COSC1107/1105 - Computing Theory
Assignment 1 (15%)

Total marks: 73

Deadline: Sunday August 18th 2019, 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. 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 are to **submit** all your files inside a .zip file and & **certify your submission** using this Google Form:

  `http://tinyurl.com/ct19-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 25 `.txt` and 5 `.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 or Rich Text Format.
- 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 "`w1=abc`", but just "`abc`" without quotes).
- 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.
- Question 5 is a *bonus* question. You can still get full marks without answering this question.

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 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 date 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.

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.

Late Submissions/Extensions: A penalty of 10% per day is applied to late submissions up to 5 days, after which you will lose ALL the assignment marks. Extensions will be given only in exceptional cases; please refer to Special Consideration process. Special Considerations given after solutions have been released (between 1 and 2 weeks after deadline) will automatically result in an **equivalent assessment in the form of a test**, assessing the same knowledge and skills of the assignment (location and time to be arranged by the instructor).

Exercise 1: Regular Expressions .... **20 marks**

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).

For parts (i) to (iii), 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 $R_1 = 1(1^* + 2^*)3^*(2^* + 3)1^*2(1 + 3)^*2^*$ and $R_2 = 1^*3(2 + 3)^*2^*(1 + 3)^*(1 + 3)^*$ be two regular expressions. Whenever asked to provide words, these must be non-empty ones (and different if more than one).

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: 132, 11332213. 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: 12, 132132. 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: 3, 132321. 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: 12, 123. 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: 12122, 13212. 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: $(1(1^* + 2^*)3^*(2^* + 3)1^*2(1 + 3)^*2^*) + (1^*3(2 + 3)^*2^*(1 + 3)^*(1 + 3)^*)$.
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: $11^*33^*22^*(\epsilon + 1(1 + 3)^* + 33^*(\epsilon + 22^* + 1(1 + 3)^*))$
This solution can be found by inspection by looking at which elements are shared between both expressions and reasoning on prefixes: any string have to include a prefix matching $11^*33^*22^*$ and then either terminate there or be extended in two manners.
Getting the three alternations is can be difficult by simple inspection. So, if the solution was not fully correct (in which case you full marks are awarded automatically), partial marks were given by *just* checking that the prefix $11^*33^*22^*$ was correctly captured. That must be true in every string and is accessible by simple inspection of the regular expressions.

(b) Give regular expressions for the following languages.

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

Solution: $aaaaa(aaa)^*(bb)^*bb$

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

Solution: $\overline{L_1} = L((aa + ((\lambda + a)(aaa)^*))b^* + a^*(\lambda + b(bb)^*) + (a + b)^*b(a + b)^*a(a + b)^*)$

As $\overline{L_1}$ is the complement of L_1, it accepts everything that L_1 *doesn't* accept. This means we can break the regular expression for $\overline{L_1}$ down, with respect to the properties of L_1, in the following way:

$(aa + ((\lambda + a)(aaa)^*))b^*$	incorrect number of a's, with no constraints on b's
$+a^*(\lambda + b(bb)^*)$	incorrect number of b's, with no constraints on a's
$+(a + b)^*b(a + b)^*a(a + b)^*$	any string with an a after 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] $L = \{w \mid w \in \{a, b\}^*, |w| \bmod 3 = 2\}$, where $|w|$ denotes the length of w.

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

iv. (2 marks) [E1-Qb4.txt]
$L = \{b^n w_1 \mid n > 1, w_1 \in ((\{a, c\}^* \cap \{a, b, c\}^*) \setminus (\{b\}^* \cup \{c\}))\} \cap \{w_2 c \mid w_2 \in \{a, b, c\}^*\}$.

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

Reasoning: Let $L_1 = \{b^n w_1 \mid n > 1, w_1 \in ((\{a,c\}^* \cap \{a,b,c\}^*) \setminus (\{b\}^* \cup \{c\}))\}$ and $L_2 = \{w_2 c \mid w_2 \in \{a,b,c\}^*\}$. In terms of regular expressions, common prefix between L_1 and L_2 is bbb^*; this is because $n > 1$, so $n \geq 2$. The common suffix is a bit more tricky. We aren't allowed to have w_1 be the empty word or c, because we are subtracting the set $\{b\}^* \cup \{c\}$ which contains the empty word and c. So the regular expression for for w_1 is $(a + ac + ca + cc)(a + c)^*$, because we can have anything we like over $\{a, c\}$, but we need to have at least 1 a, and we can't have a single c. L_2 also has an extra c as a suffix as well, after w_2. The term $\{a,c\}^* \cap \{a,b,c\}^*$ evaluates to the set $\{a,c\}^*$, so the middle of the expression is just $(a+c)^*$

Exercise 2: Grammars .... **18 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:

$$\begin{aligned} S &\longrightarrow AcB \\ A &\longrightarrow ac \mid bC \mid \epsilon \\ B &\longrightarrow baB \mid caB \mid D \\ C &\longrightarrow bC \mid \epsilon \\ D &\longrightarrow aD \mid b \end{aligned}$$

Solution: $(ac + b^*)c(ba + ca)^*a^*b$

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

$$\begin{aligned} S &\longrightarrow ACS \mid \epsilon \\ A &\longrightarrow aA \mid bA \mid bB \\ B &\longrightarrow cB \mid \epsilon \\ C &\longrightarrow bcC \mid acC \mid D \\ D &\longrightarrow bD \mid \epsilon \end{aligned}$$

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

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

Solution: $((ac + b^*)c(ba + ca)^*a^*b) + (((a+b)^*bc^*(bc+ac)^*b^*)^*)$

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

Solution: $bb^*c(((ba+ca)^*b) + ((ba+ca)^*aa^*b))$

There are many other more complex perfect solutions like:
- $bb^*c(ca)^*aa^*b + bb^*c(ca)^*b(a(ca)^*b)^*(\lambda + a(ca)^*aa^*b)$; or
- $bb^*(c(ba)^*b + c(ba)^*aa^*b) + bb^*c(ba)^*c(a(ba)^*c)^*(a(ba)^*b + a(ba)^*aa^*b)$.

While it is possible to get this by inspection and hard reasoning by cases, it ended up more involved than expected. Because of that, this question will be **moved to bonus marks**: if you got any point, it will be additional to the core ones!

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

$$\begin{aligned} S &\longrightarrow aSbSa \\ S &\longrightarrow bSbSa \\ S &\longrightarrow aSbSb \\ S &\longrightarrow \epsilon \end{aligned}$$

i. (4 marks) [E2-Qb1.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.

Solution: No, the language is context free.

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

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

$$\begin{aligned} S &\longrightarrow ccSa \mid < \text{missing string} > \\ A &\longrightarrow aAbb \mid aBbb \\ B &\longrightarrow ccc \end{aligned}$$

Provide the missing string as a single line in the given text file (e.g., if your response is $xSSy$, submit a file with a single line containing string "xSSy" (of course, without the quotes)).

Exercise 2 continues on the next page...

Solution: $cAba$

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 . . . . **25 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: $(bab + abca)^*(abab^* + c(a + b)(a + b)^* + \lambda)$.

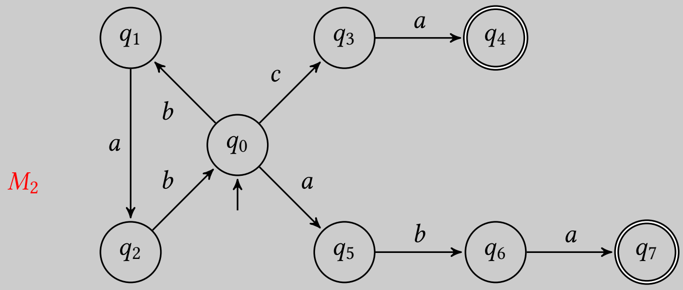
iv. (2 marks) [`E3-Qa4.jff`] Remove any redundant states from M_1 and adjust the transitions accordingly. Give your answer as a `.jff` JFLAP file.

Solution: States q_4 **or** q_5 are redundant. The machine can simply remove **one** of these states and add transitions $\delta(q_4, b) = q_6$ or $\delta(q_5, b) = q_6$ - depending on which state was removed.

- The solution *must* have *less number* of sates than M_1; if the answer has more states then it can never be correct. The idea of removing redundant states is always to get simpler machines. The original machine had 10 states; the solution machine must have 9 states.
- The task asked was to *remove* redundant states, not to change the machine. Any alteration of the machine, even if accepting the same language does not comprise as a solution to the question asked. The task was not to minimize the machine, that is a different task.

v. (4 marks) [`E3-Qa5.jff`] Create an automaton M_2 (deterministic or non-deterministic) such that it accepts the language L_1 where $L_1 = L(M_1) \cap L\big((aca + aba + bab)^*(a(ba)^* + c^*a)\big)$. Your machine should *not* accept words that are *not* in this language. Give your answer in a `.jff` JFLAP file.

Solution:



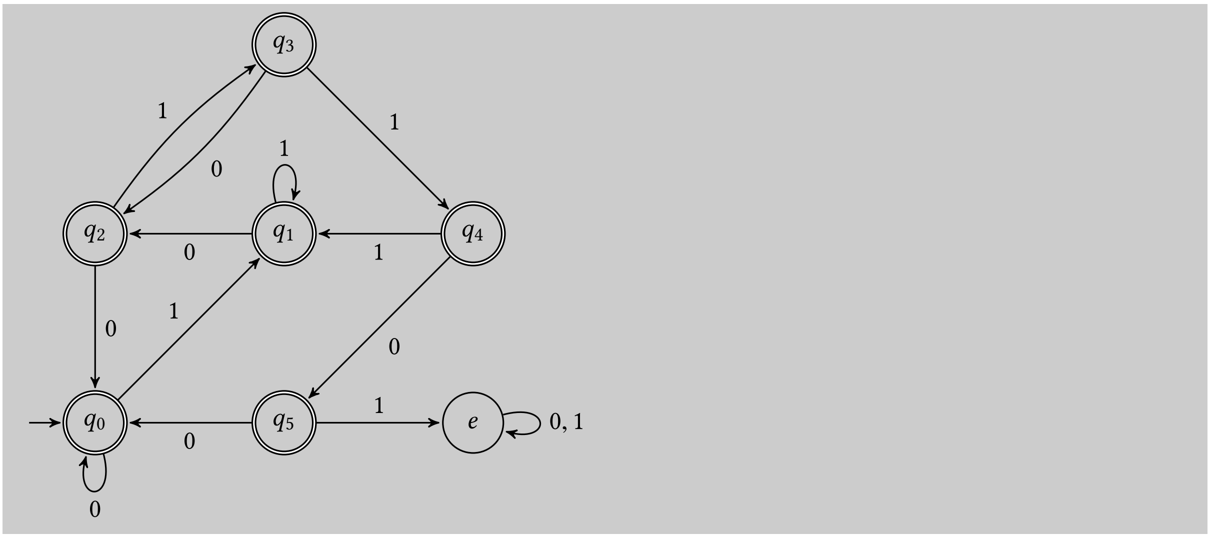
$(bab)^*(aba + ca)$

(b) Let $L_1 = \{w \mid w \in \{0, 1\}^*, w \text{ does not contain the substring } 101101\}$.

i. (4 marks) [`E3-Qb1.jff`] Give the DFA M_3 where $L(M_3) = L_1$.

Solution: The DFA looks like this:

Exercise 3 continues on the next page. . .



ii. (3 marks) [E3-Qb2.txt] Give the regular expression R such that $L(R) = L_1$.

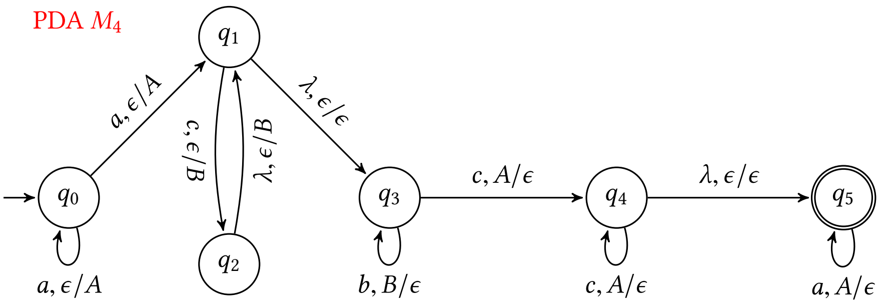
Solution: The easiest way to do this question is to just convert your DFA from the previous question to a regular expression.
The expression it produces is very complicated:

$$(0 + 11^*00 + 11^*01(01)^*00 + 11^*01(01)^*1(11^*01(01)^*1)^*(11^*00 + 11^*01(01)^*00) + 11^*01(01)^*1$$
$$(11^*01(01)^*1)^*00)^*(\lambda + 11^* + 11^*0 + 11^*01(01)^*(\lambda + 0) + 11^*01(01)^*1(11^*01(01)^*1)^*$$
$$(\lambda + 11^* + 11^*0 + 11^*01(01)^*(\lambda + 0)) + 11^*01(01)^*1(11^*01(01)^*1)^*0).$$

So you would not be expected to work it out in your head.

You could also build an NFA of a machine that accepts strings with 101101 as a substring, convert it to a DFA, find its complement and then convert *that* to a regular expression.

(c) Consider the pushdown automaton M_4 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. Here, λ stands for the "empty" input and ϵ stands for the "empty" stack symbol. Acceptance is by final state *and* empty stack.

i. (2 marks) [E3-Qc1.txt] Give 4 strings of length 11 over Σ that are accepted by PDA M_4. Remember to type each string on a new line.

Solution:
The language $L(M_4)$ is $\{a^i c^j b^{2j} c^{i-k} a^k \mid i \geq 1,\ j, k \geq 0,\ i - k \geq 1\}$

The only 4 strings of length 11 over Σ are:
- aaaacbbcccc
- aaaacbbccca
- aaaacbbccaa
- aaaacbbcaaa

ii. (2 marks) [E3-Qc2.txt] Give 4 strings of length 11 over Σ that are rejected by PDA M_4. Remember to type each string on a new line.

Solution: Multiple answers exist.

Exercise 4: Telephone System Design .... **10 marks**

Answer the following questions by submitting a JFLAP file for each.

(a) (5 marks) `[E4-Qa1.jff]` You have been tasked to design part of the internal phone system for a company that is expanding its operations from Melbourne to include a new Sydney office. Someone else in your the team will handle the internal routing of the calls; your job is to design a *deterministic* finite state machine that takes a number as input and determines whether it is valid (or should be rejected). The input to the system is a string composed in the following way:

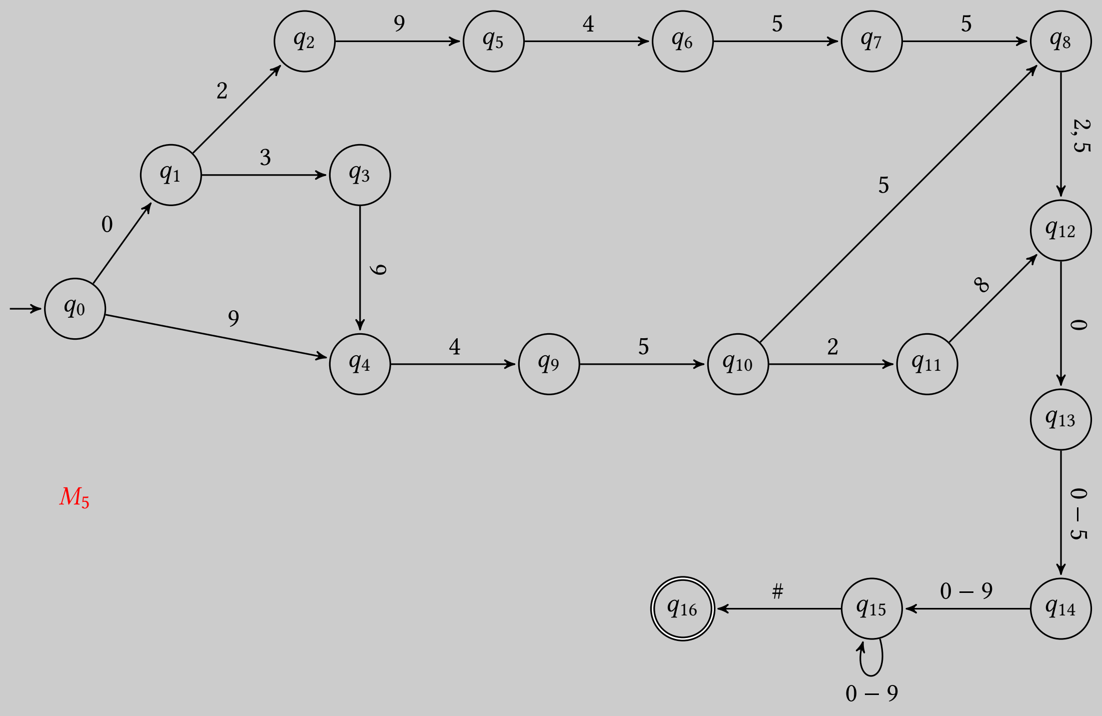
- The first part is optional and carries the *area code* of the office you want to be connected to:
  - Melbourne is 03.
  - Syndey is 02.
- The next 2 digits are 94 (regardless of the city).
- The next 3 digits determine the *department* you want to be connected to:
  - Marketing is 555.
  - R&D is 528.
  - Sales is 552.
- The final three (meaningful) digits indicate the *personal extension* of whoever you are calling:
  - Extensions are all in the range 000-059.

Besides the above format for numbers, your automaton must adhere to the following rules:

- As with all phone numbers, you should be able to enter any digits after the personal extension digits and still be connected, however *you must end your string with a # symbol* to tell the system the input is complete; otherwise the whole number is deemed invalid.
- The area code is also optional. If you are in Melbourne, you should not have to dial 03 to connect to the Melbourne office, so your automaton should accept numbers *with* an area code and numbers *without* an area code. For example, to be connected to extension 029 in the Melbourne sales department you would dial 0394552029#. However, 94552029#, 0394552029123456# and 94552029123456# should also be accepted. Your string must terminate with a # symbol, so ***you cannot have 94552029 or 94552029123456, nor can you have 94552029#123678.***
- Finally, as the Sydney office is just being set up, they do not have an R&D department yet, any phone call directed explicitly to the Sydney R&D apartment should also be rejected. If there is no area code you do not need to worry about this.

Any number that does not fit the pattern, as explained above, is invalid (and should be rejected).

Solution:



This is one solution, there are many more possible.

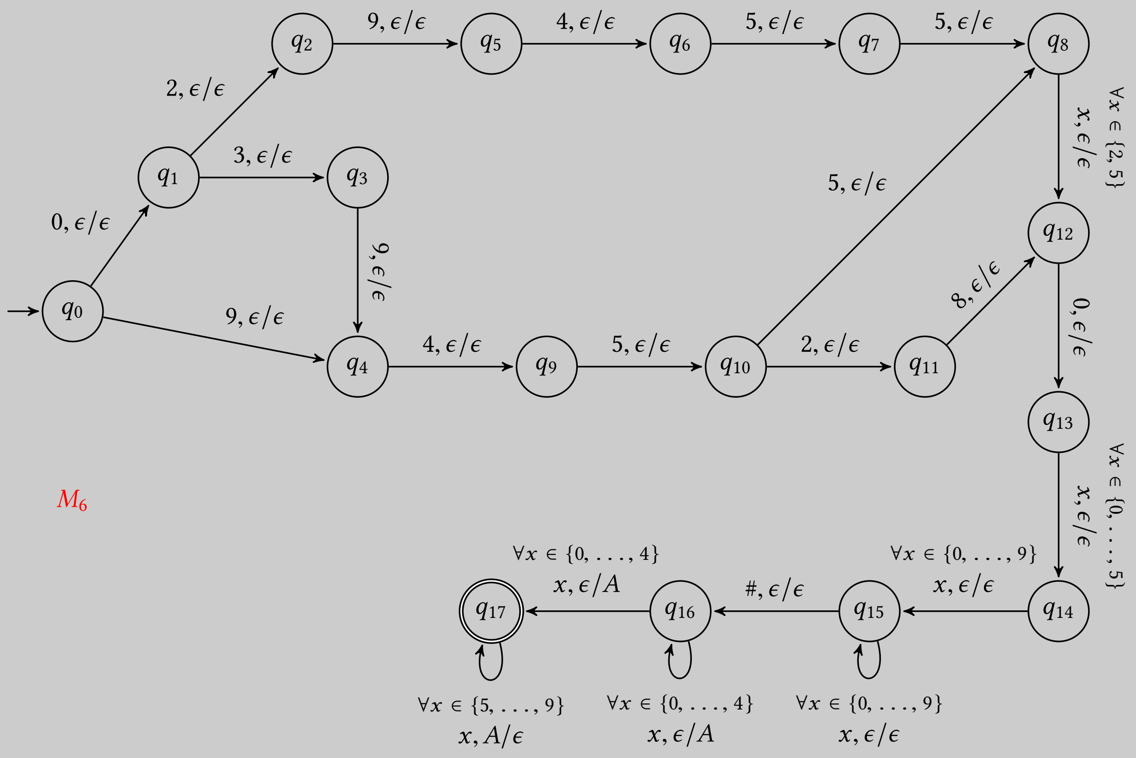
Technically the δ-function for a DFA should be total (i.e., it should include an error state), however we often omit these error transitions for the sake of neatness in diagrams. Furthermore, two DFAs that accept the same language will be considered equivalent by JFLAP even if one has a total δ-function and one does not.

Exercise 4 continues on the next page...

(b) (5 marks) `[E4-Qb1.jff]` Several months later your company is again contracted to make some modifications to the system. This time they want exactly the same functionality, however now they would like the user to be able to *enter their customer number after the # symbol.* The only catch is that the customer numbers have been designed by an ex-Computing Theory student and are needlessly complicated! The numbers are split up into two parts *of equal length*; with the first part being composed of the digits 0-4 and the second part composed of the digits 5-9. These parts can be any length, but they *must* be of equal size (and they must exist!). The rest of the machine should function exactly as before. The company asked you for the *simplest type of automaton possible* as they want to save money. Remember that whenever we use PDAs, the acceptance condition used is *both* final state *and* empty stack.

To use a similar example as before, if your customer number was 123678 and you wanted to be connected to extension 029 in the Melbourne sales department you would dial 0394552029#123678. However, 94552029#123678, 0394552029123456#123678 and 94552029123456#123678 should also be accepted.

Solution:



This is one solution, there are many more possible but they should all be a PDA (or more powerful).

Exercise 5 (Bonus): Intersection of Regular Languages .... **5 bonus marks**

`[E5-Qa1.txt]` Find the regular expression R_3 such that $L(R_3) = L(R_1) \cap L(R_2)$, where:

- $R_1 = (ca)^*a(c+b)^*(ab+ca)^*$; and
- $R_2 = ((b+c)^*a(c+a)(c+a)^*(c+a+b)^*)^*$.

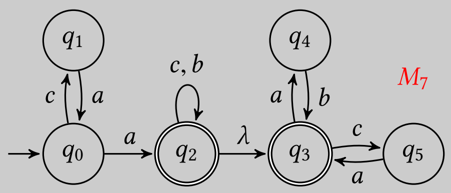
Solution: $R_3 = aab(ca+ab)^* + ca(ca)^*ab^*(\lambda + ab(ca+ab)^*) + (ac + ca(ca)^*ab^*c)(c+bb^*c)^*(\lambda + a(\lambda + (b+ca+ab)(ca+ab)^*) + bb^*(\lambda + ab(ca+ab)^*))$

This expression is very long and ugly, and hence you would not expect it to do this in your head. The question was indeed quite difficult and would involve a bit of research and going beyond the core expectation (that's why the bonus marks!).

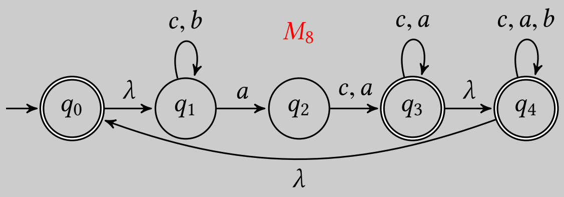
With a bit of *research*, you will find mechanical ways to calculate the intersection of two regular languages. There are, at least, **two ways** that the intersection of two regular languages can be computed, using JFLAP for example.

Using De Morgan's Law. The first approach uses 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 . More information about applying this method in JFLAP can be found here.

Using cross-product of automata. 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_7, where $L(M_7) = L(R_1)$ is:

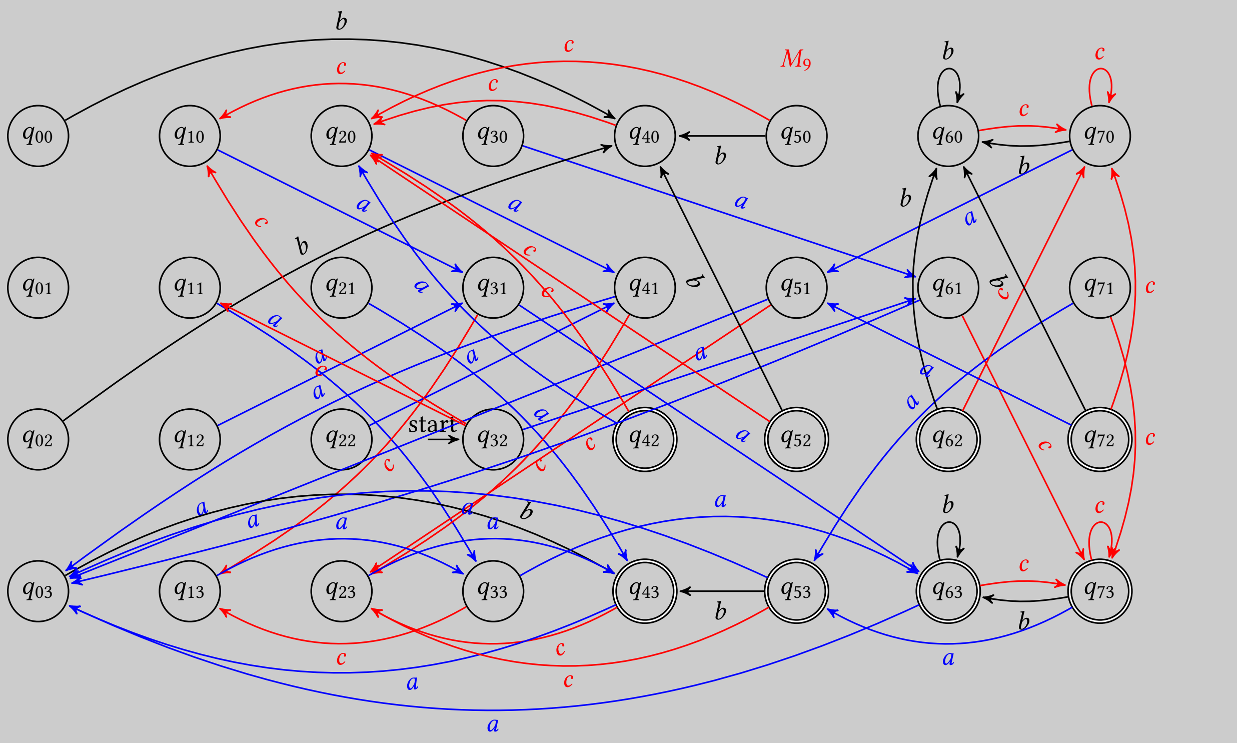


And M_8, where $L(M_8) = L(R_2)$ is:



After using JFLAP to convert M_7 and M_8 to DFAs and minimising, we can compute the cross-product M_9 of M_7 and M_8, where $L(M_7 \times M_8) = L(R_1) \cap L(R_2)$. This machine looks like this (after removing all superfluous transitions and states):

Exercise 5 continues on the next page...


M_9
start
q_{00} q_{10} q_{20} q_{30} q_{40} q_{50} q_{60} q_{70}
q_{01} q_{11} q_{21} q_{31} q_{41} q_{51} q_{61} q_{71}
q_{02} q_{12} q_{22} q_{32} q_{42} q_{52} q_{62} q_{72}
q_{03} q_{13} q_{23} q_{33} q_{43} q_{53} q_{63} q_{73}
a
b
c

In this machine, the state q_{ij} represents the combined states of state q_i from M_7 and state q_j from M_8. The initial (and final) states of M_9 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_9 iff there exists a transition $\delta(q_i, x, q_m)$ in M_7 **and** a transition $\delta(q_j, x, q_n)$ in M_8. JFLAP can then be used to produce the regular expression. *Isn't this just beautiful?*

More information on constructing the cross-product of two finite state machines can be found here.

— End of assignment paper —

Question	Points	Bonus Points
Regular Expressions	20	0
Grammars	18	2
Automata	25	0
Telephone System Design	10	0
Intersection of Regular Languages	0	5
Total:	73	7