COSC1107/1105 - Computing Theory
Assignment 2 (15%)

Total marks: 90 (15% whole course)

Deadline: Sunday October 13th 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. 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 are to submit a PDF file and **one zip file** named with your student number (e.g., `3451234.zip`) containing (in its root folder) your three `.jff` files for exercise 1 via the following Google Form:

`http://tinyurl.com/ct19-ass2-sub`

The PDF file can have *any* name as long its extension is `.pdf`, but it must must be typeset in a word processor (scanned files will *not* be marked) and have your student name and number on it. Python validator scripts for the zip file can be obtained here. The three JFLAP files must follow the exact name conventions as specified below in each question, and must load & run in JFLAP 7.0 with no errors whatsoever in reasonable amount of time (see below).

In the submission form you will be required to certify that the submitted solution *represents your own work only* by agreeing to the following statement:

> *I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honor code by typing "Yes":*

If you are not able to certify this, it is best not to submit anything. Submissions not compatible with the instructions above will attract zero marks and do not warrant a re-submission.

Corrections: From time to time, students or staff find errors (e.g., typos, wrong/missing symbols, ambiguous text) 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 us 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: Turing Machine Design .... **20 marks**

In this exercise you will build a few Turing Machines in JFLAP. Your machines will be fully automarked, so you **must** follow the following conventions (please read them carefully and ask if in doubt[1]):

- Your Turing machine will *always* start with the head in the *first symbol of the input on the tape*. For example, if your initial state is q_0 and the input string is $abcd\#cba$, then the initial configuration would be $q_0, B\underline{a}bcd\#cbaB$, where B is the blank symbol (in red for better legibility), and the underlined symbol is where the head is reading. Note this is a different convention to that used in many machines done in tutorials where the head was assumed to begin reading the blank symbol on the left of the input, and so one must do a $B/B, R$ initial transition.
- The *acceptance condition* to be used will be *halting in a final state*, as in the Sudkamp's book.
  - JFLAP has two acceptance conditions under Preferences, one can make JFLAP *"Accept by final state"* or *"Accept by halting"*, or by either of them (if both selected). We recommend to work with only *"Accept by final state"* selected, but remember that by itself *"Accept by Final State"* does not require halting, so think how to build your machine to capture the halting in final state acceptance condition that shall be used.
- The *output* of a Turing machine (when it accepts) consists of the character directly under the tape head until the first blank to the right consists of the symbol underneath the tape head and all symbols to the right of the tape head until the blank is encountered. In the event that the tape head is on a blank, the output is the empty string. For example, if the Turing Machine accepts with the tape left as $Baaaa\underline{a}bcd\#cbaB$, then the output of the Turing Machine is the string $abcd\#cba$. When a Turing Machine rejects, what is left in the tape is irrelevant—there is no output.
- Your machines must be syntactically error-free, and load and terminate *always* within a reasonable amount of time using JFLAP version 7: at most 5 secs. per run. These two are *minimal standards* that must be met. Machines with syntactic errors or requiring more than 5 seconds on *any* input will be awarded zero marks.
- No manual changes or repairs will be done to any of the submitted files. It is your responsibility to test your files in JFLAP 7 before submitting (you have the tool!).
- Your TM has to be one-tape standard "`Turing Machine`" in JFLAP (no multi-tape machine or block machine); refer to post @151. You are free however to use any advanced features provided by JFLAP for standard Turing Machines, as long as the machine submitted solves the problem. Please refer to some information and tips on JFLAP at the end of the document.
- You can assume that any input given to your machine will have the *correct shape*, so you do not need to check for the input having a valid form.

(a) (8 marks) `[E1-Qa1.jff]` Build a *deterministic* Turing machine that takes two words $w_1, w_2 \in \{a,b,c,d\}^*$, separated by a $\#$ symbol in the form $Bw_1\#w_2B$ and *determines* (by accepting/rejecting) whether string w_2 is a substring of w_1.

(b) (8 marks) `[E1-Qb1.jff]` Build a *deterministic* Turing machine that takes an input string $w \in \{a,b,c,d\}^*$ and replaces it with w^r, the reverse of w. After reversing the string, the machine must accept.

For example:

- When run on input $B\underline{a}bcdabcdB$, it should leave $B\underline{d}cbadcbaB$ on the tape.
- When run on input $B\underline{a}aaaaaaaB$, it should output $B\underline{a}aaaaaaaB$.
- When run on input $B\underline{a}bccdabcddacB$, it should output $B\underline{c}addcbadccbaB$.

Remember the output starts on the symbol the head is left until the first blank on the right. Also recall the machine must always accept (after producing the output, of course!).

(c) (4 marks) `[E1-Qc1.jff]` Combine your machines to produce a single *deterministic* machine that takes an input of the form $Bw_1\#w_2B$ and if if w_2 is a substring of w_1, it replaces w_1 with w_1^r and accepts; otherwise (if w_2 is *not* a substring of w_1), the machine rejects. When rejecting, it does not matter what is left on the tape (i.e., the tape can be left in any way).

For example:

- When run on input $B\,\overbrace{\underline{a}bcdabcdabcd}^{w_1}\#\overbrace{abcda}^{w_2}\,B$ it should accept and leave $B\,\overbrace{\underline{d}cbadcbadcba}^{w_1^r}\#\overbrace{abcda}^{w_2}\,B$ on the tape.
- When run on input $B\,\overbrace{\underline{b}bbbaaaaa}^{w_1}\#\overbrace{aaabaaa}^{w_2}\,B$, it should just reject (what is left on the tape is irrelevant).

[1] If anything is unclear here, or if you are stuck, you can ask general questions about JFLAP or Turing machines on the discussion board, or otherwise come and see Sebastian or Angus at one of their consultation times.

Exercise 2: Undecidability .... **20 marks**

Let $L_1 = \{M \mid M$ is a TM that halts on the empty tape leaving exactly two words on its tape in the form $Bw_1Bw_2B\}$.

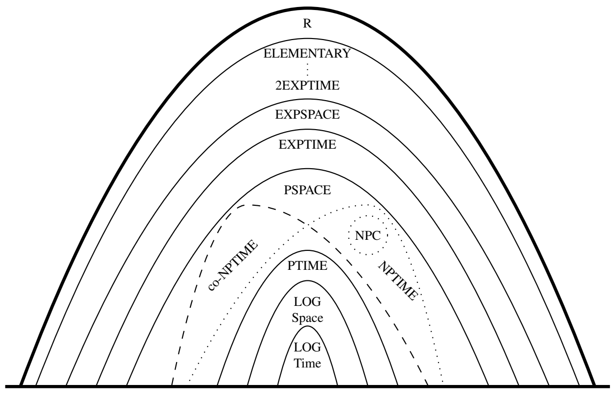
(a) (15 marks) You have seen the proof in tutorial 7 (PART 2) that A_{TM}, the problem of deciding whether an arbitrary Turing machine will *accept* an arbitrary input, is undecidable. Use this result to prove, formally using problem reduction, that given an arbitrary Turing machine M, the problem of deciding if $M \in L_1$ is undecidable.

(b) (5 marks) Is L_1 recursive, recursively enumerable, non-recursively enumerable, uncomputable? Justify your answer.

Exercise 3: Complexity .... **25 marks**

(a) (6 marks) Prove mathematically that if a Turing Machine runs in time $\mathcal{O}(g(n))$, then it runs in time $\mathcal{O}(h(g(n)) + c)$, for any constant $c \geq 0$ and any functions $g(n)$ and $h(n)$ where $h(n) \geq n$.

(b) (14 marks) A cutting-edge publisher based in Melbourne is publishing a book on complexity theory. They learnt that you are taking Computing Theory at RMIT, so they offered you a very well paid job! For the job, you are asked to write the part of the book that explains the following picture:



Your text will go to a proper book, so it has to be professional, well written, and clear. And of course, you are to explain the figure the best way possible! The publisher told you that you can use no more than one page.

(c) (5 marks) Consider the following problem SC: "given a set of elements $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ and a set $\mathcal{S}$ of subsets of $\mathcal{U}$ (i.e., $\mathcal{S} \subseteq 2^{\mathcal{U}}$) where $\bigcup_{s \in \mathcal{S}} s = \mathcal{U}$, *find the smallest set* $\mathcal{C} \subseteq \mathcal{S}$ such that $\bigcup_{c \in \mathcal{C}} c = \mathcal{U}$?"

The SC problem is a very famous one, and one much studied. Explain why stating that SC is the NP-complete complexity class (or SC is an NP-complete problem) would be *technically incorrect* to say. **<u>Note:</u>** A perfect answer can be stated in no more than 3 short sentences.

(d) (5 points (bonus)) In no more than half a page, explain and discuss the the *phase transition phenomenon* in random Random 3-SAT. **<u>Note:</u>** This requires some research. Marks will depend on quality of presentation, and depth and breath of the explanation.

Exercise 4: Pumping Lemma & Regularity .... **25 marks**

(a) (6 marks) It is Novemeber 2021. Because you did so well in Computing Theory in 2019, the course instructor (a strange guy with a weird accent who walks a lot during lectures) asked you to tutor for it, which of course you accepted given how much you liked the course! The term is almost over, and it is exam marking time. In one of the questions, students were asked to state the Pumping Lemma for regular languages. The question is worth 10 marks. You take the next exam to mark and you read the following answer from the student:

Pumping Lemma: Let L be a regular language. Then, for every integer $n \geq 1$ and every $w \in L$ with $|w| \geq n$, there exist string $x, y, z \in \Sigma^*$, such that the following holds:

1. $w = xyz$;
2. $|xy| \leq n$;
3. $y \neq \epsilon$; and
4. $xy^iz \in L$, for all $i \geq 0$.

You are amazed by the student's handwriting (it is just beautiful!). To mark it, you are asked to:

Exercise 4 continues on the next page...

1. Specify how many marks you will award to the student's answer. Remember the question is worth 10 marks.
2. If you found the answer wrong, provide *detailed feedback* to the student in one paragraph. Just identifying the mistake(s) is not enough, you must explain to the student why the answer is wrong, how it would not work, etc. Be as *clear, complete, and intuitive* as possible to help the student to learn the Pumping Lemma!

(b) (6 marks) **Emma's Pumping Lemma Dilemma:** For her job interview in a highly-regarded software company, an ex-student of CT, Emma, has been asked to prove that $L = \{yyy \mid y \in \{0,1\}^*\}$, over alphabet $\Sigma = \{0,1\}$, is not a regular language. She was allowed to submit her proof the next day. That night, she finally finished the proof, wrote it all down neatly, and went to bed. During the night, burglars came into her house with their dog, and the dog chewed up her proof. In an attempt to cover their tracks, the burglars, who also happened to have taken CT years before, tried to recreate the proof. They gathered all the shreds they could find, but on some of them the ink had washed out from the dog's saliva, and so they couldn't read some parts of it. Can you help them put the proof back together? For each shred they found, they rewrote it on a piece of paper. If they couldn't rewrite it, they wrote different interpretations of what they thought it should say onto different pieces of paper. Now they have many pieces of paper, but they don't know which of them are correct and in which order they need to go. Help Emma's burglars!

There is one big piece, that reads:

> Here is my proof for your request during your interview. To prove that $L = \{yyy \mid y \in \{0,1\}^*\}$, over alphabet $\Sigma = \{0,1\}$, is not regular, I will appeal to the well-known Pumping Lemma, that I studied during my Computing Theory course in 2014. If you are not familiar with it, you can check my lecturer Sebastian Sardina's slides here.

Here are the other pieces they've transcribed:

(a) Now consider $w = (0^k10^k10^k1)$.

(b) Now consider $w = (01)^k(01)^k(01)^k$.

(c) Then, it follows that $w \in L$ and $|w| = 3k+3 \geq k$.

(d) Then, it follows that $w \in L$ and $|w| = 6k \geq k$.

(e) So, according to the Pumping lemma, there exist substrings $x, y, z \in \Sigma^*$ such that (1)-(4) hold.

(f) Thus $w = 0^k10^k10^k1 = xyz$ such that $|xy| \leq k$ and $|y| \geq 1$.

(g) Thus $w = (01)^k(01)^k(01)^k = xyz$ such that $|xy| \leq k$ and $|y| \geq 1$.

(h) So, it follows that $x = 0^q, y = 0^r, z = 0^s10^k10^k1$, where $q + r \leq k$, $r \geq 1$, $s \geq 0$, and $q + r + s = k$.

(i) So, it follows that $x = u^q, y = u^r, z = u^su^ku^k$, where $u = 01$, $q + r \leq k$, $r \geq 1$, $s \geq 0$, and $q + r + s = 2k$.

(j) Consider now $i = 0$. Then, $xy^iz = xy^0z = xz = 0^q0^s10^k10^k1 = 0^{q+s}10^k10^k1$.

(k) Consider now $i = 0$. Then, $xy^iz = xy^0z = xz = (01)^q(01)^s(01)^k(01)^k = (01)^{q+s}(01)^k(01)^k$.

(l) Now, because $q + r + s = k$ and $r \geq 1$, we know that $q + s \neq k$, and therefore, $xy^iz \notin L$.

(m) Assume that L is regular. Then, there exists a $k \in \mathbb{N}$ as per the Pumping Lemma.

(n) Thus we have a contradiction: our assumption (that L is regular) must be false and L is not regular.

(o) Thus we have a contradiction: our assumption (that L is regular) must be true and L is true.

You need not write out the entire proof, just **provide the corresponding letters in the correct order as a word**. For example, the word **badf** denotes the proof built from statements (b), (a), (d), and (f) in that exact order. And remember: the burglars wrote down multiple different interpretations of the same piece of the proof, so not all of these will be necessary or correct. However, all necessary pieces are listed above.

(c) (9 marks) Let $L = \{a^{(n+3)(n-2)} \mid n \geq 2\}$. Prove that there is no DFA for L by completing the following proof. Concretely, you need to fill each of the six missing parts.

1. Assume that there is a DFA M which accepts L and hence that language L is regular.
2. Then, the Pumping Lemma applies for some $n \geq 1$.
3. Then, consider string $w =$ ______________, where $k > n$.
4. Clearly, $w \in L$ and $|w| > n$.
5. So, by the Pumping Lemma, there exist words x, y, z such that $w = xyz$ with ______________, $y \neq \epsilon$, and $xy^iz \in L$ for all $i \geq 0$.

Exercise 4 continues on the next page...

6. Because $|xy| \leq n < k$, it must be the case that ______________________________.
7. Now consider the word $w' =$ _______________.
8. By the Pumping Lemma, it follows that $w' \in L$.
9. However, ______________________________.
10. So we reach a contradiction, and L cannot be regular.
11. Finally, due to the _______________ we know that a language is regular if and only if it can be recognised by a DFA. Therefore, since L is not regular, there is no DFA that recgonises language L.

(d) (4 marks) Let $L_2 = \{0,1\}^* \setminus \{yyy \mid y \in \{0,1\}^*\}$ be a language over $\Sigma = \{0,1\}$. Prove that there is no DFA M such that $L(M) = L_2$. Your proof should be short and not use the Pumping Lemma.

End of assignment paper

Question	Points	Bonus Points
Turing Machine Design	20	0
Undecidability	20	0
Complexity	25	5
Pumping Lemma & Regularity	25	0
Total:	90	5

Some tips for JFLAP:

- JFLAP has a number of shortcuts that you can use to avoid having to use too many transitions. For instance, the ! symbol can be used to indicate "anything but". So if you wanted to move the head right on the string until it reaches the character a, you can use the transition $(!a/\sim, R)$. Here, $\sim$ tells JFLAP to use the last character read. In order to move until the next blank use the transition $(!/\sim, R)$ — JFLAP inserts the blank symbol automatically.

- Keep in mind though, that you cannot combine these transitions to say "anything but a or b" by having the transitions $(q_i, !a) = (q_i, \sim, R)$ and $(q_i, !b) = (q_i, \sim, R)$. This is because if the input alphabet is $\Sigma = \{a,b,c,d\}$, $L_1 = \{a\}$ and $L_2 = \{b\}$, then $\overline{L_1} \cup \overline{L_2} = \{b,c,d\} \cup \{a,c,d\} = \Sigma$, the entire input alphabet! If you need to do this with two characters, you have to put them all in manually, I'm afraid..

- As the ! symbol is now reserved for "anything but", you can no longer use it to mean λ or "blank", as you could with regular expressions! The symbol that JFLAP uses for blank symbol is $\square$. Never copy-and-paste these symbols from PDF documents like this one, as you will get a wrong encoding and symbol in JFLAP.

- There is a way to save yourself from clicking "`Step`" a million times when testing your machine if you only want to see the final tape output. Go to "`Input/Multiple Run`" (or `Multiple Run (Transducer)`, it does not matter which) and put your inputs in.Click "`Run Inputs`" and once it has finished running them, select the input you want to check the output for and click "`View Trace`". That should give you a pop-up window with the state that it ended up in and the tape output when the machine halted.

- When combining your machines, with both of the files open, go to `Convert/Combine Automata` and then select the other file. JFLAP will put both of your machines in a new window. You will still have to make some modifications to make them play nicely together, but it will save you having to re-create everything from scratch!

- There are a number of other shortcuts that JFLAP uses, although they are reasonably fiddly and might not work under every circumstance. To avoid any problems with your machine failing to operate when being run with our automarker, we advise not to use any shortcuts except for the ! shortcut. Said so, your machine is being tested solely on the output that it produces, so you can use whatever methods you like — just be warned that you are doing so at your own risk if you try and be too fancy.