Exam 2. November 17, 2021

CS181: Fall 2021

Guidelines:

- The exam is closed book and closed notes. Do not open the exam until instructed to do so.
- Write your solutions clearly and when asked to do so, provide complete proofs. You may use results and algorithms from class without proofs or details as long as you state what you are using.
- I recommend taking a quick look at all the questions first and then deciding what order to tackle to them in. Even if you don't solve the problems fully, attempts that show some understanding of the questions and relevant topics will get partial credit.
- You can use extra sheets for scratch work, but you can only use the white space (it should be more than enough) on the exam sheets for your final solutions. The exams will be scanned into gradescope so for each problem, only the corresponding white space will be used for grading.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly and any cheating reported with the score automatically becoming zero.
- Write clearly and legibly. All the best!

Problem	Points	Maximum
1		5
2		3
3		4
4		2
5		2
6		3
7		3
8		3
Total		25

Name	
UID	

1 Problem

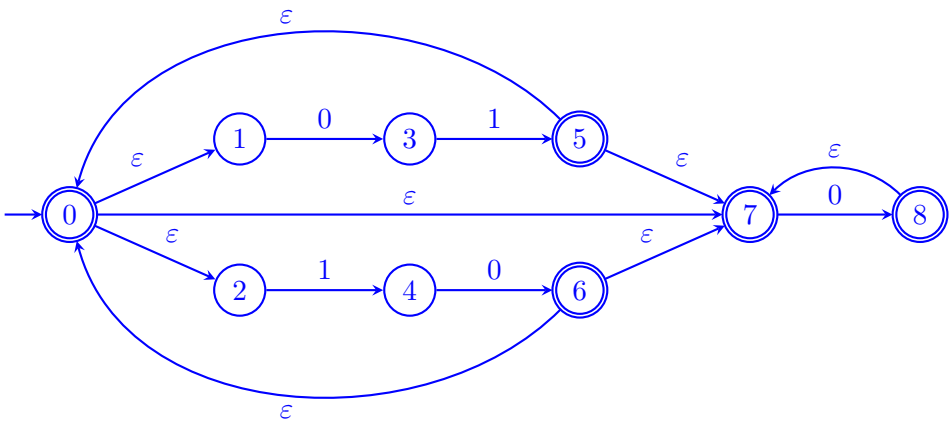
The answers to the following should fit in the white space below the question.

1. (Fill in the _____ so that the following statement is true. You may use Big Oh notation). Suppose we have a regular expression r of size m (that is, it has m symbols as in the homework) that computes a language L. Then there exists a DFA with at most $2^{O(m)}$ states which computes L. For full-credit, your answer should be correct and as small as the right answer (up to constant factors). Explain your answer in a few sentences. [1 point]

   Solution. Recall that we can convert a regular expression into an equivalent NFA with $O(m)$ states using the construction discussed in class. We also have the subset construction to convert an NFA with t states to a DFA with at most 2^t states. By combining the two results, we obtain a DFA with at most $2^{O(m)}$ states. □

2. Draw a NFA that computes the language computed by the regular expression $r = (01|10)^*(0)^*$. Only draw the final NFA in the space below. No need to draw the intermediate steps. [1.5 points]

   Solution. The following NFA is equivalent to r.



   □

3. True or False: If $f, g : \{0,1\}^* \to \{0,1\}$ are not regular functions (i.e., they cannot be computed by DFAs), then their AND $h : \{0,1\}^* \to \{0,1\}$ defined as $h(x) = f(x) \wedge g(x)$ is not regular. If true, justify your answer few sentences. If false, give an example to show why not. [1.5 points]

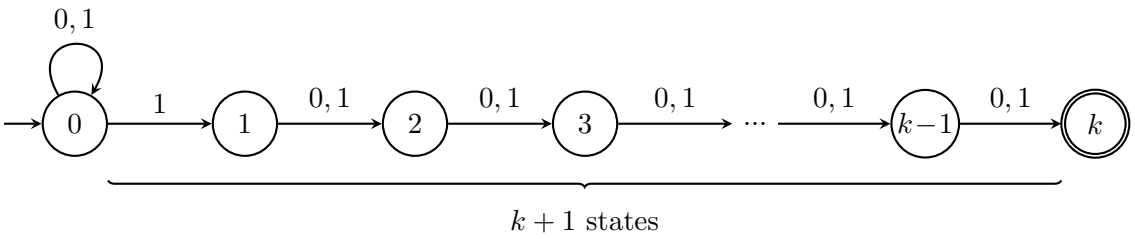
   Solution. False. Let $f(x) = 1$ if and only if $x = 0^n1^n$, for $n \geq 1$, and let $g(x) = 1$ if and only if $x = 1^n0^n$, for $n \geq 1$. Then $h(x)$ is always zero that corresponds to the empty language which is regular. □

4. Define the Eval (for TMs) and Toddler functions as we defined in class. [1 point]

Solution. $EVAL : \{0,1\}^* \to \{0,1\}*$ such that $EVAL(\langle M \rangle, x) = M(x)$ where the first argument is a binary encoding of a TM M. $TODD : \{0,1\}^* \to \{0,1\}$ such that $TODD(\langle M \rangle) = 1$ if and only if $M(\langle M \rangle)$ halts, and the first bit of its output is 0. □

2 Problem

Consider a NFA N on $k+1$ states defined by the following transitions $\delta_N(0,0) = \{0\}$, $\delta_N(0,1) = \{0,1\}$, and $\delta_N(i,0) = \{i+1\}$, $\delta_N(i,1) = \{i+1\}$. The set of accepting states is $S_N = \{k\}$. See the diagram below.



Let D be the DFA that we obtain by converting the NFA N into a DFA using the subset construction we did in class.

(A) Describe the language accepted by D in a sentence. [.5 points]

Solution. Words in which the k-th symbol from the end is 1. □

(B) What is the start state of D? Describe the accepting states of D. [.5 points]

Solution. The start state is $\{0\}$. The set of accepting states consists of all the 2^k states that correspond to sets which contain k. □

(C) In our conversion, D would have 2^{k+1} states with each state corresponding to a subset of $\{0, 1, \ldots, k\}$. Call a state I *reachable* from start if there is some sequence of transitions from start that lead the machine to state I. Are all elements of the power set of $\{0, 1, \ldots, k\}$ reachable from start? If not, describe what states are reachable from the start. You don't have to justify your answer. [.5 points]

Solution. Reachable states are those that correspond to subsets which contain 0. One way to guess this is that the state 0 is always present if you are in state that already has 0. □

(D) For this problem, suppose $k \geq 7$. Prove or disprove that the following states are reachable from start. If true, give an input that will make the DFA visit the specific state. If false, explain why not.

[Do not attempt to draw the entire DFA to figure this out - that could be tedious and too time consuming. Try to reason about the construction.]

The idea here is to use the structure of the transition function of D and work backwards. When you are at a subset I that contains 0, then taking the 0 edge gives you a subset that shifts all elements other than 0 and includes 0, and taking the 1 edge will give you a subset that shifts all elements and includes 0.

(a) Is the state corresponding to the subset $\{0, 5\}$ in D reachable? [.5 points]

Solution. Yes. You can work backwards here: To get to $\{0, 5\}$, it suffices to get to $\{0, 4\}$ and take edge 0; to get to $\{0, 4\}$ it suffices to get to $\{0, 3\}$ and take edge 0, and so on. To get to $\{0, 1\}$ we just need to take edge labeled 1. So $x = 10000$ will suffice. □

(b) Is the state corresponding to the subset $\{0, 2, 7\}$ reachable? [.5 points]

Solution. Yes. Working backwards, to get to $\{0, 2, 7\}$ it suffices to get to $\{0, 1, 6\}$ and take edge 0; to get to $\{0, 1, 6\}$ it suffices to get to $\{0, 5\}$ and take edge labeled 1. We can now use how to get to $\{0, 5\}$ from above. So for example, $x = 1000010$ will suffice. □

(c) Is the state corresponding to the subset $\{0, 3, 5\}$ reachable? [.5 points]

Solution. Yes. We can follow similar logic as above and work backwards. For example, $x = 10100$ will suffice. □

Bonus remark: As an aside, we can actually formally prove that all states that contain 0 are reachable by extrapolating the above idea of working backwards. Consider state $S = \{0, i_1, i_2, \ldots, i_\ell\}$, and suppose it doesn't contain k. Its 0 edge leads to state $\{0, i_1 + 1, i_2 + 1, \ldots, i_\ell + 1\}$. Indeed, from state 0 in N there is only one 0-edge which is a loop, and from state i in N we have only a 0-transition to state $i+1$. Now, the 1-edge advances us one step further along the path in N but there is also a loop on 1 in state 0 of N. Therefore, in state S of D, the outgoing 1-edge leads to state $\{0, 1, i_1+1, i_2+1, \ldots, i_\ell+1\}$. In words, in state S we can take two actions:

- Use 0-transition and increment all the nonzero elements of set S,
- Use 1-transition, increment all the nonzero elements of set S, add 1 to the set.

If k is in the set, then any of the two actions removes it from S, and then changes S as described above, because N has no transitions from k. Now it is not very hard to see that these two actions are enough to produce any set $S = \{0, i_1, i_2, \ldots, i_\ell\}$ from a start set $\{0\}$.

For example, consider $S = \{0, 2, 4, 7\}$. Let I be the current state. In the beginning, $I = \{0\}$. Our path is the following.

i. Use 1-edge to add 1 to I, so $I = \{0, 1\}$.
ii. Use 0-edge to increment $1 \in I$ two times. We want to have a state $\{0, 3\}$ because $3 - 0 = 7 - 4$, i.e., after further increments, these two elements will become 4 and 7, so $I = \{0, 3\}$.
iii. Use 1-edge to add 1 to I, so $I = \{0, 1, 4\}$ (nonzero elements get incremented).
iv. Use 0-edge to get the second difference which is $2 - 0 = 4 - 2$, so $I = \{0, 2, 5\}$.
v. Use 1-edge to add another 1 to I, so $I = \{0, 1, 3, 6\}$.
vi. Use 0-edge once to finally reach S.

The complete path is 1001010.

Now we are ready to prove that the reachable states are exactly states that contain 0. Obviously, a state without 0 is not reachable because N has both loops on zero, so if a state contains zero, then the next states will also contain zero. We'll use the algorithm described in the example above and induction to formally show that any state that contain 0 can be reached from $\{0\}$. Let $S = \{0, i_1, i_2, \ldots, i_\ell\}$ where $0 < i_1 < i_2 < \ldots < i_\ell$. If $\ell = 1$, so a state is $\{0, m\}$, it is reachable via 10^{m-1}. Now suppose all the states with fewer than ℓ nonzero elements are reachable. We'll show that S is also reachable. First, we use induction hypothesis to reach $\{0, i_2 - i_1, i_3 - i_1, \ldots, i_\ell - i_1\}$ which is possible because it contains $\ell - 1$ nonzero elements. Then, we use 1-edge once to get to $\{0, 1, i_2 - i_1 + 1, i_3 - i_1 + 1, \ldots, i_\ell - i_1 + 1\}$. Finally, we use 0-edge $i_1 - 1$ times to increment all the nonzero states and reach S.

3 Problem

Write down regular expressions for the following two languages. Please provide a sentence or two of explanations for your regular expressions.

1. $L = \{x : \ x \text{ contains both a 0 and a 1 }\}$. [2 points]

   Solution. The language is generated by regular expression

   $$\Big(\{0 \mid 1\}^* 01 \{0 \mid 1\}^*\Big) \mid \Big(\{0 \mid 1\}^* 10 \{0 \mid 1\}^*\Big).$$

   The first part generates words that start with 0 and at some point switch to 1. Similarly, the second part handles words that start with 1 and have 0 somewhere in the middle. □

2. $L = \{x : \ \text{number of 1's in } x \text{ is one more than a multiple of 3}\}$. For instance, $1000, 1111000, 1100110$ are in L but $111, 1100, 00101$ are not in L. [2 point]

   Solution. The following regular expression generates L.

   $$\Big(0^*10^*10^*10^*\Big)^* 0^*10^*.$$

   Expression in parentheses generates words with exactly three ones. Applying Kleene-star to it, we get words with $3k$ ones. Adding the final 0^*10^* gives us words with $3k + 1$ ones for $k \geq 0$. □

4 Problem

Prove that $L = \{1^{n^3} : n \geq 1\}$ is not a regular language. [2 points]

Solution. Suppose L is regular. Then there exists some $p > 0$ such that the pumping lemma holds. Let $x = 1^{p^3} \in L$. Suppose $x = abc$ such that

1. $length(ab) \leq p$,
2. b is nonempty,
3. $ab^i c \in L$ for every $i = 0, 1, 2, \ldots$.

Now let $k = length(b)$. We know that $0 < k \leq p$. Consider word $x' = ab^2c$. Notice that

$$p^3 = length(x) < length(x') = length(x) + k \leq p^3 + p < p^3 + 3p^2 + 3p + 1 = (p+1)^3.$$

Hence, $length(x')$ is a number strictly between p^3 and $(p+1)^3$, so x' cannot be in L, a contradiction. Therefore, L is not regular. □

5 Problem

Let L be the language that contains odd-length binary strings where the first, and middle symbol are the same. For instance, $110, 1011010 \in L$ but $100 \notin L$. Prove that L is not a regular language. [2 points]

Solution. Suppose L is regular. Then there exists some $p > 0$ such that the pumping lemma holds. Let $x = 0^p 01^p \in L$. Suppose $x = abc$ such that

1. $length(ab) \leq p$,
2. b is nonempty,
3. $ab^i c \in L$ for every $i = 0, 1, 2, \ldots$.

Since the first p symbols are zeros, we know that $b = 0^k$ for some $0 < k \leq p$. Consider word $x' = ac = 0^{p-k+1}1^p$. If k is odd, then $length(x')$ is even, and $x' \notin L$. Assume $k = 2m$. Notice that x' still starts with 0. For an odd-length word of length n, the middle symbol has index $(n+1)/2$, if the first symbol has index 1. Hence, the middle symbol of x' has index

$$\frac{length(x') + 1}{2} = \frac{(2p - 2m + 1) + 1}{2} = p - m + 1.$$

However, we have only $p - 2m + 1$ zeros in x', so the $(p - m + 1)$-th symbol must be 1 because $m > 0$. Hence, the first symbol of x' is not equal to the middle symbol, so x' cannot be in L in any case, a contradiction. Therefore, L is not regular. □

6 Problem

Let $L = \{x :$ some non-trivial prefix of x is equal to a suffix of $x\}$. Here, non-trivial prefix means a prefix that is neither empty nor the entire string. For instance $01011001, 101011$ are in L. Prove that L is not a regular language. [3 points]

Solution. Suppose L is regular. Then there exists some $p > 0$ such that the pumping lemma holds. Let $x = 0^p 10^p 1 \in L$. Suppose $x = abc$ such that

1. $length(ab) \leq p$,
2. b is nonempty,
3. $ab^i c \in L$ for every $i = 0, 1, 2, \ldots$.

Since the first p symbols are zeros, we know that $b = 0^k$ for some $0 < k \leq p$. Consider word $x' = ab^2c = 0^{p+k}10^p1$. Notice that all suffixes of length greater than $p+1$ have two 1's, while all the non-trivial prefixes have at most one 1. On the other hand, suffixes of length at most $p+1$ end with 1, so they also cannot match any non-trivial prefix because prefixes of length at most $p+1$ contain only 0's. Indeed, the first $p+k$ symbols of x' are zeros and $k > 0$.

Hence, none of the non-trivial prefixes of x' are equal to a suffix of x', a contradiction. Therefore, L is not regular. □

7 Problem

Define the function $Sort : \{0,1\}^* \to \{0,1\}^*$ as a function that on input x outputs a reordering of x that has all the 0's in x appears before the 1's. For instance, $Sort(0100) = 0001$, $Sort(11001) = 00111$, $Sort(011010) = 000111$.

Describe a single-tape TM for computing $Sort$. Describe the TM in pseudocode at a level of detail as we did in class for Maj. [3 points]

Solution. TM M operates as follows. In each iteration, it starts in the beginning of the tape.

- M scans the input left to right. For each found 1, it checks the next symbol.
  - If the next symbol is 0, swap the two symbols $10 \to 01$, and return to the start, i.e., M writes 1, moves left, writes 0 and returns to the start.
  - If the next symbol is 1, keep scanning and checking the next symbol.
  - If the next symbol is $\varnothing$, return to the start and halt.
- if M reaches the end of the input without seeing any 1's, it returns to the start and halts.

□

8 Problem

Define a function $Square : 1^* \to 1^*$ that takes as input $Square(1^n)$ and outputs 1^{n^2}. For instance, $Square(1) = 1$, $Square(11) = 1111, Square(111) = 111111111$ and so on. (We do not care about non unary inputs.)

Describe a TM for computing $Square$. Describe the TM in pseudocode at a level of detail as we did in class for Maj. You may use multiple tapes. [3 points]

Solution. We design a 3-tape Turing machine M with the first tape being the input tape, and the last tape being the output tape.

- First, M copies the input to the second tape which will be used as a counter.
- In each iteration, M appends x to the third tape.
  - If M points to $\varnothing$ on the second tape, cleanup and halt.
  - If M points to 1 on the second tape, move the corresponding head to the right and start copying x.
  - When copying x, M moves its heads on the first and the third tapes to the right simultaneously. At every step, M writes 1 on the third tape until it sees $\varnothing$ on the first tape.
  - When copying is finished, return the first head to the beginning of x, and proceed to the next iteration.

□