# The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

## *Detailed Solutions*
## *Chapter 7, Problem 7.39*

We want to identify a variable, $x$, such that the clauses imply $(x \to \overline{x})$ and $(\overline{x} \to x)$, a contradiction; in fact, we will prove that a set of 2-CNF clauses is a contradiction if and only if such a variable exists. Since $\to$ is transitive, the implications may be through a chain, $(x \to y_1)(\overline{y}_1 \to y_2) \ldots (\overline{y}_k \to \overline{x})$ and $(\overline{x} \to z_1)(\overline{z}_1 \to z_2) \ldots (\overline{z}_l \to x)$. Recalling that $(a \to b)$ is equivalent to $(\overline{a} + b)$, we then claim:

> A Boolean formula in 2-CNF is not satisfiable if and only if there exist two subsets of clauses $\{(x + x_1) \cdot (\overline{x}_1 + x_2) \ldots (\overline{x}_k + x)\}$ and $\{(\overline{x} + x_{k+1}) \cdot (\overline{x}_{k+1} + x_{k+2}) \ldots (\overline{x}_n + \overline{x})\}$.

The if part is trivial. If we can prove the only if part, then it is obvious that the problem is in NL, since we can guess the series of clauses and then verify their existence in the original expression.

To prove the only if part, we proceed by induction on the number of clauses. The basis is trivial. Suppose that the theorem holds true for all instances of $n$ and fewer clauses. The formula is not satisfiable if and only if neither of its restrictions to $x = true$ and $x = false$ is satisfiable either. But each restriction can be simplified down to a 2-CNF formula with fewer clauses: each clause in which $x$ appears either has a literal set true—in which case it can be dropped—or a literal set to false, in which case we set the other one to true, continuing until we get a contradiction (a clause wit both literals set to true) or until we are left only with clauses of 2 as-yet-undefined literals. In the latter case, the resulting 2-CNF must also be unsatisfiable, so that the inductive hypothesis applies and we get the desired collection of clauses. In the earlier case, another induction proof (this time trivial) on the number of clauses involved in forcing other literals shows that we have the desired pattern of clauses. Hence our proof is complete.

Now, to show that the problem is complete, we transform *Digraph Reachability* to it. Each vertex, $i$, will become a variable, $x_i$; for each arc $(i, j)$, we set up the clause $(\overline{x}_i + x_j)$. Finally, we add the clauses $(\overline{x}_n + \overline{x}_1)$, $(x_1 + x_{n+1})$, and $(x_1 + \overline{x}_{n+1})$ (where $x_{n+1}$ is a new variable). The last two clauses can only be satisfied by setting $x_1 =$ true; using this in the previous clause forces $x_n = \text{false}$. But then, if vertex $n$ is reachable from vertex 1, we have the implication $x_1 \rightarrow x_n$, with $x_1 = \text{true}$ and $x_n = \text{false}$, a contradiction. The converse is equally easy.