Harvard University
Computer Science 121

Problem Set 7

Due November 10, 2015 11:59pm

Problem set by Kevin Zhang

Collaboration Statement: I collaborated with Tomoya Hasegawa, got help from no one else, and referred to no other resources.

PART A (Graded by Sam and Charles)

PROBLEM 1 (4+4+4 points, suggested length of 1 page)

Indicate whether each of the following languages is decidable. If it is decidable, give a **high-level**[1] (but complete) description of a Turing Machine that decides it. Otherwise, prove it is not decidable by contradiction: show that if a decider M_L existed for it, one could use M_L to construct a new decider M'_L that decides a problem we have proven to, in fact, be undecidable (such as $\textsc{Halt}_{\mathsf{TM}}$). You do not need to define the mapping reduction function f for this problem.

In all cases, let T be a deterministic one-tape TM.

(A) $L = \{\langle T \rangle : T \text{ takes more than 53 steps on at least one input.}\}$

(B) $L = \{\langle T \rangle : T \text{ takes more than 77 steps on at least one input accepted by } T.\}$

(C) $L = \{\langle T \rangle : T \text{ takes more than 824 steps on all inputs accepted by } T.\}$

Solution.

(A) This is decidable. Let T be the TM for L, and let an arbitrary input to T be t. We attempt to decide t by giving it inputs. Since we only care about up to 53 steps, we don't have to consider strings of length greater than 53; if t ever tries to move past the 53th character, then it must take more than 53 steps on that input, so t is accepted. T decides an input t as follows:

- For every input string s of length 53 or less, do the following:
  - Run t on s.
  - If by the 53rd step t has not stopped, accept t.
  - If by the 53rd step t has stopped, move on to the next string. If there are no more strings, reject t.

[1]See Piazza for an explanation of what a high-level description of a TM entails.

This TM clearly stops at some point because there are finitely many strings of length 53 or less, and we only run up to 53 steps on each string.

(B) This is undecidable. Suppose R decides L. We construct a reduction as follows; given $\langle M, w\rangle$, we construct M':

- If an input $x \neq w$, reject.
- If an input $x = w$, do 77 meaningless steps (this can be done in a variety of ways), then run M on w.
- If M accepts, accept. If M rejects, reject.

Now, we run R on M'; if R accepts, accept, and if R rejects, reject. M' works by either accepting the empty language, or by accepting only w if M accepts w; i.e. it is non-empty iff M accepts w. Hence, either M' recognizes the empty language or it recognizes a language consisting of exactly one string that takes more than 77 steps to recognize. Hence, we can use R to decide M', and therefore A_{TM} is a reduction of the given problem. A_{TM} is undecidable, so we have a contradiction; therefore, no TM decides L.

(C) This is decidable. Similarly to the above problem, we decide this by considering all strings of length up to 824. If no strings are accepted with length at most 824, then we are done because all strings that are accepted must therefore take more than 824 steps. Otherwise, if a string of length 824 is accepted, we check to make sure it still takes > 824 steps to accept. Note further that if a string has length greater than 824, we don't care about it either way; it's okay if an input string to the input TM is accepted or rejected after more than 824 steps. We construct T that takes an input TM t as follows:

- For every input string s of length 824 or less, do the following:
  - Run t on s.
  - If by the 824th step t has not stopped, move on to the next string.
  - If by the 824th step t has accepted, reject.
  - If by the 824th step t has rejected, reject.

This TM runs an input TM over all strings of length 824 (which is a finite process) for up to 824 steps (which is also a finite process), so it must terminate. Moreover, it is clear that this TM accepts exactly L.

PROBLEM 2 (3+3 points, suggested length of 1/2 a page)

In the near future you're working as an engineer at Google/Microsoft/Facebook when your manager asks you to write the following two programs. Is this a problem? Provide a proof for your argument.

(A) Take another program's code as input and decide if that program is implemented with the shortest amount of code possible.

(B) Take another program's code and remove all inaccessible (dead) code from it.

Solution.

(A) This is an undecidable problem! Sipser proves that MIN_{TM} is not recognizable in 6.1; since MIN_{TM} is not recognizable, it is impossible for us to determine if a given program (TM) is in MIN_{TM}. That is, we can't determine if a given program is minimal.
(B) This is impossible. Assume we have a program P that removes all dead code. We use this to solve E_{TM}. Take an arbitrary TM T. Run P on T and call the result T'. In T', check if there are any accept states. All states in T' are reachable; if we find any accept states in T', then T must accept some string. If we don't find any accept states in T', then there's no way for T to accept any string. Hence, we have just determined whether T is empty, an impossibility. Therefore, the given problem is impossible.

PROBLEM 3 (4 points, suggested length of 1/3 a page)

Let $A = \{\langle R, S\rangle :\ R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S)\ \}$. Show that A is decidable by giving a high level description.

Solution.
Since R and S are regular expressions, we can construct DFAs for each of them. If we have two DFAs for R and S (let's call them X and Y, respectively), we can decide any input by simulating all possible pairs of states for the two DFAs; if it is possible to reach a case where X is in an accept state and Y is in a reject state, then we reject. Otherwise, we accept. We can do this by keeping a table of all possible pais of states for R and S; let rows in a table correspond to states of X and columns correspond to states of Y. Then starting from the cell that corresponds to the start states of both DFAs, exhaust all possible transitions and mark each cell reached as "reachable". If we ever mark a cell that contains an accept state for X and a reject state for Y, reject; otherwise, once we have exhausted all possible transitions (of which there are finitely many), accept. This process must terminate, so we are done.

PART B (Graded by Juan and Zhengyu)

PROBLEM 4 (6 points, suggested length of 1/3 page)

Prove that any decidable language L is mapping reducible to any non-trivial decidable language L'. (Any decidable language other than Σ^*, $\emptyset$)

Solution.
To prove L is mapping reducible to L', we just need to construct a mapping function such that for any w that is an input to L, $f(w) \in L'$ if $w \in L$ and vice versa. Note that we only have to consider inputs to L, NOT L'. To do this, we only need one each of elements in and not in L'; let's call them A and B respectively. Our mapping function sends all $w \in L$ to A, and all $w \notin L$ to B. It is obvious that if $w \in L$, $f(w) \in L$. In addition, for some w, if $f(w) \in L$, our function ensures that w must be in L. Hence, we have constructed a satisfactory mapping and are done.

PROBLEM 5 (6 points, suggested length of 1/3 page)

For any Turing Machine M, the language accepted by M, $L(M)$ is always recursively enumerable. Let M_F be the machine created by flipping the accept and reject states of M. Prove that $L(M_F)$ is not always recursive whenever $L(M)$ is recursive.

Solution.
We only know that $L(M)$ is recursive, not that M itself always halts; there exists a TM that recognizes $L(M)$ and always halts, but that TM does not have to M. Consider the following TM T; given an input TM m, T runs m on all strings of length 77 for up to 77 steps each. If any of the inputs halt and are accepted by m, T rejects m. Otherwise, make T run forever. This TM's accepted language is exactly the empty language, which is decidable. Now, if we flip the accept and reject states here, we get a TM that instead accepts an input TM m iff any inputs are accepted within 77 steps and runs forever otherwise. From 1B in this problem set, we know that this new TM's language is undecidable! The given TMs constructed here are counterexamples to the claim $L(M_F)$ recursive iff $L(M)$ recursive, so we are done.

PROBLEM 6 (6 points, suggested length of 1/3 page)

A computation of a Turing Machine *loops* if it repeats a configuration, that is, re-enters the same state with the identical tape and the head position. Prove that if every non-halting computation of M loops, then $L(M)$ is decidable. In particular, you need to give a high level description of a Turing machine that decides $L(M)$, and provide a short justification.

Solution.
Note that there are finitely many different configurations a TM can be in; given a set of states Q and an input tape alphabet Σ, the number of possible configurations is $|Q| \times |\Sigma|$. Thus, it is possible for us to keep track of every configuration that the Turing Machine has visited. Thus, we can decide $L(M)$ based on M as follows: run M on an input string s for up to $N + 1$ steps. Keep track of the visited configurations along the way (e.g by writing them to the tape somewhere). If at any point M halts, return what it ends with (either accept or reject). If at any point we visit a configuration that has already been visited, reject the input string because it must loop. By the $N + 1$th step, at least one configuration must have been visited twice according to the Pigeonhole Principle, so the TM we have constructed will halt. This TM always halts, and accepts only if the given TM M accepts the input string; it decides $L(M)$, as required.