

Notes

The range of  $f$  is  $L = \{ \Delta \Gamma^n \mid f(m) = n \text{ for some } m \}$ .  
 Let  $F$  be a Turing machine that computes  $f$ .  
 Consider the following three tape Turing machine  $T$  running on  $w$ , leaves  $w$  on the first tape. On the second tape  $T$  begins with zero. Then  $T$  copies the content of the second tape to the third tape and runs  $F$  on the third tape. After  $F$  completes, the output of  $F$  is compared to  $w$  on tape one, if it matches,  $T$  accepts. Otherwise,  $T$  increments the second tape, copies it to the third, and runs  $F$  again. It keeps repeating this.

In general,  $T$  is a Turing-accepter for  $L$ . If  $w$  is in  $L$ , eventually  $T$  will run  $F$  on some  $m$  and find that  $f(m) = w$ . In general,  $T$  will keep looking for some  $m$  such that  $f(m) = w$  forever.

But in this case we can do better. Since  $f$  is strictly increasing, we know that  $f(m) > w$ , we can stop since any  $x > m$  will have  $f(x) > f(m) > w$ . Thus we add a check to  $T$  that sees if the result of  $F$  is greater than  $w$  and reject if it is. This new machine stops even when  $w \notin L$ . Thus  $L$  is Turing-decidable.

10.30

2

10pts

ⓑ To find a contradiction, suppose that  $S$  is countable. Then let  $s_0, s_1, s_2, \dots$  be  $S$ . Let  $a_{ij}$  be the  $j$ th digit of  $s_i$ , i.e.

$$s_i = a_{i0} a_{i1} a_{i2} \dots$$

Now consider the table

$$\begin{array}{l}
 s_0 = \boxed{a_{00}} a_{01} a_{02} a_{03} a_{04} \dots \\
 s_1 = a_{10} \boxed{a_{11}} a_{12} a_{13} a_{14} \dots \\
 s_2 = a_{20} a_{21} \boxed{a_{22}} a_{23} a_{24} \dots \\
 s_3 = a_{30} a_{31} a_{32} \boxed{a_{33}} a_{34} \dots \\
 s_4 = a_{40} a_{41} a_{42} a_{43} \boxed{a_{44}} \dots \\
 \vdots
 \end{array}$$

By hypothesis, every infinite sequence of 0's and 1's is in that table. But consider the sequence

$$\Gamma = x_0 x_1 x_2 x_3 x_4 \dots$$

where

$$x_j = \begin{cases} 0 & \text{if } a_{jj} = 1 \\ 1 & \text{if } a_{jj} = 0. \end{cases}$$

Consider  $\Gamma$  and some  $s_n$ . By construction  $\Gamma$  and  $s_n$  differ in at least the  $n$ th position. Thus  $\Gamma \neq s_n$ . So,  $\Gamma$  is not included in the above table. This is a contradiction, thus  $S$  is not countable.

Note that we can think of each  $s_n$  encoding a subset of a countably infinite set. If  $a_{nj} = 1$ , the  $j$ th member is in the subset encoded by  $s_n$ . This is how the above proof is related to Theorem 10.15.

10.31

WPTS

3

① We present two ways to prove that  $S$ , the set of all finite subsets of  $\mathbb{N}$  is countable.

Let  $S_n$  be set of all subsets of  $\mathbb{N}$  with at most  $n$  elements. We prove by induction on  $n$  that each  $S_n$  is countable.  $S_1$  is countable because it is basically  $\mathbb{N}$ . Now suppose  $S_{k-1}$  is countable. Then,  $S_k$  can be built from elements of  $S_{k-1}$  by adding a new number. In other words

$$S_k = \bigcup_{m \in \mathbb{N}} \bigcup_{A \in S_{k-1}} (A \cup \{m\}),$$

which is a countable union of countable sets (since  $S_{k-1}$  is countable by the induction hypothesis). Thus each  $S_n$  is countable by Theorem 10.13. Now we can apply Theorem 10.13 again to

$$S = \bigcup_{n \in \mathbb{N}} S_n$$

get the desired result.

Another way is to let  $S_n$  be the set of subsets of  $\mathbb{N}$  whose largest element is at most  $n$ . Each  $S_n$  is finite and thus countable. Now

$$S = \bigcup_{n \in \mathbb{N}} S_n$$

which is a countable union of finite (and thus countable) sets and is thus countable.

10.31

4

QPTS

⑤ The set of all functions  $\mathbb{N} \rightarrow \mathbb{N}$  is a subset set of the set of all functions from  $\mathbb{N}$  to  $\{0,1\}$ . Since the latter is uncountable, it follows that the former is also uncountable.

(Note: this uses the contrapositive of theorem 10.13).

⑥ There are several ways of showing that  $R$ , the set of regular languages, is countable over  $\{0,1\}$ .

QPTS

For each regular language there is a Turing machine that accepts. Since the set of all Turing machines is countable, it follows that  $R$  is countable.

For each regular language there is a corresponding regular expression. All regular expressions are subsets of strings over  $\{0,1,(,),+,*\}$  which is countable by Example 10.7. Thus  $R$  is countable.

We can also show that  $R$  is countable by showing that the set of DFAs is countable. Let  $D_n$  be the set of DFAs with  $n$  states.  $D_n$  is finite. In fact we can even calculate its size:

$$\begin{aligned}
 D_n &= (\text{possible ways of setting accepting states}) \\
 &\quad \times (\text{for each state, possible transitions on } 0) \\
 &\quad \times (\text{possible transitions on } 1) \\
 &= 2^n n n = 2^n n^2
 \end{aligned}$$

Now the set of all DFAs is  $\bigcup_{n \in \mathbb{N}} D_n$  which is countable since it is the union of a countable number of finite sets.

Q.34

20 pts

5

To find a contradiction, suppose that  $T$  only loops on the strings  $x_1, x_2, \dots, x_n$ . We will show that there exists a Turing machine that decides  $L$  and thus  $L$  is Turing-decidable. This is a contradiction, thus  $T$  must loop on infinitely many strings.

The Turing machine  $T'$  that decides  $L$  will be constructed from  $T$  as follows: First,  $T'$  will check to see if its input is one of the bad words  $x_1, x_2, \dots, x_n$ . There are finitely many such words so  $T'$  can encode them all in finite state. If the input is any of the bad words,  $T'$  clears the tape and then halts with  $\Delta 0$  on the tape. If the input isn't one of the bad words,  $T'$  simulates  $T$ . If  $T$  (crashes, rejects, or tries to run off the tape)  $T'$  halts with  $\Delta 1$  on the tape. If  $T$  accepts,  $T'$  halts with  $\Delta 1$ . The simulation of  $T$  is guaranteed to halt because it isn't running on a bad word. Thus  $T'$  computes the characteristic function of  $L$ . Thus  $L$  is Turing-decidable.

We can detect crashes by adding missing transitions to  $T$  that "catch" a crash.

We can detect when  $T$  runs off the tape by marking the left edge with a special symbol that will cause  $T$  to crash instead. We can detect crashes as above.

For concatenation, let  $L_1$  and  $L_2$  be the languages, and  $T_1$  and  $T_2$  the Turing machines that accept  $L_1$  and  $L_2$  respectively. The following Turing machine  $T$  accepts  $L_1 L_2$ :  $T$  has two tapes. Given  $w$ , it nondeterministically cuts and pastes some prefix of  $w$  onto the second tape. It then runs  $T_1$  on the second tape. If  $T_1$  crashes, rejects. We can do the same. If  $T_1$  accepts, clear the 2nd tape, copy the remainder of  $w$  to it, and run  $T_2$  and accept if  $T_2$  accepts.

Note that unlike for unions, it is ok if  $T_1$  loops forever and  $T_2$  is never run. This is because both  $T_1$  and  $T_2$  must accept for  $w$  to be in the concatenation.

If  $w \in L_1 L_2$ , there is some  $x, y$  such that  $w = xy$  and  $x \in L_1$  and  $y \in L_2$ . This some  $T$  computation will run  $T_1$  on  $x$  and then  $T_2$  on  $y$  and both will accept, thus  $T$  will accept  $w$ . If  $T$  accepts a word  $w$ , then  $T$  found an  $x \in L_1$  and a  $y \in L_2$  such that  $w = xy$ . Thus  $w \in L_1 L_2$ .

For Kleen\*, we use the same logic but add a loop.  $T$  will cut and paste some nondeterministically chosen prefix of  $w$  to the second tape and then  $T_1$ . If  $T_1$  accepts, clear the second tape, cut and paste a prefix of what's left of  $w$  and run  $T_1$  again. Keep repeating this until nothing of  $w$  is left. If all the runs of  $T_1$  accepted,  $T$  accepts. Thus  $T$  accepts the language  $L_1^*$ .

11.5 10 pts

7

Consider the following two tape Turing machine  $T$ .  $T$  enumerates elements of  $\mathbb{N}^4$ . I.e.  $T$  generates a stream of  $(x, y, z, n) \in \mathbb{N}^4$ . When it gets a new  $(x, y, z, n)$  it checks that  $n > 2$  and if  $x^n + y^n = z^n$ . If both are true,  $T$  halts. If they aren't true,  $T$  generates the next 4-tuple and checks that and so on.

If we could solve the halting problem we could answer the question "does  $T$  halt?" If  $T$  halts, then it found an  $x, y, z$  and  $n > 2$  such that  $x^n + y^n = z^n$  which would be a counter-example to Fermat's Last Theorem. If  $T$  never halts, then no such example exists and Fermat's Last Theorem is true.