A ***literal*** is a Boolean variable or a negated Boolean variable, as in x or $\overline{x}$. A ***clause*** is several literals connected with $\vee$s, as in $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$. A Boolean formula is in ***conjunctive normal form***, called a ***cnf-formula***, if it comprises several clauses connected with $\wedge$s, as in

$$(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6}).$$

It is a ***3cnf-formula*** if all the clauses have three literals, as in

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4) \wedge (x_4 \vee x_5 \vee x_6).$$

Let $3SAT = \{\langle\phi\rangle \mid \phi \text{ is a satisfiable 3cnf-formula}\}$. In a satisfiable cnf-formula, each clause must contain at least one literal that is assigned 1.

The following theorem presents a polynomial time reduction from the *3SAT* problem to the *CLIQUE* problem.

THEOREM 7.32

3SAT is polynomial time reducible to *CLIQUE*.

PROOF IDEA The polynomial time reduction f that we demonstrate from *3SAT* to *CLIQUE* converts formulas to graphs. In the constructed graphs, cliques of a specified size correspond to satisfying assignments of the formula. Structures within the graph are designed to mimic the behavior of the variables and clauses.

PROOF Let ϕ be a formula with k clauses such as

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \quad \cdots \quad \wedge (a_k \vee b_k \vee c_k).$$

The reduction f generates the string $\langle G, k\rangle$, where G is an undirected graph defined as follows.

The nodes in G are organized into k groups of three nodes each called the ***triples***, $t_1, \ldots, t_k$. Each triple corresponds to one of the clauses in ϕ, and each node in a triple corresponds to a literal in the associated clause. Label each node of G with its corresponding literal in ϕ.

The edges of G connect all but two types of pairs of nodes in G. No edge is present between nodes in the same triple and no edge is present between two nodes with contradictory labels, as in x_2 and $\overline{x_2}$. The following figure ill[redacted] this construction when $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

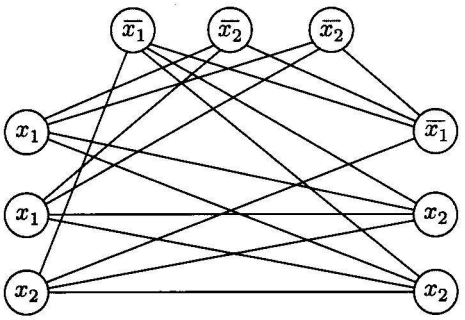

FIGURE 7.33
The graph that the reduction produces from
$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$

Now we demonstrate why this construction works. We show that ϕ is satisfiable iff G has a k-clique.

Suppose that ϕ has a satisfying assignment. In that satisfying assignment, at least one literal is true in every clause. In each triple of G, we select one node corresponding to a true literal in the satisfying assignment. If more than one literal is true in a particular clause, we choose one of the true literals arbitrarily. The nodes just selected form a k-clique. The number of nodes selected is k, because we chose one for each of the k triples. Each pair of selected nodes is joined by an edge because no pair fits one of the exceptions described previously. They could not be from the same triple because we selected only one node per triple. They could not have contradictory labels because the associated literals were both true in the satisfying assignment. Therefore G contains a k-clique.

Suppose that G has a k-clique. No two of the clique's nodes occur in the same triple because nodes in the same triple aren't connected by edges. Therefore each of the k triples contains exactly one of the k clique nodes. We assign truth values to the variables of ϕ so that each literal labeling a clique node is made true. Doing so is always possible because two nodes labeled in a contradictory way are not connected by an edge and hence both can't be in the clique. This assignment to the variables satisfies ϕ because each triple contains a clique node and hence each clause contains a literal that is assigned TRUE. Therefore ϕ is satisfiable.

Theorems 7.31 and 7.32 tell us that, if *CLIQUE* is solvable in polynomial time, so is *3SAT*. At first glance, this connection between these two problems appears quite remarkable because, superficially, they are rather different. But polynomial time reducibility allows us to link their complexities. Now we turn to a definition that will allow us similarly to link the complexities of an entire class of problems.