Monday

Three different CFGs that each generate the language $\{abba\}$

$$(\{S, T, V, W\}, \{a, b\}, \{S \to aT, T \to bV, V \to bW, W \to a\}, S)$$

"Left to right" ~~sample~~ only derivation

$S \Rightarrow aT \Rightarrow abV \Rightarrow abbW \Rightarrow abba$

$$(\{Q\}, \{a, b\}, \{Q \to abba\}, Q)$$

" all at once" ~~sample~~ only derivation

$Q \Rightarrow abba$

$$(\{X, Y\}, \{a, b\}, \{X \to aYa, Y \to bb\}, X)$$

"inside out" ~~sample~~ only derivation

$X \Rightarrow aYa \Rightarrow abba$

Design a CFG to generate the language $\{a^n b^n \mid n \geq 0\}$ $G = (\{S\}, \{a, b\}, R, S)$

where rules are $\underset{\text{var}}{S} \to \underline{\overset{1}{aSb}} \mid \underline{\overset{2}{ab}} \mid \underline{\overset{3}{\varepsilon}}$

Sample derivations:

$S \overset{2}{\Rightarrow} ab$ so have $ab \in L(G)$

$S \overset{3}{\Rightarrow} \varepsilon$ so have $\varepsilon \in L(G)$

$S \overset{1}{\Rightarrow} aSb \overset{1}{\Rightarrow} aaSbb \overset{1}{\Rightarrow} aaaSbbb \overset{3}{\Rightarrow} aaabbb$ so have $a^3b^3 \in L(G)$.

Design a CFG to generate the language $\{a^i b^j \mid j \geq i \geq 0\}$

$G_1 = (\{S\}, \{a, b\}, R_1, S)$

R_1: $S \to \overset{1}{aSb} \mid \overset{2}{Sb} \mid \overset{3}{\varepsilon}$

It's tempting to try $S \to aSb \mid b \mid \varepsilon$ BUT NOT ENOUGH OPPORTUNITIES TO ADD B'S

Sample derivation:

$S \overset{1}{\Rightarrow} aSb \overset{2}{\Rightarrow} aSbb \overset{1}{\Rightarrow} aaSbbb \overset{3}{\Rightarrow} aabbb$

Theorem 2.20: A language is generated by some context-free grammar if and only if it is recognized by some push-down automaton.

Definition: a language is called **context-free** if it is the language generated by a context-free grammar. The class of all context-free language over a given alphabet Σ is called **CFL**.

Consequences:

- Quick proof that every regular language is context free
- To prove closure of the class of context-free languages under a given operation, we can choose either of two modes of proof (via CFGs or PDAs) depending on which is easier
- To fully specify a PDA we could give its 6-tuple formal definition or we could give its input alphabet, stack alphabet, and state diagram. An informal description of a PDA is a step-by-step description of how its computations would process input strings; the reader should be able to reconstruct the state diagram or formal definition precisely from such a descripton. The informal description of a PDA can refer to some common modules or subroutines that are computable by PDAs:
  - PDAs can "test for emptiness of stack" without providing details. *How?* We can always push a special end-of-stack symbol, \$, at the start, before processing any input, and then use this symbol as a flag.
  - PDAs can "test for end of input" without providing details. *How?* We can transform a PDA to one where accepting states are only those reachable when there are no more input symbols.

NOT

Over $\Sigma = \{a, b\}$, let $L = \{a^n b^m \mid n \neq m\}$. **Goal**: Prove L is context-free.

Rewrite $L = \underbrace{\{ a^n b^n b^i \mid n \geq 0, i > 0 \}}_{L_1} \cup \underbrace{\{ a^i a^n b^n \mid n \geq 0, i > 0 \}}_{L_2}$

setwise concatenation

union

setwise concatenation

Strategy: use general constructions for union and concatenation of languages generated by CFGs to build up our CFG.

Extra practice: informal description of PDA that recognizes L_1
"Read a's and push them onto stack, when see first b, pop a off the stack for each b we read and when stack is empty, if the rest of the input string has positive number of b's (and nothing else), accept (otherwise, reject).

Suppose L_1 and L_2 are context-free languages over Σ. **Goal**: $L_1 \cup L_2$ is also context-free.

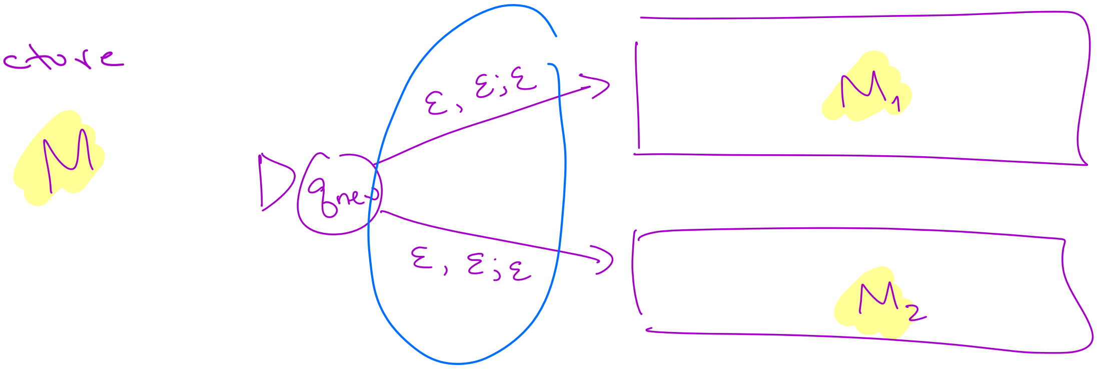
Approach 1: with PDAs

Let $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, F_2)$ be PDAs with $L(M_1) = L_1$ and $L(M_2) = L_2$.

Define $M =$

(details left as exercise/see textbook)

Big picture


M
q_{new}
$\varepsilon, \varepsilon; \varepsilon$
$\varepsilon, \varepsilon; \varepsilon$
M_1
M_2

Computation starts at q_{new} with empty stack $\xrightarrow{\text{1step}}$ move to start state of M_1 or M_2

derivations of strings in L_1

derivations of strings in L_2

Approach 2: with CFGs

Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ be CFGs with $L(G_1) = L_1$ and $L(G_2) = L_2$.

Define $G = (V_1 \cup V_2 \cup \{S_{new}\}, \Sigma, R_1 \cup R_2 \cup \{S_{new} \to S_1 | S_2\}, S_{new})$

①

$S_1 \Rightarrow \_\_ \Rightarrow \dots$

Assume.

$V_1 \cap V_2 \neq \emptyset$

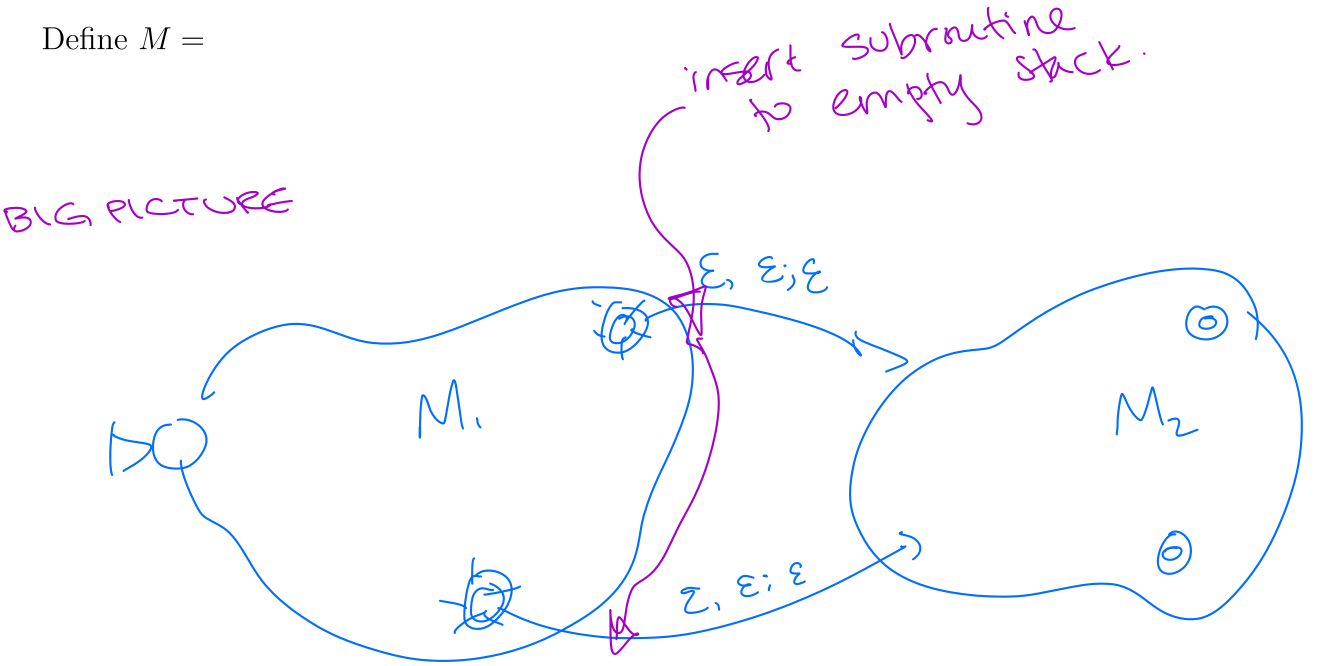
$S_{new} \notin V_1 \cup V_2$

Suppose L_1 and L_2 are context-free languages over Σ. **Goal**: $L_1 \circ L_2$ is also context-free.

Approach 1: with PDAs

Let $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, F_2)$ be PDAs with $L(M_1) = L_1$ and $L(M_2) = L_2$.

Define $M =$



$$L_1 \circ L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$$

Approach 2: with CFGs

Let $G_1 = (\overset{\text{vars}}{V_1}, \Sigma, \overset{\text{rules}}{R_1}, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ be CFGs with $L(G_1) = L_1$ and $L(G_2) = L_2$.

Define $G =$ $(V_1 \cup V_2 \cup \{S_{new}\}, \Sigma, R_1 \cup R_2 \cup \{S_{new} \to S_1 S_2\}, S_{new})$

Assume
$V_1 \cap V_2 = \emptyset$
$S_{new} \notin V_1 \cup V_2$

$$S_{new} \to S_1 \, S_2$$

start w/ string derived from S_1

end w/ string derived from S_2

Example: $\{a^n b^n b^i \mid n \geq 0, i \geq 0\} = \{a^n b^n \mid n \geq 0\} \circ \{b^i \mid i \geq 0\}$

Grammar generating $\{a^n b^n \mid n \geq 0\}$ $(\{S\}, \{a, b\}, \{S \to aSb \mid \varepsilon\}, S)$

Grammar generating $\{b^i \mid i \geq 0\}$ $(\{T\}, \{a, b\}, \{T \to b \mid Tb\}, T)$

Grammar generating $\{a^n b^n \mid n \geq 0\} \circ \{b^i \mid i \geq 0\}$

$$(\{S, T, S_{new}\}, \{a, b\}, \{S \to aSb \mid \varepsilon, T \to b \mid Tb, S_{new} \to ST\}, S_{new})$$

Summary

Over a fixed alphabet Σ, a language L is **regular**

iff it is described by some regular expression
iff it is recognized by some DFA
iff it is recognized by some NFA

Over a fixed alphabet Σ, a language L is **context-free**

iff it is generated by some CFG
iff it is recognized by some PDA

Fact: Every regular language is a context-free language.

Fact: There are context-free languages that are (typo) ~~not~~ nonregular.

Fact: There are countably many regular languages.

Fact: There are countably inifnitely many context-free languages.

Consequence: Most languages are **not** context-free!

Examples of non-context-free languages

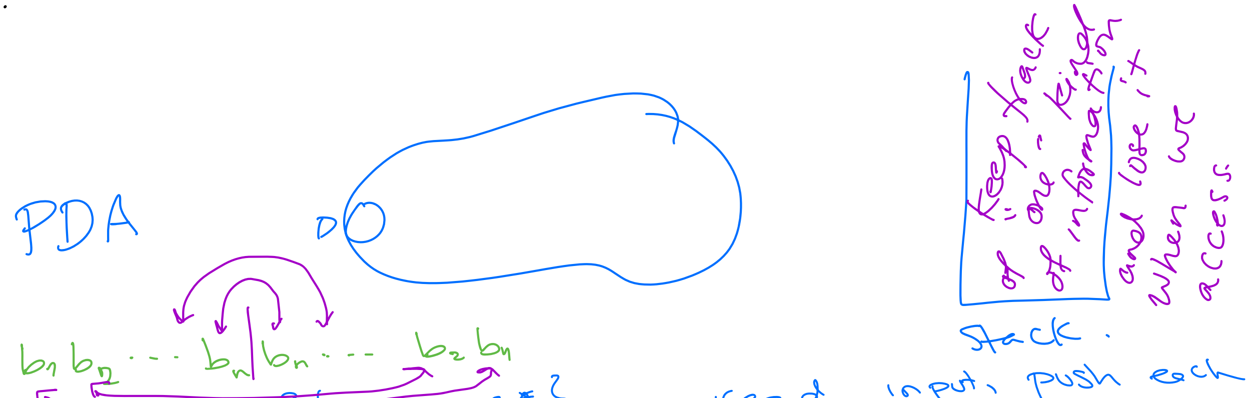
Need to access "n" twice to detect this pattern!

$$\{a^n b^n c^n \mid 0 \leq n, n \in \mathbb{Z}\}$$
$$\{a^i b^j c^k \mid 0 \leq i \leq j \leq k, i \in \mathbb{Z}, j \in \mathbb{Z}, k \in \mathbb{Z}\}$$
$$\{ww \mid w \in \{0,1\}^*\}$$

(Sipser Ex 2.36, Ex 2.37, 2.38)

$b_1 b_2 - - - b_n \; b_1 b_2 \cdot - - \; b_n$

There is a Pumping Lemma for CFL that can be used to prove a specific language is non-context-free: If A is a context-free language, there there is a number p where, if s is any string in A of length at least p, then s may be divided into five pieces $s = uvxyz$ where (1) for each $i \geq 0$, $uv^ixy^iz \in A$, (2) $|uv| > 0$, (3) $|vxy| \leq p$. *We will not go into the details of the proof or application of Pumping Lemma for CFLs this quarter.*


PDA
Keep track of one-kind of information
and lose it when we access
stack.
$b_1 b_2 \cdots b_n | b_n \cdots b_2 b_1$

$\{ww^R \mid w \in \{0,1\}^*\}$: read input, push each char to stack, at some point guess we've reached middle and compare to top of stack and match input char

Review: Week 5 Monday

Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

Please complete the review quiz questions on Gradescope about context-free grammars

Pre class reading for next time: Figure 3.1 (Pages 165-167)

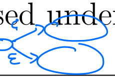
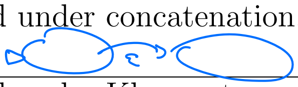
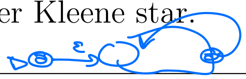
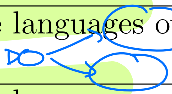
Suggested fixes for the limitations of PDAs

① 2 stacks.

② queue instead of stack

Wednesday

A set X is said to be **closed** under an operation OP if, for any elements in X, applying OP to them gives an element in X.

True/False	Closure claim
True	The set of integers is closed under multiplication. $\forall x \forall y \, (\, (x \in \mathbb{Z} \land y \in \mathbb{Z}) \to xy \in \mathbb{Z} \,)$
True	For each set A, the power set of A is closed under intersection. $\forall A_1 \forall A_2 \, (\, (A_1 \in \mathcal{P}(A) \land A_2 \in \mathcal{P}(A) \in \mathbb{Z}) \to A_1 \cap A_2 \in \mathcal{P}(A) \,)$
True	The class of regular languages over Σ is closed under complementation. Using DFA: "Flip accept/reject"
True	The class of regular languages over Σ is closed under union. Using DFA: Cartesian Product Using NFA  using regex: $R_1 \cup R_2$
True	The class of regular languages over Σ is closed under intersection. Using DFA: Cartesian Product
True	The class of regular languages over Σ is closed under concatenation. Using NFAs  $R_1 \circ R_2$
True	The class of regular languages over Σ is closed under Kleene star. Using NFAs  R_1^*
False	The class of context-free languages over Σ is closed under complementation.
True.	The class of context-free languages over Σ is closed under union. Using PDA  Using CFGs
False.	The class of context-free languages over Σ is closed under intersection.
True	The class of context-free languages over Σ is closed under concatenation. Using CFGs
True.	The class of context-free languages over Σ is closed under Kleene star. Using CFGs.

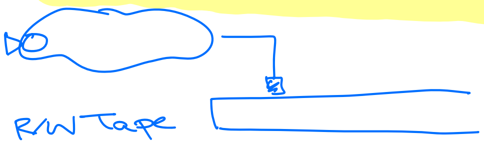
Extra practice:

Assume $\Sigma = \{0, 1, \#\}$

Σ^*	Regular /	nonregular and context-free /	not context-free
$\{0^i \# 1^j \mid i \geq 0, j \geq 0\}$	Regular /	nonregular and context-free /	not context-free
$\{0^i 1^j \# 1^j 0^i \mid i \geq 0, j \geq 0\}$	Regular /	nonregular and context-free /	not context-free
$\{0^i 1^j \# 0^i 1^j \mid i \geq 0, j \geq 0\}$	Regular /	nonregular and context-free /	not context-free

So what's next ? (Ch 3.1)

Turing machines: unlimited read + write memory, unlimited time (computation can proceed without "consuming" input and can re-read symbols of input)



- Division betweeen program (CPU, state diagram) and data
- Unbounded memory gives theoretical limit to what modern computation (including PCs, supercomputers, quantum computers) can achieve
- State diagram formulation is simple enough to reason about (and diagonalize against) while expressive enough to capture modern computation

For Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ the **computation** of M on a string w over Σ is:

- Read/write head starts at leftmost position on tape.
- Input string is written on $|w|$-many leftmost cells of tape, rest of the tape cells have the blank symbol. **Tape alphabet** is Γ with $\text{\textvisiblespace} \in \Gamma$ and $\Sigma \subseteq \Gamma$. The blank symbol $\text{\textvisiblespace} \notin \Sigma$.
- Given current state of machine and current symbol being read at the tape head, the machine transitions to next state, writes a symbol to the current position of the tape head (overwriting existing symbol), and moves the tape head L or R (if possible). Formally, **transition function** is

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

- Computation ends if and when machine enters either the accept or the reject state. This is called **halting**. Note: $q_{accept} \neq q_{reject}$.

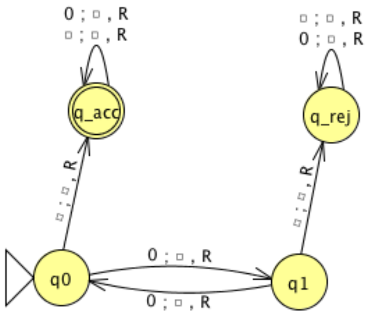
The **language recognized by the Turing machine** M, is

$\{w \in \Sigma^* \mid \text{computation of } M \text{ on } w \text{ halts after entering the accept state}\} = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$

deferred to Monday.

An example Turing machine: $\Sigma =$, $\Gamma =$

$\delta((q0, 0)) =$



Formal definition:

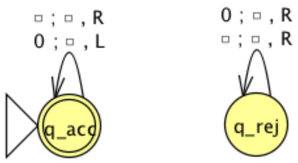
Sample computation:

$q0 \downarrow$						
0	0	0	␣	␣	␣	␣

deferred to Monday

The language recognized by this machine is . . .

Extra practice:



Formal definition:

Sample computation:

Review: Week 5 Wednesday

Please complete the review quiz questions on Gradescope about context-free languages

Friday: Midterm exam

The midterm exam policies are below

INSTRUCTIONS — READ THIS NOW

- Write your name, PID, current seat number, exam time, and write out the academic integrity pledge in the indicated space above. Your score **will not be recorded** if any of this identifying information is missing, or if the academic integrity pledge is illegible.
- Write your answers on the designated **answer sheet** in the specified areas, or your work will not be graded. If you need more space, raise your hand.
- We will not be answering questions about the exam during the exam period. If any bugs are found in the exam after the exam period, the affected question(s) will be removed.
- Please show your ID to a proctor when you hand in your exam.
- You may not speak to any other student in the exam room while the exam is in progress (including after you hand in your own exam). You may not share **any information** about the exam with anyone who has not taken it.
- Turn off and put away all cellphones, calculators, and other electronic devices. You may not access any electronic device during the exam period. You may use your one sheet of notes, but no other books, notes, or aids.
- To receive full credit, your answers must be written neatly, legibly, and sufficiently darkly to scan well. Your solution will be evaluated both for correctness and clarity. Read the instructions for each part carefully to determine what is required for full credit.
- This exam is **45 minutes** long. Read all the problems first before you start working on any of them, so you can manage your time wisely.
- This test has 3 problems worth a total of 50 points.