CHAPTER 2

Logic

Aims

To introduce those elements of propositional and predicate logic which are prerequisite to the study of the fundamentals of Z.

Learning objectives

When you have completed this chapter, you should be able to:

- appreciate the importance of logic in the field of computing in general and formal specification in particular;
- draw truth tables for simple logic expressions;
- recognise tautologies, contradictions and logically equivalent expressions;
- convert between English sentences and logic expressions containing predicates, and understand the compromises necessary to do so.

2.1 Introduction

We all use logic in our daily lives when we attempt to draw valid conclusions from information we are given; in other words, logic is about reasoning. The mathematics of formal logic was developed by George Boole (1815–1864), in an attempt to describe human thought processes. In 1938, his *boolean algebra* was successfully applied to the design of switching networks by Claude Shannon, and more recently has found wide application in the field of computing, for example in design of computer circuits, control flow in programs, reasoning about properties of programs, and as a description language and reasoning mechanism for formal specification languages such as Z. This chapter, of necessity, contains only those bare essentials of propositional and predicate logic which are required for our study of the essence of Z; for a more comprehensive introduction, the reader is referred to Kelly (1997).

2.2 Propositions

A *proposition* is a statement which is either true or false, but not both. These truth values are represented by T and F respectively. The following are all propositions:

'This book is about Z.'
'There is a z in the word zoo.'
'The Earth orbits the Moon.'
'$42 < 42$'
'Liverpool are the best football team in England.'
'All cats wear hats.'
true
false

The last two are constant propositions, always true and always false respectively.

Questions and commands are not propositions. For example, the following are not propositions:

'Is the answer 42?'
'Don't talk to me like that!'

Exercises 2.1

1. What are the truth values of the above propositions?
2. Why is $x > 0$ not a proposition?

For brevity, the letters *P*, *Q*, *R*, *S*, etc., are often used to stand for propositions.

2.3 Compound propositions and logical connectives

The propositions above are *atomic*; that is, they are indivisible truth-valued statements. In everyday situations, we combine such propositions with words such as *and*, *or* and *not* to produce *compound propositions*. For example,

'This book is about Z and all cats wear hats.'
'There is a z in the word zoo or the Earth orbits the Moon, and Liverpool are the best football team in England.'
'There is not a z in the word zoo.'

Note that although such sentences do not always make sense, they are valid propositions; that is, they do have a truth value.

In formal logic, the words *and*, *or* and *not* are logical connectives, or operators. They are represented by the symbols $\wedge$ (conjunction), $\vee$ (disjunction) and $\neg$ (negation) respectively. The symbols $\wedge$ and $\vee$ are binary operators; that is, they are placed between two propositions to construct a new proposition. The symbol $\neg$ is a unary operator, placed in front of a proposition to construct a new one.

We must now define precisely what we mean when we use the symbols $\wedge$, $\vee$ and $\neg$ in logic expressions. We can do this by writing down the truth values of the relevant expression for each combination of truth values of its operand propositions (say P and Q), in a so-called *truth table*. The truth tables for the above operators are given in Tables 2.1–2.3.

Table 2.1 *Truth table for* $\wedge$

P	Q	$P \wedge Q$
F	F	F
F	T	F
T	F	F
T	T	T

$P \wedge Q$ is true if and only if P and Q are both true.

Table 2.2 *Truth table for* $\vee$

P	Q	$P \vee Q$
F	F	F
F	T	T
T	F	T
T	T	T

$P \vee Q$ is true when either or both of P and Q are true.

Table 2.3 *Truth table for* $\neg$

P	$\neg P$
F	T
T	F

$\neg P$ is true when P is false, and vice versa.

Exercise 2.2

Exclusive or.

The meanings of $\wedge$ and $\neg$ are fairly intuitive, but that of $\vee$ is rather different from our common interpretation. If P is the proposition 'This afternoon I will go to play football', and Q is the proposition 'This afternoon I will go to play golf', what can you say about the above definition of the connective $\vee$ in comparison with the usual meaning of the word *or* in everyday life? Write down a truth table to capture the latter meaning, using the symbol $\oplus$ for your connective.

Two more connectives, the implication operator $\Rightarrow$ and the equivalence or biconditional operator $\Leftrightarrow$, are given in Tables 2.4 and 2.5 respectively.

Table 2.4 *Truth table for* $\Rightarrow$

P	Q	$P \Rightarrow Q$
F	F	T
F	T	T
T	F	F
T	T	T

Table 2.5 *Truth table for* $\Leftrightarrow$

P	Q	$P \Leftrightarrow Q$
F	F	T
F	T	F
T	F	F
T	T	T

The implication operator is read as 'P implies Q' or 'if P then Q'. For example, 'if it is a nice day then I will walk to work'. However, propositions constructed with this operator do not have to make sense in everyday language. From the table we can see that the proposition 'if pigs can fly then there is a spaceship in my teacup' is true! A reasonable way to interpret the table is that the last two lines capture the 'if P is true then Q is true' part, and the first two lines capture the fact that if P isn't true, the expression says nothing about the value of Q; it may be true or false.

The equivalence or biconditional operator may be read as 'P is equivalent to Q' or 'P if and only if Q'. For example, 'I fail my exam if and only if I score less than 40%'. The phrase 'if and only if' may be shortened to 'iff', and this abbreviation will be used in this book.

The operators $\wedge$, $\vee$ and $\Leftrightarrow$ associate to the left, while $\Rightarrow$ associates to the right. This means that, for example,

$P \Rightarrow Q \Rightarrow R$ is equivalent to $P \Rightarrow (Q \Rightarrow R)$ and
$P \Leftrightarrow Q \Leftrightarrow R$ is equivalent to $(P \Leftrightarrow Q) \Leftrightarrow R$

The order of precedence of the connectives, from highest to lowest, is $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$.

In writing down compound propositions, we can take advantage of these properties to reduce the number of brackets needed. For example,

$$(\neg P) \Rightarrow (Q \wedge R)$$

may be written as

$$\neg P \Rightarrow Q \wedge R$$

and

$$(P \Rightarrow (Q \Rightarrow \neg P)) \Leftrightarrow Q$$

may be simplified to

$$P \Rightarrow Q \Rightarrow \neg P \Leftrightarrow Q$$

2.4 Truth tables for compound propositions

To draw the truth table of a compound proposition with more than one operator in it, we can break the proposition down into its constituent subexpressions, work out the truth tables for these, and then combine the result.

Example
Draw the truth table for the expression

$$(\neg P \wedge Q) \Rightarrow \neg(Q \vee R)$$

First, we will note that the number of rows in the truth table for a given expression is always 2^n where n is the number of distinct logic variables in the expression. There are three distinct logic variables in the above expression, P, Q and R respectively, and therefore the table will have eight rows. The easiest way to write down the permutations of the values of P, Q and R is to fill the first four rows in column P with F and the next four with T. In the next column, we put F in the first two rows, T in the next two, and so on, and finally in the third column we alternate each row with F and T.

We must now identify the 'first-level' subexpressions, according to the precedence and association rules above; that is, we express our proposition in the form

$$\neg\langle exp\rangle$$

or the form

$$\langle exp1\rangle \ \langle op\rangle \ \langle exp2\rangle$$

as appropriate, where $\langle exp\rangle$, $\langle exp1\rangle$ and $\langle exp2\rangle$ are the subexpressions and $\langle op\rangle$ is one of the above operators. If the subexpressions are complex, we would break these down similarly, until we have expressions for which we are able to write down a column in our truth table. For the above example, we can see that it has the form

$$\langle exp1\rangle \ \langle op\rangle \ \langle exp2\rangle$$

with

$$\langle exp1\rangle = (\neg P \wedge Q) \quad \langle op\rangle = \Rightarrow \quad \text{and} \quad \langle exp2\rangle = \neg(Q \vee R)$$

We write down columns for these expressions, and finally combine these columns to give the column for the entire expression. The result is shown in Table 2.6.

Table 2.6 *Truth table for* $(\neg P \wedge Q) \Rightarrow \neg(Q \vee R)$

P	Q	R	$(\neg P \wedge Q)$	$\neg(Q \vee R)$	$(\neg P \wedge Q) \Rightarrow \neg(Q \vee R)$
F	F	F	F	T	T
F	F	T	F	F	T
F	T	F	T	F	F
F	T	T	T	F	F
T	F	F	F	T	T
T	F	T	F	F	T
T	T	F	F	F	T
T	T	T	F	F	T

It can be seen that the above proposition is true for some rows of the table and false for others. A proposition which is true for all the rows of the table is called a *tautology*, and a proposition which is false for all the rows is called a *contradiction*.

Example

Establish whether the proposition

$$A = (P \wedge Q) \vee (P \wedge \neg Q \wedge R) \Leftrightarrow P \wedge (Q \vee R)$$

is a tautology, a contradiction or neither.

We draw the truth table for the subexpressions

$$B = (P \wedge Q) \vee (P \wedge \neg Q \wedge R)$$

and

$$C = P \wedge (Q \vee R)$$

and then combine these using the operator $\Leftrightarrow$ to obtain the column for the entire expression.

Again, you may wish to break the expression down further, creating more columns, to make the problem more tractable. The result is shown in Table 2.7.

From the final column of the table, we can see that the expression is a tautology.

Table 2.7 *Truth table for* $(P \wedge Q) \vee (P \wedge \neg Q \wedge R) \Leftrightarrow P \wedge (Q \vee R)$

P	Q	R	B	C	A
F	F	F	F	F	T
F	F	T	F	F	T
F	T	F	F	F	T
F	T	T	F	F	T
T	F	F	F	F	T
T	F	T	T	T	T
T	T	F	T	T	T
T	T	T	T	T	T

Exercises 2.3

1. Draw truth tables for the following propositions, and state whether each is a tautology, a contradiction or neither. Which are *logically equivalent*; that is, which have the same truth table?

   (i) $P \vee (Q \wedge R)$
   (ii) $\neg P \vee Q$
   (iii) $(P \Rightarrow Q) \Leftrightarrow (Q \Rightarrow P)$
   (iv) $\neg(P \vee Q)$
   (v) $\neg(P \wedge \neg Q)$
   (vi) *false* $\vee$ *true*
   (vii) *false* $\wedge \neg(P \vee Q)$

2. Read the following paragraph and write a logic expression to determine whether or not to cycle to work, with R standing for the proposition 'it is raining', S standing for the proposition 'the car starts', and P standing for the proposition 'a push start is available'.

   I either cycle to work, or I use my car. If it isn't raining, I cycle to work. If it is raining, I use my car, unless the car doesn't start, in which case I have to cycle in the rain, unless I can get a push start from my neighbour.

2.6 Laws of boolean algebra

Propositional logic (and set theory; see Chapter 3) are examples of *boolean algebras*. There are five basic laws, or postulates, of boolean algebra. For propositional logic, the laws are as follows, with P, Q and R standing for any propositions:

Commutative laws:

$$P \wedge Q \Leftrightarrow Q \wedge P$$
$$P \vee Q \Leftrightarrow Q \vee P$$

Associative laws:

$$(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$$
$$(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$$

Distributive laws:

$$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$$
$$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$$

Complement laws:

$P \vee \neg P \Leftrightarrow true$
$P \wedge \neg P \Leftrightarrow false$

Identity laws:

$P \vee false \Leftrightarrow P$
$P \wedge true \Leftrightarrow P$

Note that all laws come in pairs. Given one valid law, we can derive another one, its *dual*, by replacing all instances of $\wedge$ with $\vee$ and vice versa, and similarly with all instances of *false* and *true*.

2.7 Proof of theorems

Other laws, or *theorems*, can be defined, the validity of which may be proved by using the above basic laws and other, previously proven, theorems.

Proving laws by repeated application of previously derived laws is called *deduction*. Alternatively, we can prove laws by using truth tables. We write down the truth tables for each side of the law, and check that they are identical. This is called *perfect induction*. It can be tedious, but is often easier than proof by deduction.

Example
Prove $P \vee P \Leftrightarrow P$.

Proof:

$P \vee P$	
$\Leftrightarrow (P \vee P) \wedge true$	identity law
$\Leftrightarrow (P \vee P) \wedge (P \vee \neg P)$	complement law
$\Leftrightarrow P \vee (P \wedge \neg P)$	distributive law
$\Leftrightarrow P \vee false$	complement law
$\Leftrightarrow P$	identity law

Example
Prove $P \vee true \Leftrightarrow true$.

Proof:

$P \vee true$	
$\Leftrightarrow P \vee (P \vee \neg P)$	complement law
$\Leftrightarrow (P \vee P) \vee \neg P$	associative law
$\Leftrightarrow P \vee \neg P$	theorem above
$\Leftrightarrow true$	complement law

Example
Prove $P \vee (P \wedge Q) \Leftrightarrow P$.

Proof:

$P \vee (P \wedge Q)$	
$\Leftrightarrow (P \wedge true) \vee (P \wedge Q)$	identity law
$\Leftrightarrow P \wedge (true \vee Q)$	distributive law
$\Leftrightarrow P \wedge true$	theorem above and commutative law
$\Leftrightarrow P$	identity law

Note that we have used the operator $\Leftrightarrow$ in the above. This is a part of the language of propositional logic, and, strictly, this means that the above are not laws, but propositions. We should really have used the meta symbol $\equiv$, which does not construct a proposition, but denotes *logical equivalence* of its operands. However, we will not require this symbol in the rest of this book.

Exercise 2.4

Prove the dual of each of the above laws, both by perfect induction and by deduction.

2.8 Variables and types

Z is a *typed* language. To introduce a *variable* (i.e. a name which denotes a value) into a specification, we write a *declaration* associating the name with a *type*, which is the set of all the values which may be associated with the name. The set of all whole numbers (integers), denoted by $\mathbb{Z}$, is a type which is built into the language:

$$\mathbb{Z} = \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$$

The set of all natural numbers

$$\mathbb{N} = 0, 1, 2, 3, \ldots$$

although actually a subset of the type $\mathbb{Z}$ rather than a type itself, may also be used in declaring variables. For example,

$$x : \mathbb{Z}$$

declares a variable x which may stand for any integer, and

$$x : \mathbb{N}$$

declares a variable x which may stand for any natural number.

$$x, y : \mathbb{N}$$

declares two such variables. There are also mechanisms for defining additional types. We will say a lot more about sets and types in the next chapter, but this is sufficient for our present purpose, which is to extend the notation introduced so far by introducing the concepts of *predicates* and *quantification*.

2.9 Predicates

A predicate is an expression containing one or more *free variables* which act as place holders for values drawn from specified sets. Substituting values for all the free variables in the expression yields a proposition. For example, given

$$x, y : \mathbb{N}$$

the expression

$$x = y + 3$$

is a predicate with two free variables x and y. Replacing x with the value 4 and y with the value 3 yields a proposition with the value F. Replacing x with 5 and y with 2 yields a proposition with the value T. Thus, a predicate may be viewed as a template for constructing propositions by 'plugging in' values.

Exercises 2.5

1. Given $x, y, z : \mathbb{N}$, which of the following expressions are predicates?
   (i) $(x + 2y) - 7$
   (ii) $x + 10 = x$
   (iii) $(2x + y) \wedge z$
   (iv) $((x + y) < 7) \wedge (z = y)$
2. If we substitute the values $x = 3$, $y = 4$ and $z = 5$ in the above predicates, what are the truth values of the resulting propositions?

To construct a proposition from a predicate, we must remove all the free variables. We can remove a free variable either by replacing it with a particular value as above, or by *binding* it by *quantification*.

The universal quantifier

Consider the expression

$$\forall x : \mathbb{N} \mid x < 10 \bullet x + 9 > 12$$

Which may be explained as follows:

$\forall$ is the symbol for the universal quantifier, read as 'for all'.
$x : \mathbb{N}$ is the declaration of the variable which is *bound* by the quantifier. A bound variable is not free, and cannot be replaced with particular values. In the above expression, x stands for any value from $\mathbb{N}$.
$\mid$ is read as 'such that'.
$x < 10$ is an optional *constraint*. If we choose to omit it, then we also omit the symbol $\mid$.
$\bullet$ may be read as 'it is true that'.
$x + 9 > 12$ is the predicate being quantified.

Thus the expression may be read as 'For all natural numbers x such that x is less than 10, it is true that $x + 9 > 12$'. The expression contains no free variables, and is a proposition with the value F.

The general form of a proposition constructed using the universal quantifier is

$$\forall \langle \text{name} \rangle : \langle \text{type} \rangle \mid \langle \text{optional constraint} \rangle \bullet \langle \text{predicate} \rangle$$

which is read as 'for all ⟨name⟩ of type ⟨type⟩ such that ⟨optional constraint⟩ is true, ⟨predicate⟩ is true'.

The existential quantifier

Consider the expression

$$\exists x : \mathbb{N} \mid x < 10 \bullet x + 9 > 12$$

Here:

$\exists$ is the symbol for the existential quantifier, read as 'there exists at least one'.

The other parts of the expression are as before. The expression may be read as 'There exists at least one natural number x such that x is less than 10, for which it is true that $x+9>12$'. The expression contains no free variables, and is a proposition with the value T.

Note that there may be many natural numbers which make the predicate true; if we want to state that there is precisely one such number, we may use the *unique quantifier* $\exists_1$. The expression

$$\exists_1 x : \mathbb{N} \mid x < 10 \bullet x + 9 > 12$$

may be read as 'There exists *precisely one* natural number x such that x is less than 10, for which it is true that $x+9>12$'. Clearly, the value of this expression is F.

The general form of a proposition constructed using the existential quantifier is

$$\exists \langle \text{name} \rangle : \langle \text{type} \rangle \mid \langle \text{optional constraint} \rangle \bullet \langle \text{predicte} \rangle$$

which is read as 'There exists a ⟨name⟩ of type ⟨type⟩ for which ⟨optional constraint⟩ is true, such that ⟨predicate⟩ is true'.

Exercise 2.6

Which of the following are predicates, and which are propositions? For those which are propositions, what are their truth values?

(i) $\forall x : \mathbb{N} \mid x = 4 \bullet x > 5$
(ii) $\forall x : \mathbb{N} \bullet (x > 42) \vee (x \leqslant 42)$
(iii) $\forall x : \mathbb{N} \bullet (\exists y : \mathbb{N} \bullet y = 2x)$
(iv) $\exists x : \mathbb{N} \bullet 2 = 3$
(v) $\forall x : \mathbb{N} \mid (\exists y : \mathbb{N} \bullet x = 2y) \bullet x \neq 42$
(vi) $\exists_1 x : \mathbb{N} \bullet x = 42$
(vii) $\forall x : \mathbb{N} \bullet x = y$

2.11 A note about proof

Z is a formal language based on logic and set theory. There are proof theories associated with both propositional and predicate logic, which can be used to prove that Z specifications have various desirable properties and thus to gain more confidence in them. Proof is also useful in refining a more abstract specification into a more concrete one, usually closer in nature to the programming language in which the specification is to be implemented. Proof can give us more confidence that such a refinement retains the properties of the original specification. The use of proof in specifications is important to an advanced study of Z, but is beyond the scope of this book. See Diller (1994) and Woodcock and Davies (1996) for further reading on this important topic.