In no way we can choose vwx from $0^m1^m0^m1^m$ such that all the strings from uv^iwx^iy is in L.

Using PL show that { $0^n1^n2^n$, n >0 } is not a CFL.

Decision properties of CFLs

- Testing Emptiness
  - Eliminate variables that derive no terminal strings.
- If the start symbol is one of these then, the CFL is empty otherwise not.

Testing Membership

- Is string w in CFL L?

✓ Assume G is in CNF
✓ $w = \epsilon$ is a special case, solved by testing if the start symbol is nullable.
✓ Algorithm CYK is a good example of Dynamic Programming and runs in time $O(n^3)$ where $n=|w|$

Testing infiniteness

- The idea is essentially same as for regular languages
  - Use the PL constant n
  - If there is a string in the language of length between n and 2n-1 then the language is infinite otherwise not.

Non-decision properties

- Many questions that can be decided for regular sets cannot be decided for CFLs
  - Example: Are two CFLs same?
  - Example: Are two CFLs disjoint.

~~Need~~ With the theory of Turing Machines and Decidability ~~to~~ we can prove no such algorithm exists.

Closure of CFLs Under Union

- Let L and M be CFLs with grammars G and H respectively,
- Assume G and H have no variables in common.
- Let S_1 and S_2 be the two start symbols of G and H respectively.
- Form a new grammar L U M by combining all symbols and productions of G and H
- Then add a new start symbol S
- Add production $S \rightarrow S_1 \mid S_2$

✓ In the new grammar, all derivations start with S
✓ The first step replaces S by either S_1 or S_2
✓ In the first case the result will be a string in L(G)=L
✓ In the second case, the result must be a string L(H)= M

Closure of CFLs under Concatenation

$S \rightarrow S_1S_2$

Closure of CFLs under Star

- Let L have a grammar G, with start symbol S.
- Form a new grammar L* by introducing to G a new start symbol and the productions $S \rightarrow S_1S \mid \varepsilon$
- A rightmost ~~generation~~ derivation from S generates a sequence of zero or more S_1 which generates some string in L.

Closure of CFLs under Reversal

Let L ~~=~~ be a CFL with a grammar G.
~~$L^R = ?$~~
$(L^R)^R = L$

Form a grammar for L^R by reversing the right side of every production in G.

Example: Let G have $S \rightarrow 0S1 \mid 01$
The reversal of L(G) has grammar
$S \rightarrow 1S0 \mid 10$

Closure of CFLs under Homomorphism

- Let L be a CFL with grammar G
- Let ~~H~~ h be a homomorphism on the terminal symbols of G
- Construct a grammar for h(L) by replacing each terminal symbol a by h(a).
- Example : G has productions $S \rightarrow 0S1 \mid 01$
  - ~~H~~ h is defined by h(0)= ab, h(1) = ε
  - h(L(G)) has the grammar with $S \rightarrow abS \mid ab$

Date January 2, 2021

Closure of CFLs under Inverse Homomorphism

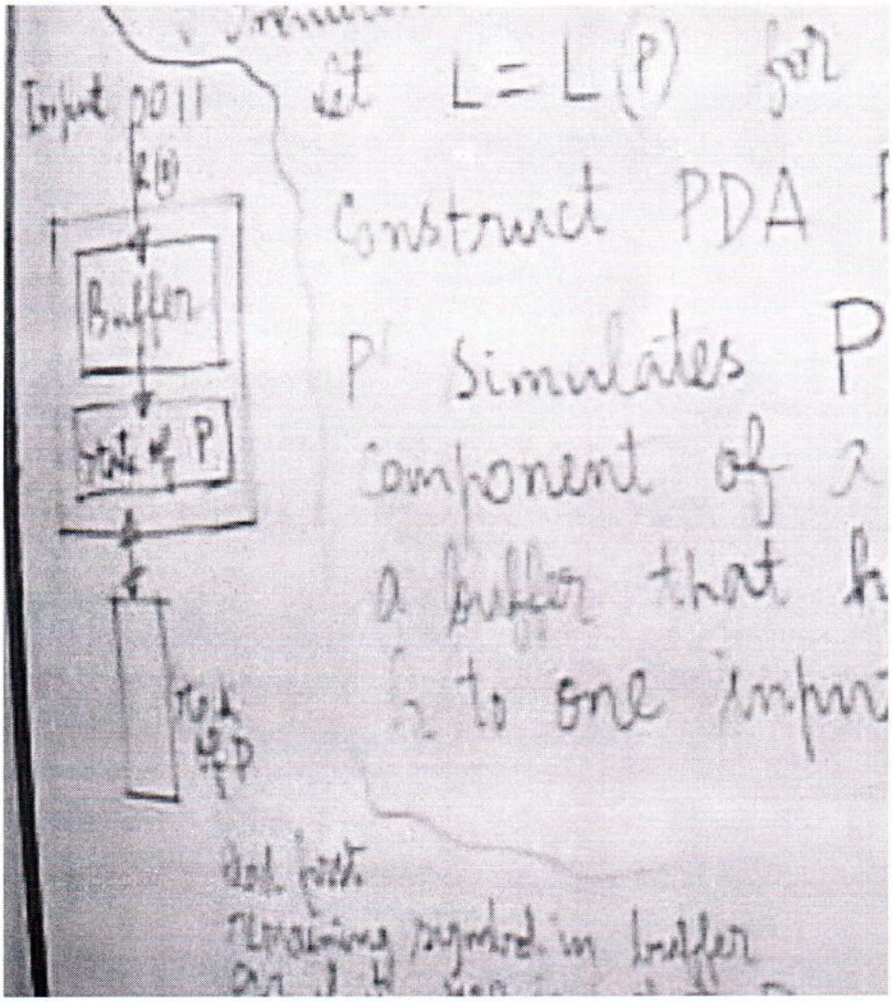
Inverse Homomorphism

- If h is a homomorphism and L is any language then **h^{-1} (L)** is the set of strings w such that **h(w) is in L**.
  - **h^{-1} (L) = {w | h(w) is in L }.**

- Intuition:

Let L = L(P) for some PDA P. C~~c~~onstruct PDA **P'** to accept h^{-1}(L).

P' simulates P, but keeps , as one component of a two-component state **a buffer** that holds the result of applying h to one input symbol.



Read the first remaining symbol in the buffer as if it were input to P'

- States are pairs [a, b], where q is a state of P
- b is a suffix of h(a) for some symbol a.
- Stack symbols of P' are those of P.
- Start state of P' is $[q_0, \varepsilon]$
- Input symbols of P' are the symbols to which **h** applies.
- Final states of P' are states $[q, \varepsilon]$ such that q is a final state of P

Transitions of P'

1. $\boldsymbol{\delta'([q, \varepsilon], a, X) = \{([q, h(a)], X)\}}$
   For any input symbol **a** of P' and any stack symbol X.
2. $\boldsymbol{\delta'([q, bw], \varepsilon, X)}$ **contains** $\boldsymbol{([p, w], \alpha)}$
   If $\boldsymbol{\delta(\cancel{a}\ q, b, X)}$ contains $\boldsymbol{(p, \alpha)}$ where b is either an input symbol ~~simulates~~ of P or ε ~~from the buffer.~~

   * Simulate P from the buffer.

✓ Non Closure under Intersection

✓ Unlike the Regular languages, the class of CFL's is not under $\cap$.

✓ By Pumping Lemma, we can prove that $L_1 = \{0^n 1^n 2^n \mid n \geq 1\}$ is not a CFL

✓ However, $L_2 = \{0^n 1^n 2^i \mid n \geq 1, i \geq 1\}$ is

✓ CFG $S \rightarrow AB$, $A \rightarrow 0A1 \mid 01$, $B \rightarrow 2B \mid 2$

✓ So is $L_3 = \{0^i 1^n 2^n \mid n \geq 1, i \geq 1\}$

✓ But $L_1 = L_2 \cap L_3$

✓ Nonclosure under Difference

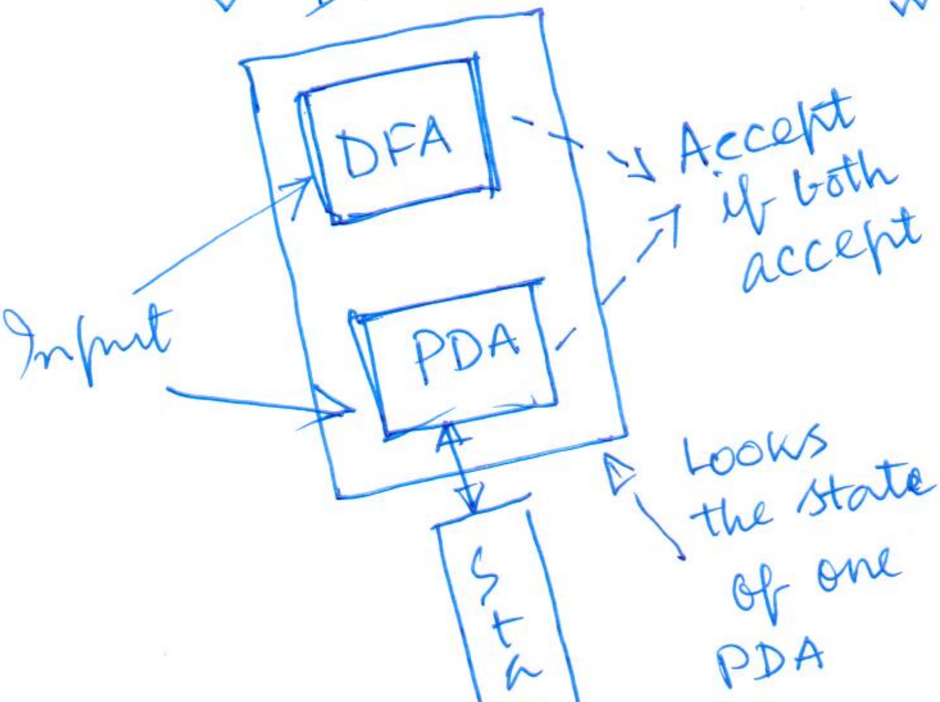
✓ Proof: $L \cap M = L - (L - M)$

Thus if CFL's were closed under Difference, they would be closed under intersection, but they are not

✓ Intersection with Regular Language

- ✓ Intersection of a CFL with a Regular Language is always a CFL.
- ✓ Proof involves running a DFA in parallel with a PDA, and noting that the combination is a PDA.
  - ✓ PDA's accept by final state.

✓ DFA & PDA in Parallel



- Let DFA A has transition function δ_A
- Let PDA P has transition function δ_P
- States of combined PDA are $[q, p]$ where q is a state of A and p is a state of P.
- $\delta([q,p], a, X)$ contains $([\delta_A(q,a), r], \alpha)$ if $\delta_P(p,a,X)$ contains (r,α)

Note: a could be ϵ in which case $\delta_A(q,a)=q$

✓ Accepting States of combined PDA are those $[q, p]$ such that q is an accepting state of A and p is an accepting state of P.

✓ Easy Induction:

$$([q_0, p_0], w, Z_0) \vdash^* ([q, p], \epsilon, \alpha)$$

if and only if $\delta_A(q_0, w) = q$ and in P: $(p_0, w, Z_0) \vdash^* (p, \epsilon, \alpha)$.