The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Detailed Solutions
Chapter 7, Problem 7.1, Part 2

Prove that *Positive NAE3SAT* is NP-complete.

Instance: a set of m clauses written over n variables; each clause contains exactly three uncomplemented variables, with no variable appearing more than once in any given clause.
Question: is there a truth assignment that makes one or two variables true in each clause?

Input Encoding and Size: An instance of this problem can be described by m triples of labels, each label identifying a variable. Since there are n variables, it takes $\lceil \log_2 n \rceil$ bits to identify each; thus the input size is $m \cdot \lceil \log_2 n \rceil$ bits. Note, however, that m and n are related: n cannot exceed $3m$ (obviously) and m cannot exceed $\binom{n}{3}$ (for equally obvious reasons). Hence m is a polynomial function of n, so that the input size is simply bounded above and below by a polynomial in n—$n^4/6$ is a safe upper bound and $n/3$ a safe lower bound.

Certificate Description and Verification: An obvious choice for the certificate is a truth assignment, which can be represented by n bits. Verifying the certificate is done clause by clause; for each clause, we check that one or two of the three variables are set to true. This takes at most time proportional to n for each clause (with a rather clumsy mechanism for reading the certificate) and thus at most time $m \cdot n$ to verify the certificate. Since $m = O(n^3)$, the certificate can be verified in time $O(n^4)$, which is polynomial in the input size. Hence our problem is in NP.

Mapping: We prove this problem NP-complete by transforming the known NP-complete problem *NAE3SAT* to it. The sole difference between the two problems is that the variables may appear complemented in clauses of *NAE3SAT*. We transform clause by clause; the new instance will have the same variables as the original instance, plus additional variables. Thus, we have four cases to consider, according to how many complemented literals the *NAE3SAT* clause has. If no complemented literals appear, then we leave the clause unchanged and place it in the set of clauses for *Positive NAE3SAT*. The other three cases require changes.

1. One complemented literal: $\{\overline{a}, b, c\}$. Replace by the clauses $\{a, \alpha, \beta\}$, $\{a, \alpha, \gamma\}$, $\{a, \alpha, \delta\}$, $\{\beta, \gamma, \delta\}$, and $\{b, c, \alpha\}$. The first four clauses force $a = \overline{\alpha}$, so that the last clause becomes exactly the same as the replaced clause.
2. Two complemented literals: $\{\overline{a}, \overline{b}, c\}$. Simply replace by the clause $\{a, b, \overline{c}\}$ with one complemented literal and apply the replacement given above. (Remember that *NAE3SAT* is symmetric.)
3. Three complemented literals: $\{\overline{a}, \overline{b}, \overline{c}\}$. Simply replace by the clause $\{a, b, c\}$. (Remember that *NAE3SAT* is symmetric.)

Resource Requirements: For each clause the transformation takes time linear in the size of the new clause. At worst, each clause gives rise to five new clauses and four new variables; thus, from an original instance of size $m \log n$ we get a new instance of size $5m \log(n + 4m)$ in time linear in the output; but the first is bounded above and below by a polynomial in n and so is the second, so that we have indeed a polynomial transformation.

Correctness: Each clause has its own set of new variables. It is easily verified for each clause (by exhaustion) that the substitute clauses admit a solution *iff* the original one does. Since all clauses are transformed independently, it follows that the entire set of new clauses is satisfiable *iff* the original set is. This completes our proof.