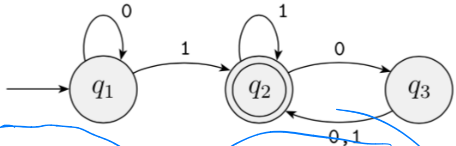
# Administrivia

- average 30.6
- pass 32.5

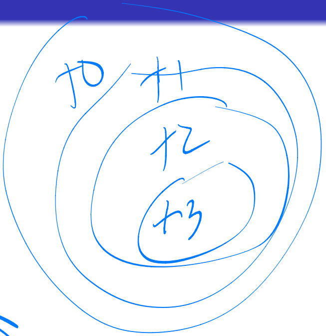
- Optional Exam #1: grades posted, comment by 6pm Thursday
- Exercise sheet #4 assigned today, due next Wednesday
- Optional Exam #2: Wednesday, November 12, 9:00-10:00
- Reading: Chapter 2, 3
- Last time: Context Free Languages, Push-down Automata, Grammars, Normal Forms
- Today: PDAs $\Longleftrightarrow$ CFGs, Chomsky hierarchy, non-CFL languages

# Type 3 Grammars




$S \to 0S | 1A$
$A \to 1A | 0B | \epsilon$
$B \to 0A | 1A$

$S \to 1$
$A \to 1$
$B \to 0 | 1$




## Definition

A grammar for a regular language is a type3 right-linear grammar if every rule is of the form:

$A \to aB$

$A \to a$

where $a$ is any terminal, and $A$ and $B$ are any variables. If $\epsilon$ is in the language than $S \to \epsilon$ is a rule, where $S$ is the start variable.

if we remove $A \to \epsilon$ we add 4 rules to grammar

But what about the rule $A \to \epsilon$ in the grammar above?

# Type 3 Grammars and Regular Languages

Consider the grammar:

$S \rightarrow aA|a$

$A \rightarrow aA|a|bB|b$

$B \rightarrow bB|b$

What makes this a type 3 grammar?

- Chomsky's hierarchy: type 0, type 1, type 2, type 3
- the most general rule is $\alpha \rightarrow \beta$, where $\alpha$ and $\beta$ are strings of terminals and variables
- rule restrictions: $\alpha \in V$, $|\alpha| = 1$, $|\beta| \leq 2$, etc.
- e.g., the grammar above it not only type 3 but also a right linear grammar

left linear

# Chomsky's Language Hierarchy (1956)

There are 4 languages (4 grammar types) in the hierarchy:

- Type 3 grammars (regular languages):
  $\alpha \to \beta$
  $\alpha \in V$ and $\beta \in \Sigma V \cup \Sigma$ ($S \to \epsilon$ if $\epsilon \in L$)
- Type 2 grammars (context-free languages):
  $\alpha \to \beta$
  $\alpha \in V$ and $\beta \in (\Sigma \cup V)^*$
- Type 1 grammars (context-sensitive languages):
  $\alpha A \beta \to \alpha \gamma \beta$
  $A \in V$, $\alpha, \beta \in (\Sigma \cup V)^*$, $\gamma \in (\Sigma \cup V)^+$
- Type 0 grammars (unrestricted languages):
  $\alpha \to \beta$
  $\alpha, \beta \in (\Sigma \cup V)^*$

# Type 3 Grammars and Regular Languages

**Theorem**

*A language is regular if and only if a Type3 grammar generates it.*

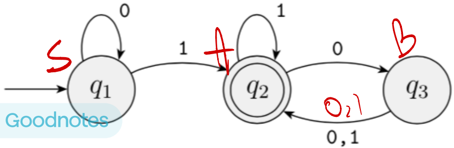
**Proof.**

$\Rightarrow$ Let $M = \{Q, \Sigma, \delta, q_S, F\}$ be the DFA for the language. We use $M$ to construct the grammar $G = (V, \Sigma_G, R, S)$ as follows.

- $\Sigma_G = \Sigma$
- $V = Q$
- $S = q_0$
- $R = \begin{cases} q_i \rightarrow aq_j, & \delta(q_i, a) \rightarrow q_j \\ q \rightarrow \epsilon, & q \in F \end{cases}$

final answer

$S \rightarrow 1A \mid 0S \mid 1$
$A \rightarrow 0B \mid 1A \mid 1$
$B \rightarrow 0A \mid 1A \mid 0 \mid 1$




$S \rightarrow 1A \mid 0S$
$A \rightarrow 0B \mid 1A \mid \varepsilon$
$B \rightarrow 0A \mid 1A$

# Type 3 Grammars and Regular Languages

**Theorem**

*A language is regular if and only if a Type3 grammar generates it.*

**Proof.**

$\Leftarrow$ Let $G = (V, \Sigma_G, R, S)$ be a Type3 grammar. We construct equivalent NFA $N = \{Q, \Sigma, \delta, q_S, F\}$ from it as follows.

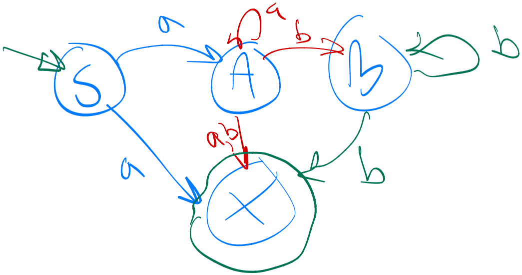
- $\Sigma = \Sigma_G$
- $Q = V \cup X$, where $X$ is not already in $V$
- $q_0 = S$
- $F = \{X\}$ (or $F = \{X \cup S\}$ when $S \to \epsilon$ is a rule of $G$)
- $\delta \begin{cases} B \in \delta(A, a), & \text{if } A \to aB \in R \\ X \in \delta(A, a), & \text{if } A \to a \in R \end{cases}$

$\square$

Example grammar to NFA conversion:

# Example grammar to NFA conversion

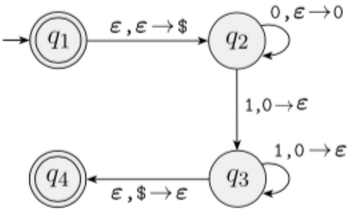
$\Sigma = \{a, b\}$




$S \to aA \mid a$
$A \to aA \mid a \mid bB \mid b$
$B \to bB \mid b$

$$\delta \begin{cases} B \in \delta(A, a), & \text{if } A \to aB \in R \\ X \in \delta(A, a), & \text{if } A \to a \in R \end{cases}$$

# PDA $\Longleftrightarrow$ CFG




$$S \rightarrow 0S1|\epsilon$$

# Languages Recognized by PDAs and Generated by CFGs

**Theorem**

*A language is context free if and only if some pushdown automaton recognizes it.*

**Proof.**

Recall that by definition, a language is CF if some CFG generates it. Then we need to show that the class of CFLs is equivalent to the class of languages recognized by PDAs.

- $\Rightarrow$ If $G$ is a CFG for language $A$, then there exists PDA $P$ that recognizes $A$
- $\Leftarrow$ If PDA $P$ recognizes a language $A$, then there exists a CFG $G$ that generates $A$

$\square$

# CFG $G \Rightarrow$ PDA P

**Lemma**

*Let $G$ be a CFG for language $A$. Then there exists PDA $P$ that recognizes $A$.*

**Proof.**

(Rough sketch) We construct a PDA $P$ that accepts its input $w$, if $G$ generates that input, by determining whether there exists a valid derivation for $w$ in $G$. A derivation is a sequence of substitutions made as a grammar generates a string. Each step of the derivation yields an intermediate string of variables and terminals. $P$ determines whether some series of substitutions using the rules of $G$ can lead from the start variable to $w$. But which of many possible derivations do we try? The PDA's nondeterminism allows it to "guess" the sequence of correct substitutions. □

# CFG $G \Rightarrow$ PDA P

Consider the CFG $G$ below; what is $L(G)$?

$S \rightarrow aTb|b$
$T \rightarrow Ta|\epsilon$

✗ a    b ✓
✗ aa    ab ✓
abb ✗
aab ✓

$a^*b$

# CFG $G \Rightarrow$ PDA P

Consider the CFG $G$ below; what is $L(G)$?

$S \rightarrow aTb|b$
$T \rightarrow Ta|\epsilon$

Create a PDA using the idea from the proof sketch:

# CFG $G \Rightarrow$ PDA P

Consider the CFG $G$ below; what is $L(G)$?

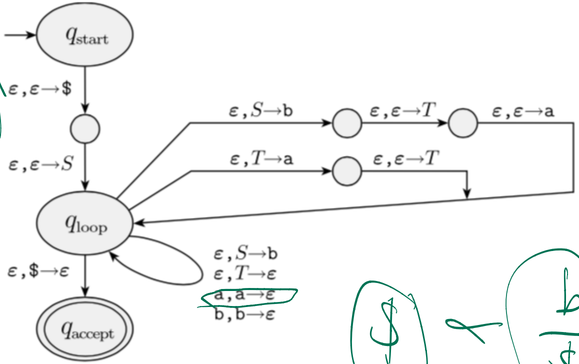
$S \to aTb | b$

$T \to Ta | \epsilon$

ab

$S \to aTb \to ab$

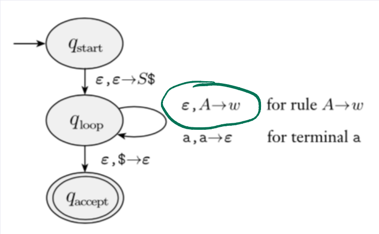
Create a PDA using the idea from the proof sketch:

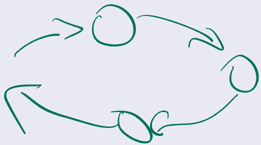


# CFG $G \Rightarrow$ PDA P

**Proof.**

We construct $P = (Q, \Sigma, \Gamma, \delta, q_s, F)$ from $G$ as follows. Let $(r, u) \in \delta(q, a, s)$ mean that when $P$ is in state $q$, with $a$ as the next input symbol and symbol $s$ on top of the stack, $P$ reads $a$, pops $s$, pushes the string $u$ onto the stack and goes to state $r$.
Then $Q = \{q_s, q_{loop}, q_{accept}\} \cup E$, where $E$ is the set of states needed for the $(r, u) \in \delta(q, a, s)$ trick.
The transition function is shown in this figure:



$\square$

# PDA $P \Rightarrow$ CFG $G$

## Lemma

*Let PDA $P$ recognize a language $A$. Then there exists a CFG $G$ that generates $A$.*

## Proof.

(Rough sketch) We design $G$ such that $G$ generates a string if that string causes the PDA to go from its start state to an accept state. For each pair of states $p$ and $q$ in $P$, the grammar will have a variable $A_{pq}$. This variable generates all the strings that can take $P$ from $p$ with an empty stack to $q$ with an empty stack. We further simplify $P$ to have the following three features.

1. $P$ has a single accept state.
2. $P$ empties its stack before accepting.
3. Each transition either pushes a symbol onto the stack (a push move) or pops one off the stack (a pop move), but it does not do both at the same time.

# PDA $P \Rightarrow$ CFG $G$

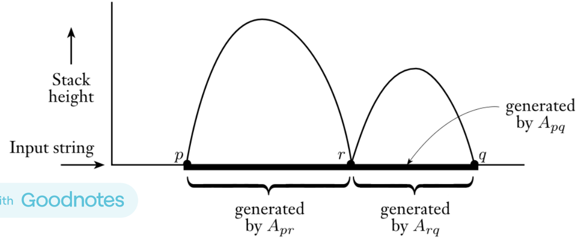
## Lemma

*Let PDA $P$ recognize a language $A$. Then there exists a CFG $G$ that generates $A$.*

## Proof.

Let $P = (Q, \Sigma, \Gamma, \delta, q_s, q_a)$. We construct $G$ from $P$ as follows.
The variables of G are $\{A_{pq} | p, q \in Q\}$. The start variable is $A_{q_s, q_a}$ and the rules of $G$ are:

1. For each $p, q, r, s \in Q, u \in \Gamma$, and $a, b \in \Sigma_\epsilon$, if $\delta(p, a, \epsilon)$ contains $(r, u)$ and $\delta(s, b, u)$ contains $(q, \epsilon)$, put the rule $A_{pq} \to aA_{rs}b$ in $G$



# PDA $P \Rightarrow$ CFG $G$

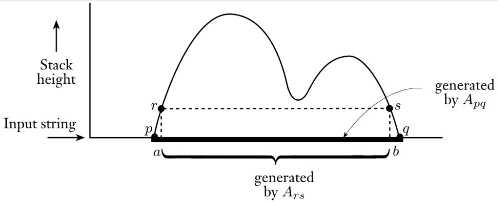
## Lemma

*Let PDA $P$ recognize a language $A$. Then there exists a CFG $G$ that generates $A$.*

## Proof.

Let $P = (Q, \Sigma, \Gamma, \delta, q_s, q_a)$. We construct $G$ from $P$ as follows.
The variables of G are $\{A_{pq} | p, q \in Q\}$. The start variable is $A_{q_s, q_a}$ and the rules of $G$ are:

1. For each $p, q, r, s \in Q, u \in \Gamma$, and $a, b \in \Sigma_\epsilon$, if $\delta(p, a, \epsilon)$ contains $(r, u)$ and $\delta(s, b, u)$ contains $(q, \epsilon)$, put the rule $A_{pq} \to aA_{rs}b$ in $G$
2. For each $p, q, r \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in $G$



# PDA $P \Rightarrow$ CFG $G$

## Lemma

*Let PDA $P$ recognize a language $A$. Then there exists a CFG $G$ that generates $A$.*

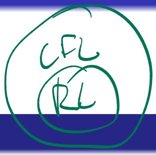
## Proof.

Let $P = (Q, \Sigma, \Gamma, \delta, q_s, q_a)$. We construct $G$ from $P$ as follows.
The variables of G are $\{A_{pq} | p, q \in Q\}$. The start variable is $A_{q_s, q_a}$ and the rules of $G$ are:

1. For each $p, q, r, s \in Q$, $u \in \Gamma$, and $a, b \in \Sigma_\epsilon$, if $\delta(p, a, \epsilon)$ contains $(r, u)$ and $\delta(s, b, u)$ contains $(q, \epsilon)$, put the rule $A_{pq} \to aA_{rs}b$ in $G$
2. For each $p, q, r \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in $G$
3. For each $p \in Q$, put the rule $A_{pp} \to \epsilon$ in $G$

Then we need a very careful proof by induction that if $A_{pq}$ generates $x$, then $x$ can bring $P$ from $p$ with empty stack to $q$ with empty stack.

# RL ⊂ *CFL*




**Theorem**

*Every regular language is also a context free language.*

**Proof.**

- By definition, a language is context free if there exists a PDA for it.
- By definition, a language is regular if there exists an NFA for it.
- And an NFA is a (special type of) PDA that does not use its stack.

# The Pumping Lemma (PL) for Context Free Languages

**Theorem**

*(PL) If $A$ is a context free language, then there exists a number $p$ (the pumping length) so that if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into five pieces, $s = uvxyz$, satisfying the following conditions:*

1. $uv^ixy^iz \in A$, *for each* $i \geq 0$,
2. $|vy| > 0$, *and*
3. $|vxy| \leq p$.

# The Pumping Lemma (PL) for Context Free Languages

**Theorem**

*If $A$ is context free, then $\exists p$ so that if $s \in A$ and $|s| > p$, then $s$ may be divided into five pieces, $s = uvxyz$, to satisfy:*

1. $uv^ixy^iz \in A$, *for each* $i \geq 0$,
2. $|vy| > 0$, *and*
3. $|vxy| \leq p$.


**Proof.**

(Sketch) Let $G$ be a CFG for CFL $A$. Then we set $p = |V|$, where $V$ are the variables of $G$. Consider some long string $s \in A$: long enough that in its derivation we must repeat some variable $R$ (again, by the pigeonhole principle). Then the following picture completes the sketch:


□

# The Pumping Lemma (PL) for Context Free Languages

**Proof.**

Let $G$ be a CFG for CFL $A$. Let $b \geq 2$ be the maximum number of RHS symbols in any rule. Any node in a parse tree has $\leq b$ children. The depth of the tree is proportional to this branching factor $b$: At depth $d$ from the root, there are at most $b^d$ leaves, i.e., the longest string that can be generated after $d$ rules of the grammar has size $\leq b^d$. Then if a generated string is at least $b^d + 1$ long, its parse tree must be at least $d + 1$ deep.

If $|V|$ is the number of variables in $G$, let $p = b^{|V|+1}$. If $s \in A$ and $|s| \geq p$, then its parse tree must have depth $\geq |V| + 1$, as $b^{|V|+1} \geq b^{|V|} + 1$.

Now consider a parse tree $T$ for $s$ (if more than one, choose the one with fewest nodes). $T$ must be at least $|V| + 1$ deep, so its longest root-to-leaf path has length at least $|V| + 1$. But this path has at least $|V| + 2$ nodes and only the leaf is a terminal, with the others variables. Then some variable $R$ appears more than once on that path. If more than one repeat, choose $R$ to be a variable that repeats among the lowest $|V| + 1$ variables on this path. □

# Proving a Language Non-Context-Free

- To prove a language is context-free we build a PDA, or a CFG
- To prove a language is not context-free, we use the PL

**Claim**

$L = \{0^n 1^n 0^n | n \geq 0\}$ *is not a context-free language*

First, let's get some intuition:

# Proving a Language Non-Context-Free

- To prove a language is context-free we build a PDA, or a CFG
- To prove a language is not context-free, we use the PL

**Claim**

$L = \{0^n 1^n 0^n | n \geq 0\}$ *is not a context-free language*

First, let's get some intuition:

- Can we use the stack to match 0s and 1s and then 1s and 0s?

# Proving a Language Non-Context-Free

- To prove a language is context-free we build a PDA, or a CFG
- To prove a language is not context-free, we use the PL

**Claim**

$L = \{0^n 1^n 0^n | n \geq 0\}$ *is not a context-free language*

First, let's get some intuition:

- Can we use the stack to match 0s and 1s and then 1s and 0s?
- How about we push two 0s for each 0 and then match the first $n$ 0s to $n$ 1s and then the remaining 0s to the trailing 0s?

# Proving a Language Non-Context-Free

- To prove a language is context-free we build a PDA, or a CFG
- To prove a language is not context-free, we use the PL

**Claim**

$L = \{0^n1^n0^n | n \geq 0\}$ *is not a context-free language*

First, let's get some intuition:

- Can we use the stack to match 0s and 1s and then 1s and 0s?
- How about we push two 0s for each 0 and then match the first $n$ 0s to $n$ 1s and then the remaining 0s to the trailing 0s?
- What language would such a PDA recognize?

001100 ✓

001111 ✓ ✗

# Proving a Language Non-Context-Free

$v = 0$ $y = 1$

## Claim

$L = \{0^n 1^n 0^n \mid n \geq 0\}$ *is not a context-free language*

## Proof.

Suppose $L$ is context-free. Then the PL applies. Let $p$ be the pumping length. Then consider the string $s = 0^p 1^p 0^p$ and all possible ways to break $s$ down into $s = \cancel{xuvyz}$ $uvxyz$ subject to the 3 PL conditions.

- Consider all cases depending on what $v$ and $y$ might contain
- Show that in all of these cases, pumping up results in a string that is not in $L$.

$\square$

$\boxed{3)\ |vxy| \leq p}$

$2)\ |vy| > 0$

$\overbrace{00\ldots0}^{p}\overbrace{11\ldots1}^{p}\overbrace{00\ldots0}^{p}$

1

2

# Proving a Language Non-Context-Free

**Claim**

$L = \{ww | w \in \{0,1\}^*\}$ *is not a context-free language.*

- Recall that $L = \{ww^R | w \in \{0,1\}^*\}$ is indeed a CFL
- The PDA memory restriction makes it easy to recognize $ww^R$
- Why not $ww$?

**Proof.**

- Let's try to use the PL to show that $L$ is not a CFL

# Proving a Language Non-Context-Free

**Claim**

$L = \{ww | w \in \{0,1\}^*\}$ *is not a context-free language.*

**Proof.**

- Assume $L$ is a CFL and obtain a contradiction with the PL
- Let $p$ be the pumping length given by the pumping lemma
- We now get to choose a string $s \in L$ of length $\geq p$
- How about $s = 0^p10^p1$?
- Clearly, $s \in L$ and has length greater than $p$, so it appears to be a good candidate
- However, $s$ **can** be pumped by dividing it as follows!

$$\overbrace{\underbrace{000\cdots000}_{u}\;\underbrace{0}_{v}\;\underbrace{1}_{x}}^{0^p1}\;\overbrace{\underbrace{0}_{y}\;\underbrace{000\cdots0001}_{z}}^{0^p1}$$

# Proving a Language Non-Context-Free

**Claim**

$L = \{ww | w \in \{0,1\}^*\}$ *is not a context-free language.*

**Proof.**

- Assume $L$ is a CFL and obtain a contradiction with the PL
- Let $p$ be the pumping length given by the pumping lemma
- We choose $s = 0^p 1^p 0^p 1^p$
- Clearly, $s \in L$ and has length greater than $p$
- Condition 3 of the PL restricts the ways that $s$ can be divided ($s = uvxyz$, where $|vxy| \leq p$)
- if the substring $xyz$ occurs only in the first 0s of $s$, pumping $s$ up to $uv^2xy^2z$ creates more 0s on the left than on the right
- if the substring $xyz$ occurs only in the first 1s of $s$, pumping $s$ up to $uv^2xy^2z$ creates more 1s on the left than on the right
- symmetrically if substring $xyz$ is on the right with only 0s or only 1s

# Proving a Language Non-Context-Free

**Claim**

$L = \{ww | w \in \{0,1\}^*\}$ *is not a context-free language.*

**Proof.**

- if the substring $xyz$ contains 0s and 1s in the first half of $s$, pumping $s$ up to $uv^2xy^2z$ creates a different string on the left than on the right (not $ww$)
- if the substring $xyz$ contains 1s and 0s and straddles the middle of $s$, then $uv^2xy^2z$ creates a different number of 0s and 1s in the left and right parts

□