Informatics 1 - Computation & Logic: Tutorial 6

Computation: Non-Deterministic FSMs and Regular Expressions

Week 8: 7 - 11 November 2016

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.

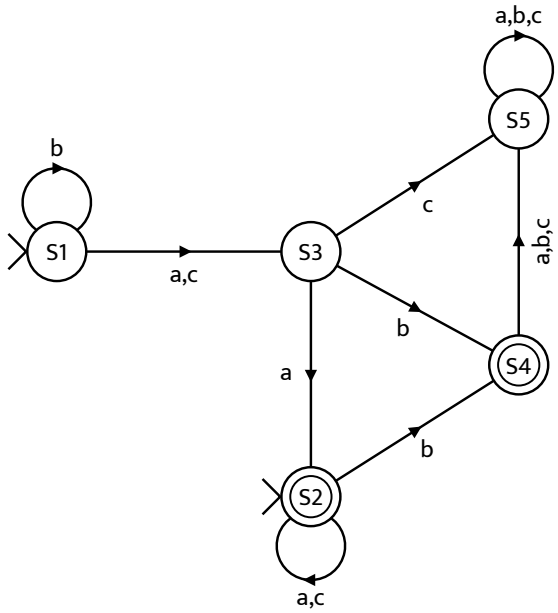
You may work with others, but you must understand the work; you can't phone a friend during the exam.

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.

You may find it useful to refer to the FSM Workbench question set which accompanies this tutorial at homepages.inf.ed.ac.uk/s1020995/tutorial6.

1. Consider the finite state machine in the diagram below.

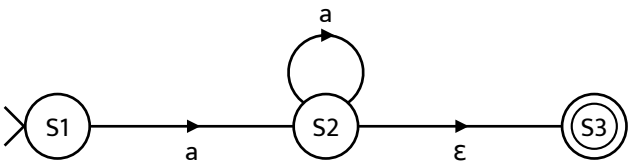


   (a) For each input sequence in the table below, record whether it is accepted by the FSM.

Input	Is Accepted?
⟨ ⟩	Y
b	Y
aa	Y
ba	N
abaab	N
acaca	Y
aaab	Y
bbbcb	Y
cacba	N

   (b) Is the FSM deterministic? Justify your answer.
   NO. As it has two initial states it can be in more than one state for some input sequences.

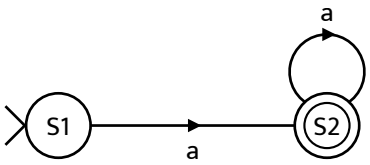
2. This NFA over the alphabet {a} uses an ε transition.



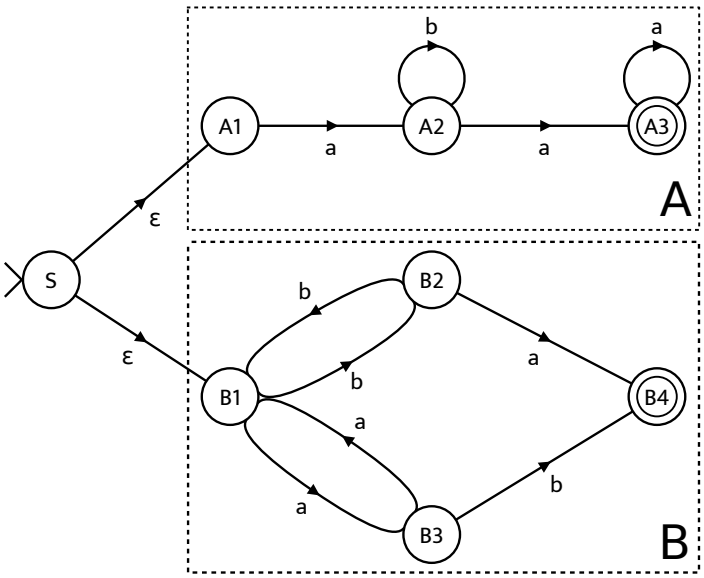
(a) Describe the language accepted by this machine in words.
All sequences of 'a' that are at least one character long

(b) Describe the language accepted by this machine using a regular expression. aa^*

(c) Design a deterministic machine that accepts the same language as this machine.



3. ε-transitions provide a simple way of combining FSMs. The machine below has been composed from two machines A and B, which had initial states A1 and B1.



(a) Considering machines A and B separately, give a regular expression which describes the language they accept. A: ab*aa*
B: (bb|aa)*(ba|ab)

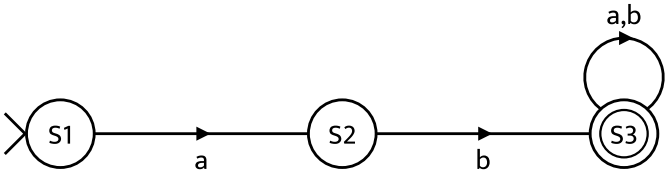
(b) Considering the whole machine, give a regular expression which describes the language the machine accepts.
(ab*aa*)|((bb|aa)*(ba|ab))

(c) L_A and L_B are the sets of inputs accepted by machines A and B. Give an expression relating L_A and L_B to L, where L is the set of input accepted by the whole machine. $L = L_A \cup L_B$

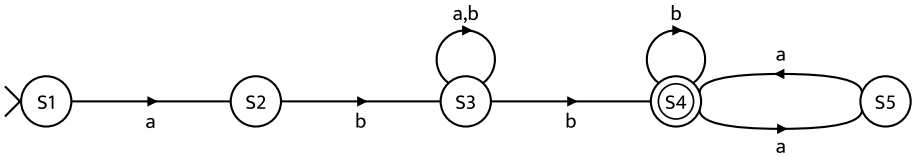
4. Consider the regular expression ab(a|b)*

(a) Describe in words the language that the expression matches. Include two examples of strings that are matched. The string 'ab' followed by zero or more 'a's and 'b's. Examples of accepted strings include 'ab', 'abaaa', and 'ababba'.

(b) Design a finite state machine that accepts that language.

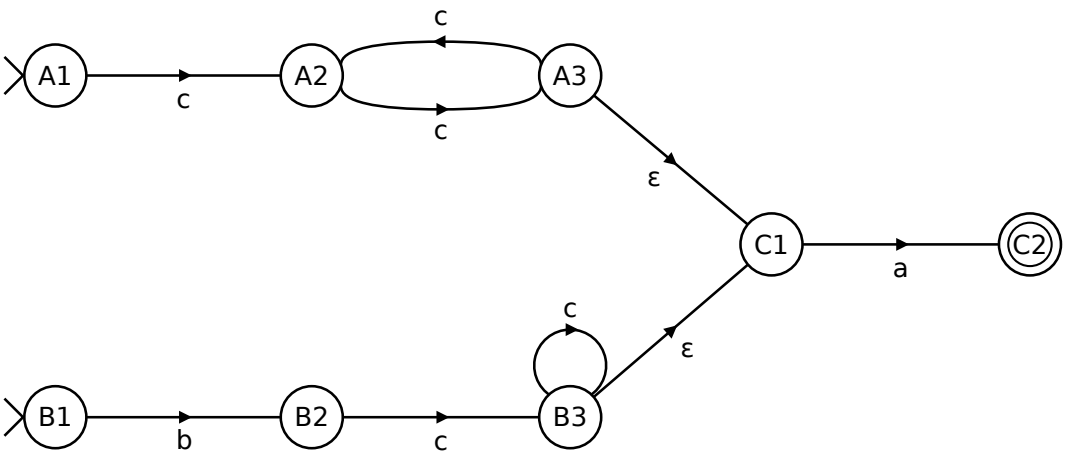


(c) Building on your answer to (b), design a finite state machine that accepts ab(a|b)*bb*(aa)*.



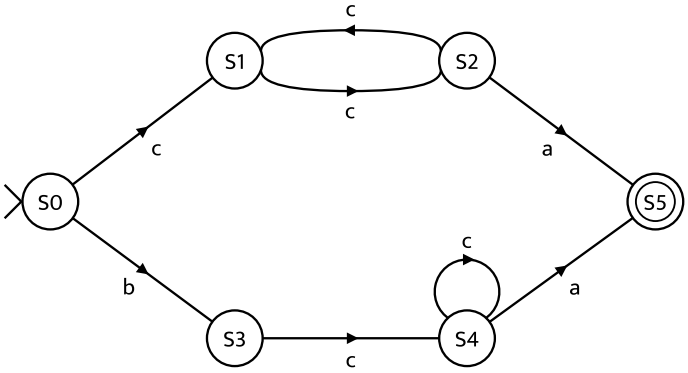
Is perhaps the obvious answer, but it may be puzzling to observe that a natural regex for this machine is ab(a|b)*b(b|aa)*. In fact, ab(a|b)*b(b|aa)* ≡ ab(a|b)*bb*(aa)* — why?
We can give a simpler FSM by observing that ab(a|b)*bb*(aa)* ≡ ab(a|b)*b(aa)*, which follow from the facts that bb* ≡ b*b and (a|b)*b* ≡ (a|b)*
Thus we can eliminate the b transition from S4 to S4 in the answer, to give an equivalent machine.

5. Consider this NFA over the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.



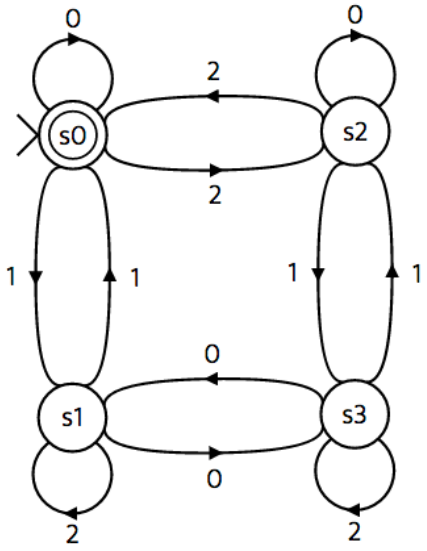
(a) Describe, both in words and with a regular expression, the language accepted by this machine. Hint: think about the sequences that end in A3 and B3. The machine accepts strings consisting of an even number (>0) of '`c`'s followed '`a`' or '`b`' followed by at least one '`c`' followed by '`a`'.
$(\mathtt{c}(\mathtt{cc})^*\mathtt{ca})|(\mathtt{bcc}^*\mathtt{a})$

(b) Design a DFA that accepts the same language.



(c) Are there any NFAs that cannot be converted into an equivalent DFA? No – all NFAs have an equivalent DFA.

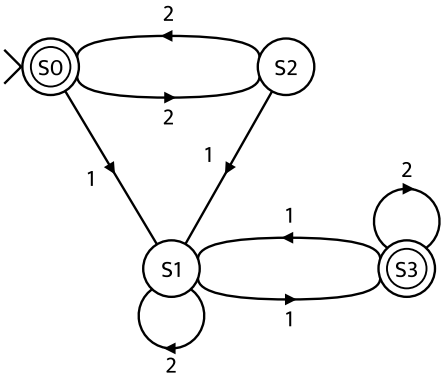
6. Consider this DFA over the alphabet $\{0, 1, 2\}$. It should be familiar.



(a) Describe, in words, the language accepted by this machine. Hint: Your description in words should refer to ternary numbers. This machine accepts ternary representations of natural numbers divisible by 4.

(b) Replace each transition labelled 0 by a transition labelled ε, between the same two states. The resulting automaton is not a DFA. (Why not?)
Because a DFA has no ε transitions.

i. Construct an equivalent DFA.

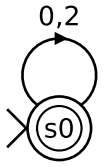


ii. Describe, both in words and with a regular expression, the language accepted by this machine.

Any string on the alphabet $\{1, 2\}$ which has an even number of 1's, and such that if that number is zero (i.e. if there are no 1's) it has an even number of 2's : $(2^*12^*12^*)^*(22)^*$

(c) Next, replace each transition (of the original machine) labelled 1 by a transition labelled ε, between the same two states.

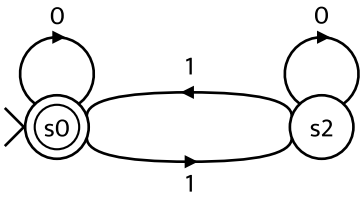
i. Again, construct an equivalent DFA, and, ii, describe the language it accepts.



Any string on the alphabet $\{0, 2\}$: $(2 \mid 0)^*$

(d) Repeat the exercise replacing each transition (of the original machine) labelled 2 by a transition labelled ε, between the same two states.

i. Construct an equivalent DFA, and, ii, describe the language it accepts



Any string on the alphabet $\{0, 1\}$ with an even number of 1's : $(0^*10^*10^*)^*0^*$

(e) BONUS QUESTION: Give a regular expression that describes the language accepted by the original machine. Test your answer using the grep utility.

This bonus question goes somewhat beyond the call of duty. Feel free not to attempt it. That said, by the end of week 7 you should have all the tools required to complete it. If you do choose to try it, I suggest you use cut and paste in some suitable editor to make, and keep track of the algebraic substitutions that are required.

By repeated application of Arden's Lemma, substitution and simplification, from the equations

$$\begin{aligned} L_0 &= L_0 0 \mid L_1 1 \mid L_2 2 \mid \varepsilon \\ L_1 &= L_0 1 \mid L_1 2 \mid L_3 0 \\ L_2 &= L_0 2 \mid L_2 0 \mid L_3 1 \\ L_3 &= L_1 0 \mid L_2 1 \mid L_3 2 \end{aligned}$$

For example, applying Arden's Lemms to the final equation for L_3 and then substituting the result for L_3 in the equations for L_1 and L_2, we obtain,

$$\begin{aligned} L_0 &= L_0 0 \mid L_1 1 \mid L_2 2 \mid \varepsilon \\ L_1 &= L_0 1 \mid L_1 2 \mid (L_1 0 \mid L_2 1) 2^* 0 \\ L_2 &= L_0 2 \mid L_2 0 \mid (L_1 0 \mid L_2 1) 2^* 1 \\ L_3 &= (L_1 0 \mid L_2 1) 2^* \end{aligned}$$

We can apply distributivity $(x \mid y)z = (xz \mid yz)$, for regular expressions, together with the commutativity and associativity of $\mid$, to regroup the equation for L_2 to a form suitable for Arden.

$$\begin{aligned} L_2 &= L_0 2 \mid L_2 0 \mid (L_1 0 \mid L_2 1) 2^* 1 \\ &= L_0 2 \mid L_2 0 \mid L_1 0 2^* 1 \mid L_2 1 2^* 1 && \text{(distrib)} \\ &= L_0 2 \mid L_1 0 2^* 1 \mid L_2 0 \mid L_2 1 2^* 1 && \text{(comm)} \\ &= L_0 2 \mid L_1 0 2^* 1 \mid L_2 (0 \mid 1 2^* 1) && \text{(distrib)} \\ L_2 &= (L_0 2 \mid L_1 0 2^* 1)(0 \mid 1 2^* 1)^* && \text{(Arden)} \end{aligned}$$

Continuing in similar vein, to eliminate L_2 and L_1, we eventually apply Arden to an equation for L_0, to derive
$L_0 = (0|2(0|12^*1)^*2|(1|2(0|12^*1)^*12^*0)(2|02^*0|02^*1(0|12^*1)^*12^*0)^*(1|02^*1(0|12^*1)^*2))^*$