CHAPTER 4

# Logic

## 4.1 Logic and computing

In this chapter, we introduce the study of logic from a mathematical point of view. Mathematical logic finds applications in many areas of computing. The laws of logic are employed in the design of the digital circuitry in a computer. Logical expressions occur as conditions in the control structures in algorithms and computer programs, and in the commands used for querying databases. Expert systems employing knowledge-based software use rules of logical inference to draw conclusions from known facts. Formal specification documents, which state in a precise way what computer systems are required to do, are written in specification languages, such as Z, which use the theory and notation of symbolic logic.

We begin this chapter by looking at examples involving everyday English sentences. This is followed by an introduction to the more formal mathematical approach used in propositional and predicate logic.

## 4.2 Propositions

The fundamental objects we work with in arithmetic are numbers. In a similar way, the fundamental objects in logic are *propositions*.

*Definition* A *proposition* is a statement that is either true or false. Whichever of these (true or false) is the case is called the *truth value* of the proposition.

Here are some examples of English sentences that are propositions:

'Canberra is the capital of Australia.'
'There are 8 days in a week.'
'Isaac Newton was born in 1642.'
'5 is greater than 7.'
'Every even number greater than 2 can be expressed as the sum of two prime numbers.'

The first and third of these propositions are true, and the second and fourth are false. It is not known at present whether the fifth proposition is true or false.[1]

The following sentences are not propositions:

'Where are you going?'
'Come here.'
'This sentence is false.'

The first sentence is a question and the second is a command, so clearly neither is a proposition.

The third sentence is rather more subtle. It is a *self-referential* statement (that is, it makes a statement about itself). Although at first sight it appears to be a proposition, we run into difficulty when we try to determine whether it is true or false. If we assume it is true, we find the sentence telling us that it is false, which contradicts our assumption. But assuming the sentence is false doesn't work either, because if what the sentence is telling us is false, then the sentence is true! The sentence is an example of a *paradox*, and the only way to avoid the difficulty is simply not to admit the sentence as a proposition. In fact, we will not allow self-referential statements at all in our work in logic. (This does not mean that there is no place for self-reference in logic; in fact, some of the most important results in modern logic involve self-referential propositions.)

Now, what about sentences like these?

'Anne is tall.'
'Ice cream is delicious.'
'$x > 5$.'

In some textbooks, sentences like the first two are not regarded as propositions, because it could be argued that their truth values are not well defined. The first sentence refers to someone called Anne (Anne who?), and states that she is tall (just how tall is 'tall'?). The second sentence is clearly a matter of personal opinion. Quite frankly, it is not worthwhile arguing about whether these sentences are propositions or not. We will feel free to use sentences like these in our examples. In practice, the kinds of propositions that arise when logic is applied to mathematics and computing are always precisely defined anyway, so no difficulties should arise.

The last of the three sentences given above is an example of a *predicate*. A predicate is a statement containing one or more variables; it cannot be assigned a truth value until the values of the variables are specified. We will investigate predicate logic in Section 4.7.

Statements containing variables commonly occur in algorithms and computer programs. For example, an algorithm might contain the statement '$x > 5$' as the condition in a control structure such as an **If-then**. In this case, however, the truth value of the statement is determined

1 The fifth proposition is known as Goldbach's conjecture, after the German mathematician Christian Goldbach (1690–1764).

when the line is executed while the program is being run with a particular set of inputs, so statements of this type can be treated as propositions.

## 4.3 Connectives and truth tables

Logic is not concerned with determining the truth values of propositions of the kind we have seen so far. (The truth value of 'Canberra is the capital of Australia' is a question of geography, not logic.) The next example is different, however:

> 'If Brian and Angela are not both happy, then either Brian is not happy or Angela is not happy.'

We do not need to know whether Brian is happy, or whether Angela is happy, in order to determine whether the proposition is true; with a moment's thought, we can see it *must* be true *because of its logical structure*. In fact, any sentence with the same logical structure must be true; for example:

> 'If 2 and $\sqrt{2}$ are not both rational numbers, then either 2 is not a rational number or $\sqrt{2}$ is not a rational number.'

It is the structure of propositions such as these that we study in propositional logic.

The sentence about Brian and Angela is an example of a *compound proposition*. It is built up from the *atomic* propositions 'Brian is happy' and 'Angela is happy' using the words **and**, **or**, **not** and **if-then**. These words are known as *connectives*. As we will see, the role of connectives in logic is analogous to the role played by operations such as + and × in algebra.

The study of the structure of compound propositions is made easier by the use of symbols for atomic propositions and connectives. We will use lower-case letters such as $p$, $q$ and $r$ to denote atomic propositions. There are five connectives that we will use in our work; they are listed in Table 4.1, together with their symbols.

**Table 4.1**

| *Connective* | *Symbol* |
|---|---|
| **and** | $\wedge$ |
| **or** | $\vee$ |
| **not** | $\neg$ |
| **if-then** | $\rightarrow$ |
| **if-and-only-if** | $\leftrightarrow$ |

The connectives **if-then** and **if-and-only-if** are also known as **implies** and **is-equivalent-to** respectively.

With the exception of **not**, the symbols for these connectives are written between the two operands (the propositions they connect). For example, if $p$ denotes the proposition 'Today is Monday', and $q$ denotes the proposition 'It is raining', then we can write $p \wedge q$ to denote the proposition 'Today is Monday and it is raining'. The symbol $\neg$ (**not**) is placed before the proposition to which it applies; thus, $\neg p$ means 'Today is not Monday'.

The connective **and** can be formally defined by stating the truth value of the proposition $p \wedge q$ for each possible combination of the truth values of the propositions $p$ and $q$. The other connectives can be defined in a similar manner. This information is usually presented in the form of a *truth table*. The truth table for **and** is shown in Table 4.2.

**Table 4.2**

| $p$ | $q$ | $p \wedge q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

The truth values 'true' and 'false' are denoted in the table by T and F respectively. The first two columns of the table contain all four possible combinations of the truth values of the two propositions $p$ and $q$. The truth table reflects our everyday understanding of what **and** means – if $p$ and $q$ are both true then $p \wedge q$ is true, otherwise $p \wedge q$ is false.

The word **or** is used in English in two different ways. If you are offered tea or coffee, you are expected to choose one or the other, but not both! On the other hand, if a discount is available to anyone who is a student or a pensioner, it is presumably available to someone who is both a student and a pensioner. In the first example, **or** is used *exclusively*, while in the second example it is used *inclusively*. By convention, **or** in logic (and in computing and mathematics generally) means 'inclusive-**or**' unless the contrary is stated explicitly. The symbol $\vee$ always means 'inclusive-**or**'; thus $p \vee q$ means '$p$ or $q$ or both'. If we wanted to use 'exclusive-**or**' (or **xor**, as it is sometimes called) in our work, we would have to define it as another connective, but we will not do this here.

The truth table for **or** is shown in Table 4.3.

The truth table for **not** is straightforward; it is shown in Table 4.4.

It takes a little more thought to construct the truth table for **if-then**. Suppose your lecturer makes the following claim:

> 'If you pass this subject, then you will progress to the next year of your course.'

**Table 4.3**

| $p$ | $q$ | $p \vee q$ |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

**Table 4.4**

| $p$ | $\neg p$ |
|---|---|
| T | F |
| F | T |

Consider the different possibilities that could arise. If you pass the subject and progress to the next year of your course, then clearly your lecturer's statement is true. If you pass the subject but don't progress to the next year of your course, then you could accuse your lecturer of making a false statement.

What happens if you fail the subject? Whether or not you progress to the next year of your course, you could not accuse your lecturer of making a false statement. (The statement said only what would happen if you passed, not what would happen if you failed.) We treat the lecturer's statement as true in these cases.

The truth table for **if-then** is shown in Table 4.5.

**Table 4.5**

| $p$ | $q$ | $p \rightarrow q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

While the argument given above is intended to make the truth table for **if-then** appear reasonable, the truth table itself is really the *definition* of the connective **if-then** in logic. It follows that we can assign truth values to some rather strange English sentences. For example, the sentence 'If snow is white then lions roar' is a true proposition (according to the first line of the truth table), even though the whiteness of snow does not cause

lions to roar. Another example is provided by the sentence: 'If Paris is in Germany then grass is purple'; according to the last line of the table, the sentence is true.

It is important to understand the difference between the connective **if-then** and the control structure **If-then** which we encountered in our study of algorithms. When **If-then** is used as a control structure, the sentence following **then** is an *instruction*, not a proposition.

Finally, the connective **if-and-only-if** is true precisely when the two propositions have the same truth value (both true or both false). Its truth table is shown in Table 4.6.

**Table 4.6**

| $p$ | $q$ | $p \Leftrightarrow q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

For example, the proposition 'Birds have three legs if and only if $2 + 2 = 5$' is true, because the two propositions from which it is built up are both false.

## 4.4 Compound propositions

We now have the notation we need in order to be able to write compound propositions in symbolic form. Example 4.4.1 shows how this is done.

**Example 4.4.1** Express the proposition 'Either my program runs and it contains no bugs, or my program contains bugs' in symbolic form.

**Solution** Let $p$ denote the statement: 'My program runs.'
Let $q$ denote the statement: 'My program contains bugs.'

Then the proposition can be written in symbolic form as follows:

$$(p \wedge \neg q) \vee q$$

Notice in this example how parentheses are used to group sub-expressions within the whole expression, just as in arithmetic and algebra. If the sub-expressions had been grouped differently, the meaning would have been different: $p \wedge (\neg q \vee q)$ means 'My program runs, and either it does not contain bugs or it contains bugs.' Parentheses should always be used to group sub-expressions in compound propositions, in

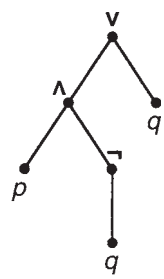
order to avoid any ambiguity in the meaning. (There is one exception – by convention, the symbol ¬ immediately preceding a proposition applies only to that proposition. Thus $\neg p \wedge q$ always means $(\neg p) \wedge q$ and never $\neg(p \wedge q)$.)

In Example 4.4.1, $(p \wedge \neg q) \vee q$ is the symbolic notation for a proposition; we could find its truth value if we knew the truth values of 'My program runs' and 'My program contains bugs'. In the study of logic, it is often useful to analyse expressions such as $(p \wedge \neg q) \vee q$ in which $p$ and $q$ are treated as variables rather than as symbols denoting specific propositions. If we do this, then $(p \wedge \neg q) \vee q$ is no longer a proposition but a *logical expression*; it becomes a proposition only if $p$ and $q$ are replaced by propositions. We can think of a logical expression in the same way as we think of an expression containing a variable $x$ in algebra – the expression can't be evaluated unless $x$ is assigned a value, but this does not prevent us from studying the expression and investigating its properties.

We can analyse the structure of the logical expression $(p \wedge \neg q) \vee q$ in the following way. The entire expression takes the form $A \vee B$, where $A$ is the expression $p \wedge \neg q$ and $B$ is the variable $q$. The connective **or** is the *principal* connective in the original expression. In turn, $p \wedge \neg q$ takes the form $C \wedge D$, where $C$ is the variable $p$, $D$ is the expression $\neg q$, and the principal connective is **and**. Finally, the principal connective in $\neg q$ is **not**. This way of breaking down the structure of an expression is useful in constructing its truth table, as we will see shortly.

Looking ahead to Chapter 11, the structure of the expression $(p \wedge \neg q) \vee q$ can be depicted using an expression tree, as shown in Figure 4.1.

**Figure 4.1**




The truth value of the expression $(p \wedge \neg q) \vee q$ for each possible combination of truth values of $p$ and $q$ can be found by constructing a truth table.

**Example 4.4.2** Construct the truth table for the expression $(p \wedge \neg q) \vee q$.

**Solution** The solution is shown in Table 4.7. The first two columns of the table contain all the possible combinations of the truth values of $p$ and $q$. Column 3 is obtained from Column 2 using the truth table for **not**. Column 4 is obtained from Columns 1 and 3 using the truth table for **and**.

Finally, Column 5, which contains the truth values for the entire expression, is obtained from Columns 4 and 2 using the truth table for **or**.

**Table 4.7**

| $p$ | $q$ | $\neg q$ | $p \wedge \neg q$ | $(p \wedge \neg q) \vee q$ |
|---|---|---|---|---|
| T | T | F | F | T |
| T | F | T | T | T |
| F | T | F | F | T |
| F | F | T | F | F |

Notice that each column is obtained using the truth table for the principal connective in the expression at the top of the column.

If an expression contains three variables ($p$, $q$ and $r$, say), then the table will have eight lines instead of four (there are $2^3 = 8$ different ways of allocating truth values to three expressions), but the method is the same.

Now look again at the proposition we introduced at the beginning of Section 4.3:

> 'If Brian and Angela are not both happy, then either Brian is not happy or Angela is not happy.'

If $p$ and $q$ denote respectively 'Brian is happy' and 'Angela is happy', the proposition can be expressed symbolically in the following way:

$$\neg(p \wedge q) \rightarrow (\neg p \vee \neg q)$$

The truth table for this expression is given in Table 4.8.

**Table 4.8**

| $p$ | $q$ | $p \wedge q$ | $\neg(p \wedge q)$ | $\neg p$ | $\neg q$ | $\neg p \vee \neg q$ | $\neg(p \wedge q) \rightarrow (\neg p \vee \neg q)$ |
|---|---|---|---|---|---|---|---|
| T | T | T | F | F | F | F | T |
| T | F | F | T | F | T | T | T |
| F | T | F | T | T | F | T | T |
| F | F | F | T | T | T | T | T |

The final column of the truth table contains only T. This tells us that the expression is always true, regardless of the truth values of $p$ and $q$.

An expression that is always true, regardless of the truth values of the variables it contains, is called a *tautology*.

Now consider the following proposition:

'It is raining and windy, and it is not raining.'

Even without looking at the weather, we can tell that the proposition is false from its logical structure. We can confirm this by writing the proposition in symbolic form and constructing the truth table for the resulting expression. Using $p$ and $q$ to denote respectively 'It is raining' and 'It is windy', we obtain Table 4.9.

**Table 4.9**

| $p$ | $q$ | $p \wedge q$ | $\neg p$ | $(p \wedge q) \wedge \neg p$ |
|---|---|---|---|---|
| T | T | T | F | F |
| T | F | F | F | F |
| F | T | F | T | F |
| F | F | F | T | F |

An expression that is always false, regardless of the truth values of the variables it contains, is called a *contradiction*.

## 4.5 Logical equivalence

Here is a rather complicated proposition:

'It is not the case that both the input file and the output file are not on the disk.'

The proposition below expresses the same idea more simply:

'Either the input file or the output file is on the disk.'

If we were to express these propositions symbolically, we would expect the resulting logical expressions to have the same truth table. Let $p$ and $q$ denote respectively the propositions 'The input file is on the disk' and 'The output file is on the disk'. Then we have the following result, in which for convenience we have combined the truth tables for the two expressions into a single table (Table 4.10).

**Table 4.10**

| $p$ | $q$ | $\neg p$ | $\neg q$ | $\neg p \wedge \neg q$ | $\neg(\neg p \wedge \neg q)$ | $p \vee q$ |
|---|---|---|---|---|---|---|
| T | T | F | F | F | T | T |
| T | F | F | T | F | T | T |
| F | T | T | F | F | T | T |
| F | F | T | T | T | F | F |

The sixth and seventh columns are the truth tables for the first and second expressions respectively, and we can see that their truth values are the same.

*Definition* Two expressions (composed of the same variables) are *logically equivalent* if they have the same truth values for every combination of the truth values of the variables.

Informally, we could say that two expressions are logically equivalent if they yield the same truth table.

There is a subtle but important distinction between the connective **if-and-only-if** and the concept of logical equivalence. When we write $p \Leftrightarrow q$, we are writing a *single* logical expression. Logical equivalence, on the other hand, is a relationship between *two* logical expressions. The two concepts are related in the following way: two expressions $A$ and $B$ are logically equivalent if and only if the expression $A \Leftrightarrow B$ is a tautology.

Some important questions about logical equivalence arise when we consider expressions of the form $p \rightarrow q$. Such expressions are called *implications*. We investigate these questions now.

*Definitions* The *converse* of $p \rightarrow q$ is $q \rightarrow p$.

The *contrapositive* of $p \rightarrow q$ is $\neg q \rightarrow \neg p$.

**Example 4.5.1** Write down English sentences for the converse and the contrapositive of:

'If 250 is divisible by 4 then 250 is an even number.'

**Solution** The sentence takes the form $p \rightarrow q$, where $p$ denotes '250 is divisible by 4' and $q$ denotes '250 is an even number'.

The converse is $q \rightarrow p$, which we can write in English as follows:

'If 250 is an even number then 250 is divisible by 4.'

The contrapositive is $\neg q \rightarrow \neg p$, which we write as follows:

'If 250 is not an even number then 250 is not divisible by 4.'

Not only is the original proposition in this example a true mathematical statement as it stands; it remains true if 250 is replaced by any other integer. This is also the case for the contrapositive. The converse, however, is false – 250 is an even number, but it is not divisible by 4. Example 4.5.1 suggests that $p \rightarrow q$ is not logically equivalent to its

converse, but that it is logically equivalent to its contrapositive. We can confirm that this is the case by constructing a truth table (Table 4.11).

**Table 4.11**

| $p$ | $q$ | $p \to q$ | $q \to p$ | $\neg q$ | $\neg p$ | $\neg q \to \neg p$ |
|---|---|---|---|---|---|---|
| T | T | T | T | F | F | T |
| T | F | F | T | T | F | F |
| F | T | T | F | F | T | T |
| F | F | T | T | T | T | T |

The columns for $p \to q$ and $\neg q \to \neg p$ are identical to each other, but they differ from the column for $q \to p$. Therefore an implication and its contrapositive are logically equivalent, while an implication and its converse are not.

## 4.6 Laws of logic

We began Section 4.5 with an example of a complicated proposition that we showed to be logically equivalent to a simpler one. Occasions often arise in practice where it is desirable to replace a logical expression with a simpler expression that is logically equivalent to it. For example, we have seen how logical expressions representing propositions can occur in algorithms and computer programs. By writing these expressions as simply as possible, we can make a program more efficient and reduce the chance of error.

In order to be able to simplify logical expressions effectively, we need to establish a list of pairs of expressions that are logically equivalent. We will use the symbol $\equiv$ placed between two expressions to indicate that they are equivalent. A statement of the form $P \equiv Q$ where $P$ and $Q$ are logical expressions is called a *law* of logic. A list of the most important laws of logic is given in Table 4.12.

The first two laws in Table 4.12 allow the connectives **if-then** and **if-and-only-if** to be removed from any expression containing them. The remaining laws involve just the connectives **and**, **or** and **not**. Except for the double negation law, these laws occur in pairs, in which the second law in the pair can be obtained from the first by interchanging $\wedge$ with $\vee$ and T with F. (Here, T means any true proposition, and F means any false proposition.) The second law in each pair is said to be the *dual* of the first, and vice versa. The double negation law is its own dual.

The list of laws in Table 4.12 is very comprehensive, and it might appear rather daunting at first. However, many of the laws are obvious after a moment's thought, such as the double negation law ('It is not the case that it is not raining' is a convoluted way of saying 'It is raining'),

**Table 4.12**

| *Law(s)* | | *Name* |
|---|---|---|
| $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$ | | Equivalence law |
| $p \rightarrow q \equiv \neg p \vee q$ | | Implication law |
| $\neg \neg p \equiv p$ | | Double negation law |
| $p \wedge p \equiv p$ | $p \vee p \equiv p$ | Idempotent laws |
| $p \wedge q \equiv q \wedge p$ | $p \vee q \equiv q \vee p$ | Commutative laws |
| $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ | $(p \vee q) \vee r \equiv p \vee (q \vee r)$ | Associative laws |
| $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ | $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ | Distributive laws |
| $\neg(p \wedge q) \equiv \neg p \vee \neg q$ | $\neg(p \vee q) \equiv \neg p \wedge \neg q$ | de Morgan's laws |
| $p \wedge \mathrm{T} \equiv p$ | $p \vee \mathrm{F} \equiv p$ | Identity laws |
| $p \wedge \mathrm{F} \equiv \mathrm{F}$ | $p \vee \mathrm{T} \equiv \mathrm{T}$ | Annihilation laws |
| $p \wedge \neg p \equiv \mathrm{F}$ | $p \vee \neg p \equiv \mathrm{T}$ | Inverse laws |
| $p \wedge (p \vee q) \equiv p$ | $p \vee (p \wedge q) \equiv p$ | Absorption laws |

and the idempotent laws ('I am happy and I am happy' just means 'I am happy'). The less obvious laws can be checked using truth tables.

Notice that some of the laws take the same form as laws of ordinary algebra, with = replaced by $\equiv$, $\times$ by $\wedge$, + by $\vee$, 1 by T and 0 by F. The commutative, associative and identity laws are of this type, and so is the first of the distributive laws, because it corresponds to the familiar rule for 'multiplying out brackets': $x \times (y + z) = (x \times y) + (x \times z)$. This is not the case with the second distributive law, because $x + (y \times z) = (x + y) \times (x + z)$ is not a law of algebra. Working with the laws of logic can sometimes have the same 'feel' as doing algebra with numbers, but it is essential to make sure that each step in the solution to a problem can be justified using one of the laws of logic.

**Example 4.6.1** Use a truth table to verify the first de Morgan's law: $\neg(p \wedge q) \equiv \neg p \vee \neg q$.

**Solution** Note that the law can be paraphrased as follows: 'If it is not the case that $p$ and $q$ are both true, then that is the same as saying that at least one of $p$ or $q$ is false.'

The truth table is shown in Table 4.13. The columns for $\neg(p \wedge q)$ and $\neg p \vee \neg q$ are identical, and therefore the two expressions are logically equivalent.

**Table 4.13**

| $p$ | $q$ | $p \wedge q$ | $\neg(p \wedge q)$ | $\neg p$ | $\neg q$ | $\neg p \vee \neg q$ |
|---|---|---|---|---|---|---|
| T | T | T | F | F | F | F |
| T | F | F | T | F | T | T |
| F | T | F | T | T | F | T |
| F | F | F | T | T | T | T |

The next example illustrates how the laws of logic can be applied to the problem of simplifying a logical expression. Starting with the given expression, a sequence of equivalent expressions is obtained by applying one of the laws at each step. You need to keep in mind that 'applying a law' often means replacing the variables in the law with logical expressions in order to put it into the required form.

**Example 4.6.2** Use the laws of logic to simplify the expression:

$$p \vee \neg(\neg p \rightarrow q)$$

**Solution** As this is our first example of simplifying an expression using the laws of logic, the solution is given in more detail than would normally be shown in practice, to demonstrate at each step exactly how the relevant law of logic from Table 4.12 has been applied.

| | | |
|---|---|---|
| $p \vee \neg(\neg p \rightarrow q)$ | $\equiv p \vee \neg(\neg \neg p \vee q)$ | implication law (with $\neg p$ in place of $p$) |
| | $\equiv p \vee \neg(p \vee q)$ | double negation law |
| | $\equiv p \vee (\neg p \wedge \neg q)$ | second de Morgan's law |
| | $\equiv (p \vee \neg p) \wedge (p \vee \neg q)$ | second distributive law (with $\neg p$ and $\neg q$ in place of $q$ and $r$ respectively) |
| | $\equiv \mathrm{T} \wedge (p \vee \neg q)$ | second inverse law |
| | $\equiv (p \vee \neg q) \wedge \mathrm{T}$ | first commutative law (with T and $(p \vee \neg q)$ in place of $p$ and $q$ respectively) |
| | $\equiv p \vee \neg q$ | first identity law (with $(p \vee \neg q)$ in place of $p$) |

There are no hard and fast rules for determining which law to apply at each step in this type of problem. If the connectives $\leftrightarrow$ or $\rightarrow$ appear in the given expression, they should be eliminated using the first two laws. After that, it can sometimes be a matter of trying a law to see if it helps to simplify the expression, and then trying another if it doesn't.

An important practical application of the laws of logic is the simplification of logical expressions in algorithms. De Morgan's laws are often particularly useful in this type of problem.

**Example 4.6.3** An algorithm contains the following line:

**If** not($x > 5$ and $x \leq 10$) **then** ...

How could this be written more simply?

**Solution** Apply the first de Morgan's law: $\neg[(x > 5) \wedge (x \leq 10)]$ is equivalent to $\neg(x > 5) \vee \neg(x \leq 10)$, which in turn is equivalent to $(x \leq 5) \vee (x > 10)$. The line of the algorithm can therefore be written:

**If** $x \leq 5$ or $x > 10$ **then** ...

The next example shows how a type of problem that we previously dealt with using truth tables can also be solved using the laws of logic.

**Example 4.6.4** Use the laws of logic to show that $[(p \to q) \wedge \neg q] \to \neg p$ is a tautology.

**Solution**

| | | |
|---|---|---|
| $[(p \to q) \wedge \neg q] \to \neg p$ | $\equiv \neg[(\neg p \vee q) \wedge \neg q] \vee \neg p$ | implication law (twice) |
| | $\equiv \neg[\neg q \wedge (\neg p \vee q)] \vee \neg p$ | first commutative law |
| | $\equiv \neg[(\neg q \wedge \neg p) \vee (\neg q \wedge q)] \vee \neg p$ | first distributive law |
| | $\equiv \neg[(\neg q \wedge \neg p) \vee (q \wedge \neg q)] \vee \neg p$ | first commutative law |
| | $\equiv \neg[(\neg q \wedge \neg p) \vee \mathrm{F}] \vee \neg p$ | first inverse law |
| | $\equiv \neg(\neg q \wedge \neg p) \vee \neg p$ | second identity law |
| | $\equiv (\neg \neg q \vee \neg \neg p) \vee \neg p$ | first de Morgan's law |
| | $\equiv (q \vee p) \vee \neg p$ | double negation law (twice) |
| | $\equiv q \vee (p \vee \neg p)$ | second associative law |
| | $\equiv q \vee \mathrm{T}$ | second inverse law |
| | $\equiv \mathrm{T}$ | second annihilation law |

Therefore $[(p \to q) \wedge \neg q] \to \neg p$ is a tautology.

Both methods have advantages and disadvantages. The truth table method can be lengthy, but it is a mechanical procedure guaranteed to lead to the answer eventually. Applying the laws of logic can be more difficult, because it is not always easy to decide which law should be

applied in a given situation. On the other hand, the laws of logic will often lead to a solution more quickly.

The next example shows how the laws of logic can be used to determine the validity of an argument.

**Example 4.6.5** Determine whether the following argument is valid:

> 'The file is either a binary file or a text file. If it is a binary file then my program won't accept it. My program will accept the file. Therefore the file is a text file.'

**Solution** An argument of this type consists of some *premises* (in this example, the first three sentences), which together are supposed to imply the *conclusion* (the last sentence). The argument takes the form of the logical expression:

$$(P_1 \wedge P_2 \wedge P_3) \rightarrow Q$$

where $P_1$, $P_2$ and $P_3$ are the premises, and $Q$ is the conclusion. (There is no ambiguity in writing $P_1 \wedge P_2 \wedge P_3$ without brackets, because the connective $\wedge$ obeys the associative law.) If the argument is valid, the expression should be a tautology.

Let $p$ denote the proposition 'The file is a binary file', let $q$ denote 'The file is a text file', and let $r$ denote 'My program will accept the file'. Then:

$$\begin{aligned} P_1 &\equiv p \vee q \\ P_2 &\equiv p \rightarrow \neg r \\ P_3 &\equiv r \\ Q &\equiv q \end{aligned}$$

The argument now takes the form:

$$[(p \vee q) \wedge (p \rightarrow \neg r) \wedge r] \rightarrow q$$

We can find out whether this expression is a tautology either by constructing a truth table or by trying to simplify it using the laws of logic. Since a truth table would require eight rows and a large number of columns, and would be fairly tedious to construct, we try the latter approach.

$$\begin{aligned} [(p \vee q) \wedge (p \rightarrow \neg r) \wedge r] \rightarrow q &\equiv \neg[(p \vee q) \wedge (\neg p \vee \neg r) \wedge r] \vee q \\ &\equiv \neg[(p \vee q) \wedge r \wedge (\neg p \vee \neg r)] \vee q \\ &\equiv \neg\{(p \vee q) \wedge [(r \wedge \neg p) \vee (r \wedge \neg r)]\} \vee q \\ &\equiv \neg\{(p \vee q) \wedge [(r \wedge \neg p) \vee \mathrm{F}]\} \vee q \\ &\equiv \neg[(p \vee q) \wedge r \wedge \neg p] \vee q \\ &\equiv \neg[\neg p \wedge (p \vee q) \wedge r] \vee q \\ &\equiv \neg\{[(\neg p \wedge p) \vee (\neg p \wedge q)] \wedge r\} \vee q \end{aligned}$$

$$\equiv \neg\{[\mathrm{F} \vee (\neg p \wedge q)] \wedge r\} \vee q$$
$$\equiv \neg(\neg p \wedge q \wedge r) \vee q$$
$$\equiv \neg(q \wedge \neg p \wedge r) \vee q$$
$$\equiv \neg q \vee \neg(\neg p \wedge r) \vee q$$
$$\equiv \neg q \vee q \vee \neg(\neg p \wedge r)$$
$$\equiv \mathrm{T} \vee \neg(\neg p \wedge r)$$
$$\equiv \mathrm{T}$$

The proposition is a tautology, so the argument is valid.

## 4.7 Predicate logic

Propositional logic provides a useful setting in which we can analyse many types of logical argument. There are situations, however, where propositional logic is inadequate, because it cannot deal with the logical structure that is sometimes present *within* atomic propositions.

Consider the following arguments:

> 'All even numbers are integers. 8 is an even number. Therefore 8 is an integer.'
>
> 'It is not true that all prime numbers are odd. Therefore there must be at least one prime number that is not odd.'

Both of these arguments appear to be perfectly valid on the basis of everyday reasoning, yet if we try to show their validity using propositional logic we run into difficulties.

In the first argument, the atomic propositions are 'All even numbers are integers', '8 is an even number', and '8 is an integer'. The argument takes the following form:

$$(p \wedge q) \rightarrow r$$

This expression is false if $p$ and $q$ are true and $r$ is false, so it is not a tautology.

Similarly, if we take the atomic propositions in the second argument to be 'All prime numbers are odd' and 'There must be at least one prime number that is not odd', the argument takes the form:

$$\neg p \rightarrow q$$

This expression is also not a tautology, because it is false if $p$ and $q$ are both false.

In order to be able to analyse arguments such as these, we need to look at the logical structure within atomic propositions. Predicate logic allows us to do this.

*Definition* A *predicate* is a statement containing one or more variables. If values are assigned to all the variables in a predicate, the resulting statement is a proposition.

For example, '$x < 5$' is a predicate, where $x$ is a variable denoting any real number. If we substitute a real number for $x$, we obtain a proposition; for example, '$3 < 5$' and '$6 < 5$' are propositions with truth values T and F respectively.

A variable need not be a number. For example, '$x$ is an employee of the Ezisoft Software Company' becomes a proposition with a well defined truth value when $x$ is replaced by a person's name: 'Frederick Firestone[2] is an employee of the Ezisoft Software Company.'

There are other ways of obtaining a proposition from a predicate apart from assigning values to the variables. For example, consider the predicate '$x < 5$ or $x \geq 5$'. This predicate is true no matter what value we substitute for $x$, so we can form a true proposition by writing:

'For all $x$, $x < 5$ or $x \geq 5$'.

If we had used '$x < 5$' as the predicate instead of '$x < 5$ or $x \geq 5$', we would have obtained:

'For all $x$, $x < 5$',

which is also a proposition (albeit a false one).

While the predicate '$x < 5$' is not always true, it *is* true for some values of $x$, so we can form a true proposition by writing:

'There exists an $x$ such that $x < 5$.' (Here, 'an' means 'at least one'.)

The expressions 'for all' and 'there exists' are called *quantifiers*. The process of applying a quantifier to a variable is called *quantifying* the variable. A variable which has been quantified is said to be *bound*. For example, the variable $x$ in 'There exists an $x$ such that $x < 5$' is bound by the quantifier 'there exists'. A variable that appears in a predicate but is not bound is said to be *free*.

We now introduce a notation that will allow us to write predicates and quantifiers symbolically. We will use capital letters to denote predicates. A predicate $P$ that contains a variable $x$ can be written symbolically as $P(x)$. A predicate can contain more than one variable; a predicate $P$ with two variables, $x$ and $y$ for example, can be written $P(x,y)$. In general, a predicate with $n$ variables, $x_1, x_2, ..., x_n$, can be written $P(x_1, x_2, ..., x_n)$.

The quantifiers 'for all' and 'there exists' are denoted by the symbols $\forall$ and $\exists$ respectively. With this notation, expressions containing predicates and quantifiers can be written symbolically.

2 The name 'Frederick Firestone' appealed to the author when he saw it on a freeway exit sign in Colorado. (The sign actually marks the exit to two towns called Frederick and Firestone.)

**Example 4.7.1** Write in symbols: 'There exists an $x$ such that $x < 4$.'

**Solution** Let $P(x)$ be '$x < 4$'. Then the proposition can be written:

$$\exists x P(x)$$

**Example 4.7.2** Write in symbols: 'For all $x$, $x < 5$ or $x \geq 5$'.

**Solution** Let $P(x)$ be '$x < 5$', and let $Q(x)$ be '$x \geq 5$'. Then the proposition can be written:

$$\forall x[P(x) \vee Q(x)]$$

If we use the fact that $Q(x)$ is equivalent to $\neg P(x)$, we can also write:

$$\forall x[P(x) \vee \neg P(x)]$$

Here is a more complicated example, using a predicate with two variables.

**Example 4.7.3** Write the following two propositions in symbols:

'For every number $x$ there is a number $y$ such that $y = x + 1$.'
'There is a number $y$ such that, for every number $x$, $y = x + 1$.'

**Solution** Let $P(x,y)$ denote the predicate '$y = x + 1$'. The first proposition is:

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

The second proposition is:

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

Note carefully the difference in meaning between the two propositions in Example 4.7.3. In the first proposition the value of $y$ can depend on $x$ (that is, different values of $x$ can give different values of $y$), whereas in the second proposition it cannot. In fact, the first proposition is a true statement about numbers, while the second is a false statement. This example shows that the order in which the quantifiers appear can affect the meaning.

The following example illustrates a practical problem in which the notation of predicate logic is useful.

**Example 4.7.4** In the specification of a system for booking theatre seats, $B(p,s)$ denotes the predicate 'person $p$ has booked seat $s$'. Write the following sentences in symbolic form:

(a) Seat $s$ has been booked.
(b) Person $p$ has booked a (that is, at least one) seat.
(c) All the seats are booked.
(d) No seat is booked by more than one person.

**Solution**

(a) $\exists p B(p,s)$
(b) $\exists s B(p,s)$
(c) $\forall s \exists p B(p,s)$
(d) If no seat is booked by more than one person, then $B(p,s)$ and $B(q,s)$ cannot both be true unless $p$ and $q$ denote the same person. In symbols:

$$\forall s \forall p \forall q \{[B(p,s) \wedge B(q,s)] \rightarrow (p = q)\}$$

It would be possible to formulate various laws of logic involving the two quantifiers $\forall$ and $\exists$ and the connectives we introduced earlier. Here we will look just at the relationship between the two quantifiers and the connective **not**.

Suppose we want to apply the connective **not** to the following proposition:

'All swans are black.'

Applying **not** to a proposition is called *negating* the proposition.

The original proposition can be written in symbols:

$$\forall x P(x)$$

where $P(x)$ is the predicate 'Swan $x$ is black'.

Here is one way of forming the negation:

'It is not true that all swans are black.'

Or, more simply:

'Not all swans are black.'

We can write this proposition in symbols as follows:

$$\neg[\forall x P(x)]$$

Note that it would be incorrect to give the negation as 'All swans are not black'. This would be saying something different – that there are *no* black swans.

There is another way of saying that not all swans are black; we can say that *there must be at least one swan that is not black*. This gives us an alternative way of expressing the negation of the original proposition:

'There is a swan that is not black.'

In symbols:

$$\exists x[\neg P(x)]$$

By comparing this form of the negation with the original proposition 'All swans are black', we can see that forming the negation corresponds to *negating the predicate and changing the quantifier*. We can express this observation as a law of predicate logic:

$$\neg[\forall x P(x)] \equiv \exists x[\neg P(x)]$$

There is a second law, which can be thought of as the dual of the first, for negating a proposition containing 'there exists'. It also corresponds to negating the predicate and changing the quantifier:

$$\neg[\exists x P(x)] \equiv \forall x[\neg P(x)]$$

For example, the negation of 'There is a number $x$ such that $x^2 = 2$' is 'For every number $x$, $x^2 \neq 2$'.

**Example 4.7.5** Write down the negation of the following proposition:

'For every number $x$ there is a number $y$ such that $y < x$.'

**Solution** Write the negation in symbols and simplify it using the laws of logic:

$$\begin{aligned}\neg[\forall x \exists y(y < x)] &\equiv \exists x\{\neg[\exists y(y < x)]\} \\ &\equiv \exists x \forall y[\neg(y < x)] \\ &\equiv \exists x \forall y(y \geq x)\end{aligned}$$

Write the answer as an English sentence:

'There is a number $x$ such that, for every number $y$, $y \geq x$.'

It is important in a problem such as this to check that the answer makes sense in terms of what the proposition and its negation mean, rather than just mechanically applying the laws of logic. In Example 4.7.5, the original proposition is a true mathematical statement about real numbers, while its negation is a false statement.

The rule for negating the quantifier 'for all' can be used to verify one of the arguments we quoted as an example at the beginning of this section:

> 'It is not true that all prime numbers are odd. Therefore there must be at least one prime number that is not odd.'

Let $P(x)$ denote the predicate '$x$ is a prime number', and let $Q(x)$ denote the predicate '$x$ is odd'. The proposition 'all prime numbers are odd' can be rephrased as 'for all $x$, if $x$ is a prime number then $x$ is odd', and written in symbolic form in the following way:

$$\forall x[P(x) \to Q(x)]$$

Therefore the first sentence of the argument can be written in symbols as follows:

$$\neg\{\forall x[P(x) \rightarrow Q(x)]\}$$

If we apply the negation rule and the laws of propositional logic we met in Section 4.6, we obtain the following equivalent expressions:

$$\begin{aligned}\neg\{\forall x[P(x) \rightarrow Q(x)]\} &\equiv \exists x\{\neg[P(x) \rightarrow Q(x)]\} \\ &\equiv \exists x\{\neg[\neg P(x) \vee Q(x)]\} \\ &\equiv \exists x[\neg\neg P(x) \wedge \neg Q(x)] \\ &\equiv \exists x[P(x) \wedge \neg Q(x)]\end{aligned}$$

The last line reads: 'There is a number $x$ such that $x$ is a prime number and $x$ is not odd', which we can rephrase as: 'There must be at least one prime number that is not odd.'

## 4.8 Proof techniques

Proofs play a central role in mathematics. The study of any branch of mathematics begins with a set of *axioms*, or postulates: statements that we assume are true without proof, and which serve to define that particular branch of mathematics. For example, you may have encountered the axioms of Euclidean geometry; one of the axioms is the statement that a line can be drawn through any two points. With the axioms in place, we can proceed to develop theorems. A *theorem* is a statement that we accept as true because we can deduce it from the axioms, or from other theorems we have already established, using logical reasoning. A *proof* is a logical argument used to establish the truth of a theorem.

Mathematical proofs can be presented in a very formal style: the axioms and theorems are written using the notation of propositional and predicate logic, and rules of deduction are applied at each step. Such an approach is appropriate in some circumstances, for example, in the study of automated theorem proving using a computer. However, proofs are more commonly presented as ordinary sentences, using a mixture of words and mathematical notation. In this section, we introduce some techniques for constructing proofs. A further technique, induction, is introduced in Chapter 7.

Proving theorems is a skill to be developed. Unlike much of the mathematics you have studied, it is not a matter of learning a technique that is guaranteed to work, and applying it to the problem at hand. Often, an attempted proof using one technique will fail, and another method must be tried. Finding a method that works requires skill and ingenuity. We shall illustrate some common proof techniques by means of examples; the rest is a matter of practice!

**Example 4.8.1** Prove that the sum of any two even numbers is an even number.

**Solution** Let $x$ and $y$ be even numbers. Then $x = 2m$ for some integer, $m$, and $y = 2n$ for some integer, $n$. Therefore, $x + y = 2m + 2n = 2(m + n)$. Since $m$ and $n$ are integers, so is $m + n$, so $2(m + n)$ is even. Hence, $x + y$ is even.

The solution to Example 4.8.1 is an example of the simplest type of proof, known as a *direct* proof. A direct proof starts with the premises, and proceeds by logical deduction until the required conclusion is reached.

How do you know what to do at each step in the reasoning? At some points, there is really only one sensible option; for example, after "Let $x$ and $y$ be even numbers", the next step is to invoke the definition of an even number. The one point in the proof where it might not be obvious what to do next – arguably the key step – occurs after "$x + y = 2m + 2n$". Here, we need to recall what it is that we want to prove: that $x + y$ is even. Therefore, our aim is to express $2m + 2n$ in the form $2 \times$ something. We see that we can do this by taking 2 out as a common factor.

Here is another example of a direct proof.

**Example 4.8.2** Prove that, if $x$ is any number of the form $3k + 1$ for some integer, $k$, then $x^2$ is also of that form.

**Solution** Let $x = 3k + 1$. Then $x^2 = (3k + 1)^2$. Our aim now is to express $(3k + 1)^2$ in the form $3 \times$ (something) $+ 1$. We try expanding $(3k + 1)^2$. (There is no guarantee that this strategy will work – we won't know until we try.) On expanding, we obtain $(3k + 1)^2 = 9k^2 + 6k + 1$. We can get this into the form we want by taking a factor of 3 out of the first two terms: $9k^2 + 6k + 1 = 3(3k^2 + 2k) + 1$. To complete the proof, we let $n = 3k^2 + 2k$. Then $n$ is an integer, and $x^2 = 3n + 1$, so $x^2$ is of the required form.

Not every theorem yields to the method of direct proof. The next example illustrates another technique.

**Example 4.8.3** Prove that, if $x^2$ is even, then $x$ is even.

**Solution** We try a direct approach first. Let $x^2$ be even. Then $x^2 = 2n$ for some integer, $n$. But now we run into difficulties; we want to prove something about $x$, but in order to isolate $x$ we would have to take the square root of both sides of the equation. After that, there appears to be no way of proceeding further.

Here is another approach, which sometimes works when a direct approach fails. Recall that an implication and its contrapositive are logically equivalent. It follows that if we can prove the contrapositive of an implication, then we have proved the implication. In this example, the

contrapositive is: if $x$ is not even then $x^2$ is not even. Put another way, if $x$ is odd then $x^2$ is odd.

Let $x$ be odd. Then $x = 2n + 1$ for some integer, $n$. Therefore, $x^2 = (2n + 1)^2 = 4n^2 + 4n + 1 = 2(2n^2 + 2n) + 1 = 2k + 1$, where $k = 2n^2 + 2n$. Since $k$ is an integer, $2k + 1$ is odd, so $x^2$ is odd. This completes the proof.

**Example 4.8.4** Prove that, if $xy = z^2$ and $x < z$ then $y > z$, for any positive numbers $x$, $y$ and $z$.

**Solution** A direct proof of this result can be constructed, but we will illustrate another approach here. Suppose the conclusion is *not* true. In other words, suppose $xy = z^2$ and $x < z$, but $y$ is not greater than $z$. Then $y \leq z$. Multiply both sides of this inequality by $x$ (which we may do, because $x$ is positive): $xy \leq xz$. Hence, $z^2 \leq xz$. Now, divide both sides of this last inequality by $z$ (again, this is a valid operation, because $z$ is positive), to obtain $z \leq x$. But this can be written as $x \geq z$, which contradicts the fact that $x < z$.

What has gone wrong? Nothing; we made an assumption – that $y$ is not greater than $z$ – and we have deduced a contradiction. If an assumption leads to a contradiction, there is only one conclusion we can draw: the assumption must have been wrong. We conclude that $y$ *is* greater than $z$.

The method used in the solution of Example 4.8.4 is called *proof by contradiction*. In a proof by contradiction, we start by assuming that the conclusion is false, and deduce a contradictory statement. Since the assumption led to a contradiction, we conclude that the assumption was wrong, and hence that the theorem is true.

In each example in this section, we have proved a statement about all numbers of a certain type. Sometimes, however, we want to *disprove* a statement of this form. In order to do this, all we need to do is to find just one number for which the statement is false. Such a number is called a *counterexample*.

**Example 4.8.5** Disprove the statement: Every natural number can be expressed in the form $x^2 + y^2$, where $x$ and $y$ are non-negative integers.

**Solution** We show that 3 is a counterexample. In other words, we show that there are no non-negative integers $x$ and $y$ such that $x^2 + y^2 = 3$.

If $x \leq 1$ and $y \leq 1$, then $x^2 + y^2 \leq 2$, so $x^2 + y^2$ cannot equal 3 in this case. If either $x \geq 2$ or $y \geq 2$, then $x^2 + y^2 \geq 4$, so $x^2 + y^2$ cannot equal 3 in this case either. Therefore, 3 cannot be expressed in the form $x^2 + y^2$.

Many of the ideas we have met in this chapter play a fundamental role in computing and mathematics, and they will reappear frequently in one form or another in the following chapters. In particular, we will see in Chapter 8 how the laws of logic (in the guise of Boolean algebra) can be used to study the design of the circuitry on which modern digital computers are based.

## EXERCISES

**1** Express the following propositions in symbolic form, and identify the principal connective:

(a) Either Karen is studying computing and Minh is not studying mathematics, or Minh is studying mathematics.

(b) It is not the case that if it is sunny then I will carry an umbrella.

(c) The program will terminate if and only if the input is not numeric or the escape key is pressed.

(d) If $x = 7$ and $y \neq 4$ and $z = 2$, then if it is not true that either $y = 4$ or $z \neq 2$ then $x = 7$ or $z = 2$.

(Assume that this sentence arises in a context in which $x$, $y$ and $z$ have been assigned values, so that it is a genuine proposition.)

**2** Let $p$ and $q$ denote respectively the propositions 'It is snowing' and 'I will go skiing'. Write down English sentences corresponding to the following propositions:

(a) $\neg p \wedge q$

(b) $p \rightarrow q$

(c) $\neg q \rightarrow p$

(d) $(p \vee \neg q) \wedge p$

**3** (a) Construct the truth table for the connective **xor** with symbol $\oplus$, where $p \oplus q$ means 'either $p$ or $q$ but not both'.

(b) Construct a truth table to show that $p \oplus q$ is logically equivalent to $(p \vee q) \wedge \neg(p \wedge q)$.

**4** Write down English sentences for the converse and contrapositive of the following propositions:

(a) If the input file exists, then an error message is not generated.

(b) If the database is not accessible, then my program cannot run.

(c) If my program contains no bugs, then it produces correct output.

**5** Write down English sentences corresponding to the converse and the contrapositive of $p \rightarrow q$, where $p$ and $q$ are defined in Exercise 2.

**6** Construct truth tables for the following expressions. In each case, state whether the expression is a tautology, a contradiction, or neither.

(a) $\neg(p \vee \neg q) \vee p$

(b) $[p \rightarrow (p \wedge q)] \rightarrow \neg q$

(c) $(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$

(d) $[(p \wedge r) \vee (q \wedge r)] \rightarrow (p \rightarrow \neg q)$

**7** Let $P$ and $Q$ denote two logical expressions. If $P$ is false for a particular set of truth values of the variables, then $P \wedge Q$ must be false for that set of values, so there is no need to find the truth value of $Q$.

(a) State a similar rule involving $P \vee Q$.

(b) Using these two rules as short-cuts, construct the truth tables for the following expressions. (The rules mean that some of the entries in the table may be left blank, but the last column must still be complete.)

(i) $[\neg(p \wedge q) \wedge (p \vee \neg r)] \wedge [(p \wedge r) \vee \neg q]$

(ii) $\neg[\neg p \wedge (q \vee r)] \vee (\neg p \wedge \neg r)$

**8** Use truth tables to show that $\neg(p \vee \neg q)$ and $\neg p \wedge q$ are logically equivalent.

**9** Using truth tables, prove the following laws of logic:

(a) $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

(b) $p \wedge (p \vee q) \equiv p$

**10** Use the laws of logic to simplify the following expressions as far as possible:

(a) $(p \vee \neg q) \wedge (p \vee q)$

(b) $\neg[p \rightarrow \neg(p \wedge q)]$

(c) $\neg[p \vee (q \wedge \neg p)]$

(d) $[(p \leftrightarrow q) \rightarrow \neg(r \rightarrow p)] \vee (r \rightarrow \neg q)$

**11** An algorithm contains the following line:

**If** not($x \geq 3$ and $x < 6$) **then** ...

How could this be written more simply?

**12** Rewrite the following pseudocode using a **While-do** in place of the **Repeat-until**:

1. $n \leftarrow 0$
2. *term* $\leftarrow 1$
3. *sum* $\leftarrow 0$
4. **Repeat**
   4.1. $n \leftarrow n + 1$
   4.2. *term* $\leftarrow$ *term* / 2
   4.3. *sum* $\leftarrow$ *sum* + *term*
   **until** *term* $< 0.001$ or $n = 100$

**13** Use the laws of logic to classify the following expressions as tautologies or contradictions:

(a) $(p \wedge \neg q) \vee (\neg p \vee q)$

(b) $[p \rightarrow (q \rightarrow p)] \leftrightarrow (p \wedge \neg p)$

(c) $[p \wedge (p \rightarrow q)] \rightarrow q$

**14** Express the following argument in symbolic form and test its validity using the laws of logic:

'If $n > 10$ when the subroutine call statement is reached, then the subroutine is called. The subroutine is called. Therefore $n > 10$ when the subroutine call statement is reached.'

**15** Express the following argument in symbolic form and test its validity using the laws of logic:

'Sandra is studying Computing or Sandra is not studying Accounting. If Sandra is studying Accounting then Sandra is not studying Computing. Therefore Sandra is studying Computing.'

**16** Find an expression that is logically equivalent to $p \vee q$ but which uses only the connectives **and** and **not**.

**17** The connective **nand**, with symbol | (sometimes called the *Sheffer stroke*), is defined by the truth table shown in Table 4.14.

**Table 4.14**

| $p$ | $q$ | $p \mid q$ |
|---|---|---|
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | T |

(a) Find an expression that is logically equivalent to $\neg p$ using only the connective **nand**.

(b) Find an expression that is logically equivalent to $p \wedge q$ using only the connective **nand**.

(c) Find an expression that is logically equivalent to $p \vee q$ using only the connective **nand**.

(This exercise shows that any expression built up using the connectives **and**, **or** and **not** can be converted to a logically equivalent expression using just the connective **nand**.)

**18** Write the following propositions symbolically in the notation of predicate logic, and state their truth values:

(a) 'There is a real number $x$ such that $x^2 - 3x + 2 = 0$.'

(b) 'For every real number $x$ there is a real number $y$ such that $x = y^2$.'

**19** Write the negations of the propositions in Exercise 18 in symbolic form and in English.

**20** In the design specification of a library borrowing system, $B(p,b)$ denotes the predicate 'person $p$ has borrowed book $b$', and $O(b)$ denotes the predicate 'book $b$ is overdue'.

Write the following sentences in symbolic form:

(a) Person $p$ has borrowed a book. (Assume that 'a' means 'at least one'.)

(b) Book $b$ has been borrowed.

(c) Book $b$ is on the shelf.

(d) Person $p$ has borrowed at least two books.

(e) No book has been borrowed by more than one person.

(f) There are no overdue books.

(g) If a book is overdue, then it must have been borrowed.

(h) Person $p$ has an overdue book.

**21** Prove each of the following statements:

(a) The sum of any even number and any odd number is odd.

(b) The product of any two odd numbers is odd.

(c) If $x + y < 2$ then $x < 1$ or $y < 1$, for any real numbers $x$ and $y$.

(d) The sum of any five consecutive integers is divisible by 5.

(e) If $n$ is an integer, then $n^2 + n$ is even.

(f) If $n$ is an odd integer, then $n^2 - 1$ is divisible by 4.

**22** Fill in the details in the following outline of a proof that $\sqrt{2}$ is irrational:

1. Assume $m / n = \sqrt{2}$ where $m$ and $n$ are natural numbers. Explain why we can assume that $m$ and $n$ are not both even.
2. Deduce that $m^2 = 2n^2$, and hence explain why $m$ must be even.
3. Let $m = 2k$ (where $k$ is a natural number), and deduce that $n$ is even.
4. Explain how this proves that $\sqrt{2}$ is irrational.

**23** Find a counterexample for each of the following statements:

(a) Any natural number that is divisible by both 4 and 6 is also divisible by 24.

(b) If $n$ is a natural number, then $n^4 + 4$ is a multiple of 5.

(c) Every natural number can be expressed in the form $x^2 + y^2 + z^2$ for some non-negative integers $x$, $y$ and $z$.

(d) For every natural number $n$, $n^3 \geq 2^n - 1$.

**24** Consider the following self-referential statement:

'This statement has five words.'

(a) What is the truth value of the statement?

(b) Write down the negation of the statement. What is its truth value?

(This exercise shows the kind of difficulty that can arise with a self-referential statement, even if it appears to have a well defined truth value.)

**25** On one side of a card is written:

'The statement on the other side of this card is true.'

On the other side of the card is written:

'The statement on the other side of this card is false.'

Explain how a paradox arises in this situation. (The problem is known as the Jourdain card paradox.)

**26** Four people are using computers in a computing laboratory. You know that the first person is a student and the second is not, but you do not know whether they are using the software on the network. You know that the third person is using the software on the network and the fourth is not, but you do not know whether they are students.

As the laboratory supervisor, you are required to enforce the rule that only students are allowed to use the software on the network. Which two people should you question, and what should you ask them?