Braylee Cumro

CS397 Hw4

CFGs

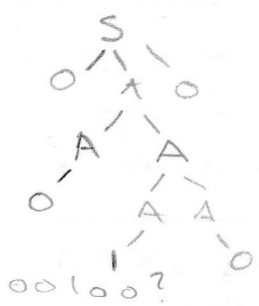
Provide a CFG that recognizes the language

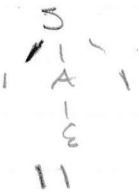
$\{w | w$ starts and ends with the same symbol$\}$

alphabet $\{0,1\}$

$S \rightarrow 0A0 \mid 1A1$

$A \rightarrow 0 \mid 1 \mid \epsilon \mid AA$





Provide a CFG that recognizes the language

$\{w | w = w^R$ (in other words, w is a palindrome)$\}$

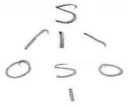
alphabet $\{0,1\}$ palindrome - reads same backwards as forwards

$S \rightarrow \epsilon \mid 0S0 \mid 1S1 \mid 0 \mid 1$

0S0 1S1 → start + end same

If S is a palindrome

0S0 + 1S1 are palindromes



Ambiguity

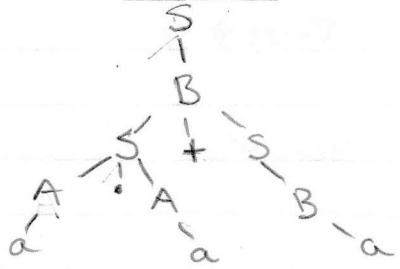
Provide an example of (two different parse trees for a single string) proving that the following CFG is ambiguous

$S \rightarrow A \cdot A \mid B$

$A \rightarrow a \mid S$

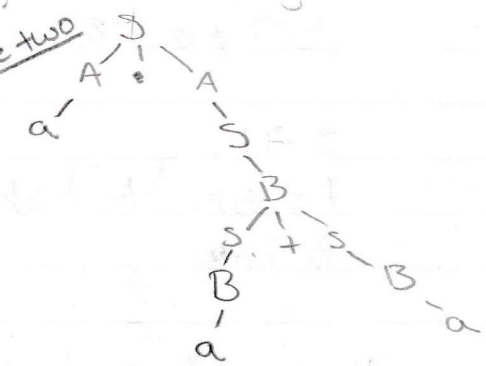
$B \rightarrow S + S \mid a$

tree one



axa+a

tree two



axa+a

Chomsky normal form

Convert the following CFG to chomsky Normal Form

$A \to BAB \mid B \mid \varepsilon$

$B \to \emptyset\emptyset \mid \varepsilon$

Step 1 : New start variable (ensures start variable doesn't occur on right side of a rule)

$S \to A$

$A \to BAB \mid B \mid \varepsilon$

$B \to \emptyset\emptyset \mid \varepsilon$

Step 2: Remove ε-rules

$S \to A$

$A \to BAB \mid B \mid \varepsilon$

$B \to \emptyset\emptyset \mid \varepsilon$

$\Rightarrow$

$S \to A$

$A \to BAB \mid AB \mid BA \mid A \mid B \mid \varepsilon$

$B \to \emptyset\emptyset$

$S \to A$

$A \to BAB \mid AB \mid BA \mid A \mid B \mid \varepsilon$

$B \to \emptyset\emptyset$

$\Rightarrow$

$S \to A \mid \varepsilon$

$A \to BAB \mid AB \mid BA \mid A \mid B$

$B \to \emptyset\emptyset$

Okay to have ε-rule for start variable

Step 3: handle unit rules

$S \to A \mid \varepsilon$
$A \to BAB \mid AB \mid BA \mid A \mid B$
$B \to \emptyset\emptyset$

$S \to A \mid \varepsilon$
$A \to BAB \mid AB \mid BA \mid A \mid \emptyset\emptyset$ → replace B (can drop A b/c the start is A)
$B \to 00$

$S \to BAB \mid AB \mid BA \mid \emptyset\emptyset \mid \varepsilon$ → replace A
$A \to BAB \mid AB \mid BA \mid \emptyset\emptyset$
$B \to \emptyset\emptyset$

Step 4: If $A \to u_1 u_2 \dots u_k$ for $k \geq 3$ w/ u_i a terminal or variable can make rule $A \to u_1 A_1$ $A_1 \to u_2 A_2$ $A_2 \to u_3 A_3$

Any u_i that is terminal replace w/ $U_i \to u_i$

fix BAB $A_1 = AB$
$S \to BA_1 \mid AB \mid BA \mid \emptyset\emptyset \mid \varepsilon$
$A \to BA_1 \mid AB \mid BA \mid \emptyset\emptyset$
$B \to \emptyset\emptyset$
$A_1 \to AB$

handle $\emptyset\emptyset$

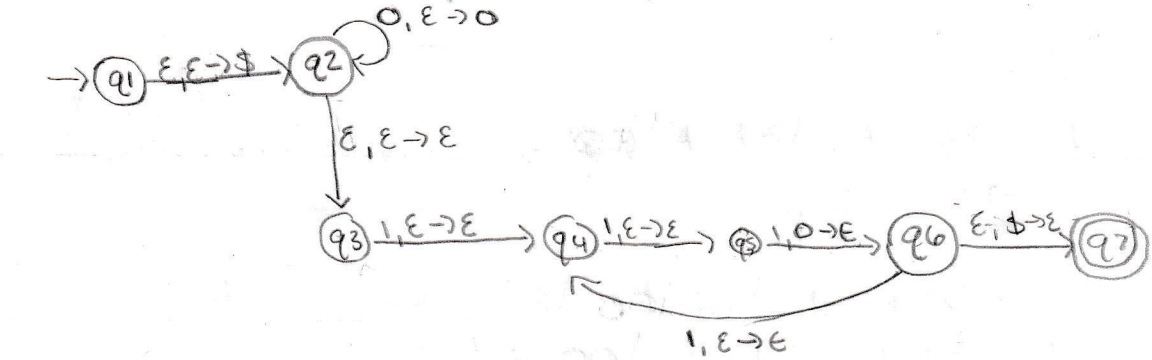
$S \to BA_1 \mid AB \mid BA \mid UU \mid \varepsilon$
$A \to BA_1 \mid AB \mid BA \mid UU$
$B \to UU$
$A_1 \to AB$
$U \to \emptyset$

PDAs

Provide a PDA that recognizes the following language

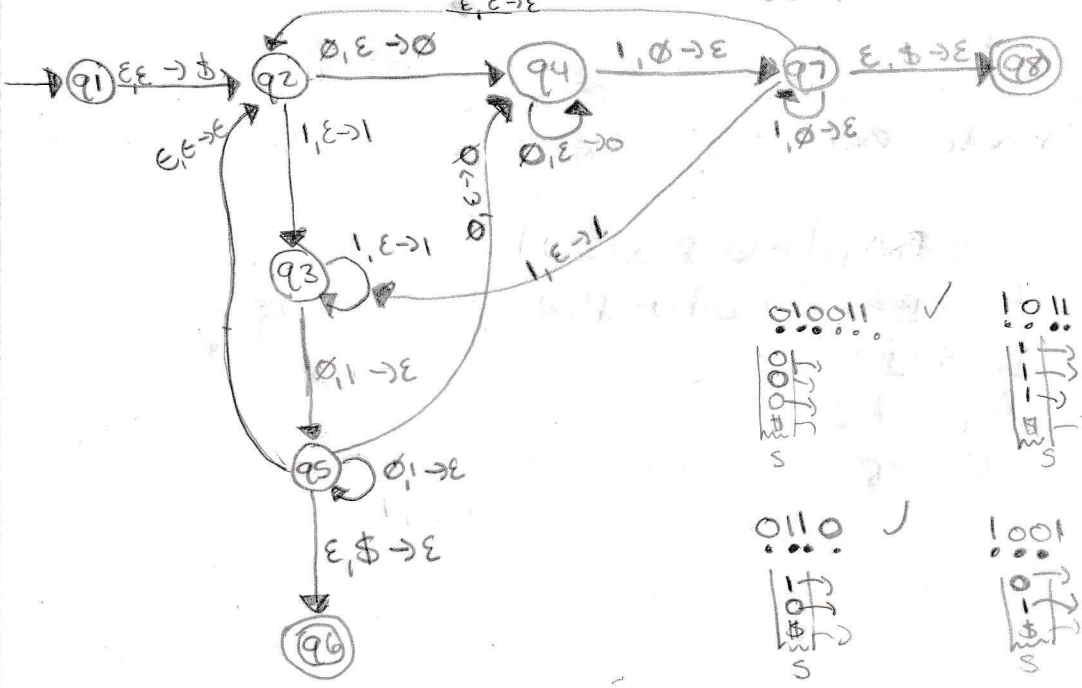
$$\{0^n 1^{3n} \mid n \geq 0\}$$

You can assume the alphabet $\{0, 1\}$



Provide a PDA that recognizes the following language

$$\{w \mid w \text{ has the same number of 0s + 1s}\}$$



Non-context free languages

Show that the following language is not context free using the pumping lemma for context free languages

$$\{0^n 1^n 0^n 1^n \mid n \geq 0\}$$

Assume for contradiction that $A = \{0^n 1^n 0^n 1^n \mid n \geq 0\}$ is context free. Then let p be the pumping length.

Let $s = 0^p 1^p 0^p 1^p$. We see that $s \in A$ and $|s| \geq p$. Then by the pumping lemma, s can be pumped.

$s = 000\ldots0\ 111\ldots1\ 000\ldots0\ 111\ldots1$

Case 1: $s = uvxyz$ ⇝ we can pump v & y & one must be non-empty

both v & y have the same symbol

ex: v & y both have only 0's or v & y both contain 1's

$\underbrace{000\ldots0}_{u}\ \underbrace{1}_{v}\ \underbrace{111\ldots1\ 000\ldots0}_{x}\ \underbrace{0}_{y}\ \underbrace{111\ldots1}_{z}$ ⇝ something like this

If we pump v & y the # of zeros will no longer match the # of 1's that follows. This is a contradiction. The resulting string of pumping in this case would not be part of the language A anymore.

Case 2: $s = uvxyz$

v or y contains more than 1 type of symbol. In other words either v or y contains both 0's and 1's

Ex: v contains both 0's & 1's

$\underbrace{000\ldots0}_{u}\ \underbrace{111\ldots1}_{v}\ \underbrace{000\ldots0}_{x}\ \underbrace{111\ldots1}_{y\ z}$

or

y contains both 0's & 1's

$\underbrace{000\ldots0}_{u}\ \underbrace{1}_{v}\ \underbrace{111\ldots1\ 000\ldots0}_{x\ y}\ \underbrace{111\ldots1}_{z}$

If we pump v & y in this case we will also reach a contradiction.

⇒

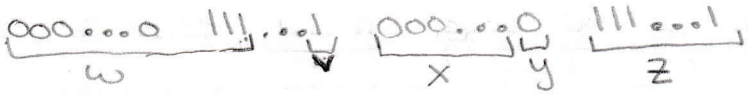
The contradiction we will reach here is that our output string will no longer be in the correct order where we have a group of zeros followed by a group of 1's of the same size followed by a group of 0's with the same size and finally another group of 1's in this size. If we pump in this case we will get something like

000...01 01 11..1 000...0 111...1

where these sections violate the pattern for the string thus string s is no longer a part of language A

Case 3: v and y have only 1 symbol but not the same
in other words v is all 0's & y is all 1's or $s = uvxyz$
v is all 1's & y is all 0's

Examples



v has all 1's y has all zeros or

v has all 0's and y has all 1's

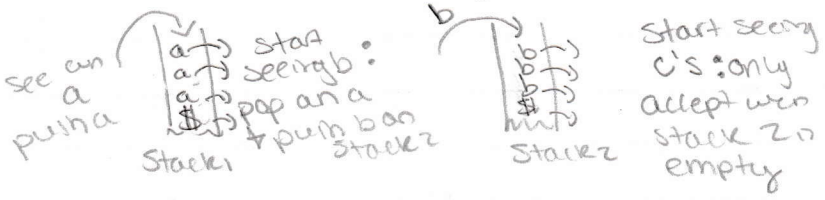
There is a contradiction here as well. If we pump up v and y then there will be two remaining sections (one of zeros and one of 1's) that would have a different length then the other two. We want all four sections to be consisted of the same number of elements but in this case two of them will become shorter then the other two. Since the pattern of the language is broken, s is no longer a part of language A.

Conclusion: We covered all cases of what v and y can be in this language $A = \{0^n 1^n 0^n 1^n \mid n \geq 0\}$. All three cases contradict the pumping lemma for a context free language. Thus A must not be context free. We have no case that we can pump s and s remains in language A.

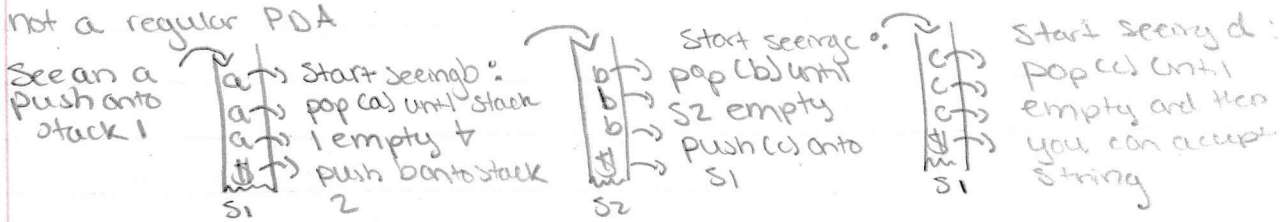
□

Bonus Problem: 2-PDA vs PDA

A 2PDA would recognize the $\{a^n b^n c^n \mid n \geq 0\}$ language at the very least. You could push 'a's on stack 1 then remove those when you see a b and push a 'b' onto stack 2. Then you add c's until stack 2 is empty



You could also recognize $\{a^n b^n c^n d^n \mid n \geq 0\}$ w/a 2-PDA and not a regular PDA



We know that these cannot be recognized by context free grammars / PDA's, but if we add a 2nd stack to a PDA we can recognize more languages. If we can find two examples of languages that can be recognized only if we use a 2nd stack it stands to reason that there may be more examples. 2 stacks changes what can be recognized because you can keep track of more than 1 previous value easily.