Informatics 1 - Computation & Logic: Tutorial 6

Computation: Introduction to Finite State Machines

Week 8: 9 - 13 November 2015

Please attempt the entire worksheet in advance of the tutorial, and bring with you all work, including (if a computer is involved) printouts of code and test results. Tutorials cannot function properly unless you do the work in advance.

You may work with others, but you must understand the work; you can't phone a friend during the exam.

Assessment is formative, meaning that marks from coursework do not contribute to the final mark. But coursework is not optional. If you do not do the coursework you are unlikely to pass the exams.

Attendance at tutorials is **obligatory**; please let your tutor know if you cannot attend.

For this tutorial you may find it helpful to use Matthew Hepburn's online FSM tool.

1. The ATM design at
   `http://homepages.inf.ed.ac.uk/s1020995/atm.html`
   is somewhat unforgiving. If you enter the wrong PIN it keeps your card.

   Modify the machine so that it gives you three chances to enter the right PIN, before keeping the card, and so it allows you to get the card back after either of the first two failures, if you want to.

   For the remainder of this tutorial, you will probably find it helpful to use
   `http://homepages.inf.ed.ac.uk/s1020995/create.html`

   You can toggle each tool on and off by clicking it. When no tool is selected you can test your FSM with different inputs - and step through an input string (you can step backwards, as well as forwards) If you have problems using the tool please start a thread on Piazza.

2. Consider the following Finite State Machine (FSM) over the (input) alphabet $\{0, 1\}$:

   There are just two states, called state A and state B. State A is the initial state. State A is the only accept state. There are transitions labelled '0' from state A to itself and from state B to itself. There are transitions labelled '1' from state A to state B and from state B to state A.

   (a) Draw this machine.

   (b) Which of the following strings are accepted by this machine? (Y/N)

ϵ	
0	
1	
00	
01	
10	
11	
101	
010	
0011	
1011	
1010	
10001	
11011000101	

   (c) What property is shared by all the strings which are accepted by this machine?

(d) Does the machine accept all strings with this property?

(e) What do the two states 'mean'?

(f) Fill out the following table so that the entry in row x column y contains the names of all states that can be reached from state x by a transition labelled y in the above machine:

	0	1
state A		
state B		

Use the table to decide whether or not the machine is deterministic, and explain why.

3. Consider the following FSM over alphabet $\{0, 1\}$:

   There are just three states — state A, state B and state C. State A is the initial state. State C is the only accept state. There are transitions labelled 0 from state A to state B, from state B to state C, and from state C to itself. There are transitions labelled 1 from state A to itself, from state B to state A, and from state C to itself.

   (a) Draw this machine.

   (b) What language is defined by this machine? (Think about what the states 'mean'.)

   (c) Fill out its transition table:

	0	1
state A		
state B		
state C		

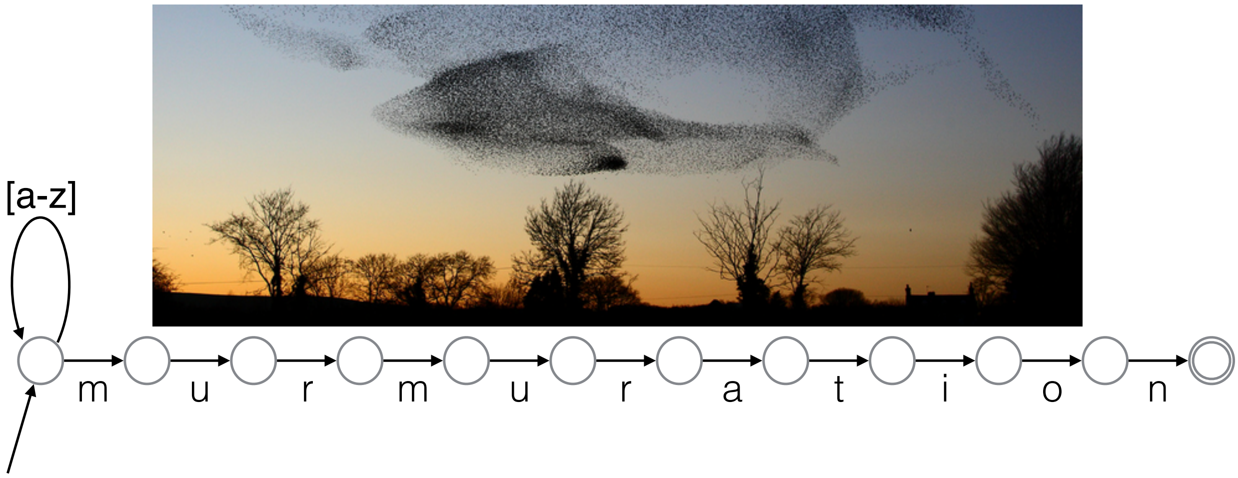
   Is the machine deterministic? Why?

4. Draw a finite state machine over alphabet $\{0, 1\}$ which accepts all and only those strings of which the string 101 is a substring.

5. Sketch a finite state machine over alphabet $\{0, 1\}$ which accepts all and only those strings consisting of a 42 zeros followed by *the same number* of ones?

6. Can you design a more general finite state machine over alphabet $\{0, 1\}$ which accepts all and only those strings consisting of some number of 0's followed by *exactly the same* number of 1's?

7. A flock of starlings is called a **murmuration**.



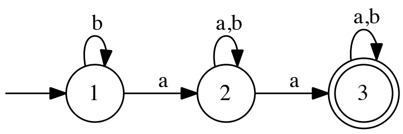
The NFA shown recognises strings, in the alphabet [a-z], that end with this word. The transition from the start state to itself can be taken with any letter of the alphabet.

Construct a DFA that recognises these strings.

Searching for strings in a long sequence, for example, searching for words in text, or genes in DNA sequences, is an important problem.

One of the best algorithms for many applications, and one of the most commonly used, was invented by Bob Boyer and J Moore, in Edinburgh, in the 1970s. If you want to learn more you can look at J Moore's page
`http://www.cs.utexas.edu/users/moore/best-ideas/string-searching/`
although he works in Texas he is still a frequent visitor to the Forum.

8. Consider the Finite State Machine in the diagram:



(a) Draw its transition table:

(b) Explain why this is a non-deterministic automaton (NFA)?

(c) Give a simple english description of the language accepted by this FSM.

(d) Give a regular expression for the language accepted by this FSM.

(e) Construct an equivalent DFA.

9. Draw a finite state acceptor which accepts an infinite set of strings each of which consists of an *odd number of* $0s$ and an *even number greater than* 0 *of* $1s$. To be more precise, if string s is accepted by the machine, then s must:

   (a) consists of an odd number of symbols;

   (b) contain only 0s and 1s;

   (c) contain an odd number of 0s; and,

   (d) contain an even number of 1s greater than 0.

   To create such acceptor, first create an acceptor for the even number (> 0) of 1s, then another acceptor for the odd numbers of 0s, and then put the two acceptors together using the correct operator. Explain your choice of the operator.

10. (a) Do DFAs and NFAs recognise the same languages?

(b) What advantages do NFAs have over DFAs, and vice-versa?

This tutorial exercise sheet was written by Paolo Besana, and revised by Thomas French, Areti Manataki, and Michael Fourman. Send comments to `Michael.Fourman@ed.ac.uk`