William Wallace
1216661

COSC341: Assignment 2

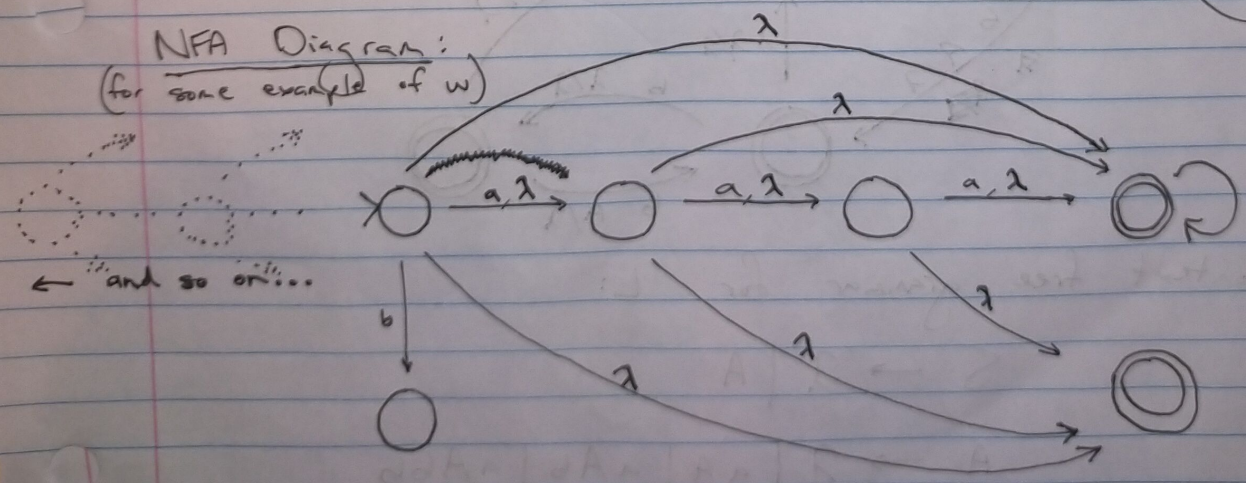
(1)

Q1) The NFA, N, for the language:
$L(N) = \{w \mid w$ is a substring of some string in $L(M)\}$:

N will contain λ transitions from any state that will eventuate into an accepting state, directly to the final accepting state. The state that will eventuate into an accepting state will also contain λ transitions to the next state on the path to the final accepting state. Lastly, each accepting state will also contain a λ transition to an accepting state that is takes no further transitions. This last state allows the accepting state on the path to stop taking any more input and remain in the accepting state.

NB: when I refer to "eventuating" to the "final" a accepting state, I mean that the final state is the entire string, and the states before it are the substring.

NFA Diagram:
(for some example of w)



b) This implies the union closure property of regular languages, as words can be made from substrings, which can be made from the union of characters.

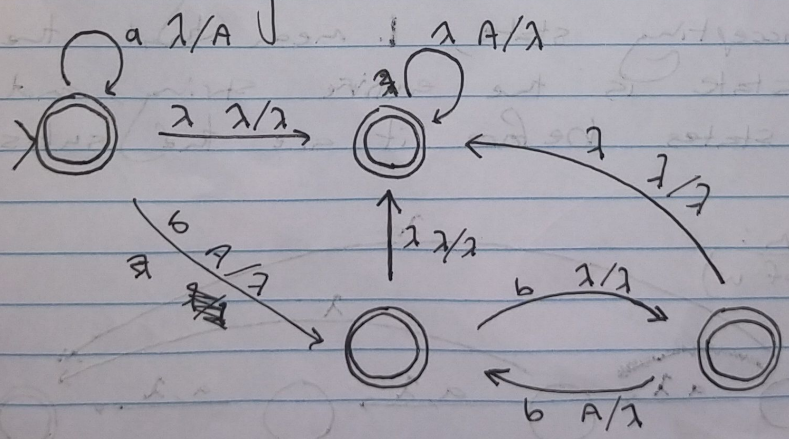
(2)

Q2) Show $L = \{a^n b^m \mid 0 \leq n, 0 \leq m \leq 2n\}$ is not regular:

Suppose that L was ~~context free~~ regular and let k be a positive integer for which the pumping lemma applied to it. Consider the word $w = a^k b^{2k}$ and the factorisation $w = xuy$ of the form guaranteed by the pumping lemma. To satisfy the pumping lemma, only x and u may be a's, since $a^k b^{2k}$ wouldn't apply in any other case. ~~Next, delete u~~

Next, delete u from our factorisation. Consequently $a^k b^{2k}$ now becomes $a^{k-1} b^{2k}$. Since $k-1 \times 2 \neq 2k$, the original supposition that L was context free must be false, thus L is ~~context free~~ not regular, since $x^{k-1} u y^{2k}$ is not of the correct form.

* b) PDA state diagram:



c) Context free grammar for L:

$S \rightarrow \lambda \mid A$

$A \rightarrow \lambda \mid aA \mid aAb \mid aAbb$

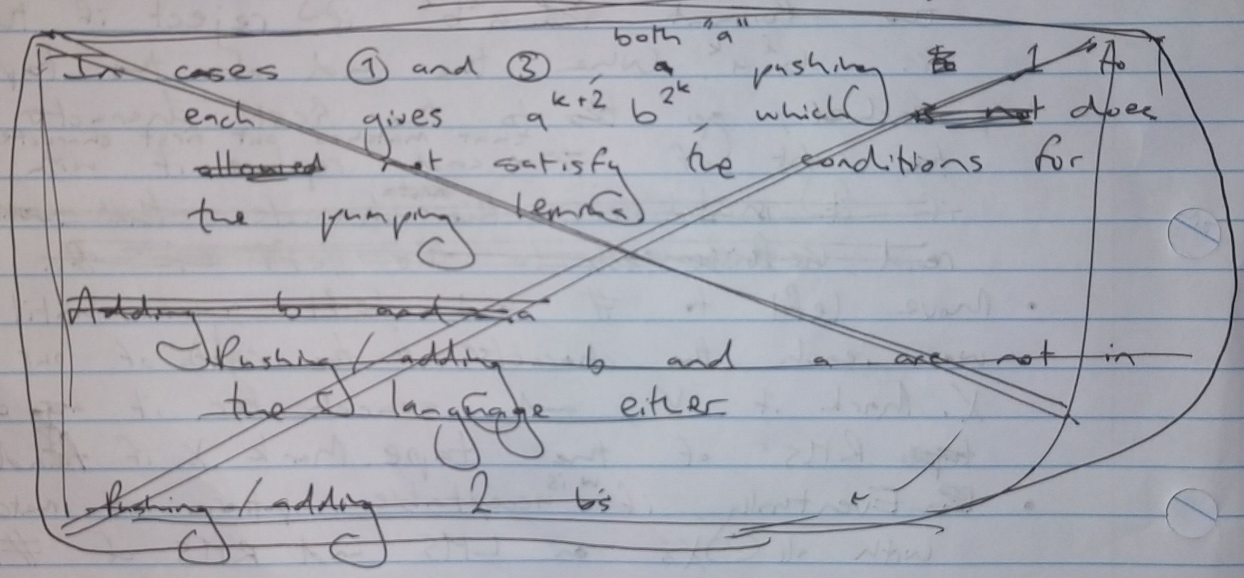
(3)

Q3) Show $L = \{a^n b^m \mid 0 \leq n, m = 2^n\}$ isn't context free:

Suppose L was context free and let k be a positive integer for which the pumping lemma applied to it. Consider the word $z = a^k b^{2^k}$ and the factorisation $z = uvwxy$ of the form guaranteed by the pumping lemma.

There are three cases to consider:

1. u is a(s) and x is a(s) (could be many as)
2. both u is b and x is b
3. u is a and x is b (or vice versa)



- Combined u and x will give i as and j bs.
- For pushing n number of times, we get two expressions: $n = k + ri$ and $m = 2^{k+rj}$, where r is number of times pumped. Since n and m do not increase at the same rate, $m = n$ will not hold, this doesn't satisfy the conditions required for the pumping lemma (language) and therefore the pumping lemma declares that the language is not context free.

(4)

Q4) The Turing machine for accepting w#v for $w \in \{a,b\}^*$ and v is concatenation of copies of w:

- This Turing machine would ~~read~~ start at (move right and) an empty/blank initial state. Then it would replace the first character with an X.
- It would move right as far as # and read the next character after #, on its right. Replace this character with X if it matches the ~~left~~ character found in the first "read". If the initial first character ~~#~~ does not match the ~~first~~ first character ~~of~~ after #, the machine breaks. If blank character, ~~accept~~ halt-accept, (as it is finished)
- The machine would move left, towards the #. Then keep going left until X character is found.
- Replace X with original character. This would be done by following the "write" command specified by the ~~state~~.
- Move left and read the next character, and replace with X.
- Repeat steps above to check if it comes after the ~~first~~ X already after the #.
- When the entire tape has been read in this ~~sequence~~, without looping errors or ~~not~~ breaking, replace the first character in w with X.
- Read the first character from the right of the blank initial character. If X, halt-accept.

4

State	Read	Write	Move	New State.
START	B	B	R	READ
READ	a	X	R	SEARCH-A
READ	b	X	R	SEARCH-A
READ	#	#	L	TO-START
SEARCH-A	a	a	R	SEARCH-A
SEARCH-A	b	b	R	SEARCH-A
SEARCH-A	#	#	R	FIND-A
FIND-A	a	X	L	RETURN TO-A
FIND-A	X	X	R	FIND-A
FIND A	B	B	L	CHECK
RETURN-TO-A	X	X	L	RETURN-TO-A
RETURN-TO-A	#	#	L	REPLACE-A
REPLACE-A	a	a	L	REPLACE-A
REPLACE-A	b	b	L	REPLACE-A
REPLACE-A	X	a	R	READ
TO-START	a	a	L	TO-START
TO-START	b	b	L	TO-START
TO START	B	B	R	READ
CHECK	a	a	L	CHECK
CHECK	b	b	L	CHECK
CHECK	#	#	L	CHECK
CHECK	X	X	L	CHECK
CHECK	B	B	R	IS-VALID
IS-VALID	X	X	R	HALT-ACCEPT

Repeat for $b

NB: there is a loop for inputs of a single # which I didn't account for, but it would accept in theory.

5

Q5) The Turing ~~Machine~~ Machine for accepting $w\#a^k b^l$ for $w \in \{a,b\}^*$ and w contains k a's and l b's:

- This Turing ~~Machine~~ machine would start at an empty/blank initial state and move right and replace the first character with a X, ~~if it is an a. If it is a b, reject.~~
- Move right across the tape to # and right once more to the first character to the right of #. ~~If it is a "b" reject, because~~ Check that all the ~~this~~ characters to the right of # are in the format ~~a^k b^l~~ $a^k b^l$ ie. reject if the format is $b^l a^k$. When the end of the tape is reached, go to the first character to the right of # that matches our first character and replace it with X. ~~if it matches the first character that was read, to the left of #.~~
- Move left to # and left again until we reach the character to the right of our first X. Mark it X, and search for it on the ~~tape~~ RHS of the tape. Mark X if found
- Eventually, if w is acceptable, pattern will match with all Xs on LHS and RHS of #.
- When we put the last X on the RHS, return to # and check if any non-X characters are to the left of #.
- If there are not, check for any remaining characters on the right of the RHS. If there are reject. If there are not, halt accept.

$k=1, l=2 : a^1 b^2$

$w \in \{abb, bab, bba\}$

Checking $w = abb$:

BXbb#abb
BXbb#Xbb
BXXb#Xbb
BXXb#XXb
BXXX#XXb
BXXX#XXX ^

accepted