

CS 273: Intro to Theory of Computation, Fall 2007

Problem Set 11 Solutions

1. Dovetailing

Fix an alphabet Σ , with $c \in \Sigma$. Let L be the set of all codes of Turing machines over Σ , which write a 'c' to the tape at some point for at least one input. For the purposes of this problem, reading a c from some tape cell without changing it does not count as "writing a c ." Show that L is Turing-recognizable.

That is, give a Turing machine which halts and accepts on an input w if and only if $w = \langle M \rangle$, where

- M is a Turing machine, and
- for some string $x \in \Sigma^*$, if you run M on input x , M writes c to the tape at some point.

For example, if $w = \langle M \rangle$ for some M , where M is a Turing machine which moves its head to the right indefinitely on all inputs, then $w \notin L$. If $w = \langle N \rangle$, where N is a Turing machine which writes a c at the 100th space on the tape if and only if the input is $cccc$, then $w \in L$.

Solution:

First, note that we can enumerate all strings over Σ . In particular, we can build a Turing machine that has the functionality to produce the i 'th string, given any i , such that this mapping covers all strings in Σ^* . For example, we can fix an ordering on Σ , and enumerate the strings in Σ^* according to the *lexicographic ordering* (see Sipser, page 14) induced by the order (e.g. the alphabetical order) on Σ .

For example, if $\Sigma = \{a, b, c\}$ and we choose $a < b < c$, then we enumerate the strings in Σ^* as $\epsilon, a, b, c, aa, ab, ac, aaa, aab, aac, aba, \dots$. We can build a TM that computes, for any i , the i 'th string in this sequence; the TM does this by consecutively computing the next string in the sequence i times. To advance from one string to the next string, it traverses the string from the right to left, looking for the first letter that is not c , and replaces it with the "next" letter, and replaces all the letters to its right with a 's. If the string contains only c 's, it changes all characters to a 's and adds an extra a at the end.

We are ready to build the Turing machine R that recognizes L . The idea is to build R that, for increasing values of n , tries running M on each of the first n strings in the ordering, for n steps. Note that for any string x , and any m , R will at eventually get to simulating M on x for at least m steps.

1. Input is $\langle M \rangle$.
2. If M is not a valid TM, reject.
3. $n := 1$;
4. $i := 1$;
5. do {
6. Compute the i 'th string w_i in the ordering.

7. Simulate M on w_i for n steps; if M writes c in this simulation, accept and halt.
8. Increment i ;
9. } while ($i \leq n$);
10. Increment n ; goto 4.

Note that for any value of n , the two loops 5–9 always terminates, and n gets incremented.

We claim that R recognizes L . First, it is easy to see that if R accepts $\langle M \rangle$, then it is the case that it has found some w such that M writes c onto its tape when working on some w , and hence R is correct in accepting $\langle M \rangle$.

Let us now assume that M writes c onto its tape when running on some input x . x must occur in the enumeration of strings— let x be the k 'th string. M must, on input x , write c in some finite number of steps, say m steps. Then, when R runs, it will run through increasing values of n till it reaches $n = \max(k, m)$. At this point, when $i = k$, it will simulate M on the k 'th string (i.e. x) for n steps, and hence will find that M writes to c at the m 'th step when running on x . Hence R will accept M .

We have shown that R accepts $\langle M \rangle$ iff M writes c onto its tape at some point. Hence R is a recognizer for L . Note that R is not a decider for L : given a TM M that never writes c any input, R will never halt.

2. Yet another reduction

Show that the following language is undecidable.

$$L_{prime} = \{\langle M \rangle \mid L(M) = \text{set of all prime numbers}\}$$

Prove this using a reduction. Do not simply invoke Rice's Theorem. Although this would be a correct proof, the point of this problem is to learn how to write reductions.

However, you should use the similarity with the Rice's Theorem proof to help you decide which language you should reduce from, and how the reduction could go.

Solution:

Let us reduce $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts } w\}$ to L_{prime} . In other strings, given a decider R for L_{prime} , let us build a decider D for A_{TM} .

The decider D for A_{TM} does the following:

1. Input is $\langle M, w \rangle$
2. Construct a new Turing machine $\langle N_{M,w} \rangle$ that does the following:
 - a. The input for $N_{M,w}$ is a string w .
 - b. Check if w is the encoding of a prime number (verify whether m divides n for any $2 \leq m \leq n - 1$, where n is the number represented by w).
 - c. If w is not the encoding of a prime number, then reject. Else simulate M on w , and accept if M accepts w .
3. Feed the Turing machine $\langle N_{M,w} \rangle$ to R , the decider for L_{prime} . If R accepts, then accept, else reject.

We claim that D decides A_{TM} . First D halts on every input, because it takes in $\langle M, w \rangle$, computes the TM $N_{M,w}$, and passes $N_{M,w}$ to the decider R , and accepts according to whether R accepts. Since R is a decider, D halts on all inputs.

Note that neither $N_{M,w}$ nor M is *simulated* by D ; $N_{M,w}$ simulates M on w , and hence $N_{M,w}$ may not halt, but D never simulates N or M on w . It simply constructs the code for $\langle N_{M,w} \rangle$.

Now, observe that for any M and w , if M accepts w , then the language of $N_{M,w}$ is the set of all prime numbers. This is because $N_{M,w}$ first checks if its input is a prime, and if it is, runs M on w , and accepts if M accepts w . On the other hand, if M does not accept w , then the language of $N_{M,w}$ is the empty set, as on any input, $N_{M,w}$ requires M to accept w in order to accept the input. Summarizing, the language of $N_{M,w}$ is the set of all prime numbers iff M accepts w .

Consequently, if M accepts w , when D feeds $N_{M,w}$ to R , it will get back the answer that $N_{M,w}$ does accept L_{prime} , whereas if M does not accept w , then D will get back the answer from R that $N_{M,w}$ does not accept L_{prime} . Hence D decides A_{TM} .

More succinctly,

D accepts $\langle M \rangle$

iff R accepts $\langle N_{M,w} \rangle$

iff $\langle N_{M,w} \rangle \in L_{prime}$

iff M accepts w .

Since A_{TM} reduces to L_{prime} , it follows that L_{prime} is undecidable. In other words, we have shown that if L_{prime} is decidable, then A_{TM} is decidable, and since we know that A_{TM} is undecidable, we can conclude that L_{prime} is undecidable.

3. The proof of Rice's Theorem

Let $L = \{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) \text{ is finite}\}$.

Look at the proof of Rice's theorem in Sipser (problem 5.28 and its solution) and/or in Lecture 23. The two proofs are basically the same, except that Sipser's set P is S in the lecture notes, and the hypothetical decider for P is R_P in Sipser and N in the notes. Let's use Sipser's names for this set and this decider.

For this problem, you will write a proof that L is undecidable, following as closely as possible the proof of Rice's theorem.

(a) Clearly (but briefly) describe what's in the set \overline{P} .

(b) Is the set T_\emptyset in P ? Are you going to write your proof using P directly, or by switching to \overline{P} ? (If you plan to switch to \overline{P} , answer (c) and (d) using \overline{P} as your set.)

(c) Pick a suitable value for the Turing machine T . Describe $L(T)$ and, in no more than two sentences, how T works.

(d) The input to R_P is the encoding of a Turing machine $\langle M \rangle$. Explain briefly in English which inputs $\langle M \rangle$ accepts.

(e) Write out the proof that L is undecidable. Keep the structure of the proof the same as in the proof of Rice's theorem, substituting specific values (e.g. for the set P) as needed. (Minor variations in wording don't matter.)

Solution:

(a) P is just the set L . So the set \overline{P} contains two types of strings: encodings of Turing machines whose languages are infinite and strings that aren't valid encodings of a Turing machine.

(b) The Turing machine T_\emptyset is in P , because its language is the empty set, which is finite. This means that the proof will have to be written using \overline{P} in place of P .

(c) T needs to be some specific Turing machine which is in \overline{P} . That means its language must be infinite. One possible value for T is the Turing machine that immediately goes into an accept state, without reading any of its inputs. This TM accepts all input strings and, thus, its language is infinite.

(d) This question was badly worded. Here's the answers to several possible interpretations of the question.

R_P accepts all Turing machines whose language is infinite. (Infinite rather than finite because we switched the \overline{P} .)

Or, you might have answered that if $\langle M \rangle$ is an input to R_P , it will be accepted if its language is infinite.

Or, you might have answered that the proof constructs a specific TM M_w which is fed to R_P . The language of M_w depends on what T you picked in step (c). For the choice of T given above, $L(M_w)$ is either the empty set or all strings, depending on whether M accepts w .

(e) To prove that L is undecidable, we will first show that \overline{L} is undecidable. So, suppose that \overline{L} is decidable and the TM R is a decider for it.

Let T_\emptyset be a Turing machine that rejects all inputs. The language of T_\emptyset is finite, so T_\emptyset is not in \bar{L} . Choose T to be some Turing machine that is in L . For example, let T be a Turing machine that enters the accept state without reading any of its input, thus accepting all strings in Σ^* .

We will now build a Turing machine S that decides A_{TM} . The code for S is:

Input is $\langle M, w \rangle$.

Reject if input does not contain a TM's code plus an input string.

Construct the code for a new TM M_w as follows:

Input is x .

Simulate M on w .

If M halts and rejects w , reject.

If M halts and accepts w , simulate T on w and accept if and only if T accepts.

(Or, equivalently, if M halts and accepts w , just accept, because that's what T does.)

Feed $\langle M_w \rangle$ to the decider R . Accept if and only if R accepts.

If M accepts w , the language of M_w is Σ^* , which is infinite, so R will accept $\langle M_w \rangle$. If M does not accept w , the language of M_w is the empty set, which is finite, so R will reject $\langle M_w \rangle$. Therefore, S is a decider for A_{TM} .

But we know that A_{TM} is not decidable. So we have a contradiction, which implies that we must have been wrong in our assumption that \bar{L} was decidable. This implies that L must not be decidable either, since the complement of a decidable language is always decidable.

4. Applying Rice's Theorem

For each of the problems below, determine whether the problem is undecidable as a direct corollary of Rice's theorem. If you think it is not, state precisely why Rice's Theorem cannot be applied. If you think it does follow from Rice's theorem, argue why, and especially argue why the non-triviality condition required for Rice's theorem holds. Keep your answers to the point and brief.

- (a) L is the set of encodings of Turing machines that compute whether a number is even.

Solution: Clearly, L is the set of Turing machines whose *language* satisfies a property, i.e. whenever $L(M_1) = L(M_2)$, $\langle M_1 \rangle \in L$ iff $\langle M_2 \rangle \in L$. Also, L expresses a nontrivial property since (a) there is a TM Q that accepts all numbers, and hence $\langle Q \rangle$ does not belong to L , and (b) there is a TM R that accepts only even numbers, and hence $\langle R \rangle$ belongs to L . So Rice's theorem does apply, and L is undecidable.

- (b) L is the set of encodings of Turing machines that read any tape position at most 3 times.

Solution: Clearly, L is *not* a language of Turing machines that solely depends on the language accepted by the Turing machine. For instance, we can easily build a TM M that checks whether an input word is of even length using a single pass over the input, and hence $\langle M \rangle \in L$. However, we can also build a TM N that checks the same property (that the input word is of even length) that does 4 passes of the input (with 3 extra dummy passes), and $\langle N \rangle \notin L$. In other words, there are two Turing machines M and N such that $L(M) = L(N)$, $\langle M \rangle \in L$, but $\langle N \rangle \notin L$. Hence Rice's theorem does not apply. (Note that we cannot conclude that the language is decidable! We have merely shown that Rice's theorem does not apply.)

- (c) L is the set of encodings of Turing machines that accept *all* its inputs.

Solution: Clearly, L is the set of Turing machines whose *language* satisfies a property, i.e. whenever $L(M_1) = L(M_2)$, $\langle M_1 \rangle \in L$ iff $\langle M_2 \rangle \in L$. Also, L expresses a nontrivial property since (a) there is a TM Q that accepts all its inputs, and hence $\langle Q \rangle$ belongs to L , and (b) there is a TM R that accepts no input, and hence $\langle R \rangle$ does not belong to L . So Rice's theorem does apply, and L is undecidable.

- (d) L is the set of encodings of Turing machines that decide A_{TM} (where A_{TM} is defined as in Sipser).

Solution: First, L is the set of Turing machines whose *language* satisfies a property, i.e. whenever $L(M_1) = L(M_2)$, $\langle M_1 \rangle \in L$ iff $\langle M_2 \rangle \in L$. Though at first sight it may appear that Rice's theorem applies, it does *not* because the nontrivial property requirement fails. The reason is that *no* TM encoding belongs to L , since there is no TM that can decide A_{TM} . Hence L is actually the empty set! So Rice's theorem does not apply, as L is trivial. In fact, L is decidable because it is the empty set.