Informatics 2A 2018–19
Tutorial Sheet 1 (Week 3)

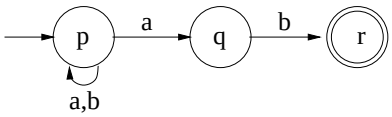
Mary Cryan

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 five questions in advance of tutorials. Whilst all of these cover material it is important to understand, you should especially prioritize questions 1, 3 and 5. Don't spend too long on any one question. If you get stuck, go on to the next.

An optional Question 6 is included for those who like tennis.

1. Use the subset construction from Lectures 3, 4 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.



2. 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.

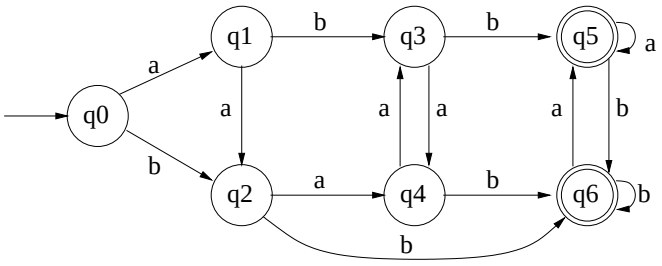
   Construct NFAs (possibly with ϵ-transitions) corresponding to the following regular expressions. The construction of your NFAs should broadly follow the structure of the regular expressions as indicated in the lectures, though you may omit ϵ-transitions that you can see to be superfluous.

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

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

   (g) $\emptyset^*$

3. Use the algorithm from Lecture 5 to minimize the DFA below.



You may do this as in the lectures, inserting a tick into the triangular grid each time you discover a pair of states that must be kept separate. Alternatively, instead of inserting a tick, you could insert an example of a string separating the two states which the algorithm has 'discovered' (see Lecture 4, Slide 33). Doing this may help your understanding of why the algorithm works.

4. (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 6 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 $0^* + 1^*$

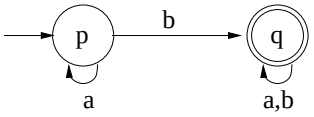
(b) $0(120)^*12$ and $01(201)^*2$

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

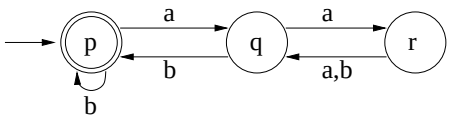
(d) $(01 + 0)^*0$ and $0(10 + 0)^*$

5. 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 6.

(a)



(b)



6. (Optional.) In this question we'll consider a finite-state model for an ordinary game of singles tennis (not a tiebreak). If you're needing to brush up on the scoring system for tennis, try this:

   `https://en.wikipedia.org/wiki/Tennis_scoring_system`

   The players are Federer and Murray, and Federer is serving. We'll model a game simply as a string over the alphabet $\{f, m\}$, where f means 'point to Federer' and m means 'point to Murray'.

   (a) Construct a DFA that accepts precisely those strings in $\{f, m\}^*$ that correspond to complete games won by Murray. Your DFA should have a state for each possible scoreline that can occur: e.g. 30-15; Deuce.

   (b) Is your DFA *minimal*? If not, which states may be identified? (Do this part just by inspection—I don't recommend applying the algorithm from lectures to your DFA!)

   (c) Now consider an entire tennis match (with up to five sets, and including tiebreaks as necessary) as a sequence of points in this way. Would it be possible to construct a DFA in this style that accepts precisely those sequences that correspond to a complete *match* won by Murray?