Tutorial 1

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 $\mathcal{M}_A, \mathcal{M}_B$ be the respective Turing machines recognising them. Consider the TM $\mathcal{N}$ that does the following on input x.

   - Run $\mathcal{M}_A$ on x and $\mathcal{M}_B$ on x in a round-robin fashion.
   - Accept if either $\mathcal{M}_A$ or $\mathcal{M}_B$ accepts.

   Clearly if $x \in A \cup B$, either $\mathcal{M}_A$ or $\mathcal{M}_B$ accepts it causing $\mathcal{N}$ to accept. $\mathcal{N}$ is a valid TM and hence $A \cup B$ is *r.e.*

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

   - Run $\mathcal{M}_A$ on x.
   - If $\mathcal{M}_A$ accepts, then run $\mathcal{M}_B$ on x.
   - Accept if $\mathcal{M}_B$ accepts.

   $\mathcal{K}$ accepts x only if both $\mathcal{M}_A$ and $\mathcal{M}_B$ accept x and hence $L(\mathcal{K}) = A \cap B$. Note that $\mathcal{M}_A$ 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 $\mathcal{M}_B$ 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 $\mathcal{K}$ deciding L. Construct an enumeration machine $\mathcal{M}$ that does the following for each string y in Σ^* according to the lexicographic order

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

   Since $\mathcal{K}$ is total, the enumeration occurs after finitely many steps.

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

   - Run $\mathcal{M}$ 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) $\{\mathcal{M} \mid \mathcal{L}(\mathcal{M})$ contains atleast 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 atleast 300 strings are accepted by $\mathcal{M}$, then halt and accept. Otherwise keep testing for other stings.

   (b) $\{\mathcal{M} \mid \mathcal{L}(\mathcal{M})$ contains atmost 300 elements$\}$

   Solution: This is a non-monotone property of *r.e.* sets. If an *r.e.* set A contains atmost 300 elements, then it is not necessary that all its supersets contain atmost 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 $\mathsf{EQUIV} = \{(\mathcal{M}_1, \mathcal{M}_2) | \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)\}$ is undecidable. We show a reduction from the halting problem i.e., $\mathsf{HP} \leq_m \mathsf{EQUIV}$. Given an instance $(\mathcal{M}, x)$ of 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, $\mathcal{L}(\mathcal{M}_1) = \Sigma^*$ and

$$\mathcal{L}(\mathcal{M}_2) = \begin{cases} \Sigma^* & \text{if } \mathcal{M} \text{ halts on } x \\ \emptyset & \text{otherwise} \end{cases}$$

   That is $(\mathcal{M}, x) \in \mathsf{HP}$ iff $(\mathcal{M}_1, \mathcal{M}_2) \in \mathsf{EQUIV}$. 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 HP is undecidable, so is EQUIV.

5. Show that none of the following languages or their complements are *r.e.*
   (a) $\mathsf{REG} = \{\mathcal{M} \mid \mathcal{L}(\mathcal{M})$ is a regular set$\}$.

Solution: We use Rice's theorem to show that REG is undecidable. Let $\mathscr{P}_{\mathsf{REG}}$ denote the property on *r.e.* sets defined as

$$\mathscr{P}_{\mathsf{REG}}(A) = \begin{cases} \mathsf{T} & \text{if } A \text{ is regular} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Then deciding this property is equivalent to deciding 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 $\mathscr{P}_{\mathsf{REG}}(A) = \mathsf{T}$, $\mathscr{P}_{\mathsf{REG}}(B) = \mathsf{F}$. For instance, we can take $A = \phi$ and $B = \{0^n 1^n \mid n \geq 0\}$. This shows that $\mathscr{P}_{\mathsf{REG}}$ is a non-monotone property. By Rice's theorem, $\mathscr{P}_{\mathsf{REG}}$ is not *r.e.* or equivalently REG is not *r.e.* Similarly, we can show that $\neg$REG or equivalently,

$$\mathscr{P}_{\neg\mathsf{REG}}(A) = \begin{cases} \mathsf{T} & \text{if } A \text{ is not regular} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

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

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

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

$\mathcal{N}$: on input y does the following.

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

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

$$\mathcal{M} \text{ does not halt on } x \implies \mathcal{N} \text{ halts on all input strings} \implies \mathcal{N} \in \mathsf{TOT}$$

$$\mathcal{M} \text{ halts on } x \implies \mathcal{N} \text{ does not halt on some input strings} \implies \mathcal{N} \notin \mathsf{TOT}$$

Since $\neg$HP is not *r.e.* , TOT is not *r.e.*

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

$\mathcal{N}$: on input y does the following.

- Store y on a separate track of the tape.
- If $|y| \leq |x|$, halt and accept.
- Run $\mathcal{M}$ on x.
- If $\mathcal{M}$ halts on y, then halt and accept.

If $\mathcal{M}$ does not halt on x, then $\mathcal{N}$ halts on all input strings of length at most $|x|$. If $\mathcal{M}$ halts on x, then $\mathcal{N}$ accepts any string y. That is,

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

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

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

6. Let

$$f(x) = \begin{cases} 3x+1 & \text{if } x \text{ is odd} \\ x/2 & \text{if } x \text{ is even} \end{cases}$$

for any natural number x. Define $\mathsf{C}(x)$ as the sequence $x, f(x), f(f(x)), \ldots$, which terminates if and when it hits 1. For example, if $x = 7$, then

$$\mathsf{C}(x) = (7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1).$$

Computer tests have shown that $\mathsf{C}(x)$ hits 1 eventually for x ranging from 1 to 87×2^{60} (as of 2017). But, the question of whether $\mathsf{C}(x)$ ends at 1 for all $x \in \mathbb{N}$ is not proven. This is believed to be true and known as the Collatz conjecture. Suppose that MP were decidable by a Turing machine $\mathcal{K}$. Use $\mathcal{K}$ to describe a TM that is guaranteed to prove or disprove Collatz conjecture.

Solution: Let $\mathcal{N}$ be a TM, that on input x, does the following.

- while $x \neq 1$,
  - if x is even, $x \leftarrow x/2$
  - otherwise $x \leftarrow 3x + 1$
- accept and halt

Let $\mathcal{M}$ be a TM, that on input y, does the following.

- erase y
- set $x \leftarrow 1$ and repeat the following
  - use $\mathcal{K}$ to determine whether $\mathcal{N}$ accepts x
  - if not, accept and halt
  - otherwise, $x \leftarrow x + 1$

If the Collatz conjecture is true, then $\mathcal{M}$ runs forever; otherwise $\mathcal{M}$ halts after finding a counter example (in fact, the smallest counter example).

We now use $\mathcal{K}$ to determine whether or not $\mathcal{M}$ accepts an arbitrary input y to decide whether or not Collatz conjecture is true.

7. (a) Show that the language

$$\{(\mathcal{M}, \mathcal{N}) \mid \mathcal{M}, \mathcal{N} \text{ are Turing machines and } L(\mathcal{M}) \cap L(\mathcal{N}) = \emptyset\}$$

is undecidable via reduction.

Solution: Let $\mathsf{EI\text{-}TM}$ the above langauge. We show that $\neg\mathsf{HP} \leq_m \mathsf{EI\text{-}TM}$. Let $(\mathcal{K}, x)$ be an instance of $\neg\mathsf{HP}$. Construct $\mathcal{M}$ so that it accepts any input. Clearly $L(\mathcal{M}) = \Sigma^*$. Let $\mathcal{N}$ be defined so that, on input y it does the following: store y on a separate track; simulate $\mathcal{K}$ on x. If it halts, then accept y. If $(\mathcal{K}, x) \in \neg\mathsf{HP}$, then $L(\mathcal{N}) = \emptyset$ and $L(\mathcal{M}) \cap L(\mathcal{N}) = \emptyset$. If $(\mathcal{K}, x) \notin \neg\mathsf{HP}$, then $L(\mathcal{N}) = \Sigma^*$ and as a result $\mathcal{M} \cap \mathcal{N} \neq \emptyset$. Since $\neg\mathsf{HP}$ is undecidable, so is $\mathsf{EI\text{-}TM}$.

(b) Prove the following extension of Rice's theorem (of which part (a) is a special case):

Every non-trivial property of pairs of r.e. sets is undecidable.

More formally, let $\mathscr{P} : \{r.e.\text{ sets}\} \times \{r.e.\text{ sets}\} \to \{\top, \bot\}$ be a non-trivial property on pairs of *r.e.* sets. Then show that

$$T_{\mathscr{P}} = \{(\mathcal{M}, \mathcal{N}) \mid \mathcal{M} \text{ and } \mathcal{N} \text{ are TMs and } \mathscr{P}(L(\mathcal{M}), L(\mathcal{N})) = \top\}$$

is undecidable.

Solution: Let $\mathscr{P}$ be a non-trivial property of pairs of *r.e.* sets. Assume w.l.g. that $\mathscr{P}(\emptyset, \emptyset) = \bot$. Since $\mathscr{P}$ is non-trivial, there exist *r.e.* sets L_1, L_2 such that $\mathscr{P}(L_1, L_2) = \top$. Let $\mathcal{M}_1$, $\mathcal{M}_2$ be TMs recognising L_1, L_2 repectively. We show that $\mathsf{HP} \leq_m T_{\mathscr{P}}$. (If $\mathscr{P}(\emptyset, \emptyset) = \top$, then we can show that $\neg\mathsf{HP} \leq_m T_{\mathscr{P}}$ via a similar argument.) Let $\mathcal{N}, x$ be an instance of HP. Construct $\mathcal{M}_1'$ that on input w does the following:

- Run $\mathcal{N}$ on x ($\mathcal{N}, x$ are hardwired in the finite control of $\mathcal{M}_1'$).
- If $\mathcal{N}$ halts, simulate $\mathcal{M}_1$ on w.
- Accept if $\mathcal{M}_1$ accepts.

Similarly, construct $\mathcal{M}_2'$ that on input w does the following:

- Simulate $\mathcal{N}$ on x
- If $\mathcal{N}$ halts, simulate $\mathcal{M}_2$ on w.
- Accept if $\mathcal{M}_2$ accepts.

If $\mathcal{N}$ halts on x, then $L(\mathcal{M}_1') = L(\mathcal{M}_1)$ and $L(\mathcal{M}_2') = L(\mathcal{M}_2)$. As a result, $\mathscr{P}(L(\mathcal{M}_1'), L(\mathcal{M}_2')) = \top$. If $\mathcal{N}$ does not halt on x, then $L(\mathcal{M}_1') = L(\mathcal{M}_2') = \emptyset$ whence $\mathscr{P}(L(\mathcal{M}_1'), L(\mathcal{M}_2')) = \bot$. In other words, $(\mathcal{N}, x) \in \mathsf{HP} \Leftrightarrow (\mathcal{M}_1', \mathcal{M}_2') \in T_{\mathscr{P}}$. Therefore, $T_{\mathscr{P}}$ is undecidable.