Computability and Computational Complexity, A.Y. 2023–2024

Guide to the answers

Monday, February 19, 2024

Exercise 1

Let Σ be a finite alphabet, and $L_{\text{R1}}, L_{\text{R2}}, L_{\text{RE1}}, L_{\text{RE2}} \subseteq \Sigma^*$ be four languages on Σ. L_{R1} and L_{R2} are recursive, while L_{RE1} and L_{RE2} are recursively enumerable, but not recursive.

1.1) For each of the following languages, state if they are recursive, recursively enumerable, or none, and motivate your answers:

- $L_1 = L_{\text{R1}} \cup L_{\text{R2}}$;
- $L_3 = L_{\text{R1}} \cup L_{\text{RE1}}$;
- $L_5 = L_{\text{RE1}} \cup L_{\text{RE2}}$;
- $L_2 = L_{\text{R1}} \cap L_{\text{R2}}$;
- $L_4 = L_{\text{R1}} \cap L_{\text{RE1}}$;
- $L_6 = L_{\text{RE1}} \cap L_{\text{RE2}}$.

1.2) State whether the following properties of Turing machines are computable or not, and motivate your statements:

- $\mathcal{P}_1 = \{\mathcal{M} : \mathcal{M} \text{ decides } L_{\text{R1}}\}$;
- $\mathcal{P}_2 = \{\mathcal{M} : \mathcal{M} \text{ decides } L_{\text{RE1}}\}$;
- $\mathcal{P}_3 = \{\mathcal{M} : \text{ if } |x| < 100, \text{ then } \mathcal{M} \text{ decides } x \in L_{\text{R1}} \text{ in no more than } |x|^2 + 1 \text{ steps}\}$;
- $\mathcal{P}_4 = \{\mathcal{M} : \text{ if } |x| < 100, \text{ then } \mathcal{M} \text{ decides } x \in L_{\text{RE1}} \text{ in no more than } |x|^2 + 1 \text{ steps}\}$.

Solution 1

By definition, we can assume that there are TMs $\mathcal{M}_{\text{R1}}$ and $\mathcal{M}_{\text{R2}}$ that respectively decide language L_{R1} and L_{R2}; likewise, let TMs $\mathcal{M}_{\text{RE1}}$ and $\mathcal{M}_{\text{RE2}}$ recognize languages L_{RE1} and L_{RE2} respectively.

1.1)

- we can define TM $\mathcal{M}_1$ that decides $x \in L_1$ by checking if $x \in L_{\text{R1}}$ or $x \in L_{\text{R2}}$:

  $$\mathcal{M}_1(x) : \text{if } \mathcal{M}_{\text{R1}}(x) \text{ accepts then accept, else run } \mathcal{M}_{\text{R2}}(x),$$

  therefore L_1 is recursive.

- Similarly, we can define TM $\mathcal{M}_2$ that decides $x \in L_2$ by checking if $x \in L_{\text{R1}}$ **and** $x \in L_{\text{R2}}$:

  $$\mathcal{M}_2(x) : \text{if } \mathcal{M}_{\text{R1}}(x) \text{ rejects then reject, else run } \mathcal{M}_{\text{R2}}(x),$$

  thus L_2 is recursive too.

- For L_3, if $x \notin L_{\text{R1}}$, then we must check if $x \in L_{\text{RE1}}$, but in general $\mathcal{M}_{\text{RE1}}(x)$ only halts if the answer is positive: therefore we can only create a machine that recognizes L_3, which is therefore recursively enumerable, but in general not recursive:

  $$\mathcal{M}_3(x) : \text{if } \mathcal{M}_{\text{R1}}(x) \text{ accepts then accept, else run } \mathcal{M}_{\text{RE1}}(x).$$

  Observe, however, that there might be special cases: for instance, if $L_{\text{RE1}} \subset L_{\text{R1}}$, then $L_3 = L_{\text{R1}}$, hence it would be recursive.

- A similar reasoning can be carried out for L_4: if $x \in L_{\text{R1}}$, then we must also check if $x \in L_{\text{RE1}}$, but this only halts if the answer is positive: therefore, we can only create a machine that recognizes L_4, which is therefore recursively enumerable, but in general not recursive:

  $$\mathcal{M}_4(x) : \text{if } \mathcal{M}_{\text{R1}}(x) \text{ rejects then reject, else run } \mathcal{M}_{\text{RE1}}(x).$$

  Consider, however, the two following special cases (among many possible others):

  - if $L_{\text{R1}} \subset L_{\text{RE1}}$, then $L_4 = L_{\text{R1}}$, hence it would be recursive;
  - if L_{R1} and L_{RE1} were disjoint, then $L_4 = \emptyset$, and it would be trivially recursive.

- If both languages are RE, then their union L_5 is RE as well, because we can combine the two machines as follows:

  $\mathcal{M}_5(x)$: alternate the execution of $\mathcal{M}_{\text{RE1}}(x)$ and $\mathcal{M}_{\text{RE2}}(x)$; as soon as one accepts, then acept.

  We need to execute them "in parallel" because either of them might run forever, but we need only one to halt to accept the union language. Again, L_5 might as well be recursive.

- Finally, the intersection language L_6 can be treated as in the previous intersection cases, since we need both computations $\mathcal{M}_{\text{RE1}}(x)$ and $\mathcal{M}_{\text{RE2}}(x)$ to accept and halt; therefore, L_6 is RE.

1.2)

- $\mathcal{P}_1$ is semantic and non trivial, therefore undecidable.
- $\mathcal{P}_2$ is trivial: since we explicitly stated that L_{RE1} is *not* recursive, then no machine can decide it, therefore the property is empty (always false). Hence, $\mathcal{P}_2$ is trivially recursive.
- To assess $\mathcal{P}_3$, we "just" need to iterate through all possible strings x of lenght at most 100'symbols, and simulate the computation $\mathcal{M}_{\text{R1}}$ for at most $|x|^2+1$ steps. As soon as one of the simulations doesn't halt within the allotted number of steps, we reject the property. Otherwise, once all strings are tested, we accept. This procedure always terminates with acceptance or rejection, therefore $\mathcal{P}_3$ is recursive.
- The same procedure works for $\mathcal{P}_4$, since recursivity of the language does not play any role in the algorithm.

Observations

- Even though L_3 is a subset of L_{R1}, this fact alone doesn't allow us to conclude that it is recursive, because a subset of a recursive language is not necessarily recursive: think of the language Σ^* of all strings, which is trivially computable and contains every other language, e.g., HALT, which is uncomputable.
- The original text of the exam contained an error in the definitions of $\mathcal{P}_3$ and $\mathcal{P}_4$ ("less than $|x|^2$ steps" in place of "no more than $|x|^2 + 1$ steps"), which made it trivially false for $x = \varepsilon$. Answers were evaluated independent of this detail, however extra points and kudos to those who noticed it.

Exercise 2

Let $L \in \mathbf{P}$ be a deterministic polynomial-time language on finite alphabet Σ, and let L' and L'' be defined as follows:

- $L' = \Sigma^* \times L \times \Sigma^* = \{w_1 w_2 w_3 : w_1, w_3 \in \Sigma^* \wedge w_2 \in L\}$, the language of all strings on alphabet Σ that contain a word from L as a substring (contiguous sequence of symbols);
- $L'' = \{\sigma_1\sigma_2\sigma_3 \ldots \sigma_n \in \Sigma^* : \exists k, i_1, i_2, \ldots i_k \, (0 \leq k \leq n \wedge 1 \leq i_1 < i_2 < \cdots < i_k \leq n \wedge \sigma_{i_1}\sigma_{i_2} \ldots \sigma_{i_k} \in L)\}$, the language of all strings containing a (non necessarily contiguous) subsequence of symbols that compose a word in L.

For instance, if "cat" $\in L$, then "location" and "catalog" belong to both L' and L'', while the words "decoration" and "croissant" only belong to L''.

2.1) Discuss the deterministic time complexity of L' and L''.

2.2) What about their non-deterministic time complexity?

Solution 2

Let $\mathcal{M}_L$ be a deterministic, polynomial-time TM that decides L.

2.1) Given a string x, to decide $x \in L'$ we need to iterate through all of its substrings x' and check if $x' \in L$:

$$\begin{aligned} \mathcal{M}_{L'}(x) \quad : \quad & \text{for all } x' \text{ substring of } x, \text{ if } \mathcal{M}_L(x') \text{ accepts then accept and halt.} \\ & \text{Finally, reject and halt.} \end{aligned}$$

Since the number of substrings of x is polynomial (quadratic) in $|x|$, then the whole procedure is still polynomial, and $L' \in \mathbf{P}$.
On the other hand, a machine $\mathcal{M}_{L''}$ that decides L'' must iterate over all non-contiguous sub-sequences of characters of x, and there are $2^{|x|}$ of them, therefore in the worst case we need to call $\mathcal{M}_L(x')$ for an *exponential* number of sub-sequences of x. Therefore, all we can say is that $L'' \in \mathbf{EXP}$.

2.2) Since $L' \in \mathbf{P}$, then obviously $L' \in \mathbf{NP}$ too. About L'': if $x \in L''$, then we know that there must be a subsequence x' of x such that $x' \in L$. That subsequence is a polynomially verifiable certificate; verifying that x' is a certificate for $x \in L''$ requires two ckecks:

1. check that x' is actually a (not necessarily contiguous) subsequence of x by checking that all symbols of x' appear in x in the same order (done with a simple scan of the two strings);
2. run $\mathcal{M}_L(x')$ to verify that $x' \in L$.

Equivalently, we can simply define a non-deterministic machime $\mathcal{N}_{L''}$ that decides L'' as follows:

$$\begin{aligned} \mathcal{N}_{L''}(x) \quad : \quad & \text{Non-deterministically select a subsequence } x' \text{ of } x; \\ & \text{Run } \mathcal{M}_L(x'). \end{aligned}$$

The non-deterministic subsequence selection can be refined as "set $x' \leftarrow \varepsilon$, then for every symbol of x non-deterministically decide whether to append it to x' or not".
The two lines of $\mathcal{N}_{L''}$ are both polynomial-time, therefore, $L'' \in \mathbf{NP}$.

Observations

- Observe that we are not assuming that L is a finite list of words: all we know is that there is a polytime DTM that decides it. Therefore, an iteration over all strings in L is not possible.

Exercise 3

State and prove Rice's theorem about the undecidability of semantic, non-trivial properties of Turing machines.

Solution 3

See the lecture notes or any textbook on computability.

Observations

- Remember that to prove it (by contradiction) we need to reduce HALT to the property, not the other way round. I.e., start with assuming that property $\mathcal{P}$ is computable and show that we can decide the halting problem.