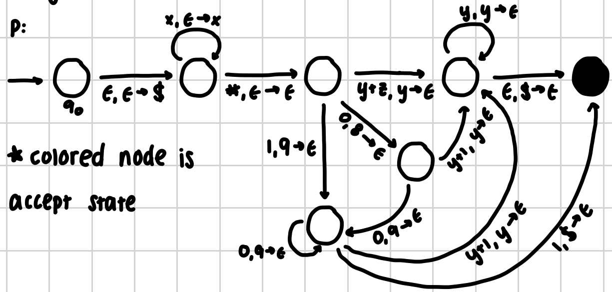
1. Let $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a pushdown automata that recognizes the language $L = \{N(i) \# N'(i+2) : i \geq 1\}$. We can let $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \#\}$. Assume $N(i)$ is simplified and contains no leading 0s and $N'(i)$ does not end in insignificant 0s. Let x represent the current character on the tape and let y represent the top of the stack. We construct P as shown below. Notice we start by pushing a $ on to start the stack. We then loop through N(i) and push each value of N(i) on to the stack. We move on to the next state when # is read. There are three options to account for cases when y (top of the stack) reads 8 or 9 — where $i+2$ would end with 0 or 1. In the case we read $y+2$ from the tape, we pop the top of the stack. We will continue to pop if the top of the stack matches the tape until the tape is empty and the stack is empty, in which case the string is accepted. If the tape reads 0 or 1, we have states to account for the top of the stack being 8, 9. ∴ we account for scenarios like $N(i) = 98$ and $N'(i+2) = 100$ as shown in the diagram for P. We have shown that the given language L is recognized by the pushdown automaton P we built.

P:



* colored node is accept state

2. The language described by the given context-free grammar, which we will call L(G), describes the language where the number of a's and b's are the same.
   1. We will start by proving that every string $w \in L(G)$ has the same number of a's and b's. Proof by induction on the number of deviations. For the base case, consider a deviation of length 1 | $S \to^1 \epsilon$ is the only solution for a deviation of length 1. ϵ has the same number of a's and b's or 0 of each so this fulfills our language grammar. Inductive hypothesis: assume that the statement every string $w \in L(G)$ has the same number of a's and b's is true for all deviations of length less than k. Using strong induction, consider a deviation of length k or $S \to^k w$ | there are three possibilities:
      i) $S \to^1 aSb \to^{(k-1)} aw'b = w$ where w' is derived from S in k-1 or less steps | based on the inductive hypothesis w' must be balanced or have the same number of a's and b's. ∴ since w' is balanced and $w = aw'b$ then clearly w has the same number of a's and b's | $w \in L(G)$.
      ii) $S \to^1 bSa \to^{(k-1)} bw'a = w$ where w' is derived from S in k-1 or less steps | based on the inductive hypothesis w' must be balanced or have the same number of a's and b's. ∴ since w' is balanced and $w = bw'a$ then clearly w has the same number of a's and b's | $w \in L(G)$.
      iii) $S \to^1 SS \to^{k-1} w'w'' = w$ where w' is derived from the first S in k-1 or less steps | based on the inductive hypothesis it must contain the same number of a's and b's and where w'' is derived from the second S in k-1 fewer steps | by our inductive hypothesis $w'' \in L(G)$. Since $w = w'w''$, $w \in L(G)$.

   ∴ every string $w \in L(G)$ has equal number of a's and b's.

2. 2. Now we'll prove the back direction: every string w with equal number of a's and b's is in $L(G)$. First we will prove that the shortest non-empty prefix of x that is in the language cannot begin and end with the same symbol. Proof by contradiction. Consider $x \in L(G)$ where $x = nm$ where n is the shortest prefix where $n \in L(G)$. Let $f(s) = \#a\text{'s} - \#b\text{'s}$ where s represents the length of the prefix $\mid f(|n|) = 0$. If n begins with an a then $f(1) = 1$ and $f(|n|-1) = -1$. This implies that $\exists$ some string s with length between 1 and $|n|-1$ where $f(|s|) = 0$. $\Rightarrow\Leftarrow$ as we initially stated that n is the shortest prefix. Similarly if n begins with a b then $f(1) = -1$ and $f(|n|-1) = 1 \mid \exists$ some string s of length $1 < |s| < |n|-1$ where $f(|s|) = 0 \mid$ we get another contradiction as we initially assumed n was the shortest prefix. $\therefore$ we have shown that the shortest non-empty prefix of x in the language cannot begin and end with the same symbol.

3. We can now do induction on the length of the string to prove that every other string w with an equal number of a's and b's is in $L(G)$. For our base case, let the length of the string w be $0 \mid w = \epsilon$ for all strings of length 0. Since this deviation exists in $L(G)$, then $w \in L(G)$. Inductive hypothesis: assume that the statement every string w with equal number of a's and b's is in $L(G)$ is true for all $|w|$ less than k. Using strong induction, consider the shortest prefix in $w \in L(G)$ where $|w| = k \mid$ there are three possibilities:

i) If $\exists$ a shortest prefix of w, we can express $w = w'w''$ where w' is the shortest prefix where $c \in L(G)$. $\therefore$ we have a proper prefix, so we can build w with $S \to' SS \to^* w'w'' = w$ where $w' = aSb$ or $w' = bSa$ since we know the shortest prefix of w cannot begin and end with the same symbol. Since w' is the shortest prefix in $L(G)$ and w'' is built from the second S, then w'' must have a length shorter than $|w| \mid$ by our inductive hypothesis $w', w'' \in L(G)$. $\therefore w \in L(G)$.

ii) If the shortest prefix of w in $L(G)$ is the whole string (or $\exists$ no proper prefix) then there are two ways to build w. One is $S \to' aSb \to^* aw'b = w$ where since w' has length shorter than w; so based on the inductive hypothesis, $w' \in L(G) \mid w \in L(G)$. The second way is $S \to' bSa \to^* bw'a = w$ where since w' has length less than w; by our inductive hypothesis $w' \in L(G)$ $\therefore$ $w \in L(G)$. By strong induction, every string w with equal number of a's and b's is in $L(G)$.

$\therefore S \to aSb \mid bSa \mid SS \mid \epsilon$ does generate a context-free language where the number of a's and b's are equal.

3. a) Consider context-free grammar where $A = (V, \Sigma, R, S)$. We have non-terminals $V = \{X, Y, S\}$ we let the alphabet $\Sigma = \{a, b, c\}$. Let the start variable $S = \{S\}$. Let R be productions $S \to XY$, $X \to aXb \mid \epsilon$, $Y \to YY \mid C \mid \epsilon$. We can define this context-free language based off of the grammar $\mid$ $A = (a^n b^n c^r, r > n \geq 0)$. Consider now the context-free grammar $B = (V_B, \Sigma_B, R_B, S_B)$. We have non-terminals $V_B = \{X, Y, S\}$. We let the alphabet $\Sigma_B = \{a, b, c\}$. Let the start variable $S_B = \{S\}$. Let R be productions $S \to YX$, $X \to bXc \mid \epsilon$, $Y \to YY \mid a \mid \epsilon$. We can define this context free language based on the grammar $\mid$ $B = (a^r b^n c^n, r > n \geq 0)$. Since we have defined context free grammars in Chomsky Normal Form for both A and B, A and B must both be context free languages.

b) Given $A = a^n b^n c^r$ and $B = a^r b^n c^n$, $C = A \cap B = a^n b^n c^n$. Assume C is a CFL, we will use the pumping lemma. pumping length $= p$ and string $s = a^p b^p c^p \mid s \in C$ and with length $\geq p$. Given $|vy| > 0$, $|vxy| \leq p$, there are two scenarios:

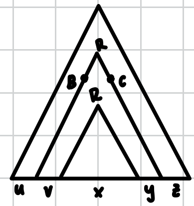
1. When both v and y contain only one character in the alphabet. In this case, string $w = uv^2xy^2z$ will not contain the same number of a's, b's, and c's and $\therefore w \notin C$. $\Rightarrow\Leftarrow$.
2. When either v or y contain >1 character in the alphabet, $w = uv^2xy^2z$ may contain equal numbers of a's, b's, and c's, but in an incorrect order so $w \notin C$. $\Rightarrow\Leftarrow$.

Since both cases lead to contradictions, C is not a CFL and CFLs are not closed under intersection.

4) a) Given that $L = \{a^i b^j c^k d^l : i = 0 \text{ or } j = k = l\}$ we have two scenarios for the CFL pumping lemma where p is the pumping length of L.

1. Consider $i = 0 \mid w = bcd$ where $|w| > p$ and $w \in L$. Considering the conditions $|vy| > 0$ and $|vxy| \leq p$ no matter how we split w into $uvxyz$ since no a is in w then no matter how we pump $w = uv^i xy^i z$ we will have $w \in L$.
2. Consider when $j = k = l \mid w = a^p bcd$ where $|w| = p+3 > p$ and $w \in L$. Considering the conditions $|vy| > 0$ and $|vxy| \leq p$ we can split w into $uvxyz$ where $u = \epsilon$, $v = a^i$, $x = a^j$, $y = a^k$ and $z = bcd$ $\mid$ pumping in any way on $w = uv^i xy^i z$ will only increase the number of a's $\mid$ $w \in L$ because the number of b, c, d are the same.

4) b) Let G be in Chomsky Normal Form | we can consider a parse tree for w where $|w| \geq 2^{k+1} = p$ where k is the number of variables in G and $\exists$ at least p marked positions in w. There exists a path with at least k+1 branch points in its path. This is true because as we move down the parse tree, if we pick the direction with the most markings then we are picking the node that has at least half of the markings that its ancestor had. As a result, we are dividing by at most 2 each time we follow nodes down the parse tree $\therefore$ there will be at least k+1 branch points along the way. Since we defined the grammar to have k variables but we have at least k+1 branch points along the parse tree, some variables must repeat.



1. The proof for all $i \geq 0$, $uv^ixy^iz \in L$ is the same as the proof for the CFL pumping lemma. We divide w into uvxyz as shown, where each occurrence of R has a subtree under it, generating a part of the string w. The upper occurrence of R has a larger subtree and generates vxy. The lower occurrence of R has a smaller subtree and generates x. Both subtrees are generated by the same variable | substituting one for the other still yields a valid parse tree. Replacing the smaller with the larger repeatedly gives parse trees for the strings uv^ixy^iz at each $i > 1$. Replacing the larger with the smaller generates the string uxz where $i = 0$. $\therefore$ we have shown that for all $i \geq 0$ then $uv^ixy^iz \in L$.
2. To prove that vy contains at least 1 marked position of w we will consider a subtree of the parse tree for L. Let R represent the top branch of the subtree | B and C are children under R. Then B and C must have marked descendants since there are no other children under R. B must see at least one character of v and C must see at least one character of y | vy must contain at least 1 marked position of w.
3. To prove vxy contains at most p marked positions of w, we will let R represent the variable with the lowest repeating variables among the remaining branch points of a subtree. Notice that when we move up a parse tree, whenever we hit branch points, the number of markings increases. Since $p = 2^{k+1}$ which is the highest the parse tree can be, then the number of marked under the upper occurrence of R or vxy can't contain more than p marked positions of w.

c) For the string $s = a^0b^Pc^Pd^P$ in L where p is the pumping length given by Ogden's Lemma and $i = 0$ and $j = k = l = p$ to satisfy the second condition in L's definition, we need to split s into $s = uvxyz$ such that $|vxy| \leq p$ and $|vy| > 0$. Since s consists of three blocks with p symbols, v and y must be selected from the middle portion of the string — c^P and d^P. By pumping the segments v and y, the new string will have more c's and d's than b's since v and y contain only c's and d's. The new string no longer meets $j = k = l$ since the number of b's, c's, and d's are not the same. The new string is not in L. Since the new string we got from pumping s leads to a string not in L, the lemma is violated and L is not a CFL.

5) We can say that there are 2 languages: L_1 and L_2. We know that CFLs are not closed under intersection. This is because we have a PDA, which we will call A_1, that accepts L_1 and a PDA, which we will call A_2, that accepts L_2. Finding a PDA that accepts the intersection of L_1 and L_2 is possible, but there will not be a PDA that rejects the parts of L_1 and L_2 that are not in the intersection. $\therefore L_1 \cup L_2 = \overline{\bar{L_1} \cap \bar{L_2}}$. CFLs would only be closed under complement if CFLs were closed under intersection, however CFLs are not closed under intersection $\therefore$ CFLs are not closed under complement.