Question 7 **(6 marks)**

This question uses the following **Definitions** and **Facts**.

Definitions:

- If L is any language, the *reversal* of L is the set of all reversals of strings in L. In other words, you obtain the reversal of L by taking every string in L and writing it backwards. So, for example, the string abb becomes bba.
- If L is any language over the alphabet $\{\mathtt{a},\mathtt{b}\}$, then the *interchange of* a *and* b forms a new language as follows: take every string in L, and replace every a by b and every b by a, simultaneously. So, for example, the string abb becomes baa.

Facts:

- The class of regular languages is closed under reversal.
- The class of regular languages is closed under interchange of a and b.
- The language $\mathrm{AB} := \{\ \mathtt{a}^n\mathtt{b}^n\ :\ n \in \mathbb{N}\ \}$ is not regular.

Your task:

Using these facts, and any other closure properties of regular languages you like, *prove by contradiction* that the language

$$\{\ \mathtt{a}^m\mathtt{b}^n\ :\ m \le n\ \}$$

is not regular.

Assume, by way of contradiction, that the language $\{\ \mathtt{a}^m\mathtt{b}^n\ :\ m \le n\ \}$ is regular. Then its reversal, namely $\{\ \mathtt{b}^n\mathtt{a}^m\ :\ m \le n\ \}$, is regular, since the class of regular languages is closed under reversal (using the first fact). Then we interchange a and b, giving the language $\{\ \mathtt{a}^n\mathtt{b}^m\ :\ m \le n\ \}$, which must be regular, since the class of regular languages is closed under interchange of a and b (using the second fact). Take the intersection of the language we started with, and this last language. The former's b-part is at least as long as its a-part, while the latter's a-part is at least as long as its b-part. Their intersection consists of strings where the two parts have the same size:

$$\{\ \mathtt{a}^m\mathtt{b}^n\ :\ m \le n\ \} \cap \{\ \mathtt{a}^n\mathtt{b}^m\ :\ m \le n\ \} = \{\ \mathtt{a}^n\mathtt{b}^n\ :\ n \in \mathbb{N}\ \} = \mathrm{AB},$$

which must be regular, since the class of regular languages is closed under intersection (as discussed in lectures). But we know that AB is not regular (third fact). So we have a contradiction. So the original assumption, that the language given in the question is regular, is wrong. So that language is not regular.

Question 8 **(3 marks)**

Explain how to derive, for any Finite Automaton, a regular grammar for the language recognised by the FA.

For each state, create a new non-terminal symbol to represent that state. The initial state is represented by the start symbol S.

For each transition, create a production rule as follows. If the transition has label x and goes from state P to state Q, the production rule is $P \to xQ$. For each final state, create a production rule whose right-hand side is the empty string. So, for a final state represented by non-terminal symbol P, we create the rule $P \to \varepsilon$.

Official use only
9

Question 5 **(4 marks)**

Let R be any regular expression.

(a) Give, in terms of R, a regular expression for the language of all strings that can be divided into two substrings, each of which matches R.

RR

(b) Prove that the language EVEN(R) of all strings that can be divided into *any even number* of substrings, each of which matches R, is regular.

> For example, if R is `a` $\cup$ `bb`, then the string $w =$ `aabba` can be divided into four substrings `a`,`a`,`bb`,`a` which each match R. So this w belongs to EVEN(R). The empty string is also in EVEN(R), noting that zero is an even number. But `aabb` and `bbb` do not belong to EVEN(R).

Suppose that a string s can be divided into an even number of substrings, each of which matches R. Let the number of such substrings be $2k$. Then s can be divided into k pairs of substrings, with each pair consisting of two strings that match R. Each pair must therefore match RR, by (a). So s can be divided into k strings that match RR. Since k can be any number (including 0), our string s must match $(RR)^*$. Conversely, any string that matches $(RR)^*$ must consist of some number of substrings that each match RR, and therefore it must consist of an even number of substrings that each match R.

This shows that EVEN(R) is described by the regular expression $(RR)^*$, so it must be regular.

Official use only
4

(c) Use the Pumping Lemma for Regular Languages to prove that GOAL is not regular.

Assume, by way of contradiction, that GOAL is regular. Then, by Kleene's Theorem, there is a Finite Automaton that accepts it. Let N be the number of states of this FA.

Let w be the string $\mathtt{g}\mathtt{o}^{2N}\mathtt{a}^{N}\mathtt{l}$. This has length $> N$. Therefore, by the Pumping Lemma for Regular Languages, w can be divided into substrings x, y, z such that $y \neq \varepsilon$, $|xy| \leq N$, and for all $i \geq 0$ the string $xy^iz \in$ GOAL.

The constraint $|xy| \leq N$ tells us that y is within the first N letters of w. This means that y *either* contains the `g` at the very start of w *or* lies within the substring $\mathtt{o}^{2N}$. If it contains `g`, then repeating it to form $xyyz$ gives a string with more than one `g`, which therefore cannot belong to GOAL (since every string in GOAL just has one `g`, the one at the beginning). If y lies within the substring $\mathtt{o}^{2N}$, then repeating it to form $xyyz$ increases the length of the `o`-substring: it now has length $> 2N$ (using $y \neq \varepsilon$). But the length of the `a`-substring has not changed: it's still N. So the length of the `o`-substring is no longer exactly twice the length of the `a`-substring. This breaks the rules of the language GOAL, so $xyyz$ cannot belong to GOAL. So, regardless of where y is located in w, the string $xyyz \notin$ GOAL.

This violates the conclusion of the Pumping Lemma. So we have a contradiction. Therefore our initial assumption, that GOAL is regular, was wrong. So GOAL is not regular.

Deletion of y works just as well as repeating it, since the Pumping Lemma says that $xy^iz \in$ GOAL for all $i \geq \mathbf{0}$; the $i = 0$ case tells us that $xz \notin$ GOAL.

Official use only
13

Question 7 **(9 marks)**

The language DOG consists of all strings of the form

$$\texttt{gr}^n(\texttt{woof})^n$$

where n is any positive integer. For example, the strings `grwoof` and `grrwoofwoof` both belong to DOG, but `grrwoof` does not.

(a) Use the Pumping Lemma for Regular Languages to prove that DOG is not regular.

Assume, by way of contradition, that DOG is regular.
Then there is a FA that recognises it. Let N be the number of states in such an FA.
Let w be the string $\texttt{gr}^N(\texttt{woof})^N$.
By the Pumping Lemma, w can be divided up into three parts, $w = xyz$, such that y is nonempty, $|xy| \leq N$, and $xy^iz \in$ DOG for all $i \geq 0$.
The requirement that $|xy| \leq N$ forces y to fall within the first part, $\texttt{gr}^N$, of w.
Consider the string $xyyz$. If y contains `g`, then $xyyz$ has two `g`s, so $xyyz \notin$ DOG, since every string in DOG has exactly one `g`.
If y contains no `g`, then it's all-`r`, so repetition of `y` creates at least one extra `r` (since y is nonempty), so the number of `r`s is greater than the number of `woof`s, which violates the definition of DOG. So $xyyz \notin$ DOG, which contradicts the conclusion of the Pumping Lemma.
So our initial assumption, that DOG is regular, must be incorrect.
So DOG is not regular.

Other choices of w are possible. You could let w be any string of the form $\texttt{gr}^n(\texttt{woof})^n$ whose total length is $> N$. But then you'd have to have three cases in the proof, according to whether y lies within $\texttt{gr}^n$, or within $(\texttt{woof})^N$, or contains some of each (i.e., it straddles the boundary between the two parts). This is ok if done correctly.

Instead of the string $xyyz$, we could have chosen xy^iz for any $i \geq 2$. The argument would be almost identical. We also could have chosen xz (i.e., $i = 0$). In that case, the argument is slightly different (but not much): if y includes `g`, then xz contains *no* `g`, which violates the requirement that every string in DOG has exactly one `g`; if it does not contain `g`, then it's all-`r`, so the string xz has *fewer* `r`s than `woof`s, which violates the defintion of DOG.