COMP 3803 — Solutions Assignment 4

Question 1: Write your name and student number.

Solution: Pelé, 10

Question 2: Construct a Turing machine with one tape that shifts a non-empty string over the alphabet $\{a, b\}^*$ one cell to the right.

Start of the computation: The tape contains the string

$$*w_1 w_2 \dots w_n$$

where $n \geq 1$. The tape head is at the special symbol $*$ and the Turing machine is in the start state.

End of the computation: The tape contains the string

$$*\Box w_1 w_2 \dots w_n$$

The tape head is at the special symbol $*$ 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.

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.

Solution: The Turing machine does the following:

- Starting in the start state, the first symbol read is $*$. The TM stays in the start state and moves one cell to the right.
- The second symbol read is a or b, because $n \geq 1$. The TM writes $\Box$, moves one cell to the right, and remembers the symbol it just read.
- The TM walks to the right. At each step, it remembers the previously read symbol. The TM remembers the current symbol, writes the previous symbol, and moves one cell to the right.
- Once it reaches the end of the string, the TM walks left to the $*$.

We will use the following states:

q_s : start state; tape head is at $*$ or w_1
q_f : final state; tape head is at $*$
q_a : the symbol read in the previous step is a
q_b : the symbol read in the previous step is b
q_ℓ : the input has been shifted; tape head moves left

Here are the instructions:

$$\begin{array}{lcl} q_s* & \to & q_s * R \\ q_s a & \to & q_a \Box R \\ q_s b & \to & q_b \Box R \\ q_a a & \to & q_a a R \\ q_a b & \to & q_b a R \\ q_a \Box & \to & q_\ell a L \\ q_b a & \to & q_a b R \\ q_b b & \to & q_b b R \\ q_b \Box & \to & q_\ell b L \\ q_\ell a & \to & q_\ell a L \\ q_\ell b & \to & q_\ell b L \\ q_\ell \Box & \to & q_\ell \Box L \\ q_\ell * & \to & q_f * N \end{array}$$

Question 3: Construct a Turing machine with two tapes that "doubles" a non-empty string over the alphabet $\{a\}$.

Start of the computation: Tape 1 contains a string of the form a^n, for some $n \geq 1$, and its head is on the **leftmost** a. Tape 2 is empty (that is, it contains only $\Box$'s) and its head is at an arbitrary position. At the start, the Turing machine is in the start state.

End of the computation: Tape 1 contains the same string a^n and its head is at the **leftmost** a. Tape 2 contains the string a^{2n} and its head is at the **leftmost** a. At the end, 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.

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.

Solution: The Turing machine does the following:

- On tape 1, walk to the right. For each a on tape 1:
  - The TM is in the start state q_s. Write one a on tape 2, move right on both tapes, switch to state q'_s (this state means that we have to write one more a on tape 2).

- The TM is in the state q'_s. Write one a on tape 2, do not move on tape 1, move right on tape 2, switch back to the start state q_s.

- At the end, walk to the left on both tapes.

We will use the following states:

q_s : start state; walking right
q'_s : write one more a on tape 2
q_f : final state; both tape heads are at the leftmost a
q_ℓ : the input has been doubled; both tape head move left

Here are the instructions:

$$\begin{array}{lcl} q_s a\Box & \to & q'_s aaRR \\ q_s \Box\Box & \to & q_\ell \Box\Box LL \\ q'_s a\Box & \to & q_s aaNR \\ q'_s \Box\Box & \to & q_\ell \Box aLN \\ q_\ell aa & \to & q_\ell aaLL \\ q_\ell \Box a & \to & q_\ell \Box aNL \\ q_\ell \Box\Box & \to & q_f \Box\Box RR \end{array}$$

Question 4: Consider the language

$$\begin{array}{ll} \mathit{tztt} = \{\langle M\rangle: & M \text{ is a Turing machine,} \\ & \text{for every input string } w \in \{0,1\}^*, \text{ the computation of } M \text{ on} \\ & \text{input } w \text{ terminates within 2022 steps}\}. \end{array}$$

Professor Justin Bieber claims that *tztt* is undecidable.
Is Professor Bieber's claim correct? As always, justify your answer.

Solution: As always, Professor Bieber is wrong. We will show that *tztt* is decidable.

- By definition, $\langle M\rangle \in \mathit{tztt}$ if and only if M terminates within 2022 steps, on any input string w. Note that there are infinitely many such strings w.

- On an input string w, M starts at the leftmost bit. If $|w| \geq 2023$, then, within 2022 steps, M can only look at the leftmost 2022 bits of w.

- In other words: Let $w = w_1 w_2 \dots w_n$ be a bitstring of length $n \geq 2023$. Then M terminates within 2022 steps on input w if and only if M terminates within 2022 steps on input $w_1 w_2 \dots w_{2022}$.

Based on this, we obtain the following algorithm that decides whether or not $\langle M\rangle$ belongs to *tztt*:

1. For every bitstring w of length at most 2022:

   (a) Run M on input w for up to 2022 steps.

(b) If M did not terminate within 2022 steps: terminate and output the message "$\langle M \rangle \notin tztt$".

2. If the algorithm reaches here: terminate and output the message "$\langle M \rangle \in tztt$".

Note that this algorithm terminates, because there are "only"

$$1 + 2 + 2^2 + 2^3 + \cdots 2^{2022} = 2^{2023} - 1$$

bitstrings of length at most 2022. For each such string we run M for only 2022 steps.

Question 5: Consider the language

$$Weird = \{\langle M \rangle : \ M \text{ is a Turing machine that does not accept the string } \langle M \rangle\}.$$

Use a proof by contradiction to show that *Weird* is undecidable.

Solution: Assume that *Weird* is decidable. Then there exists a Turing machine A such that for any Turing machine M:

- If $\langle M \rangle \in Weird$: on input $\langle M \rangle$, A terminates and accepts.
- If $\langle M \rangle \notin Weird$: on input $\langle M \rangle$, A terminates and rejects.

Now we ask ourselves: Is $\langle A \rangle$ in *Weird*?

- Assume that $\langle A \rangle \in Weird$. By taking $M = A$, we know that A accepts $\langle A \rangle$. But then, by the definition of the language *Weird*, $\langle A \rangle$ is not in *Weird*. This is a contradiction.
- Assume that $\langle A \rangle \notin Weird$. By taking $M = A$, we know that A does not accept $\langle A \rangle$. But then, by the definition of the language *Weird*, $\langle A \rangle$ is in *Weird*. This is a contradiction.

Thus, in either case, we get a contradiction. Therefore, our assumption that *Weird* is decidable is wrong. We conclude that *Weird* is undecidable. This was weird eh!

Bonus Question: Consider the following algorithm:

Algorithm *Collatz*(n):
comment: $n \geq 1$ is an integer
if $n = 1$
then terminate
else if n is even
    then *Collatz*$(n/2)$
    else *Collatz*$(3n + 1)$
    endif
endif

A famous conjecture in mathematics (that nobody has been able to prove) is the following:

Collatz Conjecture: For every integer $n \geq 1$, algorithm $Collatz(n)$ terminates.

In class, we have seen the language

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

Prove the following claim: If the language *Halt* is decidable, then we can prove whether or not the Collatz Conjecture is true.

Solution: We assume that *Halt* is decidable. Then there exists an algorithm H such that for any Java program P and any bitstring w:

- If P terminates on input w: On input $\langle P, w\rangle$, H terminates and outputs YES.
- If P does not terminate on input w: On input $\langle P, w\rangle$, H terminates and outputs NO.

We will denote the output of algorithm H on input $\langle P, w\rangle$ by $H(\langle P, w\rangle)$. Thus,

$$H(\langle P, w\rangle) = \begin{cases} \text{YES} & \text{if } P \text{ terminates on input } w, \\ \text{NO} & \text{if } P \text{ does not terminate on input } w. \end{cases}$$

Consider the following algorithm C, which takes the empty string ε as input:

Algorithm C:
Input: empty string ε
$stop = false$;
$n = 1$;
while $stop = false$
do if $H(\langle Collatz, n\rangle) = \text{NO}$
    then $stop = true$
    endif;
    $n = n + 1$
endwhile

Observe that every iteration of the while-loop terminates, because H always terminates. Now we ask ourselves what it means that C terminates or does not terminate:

- Assume that C terminates. Then there is an integer n such that the value of the variable *stop* is set to *true* during the n-th iteration of the while-loop. This means that $H(\langle Collatz, n\rangle) = \text{NO}$, which means that algorithm *Collatz* does not terminate on input n, which means that the Collatz Conjecture is false.
- Assume that C does not terminate. Then the while-loop does not terminate, which means that for all integers $n \geq 1$, during the n-th iteration of the while-loop, we have $H(\langle Collatz, n\rangle) = \text{YES}$. This means that for all integers $n \geq 1$, algorithm *Collatz* terminates on input n, which means that the Collatz Conjecture is true.

- In other words:
  - If C terminates on input ε, then the Collatz Conjecture is false.
  - If C does not terminate on input ε, then the Collatz Conjecture is true.

Based on this, we run the following algorithm C', which takes the empty string ε as input:

Algorithm C':
Input: empty string ε
Run H on input $\langle C, \varepsilon \rangle$;
if H returns YES
then output "The Collatz Conjecture is false"
else output "The Collatz Conjecture is true"
endif