COSC1107/1105 - Computing Theory
Assignment 1 (10%)

Total marks: 75

Deadline: Sunday March 26th 2017, 11:59 PM

Please read all the following information before attempting your assignment. This is an *individual* assignment. You may not collude with any other individual, or plagiarise their work. Students are expected to present the results of their own thinking and writing. Never copy another student's work (even if they "explain it to you first") and never give your written work to others. Keep any conversation high-level and never show your solution to others. Never copy from the web or any other resource. Remember you are meant to generate the solution to the questions by yourself. Suspected collusion or plagiarism will be dealt with according to RMIT policy.

Submission Instructions: You will need to submit all your files inside a .zip file via this Google Form:

`http://tinyurl.com/ct17-ass1-sub`

- Answers to questions must be submitted in individual text or JFLAP files. The exact file names for all files must be used (including exact capitalization and file extension), as indicated in each question. All such files must be put inside a *single zip file* called `<your student number>.zip`. For example, `3234212.zip` In total, this zip file should contain 28 `.txt` and 3 `.jff` files.
- Place all files in the *root* of the zip file; *do not* place the files in a folder inside your submission file.
- Files with extension `.txt` must be text ASCII/UTF-8 files (not Word or any other format). If in doubt, please check the type of your files, for example using `file` command in Unix-based systems. Confirm it is in ASCII or UTF-8 format, and not in other formats like Little-endian UTF-16, Rich Text Format, or others.
- When requested to submit a file with one or more words/strings (e.g., regexp or words), you should submit a *file with each string on a new line, and nothing else*. Do not include your name, quotes, dummy lines, or any other information or character (e.g., do not type a line "`w1=abc`", but just "`abc`" without quotes).
- When requested to submit a file with a binary *yes/no* or *true/false* answer, you should submit a *file with a single 1 or 0, and nothing else* (except in the case of question 2.c(ii), where a string of 1's and 0's is required).
- When asked to submit a "`.jff`" file, it should be in proper JFLAP format. Your JFLAP files *must* run error-free in **version 7** of JFLAP. It is your responsibility to test your files in JFLAP 7 before submitting.

Submissions not compatible with these instructions will attract zero marks and do not warrant a re-submission.

Corrections: From time to time, students or staff find errors (e.g., typos or wrong symbols) in the assignment specification. In that case, a corrected version will be uploaded to the course web page as quickly as possible, an announcement will be made on the course web page as well as in the forum. Check the version on the bottom left.

Forum postings on assignment: **Never** post any information on the forum that may disclose how to solve a question or what the solution may be. You may only post assignment related questions for *clarification*, for example, to clarify certain notation. Any post discussing possible solutions or strategies may directly be considered plagiarism, see above. **If in doubt, do not post** and ask your tutor the question instead.

Regular Expressions Syntax: The $+$ operator in regular expressions is used in some books to denote one or more applications of Kleene star. However, in other places, such as JFLAP, $+$ denotes the alternation operator, equivalent to $|$. For the purpose of this assignment, we follow JFLAP and use operator $+$ to denote an alternation. In JFLAP, you can use λ or ϵ to denote the empty string (see Preferences menu).

Silence Policy: A silence policy will take effect 48hrs before this assignment is due. This means no questions about this assignment will be answered, whether they are asked on the discussion board, by email, or in person.

Exercise 1: Regular Expressions . . . . . **27 marks**

For parts (a) to (c), please type each word on a new line. The notation $w_i \cdot w_j$ denotes concatenation of words w_i and w_j, and w^r denotes the word obtained by reversing w.

(a) Let $R_1 = (a + b^*)cc^*a^*(c + b^*)^*$ and $R_2 = (a + b)^*c^*c(a^* + b)^*$ be two regular expressions.

i. (2 marks) [`E1-Qa1.txt`] Give two non-empty words w_1 and w_2 such that $\{w_1, w_2\} \subseteq L(R_1) \cap L(R_2)$.

Solution: ac, $acab$. Multiple answers exist.

ii. (2 marks) [`E1-Qa2.txt`] Give two non-empty words w_1 and w_2 such that $\{w_1, w_2\} \subseteq L(R_1) \setminus L(R_2)$.

Solution: cac, $bbcaac$. Multiple answers exist.

iii. (2 marks) [`E1-Qa3.txt`] Give two non-empty words w_1 and w_2 such that $\{w_1, w_2\} \subseteq L(R_2) \setminus L(R_1)$.

Solution: $caba$, bac. Multiple answers exist.

iv. (1 mark) [`E1-Qa4.txt`] Give a non-empty word $w \in L(R_1)$ such that $w \cdot w^r \in L(R_1)$.

Solution: bcb, $bcccb$. Multiple answers exist.

v. (1 mark) [`E1-Qa5.txt`] Give a non-empty word $w \in L(R_1)$ such that $w \cdot w^r \notin L(R_1)$.

Solution: acb, $accb$. Multiple answers exist.

vi. (1 mark) [`E1-Qa6.txt`] Give a regular expression R such that $L(R) = L(R_1) \cup L(R_2)$.

Solution: $((a + b^*)cc^*a^*(c + b^*)^*) + ((a + b)^*c^*c(a^* + b)^*)$.
The union of two regular expressions can be produced by putting an "or" operator between them.

vii. (3 marks) [`E1-Qa7.txt`] Give a regular expression R such that $L(R) = L(R_1) \cap L(R_2)$.

Solution: $(a + b^*)c^*ca^*b^*$.
This solution can be found by inspection by looking at which elements are shared between both expressions.

(b) Give regular expressions for the following languages.

i. (2 marks) [`E1-Qb1.txt`] $L_1 = \{a^n b^{4m+3} \mid n \geq 1, m \geq 0, (n + 1) \bmod 2 = 0\}$.

Solution: $a(aa)^*(bbbb)^*bbb$

ii. (2 marks) [`E1-Qb2.txt`] $L_2 = \overline{L_1}$, where $\overline{L}$ stands for the complement of L; we assume alphabet $\Sigma = \{a, b\}$.

Solution: Basically, a string is in $\overline{L_1}$ if it either contains even number of a's with no constraints on b's, b's such that their number is not equal to $4m + 3$ where $m \geq 0$ with no constraints on a's or any word in which a comes after b. This gives us the following regular expression:
$(aa)^*b^* + a^*(\lambda + b + bb + bbbb)(bbbb)^* + (a + b)^*b(a + b)^*a(a + b)^*$.

One can also obtain the regular expression mechanically using JFLAP by obtaining the complement of the automaton for L_1 and then producing its regular expression, which will be much larger and unreadable but still correct and equivalent to the "human" one.

iii. (2 marks) [`E1-Qb3.txt`] $L = \{w \mid w \in \{a, b\}^*, |w| \mod 4 = 0\}$, where $|w|$ denotes the length of word w.

Solution: $((a + b)(a + b)(a + b)(a + b))^*$

iv. (2 marks) [`E1-Qb4.txt`] $L = \{a^n w_1 \mid n > 0, w_1 \in ((\{a, b\}^* \setminus \{a, c\}^*) \cup \{c\})\} \cap \{w_2 b \mid w_2 \in \{a, c\}^*\}$.

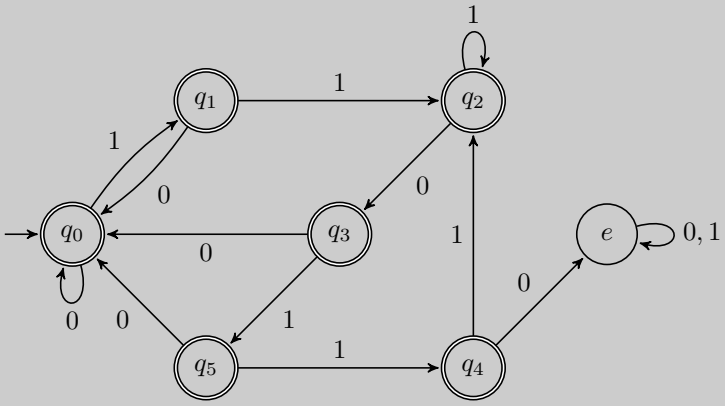
Solution: aa^*b

Reasoning: Let $L_1 = \{a^n w \mid n > 0, w \in ((\{a, b\}^* \setminus \{a, c\}^*) \cup \{c\})\}$ and $L_2 = \{wb \mid w \in \{a, c\}^*\}$. In terms of regular expressions, common suffix between L_1 and L_2 is b and common prefix is aa^*. The term $\{a, b\}^* \setminus \{a, c\}^*$ evaluates to the set $\{b\}^*$; taking the union of this with the set $\{c\}$ gives $\{b\}^* \cup \{c\}$. However the c cannot be part of L_1 as it must have the suffix b.

Exercise 1 continues on the next page. . .

v. (3 marks) [E1-Qb5.txt] $L = \{w \mid w \in \{0,1\}^*, w \text{ does not contain the substring } 110110\}$.

Solution: $(0 + 10 + 11(1 + 0111)^*0(0 + 10))^*(1 + 11(1 + 0111)^*(\lambda + 0(\lambda + 1(\lambda + 1))))$
The easiest way to do this question is to create the finite state automaton in JFLAP and convert it to a regular expression. The DFA looks like this:



You could also build an NFA of a machine that accepts strings with 110110 as a substring, convert it to a DFA, find its complement and then convert *that* to a regular expression.

(c) Let R_1 and R_2 be two regular expressions over a non-empty alphabet Σ. Answer true or false by placing a `1` for true and a `0` for false in a text file as you have for the previous questions.

i. (2 marks) [E1-Qc1.txt] If we assume that $L(R_1) \subset L(R_2)$, then it is the case that $L(R_1^*) \subseteq L(R_2^*)$.

Solution: The statement is true. Consider $R_1 = a$ and $R_2 = a^*$ over the alphabet $\Sigma = \{a\}$. Then $L(R_1) \subset L(R_2)$ because $L(R_1) = \{a\}$ with $a \in L(R_2)$, and $aa \in L(R_2)$ but $aa \notin L(R_1)$. Therefore, $L(R_1^*) = L(R_2^*) = L(a^*)$. That is, $L(R_1^*) \subseteq L(R_2^*)$.

ii. (2 marks) [E1-Qc2.txt] Given the above, is the following statement true? If $L(R_1^*) \subset L(R_2^*)$, then $L(R_1) \subseteq L(R_2)$.

Solution: The statement is false. Consider $R_1 = bb$ and $R_2 = b$. $L(R_1^*) = \{b^{2n} \mid n \geq 0\}$ and $L(R_2^*) = \{b^m \mid m \geq 0\}$. Hence, $L(R_1^*) \subset L(R_2^*)$ as $L(R_2^*)$ will contain both odd and even number of b's. Clearly, $\{bb\} \not\subset \{b\}$ hence $L(R_1) \not\subseteq L(R_2)$.

Exercise 2: Grammars ..... **26 marks**

(a) Provide regular expressions for the following languages.

i. (3 marks) [E2-Qa1.txt] Give a regular expression R such that $L(R) = L(G_1)$, where G_1 is:

$$S \to AbB$$
$$A \to Aab \mid \epsilon$$
$$B \to Bbc \mid Bab \mid C$$
$$C \to Ca \mid Cc \mid \epsilon$$

Solution: $(ab)^*b(a + c)^*(bc + ab)^*$

ii. (3 marks) [E2-Qa2.txt] Give a regular expression R such that $L(R) = L(G_2)$, where G_2 is:

$$S \to AbBCS \mid \epsilon$$
$$A \to Ac \mid Aa \mid \epsilon$$
$$B \to Ba \mid Bb \mid a \mid b$$
$$C \to Ca \mid Cb \mid Cc \mid \epsilon$$

Solution: $((c + a)^*b(a + b)(a + b)^*(a + b + c)^*)^*$

iii. (2 marks) [E2-Qa3.txt] Give a regular expression R such that $L(R) = L(G_1) \cup L(G_2)$.

Exercise 2 continues on the next page...

Solution: $((ab)^*b(a+c)^*(bc+ab)^*)+(((c+a)^*b(a+b)(a+b)^*(a+b+c)^*)^*)$

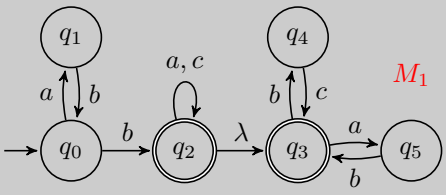
iv. (2 marks) [`E2-Qa4.txt`] Give a regular expression R such that $L(R) = L(G_1) \cap L(G_2)$.

Solution: Not disclosed at this point; now that you have the solutions for $L(G_1)$ and $L(G_2)$, you can re-submit before the class test for 5 marks, to recover your 2 marks (if you have not received them yet) and get 3 extra bonus ones! Instructions to follow...

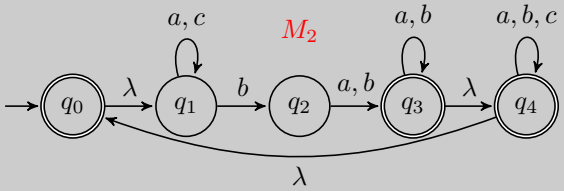
Solution: $bbc(ab+bc)^* + ab(ab)^*bc^*(\lambda + bc(ab+bc)^*) + (ba + ab(ab)^*bc^*a)(a+cc^*a)^*(\lambda + b(\lambda + (c + ab + bc)(ab+bc)^*) + cc^*(\lambda + bc(ab+bc)^*))$
This expression is extremely long and ugly, and hence you would not expect it to do this in your head. The question was indeed quite difficult, but it was just set at 2 marks only despite that (and because of that!).
However, with a some *research*, you will find mechanical ways to calculate the intersection of two regular languages. There are two ways that the intersection of two regular languages can be computed, using JFLAP for example.

The first is by using De Morgan's Law $A \cap B = \overline{\overline{A} \cup \overline{B}}$, where $\overline{A}$ is the complement of A; so $\overline{\overline{A} \cup \overline{B}}$ is the complement of the union of the complements of A and B. Check Theorem 7.4.3 (second edition) or Theorem 7.5.2 (third edition) in the book .

The second way involves computing the cross-product of the finite state machines which represent the languages you wish to find the intersection of. In this case M_1, where $L(M_1) = L(G_1)$ is:

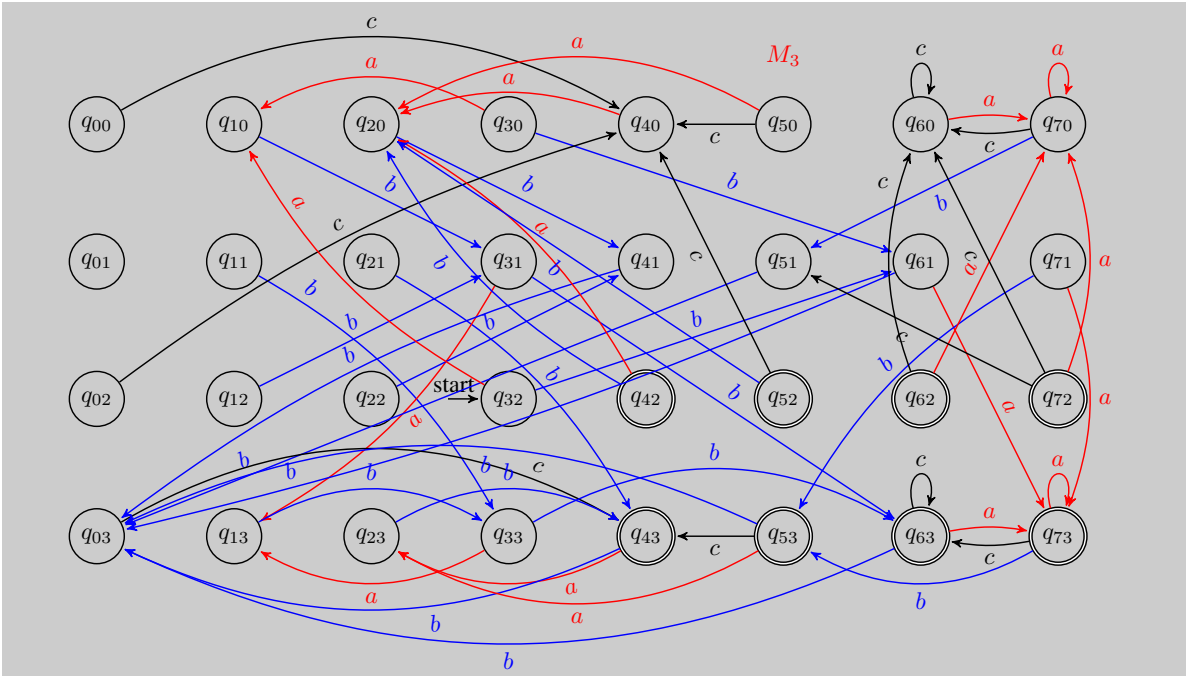


And M_2, where $L(M_2) = L(G_2)$ is:



After using JFLAP to convert M_1 and M_2 to DFAs and minimising, we can compute the cross-product M_3 of M_1 and M_2, where $L(M_1 \times M_2) = L(G_1) \cap L(G_2)$. This machine looks like this (after removing all superfluous transitions and states):

Exercise 2 continues on the next page...



In this machine, the state q_{ij} represents the combined states of state q_i from M_1 and state q_j from M_2. The initial (and final) states of M_3 are q_{ij} where q_i is an initial (or final) state in M_1 and q_j is an initial (or final) state in M_2. Transition $\delta(q_{ij}, x, q_{mn})$ for $x \in \Sigma$ exists in M_3 iff there exists a transition $\delta(q_i, x, q_m)$ in M_1 **and** a transition $\delta(q_j, x, q_n)$ in M_2. JFLAP can then be used to produce the regular expression. More information on constructing the cross-product of two finite state machines can be found here.

(b) Let $G_3 = (\{S\}, \{a, b\}, \Gamma, S)$ be a grammar, where the set of rules Γ is defined as follows:

$$S \to SabbS$$
$$S \to SbabS$$
$$S \to SbbaS$$
$$S \to \epsilon$$

i. (3 marks) [`E2-Qb1.txt`] Is G_3 an ambiguous grammar? Give your answer as 1 for *yes*, 0 for *false*.

Solution: Yes, because there exist two different left-most derivations for the same string. Consider the following partial derivations:

S
$\to SbabS$
$\to babS$
$\to babSbabS$
$\to babbab$

S
$\to SbabS$
$\to SbabSbabS$
$\to babbab$

Clearly both derivations can be generate $babbab$. Since there are 2 different left most derivations for the same string and hence it is ambiguous. Note, there may be other examples showing ambiguity in the grammar. Till the time an example shows 2 different left most derivations it is correct.

ii. (3 marks) [`E2-Qb2.txt`] Is there a relationship between the number of as and bs in $L(G_3)$? Give your answer as 1 for *yes*, 0 for *no*.

Solution: Yes, there must be twice as many bs as as.

iii. (4 marks) [`E2-Qb3.txt`] Does there exist a regular expression R such that $L(R) = L(G_3)$? If it exists, provide such R; otherwise, simply put 0.

Exercise 2 continues on the next page...

Solution: Yes. Basically, the grammar concatenates abb, bab or bba in any order. Hence an equivalent regular expression is $(abb + bab + bba)^*$.

(c) Given the language $L = \{a^n b^{3m+1} a^3 c^{2m} b^n \mid n, m > 0\}$.

i. (3 marks) [`E2-Qc1.txt`] Complete the following *context-free* grammar G such such $L(G) = L$.

$$S \to aSb \mid < \text{missing string} >$$
$$A \to bbbAcc \mid bbbBcc$$
$$B \to aaa$$

Provide the missing string as a single line in the given text file (e.g., if your response is $xSSy$, submit a file with the single line `xSSy`).

Solution: $abAb$
The first a and final b is because $n > 0$ and the middle b is to account for the $+1$ in b^{3m+1}.

ii. (3 marks) [`E2-Qc2.txt`] Does there exist a regular expression, DFA, regular grammar or PDA over the alphabet $\Sigma = \{a, b, c\}$ which is equivalent to the language L? (Answer the following question as a string of bits that translate to 1 for *yes* and 0 for *no*. For example, if your answer is "no, no, yes, no" give your response as `0010`).

Solution: `0001`
No regular expression exists for L, neither does a DFA or regular grammar. The context-free grammar is given in the previous question, which means that there must be a PDA as PDAs have the same expressive power as context-free grammars.

Exercise 3: Automata ..... **22 marks**

(a) Answer the following questions based on the finite state automaton M_1 present in the JFLAP file FA-3.a.jff available in Assessments section of the course website. Assume alphabet $\Sigma = \{a, b, c, d, e, f\}$.

i. (2 marks) [`E3-Qa1.txt`] Give four strings of length 4 accepted by M_1. Please type each string on a new line.

Solution: Multiple answers exist

ii. (2 marks) [`E3-Qa2.txt`] Give four strings of length 4 rejected by M_1. Please type each string on a new line.

Solution: Multiple answers exist

iii. (4 marks) [`E3-Qa3.txt`] Give the language of this machine M_1 as a regular expression.

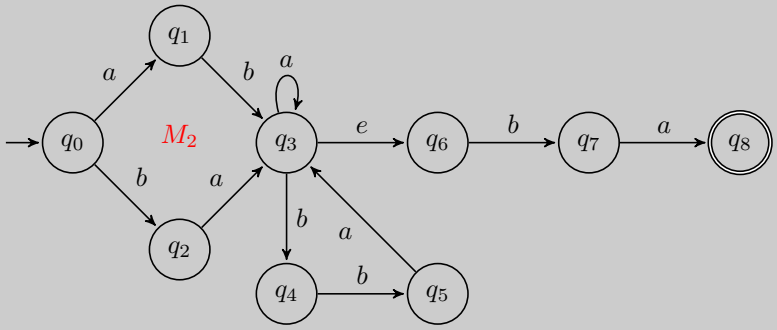
Solution: $(ab+ba)(a+fbad+bba)^*((e(a+b)^*)+(cc^*))$. Another answer could be $(ab+ba)(a^*+(fbad)^*+(bba)^*)^*(e(a+b)^*+(cc^*))$.

iv. (2 marks) [`E3-Qa4.jff`] Remove any redundant states from M_1 and adjust the transitions accordingly. Give your answer as a `.jff` JFLAP file.

Solution: States q_3 **or** q_6 are redundant. The machine can simply remove **one** of these states and add transitions $\delta(q_3, a, q_3)$ and $\delta(q_3, b, q_3)$ or $\delta(q_6, a, q_6)$ and $\delta(q_6, b, q_6)$ - depending on which state was removed.

v. (4 marks) [`E3-Qa5.jff`] Create an automaton M_2 (deterministic or non-deterministic) such that it accepts the language L where $L = L(M_1) \cap L((b + a)^*(a + bad + bba + fba)^* eba)$. Your machine should *not* accept words that are *not* in this language. Give you answer in a `.jff` JFLAP file.

Solution:

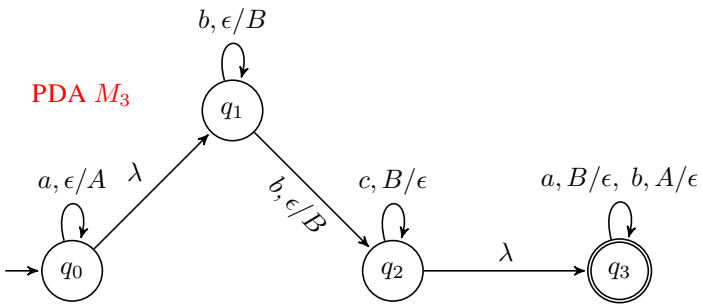


 Exercise 3 continues on the next page...

$(ab + ba)(a + (bba))^* eba$

Exercise 3 continues on the next page...

(b) Consider the pushdown automaton M_3 over the alphabet $\Sigma = \{a, b, c\}$ as shown below.



Notation $x, A/X$ means a transition where x is the input symbol being read, A is the symbol on top of the stack that is popped, and X is the symbol pushed onto the stack. The symbol ϵ stands for the "empty" string. Acceptance is by final state *and* empty stack.

i. (2 marks) [`E3-Qb1.txt`] Give 4 strings of length 6 over Σ that are accepted by PDA M_3. Remember to type each string on a new line.

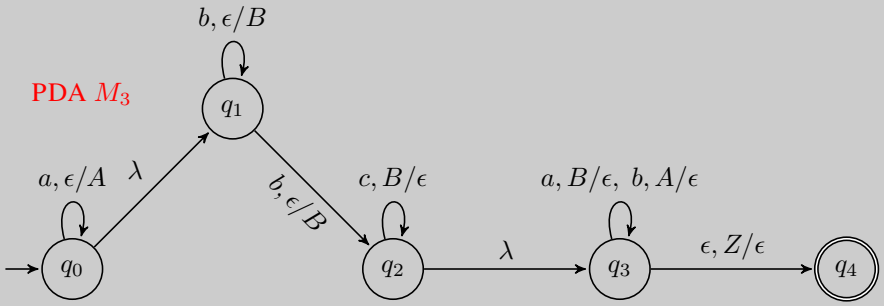
Solution: $abbcab, bbbcaa, abbccb, aababb$. Multiple answers exist.
The language $L(M_3)$ is $\{a^i b^j c^{j-k} a^k b^i \mid i, k \geq 0, j > 0, j - k \geq 0\}$

ii. (2 marks) [`E3-Qb2.txt`] Give 4 strings of length 6 over Σ that are rejected by PDA M_3. Remember to type each string on a new line.

Solution: Multiple answers exist.

iii. (4 marks) [`E3-Qb3.jff`] It is a well known result that every PDA with acceptance condition of an empty stack and reachability of a final state can be transformed to an equivalent PDA with acceptance condition requiring only reachability of a final state. Transform PDA M_3 to an equivalent PDA with acceptance requiring only reachability of a final state.

Solution:



JFLAP starts by pushing a Z symbol onto the stack. So, in every old final state, we must allow an empty transition to a special final state with no transition at all by popping Z from the stack. All old final states are not final anymore.

End of assignment paper

Question	Points
Regular Expressions	27
Grammars	26
Automata	22
Total:	75