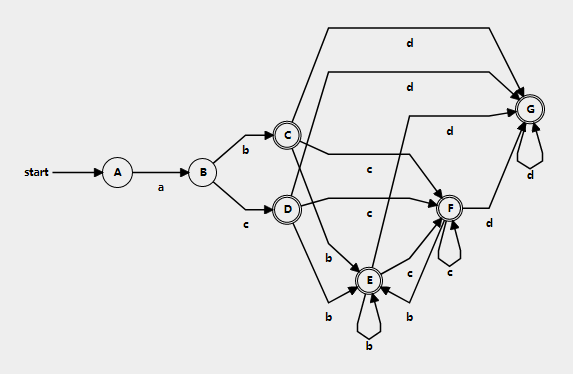
**COMP1003: Maths Worksheet**

For questions 1 to 3 consider the regular expression a(b|c)+d\*

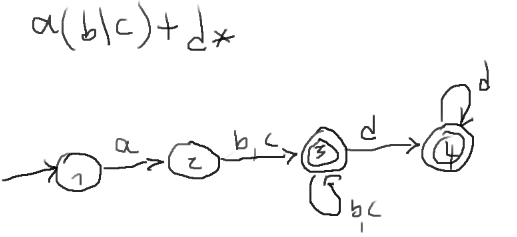
1. List the 5 shortest strings of the language characterised by the expression.

{ ab, ac, abb, abc, acb, acc, abd, acd, … } Apparently there are more than just 5 “shortest” strings ..

1. Draw a deterministic finite state automaton that accepts the language generated by the expression.



Or, alternatively



This one is non-deterministic, because not all possible transitions are defined.

1. Formulate a grammar that is equivalent to the expression, which means it generates the same strings.

S -> aQ ;

Q -> bT | cT ;

T -> bT | cT | D ;

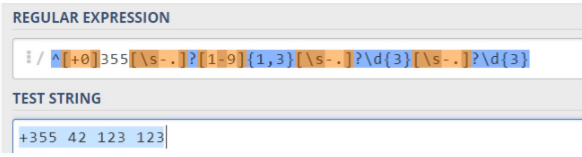
D -> dD | eps; # eps is the empty word

Alternatively, making use of the + and \* symbols: S -> a(b|c)+d\* , which obviously is the same as the original expression.

1. Construct a regular expression for international phone numbers.

Many examples on the internet …

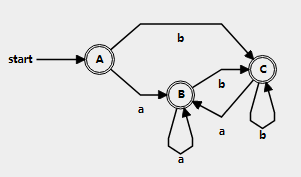
For Albania: \s = white space \d digit



By the way you can use regular expressions in Notepad++

<https://www.kreatx.com/regular-expression-regex/>

1. Find a regular expression that is equivalent to this DFA.



(a|b)\* the set of all strings of a and b including the empty string.

(there are obvious simpler automata for this set … )

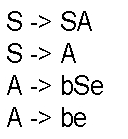
You can play with automata and regular expressions here:

<https://cyberzhg.github.io/toolbox/nfa2dfa>

A picture containing graphical user interface

Description automatically generated

1. Consider the grammar



Describe informally (in words) the language generated by the grammar.

It easier to consider the equivalent language

S->SA

S->A

A->(S)

A->()

First two rules create an arbitrary positive number of As

Rules 4 creates a matching pair of () for an A

Rule 3 embeds whatever the grammar can generate in matching () for an A

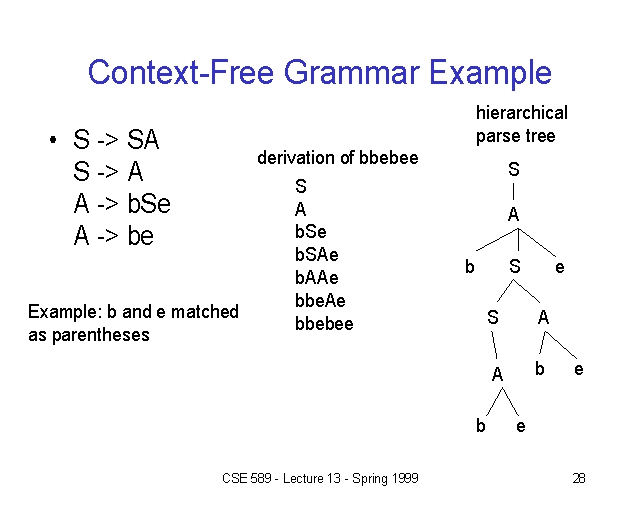
These are all legal arrangements of nested matching parentheses

Write down a derivation of the word bbebbe. (()()) ))((

* Below

For the word beebee draw a syntax tree.

* Below



1. Find context free grammars for the following formal languages



S -> TC # this splits the problem into two halves

T -> aTb | eps # this generates anbn

C -> Cc | ccc # this generates at least 3 c’s



S -> AT | TB

T -> aaTb | eps

A -> Aa | a

B -> Bb | b

The second rule generates {anbm : n=2m} these are exactly the strings we don’t want. We therefore have to either add at least one a to the left or at least one b to the right. It does not matter how many more we add, the resulting string will not satisfy n=2m. The first rule distinguishes these two possibilities and rules 3 and 4 add the a’s and b’s.

Chart

Description automatically generated with medium confidence

S -> aSb | T

T -> Tb | a | aa | aaa | eps

The first rule creates anbn for some >=0. For these strings the constraint n<=m+3=n+3 clearly holds.

The second inserts an arbitrary number of b’s (which cannot invalidate the inequality) or inserts up up to 3 more a’s.

Note that in all these three examples on top of stating the rules and explain what they do, one would also have to show that the rules indeed generate \*all\* of the requested strings.

Graphical user interface, application

Description automatically generated

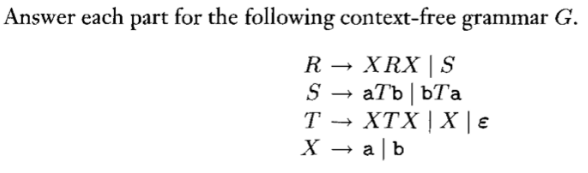
Note: a rule “a; b; X” means: If an a was read, write back a b and do X, where X can be move to the right (R), move to the left (L), or don’t move. The triangle on the left indicates the start state and the little square is the blank tape symbol.

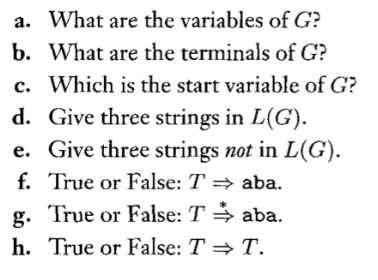
Solution: The TM computes the identity function over low-caps strings of a’s and b’s

A word, eg, “abbaa” is put on the tape and the machine starts. If it reads an a it replaces it by an a and goes one position to the right; likewise for a b. The other two rules are never used as the question allows for inputs from {a|b}\* only but not for capital letters ..) . When the word has been read (and copied back) a blank is reached, and the TM goes immediately into the accepting state q1.

If we would allow for capital A’s and B’s, too, the machine would compute the low-caps version of a string from {a|A|b|B}\*, ie, aaaBbBA -> aaabbba.

The empty word is also accepted because the machine can go right away into q1 if there is nothing on the tape.

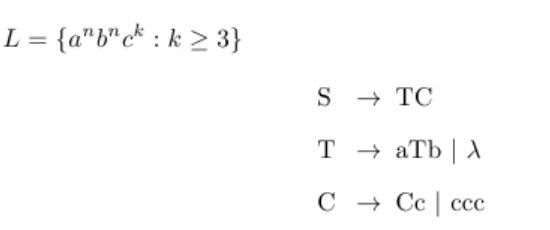
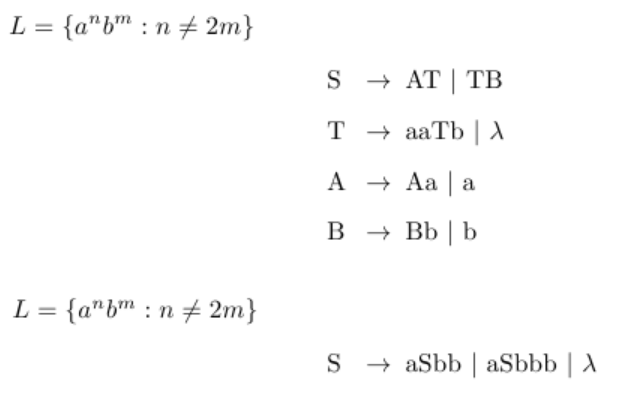
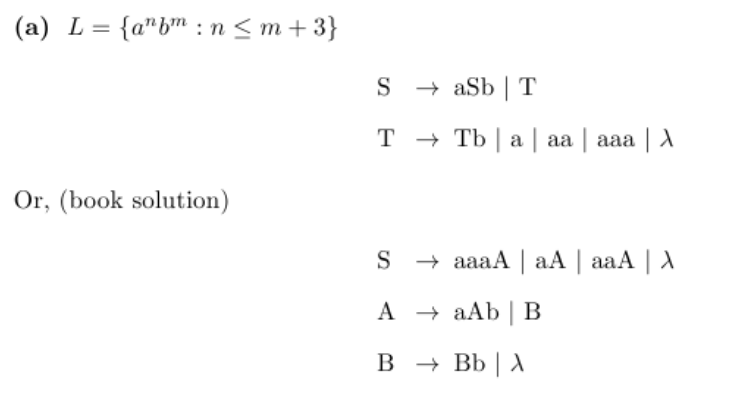


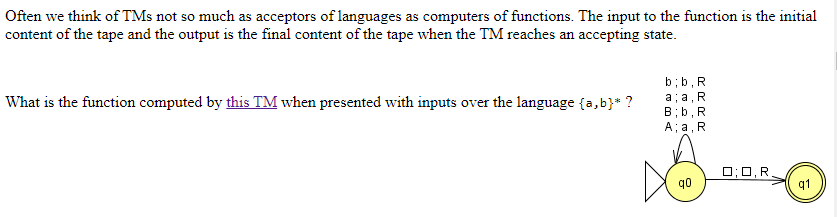


https://stackoverflow.com/questions/7814904/what-are-these-arrow-operators-in-context-free-grammar/7814951

https://math.stackexchange.com/questions/58773/question-regarding-context-free-grammar-exercises

1. Find context free grammars for the following formal languages





Note: a rule “a; b; X” means: If an a was read, write back a b and do X, where X can be move to the right (R), move to the left (L), or don’t move. The triangle on the left indicates the start state and the little square is the blank tape symbol.

https://www.cs.odu.edu/~zeil/cs390/latest/Public/turing-jflap/index.html

