Tutorial 2
Recursive and R.E. Sets, (Un)Decidability, Rice’s Theorem

Guidelines: Solve all problems in the class. Do not search for solutions online.

1. Show that the class of recursively enumerable sets is closed under union and intersection.

Solution: Let $A, B$ be r.e. sets and let $M_A, M_B$ be the respective Turing machines recognising them. Consider the TM $N$ that does the following on input $x$.

- Run $M_A$ on $x$ and $M_B$ on $x$ in a round-robin fashion.
- Accept if either $M_A$ or $M_B$ accepts.

Clearly if $x \in A \cup B$, either $M_A$ or $M_B$ accepts it causing $N$ to accept. $N$ is a valid TM and hence $A \cup B$ is r.e.

Next, let $K$ be a TM that, on input $x$, does the following.

- Run $M_A$ on $x$.
- If $M_A$ accepts, then run $M_B$ on $x$.
- Accept if $M_B$ accepts.

$K$ accepts $x$ only if both $M_A$ and $M_B$ accept $x$ and hence $L(K) = A \cap B$. Note that $M_A$ could loop on $x$ in which case $x$ is clearly not in $A$ or $A \cap B$. So, it does not matter if we do not check whether or not $M_B$ accepts $x$.

2. Prove that a language $L$ is recursive if and only if there is an enumeration machine enumerating the strings of $L$ in lexicographic order.

Solution: Suppose that $L \subseteq \Sigma^*$ is a recursive language. Then there exists a total TM $K$ deciding $L$. Construct an enumeration machine $M$ that does the following for each string $y$ in $\Sigma^*$ according to the lexicographic order.

- Simulate $K$ on $y$.
- Enumerate $y$ if $K$ accepts $y$.
- If $K$ rejects $y$, then do not enumerate.

Since $K$ is total, the enumeration occurs after finitely many steps.

Conversely, suppose that there is an enumeration machine $M$ enumerating strings of a language $L$ in lexicographic order. Define a TM $K$ that does the following on input $y \in \Sigma^*$.

- Run $M$ until it enumerates $y$ or a string that comes after $y$ in a lexicographic order.
- If $y$ is enumerated, then accept and halt.
If the previous string enumerated is different from \( y \), then reject and halt.

Since \( \mathcal{M} \) enumerates all strings of \( L \) in lexicographic order, it should eventually enumerate \( y \) if \( y \in L \) or a string that follows \( y \) in the lexicographic order. The latter happens when \( y \notin L \). As a result, in finite time \( \mathcal{K} \) determines if \( y \in L \) or not. Therefore, \( \mathcal{K} \) is total and \( L \) is recursive.

3. One of the following sets is r.e. and the other is not. Determine which one is and which one is not. Justify your answers.

(a) \( \{ \mathcal{M} \mid L(\mathcal{M}) \text{ contains at least 300 elements} \} \)

**Solution:** This set is r.e. It is possible to construct a TM that runs a given \( \mathcal{M} \) all strings in a round-robin fashion. If at least 300 strings are accepted by \( \mathcal{M} \), then halt and accept. Otherwise keep testing for other strings.

(b) \( \{ \mathcal{M} \mid L(\mathcal{M}) \text{ contains at most 300 elements} \} \)

**Solution:** This is a non-monotone property of r.e. sets. If an r.e. set \( A \) contains at most 300 elements, then it is not necessary that all its supersets contain at most 300 elements. Hence, it follows by Rice’s theorem that the given set is not r.e.

4. True or False? It is decidable whether two given TMs accept the same set.

**Solution:** The problem is to show that the language \( \text{EQUIV} = \{ \mathcal{M}_1 \# \mathcal{M}_2 | L(\mathcal{M}_1) = L(\mathcal{M}_2) \} \) is undecidable. We show a reduction from the halting problem i.e., \( \text{HP} \leq_m \text{EQUIV} \). Given an instance \( \mathcal{M}_1 \# x \) of \( \text{HP} \), construct an instance \( \mathcal{M}_1 \# \mathcal{M}_2 \) such that \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) have the following descriptions.

\( \mathcal{M}_1 \): on input \( y \) does the following
- Erase the input \( y \).
- Accept and halt.

\( \mathcal{M}_2 \): on input \( y \) does the following
- Erase the input \( y \).
- Run \( \mathcal{M} \) on \( x \).
- Accept and halt if \( \mathcal{M} \) halts on \( x \).

Clearly, \( L(\mathcal{M}_1) = \Sigma^* \) and
\[
L(\mathcal{M}_2) = \begin{cases} \Sigma^* & \text{if } \mathcal{M} \text{ halts on } x \\ \emptyset & \text{otherwise} \end{cases}
\]

That is \( \mathcal{M}_1 \# x \in \text{HP} \) iff \( \mathcal{M}_1 \# \mathcal{M}_2 \in \text{EQUIV} \). The descriptions of \( \mathcal{M}_1, \mathcal{M}_2 \) can be written down by a total TM. (Note that writing down descriptions of these TMs does not require running \( \mathcal{M} \).) Since \( \text{HP} \) is undecidable, so is \( \text{EQUIV} \).

5. Show that none of the following languages or their complements are r.e.

(a) \( \text{REG} = \{ \mathcal{M} \mid L(\mathcal{M}) \text{ is a regular set} \} \).
Solution: We use Rice’s theorem to show that REG is undecidable. Let \( P_{\text{REG}} \) denote the property on r.e. sets defined as

\[
P_{\text{REG}}(A) = \begin{cases} 
T & \text{if } A \text{ is regular} \\
F & \text{otherwise}
\end{cases}
\]

Then deciding this property is equivalent to deciding \( \text{REG} \). If a set \( A \) is regular it is not necessary that all its supersets are regular. In other words, there exist \( A, B \) such that \( A \subseteq B \) and \( P_{\text{REG}}(A) = T \), \( P_{\text{REG}}(B) = F \). For instance, we can take \( A = \phi \) and \( B = \{0^n1^n \mid n \geq 0\} \). This shows that \( P_{\text{REG}} \) is a non-monotone property. By Rice’s theorem, \( P_{\text{REG}} \) is not r.e. or equivalently \( \text{REG} \) is not r.e.

Similarly, we can show that \( \neg \text{REG} \) or equivalently, \( P_{\neg \text{REG}}(A) = \begin{cases} 
T & \text{if } A \text{ is not regular} \\
F & \text{otherwise}
\end{cases} \)

is not r.e. by proving that \( P_{\neg \text{REG}} \) is a non-monotone property. We only need to exhibit two sets \( A, B \) with \( A \subseteq B \) such that \( P_{\neg \text{REG}}(A) = T \) and \( P_{\neg \text{REG}}(B) = F \). Taking \( A = \{0^n1^n \mid n \geq 0\} \) and \( B = \{0^*1^*\} \) suffices.

(b) \( \text{TOT} = \{M \mid M \text{ halts on all inputs}\} \).

Solution: We show \( \text{TOT} \) is not r.e. via a reduction from \( \neg \text{HP} \). Given an instance \( M \# x \) of \( \neg \text{HP} \), we construct a TM \( N \) such that \( M \# x \in \neg \text{HP} \) iff \( N \in \text{TOT} \). Below is a description of \( N \).

\( N \): on input \( y \) does the following.

- Store \( y \) on a separate track of the tape.
- Run \( M \) on \( x \) for \( |y| \) steps. (\( M \) and \( x \) are hardwired in \( N \)’s finite control.)
- If \( M \) does not halt within \( |y| \) steps, then halt and either accept or reject.
- Otherwise, enter a trivial loop.

If \( M \) does not halt on \( x \), then \( N \) halts on all input strings. Suppose that \( M \) halts on \( x \) in \( n \) steps. Then for any string \( y \) with \( |y| < n \), \( N \) accepts \( y \) since \( M \) does not halt on \( x \) within \( |y| \) steps. On the other hand, for any string \( y \) with \( |y| \geq n \), \( N \) does not halt. That is,

\[
\begin{align*}
& M \text{ does not halt on } x \implies N \text{ halts on all input strings} \\
& M \text{ halts on } x \implies N \text{ does not halt on some input strings}
\end{align*}
\]

Since \( \neg \text{HP} \) is not r.e., \( \text{TOT} \) is not r.e.

Next, we prove that \( \neg \text{TOT} \) is not r.e. through a reduction from \( \neg \text{HP} \). Given an instance \( M \# x \) of \( \neg \text{HP} \), we construct a TM \( N \) such that \( M \# x \in \neg \text{HP} \) iff \( N \in \neg \text{TOT} \). Below is a description of \( N \).

\( N \): on input \( y \) does the following.

- Store \( y \) on a separate track of the tape.
- If \( |y| \leq |x| \), halt and accept.
- Run \( M \) on \( x \).
- If \( M \) halts on \( y \), then halt and accept.
If $M$ does not halt on $x$, then $N$ halts on all input strings of length at most $|x|$. If $M$ halts on $x$, then $N$ accepts any string $y$. That is,

$M$ does not halt on $x \implies N$ does not halt on some inputs $\implies N \in \neg \text{TOT}$

$M$ halts on $x \implies N$ halts on all input strings $\implies N \notin \neg \text{TOT}$

Since $\neg \text{HP}$ is not r.e., $\neg \text{TOT}$ is not r.e.