5CCS2FC2: Foundations of Computing II

Tutorial Sheet 1

Solutions

1.1 Over the alphabet $\Sigma = \{0, 1\}$, construct a Finite Automaton (deterministic or non-deterministic) that accepts each of the following languages:

(i) The language of all binary strings that begin and end with a 1,

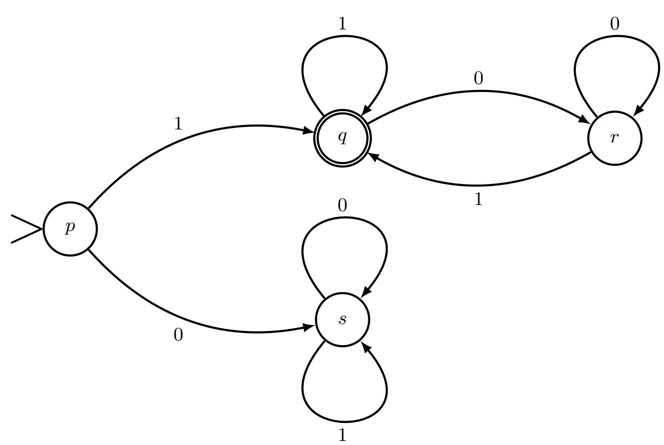
(ii) The language of all strings that contain exactly three 1s

(iii) The language of all strings that contain the substring 1010,

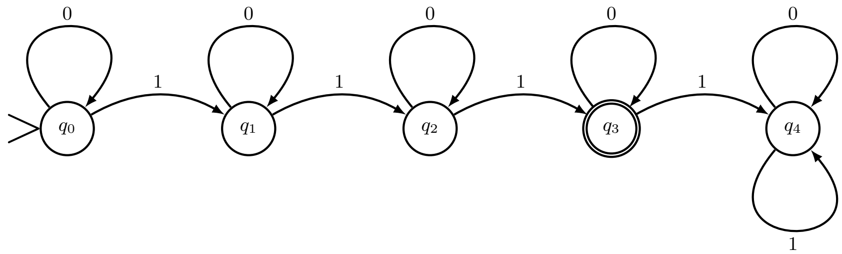
(iv) The regular language represented by the expression $1(01)^*1$,

<u>SOLUTIONS:</u>

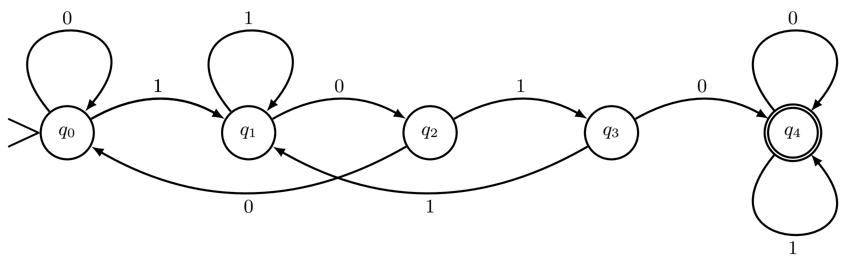
(i) The following four-state DFA accepts binary strings that begin and end with a 1,



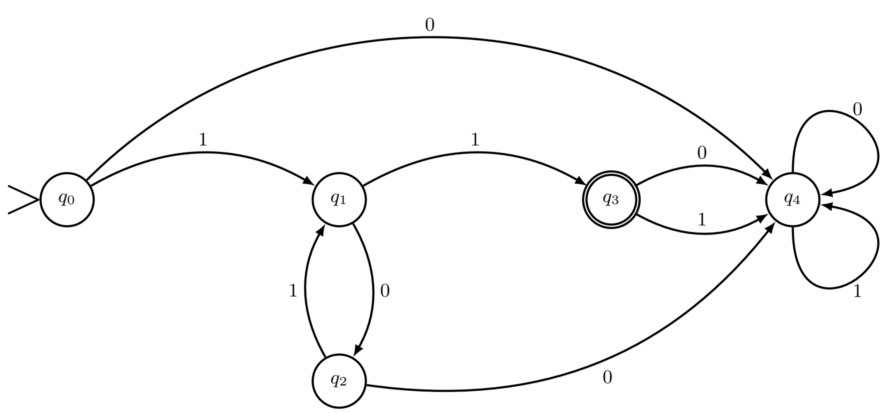
(ii) The following five-state DFA accepts only strings containg exactly three 1s.



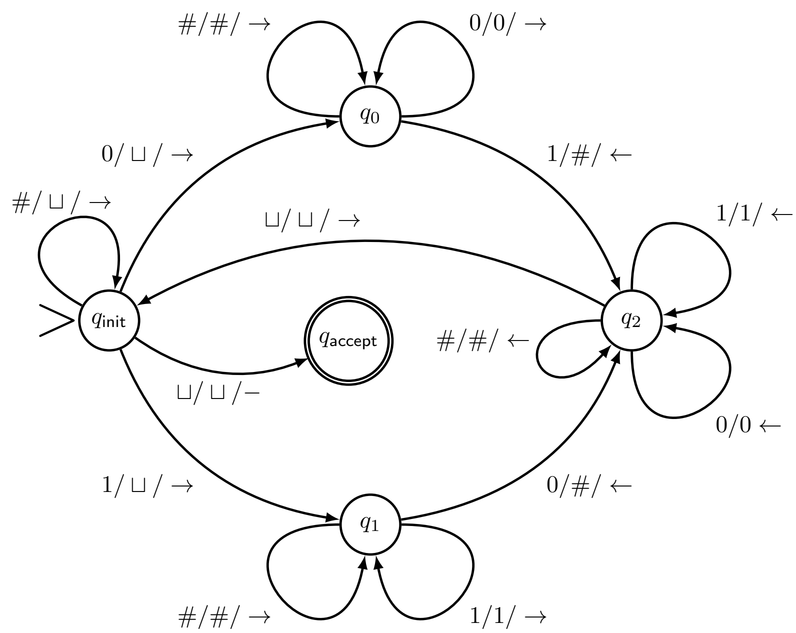
(iii) The following five-state DFA accepts only strings containg the substring 1010.



(iv) The following five-state DFA accepts the regular language whose strings can be represted by the expression 1(01)*1.



1.2 Let $\Sigma = \{0, 1, \#\}$, and consider the following Turing Machine $\mathcal{T}$ on the language, whose transition diagram is depicted below:



(Code for `turingmachinesimulator.com` is available on KEATS.)

(i) Which of the following words are accepted by this Turing Machine:
   - 1001,
   - 00,
   - 101,
   - 0011.

(ii) What is the language that is accepted by this Turing Machine? *i.e.*, what is $L(\mathcal{T})$?

<u>SOLUTIONS:</u>

(i)
   - 1001 Accept,
   - 00 Reject,
   - 101 Reject,
   - 0011 Accept.

(ii) The Turing machine accepts all words containing the same number of 1s and 0s. A rough description of the algorithm is given below:

- In the initial state, the head moves to the right seeking out a 1 or a 0.
- If there are no 1s or 0s then the machine accepts.
- If a 1 is located first then it is erased and the head continues moving to the right seeking out a matching 0. Once a matching 0 is found, a special symbol # is written in place and the head returns to the beginning of the word.
- If the head reaches the end of the word without finding a matching 0, then the machine halts and the word is rejected.
- The procedure for if 0 is located first is analogous.

1.3 Consider the language of all *palindromes* over the binary alphabet $\Sigma = \{0, 1\}$,

$$L = \left\{ w \in \{0,1\}^* \; : \; w = w^R \right\}$$

where w^R denote the reversal of $w \in \{0,1\}^*$ (*e.g.*, $(111001)^R = 100111$)

(i) Give 5 examples of words belonging to L,

(ii) Outline a pseudo-code program for a Turing machine that accepts the language L,

(iii) Convert your pseudo-code into a complete description of a Turing Machine. (Test that your machine works as intended using the `turingmachinesimulator.com`.)

<u>SOLUTIONS:</u>

(i) There are infinitely many possible answers; the 13 shortest palindromes are:

ε, 1, 0, 11, 00, 111, 000, 101, 010, 1111, 0000, 1001, 0110, ...

(Note that the empty string $\varepsilon = \varepsilon^R$ is also a palindrome.)

(ii) A possible solution is as follows:

(1) Read and remember the first/current tape symbol,

(2) Erase the current symbol, and replace with a blank symbol,

(3) Scroll right to the end of the word until you reach a blank symbol,

(4) Move one space left,

(5) If the current symbol is blank then enter an accepting state and terminate,

(6) If the current symbol does not match the remembered first character then enter a rejecting state and terminate,

(7) Otherwise, erase the current symbol and replace with a blank symbol.

(8) Scroll left to the beginning of the word until you reach a blank symbol,

(9) Move one space right,

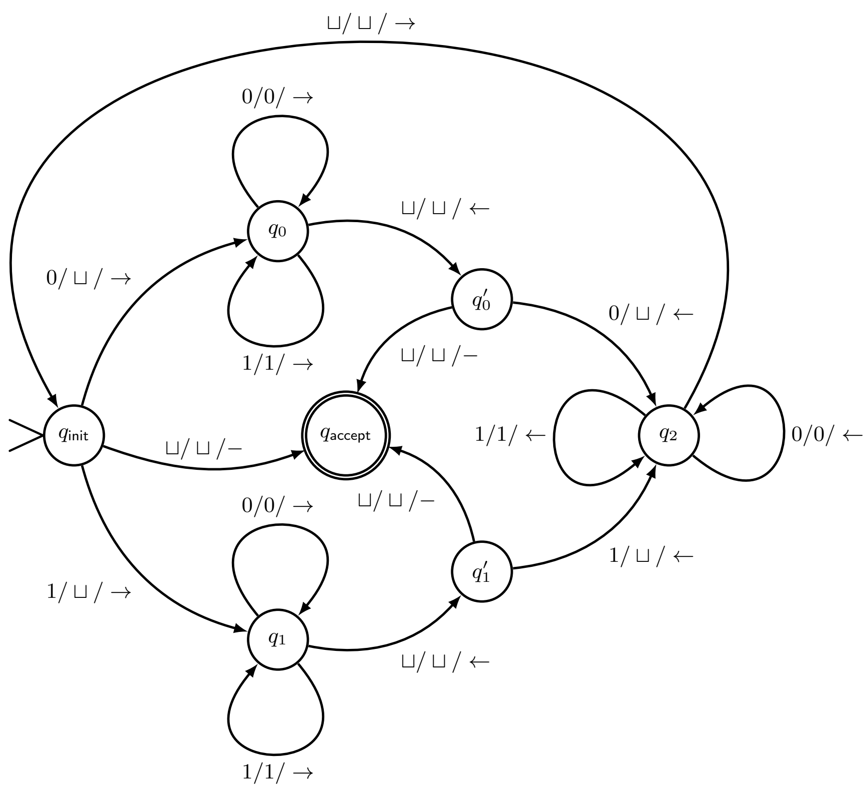
(10) Repeat from (1).

(iii) A possible solution might can be found using the following seven states, which encode the following behaviours of the machine:

Q	description of state
q_{init}	Initial state, check the first character
q_0	Remember 0 and scroll to the right
q_1	Remember 1 and scroll to the right
q_0'	Check for a matching 0
q_1'	Check for a matching 1
q_2	Scroll back to start of word
q_{accept}	Accept state

This intended behaviour can be captured with the following transition table:

Current State	Read Symbol	New State	Write Symbol	Move
q_{init}	0	q_0	$\sqcup$	$\rightarrow$
q_{init}	1	q_1	$\sqcup$	$\rightarrow$
q_{init}	$\sqcup$	q_{accept}	$\sqcup$	$-$
q_0	0	q_0	0	$\rightarrow$
q_0	1	q_0	1	$\rightarrow$
q_0	$\sqcup$	q_0'	$\sqcup$	$\leftarrow$
q_0'	0	q_2	$\sqcup$	$\leftarrow$
q_0'	$\sqcup$	q_{accept}	$\sqcup$	$-$
q_1	0	q_1	0	$\rightarrow$
q_1	1	q_1	1	$\rightarrow$
q_1	$\sqcup$	q_1'	$\sqcup$	$\leftarrow$
q_1'	1	q_2	$\sqcup$	$\leftarrow$
q_1'	$\sqcup$	q_{accept}	$\sqcup$	$-$
q_2	0	q_2	0	$\leftarrow$
q_2	1	q_2	1	$\leftarrow$
q_2	$\sqcup$	q_{init}	$\sqcup$	$\rightarrow$



1.4 Consider the language

$$L = \{w\#w \ : \ w \in \{0,1\}^*\}$$

comprising all those strings over the alphabet $\Sigma = \{0, 1, \#\}$, that consist of two copies of a binary string separated by a special character # that appears precisely once. Using the pigeon-hole principle, show that no DFA can accept the language L.

SOLUTIONS:

Suppose to the contrary that there is a DFA $\mathcal{A}$ that accepts all and only those strings of the form $w\#w$ for $w \in \{0,1\}^*$.

Let m be the number of states in our proposed automata $\mathcal{A}$.

Consider the word $1^k\#1^k \in L$, where $w = 1^k$ consists of k copies of the character 1, for some $k > m$.

Since the length of the initial substring 1^k is greater than the number of states m, the automata must revisit the some state q twice in this initial substring (by the pigeon-hole principle).

Let t_1 be the time of the first visit to q and t_2 be the time of the second visit to q.

As in the slides, we can remove the part of the string between t_1 and t_2 without affecting whether the string is accepted or not.

Therefore our proposed automaton must accept the string $1^{k-t}\#1^k$, where $t = (t_2 - t_1)$ is the length of the substring between the two appearances of q.

However, $1^{k-t}\#1^k \notin L$ is not a word belonging to L, since the initial word does not match the second word. Hence, our proposed automaton does not work as intended. This argument would work for *any* proposed DFA, which means that *no* DFA will accept all and *only* strings of the from $w\#w$. This completes the proof.

N.B. There are other possible choices for $w\#w$ that we could have chosen instead of $1^k\#1^k$, such as $0^k\#0^k$, for example. In fact any string w containing more than m of the same character would work! (why is this?) However, it is not enough that the length of w is greater than m, if w contains both 0s and 1s.