

## Homework 5.5 (FAKE)

*Due: Never**Vinod Vaikuntanathan*

This “fake” homework is intended as a study guide covering the material in lectures 12 (Stack and counter machines), and 13, (Mapping reducibility and Rice’s Theorem). It also includes two problems to review part of the material in lecture 14: on uses of the Recursion Theorem.

**Readings:** Sipser, Sections 5.3 and 6.1; Hopcroft et al., Section 8.5

**Problem 1:** (Counter Machine Examples)

Informally but clearly describe counter machines that recognize the following languages. Try to use as few counters as you can. (You may assume that a special endmarker appears at the end of the input string.)

1.  $\{a^i b^j \mid i \leq j\}$ .

**Solution:** Use one counter  $C$  initialized to zero. As the input is read, keep track in the finite state control that it is of the form  $a^* b^*$ , if it is not, immediately reject. In addition, for each  $a$ , increment  $C$ . Then, for each  $b$ , if  $C = 0$ , accept; else decrement  $C$ . Once the endmarker is seen, accept if  $C = 0$ ; and reject otherwise. (Be careful that you handle the case where  $i = 0$ .)

2.  $\{a^i b^j c^k \mid i \leq j \leq k\}$ .

**Solution:** Use two counters  $C, D$  initialized to zero. We will also keep track of two bits  $check1, check2$  in the finite state control, initializing them to zero. As the input is read, keep track in the finite state control that it is of the form  $a^* b^* c^*$ , if it is not, immediately reject. For each  $a$ , increment  $C$ . For each  $b$ , increment  $D$  and if  $C = 0$ , set  $check1 = 1$ ; else decrement  $C$ . For each  $c$ , if  $D = 0$ , set  $check2 = 1$ ; else decrement  $D$ . Once the endmarker is seen, set  $check1 = 1$  if  $C \leq 0$ ; and set  $check2 = 1$  if  $D \leq 0$ . If  $check1 = check2 = 1$ , accept; otherwise reject.

3.  $\{a^i b^j c^k \mid k = ij\}$ .

**Solution:** Use two counters  $C, D$  initialized to zero. As the input is read, keep track in the finite state control that it is of the form  $a^* b^* c^*$ , if it is not, immediately reject. For each  $a$ , increment  $C$ . For each  $b$ , add the value in  $C$  to  $D$ . For each  $c$ , if  $D = 0$ , accept; else decrement  $D$ . Once the endmarker is seen, accept if  $D = 0$ ; and reject otherwise.

**Problem 2:** (Counter Machine Simulation of Stack Machine)

Suppose we want to design a counter machine to simulate a  $k$ -stack machine, using the same strategy discussed in lecture for simulating 2 stacks with 3 counters.

1. How many counters does this require? Explain why.

**Solution:** This requires  $k+1$  counters. The set of all symbols in the stack at any point of time can be represented by a single integer in the counter (the  $q$ -ary representation of the stack, as discussed in the lecture). This requires  $k$  counters. We require one additional counter as a “temp” to simulate the *push* and *pop* operations on the stack.

2. Simulating a single step of the  $k$ -stack machine requires that the CM (counter machine) emulate the reading of the top symbol of each stack, without removing that symbol from the stack. Describe clearly how the CM can read the top symbol of a stack.

**Solution:** Let  $q$  be the size of the stack alphabet. Lets say the symbols in the stack at any point of time are  $a_1 a_2 \dots a_k$ , where  $a_k$  is the topmost symbol on the stack. Then, as noted before, the stack is represented as a number  $a_1 q^{k-1} + a_2 q^{k-2} + \dots + a_k = A$  in a counter. Reading the topmost symbol is equivalent to computing  $A \bmod q$ , which can be done using a temp counter.

3. Before the CM can determine the stack machine transition to be emulated, it must know the top symbol of every stack. Describe how the CM can remember these symbols in a way that allows it to determine the next stack machine transition.

**Solution:** Compute the top of stack symbol  $a_k$  in the manner described above. Keep the value  $a_k$  in the temp counter. Next, emulate the stack machine transition as follows: (Note that we have used an encoding – a bijection – of the stack symbols by integers in  $\{0, \dots, q-1\}$  to represent the stack in a counter.) We check if the temp counter is zero. If it is, do what the stack machine does on input symbol corresponding to the integer zero. Otherwise decrement temp and keep repeating this process.

4. After the CM determines the stack machine transition to be simulated, it must modify the simulated stacks as specified by that transition. Describe clearly how the CM can do this. How can it keep track of its progress?

**Solution:** We need to show how the counter machine can push and pop symbols from the stack. To push a symbol  $s$  onto the stack, we do the following operation on the counter :  $C_i = C_{i-1} * q + s$ , where  $C_j$  refers to the value in the counter at time  $j$ . To pop a symbol from the stack, change the counter  $C_{i-1}$  as  $C_i = \lfloor \frac{C_{i-1}}{q} \rfloor$ .

5. How does the CM simulate the state change of the simulated stack machine?

**Solution:** The finite state control changes in the same way as the FSC of the stack machine changes. The counter is changed according to the operations described above. (This can be done using the temp counter).

**Problem 3:** (Mapping Reducibility)

Define *Total* to be the language  $\{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \{0, 1\}^*\}$ , that is, the set of representations of (basic) Turing machines that accept all strings of 0s and 1s. Use mapping reducibility to prove the following statements. Cite explicitly any theorems about mapping reducibility that you use.

1. *Total* is not Turing-recognizable.

**Proof:** We prove this showing  $A_{TM} \leq_m \overline{Total}$ , which is equivalent to  $\overline{A_{TM}} \leq_m Total$ , and applying Corollary 5.23. This is reduction will require some creativity.

To do so we present a computable function  $f$  that takes input of the form  $\langle M, w \rangle$  and returns output of the form  $\langle M' \rangle$  such that:

$$\langle M, w \rangle \in A_{TM} \text{ if and only if } \langle M' \rangle \in \overline{Total}$$

The following machine  $F$  computes a reduction  $f$ .

$F =$  "On input  $\langle M, w \rangle$ ,

1. Construct the machine  $M'$  as below.

2. Output  $\langle M' \rangle$ .”

$M'$  = “On input  $x$ ,

1. Run  $M$  on  $w$  for  $x$  steps.

2. If  $M$  accepts, reject; otherwise, accept.”

Now, if  $M$  accepts  $w$ , then  $M'$  must reject *some*  $x$ , i.e.,  $\langle M' \rangle \in \overline{Total}$ . However, if  $M$  rejects  $w$  or loops forever on  $w$ , then  $L(M') = \Sigma^*$ , i.e.,  $\langle M' \rangle \notin \overline{Total}$ . Thus, we conclude that  $Total$  is not Turing-recognizable. ■

2. The complement of  $Total$  is not Turing-recognizable.

**Proof:** We prove this showing  $A_{TM} \leq_m Total$ , which is equivalent to  $\overline{A_{TM}} \leq_m \overline{Total}$ , and applying Corollary 5.23 since we know that  $\overline{A_{TM}}$  is not Turing-recognizable. This is a fairly straight-forward reduction.

To do so we present a computable function  $f$  that takes input of the form  $\langle M, w \rangle$  and returns output of the form  $\langle M' \rangle$  such that:

$$\langle M, w \rangle \in A_{TM} \text{ if and only if } \langle M' \rangle \in Total$$

The following machine  $F$  computes a reduction  $f$ .

$F$  = “On input  $\langle M, w \rangle$ ,

1. Construct the machine  $M'$  as below.

2. Output  $\langle M' \rangle$ .”

$M'$  = “On input  $x$ ,

1. Run  $M$  on  $w$ .

2. If  $M$  accepts, accept; otherwise, reject.”

Now, if  $M$  accepts  $w$ , then  $L(M') = \Sigma^*$ , i.e.,  $\langle M' \rangle \in Total$ . However, if  $M$  rejects  $w$  or loops forever on  $w$ , then  $L(M') = \emptyset$ , i.e.,  $\langle M' \rangle \notin Total$ . Thus, we conclude that  $\overline{Total}$  is not Turing-recognizable. ■

**Problem 4:** (Rice’s Theorem Applications)

Which of the following are decidable and which are undecidable? Give proofs. Use Rice’s Theorem wherever possible, in showing undecidability.

1.  $\{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ has more than 30 states}\}$ .

**Decidable;** the number of states is a checkable property of any machine (similar to checking if a program has more than 30 lines of code).

2.  $\{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) \text{ is recognized by some Turing machine } M' \text{ with more than 30 states}\}$ .

**Decidable;** this is a trivial language property. All recognizable languages can be recognized by a TM with at least 30 *dummy* states added to it.

3.  $\{\langle M \rangle \mid M \text{ is a Turing machine that accepts every word of the form } 0^n 1^n, n \geq 0 \text{ (and possibly other words)}\}$ .

**Undecidable;** Rice’s Theorem applies. We know there are languages without the property above (e.g., a TM that recognizes  $\emptyset$  does not have the property of accepting all palindromes), and languages with the property above (e.g., a TM that recognizes  $\Sigma^*$  accepts all strings of the form  $0^n 1^n$ ).

4.  $\{\langle M, N \rangle \mid M \text{ and } N \text{ are Turing machines and } L(M) \subseteq L(N) \cup 0^*\}$ , where  $A = 0^*$ , the set of all finite strings consisting of just zeros.

**Undecidable;** The modified form of Rice's Theorem (from Problem 5 below) applies. Let  $P(\langle M \rangle, \langle N \rangle)$  be the true if and only if  $L(M) \subseteq L(N) \cup 0^*$ . Consider a TM  $M_1$  that recognizes the empty language, and let  $N_1$  be an arbitrary Turing machine. Then  $P(M_1, N_1)$  is true. If  $M'_1$  is instead the Turing machine that recognizes  $\Sigma^*$ , then  $P(M'_1, N_1)$  is true. Thus  $P$  is a non-trivial property and we can apply Rice's theorem.

**Problem 5:** (Generalization of Rice's Theorem)

Consider the following generalization of Rice's Theorem:

If  $P_2$  is a non-trivial property of pairs of recognizable languages, then

$$A_{P_2} = \{\langle M, N \rangle \mid M \text{ and } N \text{ are Turing Machines and } P_2(L(M), L(N)) = TRUE\}$$

is undecidable.

(Note:  $P_2$  is defined to be *non-trivial* if there exist Turing Machines  $M_1, M_2, N_1$  and  $N_2$  such that  $P_2(L(M_1), L(N_1)) = FALSE$  and  $P_2(L(M_2), L(N_2)) = TRUE$ ).

1. Prove this theorem.

**Solution 5:** We show this by modifying the proof of Rice's Theorem. If  $P_2(\emptyset, \emptyset) = FALSE$  (where  $\emptyset$  denotes the empty language), we will show that  $A_{P_2}$  is undecidable by showing that it is not even co-recognizable via a mapping-reduction from  $A_{TM}$ . If  $P_2(\emptyset, \emptyset) = TRUE$ , we could show a similar reduction from  $A_{TM}$  to  $\overline{A_{P_2}}$ , establishing that  $A_{P_2}$  is not recognizable, which also proves that  $A_{P_2}$  is undecidable. Thus, w.l.o.g., it suffices to give the first case.

We show how to reduce  $A_{TM} \leq_m A_{P_2}$ .

Let  $M_1, N_1$  be some TMs such that they have the property  $P_2$ , i.e.,  $\langle M_1, N_1 \rangle \in A_{P_2}$ . Given our assumption above, we know that  $(\langle reject \rangle, \langle reject \rangle) \notin A_{P_2}$ , where  $\langle reject \rangle$  denotes the description of a TM that rejects all inputs.

Here is our computable function  $f(x)$  such that:

$$\langle Q, w \rangle \in A_{TM} \text{ if and only if } \langle M', N' \rangle \in A_{P_2}$$

For completeness, we will say that if  $x$  is not of the form  $\langle Q, w \rangle$ ,  $f$  returns something that isn't in  $A_{P_2}$ , e.g.  $(\langle reject \rangle, \langle reject \rangle)$ . If  $x$  is of the form  $\langle Q, w \rangle$ , then  $f$  returns  $\langle M', N' \rangle$  as described below.

$M'$  = "On input  $y$ ,

1. Run  $Q$  on  $w$ .
2. If  $Q$  accepts  $w$ , then run  $M_1$  on  $y$ .
3. If  $M_1$  accepts  $y$ , then accept; otherwise, loop forever."

$N'$  = "On input  $z$ ,

1. Run  $Q$  on  $w$ .
2. If  $Q$  accepts  $w$ , then run  $N_1$  on  $z$ .
3. If  $N_1$  accepts  $z$ , then accept; otherwise, loop forever."

Thus, if  $\langle Q, w \rangle \in A_{TM}$ , then  $L(M') = L(M_1)$  and  $L(N') = L(N_1)$ , which have property  $P_2$ . Similarly, if  $\langle Q, w \rangle \notin A_{TM}$ , then  $L(M') = \emptyset$  and  $L(N') = \emptyset$ , which does not have property  $P_2$ . We conclude that  $A_{P_2}$  is undecidable.

2. Apply this theorem to show that it is undecidable whether, for given Turing machines  $M$  and  $N$ , the language recognized by  $M$  is a proper subset of the language recognized by  $N$ . Indicate carefully why all required hypotheses are satisfied.

**Solution 5:** Let  $P2$  be the language property in question. First of all, this is a language property. Secondly, this is non-trivial. This is because, if  $M$  is a TM that recognizes the empty language and  $N$  is a TM that recognizes  $\Sigma^*$ , then clearly  $(M, N)$  has property  $P2$ . On the other hand,  $(N, M)$  does not have property  $P2$ . Therefore,  $P2$  is non-trivial, and we can apply the modified form of Rice's theorem as proved above.

**Problem 6:** (Using the Recursion Theorem to prove undecidability)

Let

$$A01 = \{\langle M \rangle \mid M \text{ is a Turing machine and } M \text{ accepts the string } 01\}$$

be the language of descriptions of machines accepting the string 01. We already know how to prove that  $A01$  is undecidable using mapping reducibility or using Rice's Theorem. Now give yet another proof that  $A01$  is undecidable, by using the Recursion Theorem.

**Solution 6:** Assume for contradiction that  $A01$  was decidable and let  $D$  be the Turing machine that decides it. Now, consider the following TM  $P$ .

$P =$  "On input  $w$ ,

1. Obtain, via the recursion theorem, own description  $\langle P \rangle$ .
2. Run  $D$  on  $\langle P \rangle$ .
3. If  $D$  accepts, reject; otherwise accept."

Here, we see that  $D$  cannot exist, because it will not answer correctly on input  $\langle P \rangle$ . Consider the  $P$  accepts 01 if and only if  $D$  says that it won't; and  $P$  rejects 01 if and only if  $D$  says that it will accept 01.

**Problem 7:** (Using the Recursion Theorem to prove non-enumerability) Let's define a Turing machine  $M$  to be *pretty good* if the length of the representation of every Turing machine  $M'$  that recognizes the same language as  $M$  is at least the square root of the length of the representation of  $M$ . (That is, for every  $M'$  such that  $L(M') = L(M)$ ,  $|\langle M' \rangle| \geq \sqrt{|\langle M \rangle|}$ ). Prove that there is no enumerator that outputs the set of representations of pretty-good Turing machines.

**Solution 7:** Assume for contradiction that there is an enumerator  $E$  that outputs representations of pretty good TMs. Construct a machine  $P$  as follows:

$P =$  "On input  $w$ ,

1. Obtain, via the recursion theorem, own description  $\langle P \rangle$ .
2. Run  $E$  till it outputs the description of a machine  $\langle M' \rangle$  such that  $|\langle M' \rangle| > |\langle M \rangle|^2$ .
3. Run the machine  $M'$  on input  $w$ . Accept if this computation accepts, and rejects if this rejects."

Lets see how this leads to a contradiction.  $L(P) = L(M')$ , but  $|\langle P \rangle| < \sqrt{|\langle M' \rangle|}$ . Therefore,  $M'$  is not pretty good, even though the enumerator outputs it. This is a contradiction.

**Problem 8:** Prove that every language that is Turing-recognizable can be mapping-reduced to the canonical Turing machine acceptance problem,  $A_{TM}$ .

**Solution 8:** Suppose a language  $L$  is Turing recognizable. This means, straight from the definition, that there exists a Turing machine  $M$  that recognizes  $L$ . The function  $f$  (for the mapping reducibility) is defined as follows:

$f$  on input  $x$ , outputs  $\langle M, x \rangle$ .

It is easy to see that if  $x \in L$ , then  $M$  accepts  $x$  (this is by definition of  $M$ ), and therefore  $\langle M, x \rangle \in A_{TM}$ . On the other hand, if  $x \notin L$ , then  $M$  does not accept  $x$  and  $\langle M, x \rangle \notin A_{TM}$ .