Outline

Why study Context-Free Grammars?

- Arise naturally in syntax of programming languages, parsing, compiling.
- Characterize languages accepted by Pushdown automata.
- Pushdown automata are an important class of system models:
  - They can model programs with procedure calls
  - Can model other infinite-state systems.
- Easier to prove properties of Pushdown languages using CFG's:
  - Pumping lemma
  - Ultimate periodicity
  - PDA = PDA without ϵ-transitions.
- Parsing algo leads to solution to "CFL reachability" problem: Given a finite A-labelled graph, a CFG G, are two given vertices u and v connected by a path whose label is in $L(G)$.

Context-Free Grammars: Example 1

CFG G_1

$$
\begin{array}{rcl}
S & \rightarrow & aX \\
X & \rightarrow & aX \\
X & \rightarrow & bX \\
X & \rightarrow & b
\end{array}
$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.

Example derivation:

$$S$$

Context-Free Grammars: Example 1

CFG G_1

$$\begin{aligned} S &\rightarrow aX \\ X &\rightarrow aX \\ X &\rightarrow bX \\ X &\rightarrow b \end{aligned}$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.

Example derivation:

$$S \Rightarrow aX$$

Context-Free Grammars: Example 1

CFG G_1

$$
\begin{array}{rcl}
S & \rightarrow & aX \\
X & \rightarrow & aX \\
X & \rightarrow & bX \\
X & \rightarrow & b
\end{array}
$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.

Example derivation:

$$S \quad \Rightarrow aX \quad \Rightarrow abX$$

Context-Free Grammars: Example 1

CFG G_1

$$
\begin{array}{rcl}
S & \rightarrow & aX \\
X & \rightarrow & aX \\
X & \rightarrow & bX \\
X & \rightarrow & b
\end{array}
$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.
Example derivation:

$$S \quad \Rightarrow aX \quad \Rightarrow abX \quad \Rightarrow abb.$$

Context-Free Grammars: Example 1

CFG G_1

$$\begin{array}{rcl} S & \rightarrow & aX \\ X & \rightarrow & aX \\ X & \rightarrow & bX \\ X & \rightarrow & b \end{array}$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.

Example derivation:

$$S \quad \Rightarrow aX \quad \Rightarrow abX \quad \Rightarrow abb.$$

Language defined by G, written $L(G)$, is the set of all terminal strings that can be generated by G.

Context-Free Grammars: Example 1

CFG G_1

$$\begin{array}{rcl} S & \rightarrow & aX \\ X & \rightarrow & aX \\ X & \rightarrow & bX \\ X & \rightarrow & b \end{array}$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.

Example derivation:

$$S \quad \Rightarrow aX \quad \Rightarrow abX \quad \Rightarrow abb.$$

Language defined by G, written $L(G)$, is the set of all terminal strings that can be generated by G.

What is language defined by G_1 above?

Context-Free Grammars: Example 1

CFG G_1

$$\begin{array}{rcl} S & \rightarrow & aX \\ X & \rightarrow & aX \\ X & \rightarrow & bX \\ X & \rightarrow & b \end{array}$$

Derivation of a string: Begin with S and keep rewriting the current string by replacing a non-terminal by its RHS in a production of the grammar.
Example derivation:

$$S \quad \Rightarrow aX \quad \Rightarrow abX \quad \Rightarrow abb.$$

Language defined by G, written $L(G)$, is the set of all terminal strings that can be generated by G.
What is language defined by G_1 above? $a(a+b)^*b$.

Context-Free Grammars: Example 2

CFG G_2

$$
\begin{array}{rcl}
S & \longrightarrow & aSb \\
S & \longrightarrow & \epsilon.
\end{array}
$$

Example derivation:

$$S$$

Context-Free Grammars: Example 2

CFG G_2

$$
\begin{aligned}
S &\longrightarrow aSb \\
S &\longrightarrow \epsilon.
\end{aligned}
$$

Example derivation:

$$S \quad \Rightarrow aSb$$

Context-Free Grammars: Example 2

CFG G_2

$$\begin{array}{rcl} S & \longrightarrow & aSb \\ S & \longrightarrow & \epsilon. \end{array}$$

Example derivation:

$$S \quad \Rightarrow aSb \quad \Rightarrow aaSbb$$

Context-Free Grammars: Example 2

CFG G_2

$$
\begin{aligned}
S &\longrightarrow aSb \\
S &\longrightarrow \epsilon.
\end{aligned}
$$

Example derivation:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb.$$

Context-Free Grammars: Example 2

CFG G_2

$$\begin{array}{lll} S & \rightarrow & aSb \\ S & \rightarrow & \epsilon. \end{array}$$

Example derivation:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb.$$

What is language defined by G_2 above?

Context-Free Grammars: Example 2

CFG G_2

$$\begin{aligned} S &\longrightarrow aSb \\ S &\longrightarrow \epsilon. \end{aligned}$$

Example derivation:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb.$$

What is language defined by G_2 above? $\{a^n b^n \mid n \geq 0\}$.

Context-Free Grammars: Example 3

CFG G_3

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

Example derivation:

$$S$$

Context-Free Grammars: Example 3

CFG G_3

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

Example derivation:

$$S \quad \Rightarrow aSa$$

Context-Free Grammars: Example 3

CFG G_3

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

Example derivation:

$$S \quad \Rightarrow aSa \quad \Rightarrow abSba$$

Context-Free Grammars: Example 3

CFG G_3

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

Example derivation:

$$S \quad \Rightarrow aSa \quad \Rightarrow abSba \quad \Rightarrow abbSbba \quad \Rightarrow abbbba.$$

Context-Free Grammars: Example 3

CFG G_3

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

Example derivation:

$$S \quad \Rightarrow aSa \quad \Rightarrow abSba \quad \Rightarrow abbSbba \quad \Rightarrow abbbba.$$

What is language defined by G_3 above?

Context-Free Grammars: Example 3

CFG G_3

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

Example derivation:

$$S \quad \Rightarrow aSa \quad \Rightarrow abSba \quad \Rightarrow abbSbba \quad \Rightarrow abbbba.$$

What is language defined by G_3 above? Palindromes: $\{w \in \{a, b\}^* \mid w = w^R\}$.

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

Context-Free Grammars: Example 4

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

S

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$S \quad \Rightarrow (S)$$

Context-Free Grammars: Example 4

CFG G_4

$$S \;\rightarrow\; (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S \quad & \Rightarrow (S) \\ & \Rightarrow (SS) \\ & \Rightarrow (SSS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad\longrightarrow\quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \to \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \to \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \\ &\Rightarrow ((()S)SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \;\rightarrow\; (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \\ &\Rightarrow ((()S)SS) \\ &\Rightarrow ((()(S))SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \\ &\Rightarrow ((()S)SS) \\ &\Rightarrow ((()(S))SS) \\ &\Rightarrow ((()())SS) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned}
S &\Rightarrow (S) \\
&\Rightarrow (SS) \\
&\Rightarrow (SSS) \\
&\Rightarrow ((S)SS) \\
&\Rightarrow ((SS)SS) \\
&\Rightarrow (((S)S)SS) \\
&\Rightarrow ((()S)SS) \\
&\Rightarrow ((()(S))SS) \\
&\Rightarrow ((()())SS) \\
&\Rightarrow ((()())(S)S)
\end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \quad \rightarrow \quad (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S \quad & \Rightarrow (S) \\ & \Rightarrow (SS) \\ & \Rightarrow (SSS) \\ & \Rightarrow ((S)SS) \\ & \Rightarrow ((SS)SS) \\ & \Rightarrow (((S)S)SS) \\ & \Rightarrow ((()S)SS) \\ & \Rightarrow ((()(S))SS) \\ & \Rightarrow ((()())SS) \\ & \Rightarrow ((()())(S)S) \\ & \Rightarrow ((()())()S) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned}
S &\Rightarrow (S) \\
&\Rightarrow (SS) \\
&\Rightarrow (SSS) \\
&\Rightarrow ((S)SS) \\
&\Rightarrow ((SS)SS) \\
&\Rightarrow (((S)S)SS) \\
&\Rightarrow ((()S)SS) \\
&\Rightarrow ((()(S))SS) \\
&\Rightarrow ((()())SS) \\
&\Rightarrow ((()())(S)S) \\
&\Rightarrow ((()())()S) \\
&\Rightarrow ((()())()(S))
\end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \\ &\Rightarrow ((()S)SS) \\ &\Rightarrow ((()(S))SS) \\ &\Rightarrow ((()())SS) \\ &\Rightarrow ((()())(S)S) \\ &\Rightarrow ((()())()S) \\ &\Rightarrow ((()())()(S)) \\ &\Rightarrow ((()())()()) \end{aligned}$$

Context-Free Grammars: Example 4

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \\ &\Rightarrow ((()S)SS) \\ &\Rightarrow ((()(S))SS) \\ &\Rightarrow ((()())SS) \\ &\Rightarrow ((()())(S)S) \\ &\Rightarrow ((()())()S) \\ &\Rightarrow ((()())()(S)) \\ &\Rightarrow ((()())()()) \end{aligned}$$

What is language defined by G_4 above?

Context-Free Grammars: Example 4

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Exercise: Derive "((()())()())".

$$\begin{aligned} S &\Rightarrow (S) \\ &\Rightarrow (SS) \\ &\Rightarrow (SSS) \\ &\Rightarrow ((S)SS) \\ &\Rightarrow ((SS)SS) \\ &\Rightarrow (((S)S)SS) \\ &\Rightarrow ((()S)SS) \\ &\Rightarrow ((()(S))SS) \\ &\Rightarrow ((()())SS) \\ &\Rightarrow ((()())(S)S) \\ &\Rightarrow ((()())()S) \\ &\Rightarrow ((()())()(S)) \\ &\Rightarrow ((()())()()) \end{aligned}$$

What is language defined by G_4 above? Balanced Parenthesis.

CFG's more formally

A Context-Free Grammar (CFG) is of the form

$$G = (N, A, S, P)$$

where

- N is a finite set of non-terminal symbols
- A is a finite set of terminal symbols.
- $S \in N$ is the start non-terminal symbol.
- $P \subseteq N \times (N \cup A)^*$ is the set of productions or rules. Productions are written $X \to \alpha$.

Derivations, language etc.

- "α derives β in 0 or more steps": $\alpha \Rightarrow_G^* \beta$.
- First define $\alpha \stackrel{n}{\Rightarrow} \beta$ inductively:
  - $\alpha \stackrel{1}{\Rightarrow} \beta$ iff α is of the form $\alpha_1 X \alpha_2$ and $X \rightarrow \gamma$ is a production in P, and $\beta = \alpha_1 \gamma \alpha_2$.
  - $\alpha \stackrel{n+1}{\Rightarrow} \beta$ iff there exists γ such that $\alpha \stackrel{n}{\Rightarrow} \gamma$ and $\gamma \stackrel{1}{\Rightarrow} \beta$.
- Sentential form of G: any $\alpha \in (N \cup A)^*$ such that $S \Rightarrow_G^* \alpha$.
- Language defined by G:

$$\{w \in A^* \mid S \Rightarrow_G^* w\}.$$

- $L \subseteq A^*$ is called a Context-Free Language (CFL) if there is a CFG G such that $L = L(G)$.

Proving that a CFG accepts a certain language

CFG G_1

$$\begin{array}{rcl} S & \rightarrow & aX \\ X & \rightarrow & aX \\ X & \rightarrow & bX \\ X & \rightarrow & b \end{array}$$

Prove that $L(G_1) = a(a+b)^*b$.

Proving that a CFG accepts a certain language

CFG G_2

$$\begin{array}{lcl} S & \rightarrow & aSb \\ S & \rightarrow & \epsilon. \end{array}$$

Prove that $L(G_2) = \{a^n b^n \mid n \geq 0\}$.

Outline

Chomsky Normal Form

A Context-Free Grammar G is in Chomsky Normal Form if all productions are of the form

$$\begin{array}{rcl} X & \rightarrow & YZ \text{ or} \\ X & \rightarrow & a \end{array}$$

Its a "normal form" in the sense that

CNF

Every CFG G can be converted to a CFG G' in Chomsky Normal Form, with $L(G') = L(G) - \{\epsilon\}$.

Why is CNF useful?

- Gives us a way to do parsing: Given CFG G and $w \in A^*$, does $w \in L(G)$?

Why is CNF useful?

- Gives us a way to do parsing: Given CFG G and $w \in A^*$, does $w \in L(G)$?
    - If G is in CNF, then length of derivation of w (if one exists) can be bounded by $2|w|$.

Why is CNF useful?

- Gives us a way to do parsing: Given CFG G and $w \in A^*$, does $w \in L(G)$?
  - If G is in CNF, then length of derivation of w (if one exists) can be bounded by $2|w|$.
- Makes proofs of properties of CFG's simpler.

Example

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

"Equivalent" grammar in CNF:

CFG G_4' in CNF

$$\begin{aligned} S &\rightarrow LX \mid SS \mid LR \\ X &\rightarrow SR \\ L &\rightarrow (\\ R &\rightarrow) \end{aligned}$$

Procedure to convert a CFG to CNF

- Main problem is “unit” productions of the form $A \longrightarrow B$ and ϵ-productions of the form $B \longrightarrow \epsilon$.
- Once these productions are eliminated, converting to CNF is easy.

Procedure to remove unit and ϵ-productions

Given a CFG $G = (N, A, S, P)$.

- Repeatedly add productions according to the steps below till no more new productions can be added.
    1. If $A \rightarrow \alpha B \beta$ and $B \rightarrow \epsilon$ then add the production $A \rightarrow \alpha\beta$.
    2. If $A \rightarrow B$ and $B \rightarrow \gamma$ then add the production $A \rightarrow \gamma$.
- Let resulting grammar be $G' = (N, A, S, P')$.
- Let G'' be grammar (N, A, S, P''), where P'' is obtained from P' by dropping unit- and ϵ-productions.
- Return G'.

Example

Apply procedure to the grammar below:

CFG G_4

$$S \rightarrow (S) \mid SS \mid \epsilon.$$

Correctness claims

- Algorithm terminates

Correctness claims

- Algorithm terminates
  - Notice that each new production added has a RHS that is a subsequence of RHS an original production in P.
- G' generates same language as G.
  - Let G'_i be grammar obtained after i-th step, with $G'_0 = G$.
  - Then clearly $L(G'_{i+1}) = L(G'_i)$.

Correctness of G''

Claim

$L(G'') = L(G) - \{\epsilon\}$.

Subclaim

Let $w \in L(G')$ with $w \neq \epsilon$. Then any minimal-length derivation of w in G' does not use unit or ϵ-productions.

Proof of Subclaim

Subclaim

Let $w \in L(G'')$ with $w \neq \epsilon$. Then any minimal-length derivation of w in G' does not use unit or ϵ-productions.

Consider a derivation of w in G' which uses a production $B \longrightarrow \epsilon$. It must be of the form

$$S \overset{l}{\Rightarrow} \alpha X \beta \overset{1}{\Rightarrow} \alpha\gamma B\delta\beta \overset{m}{\Rightarrow} \alpha'\gamma' B\delta'\beta' \overset{1}{\Rightarrow} \alpha'\gamma'\delta'\beta' \overset{n}{\Rightarrow} w.$$

Proof of Subclaim

Subclaim

Let $w \in L(G'')$ with $w \neq \epsilon$. Then any minimal-length derivation of w in G' does not use unit or ϵ-productions.

Consider a derivation of w in G' which uses a production $B \longrightarrow \epsilon$. It must be of the form

$$\begin{array}{lllllllllll} S & \stackrel{l}{\Rightarrow} & \alpha X \beta & \stackrel{1}{\Rightarrow} & \alpha\gamma B\delta\beta & \stackrel{m}{\Rightarrow} & \alpha'\gamma' B\delta'\beta' & \stackrel{1}{\Rightarrow} & \alpha'\gamma'\delta'\beta' & \stackrel{n}{\Rightarrow} & w. \\ S & \stackrel{l}{\Rightarrow} & \alpha X \beta & \stackrel{1}{\Rightarrow} & \alpha\gamma\delta\beta & \stackrel{m}{\Rightarrow} & \alpha'\gamma'\delta'\beta' & & & \stackrel{n}{\Rightarrow} & w. \end{array}$$

Now consider a derivation of w in G' which uses a production $A \longrightarrow B$. It must be of the form

$$S \stackrel{l}{\Rightarrow} \alpha A \beta \stackrel{m}{\Rightarrow} \alpha' A \beta' \stackrel{1}{\Rightarrow} \alpha' B \beta' \stackrel{n}{\Rightarrow} \alpha'' B \beta'' \stackrel{1}{\Rightarrow} \alpha''\gamma\beta'' \stackrel{p}{\Rightarrow} w.$$

Proof of Subclaim

Subclaim

Let $w \in L(G'')$ with $w \neq \epsilon$. Then any minimal-length derivation of w in G' does not use unit or ϵ-productions.

Consider a derivation of w in G' which uses a production $B \to \epsilon$. It must be of the form

$$\begin{array}{lllllllllll} S & \stackrel{l}{\Rightarrow} & \alpha X \beta & \stackrel{1}{\Rightarrow} & \alpha\gamma B \delta\beta & \stackrel{m}{\Rightarrow} & \alpha'\gamma' B \delta'\beta' & \stackrel{1}{\Rightarrow} & \alpha'\gamma'\delta'\beta' & \stackrel{n}{\Rightarrow} & w. \\ S & \stackrel{l}{\Rightarrow} & \alpha X \beta & \stackrel{1}{\Rightarrow} & \alpha\gamma\delta\beta & \stackrel{m}{\Rightarrow} & \alpha'\gamma'\delta'\beta' & & & \stackrel{n}{\Rightarrow} & w. \end{array}$$

Now consider a derivation of w in G' which uses a production $A \to B$. It must be of the form

$$\begin{array}{lllllllllllll} S & \stackrel{l}{\Rightarrow} & \alpha A \beta & \stackrel{m}{\Rightarrow} & \alpha' A \beta' & \stackrel{1}{\Rightarrow} & \alpha' B \beta' & \stackrel{n}{\Rightarrow} & \alpha'' B \beta'' & \stackrel{1}{\Rightarrow} & \alpha''\gamma\beta'' & \stackrel{p}{\Rightarrow} & w. \\ S & \stackrel{l}{\Rightarrow} & \alpha A \beta & \stackrel{m}{\Rightarrow} & \alpha' A \beta' & \stackrel{1}{\Rightarrow} & \alpha'\gamma\beta' & \stackrel{n}{\Rightarrow} & \alpha''\gamma\beta'' & & & \stackrel{p}{\Rightarrow} & w. \end{array}$$

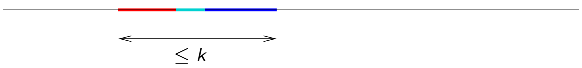
Outline

Pumping Lemma for CFL's

Pumping Lemma

For every CFL L there is a constant $k \geq 0$ such that for any word z in L of length at least k, there are strings u, v, w, x, y such that

- $z = uvwxy$,
- $vx \neq \epsilon$,
- $|vwx| \leq k$, and
- for each $i \geq 0$, the string uv^iwx^iy belongs to L.



Parse trees for CFG's

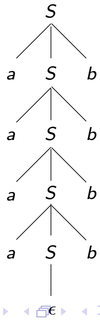
Derivations can be represented as parse trees:

CFG G_2

$$
\begin{aligned}
S &\rightarrow aSb\\
S &\rightarrow \epsilon.
\end{aligned}
$$

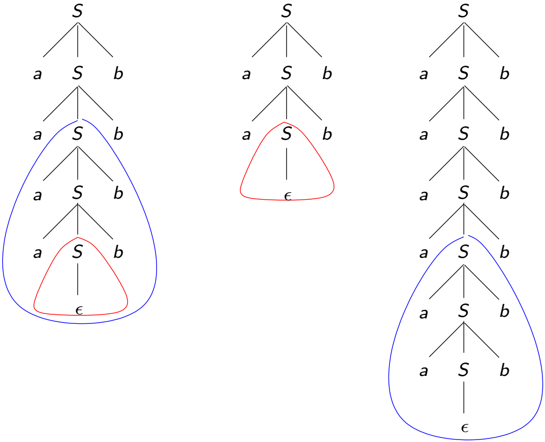
Example derivation:

$$
\begin{aligned}
S &\Rightarrow aSb\\
&\Rightarrow aaSbb\\
&\Rightarrow aaaSbbb\\
&\Rightarrow aaaaSbbbb\\
&\Rightarrow aaaabbbb.
\end{aligned}
$$



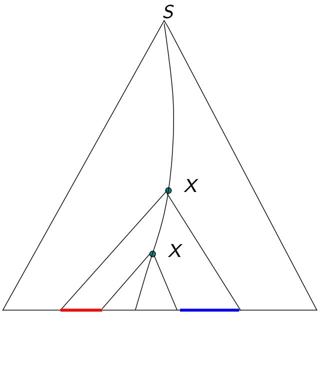
Cutting and pasting in parse trees

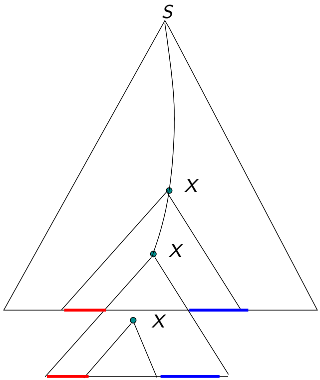
Subtrees hanging at same non-terminal can be replaced for eachother.



Proof idea

A long string must have a deep parse tree, which in turn means a path with a repeated non-terminal.



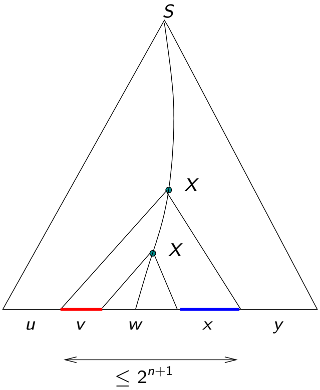


Proof

- Let G be a CNF grammar for L.
- A complete binary tree with i levels has 2^{i-1} leaf nodes.
- A parse tree in G with i levels has a terminal string ("yield") of length at most 2^{i-2}.
- Hence a string of length 2^n or more, must have a parse tree of at least $n+2$ levels.
- Take $k = 2^n$ where n is number of non-terminals in G.

Proof - II

- Consider parse tree in G of a string z of length at least $k = 2^n$.
- Consider longest path from root to leaf.
- Choose the first repeated non-terminal X starting from bottom of path.
- Path from upper X down to leaf is at most $n + 2$ levels. Also it must be the longest path in the subtree rooted at X. Hence length of vwx is at most 2^n.
- Also $vx \neq \epsilon$, as G is a CNF grammar.



Applications

Argue that the following languages are not CFL's:

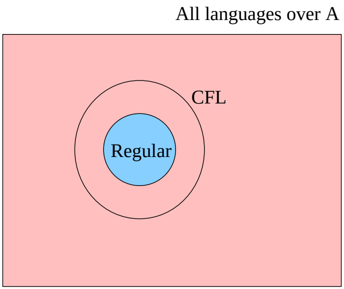
- $\{a^n b^n c^n \mid n \geq 0\}$.

Applications

Argue that the following languages are not CFL's:

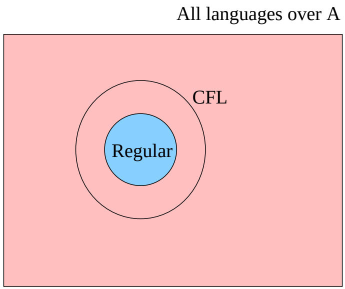
- $\{a^n b^n c^n \mid n \geq 0\}$.
- $\{ww \mid w \in \{a, b\}^*\}$.

Closure Properties of CFL's



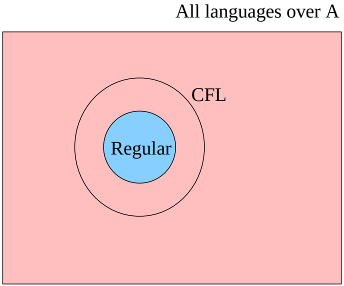
	Closed?
Union	

Closure Properties of CFL’s



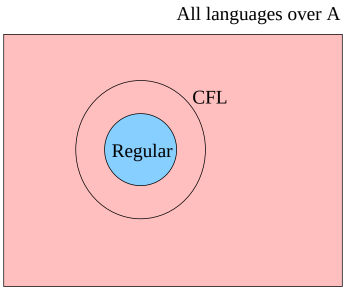
	Closed?
Union	$\surd$
Intersection	

Closure Properties of CFL's



	Closed?
Union	$\surd$
Intersection	X
Complementation	

Closure Properties of CFL's



	Closed?
Union	$\surd$
Intersection	X
Complementation	X

Outline

Parikh map of a string

- Let $A = \{a_1, \ldots, a_n\}$ be a finite alphabet.
- Parikh map of a string $w \in A^*$ is defined a vector in $\mathbb{N}^n$ given by:
$$\psi(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w)).$$
- For example if $A = \{a, b\}$, then $\psi(baabb) = (2, 3)$.
- Parikh map is also called the "letter-count" of a string.
- Extend the map to languages L over A:
$$\psi(L) = \{\psi(w) \mid w \in L\}.$$
- What is $\psi(\{a^n b^n \mid n \geq 0\})$?

Parikh map of a string

- Let $A = \{a_1, \ldots, a_n\}$ be a finite alphabet.
- Parikh map of a string $w \in A^*$ is defined a vector in $\mathbb{N}^n$ given by:
$$\psi(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w)).$$
- For example if $A = \{a, b\}$, then $\psi(baabb) = (2, 3)$.
- Parikh map is also called the "letter-count" of a string.
- Extend the map to languages L over A:
$$\psi(L) = \{\psi(w) \mid w \in L\}.$$
- What is $\psi(\{a^n b^n \mid n \geq 0\})$?
  - $\{(n, n) \mid n \geq 0\}$.
- What is $\psi(\{w \in \{a, b\}^* \mid \#_a(w) \leq \#_b(w)\})$?

Parikh map of a string

- Let $A = \{a_1, \ldots, a_n\}$ be a finite alphabet.
- Parikh map of a string $w \in A^*$ is defined a vector in $\mathbb{N}^n$ given by:

$$\psi(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w)).$$

- For example if $A = \{a, b\}$, then $\psi(baabb) = (2, 3)$.
- Parikh map is also called the "letter-count" of a string.
- Extend the map to languages L over A:

$$\psi(L) = \{\psi(w) \mid w \in L\}.$$

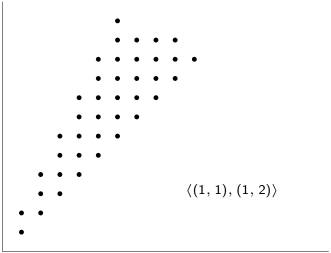
- What is $\psi(\{a^n b^n \mid n \geq 0\})$?
  - $\{(n, n) \mid n \geq 0\}$.
- What is $\psi(\{w \in \{a, b\}^* \mid \#_a(w) \leq \#_b(w)\})$?
  - $\{(i, j) \mid i \leq j\}$.

Semi-linear sets of vectors

- The set of vectors generated by a set of vectors $u_1, \ldots, u_k$ in $\mathbb{N}^n$, denoted $\rangle u_1, \ldots, u_k \langle$, is the set

$$\{d_1 \cdot u_1 + d_2 \cdot u_2 + \cdots + d_k \cdot u_k \mid d_i \in \mathbb{N}\}.$$

- A subset X of $\mathbb{N}^n$ is called linear if there exist vectors $u_0, u_1, \ldots, u_k$ such that $X = u_0 + \langle u_1, u_2, \ldots, u_k \rangle$.



- A set of vectors is called semi-linear if it is a finite union of linear sets.

Parikh's Theorem for CFL's

Theorem (Parikh's theorem)

The Parikh map of a CFL is a semi-linear set.

Some corollaries:

- Every CFL is "letter-equivalent" to a regular language.
  - For example: $\psi(\{a^n b^n\}) = \psi((ab)^*)$.
- Lengths of a CFL forms an ultimate periodic set.
- CFL's over a single-letter alphabet are regular.

Parikh's Theorem for CFL's

Theorem (Parikh's theorem)

The Parikh map of a CFL is a semi-linear set.

Some corollaries:

- Every CFL is "letter-equivalent" to a regular language.
  - For example: $\psi(\{a^n b^n\}) = \psi((ab)^*)$.
- Lengths of a CFL forms an ultimate periodic set.
- CFL's over a single-letter alphabet are regular.

Is Parikh's theorem sufficient as well?

Parikh's Theorem for CFL's

Theorem (Parikh's theorem)

The Parikh map of a CFL is a semi-linear set.

Some corollaries:

- Every CFL is "letter-equivalent" to a regular language.
    - For example: $\psi(\{a^n b^n\}) = \psi((ab)^*)$.
- Lengths of a CFL forms an ultimate periodic set.
- CFL's over a single-letter alphabet are regular.

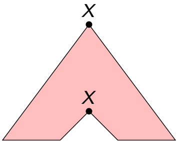
Is Parikh's theorem sufficient as well?

- No, since $\psi(\{a^n b^n c^n \mid n \geq 0\} = \{(n, n, n) \mid n \geq 0\}$ is semi-linear.

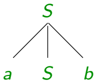
Proof: Pumps

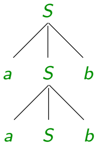
Let us fix a CFG $G = (N, A, S, P)$ in CNF form.

- A pump is a derivation tree s which has at least two nodes, and $yield(s) = x \cdot root(s) \cdot y$, for some terminal strings x, y.

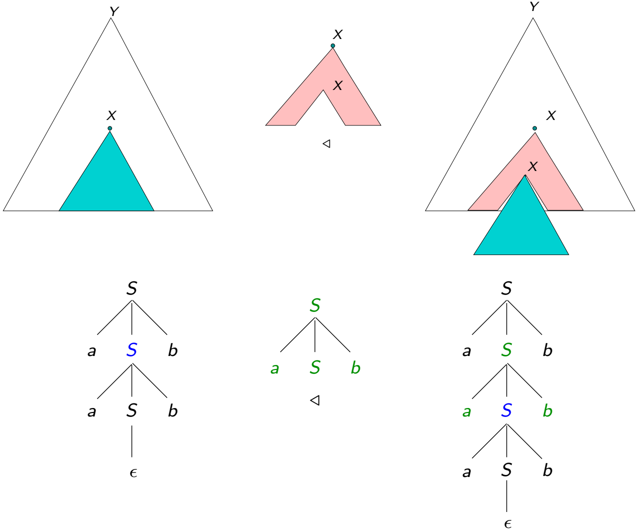


- Example pumps for grammar $S \to aSb \mid SS \mid \epsilon$:



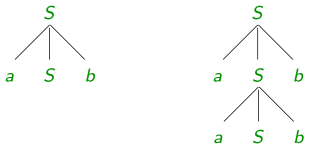


Growing and shrinking with pumps



Basic Pumps

Pumps which are $\triangleleft$-minimal: Thus a pump s is a basic pump if it cannot be shrunk by some pump and still remain a pump.



First pump is basic but second is not.
Basic pumps are finite in number (height bounded by $2.|N|.$).

$\leq$ relation on parse trees

- Let s and t be derivation trees of terminal strings starting from start symbol S.
- Then we say $s \leq t$ iff t can be grown from s by basic pumps whose non-terminals are contained in those of s (thus the pumps do not introduce any new non-terminals, and s an t have the same set of non-terminal nodes).
- A parse tree s is thus $\leq$-minimal if it does not contain a basic pump that can be cut out without reducing the set of non-terminals that occur in s.
- $\leq$-minimal trees can be seen to be finite in number (height bounded by $(p+1)(n+1)$.

Overall strategy of Proof

- Begin with the $\leq$-minimal derivation trees, say $s_1, \ldots, s_k$.
- Associate with each s_i the set of basic pumps whose non-terminals are contained in that of s_i.
- Argue that the set of derivation trees obtained by starting with s_i and growing using the associated basic pumps, gives rise to a set of strings whose Parikh map is linear.