Computability and Computational Complexity, A.Y. 2018–2019

Second written test

Mauro Brunato

Tuesday, February 12, 2019

Exercise 1

Prove that if we had an algorithm that computes the Halting Problem for a generic Turing Machine $\mathcal{M}$ on a generic input x, then we could use it to decide Goldbach's conjecture ("every even number greater than 2 is the sum of two primes").

Exercise 2

2.1) Define the probabilistic complexity classes **BPP** and **PP**; describe their relationship.

2.2) In particular, given that the "B" in **BPP** stands for "Bounded error," explain what it means and in what sense the error probability in **BPP** is bounded while in **PP** it is not.

Exercise 3

The police must intercept all cellphone communications within a group of n people and have the following information:

- their identities and phone numbers (plus any other info that is needed in such cases);
- which pairs of people know each other (people who don't know each other will not directly communicate).

They want to know if there is a way to be sure to intercept all calls within group members while putting only k phones under surveillance (we assume that a communication can be intercepted if at least one of the two phones is under surveillance).

3.1) Prove that the problem is **NP**.

3.2) Prove that the problem is **NP**-complete by reduction from some other known problem.

Exercise 4

For each of the following properties of a Turing machine $\mathcal{M}$, prove whether it is computable or not.

4.1) $\mathcal{M}$ accepts inputs x whose length is even.

4.2) $\mathcal{M}$ has an even number of states.

4.3) $\mathcal{M}$ is a universal Turing machine.

4.4) $\mathcal{M}$ never returns to its initial state when executed on an empty input tape.

Exercise 5

Prove that STCON $\in$ **NL**.

Hint — *Remember that STCON is the language of directed graphs $G = (V, E)$ and node pairs $s, t \in V$ such that there is a path from s to t in E.*

Solution traces

Exercise 1

Prove that if we had an algorithm that computes the Halting Problem for a generic Turing Machine $\mathcal{M}$ on a generic input x, then we could use it to decide Goldbach's conjecture ("every even number greater than 2 is the sum of two primes").

Solution trace

First, observe that checking whether a given number n is a sum of two primes is easy: just scan all primes p up to $\sqrt{n}$ and check if $n - p$ is a prime too. Thus, we can set up a machine that looks for counterexamples by scanning all even numbers greater that 2 and halts if and only if a number n is found *not* to be the sum of two primes. Such machine halts if and only if Goldbach's conjecture is false.
[As an alternative, one could say that since we proved that the HP is not computable, the existence of a machine that computes it would prove that Mathematics is inconsistent, and therefore even Goldbach's conjecture could be proved to be both true and false.]

Observations

- It is incorrect to state that Goldbach's conjecture is undecidable: it might as well be, but more likely we haven't found the correct way to prove it (yet).

Exercise 2

2.1 — Define the probabilistic complexity classes **BPP** and **PP**; describe their relationship.
2.2 — In particular, given that the "B" in **BPP** stands for "Bounded error," explain what it means and in what sense the error probability in **BPP** is bounded while in **PP** it is not.

Solution trace

See the lecture notes.

Observations

- It's important to say that the machine recognizing the language (in the probabilistic sense) is polynomial-time bounded.
- Another important point is the fact that the bound in **BPP** allows the probability of error to be reduced at will by repeated executions, while **PP** doesn't guarantee this.

Exercise 3

The police must intercept all cellphone communications within a group of n people and have the following information:

- their identities and phone numbers (plus any other info that is needed in such cases);
- which pairs of people know each other (people who don't know each other will not directly communicate).

They want to know if there is a way to be sure to intercept all calls within group members while putting only k phones under surveillance (we assume that a communication can be intercepted if at least one of the two phones is under surveillance).
3.1 — Prove that the problem is **NP**.
3.2 — Prove that the problem is **NP**-complete by reduction from some other known problem.

Solution trace

3.1 — The certificate is the list of k people to be put under surveillance. It is clearly polynomial wrt the problem size (it is a subset of the n people), and we just need to check that each of the n people is either in the list, or knows someone in the list. We can run this check in quadratic time (on a computer, a little more on a TM).
3.2 — We can reduce VERTEX COVER to this problem: given an undirected graph $G = (V, E)$, let us build a set of $n = |V|$ people, and let persons i and j know each other iff $\{i, j\} \in E$. Then, G has a vertex cover of size k iff we can intercept all communications by putting k people under surveillance.

Observations

- Basically, the stated problem is VERTEX COVER under disguise. The observation that the problem is VERTEX COVER and therefore it is **NP**-complete would guarantee maximum marks.
- As usual, pay attention to the sense of the reduction. Reducing our problem to something else would prove nothing.

Exercise 4

For each of the following properties of a Turing machine $\mathcal{M}$, prove whether it is computable or not.
4.1 — $\mathcal{M}$ accepts inputs x whose length is even.
4.2 — $\mathcal{M}$ has an even number of states.
4.3 — $\mathcal{M}$ is a universal Turing machine.
4.4 — $\mathcal{M}$ never returns to its initial state when executed on an empty input tape.

Solution trace

4.1 — The property is semantic (only depends on the recognized language) and non-trivial (some machines do accept even-length strings, others don't), therefore Rice's theorem applies and it is not computable.
4.2 — Any TM encoding exposes the set of states of a machine; we just need to count them. Observe that the property is not semantic: two machines might recognize the same language but have different numbers of states.
4.3 — This is a semantic property: being a UTM is defined on the basis of what is read and written on the tape, and not on the machine's structure, therefore (since it's non trivial, as some machines are not UTMs) Rice's theorem applies.
4.4 — The property is about states, therefore it is not semantic. To show that it is uncomputable, as is often the case with properties targeting the visit of specific states, let the generic machine $\mathcal{M}$ have states $Q = \{s_1, \dots, s_n\}$, where s_1 is the initial state. Let us apply to $\mathcal{M}$ the two following changes, transforming it to machine $\mathcal{M}'$:

1. Add state s_0, with the same transition rules as s_1, and let it be the new initial state;
2. replace all occurrences of the halting state in the machine's ruleset with transitions to s_0.

Note that s_0 will be revisited in $\mathcal{M}'$ iff $\mathcal{M}$ reaches the halting state. Therefore, by being able to decide property 4.4 we would be able to decide the halting problem (for empty input), which we know to be undecidable.

Observations

- Property 4.1 does not ask for the decidability of the language of even-length strings (which is obviously computable), but about the property of a TM of accepting that language or not.
- While property 4.2 only requires to *count* states, many people still write about emulating the machine: this is clearly useless, a static analysis is enough.
- One might feel uneasy with accepting that "being a UTM" is a semantic property, since UTMs are not decision machines, but perform more complex tasks. However, we can even take the direct way and "bend" the proof of Rice's theorem to show that the property is not computable: let $\mathcal{U}$ be a UTM (which we know to exist), and suppose that machine $\mathcal{N}$ can decide on the property of "being a UTM" (so, $\mathcal{N}(\mathcal{U}) = 1$). Given any machine $\mathcal{M}$ with input x, we could build machine $\mathcal{M}'$ by executing $\mathcal{M}(x)$ followed by $\mathcal{U}$ (as in Rice): $\mathcal{M}'$ is a UTM if and only if $\mathcal{M}(x)$ halts:

  $$\mathcal{N}(\mathcal{M}') = 1 \quad \Leftrightarrow \text{HALT'}(\mathcal{M});$$

  therefore we would be able to solve the halting problem.
- A few people wrote that "UTMs are computable, therefore property 4.3 is computable". I believe this is a (confused) way of stating that UTMs exist, which is true; however, the question was about being able to recognize that a given TM $\mathcal{M}$ is a UTM, not aout the existence of UTMs.

- Observations such as “property 4.* is uncomputeble because the machine could enter an infinite loop and we would never know” are useless, since they don’t explain why and, indeed, many cases of infinite loops *are* detectable.
- Finally, it is wrong to say that any infinitely running machine would eventually revisit a known configuration: this is only true in special cases, e.g., for space-bounded machines; a general machine has an infinite configuration space (otherwise, the Halting Problem would be decidable).

Exercise 5

Prove that STCON $\in$ **NL**.

Solution trace

See the notes.

Observations

- A sketch of the proof was necessary; just writing “this is Theorem 24” is not enough, since numberings tend to change between versions of the document.
- A few people actually tried to prove that STCON $\in$ DSPACE$\big((\log n)^2\big)$ (currently Theorem 25), which is way more complex and not what they were asked for.