CSC 363, Winter 2010 — Short Assignment #4 Solution

Recall that a verifier for a language A takes as input a pair $\langle w, c\rangle$ and accepts or rejects in time that is polynomial in $|w|$. For every $w \in A$, the verifier must accept $\langle w, c\rangle$ for some c, and for every $w \notin A$, the verifier must reject $\langle w, c\rangle$ for all c. We call a c for which the verifier accepts $\langle w, c\rangle$ a "certificate" that w is in A.

1) For this language, the certificate can be two sets of values for the variables in ϕ. The verifier checks that with the first set of values results, ϕ evaluates to 1, and with the second set of values, ϕ evaluates to 0 (ie, $\overline{\phi}$ evaluates to 1). Evaluating a Boolean expression takes linear time if one assumes the values of variables are immediately available, and still takes only quadratic time if getting the value of a variable requires a linear search.

2) The certificate here is a string of length k which M accepts after exactly k^3 steps. The verifier simulates M on this string, and checks that it does indeed accept after k^3 steps. Since the size of the input is greater than k, this takes polynomial time.

3) The certificate here can be a pair of integers whose product is n. The verifier rejects if w is not of the form 1^n, and otherwise counts the 1s to get n, and then verifies that the product of the two integers in the certificate is n.

   Actually, the certificate can be made trivial (so it is just ignored by the verifier), since it has recently been proved that whether an integer in binary notation is composite is decidable in polynomial time. For that matter, the integer here is presented in unary notation, so testing whether it is composite by the obvious method of trying all factors greater than one and less than n takes time that is polynomial in the length of the input.

4) This is a regular language, that is easily recognized in linear time by a Turing Machine. The certificate can be trivial, and just be ignored by the verifier. The verifier will just accepts $\langle w, c\rangle$ for any c if w is in $(01)^*$.

5) For this language, the certificate can be a set of nodes in G_1 and a set of nodes in G_2 of the same size. The verifier first checks that these sets really are the same size, and that they really do specify cliques in G_1 and G_2. It then checks that these cliques are maximal. This can be done in polynomial time by just looking at every node in the graph outside the clique, and rejecting if any such node is connected to every node in the clique (which means that that node could be added to the clique to give a bigger clique).