University of New Brunswick
Faculty of Computer Science
CS2333: Computability and Formal Languages
Homework Assignment 8, ***Due Time, Date*** *5:00 PM, April 1, 2022*

Student Name: ____________________________ Matriculation Number: ________________

Instructor: Rongxing Lu
The marking scheme is shown in the left margin and [100] constitutes full marks.

[**50**] 1. Show

$$A_{CFG} = \{(G, w) | G \text{ is a context-free grammar }, w \text{ is a string }, w \in L(G).\}$$

is a decidable language.

[**50**] 2. Show

$$A_{TM} = \{(T, w) | T \text{ is a TM }, w \text{ is a string }, T \text{ accepts } w.\}$$

is undecidable.

Solutions.

1. Show
$$A_{CFG} = \{(G, w)|G \text{ is a context-free grammar }, w \text{ is a string }, w \in L(G).\}$$
is a decidable language.

   In order to show A_{CFG} is a decidable language, we can construct a Turing machine (TM) algorithm as follows:

   TM Algorithm 3: On input (G, w)

   - Confirm that we have a valid encoding of the context-free grammar G and a string w. If not, reject.
   - Convert G to an equivalent G' in Chomsky normal form (by using the steps of construction method. We have discussed this part in chapter 3.)
   - Let n be the length of the string w. Then, if $w \in L(G) = L(G')$, any derivation of $w \in G'$, from the start variables of G', consists of exactly $2n - 1$ steps (where a step is defined as applying one rule of G').
   - Hence, we can decide whether or note $w \in L(G)$, by trying all possible derivations, in G', consisting of $2n - 1$ steps. If one of these (finite number of) derivations leads to the string w, then $w \in L(G)$. Otherwise $w \notin L(G)$.

   Therefore, A_{CFG} is a decidable language.

2. Show
$$A_{TM} = \{(T, w)|T \text{ is a TM }, w \text{ is a string }, T \text{ accepts } w.\}$$
is undecidable.

   Proof by Contradiction.

   Suppose that A_{TM} is decidable. (We will find a contradiction.)

   Then, there exists some TM H that can decide A_{TM}.

   TM Algorithm 4 (H): On input (T, w), where T is a TM and w is a string.

   - H accepts (T, w) if T accepts w.
   - H rejects (T, w) if T <u>does not accept</u> w.

   Now, we can construct a new TM D that works as follows:

   TM Algorithm 5 (D): On input any TM S

   - Call TM Algorithm 4 (H) with the following input
     - TM S
     - the encoding of S as a string
   - That is, on input (S), where S is TM
     - Run H on input $(S, (S))$
     - If H accept, then reject

  - If H reject, then accept

Summarize the algorithm, what does D do on input (S)

- D rejects (S), if H accepts $(S, (S))$, which implies

$$D \text{ rejects } (S), \text{ if } S \text{ accepts } (S)$$

- D accepts (S), if H rejects $(S, (S))$

$$D \text{ rejects } (S), \text{ if } S \text{ does not accept } (S)$$

Final step:

- D is itself a TM
- what happen if we run D with input (D)? That is, if S is D Then,

$$D \text{ rejects } (D), \text{ if } D \text{ accepts } (D)$$

$$D \text{ rejects } (D), \text{ if } S \text{ does not accept } (D)$$

Therefore, there is a contradiction. That is, the original assumption we made that "some TM exists that will decide A_{TM}" must be false. Therefore, A_{TM} is undecidable. There is no algorithm that can take any T and any w and always answer yes/no question of whether T accepts w.