Informatics 2A 2015–16
Tutorial Sheet 1 (Week 3)

John Longley

This week's tutorial is devoted to the 'pure theory' of regular languages. The relevant lectures are numbers 3, 4 and 5. Next week's tutorial will cover some applications of this theory. Please attempt the first six questions in advance of tutorials. While all 6 questions cover material it is important to understand, please make sure that you especially prioritize the even-numbered questions. Don't spend too long on any one question. If you get stuck, go on to the next. An optional Question 7 is included for students who like a challenge.

1. Give both an NFA and a regular expression for each of the following languages. Try to keep the number of states in your NFAs as small as possible.

   (a) (Over $\{a, b\}$.) The set of strings in which a and b *alternate*: that is, strings not containing aa or bb consecutively.

   (b) (Over $\{a, b\}$.) The set of strings that do contain aa or bb consecutively. (This is the *complement* of the language in part (a).)

   (c) (Over $\{a, b\}$.) The set of strings that contain the consecutive sequence $abba$.

   (d) (Over $\{a, b, c\}$.) The set of strings in which between any two occurrences of a, there must be at least one occurrence of b.

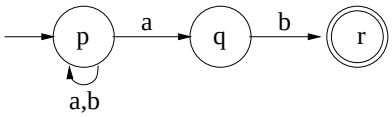
   Construct NFAs (possibly with ϵ-transitions) corresponding to the following regular expressions. The construction of your NFAs should broadly follow the structure of the regular expressions as indicated in the lectures, though you may omit ϵ-transitions that you can see to be superfluous.

   (e) $(a^* + b^*)^*$

   (f) $(((aa)^*bb) + ab)^*$

   (g) $\emptyset^*$

2. Use the subset construction from Lecture 3 to convert the following NFA into an equivalent DFA. You need only include in your NFA those states that are reachable from the initial state.



3. For a language $L \subseteq \Sigma^*$ the *reversal* of the language is defined by:

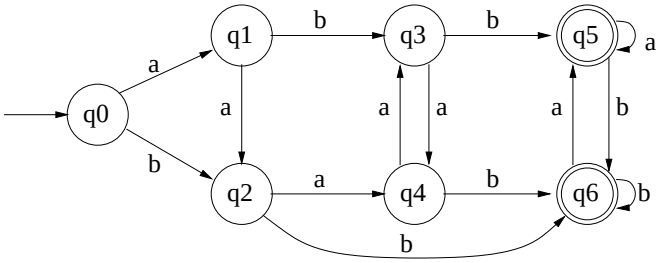
   $$L^r = \{a_n a_{n-1} \dots a_2 a_1 \mid a_1 a_2 \dots a_{n-1} a_n \in L \text{ where } n \geq 0 \text{ and } a_1, \dots, a_n \in \Sigma\} \ .$$

   In other words, L^r is the set of all strings that are obtained by reversing (i.e., writing right-to-left instead of left-to-right) strings in L.

Prove that regular languages are closed under the operation of reversal.

Hint: given an NFA for L define another NFA for L^r over the same set of states as the original NFA.

4. Use the algorithm from Lecture 4 to minimize the DFA below.



5. (Adapted from Kozen.) Convince yourself intuitively that the following identities are valid for regular expressions. Recall that an equality $\alpha = \beta$ between regular expressions means that the equality of languages $\mathcal{L}(\alpha) = \mathcal{L}(\beta)$ holds.

$$\begin{aligned} (\alpha + \beta)\gamma &= \alpha\gamma + \beta\gamma \\ \alpha(\beta + \gamma) &= \alpha\beta + \alpha\gamma \\ (\alpha\beta)^*\alpha &= \alpha(\beta\alpha)^* \end{aligned}$$

Consider the regular languages over $\{0, 1, 2\}$ denoted by the following pairs of regular expressions. In each case, say whether the two languages are equal or not. If they are, use the above laws (plus any other laws from Lecture 5 that you may require) to prove the equivalence. If they are not, give an example of a string that is in one language but not the other.

(a) $(0 + 1)^*$ and $0^* + 1^*$

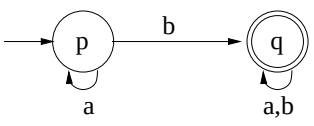
(b) $0(120)^*12$ and $01(201)^*2$

(c) $(0^*1^*)^*$ and $(0^*1)^*$

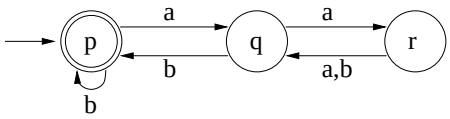
(d) $(01 + 0)^*0$ and $0(10 + 0)^*$

6. Convert each of the following DFAs into a regular expression, by using Arden's rule to solve a system of simultaneous equations as indicated in Lecture 5.

(a)



(b)



7. (Optional challenge question.) Perform the following construction starting with the DFA drawn in Question 4.

   (a) Produce an NFA by applying the "reversal" construction defined in your answer to Question 3.

   (b) Determinize the resulting NFA and extract the reachable part.

   This should result in a DFA with 4 states. (Note that, in order to carry out step (b), you are recommended *not* to carry out the full subset construction, which would build a DFA with 128 states. Instead, find a way to construct the reachable part of the determinized DFA directly.)

   Now repeat steps (a) and (b) starting with the new DFA. Do you recognise the DFA you obtain as a result of this repetition of the process?