Name: Firas Aboushamalah

Course: CS3331

Student#: 250920750

Assignment 2

1. Consider the alphanumeric alphabet Σ = {a, b, . . . , z, A, B, . . . , Z, 0, 1, . . . , 9} and let L be the language of all regular expressions over Σ: L = {w ∈ (Σ ∪ {∅,(,), ∪, ·, ∗ })\* | w is a syntactically legal regular expression over Σ}
2. Giving an unambiguous Context Free Grammar:

S → S U T

S → T

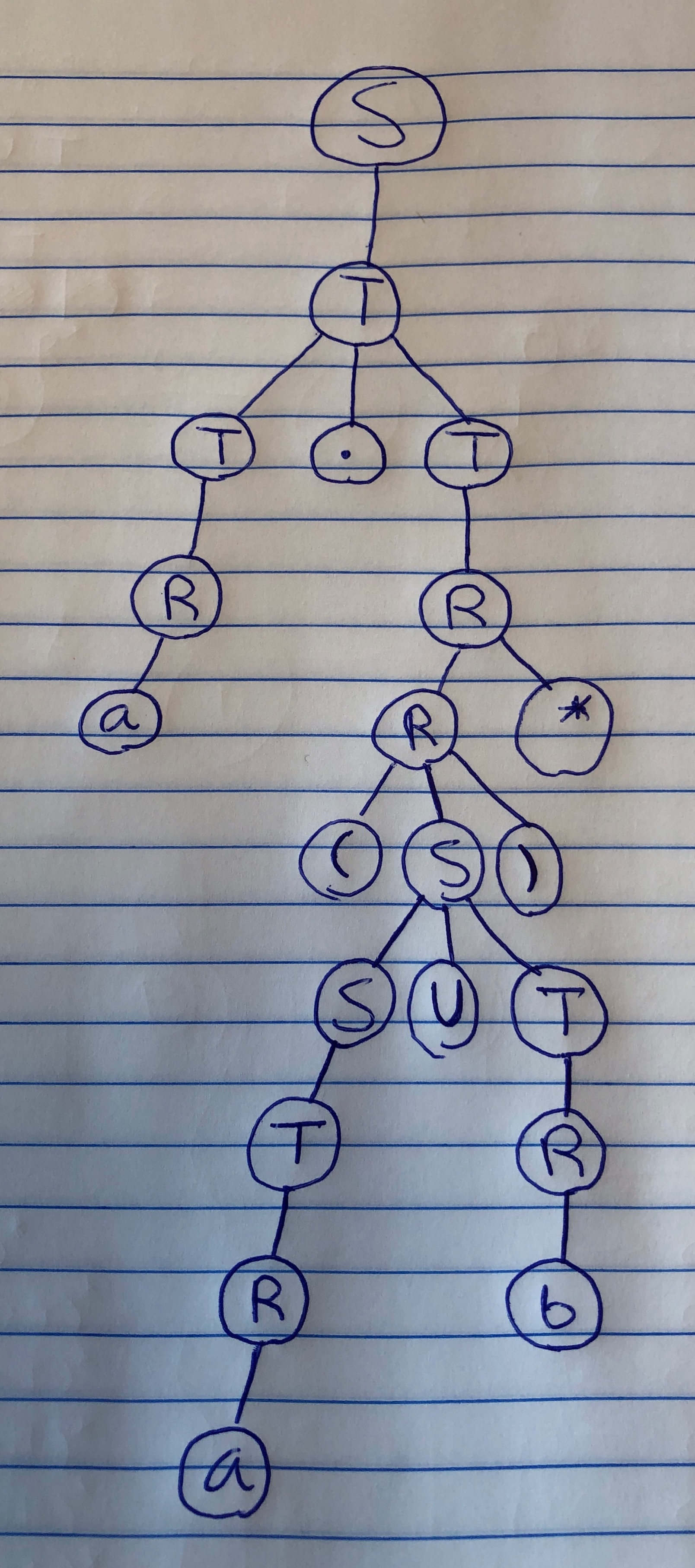
T → T · T

T → R

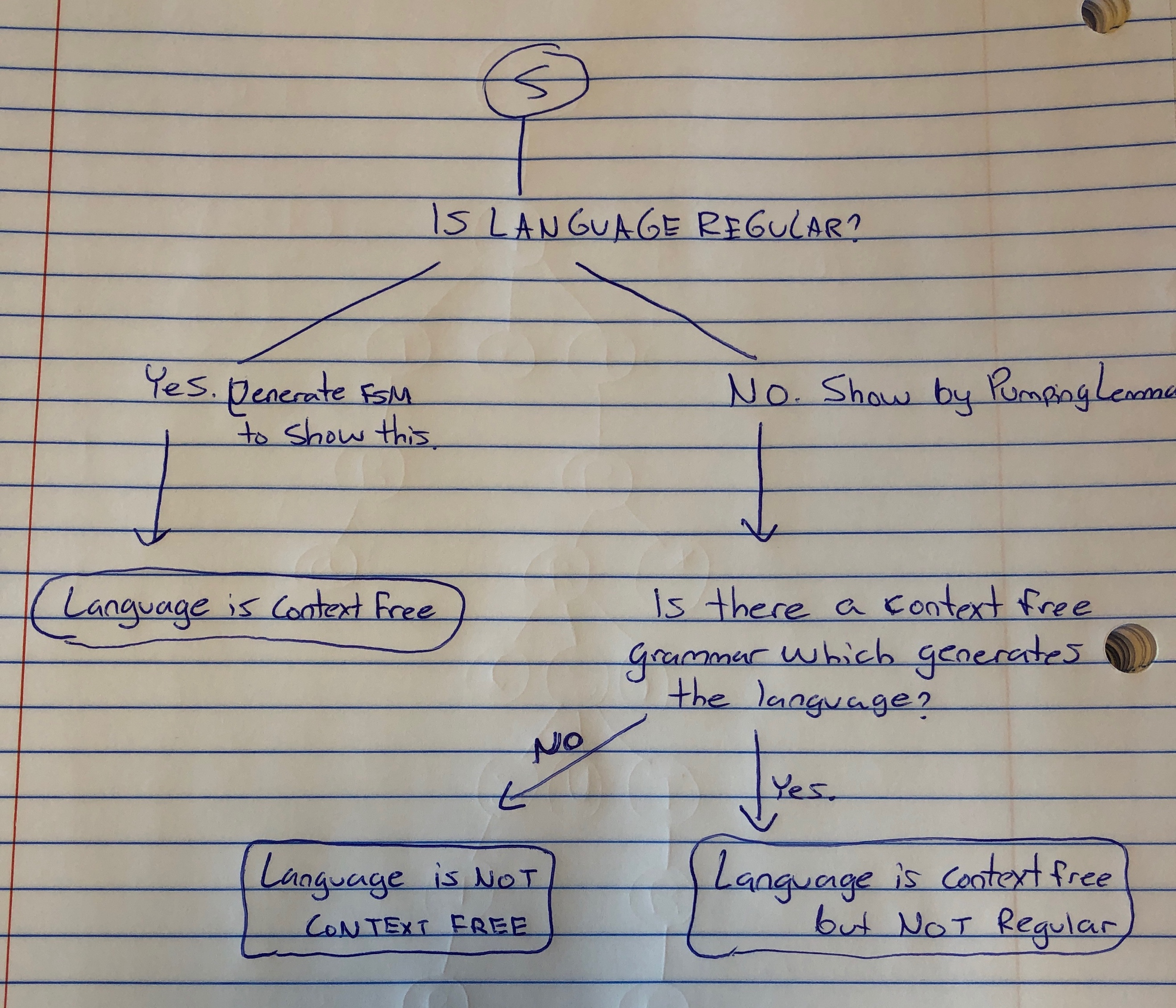
R → R\*  
R → Σ *(this is equal to the alphanumeric set {a, b,…,z, A, B…Z, 0, 1, …}*

R → ∅

R → (S)

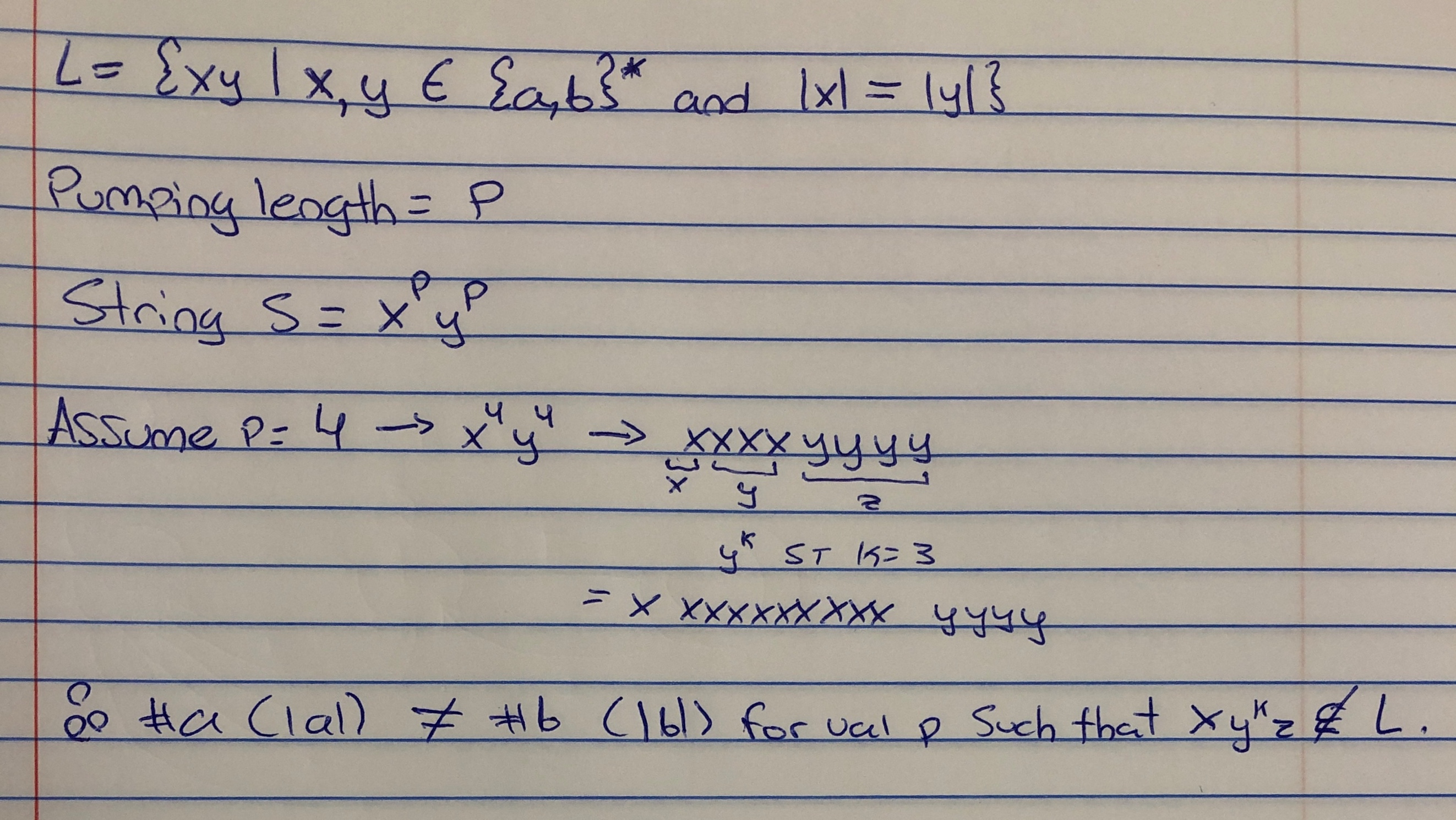
1. Show the parse tree that your grammar produces for the string *a(a ∪ b)\*.*
2. For each of the following languages L, prove whether L is regular, context-free but not regular, or not context-free

We know that every regular language is inherently context-free, but not every context-free language is regular. Therefore, if we prove that a language is regular then we have also proved that a language is context free. However, if a language is not regular, then it **may** be context free if it has a regular grammar that can generate the language. I have created a simple decision algorithm below:



(a) L = {xy | x, y ∈ {a, b} ∗ and |x| = |y|}

First, proving that L is not regular:



Now, following the decision tree I made – is there a context free grammar which generates L? Yes there is:

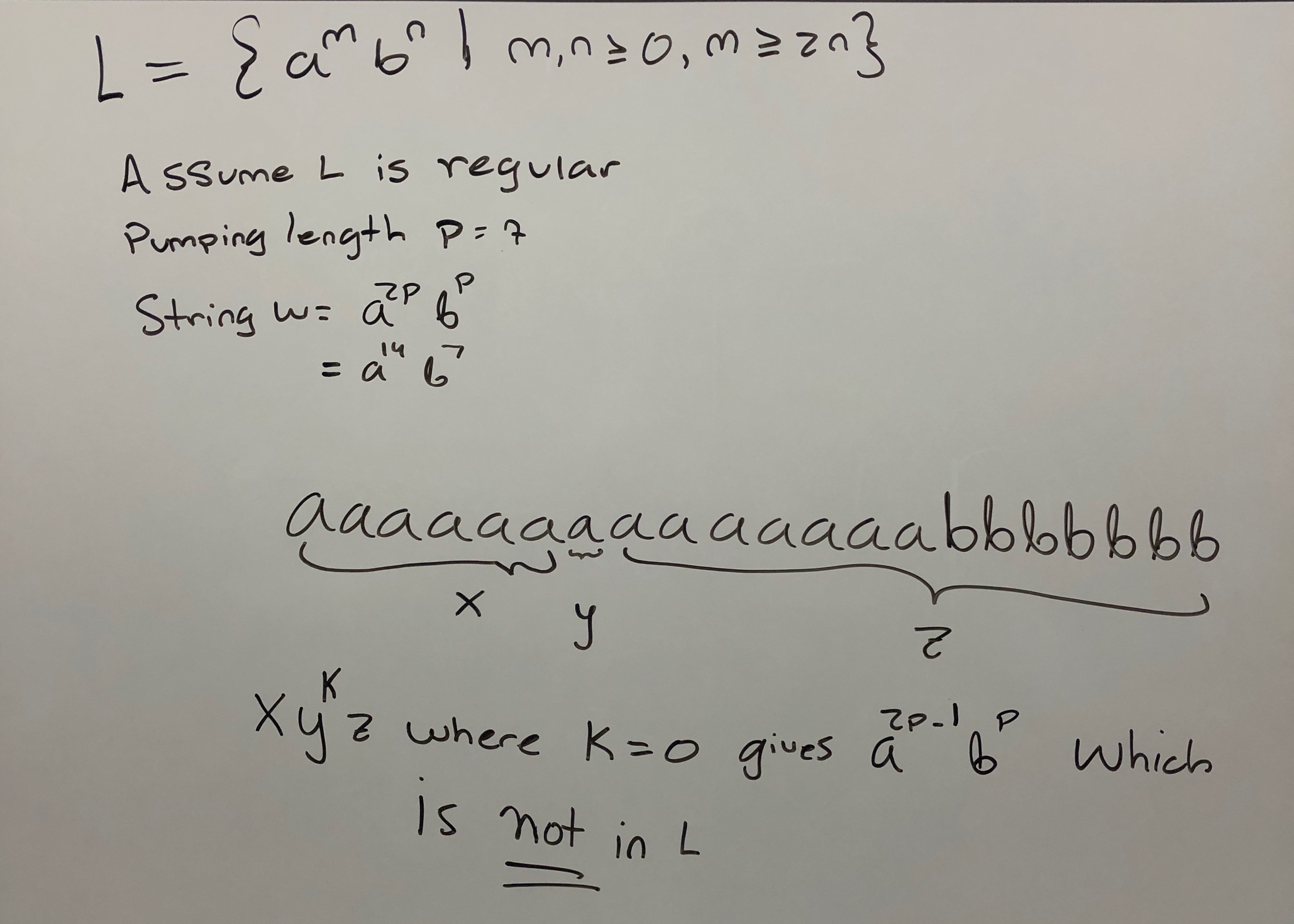
S → aRb

R → aRb | ε

Therefore, L is context-free but not regular.

* 1. {ambn | m, n ≥ 0 and m ≥ 2n}

The language is not regular. The proof is below:



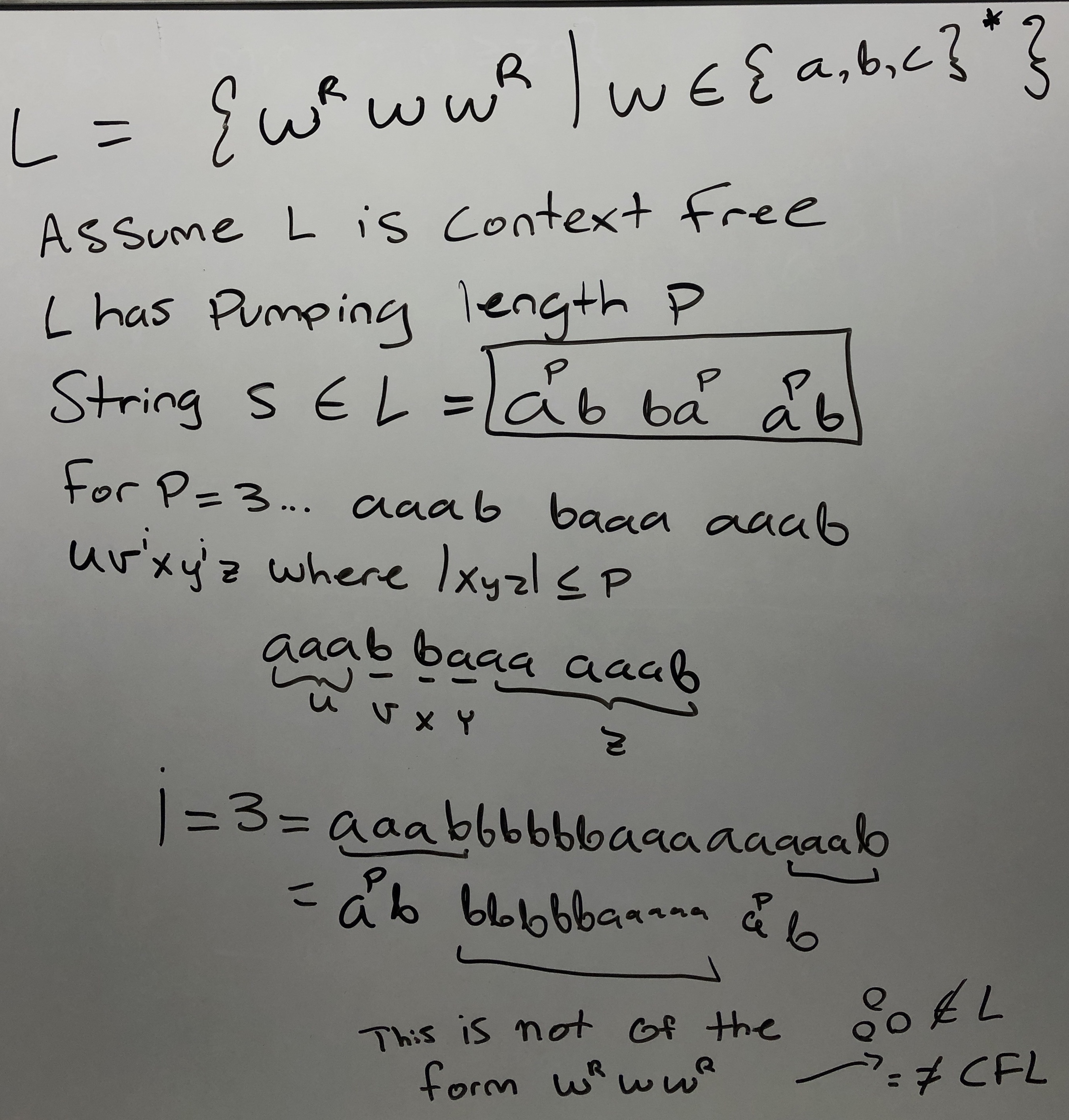
However, it **is** context free. The reason is that there exists a regular grammar as follows:

S → aaRb | R

R → aR | aaSb | a | ε

* 1. {wRwwR | w ∈ {a, b, c} ∗}

Using the pumping lemma for a context free language, we can prove this is not context free.



1. Consider the language L = {wwR|w ∈ {a, b} ∗}. Below are two proofs, one showing L is context free, the other showing the opposite. Which proof is correct and why?

The more correct proof is the first one producing a context free grammar which generates L. The reason being, if we can prove that L generates one string then we can prove it generates a maximum of k strings using the same idea as the pumping lemma. A simple string which L(g) where g = CFG, is aabbaa where the following path can be taken to produce the string:

S → aA

→ aSa

→ aaAa

→ aaSaa

→ aabBaa

→ aabSbaa

→ aabbaa

However, instead I would like to prove why the example using the pumping lemma to prove it is *not* context free is **incorrect:**

When producing string s = apbbap , we may choose an arbitrary pumping length – let’s use 4.

This produces the string aaaabbaaaa. Clearly, if we break down s into uvxyz, we notice that there are three possible cases which vxy takes on variables a or b. (because |vxy| ≤ p) These are:

Case 1) vxy = ajbk for j, k ≤ p

Case 2) vxy = bjak for j, j ≤ p

Case 3: vxy = ajbaj for j ≤ p

In the example done in the instructions, the pumping lemma was used incorrectly by giving both a’s a different number (where at least one of p/q is not 0). However, the law of the pumping lemma shows that uvxyz must have an i for which uvixyiz. This means both v and y get the same i. As a result, we notice that for any value i ≠ 1, the pumping lemma does not provide a contradiction because the resulting string is always ajbbaj ∈ L.

1. (30pt) Show that the following problem is decidable: Given a context-free grammar G, does G generate any odd-length, nonempty strings?

This can be solved by looking at the intersection and automata constructions. Firstly, recall that for a given *CFL* *L* and a *regular language* *K*, 𝐿 ∩ 𝐾 is also a context free language. Therefore, if we look at the construction of *L*:

{w | w is odd length}

We can create a DFSM for *L* to prove its regularity and further its validity by intersecting it with the CFG to produce the language :

{w | w is odd length, and w ∈ L(G)}.

Then, we can note that the language is nonempty iff the CFG G generates a word of odd-length, which it does.