

Regular Expressions

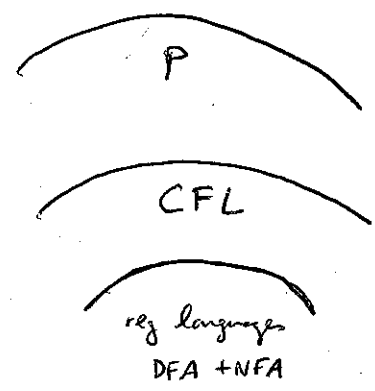
$$(a \cup ba)^*(a \cup b)$$

All Regular expressions can be written with a DFA.

where are we?

The proof is by induction on the smaller parts using the closure properties of DFA's and regular languages. The opposite is also true. All DFA's can be written as reg. ~~exp~~ expressions.

$reg\ exp = DFA = NFA$



Back to  $L = \{a^n b^n \mid n \geq 1\}$

What language is this?

Context Free Language (CFL)

- V: variables
- T: Terminals ← symbols in which final output is written
- P: Production rules ← Tells you how to convert variables into "words" consisting of variables and terminals

$$V \rightarrow w, w \in (V \cup T)^*$$

$S \in V$  : initial symbol

$L = \{ \text{all words in } T^* \text{ that can be generated from } S \text{ using rules in } P \}$

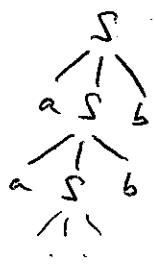
Called "context-free" because we start off with a single symbol and then produce the language by a set of rules. Doesn't matter which other symbols are near S.

$\{a^n b^n\}$

$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

production rules for  $\{a^n b^n\}$



So CFL include non-regular languages.



$vxy$  cannot contain  $a^m b^j c^k$  because this would mean  $|vxy| > k$

4. Now pump.

$uv^2xy^2z$  must also  $\in L$ .

However  $v$  and  $y$  cannot represent all three ~~letters~~ <sup>Symbols</sup> (and cannot both be empty)  
Therefore we are adding some symbols but not others.

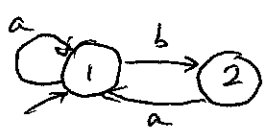
$\therefore uv^2xy^2z \notin L$

We have a contradiction.

$\therefore L$  is not CFL

Are ~~CFL~~ (DFA=NFA)  $\subset$  CFL? YES!

Can we show how to convert an NFA/DFA into a grammar

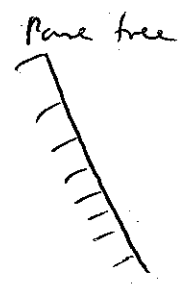


$V = \{1, 2\}$   
 $T = \{a, b\}$   
 $S = 1$

$1 \rightarrow a1, b2, \epsilon$

$2 \rightarrow a1, \epsilon$

1  
a1  
ab2  
~~aba~~ ab a1



EX:

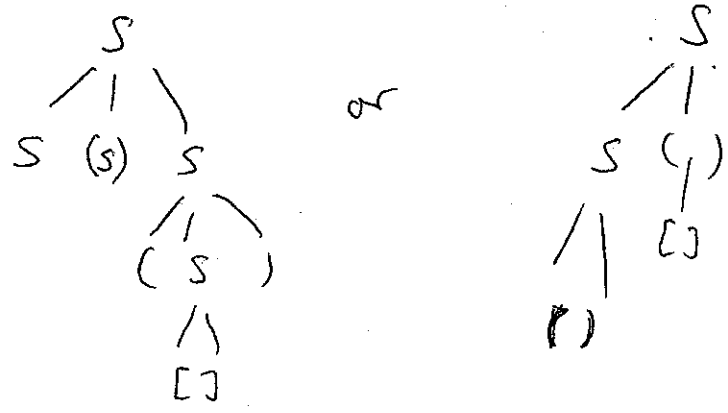
$[ \{ ( [ ] ) \} \{ ( ) \} ] \{ ( ( [ ] [ ] ) ) \}$

$S \rightarrow (S)S, [S]S, \{S\}S, \epsilon$  ← this is sufficient

$S(S)S, S[S]S, S\{S\}S$  ← if we add this we will make it ambiguous

An unambiguous grammar - One that has <sup>exactly</sup> a single parse tree for each word in the language.

$()( [ ] )$

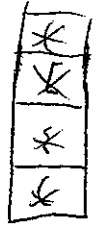


Make  $\exists$  context free languages which are inherently ambiguous.

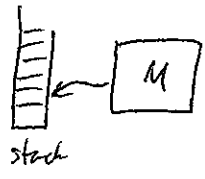
What type of machine recognizes CFL? PDA Push Down Automaton (stack automaton.)

$( \{ [ ] ( ) \} )$

Defn: 6-tuple  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$



Yes!



Let's look at the transitions

$$\delta: Q \times \Sigma_e \times \Gamma_e \rightarrow Q \times (\text{push, pop, nochange}) \times \Gamma^k$$

$\uparrow$  input       $\uparrow$  top of stack symbol

1.  $Q$  finite ~~state~~ set of states
2.  $\Sigma$  is a finite alphabet
3.  $\Gamma$  = finite set of stack alphabet
4.  $\delta: Q \times \Sigma_e \times \Gamma_e \rightarrow \mathcal{P}(Q \times \Gamma^k)$
5.  $q_0$
6.  $F \subset Q$  accept states.

Accept if you end with empty stack  
or

Accept depending on finite-state control

revisit closed under  $\Lambda$  question.

Run both machines no longer works! The cartesian product of two machines is another DFA. But you cannot do this for a PDA because of the stack!

Summary of closure properties

	$\bar{L}$	$\cap$	$\cup$	$\cdot$	$*$	$R$
reg = DFA = NFA	✓	✓	✓	✓	✓	✓
NPDA = CFL	X	X	✓	✓	✓	✓
DPDA	✓	X	X			

$L = \{a^n b^n c^n \mid n > 0\}$  ← not CFL

$\underbrace{a^n b^n c^*}_{CFL} \cap \underbrace{a^* b^n c^n}_{CFL}$

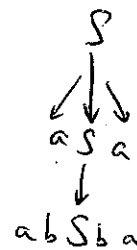
Here not closed under intersection  $\cap$

By DeMorgan's Law, not closed under complement!

Palindromes

$L = \{w w^R \mid w \in \{a, b\}^*\}$  CFL

$S \rightarrow aSa, bSb, a, b, \epsilon$



What about:  $L = \{w w\}$ ? not CFL



← not like a queue for a stack!

What about non-deterministic vs deterministic?

Are NPDA = DPDA?

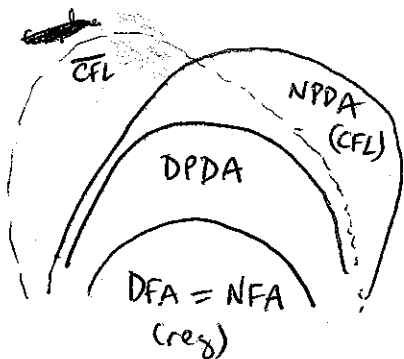
So if  $L$  is CFL but  $\bar{L}$  isn't then

$L$  can't be recognized by a DPDA.

Observation:

Deterministic machines can be thought of as closed under complement.

If they were happy before, now they're unhappy now.



Are DPDA languages closed under  $\cap$  or  $\cup$ ?

$\{a^n b^n\}$  is a DPDA language.