COMS 331: Theory of Computing, Spring 2023
Homework Assignment 6

Neha Maddali
Due at 10:00PM, Wednesday, March 15, on Gradescope.

Problem 37.
Let A be the language defined as $\{0^m1^n|\text{m is even or } m > n\}$. We know that a $\equiv_A$ b if and only if $\forall z \in \Sigma^*[az \in A \iff bz \in A]$. Say we have two strings $a = 0^x1$ and $b = 0^y1$, where x and y are odd and $x > y$. Then, we can append string $z = 1^{x-2}$ to the end of each to get 0^x1^{x-1} and 0^y1^{x-1} respectively. If these new strings are both in A, then they are in the same equivalence class by the definition stated at the beginning. However, $y \leq x-1$ because $x > y$, so 0^y1^{x-1} is not in the language A. It is evident that 0^x1^{x-1} is in the language A, which means that it is in a different equivalence class than 0^y1^{x-1}. Thus, there are infinite equivalence classes, and by the Myhill-Nerode theorem, A is not regular. ■

Problem 38.
Let A be the language defined as $\{0^k1^m0^n|n = k+m\}$. When looking at the first A-extension of x ($A_x^{(1)}$), we can see that if $x = 0^k1$, the first extension is 0^{k+1}. This is true because k=k, m=1, and n=k+1, so $0^k10^{k+1} \in A$. Thus, $|A^{(1)}| = \infty$ because k (in the first extension 0^{k+1}) is infinite, so A has infinite ordinal extensions, proving that A is not regular. ■

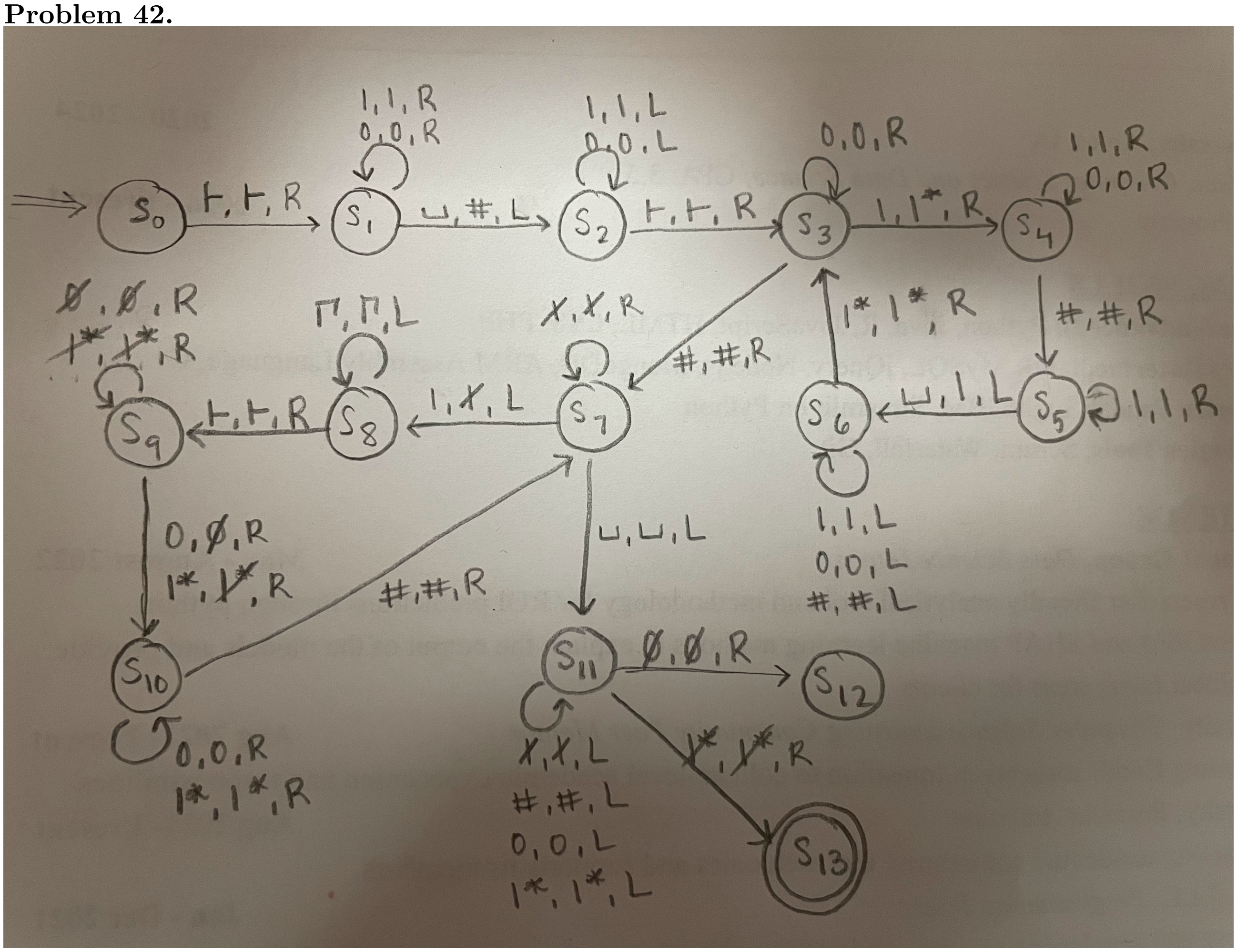
Problem 39.
Let A be the language defined as $\{x \in \{0,1\}^* || x| \text{ is a perfect square}\}$. When looking at the first A-extension of x ($A_x^{(1)}$), we can see that if $x = 0^{n^2-n}$, the first extension is 0^n. This is true because $0^{n^2-n}0^n = 0^{n^2}$, and $0^{n^2} \in A$ because $|0^{n^2}| = n^2$, which is a perfect square. Thus, $|A^{(1)}| = \infty$ because n is infinite, so A has infinite ordinal extensions, proving that A is not regular. ■

Problem 40.
Let A be the language defined as $\{0^m1^n|gcd(m,n) = 1\}$. We know that x $\equiv_A$ y if and only if $\forall z \in \Sigma^*[xz \in A \iff yz \in A]$. Say we have two strings $x = 0^p1$ and $y = 0^q1$, where p and q are different prime numbers. Then, we can append string $z = 1^{p-1}$ to the end of each to get 0^p1^p and 0^q1^p respectively. If these new strings are both in A, then they are in the same equivalence class by the definition stated at the beginning. However, gcd(p,p)=p, and $p \neq 1$, so 0^p1^p is not in the language A. It is evident that 0^q1^p is in the language A because the gcd or two prime numbers is 1, which means that it is in a different equivalence class than 0^p1^p. Thus, there are infinite equivalence classes, and by the Myhill-Nerode Theorem, A is not regular. ■

Problem 41.
Let A be the language defined as $\{xx|x \in \{0,1\}^*\}$. When looking at the first A-extension of x ($A_x^{(1)}$), we can see that if $x = 0^n1$, the first extension is 0^n1. This is true because $0^n10^n1 = xx$, and

$xx \in A$. If x=000, for example, the first extension would be 0, not 000. Thus, $|A^{(1)}| = \infty$ because n (in x and the first extension 0^n1) is infinite, so A has infinite ordinal extensions, proving that A is not regular. ■

Problem 42.



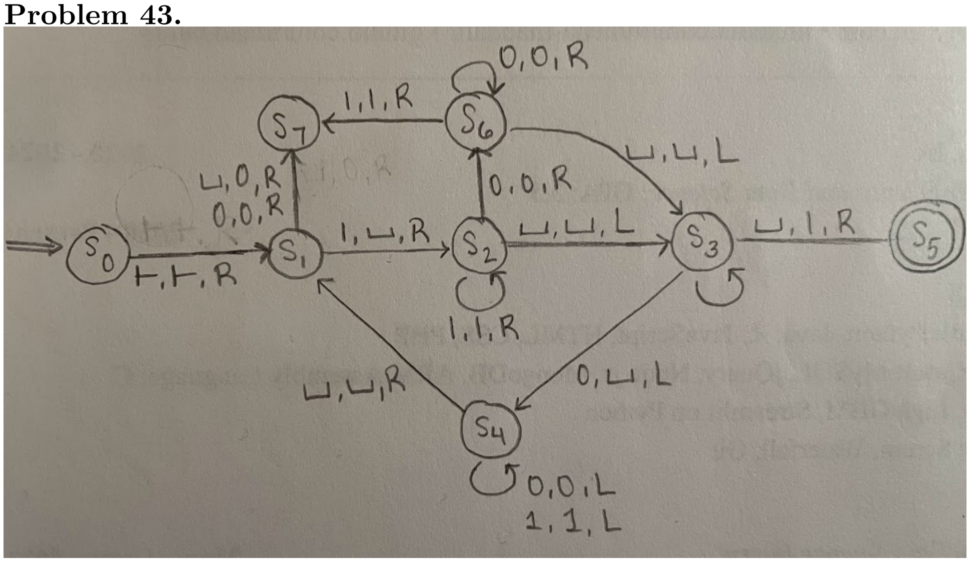
$Q = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, \vdash, B, \not{1}, 1^*, \not{1}^*, \emptyset, \#\}$ (B = blank, shown as underscore in diagram)
$\vdash = \vdash$
B = B
s = s_0
t = s_{13}
r = s_{12}
δ = (see transition diagram above)

Idea: Starting at the beginning of the string, iterate through the input string until you find a blank, then change it into a #. Then, go back to the start. Iterate through the string again, but for every 1 that is found, change it into a 1^* and add a 1 after the #. The 1s after the # represent the total number of 1s in the input string. If you find a 0 in the input string, just keep moving to the next symbol. After the input string (everything before the #) is iterated through, for each 1 after the

#, change the 1 to a $\cancel{1}$, and go back to the start of the string and start crossing off 1 symbol per 1 after the #. (1^* will change to $\cancel{1^*}$, and 0 will change to $\emptyset$). So if there are 3 1s after the #, the first 3 symbols in the string to the left of the # will be crossed off. Then if the last character crossed off is $\emptyset$, reject because the $\#(1,x)$-th symbol of x is not 1. If the last character crossed off is $\cancel{1^*}$, accept because the $\#(1,x)$-th symbol of x is 1.

Note: On s_8, there is a loop if it is any of the symbols in the tape alphabet. This actually means any of the symbols except for $\vdash$, because $\vdash$ should take you to the next state.

Problem 43.



$Q = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, \vdash, B\}$ (B = blank, shown as underscore in diagram)
$\vdash = \vdash$
B = B
s = s_0
t = s_5
r = s_7
δ =(see transition diagram above)

Idea: Since the Turing Machine starts with a $\vdash$, state s_0 goes to state s_1 if the symbol is $\vdash$, which then keeps it as a $\vdash$ and moves right. Starting at the beginning of the string, if you see a 0 or blank, go to reject state. Otherwise, if you see a 1, turn it into a blank and then move all the way right. Once a blank is seen, starting iterating left, and once a 0 is found, turn it into a blank and move all the way left. Then, once a blank is found, repeat the steps searching for a 1 while moving right. If there are more 1s than 0s, accept. Otherwise, reject. (Also, state s_6 is there to consider if there is a pattern like 101 in the string. Without s_6, a string like 1101 would be accepted.)