Computability and Computational Complexity, A.Y. 2024–2025

Guide to the answers

Monday, August 26, 2025

Exercise 1

For each of the following properties of a Turing machine $\mathcal{M}$, prove whether it is computable or not. When possible, invoke Rice's Theorem.

1.1) $\mathcal{M}$ accepts at least one input string.

1.2) $\mathcal{M}$ accepts at least one input string of length 3.

1.3) $\mathcal{M}$ accepts at least one input string of length 3 within 1000 steps.

1.4) $\mathcal{M}$ accepts at least one input string within 1000 steps.

Solution 1

1.1) The property is clearly semantic: if two TMs decide the same language L, they both have the property (if $L \neq \emptyset$) or neither has it (if $L = \emptyset$). The same observation proves that the property is not trivial. Therefore, Rice's Theorem applies, and the property is not computable.

1.2) Same as above, where the discriminant is whether L contains at least one 3-symbol string or not.

1.3) The property, while not trivial, isn't semantic: take a TM $\mathcal{M}$ that accepts a 3-symbol string in few steps (thereby having the requested property) and modify it into $\mathcal{M}'$ by adding 1000 dummy steps before accepting, so that $L(\mathcal{M}) = L(\mathcal{M}')$; however, $\mathcal{M}'$ takes too many steps and doesn't have the property. Therefore Rice's Theorem doesn't apply.
The property is computable: to check whether $\mathcal{M}$ has it, we just simulate it for at most 1000 steps on every 3-symbol string (a finite set). If $\mathcal{M}$ ever accepts an input within the step limit, then we conclude that it has the property; if we exhaust all strings without reaching acceptance, we conclude that $\mathcal{M}$ doesn't.

1.4) Same as above. Although we apparently have no limit on the accepted string size, we know that at most the first (or last, depending on the initial position of the machine) 1000 symbols will ever affect a decision within the allowed number of steps, so we really need to simulate the machine on all 1000-symbol inputs.

Remarks

- For the last two points, the two key observations are:
  - we need to check all possible input string, because we don't know which one is going to work;
  - however, the number of strings to be tested is finite, therefore our search always ends.

  Without these explicit observations, the answer is considered incomplete and cannot receive full marks.
- the fact that a property doesn't meet the conditions of Rice's Theorem doesn't mean that it is computable.

Exercise 2

Prove the following assertions by describing the appropriate polynomial-time reductions:

2.1) CLIQUE $\leq_{\mathrm{P}}$ INDEPENDENT SET;

2.2) INDEPENDENT SET $\leq_{\mathrm{P}}$ VERTEX COVER;

2.3) SATISFIABILITY $\leq_{\mathrm{P}}$ 3-SATISFIABILITY.

Solution 2

See the notes. To expand a little further:

2.1) A clique of size k corresponds to an independent set of the same size on the complementary graph (where two nodes are connected if and only if they weren't in the original graph).
Therefore, a graph has a clique of size k if and only if its complement has an independent set of the same size. Complementing a graph is clearly a polynomial-time task.

2.2) Given a graph $G = (V, E)$ and an independent set $I \subseteq V$ the folowing is true: for every edge $e \in E$, at most one of its endpoints can be in I. Therefore $C = V \setminus I$ is a vertex cover (every edge has at least one endpoint in C). The converse is also true: given a vertex cover C, every edge must have at least one endpoint in C, hence $I = V \setminus C$ is an independent set.
Therefore, a graph $G = (V, E)$ has an independent set of size k if and only if it has a vertex cover of size $|V| - k$. The reduction only requires an arithmetic operation.

2.3) Clauses with one or two literals can always be expanded by repeating one of them (Boolean operators are idempotent); a longer clause $(t_1 \vee t_2 \vee t_3 \vee \dots)$ can be split by introducing a new variable y and replacing it with the conjunction $(t_1 \vee t_2 \vee y) \wedge (\neg y \vee t_3 \vee \dots)$, where the second clause has one less term than the original one and can be recursively reduced by the same means. It is easily verifiable that any truth value assignment satisfying the original clause also satisfies the exploded ones (for an appropriate assignment to y), and vice versa.

Remarks

Shorter explanations, leaving out some details and proofs, are acceptable as long as they convey the idea.

Exercise 3

3.1) Define the classes **L**, **NL**, **P** and **NP**.

3.2) Prove that **L** $\subseteq$ **P**.

Solution 3

See the notes.

Remarks

- Remember that **L** and **NL** are *space* complexity classes (i.e., related to the number of used cell tapes), while **P** and **NP** are *time* classes (related to the number of steps).
- To define **L** and **NL** we need to mention the existence of a work tape, otherwise by just reading the input we would use linear space!
- **L** is not equal to **NL**, in fact it is probably strictly contained. Remember that **L** $=$ DSPACE$(\log n)$, while Savitch's theorem only says that **NPSPACE** $=$ DSPACE$((\log n)^2)$.