Context Free Languages and Grammars

Lecture 7

Tuesday, September 15, 2020

LaTeXed: September 1, 2020 21:21

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. DFA/NFA/Regular expressions.
   $\equiv$ constant memory computation.
2. NFA + stack
   $\equiv$ context free grammars (CFG).
3. 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. DFA/NFA/Regular expressions.
   $\equiv$ constant memory computation.
2. NFA + stack
   $\equiv$ context free grammars (CFG).
3. Turing machines DFA/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> --
```

English sentences can be described as

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

English Sentences

Examples

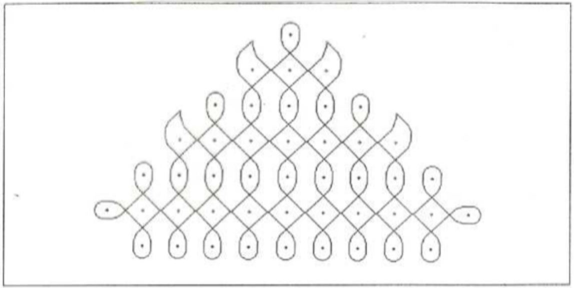
$$\overbrace{\underbrace{\text{a}}_{\text{article}}\underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}}^{\text{verb-phrs}}$$

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

Models of Growth

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

Kolam drawing generated by grammar



THE END

...

(for now)

7.2
Formal definition of convex-free languages (CFGs)

Context Free Grammar (CFG) Definition

Definition

A CFG is a quadruple $G = (V, T, P, S)$

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

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

Context Free Grammar (CFG) Definition

Definition

A CFG is a quadruple $G = (V, T, P, S)$

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

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

Context Free Grammar (CFG) Definition

Definition

A CFG is a quadruple $G = (V, T, P, S)$

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

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

Context Free Grammar (CFG) Definition

Definition

A CFG is a quadruple $G = (V, T, P, S)$

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

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

Example

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

$$S \rightsquigarrow aSa \rightsquigarrow abSba \rightsquigarrow abbSbba \rightsquigarrow abb\,b\,bba$$

What strings can S generate like this?

Example

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

$$S \rightsquigarrow aSa \rightsquigarrow abSba \rightsquigarrow abbSbba \rightsquigarrow abb\,b\,bba$$

What strings can S generate like this?

Example

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

$$S \rightsquigarrow aSa \rightsquigarrow abSba \rightsquigarrow abbSbba \rightsquigarrow abb\,b\,bba$$

What strings can S generate like this?

Example formally...

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

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

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

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

$S \rightarrow \epsilon \mid 0S1$

Examples

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

$S \rightarrow \epsilon \mid 0S1$

Notation and Convention

Let $G = (V, T, P, S)$ then

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

"Derives" relation

Formalism for how strings are derived/generated

Definition

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

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

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

"Derives" relation continued

Definition

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

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

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

Context Free Languages

Definition

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

Definition

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

Context Free Languages

Definition

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

Definition

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

Example

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

$S \rightarrow \epsilon \mid 0S1$

$L = \{0^n 1^m \mid m > n\}$

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

THE END

...

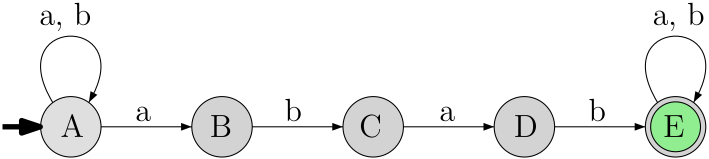
(for now)

7.3
Converting regular languages into CFL

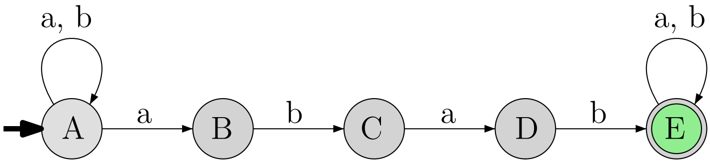
Converting regular languages into CFL

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

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



Conversion continued...



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

The result...

Lemma

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

THE END

...

(for now)

7.4
Some properties of CFLs

7.4.1
Closure properties of CFLs

Bad news: Canonical non-CFL

Theorem

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

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

Closure Properties of CFLs

$G_1 = (V_1, T, P_1, S_1)$ and $G_2 = (V_2, T, P_2, S_2)$

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

Theorem

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

Theorem

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

Theorem

CFLs are closed under Kleene star.
If L is a CFL $\implies$ L^* is a CFL.

Closure Properties of CFLs

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

Theorem

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

Theorem

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

Theorem

CFLs *are closed under Kleene star.*
If L *is a* CFL $\implies$ L^* *is a* CFL.

Closure Properties of CFLs

Union

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

Theorem

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

Closure Properties of CFLs

Concatenation

Theorem

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

Theorem

CFLs are closed under Kleene star.
If L is a CFL $\implies$ 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 Σ forms a non-regular language which is context-free.

Even more bad news: CFL not closed under complement

Theorem

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

Good news: Closure Properties of CFLs continued

Theorem

If L_1 is a CFL and L_2 is regular then $L_1 \cap L_2$ is a CFL.

THE END

...

(for now)

7.4.2
Parse trees and ambiguity

Parse Trees or Derivation Trees

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

- Rooted tree with root labeled 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

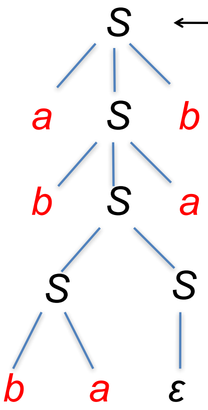
Parse Trees or Derivation Trees

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

- Rooted tree with root labeled 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*

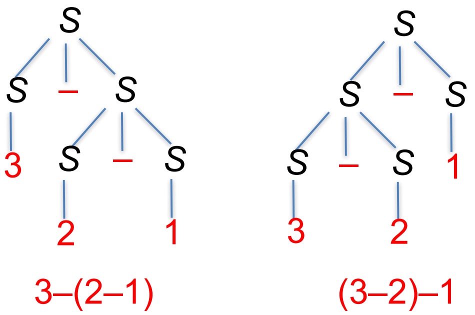
$S \Rightarrow aSb \Rightarrow abSab \Rightarrow abSSab \Rightarrow abbaSab \Rightarrow abbaab$

Ambiguity in CFLs

Definition

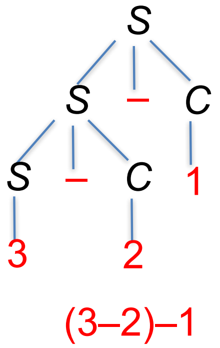
A CFG G is ambiguous if there is a string $w \in L(G)$ with two different parse trees. If there is no such string then G is unambiguous.

Example: $S \rightarrow S - S \mid 1 \mid 2 \mid 3$



Ambiguity in CFLs

- Original grammar: $S \to S - S \mid 1 \mid 2 \mid 3$
- Unambiguous grammar:
  $S \to S - C \mid 1 \mid 2 \mid 3$
  $C \to 1 \mid 2 \mid 3$



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

Inherently ambiguous languages

Definition

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

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

Inherently ambiguous languages

Definition

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

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

Inherently ambiguous languages

Definition

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

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

THE END

...

(for now)

7.5
CFGs; Proving a grammar generate a specific language

Inductive proofs for CFGs

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

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

Theorem

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

Two directions:

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

Inductive proofs for CFGs

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

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

Theorem

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

Two directions:

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

$L(G) \subseteq L$

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 $S \to^k w$ then $w = w^R$
- Let $S \rightsquigarrow^n w$ (with $n > 1$). Wlog w begin with a.
  - Then $S \to aSa \rightsquigarrow^{k-1} aua$ where $w = aua$.
  - And $S \rightsquigarrow^{n-1} u$ and hence IH, $u = u^R$.
  - Therefore $w^r = (aua)^R = (ua)^R a = au^R a = aua = w$.

$L(G) \subseteq L$

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 $w^r = (aua)^R = (ua)^R a = au^R a = aua = w$.

$L \subseteq L(G)$

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

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

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.

THE END

...

(for now)

7.6
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 $A \rightarrow BC$ or $A \rightarrow a$.
  If $\epsilon \in L$ then $S \rightarrow \epsilon$ is also allowed.
- Every CFG 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 $A \rightarrow a\beta$ 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 $A \rightarrow BC$ or $A \rightarrow a$.
  If $\epsilon \in L$ then $S \rightarrow \epsilon$ is also allowed.
- Every CFG 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 $A \rightarrow a\beta$ are allowed.
- All CFLs without ϵ have a grammar in GNF. Efficient algorithm.
- Advantage: Every derivation adds exactly one terminal.

THE END

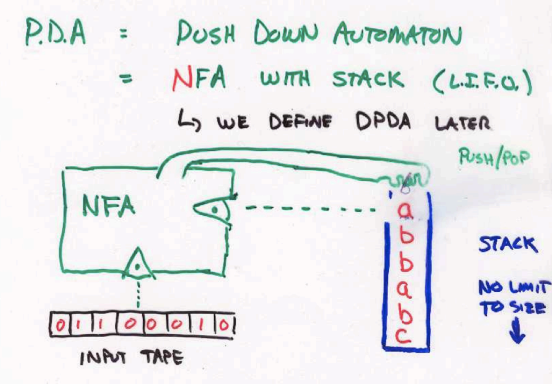
...

(for now)

7.7
Pushdown automatas

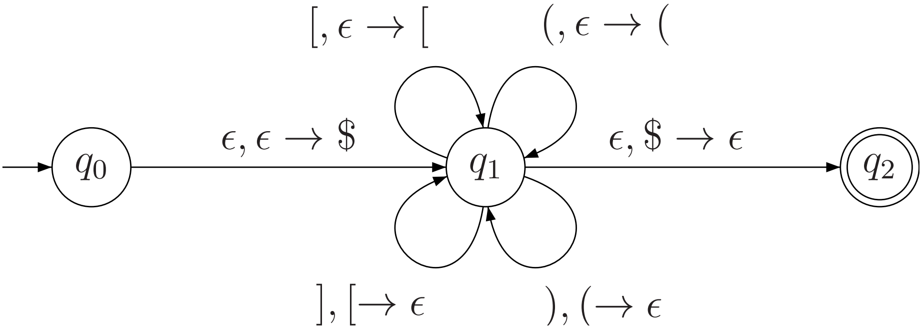
Things to know: Pushdown Automata

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



THE END

...

(for now)

7.8
Supplemental: Why $a^n b^n c^n$ is not CFL

You are bound to repeat yourself...

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

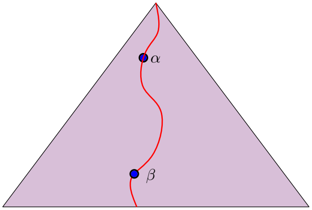
1. For the sake of contradiction assume that there exists a grammar: G a CFG for L.
2. T_i: minimal parse tree in G for $a^i b^i c^i$.
3. $h_i = \mathrm{height}(T_i)$: Length of longest path from root to leaf in T_i.
4. For any integer t, there must exist an index $j(t)$, such that $h_{j(t)} > t$.
5.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

You are bound to repeat yourself...

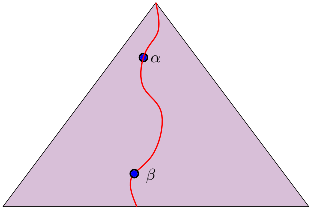
$L = \{a^n b^n c^n \mid n \geq 0\}$.

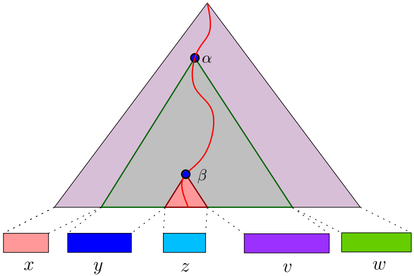
1. For the sake of contradiction assume that there exists a grammar: G a CFG for L.
2. T_i: minimal parse tree in G for $a^i b^i c^i$.
3. $h_i = \mathrm{height}(T_i)$: Length of longest path from root to leaf in T_i.
4. For any integer t, there must exist an index $j(t)$, such that $h_{j(t)} > t$.
5.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

Repetition in the parse tree...



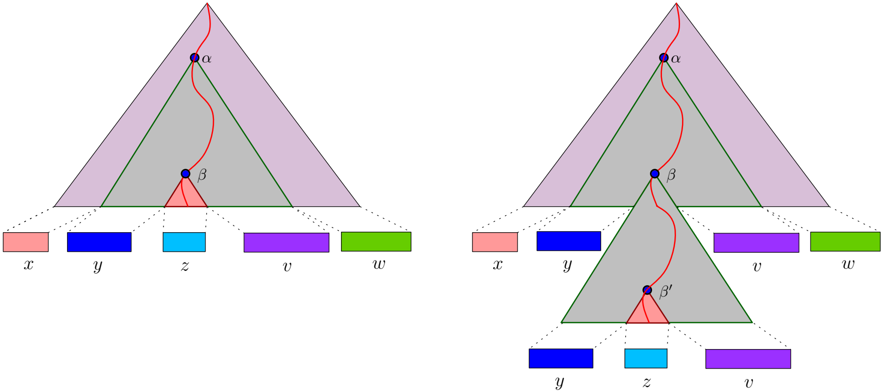
Repetition in the parse tree...





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

Repetition in the parse tree...



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

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:
  $xyzvw = a^i b^j c^j$
  $|y| + |v| > 0$.
- 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:
  $xyzvw = a^i b^j c^j$
  $|y| + |v| > 0$.
- 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:
  $xyzvw = a^j b^j c^j$
  $|y| + |v| > 0$.
- 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:
  $xyzvw = a^j b^j c^j$
  $|y| + |v| > 0$.
- 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:
  $xyzvw = a^i b^j c^j$
  $|y| + |v| > 0$.
- 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:
  $xyzvw = a^i b^j c^j$
  $|y| + |v| > 0$.
- 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:
  $xyzvw = a^j b^j c^j$
  $|y| + |v| > 0$.
- 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.

We conclude...

Lemma

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