} + \underset{\text{Stack}}{\sqcup}}_{\text{PDA}}$ $\qquad$ $\underbrace{\text{Grammars}}$

PDA $\downarrow$ Iterative procedure $\qquad$ $\downarrow$ Recursive procedure

<u>Ex</u> Design a grammar which generates

$L = \{a^n b^n : n \in \mathbb{N}\}$

What does $w \in L$?

$w = \varepsilon$ (Base case)

$w = a\,x\,b$ (Recursive case)

$\quad\quad\ \downarrow$

$\quad\quad\ \in L$

Grammar $G = (V, S, \overset{\Sigma}{\overset{\parallel}{T}}, P)$ sequence of terminals

$S \to \square \to \square \to \square \to aabb$

$\uparrow$ start variable

$\uparrow$ use P to replace S w/ some intermediate form

$S \to \varepsilon$ (1) $\to$ Base case

$\uparrow$ derives / produces

$S \to \varepsilon \Rightarrow G$ can generate ε.

$S \to aSb$ (2) $\to$ Recursive case

Generate $aabb$?

$$\underbrace{S \xrightarrow{\text{using (2)}} aSb \xrightarrow{\text{using (2)}} a \cdot aSb \cdot b \xrightarrow{\text{using (1)}} aa \cdot \varepsilon \cdot bb}_{\text{derivation}} = aabb$$

In general: this grammar G generates the language L since there is a derivation of 0 or more steps ~~starting~~ starting w/ S & ending w/ $a^n b^n$ $\forall n \in \mathbb{N}$ (& only ending in $a^n b^n$).

Def (Grammar) A grammar is a 4-tuple $G = (V, S, T, P)$

$V \to$ The finite set of variables (non-terminals)

$\neq \emptyset$ upper case A, B, S

$S \rightarrow$ the unique start variable, $S \in V$.

ϕ^{T} $T \rightarrow$ the finite set of terminals (if G generates $L \subseteq \Sigma^*$, $T = \Sigma$) ^{a,b}

$P \rightarrow$ the finite set of production rules $P \subseteq (V \cup T)^* \times (V \cup T)^*$

$V = \{A, B, S\}$ $\quad T = \{a, b\}$

$S \rightarrow aSb \quad S \rightarrow \varepsilon$

$\underbrace{AB}_{\text{head of the production rule}} \rightarrow \underbrace{SAa}_{\text{body of prod}} \qquad ABBa \rightarrow Sa$

There are different types of grammars which vary in expressiveness based on the allowed form of P:

- Right-linear grammar

  $\underbrace{A}_{\in V} \rightarrow \underbrace{x}_{\in T^*} \underbrace{B}_{\in (V \cup \{\varepsilon\})}$

  $T = \{0, 1\} \quad V = \{S\}$

  $S \rightarrow 01S$ ✓

  $S \rightarrow 0S1$ ×

  $S \rightarrow S0$ ×

  $S \rightarrow 01$

- Context-free grammar

  $\underbrace{A}_{\in V} \rightarrow \underbrace{B}_{\in (V \cup T)^*}$

  $S \rightarrow 0S1$ ✓

  $S \rightarrow S0S1S$ ✓

  $0S0 \rightarrow 11$ ×

- Context-sensitive grammar.

  $\underbrace{A}_{\in (V \cup T)^*} \rightarrow \underbrace{B}_{\in (V \cup T)^*}$

  $|A| \leq |B|$

  # of symbols in A

- Unrestricted grammar

$A \to B$

$A \in (V \cup T)^*$, $B \in (V \cup T)^*$

$S \to 0S1$ ✓

$S0S \to 0S11$ ✓

$S0 \to 0$ ×

→ Equivalent in expressiveness to Turing Machines!

Does G always Terminate?

$S \to \varepsilon \quad S \to aSb$

$S \to aSb \to aaSbb \in L(G)$

$\notin T$

G: $S \to aSb$ → Yes this is valid?

$L(G) = \emptyset$

<u>Ex</u> Give a <u>CFG</u> that generates

$L = \{ w \in \{a, b\}^* : w = w^R \}$

palindromes over $\{a, b\}$

Base case? ε, a, b

Recursive case? If $w = w^R$ & $|w| \geq 2$

then $w = \sigma x \sigma \quad \sigma \in \{a, b\}$

$x \in L$

$S \to \varepsilon \quad S \to aSa \quad S \to bSb \iff \boxed{S \to \varepsilon | a | b. \text{ (Base case)}}$

$S \to aSa \to \dots \to axa$
$S \to bSb \to \dots \to bxb$ } Recursive case

$S \to aSa \mid bSb$

$G = (V = \{S\}, S, T = \{a, b\}, P)$
where P is
$S \to \varepsilon \mid a \mid b \mid aSa \mid bSb$

__Exercise__ T/F $\exists$ RLG G s.t. $L(G) = L$ (as above.)

So far, string / lang derivation has been intuitive. How can we formalize it?

__Def__ (n-step derivation relation)
Grammar $G = (V, S, T, P)$, $\alpha, \beta \in (V \cup T)^*$, $n \in \mathbb{N}$, we write $\alpha \xrightarrow[G]{n} \beta$ if G can derive/produce β from α in exactly n steps.

In previous grammar, $S \xrightarrow[G]{1} a$
$S \xrightarrow[G]{3} abbba$

Formally: $\quad \alpha \underset{G}{\overset{0}{\Rightarrow}} \alpha \quad , \forall \alpha \in (V \cup T)^*$

$\alpha \underset{G}{\overset{1}{\Rightarrow}} \beta$ if $\exists \; \alpha_1, \alpha_2, \alpha_3 \in (V \cup T)^*$
$\gamma \in (V \cup T)^*$
where $\alpha_2 \to \gamma \in P$
& $\alpha = \alpha_1 \alpha_2 \alpha_3$
$\beta = \alpha_1 \gamma \alpha_3$

$\alpha \underset{G}{\overset{n+1}{\Rightarrow}} \beta$ if $\exists \; \gamma \in (V \cup T)^*$
s.t. $\alpha \underset{G}{\overset{n}{\Rightarrow}} \gamma$,
$\gamma \underset{G}{\overset{1}{\Rightarrow}} \beta$

This is the n-step derivation relation. but a grammar generates a string w if $\exists$ a derivation w/ any # of derivation steps. (from S to w)

<u>Def</u> (*-step derivation) $\quad G = (V, S, T, P)$
$\alpha, \beta \in (V \cup T)^*$, $\quad \alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$ if $\alpha \underset{G}{\overset{n}{\Rightarrow}} \beta$ for some $n \geq 0$.

<u>Notice</u> *-step $\underset{G}{\overset{*}{\Rightarrow}}$ is a relation on $(V \cup T)^*$

R? Yes! $\quad \alpha$, $n=0$ $\quad \alpha \underset{G}{\overset{*}{\Rightarrow}} \alpha$

T? Yes! $\quad \alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$, $\beta \underset{G}{\overset{*}{\Rightarrow}} \gamma \Rightarrow \alpha \underset{G}{\overset{*}{\Rightarrow}} \gamma$

S? No! $\quad S \to \varepsilon$, $\quad \varepsilon \to S$

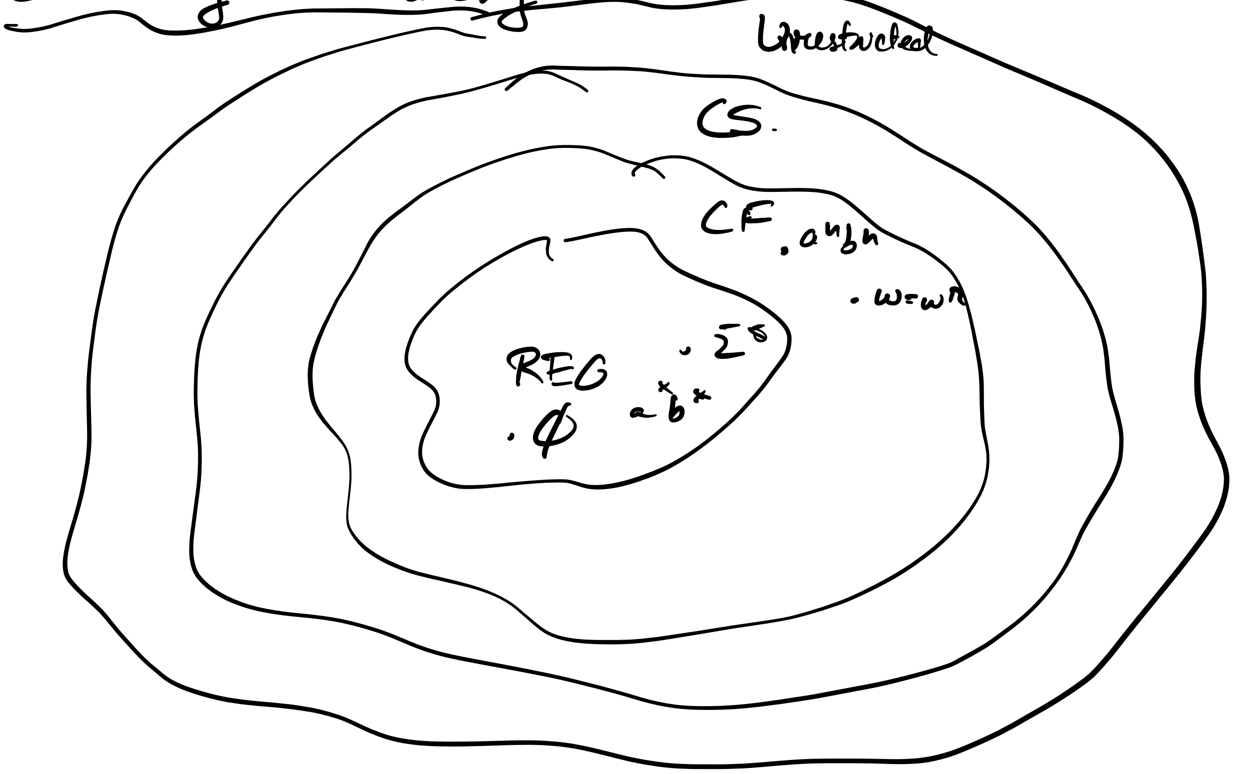
↳ Meaningless prod rule.

Def Given $G=(V,S,T,P)$

$$L(G)=\{w\in T^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

Def <u>Context-free languages</u> (CFL) $\Sigma\neq\emptyset$, $L\subseteq\Sigma^*$, Then L is a CFL if $\exists$ a CFG G s.t. $L(G)=L$.

<u>Chomsky hierarchy</u>



<u>Claim</u> $L_{REG} \subset L_{CF}$. $\Sigma\neq\emptyset$

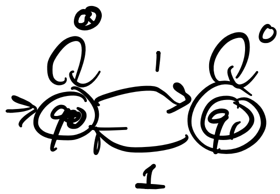
$$L_{REG} = L_{RLG} = \{L\subseteq\Sigma^* : \exists \text{ RLG } G \text{ s.t. } L(G)=L\}$$

to show $L_{REG} = L_{RLG}$ $\Rightarrow$

$L_{REG} \subseteq L_{RLG}$

if L is REG $\Rightarrow \exists$ RLG G s.t. $L(G) = L$

DFA M $L(M) = L$, convert to a RLG.



DFA M

$\Downarrow$

RLG G

Start variable $\to$ $q_0 \to 0q_0 \mid 1q_1$

$q_1 \to 0q_1 \mid 1q_0 \mid \varepsilon$

$L_{RLG} \subseteq L_{REG}$

if $L = L(G)$ for some RLG G, then L is REG

Convert RLG to an FA.

$S \to 01S \mid \varepsilon \mid 01A$

$A \to 10A \mid 10$