Harvard University
Computer Science 121

Problem Set 6

Due November 3, 2015 11:59pm

Problem set by Kevin Zhang

Collaboration Statement: I collaborated with Tomoya Hasegawa and Mandela Patrick, got help from no one else, and referred to no non-class resources.

PART A (Graded by Serena and Cecilia)

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

Prove that the class of decidable languages is closed under union.

Solution.
Let $M1$ be a TM that decides $L1$, and $M2$ be a TM that decides $L2$. Constructing a TM that decides $L1 \cup L2$ is simple; given an input w, let M operate as follows:

- Simulate $M1$ on w.
- $M1$ must halt. If it accepts, M halts and accepts. Otherwise, simulate $M2$ on w.
- $M2$ must halt. If it accepts, M halts and accepts. Otherwise, M halts and rejects.

M must halt because both $M1$ and $M2$ must halt. Clearly, M only accepts iff $M1$ or $M2$ accepts w, and rejects if both reject w. Hence, the union of any two decidable languages is decidable, and we are done.

PROBLEM 2 (6 points, suggested length of 1/3 page)

Prove that the class of Turing-recognizable languages is closed under intersection.

Solution.
Let $M1$ be a TM that recognizes $L1$, and $M2$ be a TM that recognizes $L2$. We construct a TM that recognizes $L1 \cap L2$; given an input w, let M operate as follows:

- Simulate $M1$ on w.
- If $M1$ accepts w, then jump to step 5.
- If $M1$ rejects w, then $M1$ halts and rejects w, and M halts and rejects w as well.
- If $M1$ loops, then so does M, and therefore M rejects w.
- Simulate $M2$ on w.
- If $M2$ accepts w, then $M2$ halts and accepts w, and M halts and accepts w as well.
- If $M2$ rejects w, then $M2$ halts and rejects w, and M halts and rejects w as well.

- If $M2$ loops, then M loops as well and therefore rejects w.

M only accepts if both $M1$ and $M2$ accept an input in succession; all other cases result in a rejection or a loop. Hence, M accepts exactly $L1 \cap L2$.

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

(A) Consider $L = \{\langle M \rangle : M \text{ is a TM that accepts no strings shorter than 42 characters in length}\}$. Prove that L is not decidable. Hint: Consider Rice's Theorem.

(B) Use that the fact that L is not decidable to prove that L is not recognizable.

Solution.

(A) Suppose there is a TM M that recognizes L. The question of whether an input M' is accepted by M can be rephrased as the question of whether M''s language belongs to the set of languages that consist of strings of length at least 42. But Rice's Theorem tells us that this question is undecidable, since the set of languages that consist of strings of length at least 42 contain at least one language (e.g. $\{a^42\}$), and the complement contains the language $\{a\}$. Thus, the given language is undecidable.
(B) A language is decidable iff it is recognizable and co-recognizable; that is, both it and its complement are recognizable. If it is undecidable, either it or its complement, or both are not Turing-recognizable. Since we already know L is not decidable, if we manage to prove that L^c is recognizable, then L must be the unrecognizable one. L^c is the set of all TMs that accept at least one string of length less than 42. Since there are finitely many strings of length less than 42, it is possible to test any TM to see if it accepts a string of length less than 42 (this comes from the fact that alphabets are finite); simply run instances of the TM on all such strings in parallel, and accept if any of the concurrent instances accept. Hence, L^c is recognizable, and therefore L must not be.

PROBLEM 4 (3 points, suggested length of 5 lines)

Define $\textsc{Prefix}(L) = \{x : xy \in L \text{ for some } y \in \Sigma^*\}$. Show that if L is Turing-recognizable, then $\textsc{Prefix}(L)$ is Turing-recognizable.

Solution.
If L is Turing-recognizable, then it has some enumerator E. We create a new enumerator E' using E as a subprocess; for every string that E outputs, iterate over the string and print out all of the characters up to that point. The idea here is to use every string outputted from the enumerator to generate each prefix for that particular string; e.g. if *abc* is printed out, *a*, *ab*, and *abc* are printed out by the constructed enumerator in turn. This is a construction of an enumerator that gives $\textsc{Prefix}(L)$; hence, $\textsc{Prefix}(L)$ is Turing-recognizable as well, since any language that can be enumerated is recognizable.

PART B (Graded by Erin and Zhengyu)

PROBLEM 5 (8 points, suggested length of 2/3 page)

Let $L = \{\langle M\rangle : M$ moves left on the tape at some point when run on $\varepsilon\}$. Show that L is decidable.

Solution.
We can construct a TM that decides L by only iterating up to a maximum number of times. Specifically, if an input TM M has only Q distinct states, we iterate $Q+1$ times. This TM works as follows:

- Given an input machine M, count the number of distinct states and store that number as Q.
- Do the following up to $Q+1$ times at most; at the end, reject.
  - Simulate one step of M on ε.
  - If M moved left, terminate and accept.
  - If M terminated, terminate and reject.
  - Otherwise, continue.

The reasoning behind this construction is as follows: if M moves left during the first $Q+1$ moves, we are done. Otherwise, it must move right during the entire operation of our constructed TM, at the end of which we reject the input. In addition, because we are running this on the empty string, M always reads a blank input for the next step. This means that, during the simulation of M in our machine, if M never moves left then there is only one possible "next" state for each state in M. There are exactly Q distinct states; hence, if we iterate $Q+1$ times, the Pigeonhole Principle tells us that at least one of the states appears at least twice. But if one of the states appears twice (say, q), M must be looping, since M therefore transitions to the same state out of both occurrences of q (and transitions to the same state out of the new state, and so on). The TM we have constructed always terminates after at most $Q+1$ states (and since no TM has an infinite number of states, this is finite as well), so we are done.

PROBLEM 6 (10 points, suggested length of 3/4 page)

Show that a language L_1 is Turing-recognizable if and only if there exists some decidable language L_2 such that $L_1 = \{x :$ there exists y such that $\langle x, y\rangle \in L_2\}$.

Note that the $\langle\rangle$ notation signifies that you are "stringify-ing" the contents within the bracket, since Turing Machines take strings as input.

(Hint: Imagine that y gives you information about an accepting computation on x, if one exists.)

Solution.

The forward direction is easier; we prove that first. Suppose we have a language L_1 that is Turing-recognizable. We wish to show that there exists some decidable language L_2 such that $L_1 = \{x$: there exists y such that $\langle x, y\rangle \in L_2\}$. In other words, for every object in L_1, there exists some other object y such that the pair $\langle x, y\rangle$ is in L_2.

We construct L_2 in the following manner; since L_1 is Turing-recognizable, there exists a TM M for which $L(M) = L_1$. Then let L_2 be the language $\{\langle x, y\rangle : x \text{ is accepted by } M \text{ and } y \text{ is the number of configurations } M \text{ goes through while accepting } x\}$. Because x is restricted to accepted inputs for M, y is guaranteed to be a finite natural number. We can now construct a TM M' that recognizes L_2 as follows:

- M' takes in two inputs (string-ified into one), x (the input) and y (the number of configurations to go through).
- Check to make sure x is a valid input for M. If not, reject. If yes, continue.
- Initialize a counter to 0 and start M on x.
- Perform the following steps up to y times:
  - Compute the next configuration of M on x.
  - Increment the counter.
  - If M has rejected x, reject.
  - If M has accepted x and the counter is not yet at y, reject.
  - If M has accepted x and the counter is exactly y, accept.
- Reject.

This TM works by simulating up to y configurations of M on x; if M accepts x after going through exactly y configurations, we accept the pair $\langle x, y\rangle$. Since this only runs up to y times, this TM must halt; hence, this TM's language is decidable. Note that this also satisfies the constraints of the problem; since M' depends on M and y depends exactly on x (and there is exactly one y per x in L_2), only L_1's strings are accepted as x in the constructed TM.

The backwards direction is a bit harder; assume we have some language L_1, and there exists some decidable language L_2 such that $L_1 = \{x : \text{there exists } y \text{ such that } \langle x, y\rangle \in L_2\}$. We need to prove that there exists a TM that accepts L_1; we construct a TM M as follows:

- M takes an input string w. Verify that w is in the correct format to be given to L_2; if not, reject.
- The alphabet for y for L_2 must be finite; hence, the set of strings possible for y is enumerable. Iterate over all such strings (this is possible because the set is enumerable) and do the following:
  - If L_2 accepts $\langle w, y\rangle$, where y is the current string we're testing, terminate and accept w. This process must terminate because L_2 is decidable.
  - Otherwise, continue to the next string.

This TM works by iterating over all possible strings and trying to see if L_2 accepts any particular string in conjunction with a given input w. The decidability of L_2 is critical because otherwise it would be possible to get stuck on a particular string, and be unable to test all possible strings for y. It guarantees that we can continue testing strings, and in particular that the constructed TM will eventually test any particular string. Hence, the constructed TM may fail to terminate (it "loops"), or it may end by accepting an input; the TM only explicitly rejects if the given string is of an incorrect format. We have constructed a TM that recognizes L_1, as desired.

PROBLEM 7 (Challenge!! (3) points, suggested length of 1/3 page)

Show that every infinite regular language has a subset that is Turing-recognizable but undecidable.

Solution.
Suppose we have an infinite regular language R. We need to construct a correspondence between a subset of the regular language and an undecidable one; if we do so, the subset must also be undecidable.