CMSC 330 Formal languages and regular expressions – solutions Fall 2012

1. a. $\{ a, aa, aaa \}$ $(\equiv \mathrm{A})$

   b. $\{ aa, aaa, aaaa, aaaaa, aaaaaa \}$

   c. $\{ a, aa, aaa, aaaa, aaaaa, aaaaaa \}$

   d. $\{ \epsilon, a, aa, aaa, aaaa, aaaaa, aaaaaa, \ldots \}$

   e. $\{ bb \}$ $(\equiv \mathrm{B})$

   f. $\{ bbbb \}$

   g. $\{ bbbbbb \}$

   h. $\{ bb, bbbb, bbbbbb \}$

   i. $\{ \epsilon, bb, bbbb, bbbbbb, bbbbbbbb, bbbbbbbbbb, \ldots \}$

   j. $\{ abbabb, abbaabb, abbaaabb, aabbabb, aabbaabb, aabbaaabb, aaabbabb, aaabbaabb, aaabbaaabb \}$

   k. $\{ aa, aaa, aaaa, abb, aaaaa, aabb, aaaaaa, aaabb, bba, bbaa, bbaaa, bbbb \}$

2. Note that different solutions are possible.

   a. $abab\,(a|b)^*$ ~or~ $abab\,(a^*b^*)^*$

   The first of these is arguably clearer, but both describe the same language.

   b. $(a|b)^*\,abab$ ~or~ $(a^*b^*)^*\,abab$

   c. $(\ ab\,(a|b)^*\,ba\ \mid\ aba\ )$

   d. $b^*ab^*ab^*\,(\ ab^*ab^*ab^*ab^*ab^*\ )^*$

   e. $(\ b^*\ \mid\ (\ b^*ab^*ab^*\ )^*\ \mid\ (\ (a|b)\,(a|b)\ )^*\ )$ ~or~ $(\ b^*\ \mid\ (b^*ab^*ab^*)^*\ \mid\ (\ ab \mid bb \mid ba \mid aa\ )^*\ )$

   f. $(a|b)^*\ aaa\ (a|b)^*$

   g. $b^*\,(\,(\epsilon \mid a \mid aa\,)\,b\,)^*\ (\epsilon \mid a \mid aa)$

3. One thing which is interesting which you may have noticed is that even making a small change in the regular expression can have a big effect on the results it produces.

   a. This regular expression is wrong. The two strings 1 and 11 for example are in the language, so the regular expression should recognize or accept them, but it does not.

   b. This regular expression is wrong. The string 1001 for example is in the language, so the regular expression should recognize or accept it, but it does not. The same would apply to any other valid string which begins or ends with a 1.

   c. This regular expression is wrong. The string 111 for example is not in the language, so the regular expression should not recognize or accept it, but it does. The same would apply to any other invalid string containing three or more consecutive 1s.

d. This regular expression is wrong. The string 001110 for example is not in the language, so the regular expression should not recognize or accept it, but it does. The same would apply to some other invalid strings containing three or four consecutive 1s, since the regular expression will recognize or accept strings in which alternate subsequences of 1s may have three or four consecutive 1s.

e. This regular expression is wrong. The string 0011 for example is in the language, so the regular expression should recognize or accept it, but it does not. The same would apply to any other valid string which contains more than zero 0s and which ends with a 1.

f. This regular expression is wrong. The string 010100 for example is in the language, so the regular expression should recognize or accept it, but it does not. The same would apply to other valid strings, since this regular expression requires that every alternate 0 must be followed by a 0, but not every valid string has that property.

g. This regular expression is correct.

h. This regular expression is also correct.