

Each of the problems is worth 10 points. Please write your solutions clearly and concisely. If you do not explain your answer you will be given no credit. You are encouraged to collaborate on the homework, but you *must* write your own solutions and list your collaborators on your solution sheet. Copying someone else's solution will be considered plagiarism and may result in failing the whole course.

Please turn in the solutions by 11.59pm on Thursday 24 September. The homework should be dropped off in the box labeled CSC 3130 on the 9th floor of SHB. Late homeworks will not be accepted.

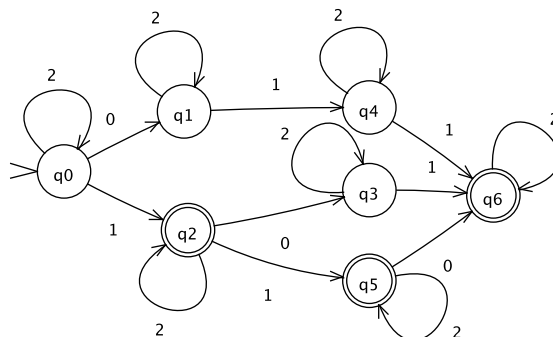
Problem 1

Give a DFA for the following languages, specified by a transition diagram. For each one of them, give a short and clear description of how the machine works. The alphabet is $\Sigma = \{0, 1, 2\}$:

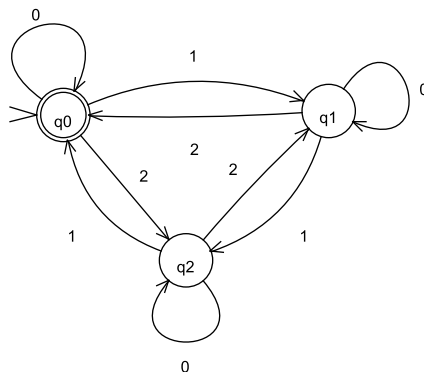
- (a) $L_1 = \{w : w \text{ has fewer 0s than 1s and at most two 1s}\}$.
- (b) $L_2 = \{w : \text{the sum of the digits of } w \text{ is divisible by 3}\}$.
- (c) L_3 is the language described by $0^*1^*2^*$.
- (d) L_4 is the language described by $(12)(0 + 1 + 2)^*(21)$.

Solution

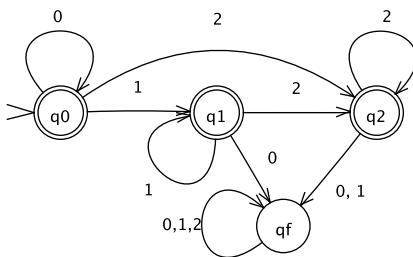
- (a) If we ignore the alphabet symbol 2, this DFA would accept only the strings 1, 11, 101, and 110. It is easy to draw a DFA that accepts those strings only. To accommodate the 2s we add self-loops around each state.



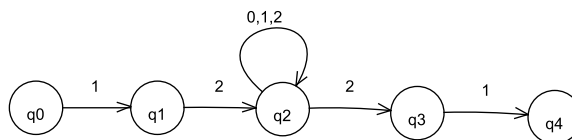
- (b) This is a 3-state DFA with states q_0, q_1 and q_2 . The DFA is in state q_i when the remainder of the sum modulo 2 is i .



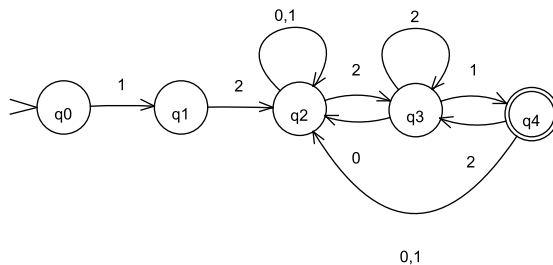
- (c) We have three states q_0, q_1, q_2 . As the 0s are coming in, we stay in state q_0 . When the 1s start coming, we move to q_1 . When we see the 2s, we switch to q_2 . If at any point we see a different symbol from what we expect, we switch to the "die" state q_f .



- (d) It is easy to construct an NFA for this expression,

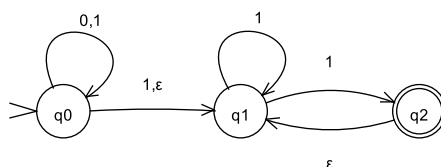


which can then be converted to the following DFA. (All the missing transitions end up in a "die" state, which we omit from the picture for the sake of clarity.)



Problem 2

This problem concerns the NFA given by the following transition table:



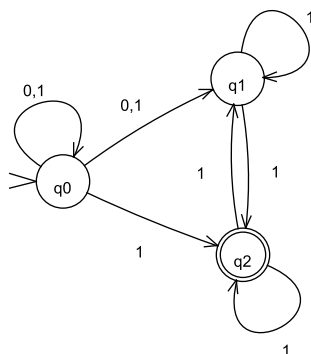
- Convert this NFA into one without ε -transitions.
- Convert the NFA from part (a) into a DFA using the method from class.

Solution

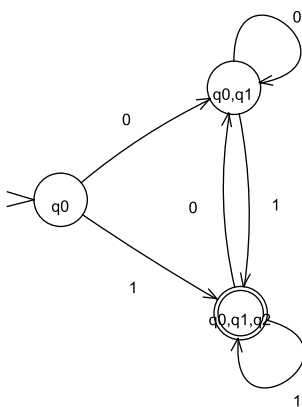
Here is the transition table of the NFA without ε -transitions.

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$
q_1	\emptyset	$\{q_1, q_2\}$
$*q_2$	\emptyset	$\{q_1, q_2\}$

In graphical representation, it looks like this:



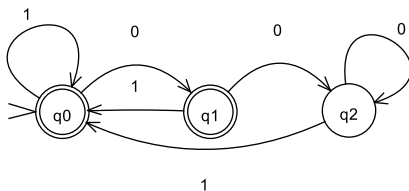
Using the conversion algorithm, we obtain the following DFA:



Problem 3

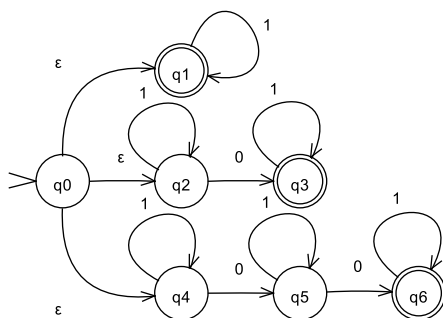
Consider the following languages over $\Sigma = \{0, 1\}$.

- (a) L_1 is the language of all strings that do not end in 00.
- (b) L_2 is the language described by $\varepsilon + 0 + 1 + (0 + 1)(0 + 1)^*$.
- (c) L_3 is the language of all strings that do not contain the pattern 00.
- (d) L_4 is the language of the following DFA:

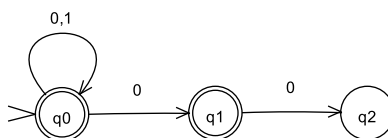


(e) L_5 is the language described by $(0 + 1)^*(01 + 10 + 11)$.

(f) L_6 is the language of the following NFA:



(g) L_7 is the language of the following NFA:



(h) L_8 is the language described by $1^*(01^*)^*(01^*)^*$.

Which of these languages are the same and which are different? To show two languages are the same give a short proof, and to show two languages are different give a string that is in one but not by the other. (You must provide an explanation to get credit.)

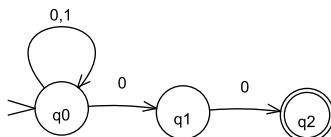
Solution

We have five equivalent groups, $\{L_2, L_7, L_8\}$, $\{L_1, L_4\}$, $\{L_3\}$, $\{L_5\}$, $\{L_6\}$. We show that the expressions in the same group are equivalent and those in different groups are not equivalent to each other.

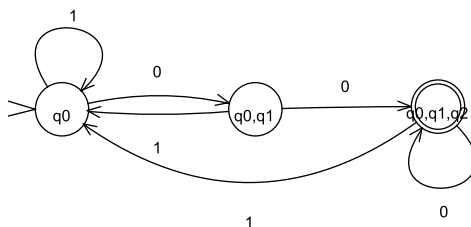
- $L_2 = L_7 = L_8 = (0 + 1)^*$: We can easily see that $L_2 = (0 + 1)^*$. For L_7 , it is clear that $L_7 = (0 + 1)^*$ because it accepts all strings in the state q_0 . For L_8 , first note that $1^*(01^*)^*(01^*)^* = 1^*(01^*)^*$. Now observe that every string can be written in the form $1^*(01^*)^*$:

Split the string in blocks that start with a 0 followed by zero or more 1s, up to the next 0 or where the end of the string is reached. Each such block is described by the pattern 01^* . Before the first block of 0s there may be a block of 1s, representing the initial pattern 1^* .

- $L_1 = L_4$: This can be seen by inspecting the DFA for L_4 . We see that the state q_2 captures exactly those strings that end in 00. Alternatively, we can first draw an NFA for the complement of L_1 :



When we convert this NFA to a DFA we obtain



which is exactly the DFA for L_4 with the accepting and rejecting states flipped.

- Notice that L_6 is the set of strings with no more than two 0s.
- $L_2 \neq L_1, L_3, L_5, L_6$ because 000 is in L_2 but not in L_1, L_3, L_5 , or L_6 .
- $L_1 \neq L_3, L_6$ because 0001 is in L_1 but not in L_3 or L_6 .
- $L_1, L_3 \neq L_5$ because $0 \in L_1$ and $0 \in L_3$ but $0 \notin L_5$.
- $L_3, L_5 \neq L_6$ because 01010 is in L_3 and L_5 but not in L_6 .

Problem 4

In this problem you will design an NFA that checks whether a web address is formatted correctly. Examples of correctly formatted web addresses are:

`http://www.gov.hk/en/about/govdirectory/index.htm`

`http://www.cse.cuhk.edu.hk/`

`http://en.wikipedia.org/wiki/automaton`

`google.com`

`http://www.travelchinaguide.com/attraction/hongkong/island/victoria.htm`

Examples of incorrect addresses are:

```
http://.invalid.domain.name/
http://no/domain/name/here.html
http:///too.many.slashes/
the://protocol.must.be.http
http://www.acme.com/web:page:name:cannot:contain:colons.html
```

In general, web addresses follow the Unified Resource Locator (URL) format, but for this problem you can make some simplifying assumptions about the kind of web addresses you are dealing with. You can assume that the address consists of the optional protocol identifier `http://`, followed by the domain name (e.g., `www.cse.cuhk.edu.hk`) ending in some suffix like `.com`, `.hk`, possibly followed by a slash `/`, then some optional directory structure (e.g., `en/about/govdirectory/`), and finally an optional web page name (e.g., `victoria.htm`).

For simplicity, assume web addresses consist only of the lowercase letters 'a', 'b', up to 'z' and the special symbols '/', ':', and '.'. When drawing the transition diagram of your NFA, you can use the shorthand notation `[a-z]` to describe transitions labeled by all the letters 'a', 'b', ... 'z'.

This is a design problem, and part of your job is to figure out a way to distinguish among correct and incorrect web addresses. There is no single right answer. You must describe your reasoning clearly in your solution. Solutions that only provide a picture of an NFA with no explanation will get no credit.

Solution

First, we look for the pattern `http://`. Then we look for the domain name: This is the “loop” between states q_1, q_2, q_3 , and finally after the last dot we move to q_4 for the suffix. (Here we assume that there is at least one string before the suffix.) This may be the end of it, so q_4 is an accepting state: But if there is a directory structure coming along, we capture it in the “loop” q_5, q_6 , and q_7 . Finally, if there is a web page extension (we assume there is at least one alphabetical character before and at least one inside the extension), we go to states q_8 and q_9 .

