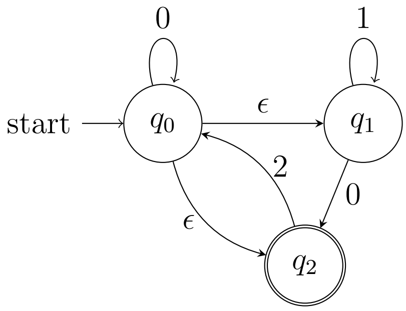
AC32008 Theory of Computation

Tutorial Sheet 3 - Equivalence Between NFA-ϵ's and NFAs. Regular Expressions. Regular Expressions and Finite Automata.

1. Consider an NFA-ϵ below. Find an NFA without ϵ-transitions that accepts the same language.



2. For each of the languages described by the regular expressions below, give *two* strings that are *members* and *two* strings that are *not members* - a total of four strings for each part. Assume that the alphabet in each case is $\{0, 1\}$.

   (a) $\mathbf{0}(\mathbf{10})^*\mathbf{1}$

   (b) $\mathbf{0}^* + \mathbf{1}^*$

   (c) $(\epsilon + \mathbf{0})\mathbf{1}^*$

   (d) $(\mathbf{0} + \mathbf{10} + \mathbf{11})\Sigma^*$

3. Find regular expressions representing each of the following languages over the alphabet $\{0, 1\}$, and justify your answers.

   (a) The set of all strings with exactly one occurrence of the substring 00.

   (b) The set of all strings in which the number of 1s is divisible by three.

   (c) The set of all strings not containing 101 as a substring.

Hint. A good way to guess a regular expression for a language is to write down a longish (probably at least 20 symbols) "typical" string in the language. Work out how to break it up into pieces which you can easily describe by a r.e., then work out how to put these together. Also you need to think about exceptions - strings which are in the language but which are not typical.

Hint. For the kind of problem in (c), it is often helpful to think of breaking a string into pieces by splitting it immediately before (or sometimes after) each occurrence of a particular symbol. Another similar approach is to note the obvious but useful fact that a binary string consists of (non-empty) blocks of zeros alternating with (non-empty) blocks of ones. Splitting the string before each block of zeros and then considering the structure of a block of zeros followed by a block of ones is often useful.

4. Construct a finite automaton which accepts the language represented by

$$\mathbf{01}((\mathbf{0}(\mathbf{10})^* + \mathbf{111})^* + \mathbf{0}).$$

5. Construct a regular expression for the language accepted by a given DFA