Context Free Languages and Grammars

Lecture 7

Tuesday, September 13, 2022

LaTeXed: October 13, 2022 14:18

7.1
A fluffy introduction to context free languages, push down automatas

What stack got to do with it?

What's a stack but a second hand memory?

1. DFA/NFA/Regular expressions.
   $\equiv$ constant memory computation.
2. Turing machines DFA/NFA $+$ unbounded memory.
   $\equiv$ a standard computer/program.

What stack got to do with it?

What's a stack but a second hand memory?

1. $\mathrm{DFA}/\mathrm{NFA}$/Regular expressions.
   $\equiv$ constant memory computation.
2. NFA + stack
   $\equiv$ context free grammars (CFG).
3. Turing machines $\mathrm{DFA}/\mathrm{NFA}$ + unbounded memory.
   $\equiv$ a standard computer/program.

What stack got to do with it?

What's a stack but a second hand memory?

1. $\mathrm{DFA}/\mathrm{NFA}$/Regular expressions.
   $\equiv$ constant memory computation.
2. NFA + stack
   $\equiv$ context free grammars (CFG).
3. Turing machines $\mathrm{DFA}/\mathrm{NFA}$ + unbounded memory.
   $\equiv$ a standard computer/program.
   $\equiv$ NFA with two stacks.

Context Free Languages and Grammars

- Programming Language Specification
- Parsing
- Natural language understanding
- Generative model giving structure
- ...

Programming Languages

```
<relational-expression> ::= <shift-expression>
                          | <relational-expression> < <shift-expression>
                          | <relational-expression> > <shift-expression>
                          | <relational-expression> <= <shift-expression>
                          | <relational-expression> >= <shift-expression>

<shift-expression> ::= <additive-expression>
                     | <shift-expression> << <additive-expression>
                     | <shift-expression> >> <additive-expression>

<additive-expression> ::= <multiplicative-expression>
                        | <additive-expression> + <multiplicative-expression>
                        | <additive-expression> - <multiplicative-expression>

<multiplicative-expression> ::= <cast-expression>
                              | <multiplicative-expression> * <cast-expression>
                              | <multiplicative-expression> / <cast-expression>
                              | <multiplicative-expression> % <cast-expression>

<cast-expression> ::= <unary-expression>
                    | ( <type-name> ) <cast-expression>

<unary-expression> ::= <postfix-expression>
                     | ++ <unary-expression>
                     | -- <unary-expression>
                     | <unary-operator> <cast-expression>
                     | sizeof <unary-expression>
                     | sizeof <type-name>

<postfix-expression> ::= <primary-expression>
                       | <postfix-expression> [ <expression> ]
                       | <postfix-expression> ( {<assignment-expression>}* )
                       | <postfix-expression> . <identifier>
                       | <postfix-expression> -> <identifier>
                       | <postfix-expression> ++
                       | <postfix-expression> --
```

Natural Language Processing

English sentences can be described as

$$
\begin{aligned}
&\langle S\rangle \to \langle NP\rangle\langle VP\rangle\\
&\langle NP\rangle \to \langle CN\rangle \mid \langle CN\rangle\langle PP\rangle\\
&\langle VP\rangle \to \langle CV\rangle \mid \langle CV\rangle\langle PP\rangle\\
&\langle PP\rangle \to \langle P\rangle\langle CN\rangle\\
&\langle CN\rangle \to \langle A\rangle\langle N\rangle\\
&\langle CV\rangle \to \langle V\rangle \mid \langle V\rangle\langle NP\rangle\\
&\langle A\rangle \to \text{a} \mid \text{the}\\
&\langle N\rangle \to \text{boy} \mid \text{girl} \mid \text{flower}\\
&\langle V\rangle \to \text{touches} \mid \text{likes} \mid \text{sees}\\
&\langle P\rangle \to \text{with}
\end{aligned}
$$

English Sentences

Examples

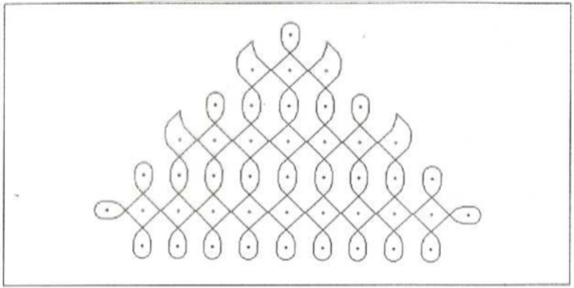
$$\overbrace{\underbrace{\text{a}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}}^{\text{verb-phrs}}$$

$$\overbrace{\underbrace{\text{the}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}\ \underbrace{\text{a flower}}_{\text{noun-phrs}}}^{\text{verb-phrs}}$$

Models of Growth

- **L**-systems
- `http://www.kevs3d.co.uk/dev/lsystems/`

Kolam drawing generated by grammar



7.2
Formal definition of convex-free languages (CFGs)

Context Free Grammar (CFG) Definition

Definition 7.1.

A CFG is a quadruple $\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S})$

- $\mathbf{V}$ is a finite set of non-terminal symbols
- $\mathbf{T}$ is a finite set of terminal symbols (alphabet)
- $\mathbf{P}$ is a finite set of productions, each of the form
  $\mathbf{A} \to \alpha$
  where $\mathbf{A} \in \mathbf{V}$ and α is a string in $(\mathbf{V} \cup \mathbf{T})^*$.
  Formally, $\mathbf{P} \subset \mathbf{V} \times (\mathbf{V} \cup \mathbf{T})^*$.
- $\mathbf{S} \in \mathbf{V}$ is a start symbol

$$\mathbf{G} = \Big(\text{ Variables, } \quad \text{Terminals, } \quad \text{Productions, } \quad \text{Start var } \Big)$$

Context Free Grammar (CFG) Definition

Definition 7.1.

A CFG is a quadruple $\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S})$

- $\mathbf{V}$ is a finite set of non-terminal symbols
- $\mathbf{T}$ is a finite set of terminal symbols (alphabet)
- $\mathbf{P}$ is a finite set of productions, each of the form
  $\mathbf{A} \to \alpha$
  where $\mathbf{A} \in \mathbf{V}$ and α is a string in $(\mathbf{V} \cup \mathbf{T})^*$.
  Formally, $\mathbf{P} \subset \mathbf{V} \times (\mathbf{V} \cup \mathbf{T})^*$.
- $\mathbf{S} \in \mathbf{V}$ is a start symbol

$$\mathbf{G} = \Big(\text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \Big)$$

Context Free Grammar (CFG) Definition

Definition 7.1.

A CFG is a quadruple $\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S})$

- $\mathbf{V}$ is a finite set of non-terminal symbols
- $\mathbf{T}$ is a finite set of terminal symbols (alphabet)
- $\mathbf{P}$ is a finite set of productions, each of the form
  $\mathbf{A} \to \alpha$
  where $\mathbf{A} \in \mathbf{V}$ and α is a string in $(\mathbf{V} \cup \mathbf{T})^*$.
  Formally, $\mathbf{P} \subset \mathbf{V} \times (\mathbf{V} \cup \mathbf{T})^*$.
- $\mathbf{S} \in \mathbf{V}$ is a start symbol

$$\mathbf{G} = \Big(\text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \Big)$$

Context Free Grammar (CFG) Definition

Definition 7.1.

A CFG is a quadruple $\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S})$

- $\mathbf{V}$ is a finite set of non-terminal symbols
- $\mathbf{T}$ is a finite set of terminal symbols (alphabet)
- $\mathbf{P}$ is a finite set of productions, each of the form
  $\mathbf{A} \to \alpha$
  where $\mathbf{A} \in \mathbf{V}$ and α is a string in $(\mathbf{V} \cup \mathbf{T})^*$.
  Formally, $\mathbf{P} \subset \mathbf{V} \times (\mathbf{V} \cup \mathbf{T})^*$.
- $\mathbf{S} \in \mathbf{V}$ is a start symbol

$$\mathbf{G} = \Big(\text{Variables}, \quad \text{Terminals}, \quad \text{Productions}, \quad \text{Start var} \Big)$$

Example

- $\mathbf{V} = \{\mathbf{S}\}$
- $\mathbf{T} = \{\mathbf{a}, \mathbf{b}\}$
- $\mathbf{P} = \{\mathbf{S} \rightarrow \epsilon \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{aSa} \mid \mathbf{bSb}\}$
  (abbrev. for $\mathbf{S} \rightarrow \epsilon, \mathbf{S} \rightarrow \mathbf{a}, \mathbf{S} \rightarrow \mathbf{b}, \mathbf{S} \rightarrow \mathbf{aSa}, \mathbf{S} \rightarrow \mathbf{bSb}$)

$$\mathbf{S} \rightsquigarrow \mathbf{aSa} \rightsquigarrow \mathbf{abSba} \rightsquigarrow \mathbf{abbSbba} \rightsquigarrow \mathbf{abb\,b\,bba}$$

What strings can $\mathbf{S}$ generate like this?

Example

- $\mathbf{V} = \{\mathbf{S}\}$
- $\mathbf{T} = \{\mathbf{a}, \mathbf{b}\}$
- $\mathbf{P} = \{\mathbf{S} \rightarrow \epsilon \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{aSa} \mid \mathbf{bSb}\}$
  (abbrev. for $\mathbf{S} \rightarrow \epsilon, \mathbf{S} \rightarrow \mathbf{a}, \mathbf{S} \rightarrow \mathbf{b}, \mathbf{S} \rightarrow \mathbf{aSa}, \mathbf{S} \rightarrow \mathbf{bSb}$)

$$\mathbf{S} \rightsquigarrow \mathbf{aSa} \rightsquigarrow \mathbf{abSba} \rightsquigarrow \mathbf{abbSbba} \rightsquigarrow \mathbf{abb\,b\,bba}$$

What strings can $\mathbf{S}$ generate like this?

Example

- $\mathbf{V} = \{\mathbf{S}\}$
- $\mathbf{T} = \{\mathbf{a}, \mathbf{b}\}$
- $\mathbf{P} = \{\mathbf{S} \to \epsilon \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{aSa} \mid \mathbf{bSb}\}$
  (abbrev. for $\mathbf{S} \to \epsilon, \mathbf{S} \to \mathbf{a}, \mathbf{S} \to \mathbf{b}, \mathbf{S} \to \mathbf{aSa}, \mathbf{S} \to \mathbf{bSb}$)

$$\mathbf{S} \rightsquigarrow \mathbf{aSa} \rightsquigarrow \mathbf{abSba} \rightsquigarrow \mathbf{abbSbba} \rightsquigarrow \mathbf{abb\,b\,bba}$$

What strings can $\mathbf{S}$ generate like this?

Example formally...

- $\mathbf{V} = \{\mathbf{S}\}$
- $\mathbf{T} = \{\mathbf{a}, \mathbf{b}\}$
- $\mathbf{P} = \{\mathbf{S} \to \epsilon \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{aSa} \mid \mathbf{bSb}\}$
  (abbrev. for $\mathbf{S} \to \epsilon, \mathbf{S} \to \mathbf{a}, \mathbf{S} \to \mathbf{b}, \mathbf{S} \to \mathbf{aSa}, \mathbf{S} \to \mathbf{bSb}$)

$$\mathbf{G} = \left(\{\mathbf{S}\}, \quad \{\mathbf{a}, \mathbf{b}\}, \quad \left\{ \begin{array}{c} \mathbf{S} \to \epsilon, \\ \mathbf{S} \to \mathbf{a}, \\ \mathbf{S} \to \mathbf{b} \\ \mathbf{S} \to \mathbf{aSa} \\ \mathbf{S} \to \mathbf{bSb} \end{array} \right\} \quad \mathbf{S} \right)$$

Palindromes

- Madam in Eden I'm Adam
- Dog doo? Good God!
- Dogma: I am God.
- A man, a plan, a canal, Panama
- Are we not drawn onward, we few, drawn onward to new era?
- Doc, note: I dissent. A fast never prevents a fatness. I diet on cod.
- `http://www.palindromelist.net`

Examples

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

$\mathbf{S \rightarrow \epsilon \mid 0S1}$

Examples

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

$$S \to \epsilon \mid 0S1$$

Notation and Convention

Let $\mathbf{G = (V, T, P, S)}$ then

- $\mathbf{a, b, c, d, \ldots,}$ in $\mathbf{T}$ (terminals)
- $\mathbf{A, B, C, D, \ldots,}$ in $\mathbf{V}$ (non-terminals)
- $\mathbf{u, v, w, x, y, \ldots}$ in $\mathbf{T}^*$ for strings of terminals
- $\alpha, \beta, \gamma, \ldots$ in $\mathbf{(V \cup T)}^*$
- $\mathbf{X, Y, X}$ in $\mathbf{V \cup T}$

“Derives” relation

Formalism for how strings are derived/generated

Definition 7.2.

Let $\mathbf{G} = (\mathbf{V}, \mathbf{T}, \mathbf{P}, \mathbf{S})$ be a CFG. For strings $\alpha_1, \alpha_2 \in (\mathbf{V} \cup \mathbf{T})^*$ we say α_1 derives α_2 denoted by $\alpha_1 \rightsquigarrow_{\mathbf{G}} \alpha_2$ if there exist strings β, γ, δ in $(\mathbf{V} \cup \mathbf{T})^*$ such that

- $\alpha_1 = \beta\mathbf{A}\delta$
- $\alpha_2 = \beta\gamma\delta$
- $\mathbf{A} \to \gamma$ is in $\mathbf{P}$.

Examples: $\mathbf{S} \rightsquigarrow \epsilon$, $\mathbf{S} \rightsquigarrow \mathbf{0S1}$, $\mathbf{0S1} \rightsquigarrow \mathbf{00S11}$, $\mathbf{0S1} \rightsquigarrow \mathbf{01}$.

"Derives" relation continued

Definition 7.3.

For integer $\mathbf{k} \geq \mathbf{0}$, $\alpha_1 \rightsquigarrow^{k} \alpha_2$ inductive defined:

- $\alpha_1 \rightsquigarrow^{0} \alpha_2$ if $\alpha_1 = \alpha_2$
- $\alpha_1 \rightsquigarrow^{k} \alpha_2$ if $\alpha_1 \rightsquigarrow \beta_1$ and $\beta_1 \rightsquigarrow^{k-1} \alpha_2$.
- Alternative definition: $\alpha_1 \rightsquigarrow^{k} \alpha_2$ if $\alpha_1 \rightsquigarrow^{k-1} \beta_1$ and $\beta_1 \rightsquigarrow \alpha_2$

$\rightsquigarrow^{*}$ is the reflexive and transitive closure of $\rightsquigarrow$.

$\alpha_1 \rightsquigarrow^{*} \alpha_2$ if $\alpha_1 \rightsquigarrow^{k} \alpha_2$ for some $\mathbf{k}$.

Examples: $\mathbf{S} \rightsquigarrow^{*} \epsilon$, $\mathbf{0S1} \rightsquigarrow^{*} \mathbf{0000011111}$.

"Derives" relation continued

Definition 7.3.

For integer $k \geq 0$, $\alpha_1 \rightsquigarrow^k \alpha_2$ inductive defined:

- $\alpha_1 \rightsquigarrow^0 \alpha_2$ if $\alpha_1 = \alpha_2$
- $\alpha_1 \rightsquigarrow^k \alpha_2$ if $\alpha_1 \rightsquigarrow \beta_1$ and $\beta_1 \rightsquigarrow^{k-1} \alpha_2$.
- Alternative definition: $\alpha_1 \rightsquigarrow^k \alpha_2$ if $\alpha_1 \rightsquigarrow^{k-1} \beta_1$ and $\beta_1 \rightsquigarrow \alpha_2$

$\rightsquigarrow^*$ is the reflexive and transitive closure of $\rightsquigarrow$.

$\alpha_1 \rightsquigarrow^* \alpha_2$ if $\alpha_1 \rightsquigarrow^k \alpha_2$ for some k.

Examples: $S \rightsquigarrow^* \epsilon$, $0S1 \rightsquigarrow^* 0000011111$.

"Derives" relation continued

Definition 7.3.

For integer $\mathbf{k} \geq \mathbf{0}$, $\alpha_1 \rightsquigarrow^{\mathbf{k}} \alpha_2$ inductive defined:

- $\alpha_1 \rightsquigarrow^{\mathbf{0}} \alpha_2$ if $\alpha_1 = \alpha_2$
- $\alpha_1 \rightsquigarrow^{\mathbf{k}} \alpha_2$ if $\alpha_1 \rightsquigarrow \beta_1$ and $\beta_1 \rightsquigarrow^{\mathbf{k-1}} \alpha_2$.
- Alternative definition: $\alpha_1 \rightsquigarrow^{\mathbf{k}} \alpha_2$ if $\alpha_1 \rightsquigarrow^{\mathbf{k-1}} \beta_1$ and $\beta_1 \rightsquigarrow \alpha_2$

$\rightsquigarrow^*$ is the reflexive and transitive closure of $\rightsquigarrow$.

$\alpha_1 \rightsquigarrow^* \alpha_2$ if $\alpha_1 \rightsquigarrow^{\mathbf{k}} \alpha_2$ for some $\mathbf{k}$.

Examples: $\mathbf{S} \rightsquigarrow^* \epsilon$, $\mathbf{0S1} \rightsquigarrow^* \mathbf{0000011111}$.

Context Free Languages

Definition 7.4.

The language generated by CFG $\mathbf{G = (V, T, P, S)}$ is denoted by $\mathbf{L(G)}$ where $\mathbf{L(G) = \{w \in T^* \mid S \rightsquigarrow^* w\}}$.

Definition 7.5.

A language $\mathbf{L}$ is context free (CFL) if it is generated by a context free grammar. That is, there is a CFG $\mathbf{G}$ such that $\mathbf{L = L(G)}$.

Context Free Languages

Definition 7.4.

The language generated by CFG $\mathbf{G = (V, T, P, S)}$ is denoted by $\mathbf{L(G)}$ where $\mathbf{L(G) = \{w \in T^* \mid S \rightsquigarrow^* w\}}$.

Definition 7.5.

A language $\mathbf{L}$ is context free (CFL) if it is generated by a context free grammar. That is, there is a CFG $\mathbf{G}$ such that $\mathbf{L = L(G)}$.

Example

$$\mathbf{L = \{0^n1^n \mid n \geq 0\}}$$

$$\mathbf{S \rightarrow \epsilon \mid 0S1}$$

$$\mathbf{L = \{0^n1^m \mid m > n\}}$$

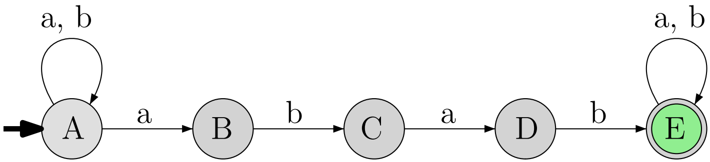
$$\mathbf{L} = \left\{\mathbf{w} \in \{(,)\}^* \;\middle|\; \mathbf{w} \text{ is properly nested string of parenthesis}\right\}.$$

7.3
Converting regular languages into CFL

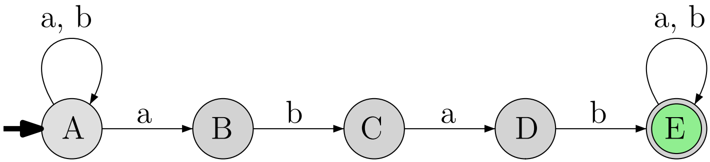
Converting regular languages into CFL

$\mathbf{M = (Q, \Sigma, \delta, s, A)}$: DFA for regular language $\mathbf{L}$.

$$\mathbf{G} = \left(\overbrace{\mathbf{Q}}^{\text{Variables}}, \overbrace{\mathbf{\Sigma}}^{\text{Terminals}}, \overbrace{\begin{array}{c}\{\mathbf{q} \to \mathbf{a}\delta(\mathbf{q},\mathbf{a}) \mid \mathbf{q} \in \mathbf{Q}, \mathbf{a} \in \mathbf{\Sigma}\} \\ \cup \{\mathbf{q} \to \varepsilon \mid \mathbf{q} \in \mathbf{A}\}\end{array}}^{\text{Productions}}, \overbrace{\mathbf{s}}^{\text{Start var}} \right)$$



Conversion continued...



$$\mathbf{G} = \left(\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}\}, \{\mathbf{a}, \mathbf{b}\}, \left\{ \begin{array}{c} \mathbf{A} \rightarrow \mathbf{aA}, \mathbf{A} \rightarrow \mathbf{bA}, \mathbf{A} \rightarrow \mathbf{aB}, \\ \mathbf{B} \rightarrow \mathbf{bC}, \\ \mathbf{C} \rightarrow \mathbf{aD}, \\ \mathbf{D} \rightarrow \mathbf{bE}, \\ \mathbf{E} \rightarrow \mathbf{aE}, \mathbf{E} \rightarrow \mathbf{bE}, \mathbf{E} \rightarrow \varepsilon \end{array} \right\}, \mathbf{A} \right)$$

The result...

Lemma 7.1.

For an regular language $\mathbf{L}$*, there is a context-free grammar* (CFG) *that generates it.*

7.4
CFL as a python program

0^n1^n

The grammar **G**:

$$\mathbf{S} \to \varepsilon \mid \mathbf{0S1}$$

Can be translated into the python program:

```
#! /bin/python3
import random

# S → epsilon | 0 S 1
def  S():
    match  random.randrange(10):
        case 0:
            return   # epsilon
        case _:
            print( "0", end='' )
            S()
            print( "1", end='' )

S()
print( "" )
```

L(G) = any string that this program might output.

Balanced parenthesis expression

The grammar **G**:

$$\mathbf{S} \rightarrow \varepsilon \mid \mathbf{(S)} \mid \mathbf{SS}$$

Can be translated into the python program:

```
#! /bin/python3
import random

# S → epsilon  |  ( S )  |  S S
def  S():
    match  random.randrange(3):
        case 0:       # epsilon
            return

        case 1:       # ( S )
            print( "(", end='' )
            S()
            print( ")", end='' )

        case _:       # SS
            S()
            S()

S()
print( "" )
```

L(G) = any string that this program might output.

7.5
Some properties of CFLs

7.5.1
Closure properties of CFLs

Bad news: Canonical non-CFL

Theorem 7.1.

$L = \{a^n b^n c^n \mid n \geq 0\}$ *is not context-free.*

Proof based on pumping lemma for CFLs. See supplemental for the proof.

More bad news: CFL not closed under intersection

Theorem 7.2.

CFLs *are* *not* *closed under intersection.*

Closure Properties of CFLs

$\mathbf{G_1 = (V_1, T, P_1, S_1)}$ and $\mathbf{G_2 = (V_2, T, P_2, S_2)}$

Assumption: $\mathbf{V_1 \cap V_2 = \emptyset}$, that is, non-terminals are not shared

Theorem 7.3.

CFLs are closed under union. $\mathbf{L_1, L_2}$ CFLs implies $\mathbf{L_1 \cup L_2}$ is a CFL.

Theorem 7.4.

CFLs are closed under concatenation. $\mathbf{L_1, L_2}$ CFLs implies $\mathbf{L_1 \bullet L_2}$ is a CFL.

Theorem 7.5.

CFLs are closed under Kleene star.

If $\mathbf{L}$ is a CFL $\implies$ $\mathbf{L^*}$ is a CFL.

Closure Properties of CFLs

$\mathbf{G_1 = (V_1, T, P_1, S_1)}$ and $\mathbf{G_2 = (V_2, T, P_2, S_2)}$

Assumption: $\mathbf{V_1 \cap V_2 = \emptyset}$, that is, non-terminals are not shared

Theorem 7.3.

CFLs are closed under union. $\mathbf{L_1, L_2}$ CFLs implies $\mathbf{L_1 \cup L_2}$ is a CFL.

Theorem 7.4.

CFLs are closed under concatenation. $\mathbf{L_1, L_2}$ CFLs implies $\mathbf{L_1 \bullet L_2}$ is a CFL.

Theorem 7.5.

CFLs are closed under Kleene star.

If $\mathbf{L}$ is a CFL $\implies$ $\mathbf{L^*}$ is a CFL.

Closure Properties of $\mathrm{CFL_S}$

Union

$\mathbf{G_1 = (V_1, T, P_1, S_1)}$ and $\mathbf{G_2 = (V_2, T, P_2, S_2)}$
Assumption: $\mathbf{V_1 \cap V_2 = \emptyset}$, that is, non-terminals are not shared.

[redacted]

$\mathrm{CFL_S}$ *are closed under union.* $\mathbf{L_1, L_2}$ $\mathrm{CFL_S}$ *implies* $\mathbf{L_1 \cup L_2}$ *is a* CFL.

Closure Properties of CFLs

Concatenation

Theorem 7.7.

CFLs *are closed under concatenation.* $\mathbf{L_1, L_2}$ CFLs *implies* $\mathbf{L_1 \bullet L_2}$ *is a* CFL.

Closure Properties of CFLs

Stardom (i.e, Kleene star)

[redacted]

CFLs *are closed under Kleene star.*

If $\mathbf{L}$ *is a* CFL $\implies$ $\mathbf{L}^*$ *is a* CFL.

Exercise

- Prove that every regular language is context-free using previous closure properties.
- Prove the set of regular expressions over an alphabet $\boldsymbol{\Sigma}$ forms a non-regular language which is context-free.

Even more bad news: CFL not closed under complement

Theorem 7.9.

CFLs *are* *not* *closed under complement.*

Good news: Closure Properties of CFLs continued

Theorem 7.10.

If $\mathbf{L_1}$ *is a* CFL *and* $\mathbf{L_2}$ *is regular then* $\mathbf{L_1} \cap \mathbf{L_2}$ *is a* CFL.

7.5.2
Parse trees and ambiguity

Parse Trees or Derivation Trees

A tree to represent the derivation $\mathbf{S} \rightsquigarrow^* \mathbf{w}$.

- Rooted tree with root labeled $\mathbf{S}$
- Non-terminals at each internal node of tree
- Terminals at leaves
- Children of internal node indicate how non-terminal was expanded using a production rule

A picture is worth a thousand words

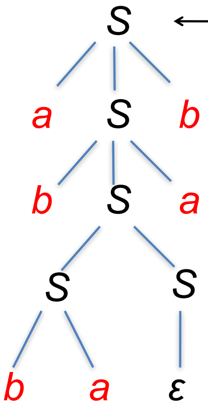
Parse Trees or Derivation Trees

A tree to represent the derivation $\mathbf{S} \rightsquigarrow^* \mathbf{w}$.

- Rooted tree with root labeled $\mathbf{S}$
- Non-terminals at each internal node of tree
- Terminals at leaves
- Children of internal node indicate how non-terminal was expanded using a production rule

A picture is worth a thousand words

Example



A derivation tree for *abbaab*

(also called "parse tree")

$S \rightarrow aSb \mid bSa \mid SS \mid ab \mid ba \mid \varepsilon$

A corresponding derivation of *abbaab*

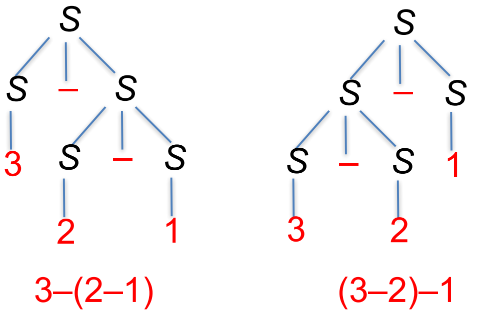
$S \rightarrow aSb \rightarrow abSab \rightarrow abSSab \rightarrow abbaSab \rightarrow abbaab$

Ambiguity in CFLs

Definition 7.11.

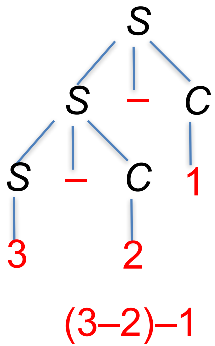
A CFG $\mathbf{G}$ is ambiguous if there is a string $\mathbf{w} \in \mathbf{L(G)}$ with two different parse trees. If there is no such string then $\mathbf{G}$ is unambiguous.

Example: $\mathbf{S} \rightarrow \mathbf{S} - \mathbf{S} \mid \mathbf{1} \mid \mathbf{2} \mid \mathbf{3}$



Ambiguity in CFLs

- Original grammar: $\mathbf{S} \rightarrow \mathbf{S} - \mathbf{S} \mid \mathbf{1} \mid \mathbf{2} \mid \mathbf{3}$
- Unambiguous grammar:
  $\mathbf{S} \rightarrow \mathbf{S} - \mathbf{C} \mid \mathbf{1} \mid \mathbf{2} \mid \mathbf{3}$
  $\mathbf{C} \rightarrow \mathbf{1} \mid \mathbf{2} \mid \mathbf{3}$



The grammar forces a parse corresponding to left-to-right evaluation.

Inherently ambiguous languages

Definition 7.12.

A CFL $\mathbf{L}$ is inherently ambiguous if there is no unambiguous CFG $\mathbf{G}$ such that $\mathbf{L} = \mathbf{L}(\mathbf{G})$.

- There exist inherently ambiguous CFLs.
  Example: $\mathbf{L} = \{\mathbf{a}^n\mathbf{b}^m\mathbf{c}^k \mid \mathbf{n} = \mathbf{m} \text{ or } \mathbf{m} = \mathbf{k}\}$
- Given a grammar $\mathbf{G}$ it is undecidable to check whether $\mathbf{L}(\mathbf{G})$ is inherently ambiguous. No algorithm!

Inherently ambiguous languages

Definition 7.12.

A CFL $\mathbf{L}$ is inherently ambiguous if there is no unambiguous CFG $\mathbf{G}$ such that $\mathbf{L} = \mathbf{L}(\mathbf{G})$.

- There exist inherently ambiguous CFLs.
  Example: $\mathbf{L} = \{\mathbf{a}^n\mathbf{b}^m\mathbf{c}^k \mid \mathbf{n} = \mathbf{m} \text{ or } \mathbf{m} = \mathbf{k}\}$
- Given a grammar $\mathbf{G}$ it is undecidable to check whether $\mathbf{L}(\mathbf{G})$ is inherently ambiguous. No algorithm!

Inherently ambiguous languages

Definition 7.12.

A CFL $\mathbf{L}$ is inherently ambiguous if there is no unambiguous CFG $\mathbf{G}$ such that $\mathbf{L} = \mathbf{L(G)}$.

- There exist inherently ambiguous CFLs.
  Example: $\mathbf{L} = \{\mathbf{a^n b^m c^k} \mid \mathbf{n} = \mathbf{m} \text{ or } \mathbf{m} = \mathbf{k}\}$
- Given a grammar $\mathbf{G}$ it is undecidable to check whether $\mathbf{L(G)}$ is inherently ambiguous. No algorithm!

7.6
CFGs; Proving a grammar generate a specific language

Inductive proofs for CFGs

Question: How do we formally prove that a CFG $\mathbf{L(G) = L}$?

Example: $\mathbf{S} \to \epsilon \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{aSa} \mid \mathbf{bSb}$

Theorem 7.1.
$\mathbf{L(G)} = \{\textit{palindromes}\} = \{\mathbf{w} \mid \mathbf{w} = \mathbf{w}^{\mathbf{R}}\}$

Two directions:

- $\mathbf{L(G)} \subseteq \mathbf{L}$, that is, $\mathbf{S} \rightsquigarrow^* \mathbf{w}$ then $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$
- $\mathbf{L} \subseteq \mathbf{L(G)}$, that is, $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$ then $\mathbf{S} \rightsquigarrow^* \mathbf{w}$

Inductive proofs for CFGs

Question: How do we formally prove that a CFG $\mathbf{L(G) = L}$?

Example: $\mathbf{S} \rightarrow \epsilon \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{aSa} \mid \mathbf{bSb}$

Theorem 7.1.

$\mathbf{L(G)} = \{\textit{palindromes}\} = \{\mathbf{w} \mid \mathbf{w} = \mathbf{w}^{\mathbf{R}}\}$

Two directions:

- $\mathbf{L(G)} \subseteq \mathbf{L}$, that is, $\mathbf{S} \rightsquigarrow^* \mathbf{w}$ then $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$
- $\mathbf{L} \subseteq \mathbf{L(G)}$, that is, $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$ then $\mathbf{S} \rightsquigarrow^* \mathbf{w}$

$L(G) \subseteq L$

Show that if $S \rightsquigarrow^* w$ then $w = w^R$

By induction on length of derivation, meaning
For all $k \geq 1$, $S \rightsquigarrow^{*k} w$ implies $w = w^R$.

- If $S \rightsquigarrow^1 w$ then $w = \epsilon$ or $w = a$ or $w = b$. Each case $w = w^R$.
- Assume that for all $k < n$, that if $S \rightarrow^k w$ then $w = w^R$
- Let $S \rightsquigarrow^n w$ (with $n > 1$). Wlog w begin with a.
  - Then $S \rightarrow aSa \rightsquigarrow^{k-1} aua$ where $w = aua$.
  - And $S \rightsquigarrow^{n-1} u$ and hence IH, $u = u^R$.
  - Therefore $w^r = (aua)^R = (ua)^R a = au^R a = aua = w$.

$\mathbf{L(G) \subseteq L}$

Show that if $\mathbf{S} \rightsquigarrow^* \mathbf{w}$ then $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$

By induction on length of derivation, meaning
For all $\mathbf{k} \geq \mathbf{1}$, $\mathbf{S} \rightsquigarrow^{*\mathbf{k}} \mathbf{w}$ implies $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$.

- If $\mathbf{S} \rightsquigarrow^{\mathbf{1}} \mathbf{w}$ then $\mathbf{w} = \epsilon$ or $\mathbf{w} = \mathbf{a}$ or $\mathbf{w} = \mathbf{b}$. Each case $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$.
- Assume that for all $\mathbf{k} < \mathbf{n}$, that if $\mathbf{S} \rightarrow^{\mathbf{k}} \mathbf{w}$ then $\mathbf{w} = \mathbf{w}^{\mathbf{R}}$
- Let $\mathbf{S} \rightsquigarrow^{\mathbf{n}} \mathbf{w}$ (with $\mathbf{n} > \mathbf{1}$). Wlog $\mathbf{w}$ begin with $\mathbf{a}$.
    - Then $\mathbf{S} \rightarrow \mathbf{aSa} \rightsquigarrow^{\mathbf{k-1}} \mathbf{aua}$ where $\mathbf{w} = \mathbf{aua}$.
    - And $\mathbf{S} \rightsquigarrow^{\mathbf{n-1}} \mathbf{u}$ and hence IH, $\mathbf{u} = \mathbf{u}^{\mathbf{R}}$.
    - Therefore $\mathbf{w}^{\mathbf{r}} = \mathbf{(aua)}^{\mathbf{R}} = \mathbf{(ua)}^{\mathbf{R}}\mathbf{a} = \mathbf{au}^{\mathbf{R}}\mathbf{a} = \mathbf{aua} = \mathbf{w}$.

$\mathbf{L \subseteq L(G)}$

Show that if $\mathbf{w = w^R}$ then $\mathbf{S \rightsquigarrow^* w}$.

By induction on $|\mathbf{w}|$
That is, for all $\mathbf{k \geq 0}$, $|\mathbf{w}| = \mathbf{k}$ and $\mathbf{w = w^R}$ implies $\mathbf{S \rightsquigarrow^* w}$.

Exercise: Fill in proof.

Mutual Induction

Situation is more complicated with grammars that have multiple non-terminals.

See Section 5.3.2 of the notes for an example proof.

7.7
CFGs normal form

Normal Forms

Normal forms are a way to restrict form of production rules

Advantage: Simpler/more convenient algorithms and proofs

Two standard normal forms for CFGs

- Chomsky normal form
- Greibach normal form

Normal Forms

Normal forms are a way to restrict form of production rules

Advantage: Simpler/more convenient algorithms and proofs

Two standard normal forms for CFGs

- Chomsky normal form
- Greibach normal form

Normal Forms

Chomsky Normal Form:

- Productions are all of the form $\mathbf{A} \rightarrow \mathbf{BC}$ or $\mathbf{A} \rightarrow \mathbf{a}$.
  If $\epsilon \in \mathbf{L}$ then $\mathbf{S} \rightarrow \epsilon$ is also allowed.
- Every CFG $\mathbf{G}$ can be converted into CNF form via an efficient algorithm
- Advantage: parse tree of constant degree.

Greibach Normal Form:

- Only productions of the form $\mathbf{A} \rightarrow \mathbf{a}\beta$ are allowed.
- All CFLs without ϵ have a grammar in GNF. Efficient algorithm.
- Advantage: Every derivation adds exactly one terminal.

Normal Forms

Chomsky Normal Form:

- Productions are all of the form $\mathbf{A} \rightarrow \mathbf{BC}$ or $\mathbf{A} \rightarrow \mathbf{a}$.
  If $\epsilon \in \mathbf{L}$ then $\mathbf{S} \rightarrow \epsilon$ is also allowed.
- Every CFG $\mathbf{G}$ can be converted into CNF form via an efficient algorithm
- Advantage: parse tree of constant degree.

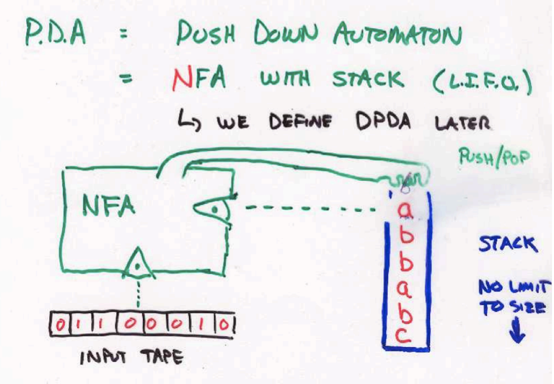
Greibach Normal Form:

- Only productions of the form $\mathbf{A} \rightarrow \mathbf{a}\beta$ are allowed.
- All CFLs without ϵ have a grammar in GNF. Efficient algorithm.
- Advantage: Every derivation adds exactly one terminal.

7.8
Pushdown automatas

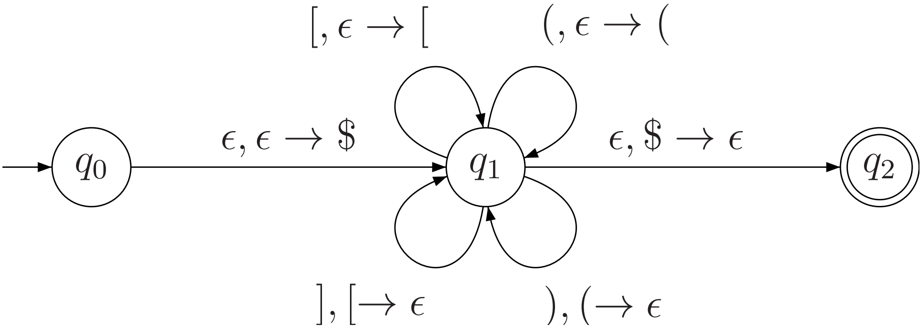
Things to know: Pushdown Automata

PDA: a NFA coupled with a stack



PDAs and CFGs are equivalent: both generate exactly CFLs.
PDA is a machine-centric view of CFLs.

Pushdown automata by example



7.9
Supplemental: Why $\mathbf{a^n b^n c^n}$ is not CFL

You are bound to repeat yourself...

$\mathsf{L} = \{\mathsf{a}^n \mathsf{b}^n \mathsf{c}^n \mid n \geq 0\}$.

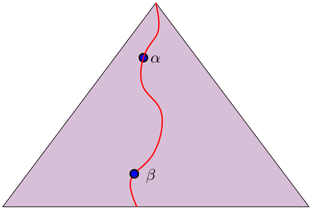
1. For the sake of contradiction assume that there exists a grammar: **G** a CFG for **L**.
2. T_i: minimal parse tree in **G** for $\mathsf{a}^i \mathsf{b}^i \mathsf{c}^i$.
3. $h_i = \mathrm{height}(\mathsf{T}_i)$: Length of longest path from root to leaf in T_i.
4. For any integer t, there must exist an index $j(t)$, such that $h_{j(t)} > t$.
5. There an index j, such that $h_j > \left(2 * \ \#\text{ variables in } \mathsf{G}\right)$.

You are bound to repeat yourself...

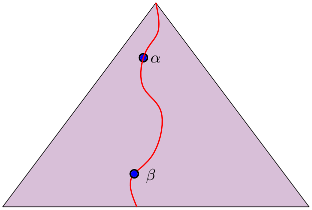
$\mathsf{L} = \{\mathsf{a}^n \mathsf{b}^n \mathsf{c}^n \mid n \geq 0\}$.

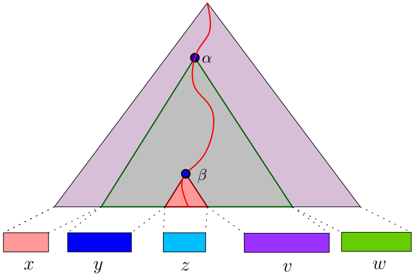
1. For the sake of contradiction assume that there exists a grammar: **G** a CFG for **L**.
2. $\mathbf{T_i}$: minimal parse tree in **G** for $\mathsf{a}^i\mathsf{b}^i\mathsf{c}^i$.
3. $h_i = \mathrm{height}(T_i)$: Length of longest path from root to leaf in $\mathbf{T_i}$.
4. For any integer t, there must exist an index $j(t)$, such that $h_{j(t)} > t$.
5. There an index j, such that $h_j > \left(2 * \ \# \text{ variables in } \mathbf{G}\right)$.

Repetition in the parse tree...



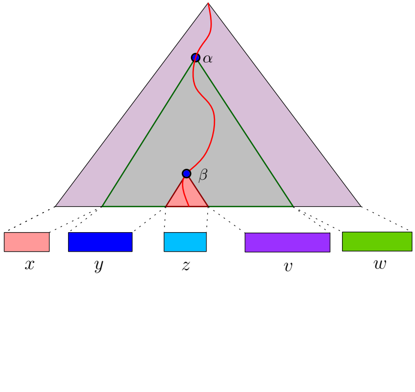
Repetition in the parse tree...





xyzvw = $\mathbf{a^j b^j c^j}$

Repetition in the parse tree...



$\mathbf{xyzvw} = \mathbf{a^j b^j c^j} \implies \mathbf{xy^2zv^2w} \in \mathbf{L}$

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a^j b^j c^j}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy^2zv^2w} \in \mathbf{L}$.
- If $\mathbf{y}$ contains both $\mathbf{a}$ and $\mathbf{b}$, then, $\tau = \mathbf{...a...b...a...b...}$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a^n b^n c^n} \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that $\mathbf{y}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{a}$ and $\mathbf{b}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- If $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{b}$s, then... $\#_{\mathbf{a}}(\tau) \neq \#_{\mathbf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
  Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{b}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a}^{\mathbf{j}}\mathbf{b}^{\mathbf{j}}\mathbf{c}^{\mathbf{j}}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy}^2\mathbf{zv}^2\mathbf{w} \in \mathbf{L}$.
- If $\mathbf{y}$ contains both $\mathbf{a}$ and $\mathbf{b}$, then, $\tau = \mathbf{...a...b...a...b...}$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a}^{\mathbf{n}}\mathbf{b}^{\mathbf{n}}\mathbf{c}^{\mathbf{n}} \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that $\mathbf{y}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{a}$ and $\mathbf{b}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- If $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{b}$s, then... $\#_{\mathbf{a}}(\tau) \neq \#_{\mathbf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
  Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{b}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a^j b^j c^j}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy^2zv^2w} \in \mathbf{L}$.
- If $\mathbf{y}$ contains both $\mathbf{a}$ and $\mathbf{b}$, then, $\tau = \mathbf{...a...b...a...b...}$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a^n b^n c^n} \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that $\mathbf{y}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{a}$ and $\mathbf{b}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- If $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{b}$s, then... $\#_{\mathsf{a}}(\tau) \neq \#_{\mathsf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
  Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{b}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a}^{\mathbf{j}}\mathbf{b}^{\mathbf{j}}\mathbf{c}^{\mathbf{j}}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy}^2\mathbf{zv}^2\mathbf{w} \in \mathbf{L}$.
- If $\mathbf{y}$ contains both $\mathbf{a}$ and $\mathbf{b}$, then, $\tau = \mathbf{...a...b...a...b...}$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a}^{\mathbf{n}}\mathbf{b}^{\mathbf{n}}\mathbf{c}^{\mathbf{n}} \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that $\mathbf{y}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{a}$ and $\mathbf{b}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- If $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{b}$s, then... $\#_{\mathbf{a}}(\tau) \neq \#_{\mathbf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
  Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{b}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a^j b^j c^j}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy^2zv^2w} \in \mathbf{L}$.
- If $\mathbf{y}$ contains both $\mathbf{a}$ and $\mathbf{b}$, then, $\tau = \mathbf{...a...b...a...b...}$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a^n b^n c^n} \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that $\mathbf{y}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{a}$ and $\mathbf{b}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- If $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{b}$s, then... $\#_{\mathbf{a}}(\tau) \neq \#_{\mathbf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
  Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{b}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

Now for some case analysis...

- We know:
  $xyzvw = a^j b^j c^j$
  $|y| + |v| > 0$.
- We proved that $\tau = xy^2zv^2w \in L$.
- If y contains both a and b, then, $\tau = ...a...b...a...b...$.
  Impossible, since $\tau \in L = \{a^n b^n c^n \mid n \geq 0\}$.
- Similarly, not possible that y contains both b and c.
- Similarly, not possible that v contains both a and b.
- Similarly, not possible that v contains both b and c.
- If y contains only as, and v contains only bs, then... $\#_a(\tau) \neq \#_c(\tau)$.
  Not possible.
- Similarly, not possible that y contains only as, and v contains only cs.
  Similarly, not possible that y contains only bs, and v contains only cs.
- Must be that $\tau \notin L$. A contradiction.

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a}^{j}\mathbf{b}^{j}\mathbf{c}^{j}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy}^2\mathbf{zv}^2\mathbf{w} \in \mathbf{L}$.
- If **y** contains both **a** and **b**, then, $\tau = \ldots\mathbf{a}\ldots\mathbf{b}\ldots\mathbf{a}\ldots\mathbf{b}\ldots$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a}^n\mathbf{b}^n\mathbf{c}^n \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that **y** contains both **b** and **c**.
- Similarly, not possible that **v** contains both **a** and **b**.
- Similarly, not possible that **v** contains both **b** and **c**.
- If **y** contains only **a**s, and **v** contains only **b**s, then... $\#_{\mathbf{a}}(\tau) \neq \#_{\mathbf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that **y** contains only **a**s, and **v** contains only **c**s.
  Similarly, not possible that **y** contains only **b**s, and **v** contains only **c**s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

Now for some case analysis...

- We know:
  $\mathbf{xyzvw} = \mathbf{a^j b^j c^j}$
  $|\mathbf{y}| + |\mathbf{v}| > \mathbf{0}$.
- We proved that $\tau = \mathbf{xy^2zv^2w} \in \mathbf{L}$.
- If $\mathbf{y}$ contains both $\mathbf{a}$ and $\mathbf{b}$, then, $\tau = \mathbf{...a...b...a...b...}$.
  Impossible, since $\tau \in \mathbf{L} = \{\mathbf{a^n b^n c^n} \mid \mathbf{n} \geq \mathbf{0}\}$.
- Similarly, not possible that $\mathbf{y}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{a}$ and $\mathbf{b}$.
- Similarly, not possible that $\mathbf{v}$ contains both $\mathbf{b}$ and $\mathbf{c}$.
- If $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{b}$s, then... $\#_{\mathbf{a}}(\tau) \neq \#_{\mathbf{c}}(\tau)$.
  Not possible.
- Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{a}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
  Similarly, not possible that $\mathbf{y}$ contains only $\mathbf{b}$s, and $\mathbf{v}$ contains only $\mathbf{c}$s.
- Must be that $\tau \notin \mathbf{L}$. A contradiction.

We conclude...

Lemma 7.1.

The language $\mathbf{L} = \{\mathbf{a^n b^n c^n} \mid \mathbf{n} \geq \mathbf{0}\}$ *is not* CFL *(i.e., there is no* CFG *for it).*