The Australian National University
School of Computing
Pascal Bercher

Semester 1, 2023
Assignment 2 of 3

Theory of Computation

Release Date: Friday, 31 March 2023
Concrete version: Friday 31st March, 2023, 23:15 (Compilation date and time)
Due Date: Tuesday, 25 April 2023, 23:55 Canberra time. *Late submissions will not be accepted.*
Maximum credit: 100 (worth 12% of final course mark)

Pledge of Academic Integrity and Originality statement

I am committed to being a person of integrity. I pledge, as a member of the Australian National University community, to abide by and uphold the standards of academic integrity outlined in the ANU statement on honesty and plagiarism. I understand that it is wrong to ever misrepresent another person's work as my own.

I am aware of the relevant legislation, and understand the consequences of me breaching those rules. I am aware that all assigments have to be done by me alone. **By submitting this work, I declare that everything I have submitted in this assignment was entirely my own work.**

Exercise 1 **The Pumping Lemmas** (22 credits)

For the purposes of this question, we define, for a language L, the following propositions

- We say that $P_R(L)$ if there exists an $n \in \mathbb{N}, n > 0$ such that for any string $z \in L$ with $|z| \geq n$, there exist strings u, v, w such that
  1. $z = uvw$,
  2. $|uv| \leq n$,
  3. $|v| > 0$, and
  4. $uv^iw \in L$ for $i \in \mathbb{N} \cup \{0\}$.
- We say that $P_{CF}(L)$ if there exists an $n \in \mathbb{N}, n > 0$ such that for any string $z \in L$ with $|z| \geq n$, there exist strings u, v, w, x, y such that
  1. $z = uvwxy$,
  2. $|vwx| \leq n$,
  3. $|vx| > 0$, and
  4. $uv^iwx^iy \in L$ for $i \in \mathbb{N} \cup \{0\}$.

Recall the following facts which have been proven in lectures:

- The pumping lemma for regular languages, which states that if a language L is regular then $P_R(L)$.
- The pumping lemma for context-free languages, which states that if a language L is context free then $P_{CF}(L)$.
- All regular languages are context free.
- Context-free languages are not always regular (e.g. $\{0^n1^n : n \geq 0\}$).

These four points are summarised in the following diagram, where the arrows represent implications. Crossed out arrows represent that there is provably not an implication.

$$\begin{tikzcd}
\boxed{P_R(L)} & \boxed{L \text{ is regular}} \arrow[l] \arrow[d, bend right] \\
\boxed{P_{CF}(L)} & \boxed{L \text{ is context free}} \arrow[l] \arrow[u, bend right, "/" marking]
\end{tikzcd}$$

For each of the seven dashed arrows in the following diagram, decide whether or not it is an implication. If it is an implication, provide convincing reasoning as to why (this should only be a few sentences). If it is not an implication, provide a counterexample, and explain why it is a counterexample (this should take no more than half a page). Your full answer to this question should be 1–2 pages.

$$\begin{tikzcd}
\boxed{P_R(L)} \arrow[r, dashed, bend right, "3"'] \arrow[d, dashed, bend right, "1"'] \arrow[dr, dashed, "4"] & \boxed{L \text{ is regular}} \arrow[l, bend right] \arrow[d, bend right] \arrow[dl, dashed, bend right=80, "6"'] \\
\boxed{P_{CF}(L)} \arrow[u, dashed, bend right, "2"'] \arrow[ur, dashed, bend right=80, "7"'] & \boxed{L \text{ is context free}} \arrow[l] \arrow[u, bend right, "/" marking] \arrow[ul, dashed, bend left, "5"]
\end{tikzcd}$$

You may find the following hints helpful in your answer.

- You may refer to and rely on proofs conducted in the lectures.
- There are relationships between these implications. You do not need to prove all of them directly, and you do not need to prove them in the order given by the numbers. Can you rely on some of the implications to prove others? Please be very careful when concluding some answer, in particular when you use it/them to conclude another, as a single error could then cause others as well.
- For arrow 3, consider the language

$$\{da^n b^n : n \geq 0\} \cup \{d^k x : k \neq 1, x \in \{a, b\}^*\}.$$

  A similar idea should also help with arrow 4.

Note that there is a "missing arrow" going from (L is context free) to $P_{CF}(L)$. We do not require you to prove or disprove that one.

Exercise 2 **CYK Algorithm** (2+1+6+9 = 18 credits)

In order to verify whether a word lies in a language, we can solve this "(word) problem" by different means:

- If the language is given an automaton, we just execute the automaton and check whether we can reach an accepting state.
- If the language is given in terms of a formal grammar, we could either translate this grammar into an appropriate automaton (and then do the previous step), or we do the verification directly based on the grammar. This process is called *parsing*. There are multiple parsing approaches in the literature. In this exercise we focus on the CYK algorithm (by Cocke, Younger, Kasami).

In tis exercise, we focus on bringing a grammar into Chomsky Normal Form and then executing CYK.

1. Consider the following context-free grammar G_1:

$$S \to aSa \mid bSb \mid a \mid b$$

   Write down the language generated by that grammar.

2. State whether $L(G_1)$ is regular, context-free but not regular, or neither. No explanation or proof is required.

3. Turn G_1 into its corresponding grammar G_1' that is in Chomsky Normal Form by following the steps of the lecture. Do not just find *any* grammar in the required form. You have to use the taught procedure. Perform the steps in the order provided in the slides and provide appropriate headlines. Make sure to explicitly write down the resulting grammar with all resulting and remaining production rules.

4. Now, we would execute the CYK algorithm on the resulting grammar G_1' with any word(s) we'd like to test. To make marking easier (since some might produce a wrong grammar), we instead provide a grammar G_2 that is already in Chomsky Normal Form. These are the production rules:

$$\begin{aligned} S &\to AB \mid AC \\ A &\to a \\ B &\to b \\ C &\to SB \end{aligned}$$

   Check whether $aabb$, $aaabb$, and $aabbb$ are in $L(G_2)$ by running CYK. Provide all values that CYK computes and state for each word whether it's accepted or not.

Exercise 3 **Turing Machine** (25 credits)

Using 10 states or less, design a Turing Machine that, given a blank input, will never halt, and will produce the following infinite string:

$$1011011101111011111 0\ldots = 1^1 01^2 01^3 01^4 01^5 01^6 \ldots$$

More formally, let $w_k = \ldots B1^1 01^2 \ldots 01^k B \ldots$ (where B indicates a blank). Your TM is satisfying the requirements if at some point it contains the string w_1, at some point later w_2, at some point later w_3, and so on, for all $k \geq 1$. There are no restrictions on which other strings are written on the tape in the process of turning some string w_i into w_{i+1}.

To illustrate, we require that the tape contains, in order:

$$\begin{aligned} w_1 &= 1 && \text{(at some point } i_1\text{)} \\ w_2 &= 1011 && \text{(at some point } i_2 > i_1\text{)} \\ w_3 &= 10110111 && \text{(at some point } i_3 > i_2\text{)} \\ w_4 &= 1011011101111 && \text{(at some point } i_4 > i_3\text{)} \\ &\ldots \end{aligned}$$

- Since marking Turing Machines can be *very* complex and time-consuming, you **must** specify it using the online tool `turingmachine.io`. You also **must provide the code** so that we can execute and test it. Make sure that we can easily copy the code. Do not just provide a link to the code (that could still be edited), but do provide the actual code. Put comments into the code. *If copying the code out of your document is not possible you will not receive any marks.* (You can also upload a plain-text file.) The website contains example code.
- You are allowed to use any other/additional tape symbols.
- The behavior of the Turing machine on any other (non-empty) input does not matter.

Exercise 4 On (Un)Decidability and Recursive Enumerability (35 credits)

Classify each of the following languages as (1) *recursive* (decidable), (2) *recursively enumerable and not recursive* (undecidable), or (3) *not recursively enumerable* (non-RE) and provide a proof for your answer. (Don't forget to show that a problem is resursively enumerable if you show that a problem is not recursive.) It is up to you how you prove the respective answer, e.g., via a decision procedure, a reduction, or any other proof. By as detailed as required. (The level of abstraction provided in the lectures and tutorials is a good indication for what you need to score high.)

Assume that only left and right movements of the head are possible. You may also assume that the input alphabet is $\{0, 1\}$.

1. $\{\langle M\rangle : M$ is a TM which, for all inputs, halts (accepts) and moves the head less than 100 times$\}$
2. $\{\langle M\rangle : M$ is a TM for which there exists an input on which it halts (accepts) and moves the head less than 100 times$\}$
3. $\{\langle M\rangle : M$ is a TM which rejects every input that has a length greater than 3 and accepts all of length 3 or shorter$\}$
4. $\{\langle M\rangle : M$ is a TM which, for every input, either rejects it or modifies the tape at least 10 times$\}$
5. $\{\langle M\rangle : M$ is a TM for which exists an input of length at most 10, for which the TM halts (accepts) after 100 or more steps$\}$
6. $\{\langle M\rangle : M$ is a TM that accepts exactly all palindromes$\}$

Note that we do not tell in advance which exercise gives how many marks since they might not be equally hard and we do not want to give anything away! (Just answer all questions, even if you are not sure.)

Do not forget that by submitting anything, you pledge to our Academic Integrity standards – see cover page.