Problem 1: Regular and context free languages (20 points)

Fixed point numbers are sequences consisting of digits and a single period to represent a few "real" numbers. The period should **not** be the first or last character of the string. Here are some examples:

- 434.001
- 0.1
- 00.1
- 0.000

(Usually 1. and .1 would be allowed, but we are making the problem easier.)

Use the `grep` *expression* `[0-9]` *in place of* (0+1+2+3+4+5+6+7+8+9) *in your answers. It's a **lot** easier to write.*

Part A: Write a regular expression specifying fixed point numbers.

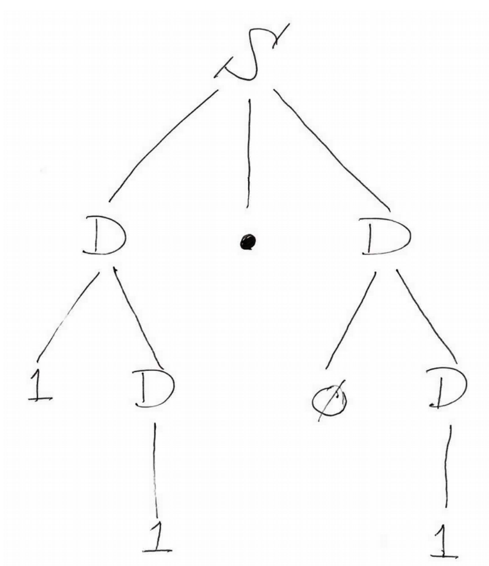
[0-9][0-9]*.[0-9][0-9]*

Part B: Write a context free grammar specifying fixed point numbers.

- **S → D.D**
- **D → [0-9] | [0-9]D** ***or*** **D → [0-9] | DD**

Part C: Using your grammar, draw a parse tree for the fixed point number 11.05 .

except this one is actually 11.01



Problem 2: CFG → CNF (20 points)

Convert the following Context Free Grammar to Chomsky Normal Form.

- The alphabet for the language is **Σ** = {**a**, **b**}.
- The start variable for the language is (as usual) **S**.

Here are the rules:

- **S → aTa | b**
- **T → bTb | ϵ**

Avoid creativity and stick with the procedure described in Example 2.10 of the textbook as it was followed in last week's class meetings.

Add $\mathbf{S_0}$ **→ S**

- S_0 **→ S**
- **S → aTa | b**
- **T → bTb | ε**

Eliminate ϵ rules, in particular T → **ε**, and add new rules with T replaced by **ε**

- S_0 **→ S**
- **S → aTa | aa | b** ;; often the aa was omitted
- **T → bTb | bb**

Remove unit rule $\mathbf{S_0}$ **→ S** and replace S with with its targets.

- S_0 **→ aTa | aa | b**
- **S → aTa | aa | b**
- **T → bTb | bb**

Get rid of unreachable S rule (not part of book's algorithm)

- S_0 **→ aTa | aa | b**
- **T → bTb | bb**

Introduce variables for alphabet symbols **a** and **b**

- S_0 **→ ATA | AA | b**
- **T → BTB | BB**
- **A → a**
- **B → b**

Introduce variables for **AT** and **BT**

- S_0 **→ MA | AA | b**
- **T → NB | BB**
- **M → AT**
- **N → BT**
- **A → a**
- **B → b**

Many solutions were not in CNF! The right side can only be:

- **One letter from the alphabet, such as A → a**
- **Two variables, such as T → NB**
- **ε, <u>but only if</u> the left hand side is the starting variable, as in S_0 → ε**

Language: $\{a^m b^{2n} a^m \mid m \geq 1 \text{ \& } n \geq 0\} \cup \{b\}$

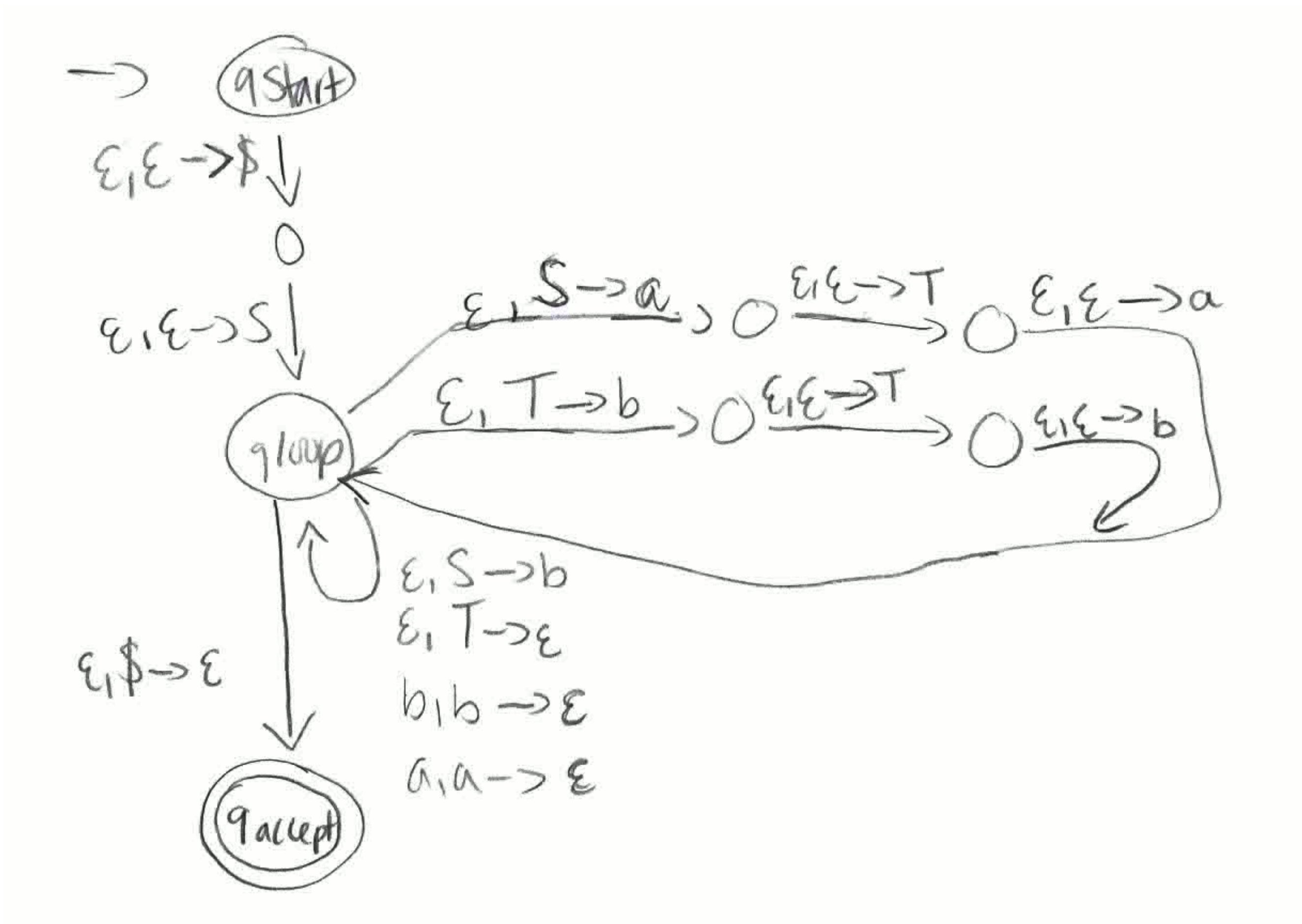
Problem 3: CFG → PDA (20 points)

Generate a PDA that accepts the Context Free Grammar of the previous problem:

- **S → aTa | b**
- **T → bTb | ϵ**

I suggest that you follow the **flower algorithm** presented in Theorem 2.20 of the textbook (which you ***must*** have used for your solution to Problem 2.12).

Your answer should be a **real** PDA. Don't label transition arcs with short cuts such as **ϵ, T → bTb** if you expect full credit.



Anitra Griffin drew this

Pumping Lemma (Theorem 1.70)

If L is a context free language, then there is a number *p* (the pumping length) where, if *s* is any string in L of length at least *p*, then *s* may be divided into five pieces *s* = *uvxyz*, satisfying the following conditions:

- for each $i \geq 0$, $uv^ixy^iz \in A$
- $|v| > 0$ or $|y| > 0$
- $|vxy| \leq p$

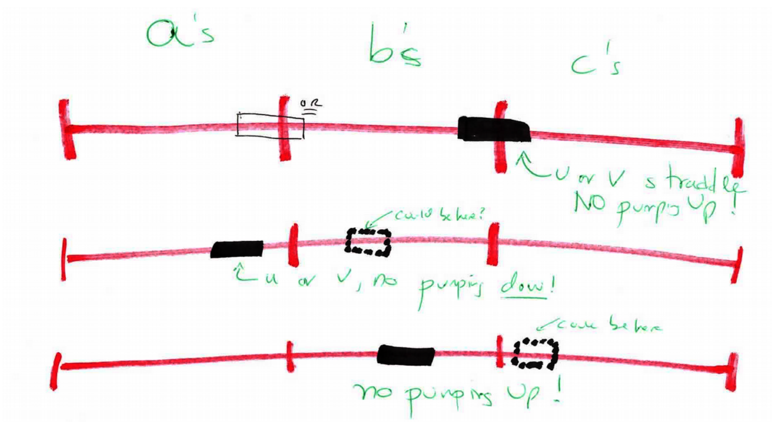
Problem 4: Proving Context (20 points)

Show that the language $\{a^ib^jc^k \mid i \geq j \text{ and } i \geq k\}$ is **not** context free.

Note that aabbcc and aac are **in**, but abbc and cab are **out**.

Assume *p* is the pumping length.
Let s be $a^pb^pc^p$ which is divided into the *uvxyz* of the pumping lemma.
Important warning: If your s does not use *p*, **you are doomed**!
Case 1: either *v* or *y* contains two different alphabet element
Suppose that one of *v* or *y* contain two different alphabet elements. WLOG assume *v* contains both a and b, i.e., $v = a^mb^n$ with $m>0$ and $n>0$. In the case, pumping v once would yield some form of $a^gb^ha^ib^kc^p$ which is **not** in the language.
Case 2: v and y contain only one (or none) type of alphabet element.
Case 2A: either *v* or *y* contains one or more a's
WLOG, assume *v* contains an a. In that case, pumping *v* down to zero, would reduce the number of a's but cannot reduce the number of c's, because that is too far away.) In that case, the number of a's would be reduced, but the number of either b's or c's would not. Therefore. the down-pumped string is **not** in the language.
Case 2B: There are no a's in either v or y
Then either v or y contains either b's or c's. WLOG, assume that y is a non-zero section of b's. In that case, pumping up will result in there being more b's than a's. Therefore. the up-pumped string is **not** in the language.

$a^pb^pc^p$ cannot be pumped: The language is not context free.



Problem 5: Proving context (20 points)

Show that the language $\{a^i b^j c^k \mid i \geq j + k\}$ is context free. I strongly suggest you create a context free grammar to solve this problem.

Note that aaabbc and aaaab are **in**, but aaabbcc and abba are **not**.

You must add comments to your answer! I'll need them.

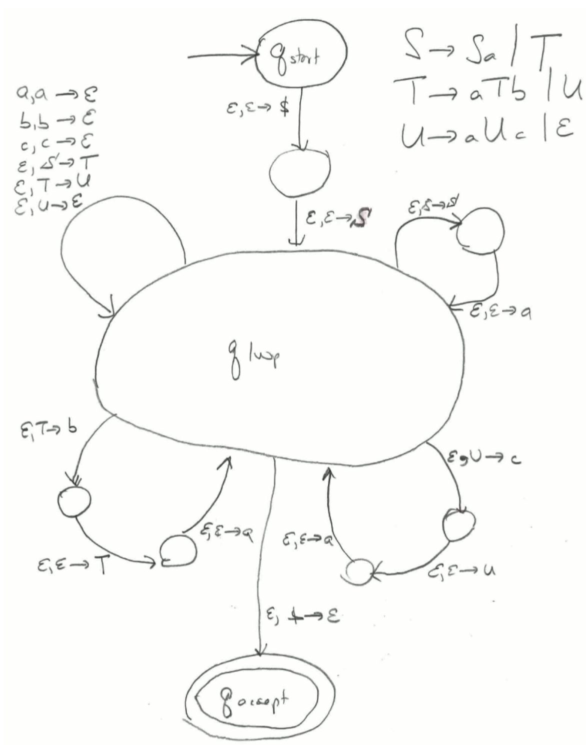
Here is a context free grammar

- **S → aS | T**
- **T → aTc | U**
- **U → aUb | ε**

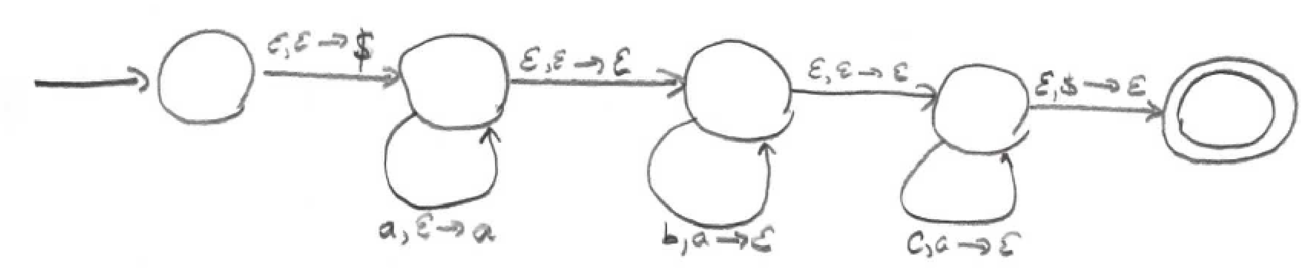
It will be easier to rewrite $a^i b^j c^k$ where $i \geq j + k$ as $a^{u+j+k} b^j c^k$ *where* $j \geq 0$, $k \geq 0$ and $u \geq 0$, that is, set u to i-(j+k). Now do the following steps:

- **Use the S → aS rule *u* times to build up to a^uS**
- **Then use S → T to get $a^u T$**
- **Now use T → aTc *k* times to build up to $a^{u+j}Tc^k$**
- **Then use T → U to get to $a^{u+j}Ub$**
- **Now use U → aUc *k* times to build up to $a^{u+j+k}Ub^j c^k$**
- ***Finally* use U → ε to get the desired $a^{u+j+k} b^j c^k$**

You could also use a PDA to demonstrate that the language is context free. Here is the complicated PDA you'd get following the book's CFG to PDA algorithm.



However, it would be possible to use a PDA similar to the ones in Figures 2.15 and 2.17 in the textbook.



It can be a little hard to argue how a PDA matches a language when there are all those ε transition.

It is possible to implement this language with a determinate PDA which would look like the following, though you really ought to add a trap state.