

WEEK 7 WORKSHOP
MATH2301, SEMESTER 2, 2025

In all problems, “binary string” refers to any string or word on the alphabet $\Sigma = \{0, 1\}$. Unless otherwise specified, our alphabet is always $\Sigma = \{0, 1\}$. If r is a regular expression, then $L(r)$ is the language described by r . This is the set of all strings that match r .

For this worksheet, make sure to discuss actively with your groupmates! Coming up with regexes that match what you want them to match, as well as describing the languages of given regexes, takes a lot of practice. It is a creative process and there is a lot of room for error. But you will learn the tricks of the trade quicker if you discuss frequently with others.

1. REGULAR EXPRESSIONS

1.1. **Problem.** If r is a regular expression, write down another regular expression s such that

$$L(s) = \{vwx \mid v, w, x \in L(r)\}.$$

Solution. The answer is rrr . This matches any string that can be cut into three pieces, each of which matches r .

1.2. **Problem.** Write down a regular expression whose language is

$$\{w \in \Sigma^* \mid w \text{ is any string except } 0 \text{ or } 1\}.$$

Solution. The matching strings either have length 0, or length larger than 2. The only possible string of length 0 is ϵ . Otherwise, the string has at least two letters, and then any number of other letters. One answer is

$$r = \epsilon \mid (0|1)(0|1)(0|1)^*.$$

1.3. **Problem.** Write down a regular expression r that matches exactly those binary strings which (when thought of as numbers in base 2) are divisible by 8. (Let us assume that we only consider a binary string to represent a valid number if it either starts with a 1, or if the whole string equals 0.)

Solution. We should only consider binary strings that begin with a 1, and as a special case, also the string 0. A non-zero binary number is divisible by 8 if and only if it ends with at least three zeroes. A string that only has 0s equals zero, and is divisible by 8. So here is an answer:

$$r = 1(0|1)^*000|0^*.$$

1.4. **Problem.** Let $r = 01^*0|10^*1$. Describe $L(r)$ in words.

Solution. This matches all strings with at least two letters that either begin and end with a 0 and have a block of 1s in the middle, or that begin and end with a 1 and have a block of 0s in the middle.

1.5. **Problem.** Write down a regular expression for the language that contains exactly those strings without two consecutive 1s. Discuss and convince each other that you haven’t missed anything or have anything extra.

Solution. First suppose the string has at least one 1. For any 1 in the string that is not initial, there must be at least one zero before and after it. So the first part should be 0^*1 , starting with some number of zeroes and ending at the initial 1. The second part should be something like $(00^*10^*)^*$, signifying that there will be at least one zero before each 1 in the string, and this repeats zero or more times for the number of ones after the initial one. Finally the string could end in even more zeroes, particularly if the second part matches the empty string, so we have to end with 0^* . Otherwise, the string may not contain any ones, in which case it looks like 0^* . So one possible answer is

$$r = 0^*1(00^*10^*)^*0^*|0^*.$$

1.6. **Problem.** Write down a regular expression whose language is

$$\{w \in \Sigma^* \mid w \text{ has exactly two 0s and at least two 1s}\}.$$

Discuss and convince each other that you haven't missed anything or have anything extra.

Solution. Mentally divide the string into three portions: before the first 0, between the two zeroes, and after the second 0. We have the following cases (not disjoint):

- (1) The first and second portions each have at least one 1 in them.
- (2) The first and third portions each have at least one 1 in them.
- (3) The second and third portions each have at least one 1 in them.
- (4) The first portion has at least two 1s.
- (5) The second portion has at least two 1s.
- (6) The third portion has at least two 1s.

(The last three cases cover the cases where, for example, there are no 1s in two out of the three portions.)

Then the first case, for example, is represented by the following regular expressions:

$$11^*011^*01^*.$$

The second and third are similar. Let r_1 be the regex representing a match to either the first, the second, or the third case. Then

$$r_1 = 11^*011^*01^* \mid 11^*01^*011^* \mid 1^*011^*011^*.$$

The fourth case, for example, is represented by the following regular expression:

$$111^*01^*01^*.$$

The fifth and sixth are similar. Let r_2 be the regex representing a match to either the fourth, fifth, or the sixth case. Then

$$r_2 = 111^*01^*01^* \mid 1^*0111^*01^* \mid 1^*01^*0111^*.$$

Overall, the regular expression is

$$r_1 r_2 = 11^*011^*01^* \mid 11^*01^*011^* \mid 1^*011^*011^* \mid 111^*01^*01^* \mid 1^*0111^*01^* \mid 1^*01^*0111^*.$$

1.7. **Problem.** For each of the problems 2, 3, and 4, try to come up with regular expressions that match precisely the strings that do not match the regular expression from the problem. You can either try to do this directly based on the descriptions of the language, or try to do it by manipulating the regular expressions. Can you find a systematic method for this?

Solution. We will see a systematic method in class shortly. For now it's probably easier to do directly.

- For question 2, it's easy to see that the complement language is exactly $\{a, b\}$, so the regular expression is $a|b$.
- For question 3, either the string is a valid number that ends with a 1, or has a 1 in the second to last position, or has a 1 in the third to last position, or the string is empty. The regular expression representing all this is

$$r_1 = \epsilon \mid (0|1)^*1 \mid (0|1)^*1(0|1) \mid (0|1)^*1(0|1)(0|1).$$

Otherwise, the string is not a valid number, which means that it starts with at least one zero and has at least one one in it. The regular expression representing the above is

$$r_2 = 00^*1(1|0)^*.$$

So the final answer is

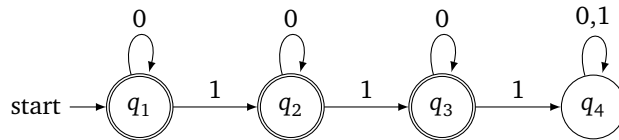
$$r_1 r_2 = \epsilon \mid (0|1)^*1 \mid (0|1)^*1(0|1) \mid (0|1)^*1(0|1)(0|1) \mid 00^*1(1|0)^*.$$

- For question 4, one possibility is that the string begins and ends with a different letter. If not, then it either begins and ends with a 0 and also has a 0 in the middle, or it begins and ends with a 1 also has a 1 in the middle, or the string is empty, or it either equals 0 or 1. So here is an answer:

$$\begin{aligned}
 r = & \epsilon | 0 | 1 | \\
 & 1(0|1)^*0 | \\
 & 0(0|1)^*1 | \\
 & 0(0|1)^*0(0|1)^*0 | \\
 & 1(0|1)^*1(0|1)^*1.
 \end{aligned}$$

2. DFAs

Answer the following questions about the DFA shown below.



2.1. **Problem.** What is the set of states?

Solution. Answer: $Q = \{q_1, q_2, q_3, q_4\}$.

2.2. **Problem.** What is the start state?

Solution. Answer: q_1

2.3. **Problem.** What is the set of accept states?

Solution. Answer: $A = \{q_1, q_2, q_3\}$

2.4. **Problem.** What is the transition function? Fill in the following table.

δ	0	1
q_1		
q_2		
q_3		
q_4		

Solution.

	0	1
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_4
q_4	q_4	q_4

2.5. **Problem.** Can you figure out a description for the language of this automaton?

Solution. This automaton M rejects all those strings that have at least three ones. We can see this because q_1 is the state at which we have “seen” no 1s, q_2 is the state at which we have seen one 1, and q_3 and q_4 are the states at which we have seen two and three 1s respectively. We reject anything beyond two ones, because at that point we land at q_4 from which we can’t escape. Therefore

$$L(M) = \{w \in \Sigma^* \mid w \text{ has at most two ones}\}.$$

2.6. **Problem.** Can you write down a regular expression that recognises the same language?

Solution. Here is a solution:

$$r = 0^* | 0^* 1 0^* | 0^* 1 0^* 1 0^*.$$