COMP 3803 — Assignment 4

Due: Thursday April 4, before 11:55pm.

Assignment Policy:

- Your assignment must be submitted as one single PDF file through cuLearn.
- Late assignments will not be accepted.
- You are encouraged to collaborate on assignments, but at the level of discussion only. When writing your solutions, you must do so in your own words.
- Past experience has shown conclusively that those who do not put adequate effort into the assignments do not learn the material and have a probability near 1 of doing poorly on the exams.
- When writing your solutions, you must follow the guidelines below.
  - You must justify your answers.
  - The answers should be concise, clear and neat.
  - When presenting proofs, every step should be justified.

Question 1: Construct a Turing machine with one tape that gets as input an integer $x \geq 0$ and returns as output the integer $x + 1$. Integers are represented in binary.

Start of the computation: The tape contains the binary representation of the input x. The tape head is on the **leftmost** bit of x and the Turing machine is in the start state.

End of the computation: The tape contains the binary representation of the number $x + 1$. The tape head is on the **leftmost** bit of $x + 1$ and the Turing machine is in the final state.

The Turing machine in this question does not have an accept state or a reject state; instead, it has a final state. As soon as this final state is entered, the Turing machine terminates. At termination, the contents of the tape is the output of the Turing machine.

Start by explaining your algorithm in plain English, then mention the states that you are going to use, then explain the meaning of these states, and finally give the list of instructions.

Question 2: Construct a Turing machine with three tapes that gets as input two integers $x \geq 0$ and $y \geq 0$, and returns as output the number $x+y$. Integers are represented in binary.

Start of the computation: Tape 1 contains the binary representation of x, its head is on the **rightmost** bit of x. Tape 2 contains the binary representation of y, its head is on the **rightmost** bit of y. Tape 3 is empty (that is, it contains only □'s), its head is at an arbitrary position. At the start, the Turing machine is in the start state.

End of the computation: Tapes 1 and 2 are empty, and tape 3 contains the binary representation of the number $x + y$. The head of tape 3 is on the **rightmost** bit of $x + y$. The Turing machine is in the final state.

The Turing machine in this question does not have an accept state or a reject state; instead, it has a final state. As soon as this final state is entered, the Turing machine terminates. At termination, the contents of the tape is the output of the Turing machine.

Start by explaining your algorithm in plain English, then mention the states that you are going to use, then explain the meaning of these states, and finally give the list of instructions.

Question 3: In class, we have seen that the language

$$Halt = \{\langle P, w\rangle : \ P \text{ is a Java program that terminates on the binary input string } w\}$$

is undecidable.

A Java program P is called a *Hello-World-program*, if the following is true: When given the empty string ϵ as input, P can do whatever it wants, as long as it outputs the string `Hello World` and terminates. (We do not care what P does when the input string is non-empty.)

Consider the language

$$HW = \{\langle P\rangle : \ P \text{ is a Hello-World-program}\}.$$

The questions below will lead you through a proof of the claim that the language HW is undecidable.

(3.1) Consider a fixed Java program P and a fixed binary string w.

We write a new Java program J_{Pw} which takes as input an arbitrary binary string x. On such an input x, the Java program J_{Pw} does the following:

```
Algorithm $J_{Pw}(x)$:
run $P$ on the input $w$;
print Hello World
```

- Argue that P terminates on input w if and only if $\langle J_{Pw}\rangle \in HW$.

(3.2) The goal is to prove that the language HW is undecidable. We will prove this by contradiction. Thus, we assume that H is a Java program that decides HW. Recall what this means:

- If P is a Hello-World-program, then H, when given $\langle P\rangle$ as input, will terminate in the accept state.
- If P is not a Hello-World-program, then H, when given $\langle P\rangle$ as input, will terminate in the reject state.

We write a new Java program H' which takes as input the binary encoding $\langle P, w\rangle$ of an arbitrary Java program P and an arbitrary binary string w. On such an input $\langle P, w\rangle$, the Java program H' does the following:

```
**Algorithm** $H'(\langle P, w\rangle)$:
construct the Java program $J_{Pw}$ described above;
run $H$ on the input $\langle J_{Pw}\rangle$;
if $H$ terminates in the accept state
then terminate in the accept state
else terminate in the reject state
endif
```

Argue that the following are true:

- For any input $\langle P, w\rangle$, H' terminates.
- If P terminates on input w, then H' (when given $\langle P, w\rangle$ as input), terminates in the accept state.
- If P does not terminate on input w, then H' (when given $\langle P, w\rangle$ as input), terminates in the reject state.

(3.3) Now finish the proof by arguing that the language HW is undecidable.

Question 4: Consider the two languages

$$\mathit{Empty} = \{\langle M\rangle : \ M \text{ is a Turing machine for which } L(M) = \emptyset\}$$

and

$$\begin{aligned}\mathit{UselessState} = \{\langle M, q\rangle: \quad & M \text{ is a Turing machine, } q \text{ is a state of } M,\\ & \text{for every input string } w, \text{ the computation of } M \text{ on}\\ & \text{input } w \text{ never visits state } q\}.\end{aligned}$$

(4.1) Use Rice's Theorem to show that *Empty* is undecidable.

(4.2) Use **(4.1)** to show that *UselessState* is undecidable.