Department of Mathematics and Computer Science
Computation Theory problems

---

1. The following are alternative definitions for a computation machine similar to the Turing machine. For each item prove that it is equevalent to some Turing machine. In formal words: "If M' is a computation machine, it can recognize (decide) whether $w \in P$, for an input w and some set P, if and only if there is a Turing machine M that can recognize (decide) the same problem.

Hint: Explain how to build a Turing machine that simulates a given alternative machine

1. A standard Turing machine, but it's memory tape is infinte in both directions.
2. A standard Turing machine, but it has 2 memory tapes. (How about n memory tapes?)
3. (Non-determinestic Turing machine) A standard Turing machine, but it's transition function is defined as follows: $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$, where P is some probability distribution.

2. Is it possible to encode Turing machines?
More formaly is there any function from the family of all Turing machines TM to the family of all finite binary strings B.
How about from TM to the natural numbers?
If there exists such encoder function, can a Turing machine compute this function? Can you introduce a simple algorithm that given the description of a Turing machine, assignes a unique natural number to that Turing machine?

3. Enumerator: An enumarator is loosely defined as a Turing machine attached to a printer, so that the machine can print some strings as outputs. Every time the machine wants to print an output, it send the output to the printer. An enumerator E starts with a blank input on its work tape. If the enumerator doesn't halt, it may print an infinite list of strings. The language

enumerated by E is the collection of all the strings that it eventually prints out. Moreover, E may generate the strings of the language in any order, possibly with repetitions.
First give a formal definition for an enumerator, then prove that a language L_0 is Turing-recognizable if and only if there is some enumerator E, where $L(E) = L_0$.

4. Explain why the following is not a valid description for a Turing machine:

$M =$ "On input p, a polynomial over variables $x_1, x_2, ..., x_k$ ($p \in \mathbb{Z}[x_1, x_2, ..., x_k]$):

1. Evaluate p over all possible settings of $x_1, x_2, ..., x_k$.
2. if p evaluates to 0 on any of these settings, *accept*, *reject* otherwise."

• Is M a non-determinestic Turing machine?

• Can you give a Turing machine description to recognize polynimials with integer roots? how about a decidable Turing machine?

5. $k - PDA$: Let a $k - PDA$ be a pushdown automaton with k stacks. Thus a $0 - PDA$ is an NFA and a $1 - PDA$ is a conventional PDA. You already know that $1 - PDA$s are more powerful (recognize a larger class of languages) than $0 - PDA$s.

1. Show that $2 - PDA$s are more powerful that $1 - PDA$s.
2. Show that $3 - PDA$s are **NOT** more powerful that $2 - PDA$s.

HInt: Simulate a Turing machine with a $2 - PDA$

6. Show the following languages are decidable. (There is a decidable Turing machine for them)

1. $A_{DFA} = \{< D, w > \mid D$ is a DFA and it accepts string $w\}$.
2. $A_{NFA} = \{< N, w > \mid N$ is an NFA and it accepts string $w\}$.
3. $A_{REX} = \{< R, w > \mid R$ is a regular expression and it generates string $w\}$.
4. $E_{DFA} = \{D \mid D$ is a DFA and $L(D) = \emptyset\}$.
5. $EQ_{DFA} = \{< D_1, D_2 > \mid D_1$ and D_2 are DFAs and $L(D_1) = L(D_2)\}$.
6. $A_{CFG} = \{< C, w > \mid C$ is a context-free grammer and it generates string $w\}$.

7. (Diagonalization) Show that $A_{TM} = \{< M, w > \mid M$ is Turing machine and it accepts $w\}$ is undecidable.

8. (Turing-decidability vs. Turing-recognizablity):
We say a language L is co-Turing-recognizable if and only if $\overline{L}$ is Turing-recognizable.
Show that a language L is Turing-decidable if and only if it is Turing-recognizable and co-Turing-recognizable.

9. Using a counting argument, show that there are uncountably infinite non-Turing-recognizable languages.
Give an exaple of a non-Turing-recognizable language.

Hint: Show that any non-Turing-decidable language, trivially admits a non-Turing-recognizable language.

10. Say that an NFA is "ambiguous" if it accepts some string along two different computation branches.
Let $AMBIGNFA = \{N \mid N$ is an ambiguous NFA$\}$. Show that $AMBIGNFA$ is decidable.

Suggestion: One elegant way to solve this problem is to construct a suitable DFA and then run E_{DFA} on it.

11. Let C be a language. Prove that C is Turing-recognizable if and only if, there exists some Turing-decidable language D such that, $C = \{x \mid \exists y(< x, y > \in D)\}$.

12. Let $L = \{< M > \mid$ Turing machine M, accepts w, if and only if M accepts $w^R\}$. Show that L is an undecidable language.

13. Show that the following languages are undecidable:

1. $HALT_{TM} = \{< M, w > \mid M$ is a Turing machine and it halts on input $w\}$
2. $E_{TM} = \{M \mid M$ is a Turing machine and $L(M) = \emptyset\}$.
3. $REGULAR_{TM} = \{M \mid M$ is a Turing machine and $L(M)$ is a regular language$\}$.
4. $EQ_{TM} = \{< M_1, M_2 > \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\}$.

14. Linear Bounded Automaton:A linear bounded automaton is a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is—in the same way that the head will not move off the left-hand end of an ordinary Turing machine's tape. By using a tape alphabet larger than that of the language, the memory space can be scaled up by a constant factor and that is where the "Linear Bounded Automaton" comes from.

Despite limited memory, LBAs are very powerful. For instance deciders for A_{DFA}, A_{CFG}, E_{DFA} and E_{CFG} are all LBAs. In fact creating a Turing-decidable language that is not LBA-decidable is not easy.

• Prove that $A_{LBA} = \{< M, w > \mid M \text{ is an LBA and it accepts } w\}$ is decidable.

Hint: Show that there are finitely many configurations for an LBA with input of size n.

• Show that $E_{LBA} = \{M| \; M \text{ is an LBA and } L(M) = \emptyset\}$ is undecidable.

15. Prove that the Post Correspondence Problem (PCP), is undecidable on any finite alphabet.

16. Non-Turing-recognizable and non-co-Turing-recognizable: In this problem we aim to show that EQ_{TM} is neither Turing-recognizable, nor co-Turing-recognizable.

1. Give a mapping reduction from A_{TM} to $\overline{EQ_{TM}}$.
2. Give a mapping reduction from A_{TM} to $\overline{\overline{EQ_{TM}}} = EQ_{TM}$

17. Let $J = \{w \mid \text{either } w = 0x \text{ for some } x \in A_{TM}, \text{ or } w = 1y \text{ for some } y \in \overline{A_{TM}}\}$. Prove that J is neither Turing-recognizable nor co-Turing-recognizable.

18. Rice's theorem.

For any non-trivial property of the language of a Turing machine M, the problem of determining whether a given Turing machine's language has property P is undecidable.

In more formal terms: Let P be be a family of Turing machine descriptions, where P satisfies two properties:

I) P is non-trivial:

Meaning that P contains at least one Turing machine description, but not all of them.

II) P is a property of the TM's language:

Meaning that if $L(M_1) = L(M_2)$ then $< M_1 >\in P$ iff $< M_2 >\in P$.
P is an undecidable language.
Prove Rice's theorem.

19. Prove the following languages are undecidable.

1. $INFINITE_{TM} = \{< M > \mid M \text{ is a TM and } L(TM) \text{ is infinite}\}$
2. $ALL_{TM} = \{< M > \mid M \text{ is a TM and } L(TM) = \Sigma^*\}$

20. Suppose that A_{TM} is decided by an oracle. Using this oracle, describe a TM that is guaranteed to state the answer to ”Goldbach's conjecture”. How about ”Collatz conjecture”?

21. Consider the problem of determining whether a single-tape Turing machine ever writes a blank symbol over a nonblank symbol during the course of its computation on any input string. Formulate this problem as a language and show that it is undecidable.

22. Let M a be a model. We define $Th(M)$ to be all true sentences in the language of M. Also we say a model is decidable iff there is a decidable Turing machine for $Th(M)$.
Church proved that $Th(\mathbb{N}, +, \times)$ is undecidable. Show that:

1. $Th(\mathbb{N}, +)$ is decidable.
2. $Th(\mathbb{N}, <)$ is decidable.
3. (*Optional) Theory of modular arithmatics is decidable; For any natural number $m \geq 2$, $Th(\mathbb{Z}/\mathbb{Z}_m)$ is decidable.

23. Using RT (recursion theorem), give a simpler proof that A_{TM} is undecidable.

24. Using RT (recursion theorem), give an alternative proof for Rice's theorem.

25. For a given TM M, we say M is *”minimal description”*, if and only if for all TMs M' that M' is equivalent to M, $| < M > | \leq | < M' > |$.
Let $MIN_{TM} = \{< M > \mid M \text{ is minimal description}\}$. Show that MIN_{TM} is not T-recognizable.
(*Optional) Show that any infinite subset of MIN_{TM} is not T-recognizable.

26. Fixed-point RT: Any T-computable transformation on Turing machines, has a fixed point.

Let $t : \Sigma^* \to \Sigma^*$ be a computable transformation. Prove that some TM F exists such that $t(< F >)$ describes a Turing machine equivalent to F. Assume that any string that is not a proper Turing machine description, is concidered the description of a Turing machine that always rejects immediately.

27. Prove that E_{TM} is Turing-reducable (decidable relative) to A_{TM}.

28. Prove that PCP is Turing-reducable (decidable relative) to A_{TM}.

29. Using a counting argument, show that there are some languages that are not recognizable by any oracle Turing machine with an oracle for A_{TM}.