CS310: Automata Theory

Context Free Grammars

Paritosh Pandya

Indian Institute of Technology, Bombay
Course URL: https://cse.iitb.ac.in/~pandya58/CS310/automata.html

Autumn, 2023

Context Free Languages and Push Down Automata

- Context free grammars and Generated languages
  Kozen L19, L20.
- Grammar Normal Forms
  Kozen L21
- Pumping Lemma for CFL
  Kozen L22
- Properties of CFL

Phrase Structured Grammars

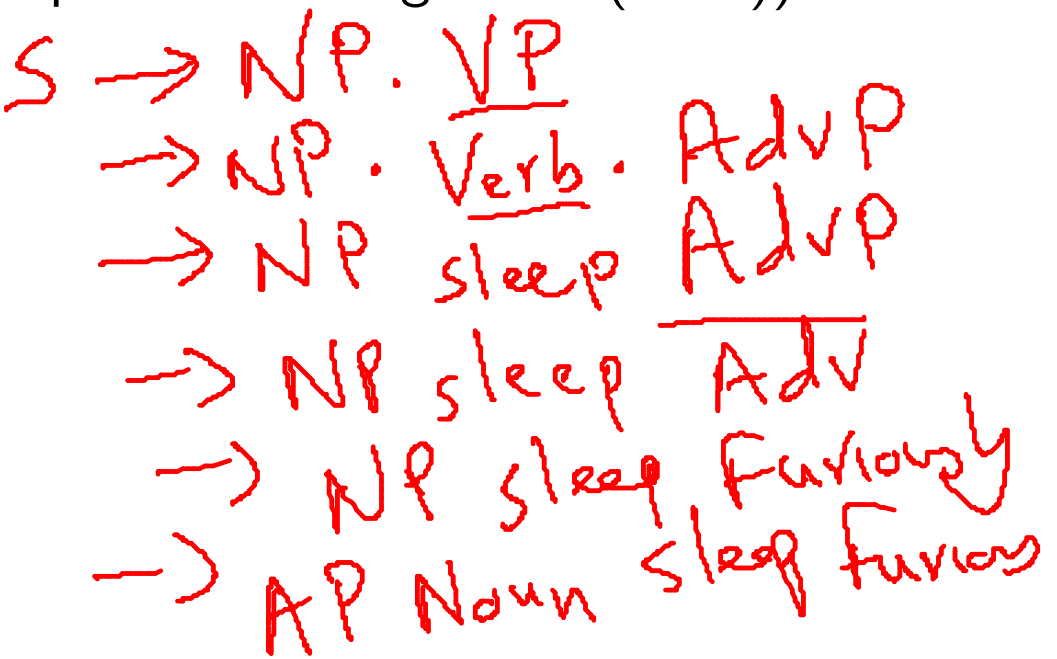
Noam Chomsky (Founded Computational Linguistics (1950))

- $S \rightarrow NP \;\; VP$
  $NP \rightarrow (AP) \;\; Noun$
  $AP \rightarrow Adj \;\; (AP)$
  $VP \rightarrow Verb \;\; (AdvP)$
  $AdvP \rightarrow Adv \;\; (AdvP)$

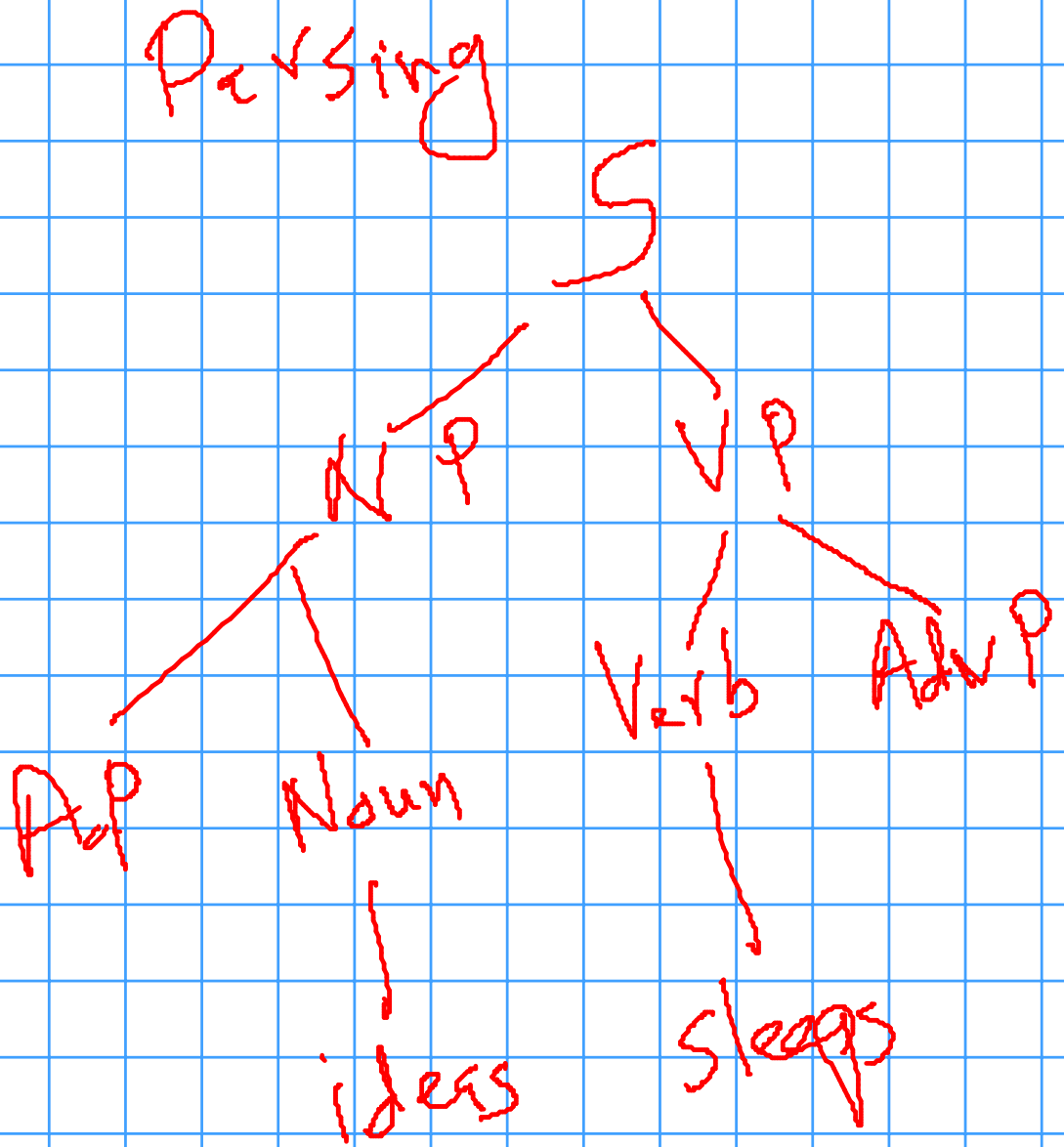
Phrase Structured Grammars

Noam Chomsky (Founded Computational Linguistics (1950))

- $S \rightarrow NP \quad VP$
  $NP \rightarrow (AP) \quad Noun$
  $AP \rightarrow Adj \quad (AP)$
  $VP \rightarrow Verb \quad (AdvP)$
  $AdvP \rightarrow Adv \quad (AdvP)$



- Derive English Sentences
  colorless green ideas sleep furiously


Parsing
S
NP
VP
PrP
Noun
Verb
AdvP
ideas
sleeps

Phrase Structured Grammars

Noam Chomsky (Founded Computational Linguistics (1950))

- $S \rightarrow NP \;\; VP$
  $NP \rightarrow (AP) \;\; Noun$
  $AP \rightarrow Adj \;\; (AP)$
  $VP \rightarrow Verb \;\; (AdvP)$
  $AdvP \rightarrow Adv \;\; (AdvP)$

- Derive English Sentences
  colorless green ideas sleep furiously
- Grammatically correct sentences.

Arithmetic Expressions

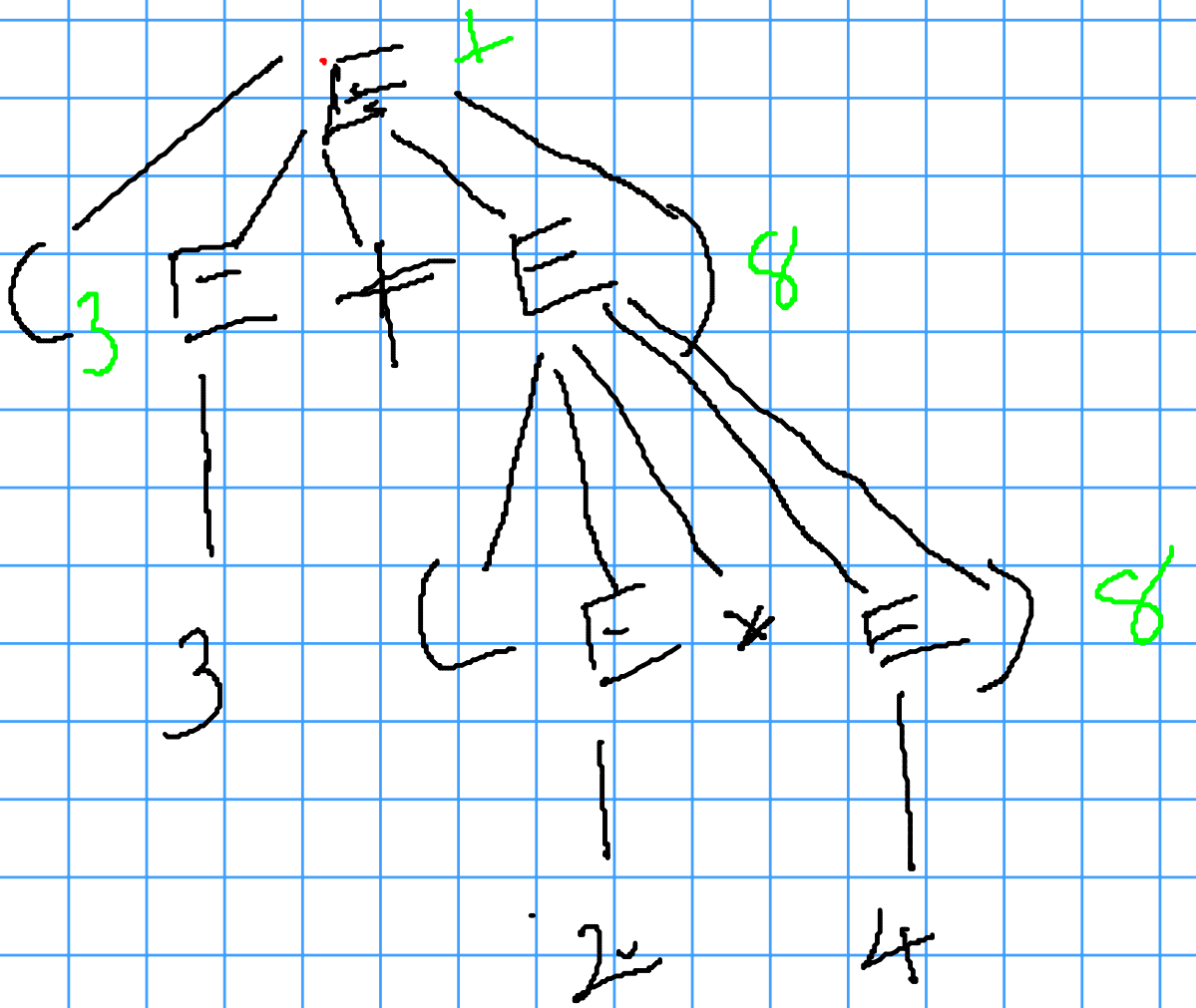
E.g. (3 + (2*4)) Fully Parenthesized expressions made from digits.

1) $E \rightarrow Digit$
2) $E \rightarrow (E + E)$
3) $E \rightarrow (E * E)$

$E \Rightarrow (E + E) \Rightarrow (3 + E) \Rightarrow (3 + (E * E)) \Rightarrow (3 + (2 * E)) \Rightarrow (3 + (2 * 4))$

E -> Digit | (E+E) | (E*E)

- Context free Grammars (CFG).
- Syntactic categories (Nonterminals) like E.
- Symbols of alphabet (Terminals) like + *) (.
- Derivation rules (Productions)
- Start Symbol like E or S
- Useful for defining artificial languages and their processing.


E
+
E
E
3
3
E
x
E
2x
4x

Context Free Grammar

CFG $G = (N, \Sigma, P, S)$ where

- N finite set of nonterminal symbols.
- Σ finite set of terminal symbols (alphabet). Disjoint from N
- S start symbol where $S \in N$.

Context Free Grammar

CFG $G = (N, \Sigma, P, S)$ where

- N finite set of nonterminal symbols.
- Σ finite set of terminal symbols (alphabet). Disjoint from N
- S start symbol where $S \in N$.
- $P \quad \subseteq_{fin} \quad N \times (N \cup \Sigma)^*$ finite set of productions.

Context Free Grammar

CFG $G = (N, \Sigma, P, S)$ where

- N finite set of nonterminal symbols.
- Σ finite set of terminal symbols (alphabet). Disjoint from N
- S start symbol where $S \in N$.
- $P \subseteq_{fin} N \times (N \cup \Sigma)^*$ finite set of productions.
- Each production has the form (A, α) with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$.
  Abbreviated $A \to \alpha$. Also $A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$

Context Free Grammar

CFG $G = (N, \Sigma, P, S)$ where

- N finite set of nonterminal symbols.
- Σ finite set of terminal symbols (alphabet). Disjoint from N
- S start symbol where $S \in N$.
- $P \subseteq_{fin} N \times (N \cup \Sigma)^*$ finite set of productions.
- Each production has the form (A, α) with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$.
  Abbreviated $A \to \alpha$. Also $A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$

E.g. $G = (\{S\}, \{a, b\}, P, S)$ with productions
$S \to aSb \mid \epsilon$.

Context Free Grammar

CFG $G = (N, \Sigma, P, S)$ where

- N finite set of nonterminal symbols.
- Σ finite set of terminal symbols (alphabet). Disjoint from N
- S start symbol where $S \in N$.
- $P \subseteq_{fin} N \times (N \cup \Sigma)^*$ finite set of productions.
- Each production has the form (A, α) with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$.
  Abbreviated $A \to \alpha$. Also $A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$

E.g. $G = (\{S\}, \{a, b\}, P, S)$ with productions
$S \to aSb \mid \epsilon$.

Derivation as a sequence of rewrites

$S \to aSb \to aaSbb \to aaaSbbb \to aaabbb$

$L(a) = \{a^n b^n \mid n \geq 0\}$

Derivation

- Sentential forms $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, Nonterminals A, B, C, Terminals a, b, c.

Derivation

- Sentential forms $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, Nonterminals A, B, C, Terminals a, b, c.
- $\alpha \xrightarrow{1}_G \beta$ provided $\alpha = \alpha_1 A \alpha_2$ and $A \to \gamma \in P$ and $\beta = \alpha_1 \gamma \alpha_2$.

Derivation

- Sentential forms $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, Nonterminals A, B, C, Terminals a, b, c.
- $\alpha \xrightarrow{1}_G \beta$ provided $\alpha = \alpha_1 A \alpha_2$ and $A \to \gamma \in P$ and $\beta = \alpha_1 \gamma \alpha_2$.
- $\alpha \xrightarrow{0}_G \alpha$ and $\alpha \xrightarrow{n+1}_G \beta$ if $\alpha \xrightarrow{n}_G \gamma \xrightarrow{1}_G \beta$ for some γ.

Derivation

- Sentential forms $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, Nonterminals A, B, C, Terminals a, b, c.
- $\alpha \xrightarrow{1}_G \beta$ provided $\alpha = \alpha_1 A \alpha_2$ and $A \to \gamma \in P$ and $\beta = \alpha_1 \gamma \alpha_2$.
- $\alpha \xrightarrow{0}_G \alpha$ and $\alpha \xrightarrow{n+1}_G \beta$ if $\alpha \xrightarrow{n}_G \gamma \xrightarrow{1}_G \beta$ for some γ.
- Derivable $\alpha \xrightarrow{*}_G \beta$ if $\alpha \xrightarrow{n}_G \beta$ for some $n \geq 0$.

Derivation

- Sentential forms $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, Nonterminals A, B, C, Terminals a, b, c.
- $\alpha \xrightarrow{1}_G \beta$ provided $\alpha = \alpha_1 A \alpha_2$ and $A \to \gamma \in P$ and $\beta = \alpha_1 \gamma \alpha_2$.
- $\alpha \xrightarrow{0}_G \alpha$ and $\alpha \xrightarrow{n+1}_G \beta$ if $\alpha \xrightarrow{n}_G \gamma \xrightarrow{1}_G \beta$ for some γ.
- Derivable $\alpha \xrightarrow{*}_G \beta$ if $\alpha \xrightarrow{n}_G \beta$ for some $n \geq 0$.
- Language generated by CFG G contains all words over Σ derivable from S
  $L(G) = \{x \in \Sigma^* \mid S \xrightarrow{*}_G x\}$

Derivation

- Sentential forms $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$, Nonterminals A, B, C, Terminals a, b, c.
- $\alpha \xrightarrow{1}_G \beta$ provided $\alpha = \alpha_1 A \alpha_2$ and $A \to \gamma \in P$ and $\beta = \alpha_1 \gamma \alpha_2$.
- $\alpha \xrightarrow{0}_G \alpha$ and $\alpha \xrightarrow{n+1}_G \beta$ if $\alpha \xrightarrow{n}_G \gamma \xrightarrow{1}_G \beta$ for some γ.
- Derivable $\alpha \xrightarrow{*}_G \beta$ if $\alpha \xrightarrow{n}_G \beta$ for some $n \geq 0$.
- Language generated by CFG G contains all words over Σ derivable from S
  $L(G) = \{x \in \Sigma^* \mid S \xrightarrow{*}_G x\}$
- A language L is context-free if $L = L(G)$ for some CFG G.

Examples

Grammar G_1 with productions $S \rightarrow aSb \mid \epsilon$

Examples

Grammar G_1 with productions $S \rightarrow aSb \mid \epsilon$

- Derivation giving *aaabbb*

Examples

Grammar G_1 with productions $S \rightarrow aSb \mid \epsilon$

- Derivation giving *aaabbb*

- Claim $L(G_1) = \{ a^n b^n \mid n \geq 0 \}$ (How to prove?)

Proof: $S \overset{*}{\Rightarrow} x \Rightarrow x \in a^n b^n$

Ind. on no of derivation steps.

$(n=0)$ $S \rightarrow \epsilon \in$ clear that $\in L$

$(n=m+1)$ $S \Rightarrow a^{m+1} b^{m+1} \in L$

$a^n b^n \in L(G)$

Prove by Ind. on n.

$(n=0)$ $\varepsilon \in S \to \varepsilon$

$(n=m+1)$ $a a^m b^m b$

$S \xrightarrow{m} a^m b^m$ Ind. Hyp

$aSb \xrightarrow{m} a a^m b^m b$

$S \xrightarrow{1} aSb \xrightarrow{m} a a^m b^m b$

Examples

Grammar G_1 with productions $S \rightarrow aSb \mid \epsilon$

- Derivation giving $aaabbb$
- Claim $L(G_1) = \{ a^n b^n \mid n \geq 0 \}$ (How to prove?)

Grammar G_2 with productions $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$

- Answer $L(G_1) = \{ w \in \{a,b\}^* \mid w = xx^R \text{ or } x = yay^R \text{ or } x = yby^R \}$

Balanced Parenthesis Language

Example [[]][] but not []][[].

PAREN contains all strings $x \in \{[,]\}^*$ s.t.

1. $\#[(x) = \#](x)$ count of left and right brackets is equal.
2. for any prefix y of x we have $\#[(y) \geq \#](y)$.

Balanced Parenthesis Language

Example [[]][] but not []][[].

PAREN contains all strings $x \in \{], [\}^*$ s.t.

1. $\#[(x) = \#](x)$ count of left and right brackets is equal.
2. for any prefix y of x we have $\#[(y) \geq \#](y)$.

Balanced Parenthesis Grammar

$S \rightarrow SS \mid [S] \mid \epsilon$

$S \rightarrow SS \rightarrow [S]S$

Theorem $PAREN = L(G)$

($\supseteq$) $S \xrightarrow{*} x$ then $x \in PAREN$

$S \xrightarrow{n} \alpha$ then α satisfies (1) & (2), $\forall n$.

Proof by Ind. on length of derivation

Basis ($n=0$) $S \xrightarrow{0} S$ Clearly S satisfies (1) & (2)

Ind. Step: $S \xrightarrow{n} \beta_1 S \beta_2 \xrightarrow{1} \alpha$

Ind Hyp: $\beta_1 S \beta_2$ satisfies (1) & (2)

$\Rightarrow$ $\beta_1 \beta_2$ also satisfies (1) & (2). ✓

Case 1 ($S \to \epsilon$): $\therefore \alpha = \beta_1 \beta_2$

Case 2 ($S \to SS$): $\alpha = \beta_1 S S \beta_2$

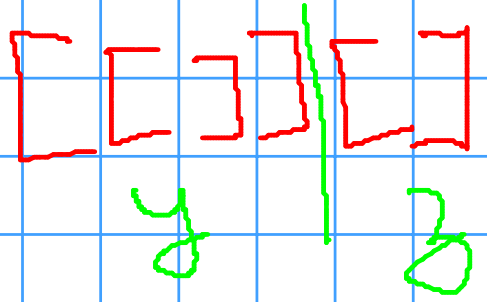
Case 3 ($S \to [S]$): $\alpha = \beta_1 [S] \beta_2$

If $x \in$ C-PAREN then $S \Rightarrow x$.

Proof: Ind on $|x|$

Basis ($|x|=0$) $S \Rightarrow \varepsilon$ ✓

Ind. step. $x \in$ C-PAREN



Case 1. $\exists y$ proper prefix of x. $|y|>0$ s.t.

y satisfies (1) & (2)

$\therefore x = yz$. &

y & z satisfy (1) & (2)

By Ind. hyp $S \overset{*}{\Rightarrow} y$. & $S \overset{*}{\Rightarrow} z$.

$\therefore S \Rightarrow SS \overset{*}{\Rightarrow} yS \overset{*}{\Rightarrow} yz = x$

Case 2. $x = [\,y\,]$

$\therefore$ y - PAREN

(Ind. Hyp.) $S \xrightarrow{*} y$.

$S \to [\,S\,] \xrightarrow{*} [\,y\,]$

Applications of CFG

- Defining syntax of programming languages.
- Parsing a given word.
  Syntax directed processing.

See HMU Section 5.3 (Self Study).

Backus Naur form grammar for Statement syntax in PL

<stmt> ::= <if-stmt> | <while-stmt> | <begin-stmt> | <assg-stmt>

<if-stmt> ::= **if** <bool-expr> **then** <stmt> **else** <stmt>

<while-stmt> ::= **while** <bool-expr> **do** <stmt>

<begin-stmt> ::= **begin** <stmt-list> **end**

<stmt-list> ::= <stmt> | <stmt> ; <stmt-list>

<assg-stmt> ::= <var> := <arith-expr>

<bool-expr> ::= <arith-expr><compare-op><arith-expr>

<compare-op> ::= $< \mid > \mid \leq \mid \geq \mid = \mid \neq$

<arith-expr> ::= <var> | <const> | (<arith-expr><arith-op><arith-expr>)

<arith-op> ::= $+ \mid - \mid * \mid /$

<const> ::= $0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

<var> ::= $a \mid b \mid c \mid \cdots \mid x \mid y \mid z$

Backus Naur Form Grammar for Statement Syntax in PL

<stmt>

<while-stmt>

while <bool-expr> **do** <stmt>

while <arith-expr><compare-op><arith-expr> **do** <stmt>

while <var><compare-op><arith-expr> **do** <stmt>

while <var> $\leq$ <arith-expr> **do** <stmt>

while <var> $\leq$ <var> **do** <stmt>

while $x \leq$ <var> **do** <stmt>

while $x \leq y$ **do** <stmt>

while $x \leq y$ **do** <begin-stmt>

while x ≤ y do begin x := x+1 ; y := y-1 end

Rightmost & Leftmost Derivation

$S \Rightarrow [S] \mid SS \mid [\,]$ ...

[ ] [ ]

- $S \to \underline{S}S \to [S]S \to [[\,]]S \to [[\,]][\,]$

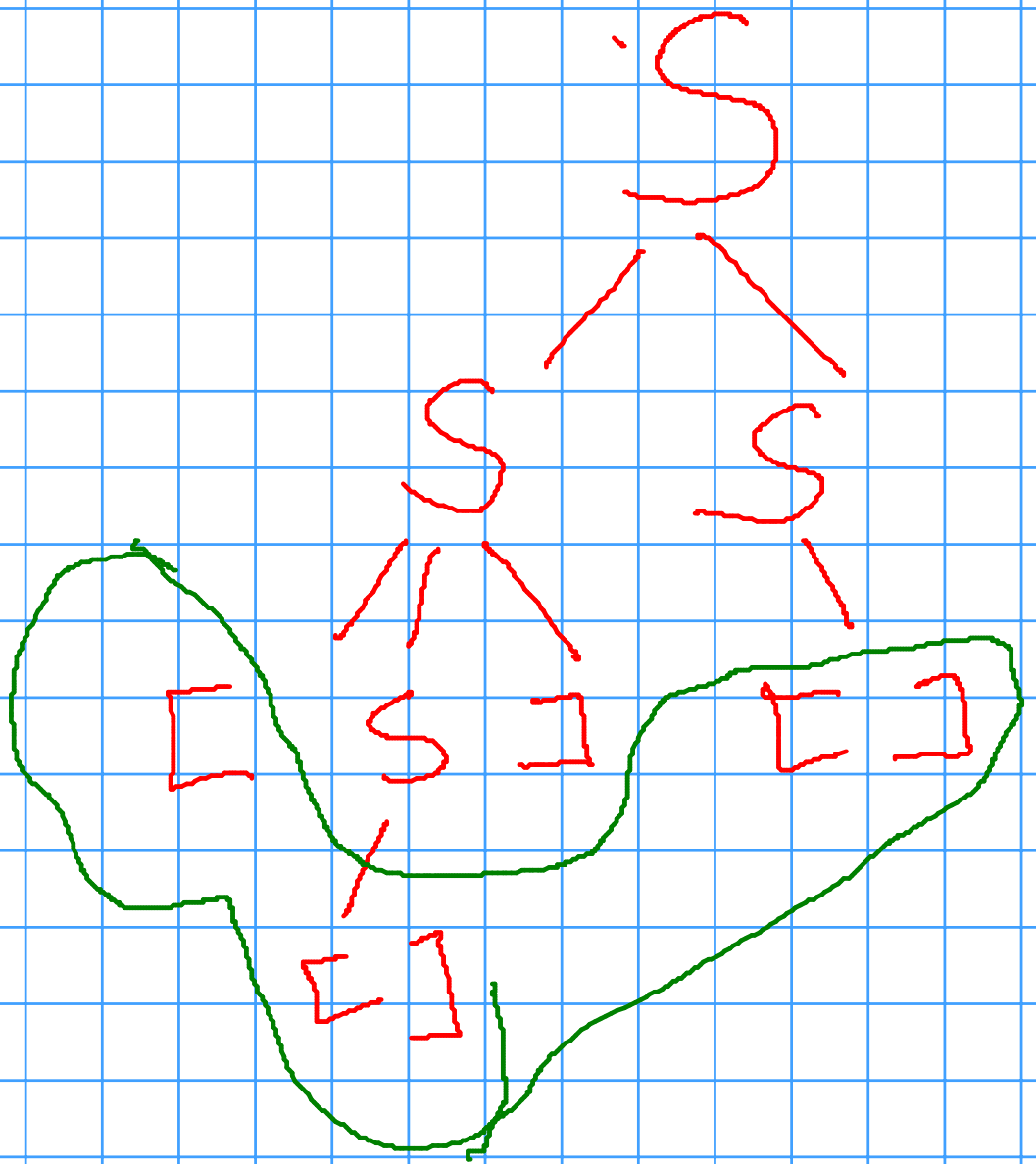
  Leftmost Derivation

- $S \to S\underline{S} \to S[\,] \to [S][\,] \to [[\,]][\,]$

  Rightmost Derivation

- $S \to S\underline{S} \to [S]S \to [S][\,] \to [[\,]][\,]$

Parse Tree



Context-Freeness Property.

- If $\gamma_1 \cdot \gamma_2 \cdots \gamma_k \xrightarrow{*} x$ then

$\exists x_1, \ldots, x_k$. $x = x_1 \cdots x_k$ and

$\forall i$. $\gamma_i \xrightarrow{*} x_i$

- If $Y_2 \to^* Y_2$ then $x_2 \to^* x_2$

$Y_1 Y_2 \cdots Y_k \to^* x_1 x_2 \cdots x_k$

$Y_1 \; Y_k \to^* x_1 Y_2 \; Y_k \to^* x_1 x_2 Y_3 \cdots Y_k$

$\to^* x_1 x_2 \; x_k$

SS

$S \overset{*}{\rightarrow} [S]S$

$S \rightarrow [\,]$

$SS \rightarrow [S]S$

Transforming Given Grammar

- Preserve language.

Example Useless symbol A ?

$S \xrightarrow{*} \alpha A \beta$.

- Remove all $A \to \alpha$ rules

Useless 2 $S \to \alpha A \beta \to \gamma$

Normal Forms for CFG

A CFG is in *Chomsky normal form* (CNF) if all productions are of the form

$$A \to BC \text{ or } A \to a,$$

where $A, B, C \in N$ and $a \in \Sigma$. A CFG is in *Greibach normal form* (GNF) if all productions are of the form

$$A \to aB_1 B_2 \cdots B_k$$

for some $k \geq 0$, where $A, B_1, \ldots, B_k \in N$ and $a \in \Sigma$. Note that $k = 0$ is allowed, giving productions of the form $A \to a$. □

For example, the two grammars

$$S \to AB \mid AC \mid SS, \qquad C \to SB, \qquad A \to [, \qquad B \to], \tag{21.1}$$

$$S \to [B \mid [SB \mid [BS \mid [SBS, \qquad B \to] \tag{21.2}$$

Conversion

Theorem

For any CFG G we can construct G' in CNF and G'' in GNF such that
$L(G') = L(G'') = L(G) - \{\epsilon\}$.

Conversion

Theorem

For any CFG G we can construct G' in CNF and G'' in GNF such that
$L(G') = L(G'') = L(G) - \{\epsilon\}$.

- Epsilon production $A \to \epsilon$ and unit production $A \to B$.

Conversion

Theorem

For any CFG G we can construct G' in CNF and G'' in GNF such that
$L(G') = L(G'') = L(G) - \{\epsilon\}$.

- Epsilon production $A \to \epsilon$ and unit production $A \to B$.
- Given G we construct G_1 without epsilon or unit productions with $L(G_1) = L(G) - \{\epsilon\}$

Conversion

Theorem

For any CFG G we can construct G' in CNF and G'' in GNF such that
$L(G') = L(G'') = L(G) - \{\epsilon\}$.

- Epsilon production $A \to \epsilon$ and unit production $A \to B$.
- Given G we construct G_1 without epsilon or unit productions with $L(G_1) = L(G) - \{\epsilon\}$
- We convert G_1 into language equivalent G' in CNF.

Conversion

Theorem

For any CFG G we can construct G' in CNF and G'' in GNF such that
$L(G') = L(G'') = L(G) - \{\epsilon\}$.

- Epsilon production $A \to \epsilon$ and unit production $A \to B$.
- Given G we construct G_1 without epsilon or unit productions with $L(G_1) = L(G) - \{\epsilon\}$
- We convert G_1 into language equivalent G' in CNF.
- We convert G' in CNF into language equivalent G'' in GNF.

Getting rid of epsilon and Unit Productions

Given $G = (N, \Sigma, P, S)$ Construct $\hat{P}$ by adding the following productions.

(a) if $A \to \alpha B \beta$ and $B \to \epsilon$ are in $\hat{P}$, then $A \to \alpha\beta$ is in $\hat{P}$; and

(b) if $A \to B$ and $B \to \gamma$ are in $\hat{P}$, then $A \to \gamma$ is in $\hat{P}$.

- Let $\hat{G} = (N, \Sigma, \hat{P}, S)$.
- Claim Let $S \xrightarrow{*}_{\hat{G}} x$ via shortest derivation, where $x \neq \epsilon$. Then the derivation does not use ϵ or unit productions.
- Remove ϵ and unit (and useless) productions from $\hat{P}$ to get required)grammar G_1.

$S \xrightarrow{*} \gamma B \delta \xrightarrow{1} \gamma \delta \xrightarrow{*} x$ $\qquad B \to \epsilon$

$S \xrightarrow{m} \eta A \theta \xrightarrow{1} \eta \alpha \beta \beta \theta \xrightarrow{n} \gamma B \delta \xrightarrow{1} \gamma \delta \xrightarrow{k} x$

$S \xrightarrow{m} \eta A \theta \xrightarrow{1} \eta \alpha \beta \theta \xrightarrow{n} \gamma \delta \xrightarrow{k} x$.

$S \xrightarrow{*} \alpha A \beta \xrightarrow{1} \alpha B \beta \xrightarrow{*} x$ $\qquad A \to B$

$S \xrightarrow{n} \alpha A \beta \xrightarrow{1} \alpha B \beta \xrightarrow{n} \eta B \theta \xrightarrow{1} \eta \gamma \theta \xrightarrow{k} x$

$S \xrightarrow{m} \alpha A \beta \to \alpha \gamma \beta \to \eta \gamma \theta \xrightarrow{k} x$

$B \to \gamma$

Example

Grammar $S \rightarrow aSa \mid bSb \mid T, \quad T \rightarrow a \mid b \mid \epsilon$.

Converting to CNF: An Example

Let G be $S \rightarrow aSb \mid \epsilon$.

Converting to CNF: An Example

Let G be $S \rightarrow aSb \mid \epsilon$.

- We get G_1 as $S \rightarrow aSb \mid ab$ by removing epsilon and unit productions.

Converting to CNF: An Example

Let G be $S \to aSb \mid \epsilon$.

- We get G_1 as $S \to aSb \mid ab$ by removing epsilon and unit productions.
- Replace terminals by fresh nonterminals.
  $S \to ASB \mid AB, \quad A \to a, \quad B \to b$.

Converting to CNF: An Example

Let G be $S \to aSb \mid \epsilon$.

- We get G_1 as $S \to aSb \mid ab$ by removing epsilon and unit productions.
- Replace terminals by fresh nonterminals.
  $S \to ASB \mid AB, \quad A \to a, \quad B \to b.$
- Replace long RHS to get G' in CNF.
  $S \to AC \mid AB, \quad C \to SB, \quad A \to a, \quad B \to b$

Converting to CNF: An Example

Let G be $S \to aSb \mid \epsilon$.

- We get G_1 as $S \to aSb \mid ab$ by removing epsilon and unit productions.
- Replace terminals by fresh nonterminals.
  $S \to ASB \mid AB, \quad A \to a, \quad B \to b.$
- Replace long RHS to get G' in CNF.
  $S \to AC \mid AB, \quad C \to SB, \quad A \to a, \quad B \to b$

It is easy to see that $L(G') = L(G_1)$.

Converting CNF to GNF

Omitted.

Some properties of CNF and GNF

- Every string in the language can be derived by left-most (or right-most) derivation.
- If $S \xrightarrow{n}_G x$ and $|x| = m$ then
  If G is CNF we have $n = ?$ (in terms of m)
  If G is GNF we have $n = ?$ (in terms of m)

non-term + 1 , each non-term = 1 term

$n = 2m - 1$

for GNF $\rightarrow$ $n = m$ (each step, $L++$)

CFL and non-CFL languages

We gave CFG for following languages to shonw that they are CFL

- $a^n b^n$
- PALINDROME $\{x \mid x = rev(x)\}$
- Balanced Paranthesis language PAREN

CFL and non-CFL languages

We gave CFG for following languages to shonw that they are CFL

- $a^n b^n$
- PALINDROME $\{x \mid x = rev(x)\}$
- Balanced Paranthesis language PAREN
- What about $a^n b^n c^n$?

CFL and non-CFL languages

We gave CFG for following languages to shonw that they are CFL

- $a^n b^n$
- PALINDROME $\{x \mid x = rev(x)\}$
- Balanced Paranthesis language PAREN
- What about $a^n b^n c^n$?
- This language is not context free. How to prove this?

CFL and non-CFL languages

We gave CFG for following languages to shonw that they are CFL

- $a^n b^n$
- PALINDROME $\{x \mid x = rev(x)\}$
- Balanced Paranthesis language PAREN
- What about $a^n b^n c^n$?
- This language is not context free. How to prove this?
- Answer: Pumping Lemma for Context Free Languages.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.
- If x is long (i.e. $|x| \geq 2^{n+1}$) then T_x has depth at least $n+1$.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.
- If x is long (i.e. $|x| \geq 2^{n+1}$) then T_x has depth at least $n+1$.
- There is a long path with at least $n+1$ internal nodes from root to leaf.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.
- If x is long (i.e. $|x| \geq 2^{n+1}$) then T_x has depth at least $n+1$.
- There is a long path with at least $n+1$ internal nodes from root to leaf.
- Let n be the number of NONTERMINALS in G.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.
- If x is long (i.e. $|x| \geq 2^{n+1}$) then T_x has depth at least $n+1$.
- There is a long path with at least $n+1$ internal nodes from root to leaf.
- Let n be the number of NONTERMINALS in G.
- Hence some non-terminal repeats on the long path.

Intuition

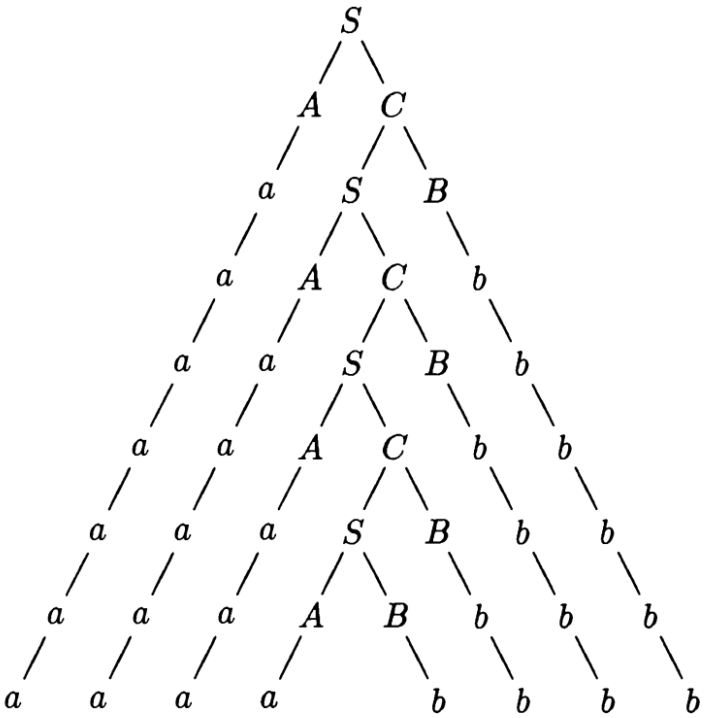
- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.
- If x is long (i.e. $|x| \geq 2^{n+1}$) then T_x has depth at least $n+1$.
- There is a long path with at least $n+1$ internal nodes from root to leaf.
- Let n be the number of NONTERMINALS in G.
- Hence some non-terminal repeats on the long path.
- Consider lowest (closest to leaf) pair of repeating non-terminals on the long path.

Intuition

- Suppose L is CFL. Hence a CNF grammar G generates it.
- Consider a parse tree T_x for a word x obtained using G.
- If x is long (i.e. $|x| \geq 2^{n+1}$) then T_x has depth at least $n+1$.
- There is a long path with at least $n+1$ internal nodes from root to leaf.
- Let n be the number of NONTERMINALS in G.
- Hence some non-terminal repeats on the long path.
- Consider lowest (closest to leaf) pair of repeating non-terminals on the long path.
- This can be pumped arbitrary i times.

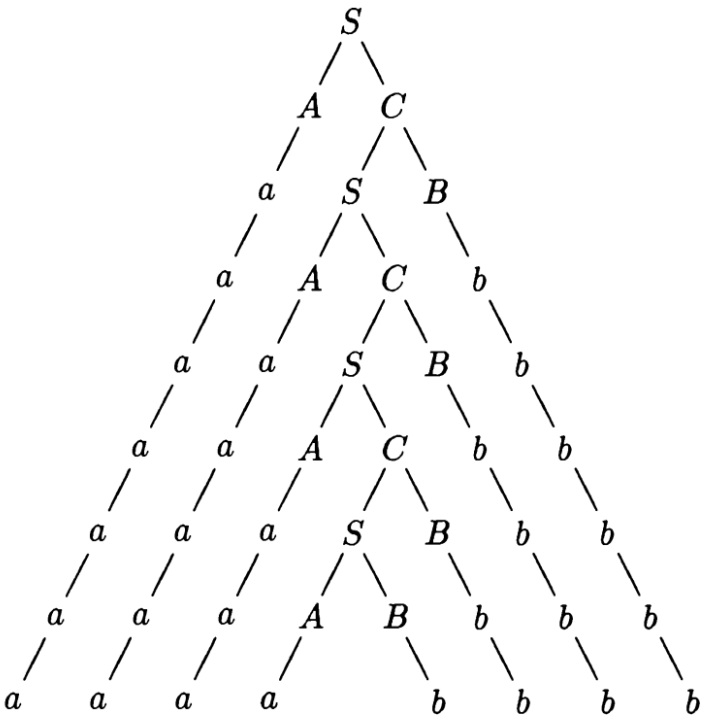
Parse Tree of CNF Grammer Derivation

Let G be $S \to AC \mid AB, \quad C \to SB, \quad A \to a, \quad B \to b.$



Parse Tree of CNF Grammer Derivation

Let G be $S \to AC \mid AB, \quad C \to SB, \quad A \to a, \quad B \to b.$

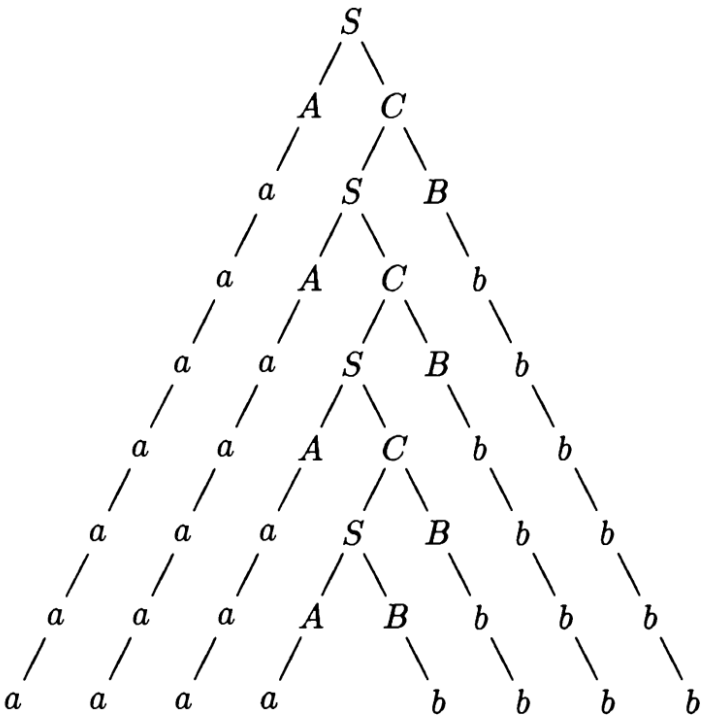


Leftmost Derivation

$$S \to AC \to aC \to aSB \to aACB \to aaCB \to aaSBB \to aaACBB$$
$$\to aaaCBB \to aaaSBBB \to aaaABBBB \to aaaaBBBB$$
$$\to aaaabBBB \to aaaabbBB \to aaaabbbB \to aaaabbbb,$$

Parse Tree of CNF Grammer Derivation

Let G be $S \to AC \mid AB, \quad C \to SB, \quad A \to a, \quad B \to b.$

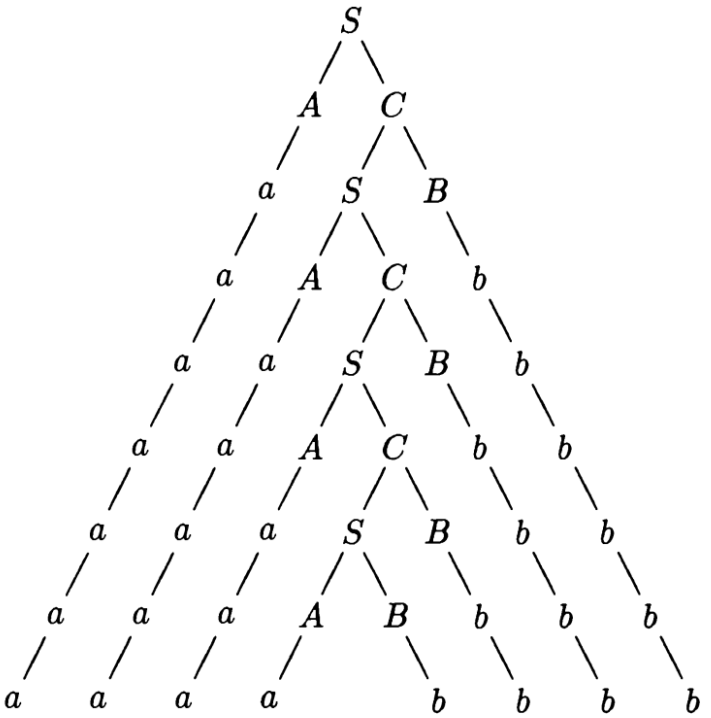


Rightmost Derivation

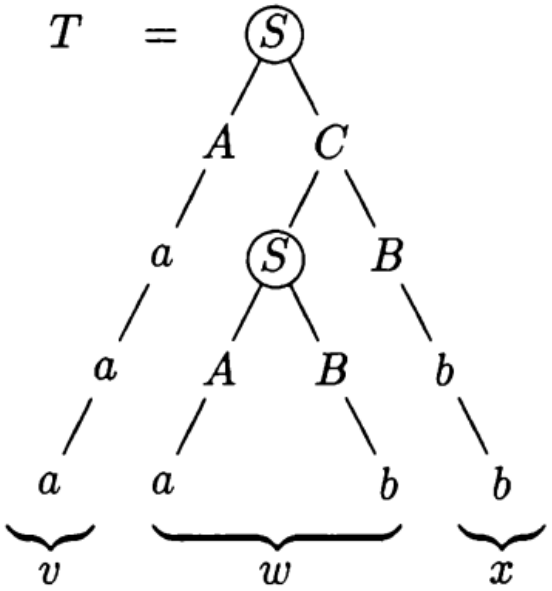
$$S \to AC \to ASB \to ASb \to AACb \to AASBb \to AASbb \to AAACbb$$
$$\to AAASBbb \to AAASbbb \to AAAABbbb \to AAAAbbbb$$
$$\to AAAabbbb \to AAaabbbb \to Aaaabbbb \to aaaabbbb,$$

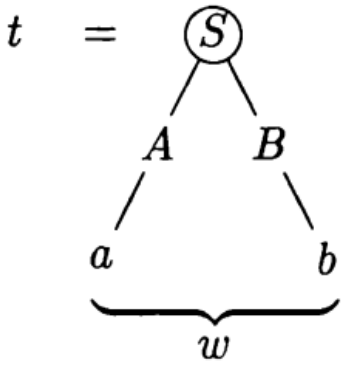
Parse Tree of CNF Grammer Derivation

Let G be $S \to AC \mid AB, \quad C \to SB, \quad A \to a, \quad B \to b.$

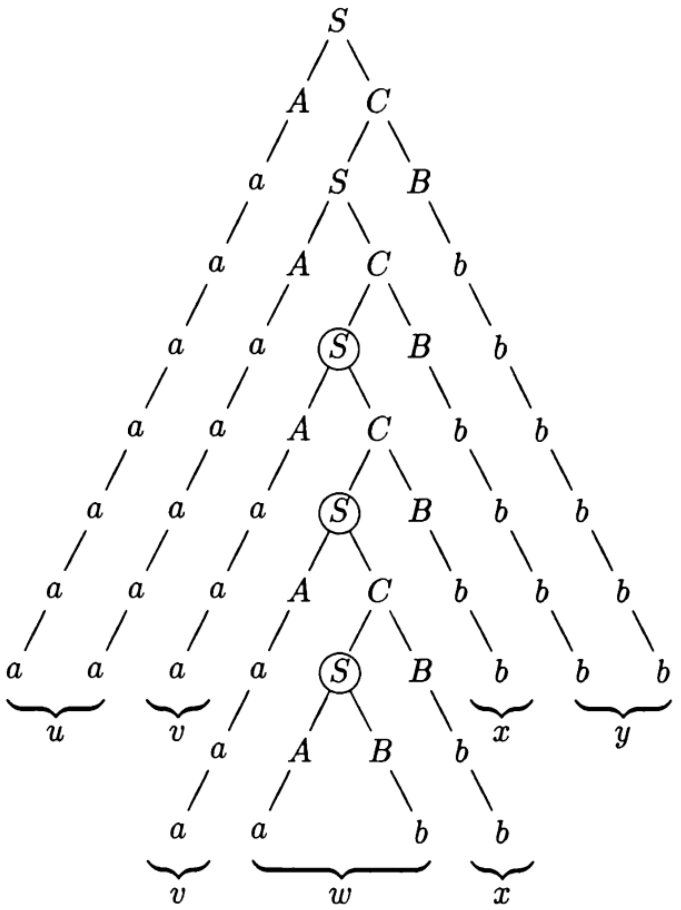


Subtrees for lowermost pair of repeating Nonterminals





Subtrees for lowermost pair of repeating Nonterminals



Pumping Lemma for CFL

Lemma If A is CFL then

$\exists k > 0$ such that

$\forall z \in A$ with $|z| \geq k$,

$\exists u, v, w, x, Y$ with $z = u \cdot v \cdot w \cdot x \cdot y$ and

$|vwx| \leq k$ and $|vx| > 0$ s.t.

$u \cdot (v^i) \cdot w \cdot (x^i) \cdot y \in A$ for all $i \geq 0$.

Pumping Lemma in Contrapositive Form

Game with demon to prove that Language A is not CFL

- Demon chooses arbitrary $k > 0$.
- You choose $z \in A$ with $|z| \geq k$,
- Demon chooses u, v, w, x, Y with $z = u \cdot v \cdot w \cdot x \cdot y$ and $|vwx| \leq k$ and $|vx| > 0$
- You choose $i \geq 0$ and show that $u \cdot (v^i) \cdot w \cdot (x^i) \cdot y \notin A$ to win the game.

You give a winning strategy for the above game to show that A is not CFL.

example

$a^n b^n c^n$ is not CFL. Proof using pumping lemma.

- Demon chooses $k > 0$.
- Chooze $z = a^k b^k c^k$ which is in A.
- Demon splits z $u \cdot v \cdot w \cdot x \cdot y$ with conditions. Some Cases:
  1. $u = \epsilon$ and $vwx = a^k$ and $y = b^k c^k$.
     You choose $i = 2$ which given $v^2 w x^2 = a^j$ with $j > k$. Hence, $u \cdot (v^2) \cdot w \cdot (x^2) \cdot y \notin A$.
  2. Demon chooses vwx within a^k part. Choosing $i = 2$ and pumping up, increases count of a without changing counts of b and c. Hence pumped word is not in A.
  3. Demon chooses $vwx = a^j b^l$. You choose $i = 2$. Pumping up increases the count of a + count of b without changing the count of c. Hence the pumped word is not in A.
  4. The Remaining cases are similar.

example

Let $A = \{w \cdot w \mid w \in \{a, b\}^*\}$. Prove that A is not CFL.

Closure Properties

Theorem

If A, B are any CFL languages then

- $A \cup B$ *is CFL*
- $rev(A)$ *is CFL*
- $A \cdot B$ *is CFL*
- A^* *is CFL*

- h is homomorphism from $\Sigma^* \to \Delta^*$ then $h(A)$ is CFL
- $h^{-1}(A)$ is CFL

Closure Property Constructions

Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, , P_2, S_2)$ with $N_1 \cap N_2 = \emptyset$.

Closure Property Constructions

Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, , P_2, S_2)$ with $N_1 \cap N_2 = \emptyset$.

- Let $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P, S)$ with $P = \{S \rightarrow S_1 \mid S_2\} \cup P_1 \cup P_2$.
  Then, $L(G) = L(G_1) \cup L(G_2)$. How?

Closure Property Constructions

Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, , P_2, S_2)$ with $N_1 \cap N_2 = \emptyset$.

- Let $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P, S)$ with $P = \{S \to S_1 \mid S_2\} \cup P_1 \cup P_2$.
  Then, $(G) = L(G_1) \cup L(G_2)$. How?
- Let $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P, S)$ with $P = \{S \to S_1 S_2\} \cup P_1 \cup P_2$.
  Then, $L(G) = L(G_1) \cdot L(G_2)$.
- $L(G) = (L(G_1))^*$

  $S \to \epsilon \mid S S_1$

$L(G) = L(G_1)^R$

$A \to \alpha \in G_1$

$\alpha \in \Sigma^*$

$A \to \alpha^R$

$S \to aSb \mid \epsilon$

$a^n b^n$

$S \to bSa \mid \epsilon$

$b^n a^n$

Closure Property Constructions

Let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, , P_2, S_2)$ with $N_1 \cap N_2 = \emptyset$.

- Let $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P, S)$ with $P = \{S \rightarrow S_1 \mid S_2\} \cup P_1 \cup P_2$.
  Then, $(G) = L(G_1) \cup L(G_2)$. How?
- Let $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P, S)$ with $P = \{S \rightarrow S_1 S_2\} \cup P_1 \cup P_2$.
  Then, $L(G) = L(G_1) \cdot L(G_2)$.
- Let $G = (N_1 \cup \{S\}, \Sigma, P, S)$ with $P = \{S \rightarrow S_1 S \mid \epsilon\} \cup P_1$.
  Then, $L(G) = (L(G_1))^*$.

CFL are not closed under $\cap$

If A, B are CFL langauges then Intersection $A \cap B$ need not be CFL.

CFL are not closed under $\cap$

If A, B are CFL langauges then Intersection $A \cap B$ need not be CFL.

- How to show non-closure?

CFL are not closed under $\cap$

If A, B are CFL langauges then Intersection $A \cap B$ need not be CFL.

- How to show non-closure?
- Give a counter example. Choose CFL A, B s.t. $A \cap B$ is not CSL.

CFL are not closed under $\cap$

If A, B are CFL langauges then Intersection $A \cap B$ need not be CFL.

- How to show non-closure?
- Give a counter example. Choose CFL A, B s.t. $A \cap B$ is not CSL.

Let $A = \{a^n b^n c^m \mid m, n \geq 0\}$, and $B = \{a^m b^n c^n \mid m, n \geq 0\}$

CFL are not closed under $\cap$

If A, B are CFL langauges then Intersection $A \cap B$ need not be CFL.

- How to show non-closure?
- Give a counter example. Choose CFL A, B s.t. $A \cap B$ is not CSL.

Let $A = \{a^n b^n c^m \mid m, n \geq 0\}$, and $B = \{a^m b^n c^n \mid m, n \geq 0\}$

- Show that A as well as B are CFL.

CFL are not closed under $\cap$

If A, B are CFL langauges then Intersection $A \cap B$ need not be CFL.

- How to show non-closure?
- Give a counter example. Choose CFL A, B s.t. $A \cap B$ is not CSL.

Let $A = \{a^n b^n c^m \mid m, n \geq 0\}$, and $B = \{a^m b^n c^n \mid m, n \geq 0\}$

- Show that A as well as B are CFL.
- $A \cap B = \{a^n b^n c^n \mid n \geq 0\}$. Show that this is not CFL.

CFL are not closed under complement $\sim$

Proof by contradiction

- Assume to contrary that for every CFL A the language $\sim A$ is also CFL.

CFL are not closed under complement $\sim$

Proof by contradiction

- Assume to contrary that for every CFL A the language $\sim A$ is also CFL.
- Let A, B be arbitrary CFL.

CFL are not closed under complement $\sim$

Proof by contradiction

- Assume to contrary that for every CFL A the language $\sim A$ is also CFL.
- Let A, B be arbitrary CFL.
- By set theory $A \cap B = \sim((\sim A) \cup (\sim B))$.

CFL are not closed under complement $\sim$

Proof by contradiction

- Assume to contrary that for every CFL A the language $\sim A$ is also CFL.
- Let A, B be arbitrary CFL.
- By set theory $A \cap B = \sim((\sim A) \cup (\sim B))$.
- Using closure under $\cup$ and $\sim$, then RHS is also CFL.

CFL are not closed under complement $\sim$

Proof by contradiction

- Assume to contrary that for every CFL A the language $\sim A$ is also CFL.
- Let A, B be arbitrary CFL.
- By set theory $A \cap B \;=\; \sim((\sim A) \cup (\sim B))$.
- Using closure under $\cup$ and $\sim$, then RHS is also CFL.
- This contradicts that CFL is not closed under $\cap$.

Decision Problems

Given CFG G and $x \in \Sigma^*$,

- We can check by an algorithm whether $x \in L(G)$.
  CYK algorithm (omitted).

Decision Problems

Given CFG G and $x \in \Sigma^*$,

- We can check by an algorithm whether $x \in L(G)$.
  CYK algorithm (omitted).
- Parsing: Constructing parse tree for given word x under CFG G.
  Widely used in compiler design.
  Specialized grammars such as LALR, SLR or LR(k) have efficient parsing algorithms. Aho, Ullman: Turing Award 2020.

Decision Problems

Given CFG G and $x \in \Sigma^*$,

- We can check by an algorithm whether $x \in L(G)$.
  CYK algorithm (omitted).
- Parsing: Constructing parse tree for given word x under CFG G.
  Widely used in compiler design.
  Specialized grammars such as LALR, SLR or LR(k) have efficient parsing algorithms. Aho, Ullman: Turing Award 2020.
- We can check by an algorithm whether $L(G) = \emptyset$.
  Emptiness of CFG is decidable.

1° Given G_1 and G_2 $L(G_1) \subseteq L(G_2)$? Inclusion

Decision Problems

Given CFG G and $x \in \Sigma^*$,

- We can check by an algorithm whether $x \in L(G)$.
  CYK algorithm (omitted).
- Parsing: Constructing parse tree for given word x under CFG G.
  Widely used in compiler design.
  Specialized grammars such as LALR, SLR or LR(k) have efficient parsing algorithms. Aho, Ullman: Turing Award 2020.
- We can check by an algorithm whether $L(G) = \emptyset$.
  Emptiness of CFG is decidable.
- There is no algorithm to decide whether $L(G) = \Sigma^*$.
  Universality of CFG is undecidable.