

CAS 705, Computability and Complexity

Assignment 1

Xiao Shu (0901994)
shux@mcmaster.ca

October 10, 2009

1. We are going to show the language $L = \{1^{2^n} | n \leq 0\}$ being non-regular by using contradiction. Suppose L is regular. By the Pumping Lemma, for a sufficiently large input in L , say $x = 1^{2^n}$, there exists $u, v, w \in \Sigma^*$ where v is non-empty, such that $x = uvw$ and $\forall i, uv^i w \in L$. However, if $|x| = |v|$, $uv^3w = x^3 = 1^{3 \cdot 2^n} \notin L$; and if $|x| > |v|$, $2^{n+1} > |uv^2w| > 2^n$ and $uv^2w \notin L$. Both scenarios contradict to the Pumping Lemma, thus, L cannot be regular.
2. Suppose $M = (Q, \Sigma, \Gamma, \delta)$ is a one-tape deterministic TM. When M starts running, it first makes sure that there is at least one 1 and no other letters on the input tape. Then M scans from left to right on the tape starting with state q_{even} . At the first time when M encounters a 1 on the tape, it changes the 1 to 0 and switches from q_{even} to q_{odd} , then moves on to the left square. After encountering another 1, M transfers back to state q_{even} , but does not erase the letter on tape. So on and so forth, M keeps flipping between q_{even} and q_{odd} , and removing 1s at state q_{even} until reaching the right end of the input. Now, if the current state of M is q_{even} , it goes back to the beginning and starts the procedure again; if the state is q_{odd} , it scans the tape and halts on q_{reject} if there is a letter 1 left on some square or halts on q_{accept} otherwise.

We can show that such M decides $L = \{1^{2^n} | n \leq 0\}$. First, let n_i be the number of 1s on the input tape after $i \geq 1$ rounds of left-to-right scanning, and specially, let n_0 be original number of 1s. In each round scanning, M removes a 1 and skip another, therefore, if it ends up at q_{even} after the $i + 1$ th round, n_i must be even and so does n_j for any $0 \leq j < i$, since otherwise, M should halt after the $j + 1$ th round. Thus, unless for the last round, $n_{i+1} = \frac{1}{2} \cdot n_i = \left(\frac{1}{2}\right)^{i+1} \cdot n_0$. On the other hand, if the state is q_{odd} after $i + 1$ th round, M stops scanning and goes back to check the remaining 1 on the tape. At this moment, the number of 1s left is,

$$\begin{aligned}n_{i+1} &= \frac{1}{2} \cdot (n_i - 1) \\2n_{i+1} + 1 &= \left(\frac{1}{2}\right)^i \cdot n_0 \\n_0 &= (2n_{i+1} + 1) \cdot 2^i\end{aligned}$$

Therefore, $n_0 = 2^k$ for some $k \geq 0$ iff $n_{i+1} = 0$, i.e., no 1 is left on the tape. This implies that M decides L .

By taking logarithm of both sides of above formula, the total number of scanning rounds, $i + 1 = O(\log n)$. Since each round takes $O(n)$ steps, M decides L in time $O(n \log n)$.

3. In order to show that any language accepted by a one-tape deterministic TM in time $o(n \log n)$ is regular, we are going to prove that if a language L can be decided by a TM M which only accesses each square on its tape for no more than constant times, then L is regular, or in other words, there exists an equivalent FSA which decides L . Such FSA can be constructed from M as follows.

First, for any string $s \in \Gamma^*$ and letter $z \in \Gamma$, we define function $g_{sz} : Q^* \rightarrow Q$ over concatenated string sz to present the state of the machine M after multiple accesses to the square storing z . As M writes on the tape freely and might change the letter on the square storing z , we still call it z square for convenience. For example, if $g_{sz}(q_1q_2 \dots q_i) = q$, it implies that the machine starts on sz and moves its head to z square for the first time on state q_1 , then the machine continues on other squares, and when the next time it moves to back to the z square from its left, the state is q_2 . So on and forth, after the head moving on z square from the left with state q_i , the machine will then turn to state q when it leaves the square to its right. If M halts in the middle of the process, it stops and outputs the current state as the result of g_{sz} immediately. To calculate g_{sz} , we bring in another function $h_{sz} : Q^* \times \Gamma \times Q \rightarrow Q^* \times \Gamma \times Q$, where the first input of h_{sz} is the crossing sequence of states from z square to its left until now with the head on z , and the second and third input are the current letter on z square and machine state respectively, the output are the corresponding sequence, letter and state when the head moves out to the right of the z square. By the description of h_{sz} , it can be defined recursively,

$$h_{sz}(S, x, q) = \begin{cases} (S, x', q') & \text{if } \delta(x, q) = (x', q', \rightarrow) \\ h_{sz}(Sq', x', g_s(Sq')) & \text{if } \delta(x, q) = (x', q', \leftarrow) \end{cases} \quad (1)$$

Enumerating each time when M moves onto z square, we can get a list of h_{sz} and build g_{sz} with the list.

$$\begin{aligned} h_{sz}(q_s, x_1, q_{sz}) &= (S_2, x_2, q'_1) & \dots & & g_{sz}(q_{sz}) &= q'_1 \\ h_{sz}(S_2, x_2, q_2) &= (S_3, x_3, q'_2) & \dots & & g_{sz}(q_{sz}q_2) &= q'_2 \\ h_{sz}(S_3, x_3, q_3) &= (S_4, x_4, q'_3) & \dots & & g_{sz}(q_{sz}q_2q_3) &= q'_3 \\ & \dots & & & & \dots \end{aligned} \quad (2)$$

In Equation 2, q_s is the state when M moves to the last square of s for the first time, so $g_s(q_s) = q_{sz}$ where q_{sz} is the state when M moves to z square for the first time. In addition, $x_1 = z$, therefore, with only the knowledge of q_s , z and g_s , we can build function g_{sz} .

As the length of crossing sequence for each square is less than a constant for any input, there are only fixed numbers of possible input and output for g_s , hence, the number of functions that can be g_s is also constant to the size of input string. Based on the fact that g_s depends only on M , we can construct a new TM M' in which each state “remembers” q_s, g_s , and transfers to q_{sz}, g_{sz} with letter z . This TM always moves its head from left to right and never goes back until it halts with $q_{sz} \in \{q_{accept}, q_{reject}\}$. Since the original TM M decides L , M' will halt eventually on any input as well, though it might not halt immediately after encountering the first empty square \sqcup .

However, this problem can be fixed by pre-running M' with an empty tape and testing whether a state transfers to q_{accept} or q_{reject} , then hard-wiring the result in transition function. Therefore, there is an FSA which equivalent to M could decide L , which implies that L is regular.

Next, we are going to show that if a language L is not regular, then it requires $\Omega_{weak}(n \log n)$ steps to decide on a single tape TM M . Before going to the proof, we define the crossing sequence of states more formally. For each $i \in \{1 \dots |x|\}$, let S_i be the i th crossing sequence of states of M on input string x , which records the states consecutively when the head moves between square i and $i + 1$.

From the previous result, we know that if L is not regular, the size of the crossing sequence on M could not be bounded by a constant, as otherwise, L has to be regular. Thus, for a randomly picked integer k_1 , there exists an input x_1 such that for at least one of the crossing sequence S_i , $|S_i| > k_1$. Suppose that x_1 is of minimal length n_1 . If $i > \frac{n_1}{2}$, then no two crossing sequences of the first half of x_1 can be the same, otherwise, we can remove the part in between from x_1 , and the result string still meets the requirement $|S_i| > k_1$ except for being even shorter, which contradicts the assumption. In other words, there are at least $i - 1$ unique crossing sequences. Considering that there are only $|Q|^m$ different crossing sequences whose lengths are m , the sum of the lengths of the first $i - 1$ crossing sequences has a lower bound with some constant c ,

$$\begin{aligned} \sum_{j=1}^{i-1} |S_j| &\geq \sum_{j=1}^{i-1} \lceil \log_{|Q|} j \rceil \\ &\geq \frac{n_1}{4} \cdot \log_{|Q|} \left(i - 1 - \frac{n_1}{4} \right) \\ &\geq cn_1 \log n_1 \end{aligned}$$

The total number of steps to decide x_1 is also lower-bounded,

$$T_M(x_1) > \sum_{j=1}^{i-1} |S_j| \geq cn_1 \log n_1 \quad (3)$$

Similarly, if $i \leq \frac{n_1}{2}$, this argument gives the same result for the second half of x_1 . Overall, x_1 has to take $cn_1 \log n_1$ steps on M .

Now, we let k_2 be a number larger than the required steps of any input of length n_1 . Based on previous result, there exists an x_2 of minimal length $n_2 > n_1$ and one of its crossing sequence is longer than k_2 . By using this method, we can find an infinite sequence $\{n_i\}$ such that $\forall n_i \exists x_i, |x_i| = n_i$ and M requires at least $c \cdot n_i \log n_i$ steps. Therefore, if a language is not regular, then it requires $\Omega_{weak}(n \log n)$ steps. In other words, if a language can be decided by a single tape TM in time $o(n \log n)$, it is regular.