Exercise sheet 2

Theory of Computation, IDC204

Quantifiers

1. Consider the predicate $P(x, y, z)$:= "$2x + 5y6z = 1$". Which of the following propositions are true:

   (a) $\forall x \forall y \forall z P(x, y, z)$

   (b) $\forall x \forall y \exists z P(x, y, z)$

   (c) $\forall x \exists y \exists z P(x, y, z)$

   (d) $\exists x \exists y \exists z P(x, y, z)$

2. Which of these are true no matter what the predicate P(x,y) is. For each of the others, find an example for which it does not hold.

   (a) $\forall x \exists y P(x, y) \exists y \forall x P(x, y)$

   (b) $\forall x \forall y P(x, y) \forall y \forall x P(x, y)$

   (c) $\exists x \exists y P(x, y) \exists y \exists x P(x, y)$

3. Let $Q(x, y)$ denote the predicate x is a question from Quiz y. Let $D(x)$ denote the predicate x is difficult. Write expressions for the following statements using quantifiers, Q, and D:

   (a) Quiz 3 has exactly one difficult question

   (b) Every quiz will have at least one difficult question

   (c) For some quizzes, every question will be difficult

   (d) For some quizzes, at most one question will be easy

4. One can define a new quantifier $\exists! x P(x)$ to mean that there exists a unique x such that $P(x)$ is true. Show that this can always be expressed by an expression involving only the usual two quantifiers, $\exists$, and $\forall$. (Hint: you want to say that it exists and is unique. What does it mean to say that there is a unique x satisfying a predicate? Try considering its negation if that helps. You may assume that there is some notion of equality between the objects under consideration so that it makes sense to ask if a=b for any given a and b)

Deterministic finite state automata

5. Design a deterministic finite state automaton over the alphabet $\Sigma = 0, 1$ that will accept a string if and only if it
   (a) is the empty string
   (b) is not the empty string
   (c) is precisely the string 111
   (d) begins with the substring 111
   (e) is the empty string or begins with the substring 111
   (f) ends with the substring 111
   (g) begins with 0 and ends with 1
   (h) contains the substring 111

6. Show that if we can design a finite state automaton that accepts strings with certain properties and rejects others, then we can also design an automaton that does the same job but has only one accept state.

7. Show that if a language L over some alphabet Σ is regular (a finite state automaton can be designed to accept strings in the language and reject the others), then even $\Sigma^* \setminus L$ is also regular.

8. Consider the alphabet $\Sigma = \{0, 1, , n, (,), +, , , /\}$. Prove that it is impossible to build a finite state automaton that will reject precisely those strings over Σ that have mis-matched parentheses but accept those with all their parentheses matched. For example, it should reject (5 + 6 - 9 but should accept (5+6)-9

9. Consider the alphabet $\Sigma = 1$ and let $L := \{1^p \mid \text{p is a prime}\}$. Prove that L cannot be recognized by a finite state automaton.

10. Consider a finite state automaton, M, that recognizes a language, L, over an alphabet, Σ. If we now enlarge the alphabet to Σ, i.e. $\Sigma\Sigma$, but consider the same set strings as before as our language, how would you modify M to recognize this language? Note that while it accepts exactly the same strings as before, it must reject a lot more strings.

11. By our definition, a finite state automaton must read the entire input even if it is already clear that the string must be rejected (for example, if the language includes all strings beginning with a 0). Let us define a haltable automaton that can also reject and stop reading the string when there is no point in doing so. Such an automaton may be represented by $(Q, \Sigma, \delta, q_0, F, H)$ where H denotes the subset of reject and halt states and the other symbols carry their usual meanings. It works as usual except that if the automaton enters a state in H, it immediately halts and rejects the string reading no more characters. Show that for any such haltable automaton, the language that it recognizes can also be recognized by a standard deterministic finite state automaton. Therefore, despite the improvement, haltable automata recognize no more languages than the usual deterministic finite state automata can.