

Solution Set 3

**Problem 1a** We want to come up with a decision algorithm for the decision property  $P : |L| \geq 100$ .

$|L| \geq 100$  can be split into two cases:

1.  $|L|$  is infinite.
2.  $|L|$  is finite, but  $\geq 100$ .

If either case 1 or 2 hold, then the decision algorithm returns YES, else it returns NO.

Case 1: Theorem:  $L$  is infinite iff there exists a string  $w \in L$  such that  $n \leq |w| < 2n$ , where  $n$  is the number of states in the DFA for  $L$ .

Proof: ( $\leftarrow$ ) Suppose we are given a string  $w \in L$  such that  $n \leq |w| < 2n$ . By the pumping lemma,  $w = xyz$ ,  $|y| > 0$  such that for all  $k \geq 0$ ,  $xy^kz$  is in  $L$ . Since there are an infinite number of possible values for  $k$ , there are an infinite number of strings in  $L$ . Hence,  $L$  is infinite.

( $\rightarrow$ ) We are given that  $L$  is infinite. This implies that there exists at least one string  $w \in L$  such that  $|w| \geq n$ . (If every string in  $L$  had a length  $< n$ , there would only be a finite number of them, which we could easily enumerate. But  $L$  is given to be infinite.)

Let  $w'$  be the shortest string in  $L$  of length  $\geq n$ .

Claim:  $|w'| < 2n$ . Proof by contradiction: Assume that  $|w'| \geq 2n$ . By the pumping lemma,  $w' = xyz$ , with  $|xy| \leq n$  and  $|y| > 0$ . Since  $|y| \leq n$ , if we choose a value of 0 for  $k$ , then  $xy^kz = xz$ , whose length must be  $\geq n$  (since the eliminated piece  $y$  had a length  $\leq n$ ). Also,  $xz \in L$ . But in that case,  $w'$  could not be the shortest string in  $L$  with length  $\geq n$ . Contradiction!

Therefore,  $n \leq |w'| < 2n$ .

So in order to check if  $L$  is infinite, we can enumerate all the strings with lengths in the range  $[n, 2n)$  and test if any of them is in  $L$ . If we do not get even one string in  $L$ , then  $L$  is finite. Else  $L$  is infinite.

If  $L$  is infinite, then answer to our decision problem is YES. If  $L$  is finite, then the answer depends on the outcome of case 2.

Case 2: Test if  $|L| \geq 100$ . If  $L$  is finite, then we know from the above proof that all strings in  $L$  must necessarily have lengths  $< n$ . We can enumerate all strings with lengths in the range  $[0, n)$ , since there are only a finite number of them. We test each enumerated string to see if it is in  $L$ , and count the number of strings that belong to  $L$ . If, after the enumeration is over, we counted  $\geq 100$  strings in  $L$ , then the answer to our decision problem is YES. Else the answer is NO.

Alternate method: We can also run a depth-first search on the DFA for  $L$ , and enumerate all the paths from the starting state to some final state of  $L$ . If any of those paths has a loop in it, the loop can be pumped, yielding an infinite number of strings in  $L$ . (making the answer YES)

If none of the paths has a loop, then we count the number of paths we found. If that number is  $\geq 100$ , answer YES, else answer NO.

**Problem 1b** Let  $L_1 =$  Language of all strings  $w$  with no occurrences of the pattern 00 and 11 in them. It is easy to come up with a DFA for  $L_1$ .

We want to come up with a decision algorithm for the decision property  $P$ : Does  $L$  contain any string from  $L_1$ ?

which is equivalent to asking "Is  $L \cap L_1 \neq \emptyset$ "?

Applying the product construction to get the intersection of the two languages, we then test the resulting language for the property of emptiness, as discussed in the class.

**Problem 2** The product construction remains the same as in the case for intersection of two input DFAs, except for the definition of a final state.

Let  $D_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$  and  $D_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ . The product construction for the union of  $D_1$  and  $D_2$  is  $D_3 = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F)$ , where  $(q_i, q_j) \in F$  iff  $q_i \in F_1$  or  $q_j \in F_2$ , and  $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$ .

We can use induction on the length of the input strings to prove that the DFA resulting from this construction indeed accepts the union of  $L(D_1)$  and  $L(D_2)$ . The proof is similar to the intersection case.

First observe by induction on  $|w|$  that  $\hat{\delta}((s_1, s_2), w) = (\hat{\delta}_1(s_1, w), \hat{\delta}_2(s_2, w))$ . But, by construction,  $D_3$  accepts  $w$  iff  $\hat{\delta}((s_1, s_2), w)$  is a pair of states where at least one of the states in the pair is an accepting state in  $D_1$  or  $D_2$ . In other words,  $\hat{\delta}_1(s_1, w) \in F_1$  or  $\hat{\delta}_2(s_2, w) \in F_2$ . It follows that  $w$  is accepted by  $D_3$  iff it is accepted by either  $D_1$  or  $D_2$  or both.

**Problem 3** First, we fill out the table of distinguished states as follows.

B	x								
C	x	x							
D	x	x	x						
E	x	x		x					
F	x	x	x		x				
G	x	x	x	x	x	x			
H	x	x	x	x	x	x	x		
I	x		x	x	x	x	x	x	
J	x	x	x	x	x	x	x	x	x
	A	B	C	D	E	F	G	H	I

This table leads to the following minimum equivalent DFA. The name of each state in the minimized DFA indicates the set of equivalent states in the original DFA.

	0	1
$\rightarrow \{A\}$	$\{B, I\}$	$\{J\}$
$\{B, I\}$	$\{H\}$	$\{C, E\}$
$*\{C, E\}$	$\{D, F\}$	$\{G\}$
$\{D, F\}$	$\{C, E\}$	$\{C, E\}$
$*\{G\}$	$\{J\}$	$\{C, E\}$
$\{H\}$	$\{B, I\}$	$\{G\}$
$\{J\}$	$\{H\}$	$\{A\}$

#### Problem 4

(a) The Turing machine is as usual, with the following transitions:

$$\begin{aligned}\delta(q_0, 1) &= (q_1, 0, L) & \delta(q_0, 0) &= (q_0, 1, L) \\ \delta(q_1, \$) &= (q_2, \$, R) & \delta(q_2, 0) &= (q_3, \$, R) \\ \delta(q_1, 1) &= (q_3, 1, R) & \delta(q_1, 0) &= (q_3, 0, R) \\ \delta(q_3, 0) &= (q_3, 0, R) & \delta(q_3, 1) &= (q_3, 1, R) \\ \delta(q_3, B) &= (q_f, B, L)\end{aligned}$$

- $q_0$ : initial state, keep going left, flipping all 0s to 1s, until it hits a 1. At this point, it flips the 1 to 0 and enters state  $q_1$  (going left).
- $q_1$ : checks that the flipped 1 was not leading. If it was, enter state  $q_2$  (going right), otherwise enter state  $q_3$  (going right).
- $q_2$ : turns new leading 0 into \$, enter state  $q_3$  (going right).
- $q_3$ : keeps going right until hits the end (Blank). Then enters  $q_f$  while doing a left move.
- $q_f$ : final state.

(b)  $\$101q_00 \vdash \$10q_011 \vdash \$1q_1001 \vdash \$10q_301 \vdash \$100q_31 \vdash \$1001q_3B \vdash \$100q_f1$ .

**Problem 5** In both cases, the modified Turing machines accept the class of recursively enumerable languages (so each of these variants has the same expressive power as the standard Turing machine).

(a) Any language accepted by a Turing machine that does not write the symbol that it reads on a transition can also be accepted by a standard Turing machine, as a Turing machine that does not write the same symbol it reads is a special case of a standard Turing machine.

On the other hand, given a standard Turing machine  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , we can convert  $M$  to a Turing machine  $M' = (Q, \Sigma, \Gamma', \delta', q_0, B, F)$  that does not write the same symbol it reads, and accepts the same language as  $M$ . For each tape symbol  $X \in \Gamma$ , we introduce a new symbol  $X' \notin \Gamma$  into the tape alphabet of  $M'$ , so that  $\Gamma'$  contains  $X$  and  $X'$  for all  $X \in \Gamma$  (a total of  $2|\Gamma|$  symbols). From the perspective of the transition function of  $M'$ , the symbol  $X'$  will be interchangeable with the symbol  $X$ .

The transition function  $\delta'$  of  $M'$  has all the transitions of  $M$  that do not read and write the same symbol. For each transition  $\delta(q, X) = (p, X, D)$  in  $M$ , where  $D \in \{L, R\}$ , we construct a corresponding transition  $\delta'(q, X) = (p, X', D)$  in  $M'$ . In addition, for every state  $q \in Q$  and every tape symbol  $X \in \Gamma$  such that  $M$  has a transition  $\delta(q, X) = (p, Y, D)$  (where  $Y \in \Gamma$  may be the same as  $X$ ),  $M'$  has the additional transition  $\delta'(q, X') = (p, Y, D)$ .

An inductive argument can be used to show that the IDs reached by TM  $M'$  on a given input string are the same as those reached by  $M$  on the same input, with the exception that some symbols  $X \in \Gamma$  on the tape in the ID for  $M$  may be replaced with the new symbols  $X'$  in the ID for  $M'$ . It follows that  $M'$  enters a final state on an input string if and only if  $M$  does as well, and so  $L(M') = L(M)$ . As  $M'$  does not write the symbol it reads on a transition, for every standard Turing machine  $M$ , there is a Turing machine that does not write the same symbol it reads, and accepts the same language as  $M$ .

- (b) Any language accepted by a standard Turing machine can also be accepted by a Turing machine with two-cell transitions, since a standard TM can be considered a TM with two-cell transitions that only uses one-cell transitions.

Now, given a Turing machine  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  that has two-cell transitions, we construct a standard Turing machine  $M' = (Q', \Sigma, \Gamma, \delta', q_0, B, F)$  such that  $L(M') = L(M)$ . For each state  $q \in Q$  and tape symbol  $X \in \Gamma$ , we construct two new states  $q_X^L$  and  $q_X^R$ . The set of states of the TM  $M'$  is  $Q' = Q \cup (\cup_{q \in Q} \cup_{X \in \Gamma} \{q_X^L, q_X^R\})$ . For each transition  $\delta(q, X) = (p, Y_1, Y_2, D_1)$  in  $M$ , where  $D \in \{L, R\}$ , we put the corresponding transition  $\delta'(q, X) = (p, Y_1, D)$  in  $M'$ .

To handle two-cell transitions in  $M$ , we first create, for each state  $q \in Q$  and pair of tape symbols  $X, Y \in \Gamma$ , the transitions  $\delta'(q_X^L, Y) = (q, X, L)$  and  $\delta'(q_X^R, Y) = (q, X, R)$ . For each two-cell transition  $\delta(q, X) = (p, Y_1, Y_2, D_2)$  in  $M$ , where  $D \in \{L, R\}$ , we create the transition  $\delta'(q, X) = (p_{Y_2}^D, Y_1, D)$  in  $M'$ . So a two-cell transition  $\delta(q, X) = (p, Y_1, Y_2, D_2)$  in the TM  $M$  is translated into two consecutive one-cell transitions in the standard Turing machine  $M'$ :  $\delta'(q, X) = (p_{Y_2}^D, Y_1, D)$  followed by  $\delta'(p_{Y_2}^D, Z) = (p, Y_2, D)$  (where  $Z$  is any tape symbol).

As in the previous part, an inductive argument can be used to show that the IDs reached by TM  $M$  on a given input string are also reached by  $M'$  on the same input ( $M'$  will go through more IDs than  $M$ , because it will use two transitions to simulate one two-cell transition of  $M$ ). So, the set of strings accepted by  $M'$  is the same as the set of strings accepted by  $M$ . For each Turing machine  $M$  with two-cell transitions, then, there is a standard Turing machine that accepts the same language as  $M$ .