ComS 331 **Spring 2025** **Name:** Mobina Amrollahi

HW 12 Due: April 25^{th} 2025

1. Are the following languages Turing-decidable, Turing-acceptable but not Turing-decidable, or not even Turing-acceptable?

   - $L = \{\rho(M)\rho(w) : M$ uses a finite number of tape cells when running on input $w\}$.
   - $L = \{\rho(M)\rho(w)01^n0 : M$ uses at most n tape cells when running on input $w\}$.

   Here, "using n cells" means that the head of the (deterministic) TM M reaches the n-th cell from the left during its computation. Justify your answers clearly; both exercises require careful thinking. Note that this exercise is in a sense relevant to "real computing", since one could argue that the computers we use in practice have a large but finite memory.

   We'll call the first language L_1 and the second one L_2 and proceed further.
   Answer for L_1

   We know that K_0, the problem of determining whether a Turing machine accepts a given input w, is undecidable:

   $$K_0 = \{\rho(M)\rho(w) : M \searrow w\}.$$

   We will prove by contradiction that if L_1 is decidable, then K_0 would also be decidable.

   Proof by Contradiction. Assume there exists a Turing machine R that decides L_1. We will construct a Turing machine S that decides K_0 as follows:

   $S =$ "On input $\rho(M)\rho(w)$, where M is a Turing machine and w is a string:

   (a) Run R on input $\rho(M)\rho(w)$.
      - If R **rejects**, then **reject**.
      - If R **accepts**:
        - Simulate M on input w.
        - If M repeats a configuration without halting, then **reject**.
        - If M halts before repeating any configuration, then **accept**."

   If R rejects, this indicates that M uses an infinite number of tape cells when running on input w. Therefore, M loops forever, and S correctly rejects. If R accepts, then M uses only a finite number of tape cells. Since there are finitely many distinct configurations, if M does not halt, it must eventually repeat a configuration and enter an infinite loop. Thus, S will reject upon detecting this repetition. If M halts before any configuration repeats, S will accept.

   Thus, S decides K_0 using R, implying that K_0 is decidable. This contradicts the known undecidability of K_0. Therefore, L_1 is not decidable.
   Now, we will show that L_1 is Turing-acceptable. Notice that L_1 can be expressed as the union of the following two languages:

   $$L_1 = L' \cup L''$$

where

$$L' = \{\rho(M)\rho(w) : M \text{ halts on input } w\},$$
$$L'' = \{\rho(M)\rho(w) : M \text{ uses a finite number of tape cells and loops on input } w\}.$$

It is known that L' is Turing-acceptable, since we can simulate M on input w and accept if M halts.

Similarly, L'' is also Turing-acceptable because if M uses only a finite number of tape cells, then the number of possible configurations is finite. By simulating M, we can eventually detect if a configuration repeats. Once a repeated configuration is observed without M halting, we can conclude that M is looping and accept. Since both L' and L'' are Turing-acceptable, and the union of two Turing-acceptable languages is also Turing-acceptable, it follows that L_1 is Turing-acceptable.

Answer for L_2

We will show that L_2 is decidable.

Note that a Turing machine using at most n tape cells can have at most qng^n distinct configurations, where:

- q is the number of states in M,
- The head can be in one of n positions,
- There are g^n possible strings of tape symbols (where g is the size of the tape alphabet).

We can use this bound to decide L_2 by constructing the following Turing machine S:

$S =$ On input $\rho(M)\rho(w)$, where M is a Turing machine and w is a string:

(a) Simulate M on input w for at most qng^n configurations.
  - If M uses more than n tape cells during computation, then **reject**.
  - If M repeats a configuration without halting, then **reject**.
  - If M halts before repeating any configuration, then **accept**.

Since S only needs to simulate M for a finite number of steps bounded by qng^n, this procedure always halts. Therefore, L_2 is **decidable**.

2. Define an unrestricted grammar for the language $\{ww : w \in \{0,1\}^*\}$.
Explain clearly how it works or, even better, formally prove that it works.
Answer

Consider the grammar $G = (\{S, T, E, P, C\}, \{0, 1\}, S, P)$ where the set of rules, P, is

$$
\begin{aligned}
&S \to TE \\
&T \to 0T0 \mid 1T1 \mid C \\
&C \to CP \\
&P00 \to 0P0 \\
&P01 \to 1P0 \\
&P10 \to 0P1 \\
&P11 \to 1P1 \\
&P0E \to E0 \\
&P1E \to E1 \\
&0E \to E0 \\
&1E \to E1 \\
&CE \to \epsilon.
\end{aligned}
$$

We will prove by induction on length of w why such grammar works.

(a) Base Case: $\mid w \mid = 1$. Suppose $w = a$ where $a \in \{0, 1\}$. Then the following derivations are going to produce $ww = aa$:

$$S \to TE \to aTaE \to aCaE \to aCPaE \to aCEa \to aa$$

(b) **Induction Hypothesis:** Assume that for $|w| = n$, the proposed grammar can generate ww.

(c) **Induction Step:** Consider $w = a_1 a_2 \cdots a_n a_{n+1}$ where $|w| = n + 1$. We will show that the following derivation reaches:

$$a_1 a_2 \cdots a_n a_{n+1} C a_n a_{n-1} \cdots a_2 a_1 E a_{n+1}$$

Since by the induction hypothesis $C a_n a_{n-1} \cdots a_2 a_1 E$ derives $a_1 a_2 \cdots a_n$, this will complete the proof.

The derivation proceeds as follows:

$$
\begin{aligned}
S \to & TE \to a_1 T a_1 E \to a_1 a_2 T a_2 a_1 E \to \cdots \to a_1 a_2 \cdots a_n a_{n+1} T a_{n+1} a_n \cdots a_2 a_1 E \\
&\to a_1 a_2 \cdots a_n a_{n+1} C a_{n+1} a_n \cdots a_2 a_1 E \\
&\to a_1 a_2 \cdots a_n a_{n+1} C P a_{n+1} a_n \cdots a_2 a_1 E \\
&\to a_1 a_2 \cdots a_n a_{n+1} C a_n P a_{n+1} a_{n-1} \cdots a_2 a_1 E \\
&\to a_1 a_2 \cdots a_n a_{n+1} C a_n a_{n-1} P a_{n+1} a_{n-2} \cdots a_2 a_1 E \\
&\to \cdots \\
&\to a_1 a_2 \cdots a_n a_{n+1} C a_n a_{n-1} \cdots a_2 P a_{n+1} a_1 E \\
&\to a_1 a_2 \cdots a_n a_{n+1} C a_n a_{n-1} \cdots a_2 a_1 P a_{n+1} E \\
&\to a_1 a_2 \cdots a_n a_{n+1} C a_n a_{n-1} \cdots a_2 a_1 E a_{n+1}
\end{aligned}
$$