Chapter 4:
More on Logical, Information, and Text Functions

Introduction

Logical functions are those that involve Boolean values. The Boolean values are TRUE and FALSE. Some logical functions return a Boolean value as their result, others use the Boolean result of a comparison to choose between alternative calculations.

There are seven functions listed in the logical category in Excel – the functions AND, FALSE, IF, NOT, OR, TRUE and IFERROR. You'll see the use of most of these in this chapter. First, however, it's worthwhile to become familiar with the **logical operators**.

Logical Operators

TRUE and FALSE are common concepts. They are values which pertain to statements. For example, the statement "It is morning." is either TRUE or FALSE. We recognize that its truth value may change, but at any particular time the statement is either TRUE or FALSE.

What may be hidden here is the existence of an implied comparison. To determine the truth value of any statement we compare our understanding of the meaning of the claim with the facts. Strictly speaking the statement "It is morning." means the time of day is after midnight and before noon. To decide if it's TRUE we need to know the actual time of day and compare it to our criteria.

It's in these comparisons that we use Logical Operators:

Comparison	**Symbol**
less than	<
less than or equal to	<=
equal to	=
greater than or equal to	>=
greater than	>
less than or greater than	<>

A comparison typically involves checking if two values are equal or if one is less than the other, for example. To test if some cell that you have named Price contains a value that is larger than some other cell that you have named oldPrice you would use the comparison Price > oldPrice. If the value in cell Price was indeed greater than the value in cell oldPrice the value of the comparison expression Price > oldPrice would be true, otherwise the value would be false.

Logical Functions

As mentioned above, there are seven functions listed in the logical category in Excel.

The functions TRUE and FALSE really do not merit much discussion. They have no arguments, and as such are no different that the Boolean values themselves. In other words, entering the formula

= FALSE()

into a cell produces the display FALSE. This same display can be caused by simply entering the word into the cell. So why is there such a function? Unfortunately, if there is a good reason, it's been lost. We can assume that historically there was a perceived need for these functions and that there has never been a good reason to eliminate them. Whatever the case, we will not use them.

The more traditional Boolean operators are AND, OR, NOT. These are used to build complex Boolean expressions. NOT() is the inverter. It evaluates the Boolean expression that is its argument and returns the opposite value. So

NOT (TRUE) = FALSE
NOT (FALSE) = TRUE

AND and OR have the meanings with which you are by now familiar, but note again their implementation as functions rather than operators. We use AND as an operator when we say "The hour is greater than midnight AND less than Noon." However, in Excel, the formula begins with the function name which is followed by the arguments in parentheses. Assume there is a cell named TimeOfDay which contains the hour portion of the current time. To build an Excel function to determine if it's morning we need a formula like this:

= AND (TimeOfDay >= 0, TimeOfDay < 12)

Of course, both of these comparisons must evaluate as TRUE in order for the formula to return TRUE – that's what AND means.

The OR function is implemented in a similar fashion. Suppose, for example, a spreadsheet is used to determine if customers are eligible for an off-hours discount offered between the hours of 10:00 p.m. and 7:00 a.m. It could use the following formula which takes advantage of the fact that Excel stores times in 24-hour format:

= OR (TimeOfDay >= 22, TimeOfDay <7)

IS Functions

There is another category of useful functions in Excel called Information Functions because they provide information about the cells to which they refer. Not all of these produce Boolean results and are therefore of limited interest to us at this time. These others, however, can be very useful:

Function	**Returns TRUE if**
ISBLANK (Reference)	Reference refers to an empty cell.
ISERR (Reference)	Reference refers to any error Reference except #N/A.
ISERROR (Reference)	Reference refers to any error Reference (#N/A, #REFERENCE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!).
ISLOGICAL (Reference)	Reference refers to a logical Reference.
ISNA (Reference)	Reference refers to the #N/A (Reference not available) error Reference.
ISNONTEXT (Reference)	Reference refers to any item that is not text. (Note that this function returns TRUE if Reference refers to a blank cell.)
ISNUMBER (Reference)	Reference refers to a number.
ISREF (Reference)	Reference refers to a reference.
ISTEXT (Reference)	Reference refers to text.

These functions are often used as the Logical_test in an IF function. For example, it's possible that the cell a formula will use as the divisor might be blank. The formula would then produce an error because the blank cell has value 0, and division by 0 is impossible. To prevent this from happening the formula can be included as one value of an IF, and one of the IS functions can test to see if it should be performed. For example a column called Average could be calculated as follows:

= IF (ISBLANK (Count), "", Total / Count)

This formula “looks” at the current value of Count to see if the cell is empty. If it is, the cell is left blank because "" represents the NULL string, i.e. a string with no characters. Only if the ISBLANK function returns FALSE will the calculation be attempted by Excel.

Of course it might happen that the contents of Count might not be a number. In such a case the ISBLANK function will report FALSE, but the text will evaluate to 0 and the error message will appear. So it might be better to use a different test:

= IF (ISNUMBER (Count), Total / Count, "")

Or if the number might be 0…

```
= IF (AND (ISNUMBER (Count), Count > 0), Total / Count, "")
```

Text Functions

Another category of Excel functions help manipulate strings of text. Of the twenty-seven listed in Excel Help we are only interested in a handful at this time.

LEFT	Returns the leftmost characters from a text value
RIGHT	Returns the rightmost characters from a text value
MID	Returns a specific number of characters from a text string starting at the position you specify
LEN	Returns the number of characters in a text string
EXACT	Checks to see if two text values are identical
CONCATENATE	Joins several text items into one text item
UPPER	Converts text to uppercase.
LOWER	Converts all uppercase letters in a text string to lowercase.
TEXT	Formats a number and converts it to text
VALUE	Converts a text argument to a number

The LEFT, RIGHT, and MID functions are provided so that a formula can examine the characters at specific positions within a string. For example:

```
= LEFT ("Hi ho", 2)
```

returns the string "Hi". It copies **2 characters** from the **left end** of the string.

Similarly,

```
= RIGHT ("Hi ho", 2)
```

returns the string "ho" because it copies **2 characters** from the **right end** of the string.

The MID function has a different form.

```
= MID ("Hi ho", 2, 3)
```

produces the string "i h", i.e. **3 characters starting from position 2**.

These can be combined in a variety of interesting ways. For example,

```
= LEFT (RIGHT ("Hi ho", 2), 1)
```

returns 1 character from the LEFT end of the string produced by taking 2 characters from the RIGHT end of the original string – the second-last character.

Frequently, we don't know the contents of the string that will be present. The LEN function returns the number of characters in a string. Here is a formula that returns all but the first character from the string stored in ProductCode:

= RIGHT (ProductCode, LEN (ProductCode) –1)

The EXACT function has a Boolean result. It returns TRUE only if 2 strings are exactly the same. It is most useful when the data is case sensitive. Since Excel treats "a" and "A" as the same for purposes of logical comparison, it's helpful to have a function that can report that they're different.

Exercise 1

Open the file Exercise 1 (ch4_Ex1.xlsx) in Support Files (Chapter 4) on the course website. (Sheet 1 should be visible and empty.)

Ex 1.1

1. In cell A1 type "HI"
2. In cell A2 type "hi"
3. In cell A3 type "=A1=A2"
   This is a formula that will produce TRUE if the cells are the same.
4. In cell B3 type "=EXACT(A1, A2)"

Note that in Excel being the same is different from being EXACT. Can you think of a good reason for Excel to be designed so that "HI" equals "hi"? Write your answer on the worksheet you just created. Rename the sheet.

The CONCATENATE function is used to create strings out of bits and pieces. For example, CONCATENATE("Hi","Ho") produces "HiHo". You can also use the ampersand (&) as an operator in place of the CONCATENATE function to join two strings. For example, "Hi" & "Ho" also produces "HiHo".

Ex 1.2

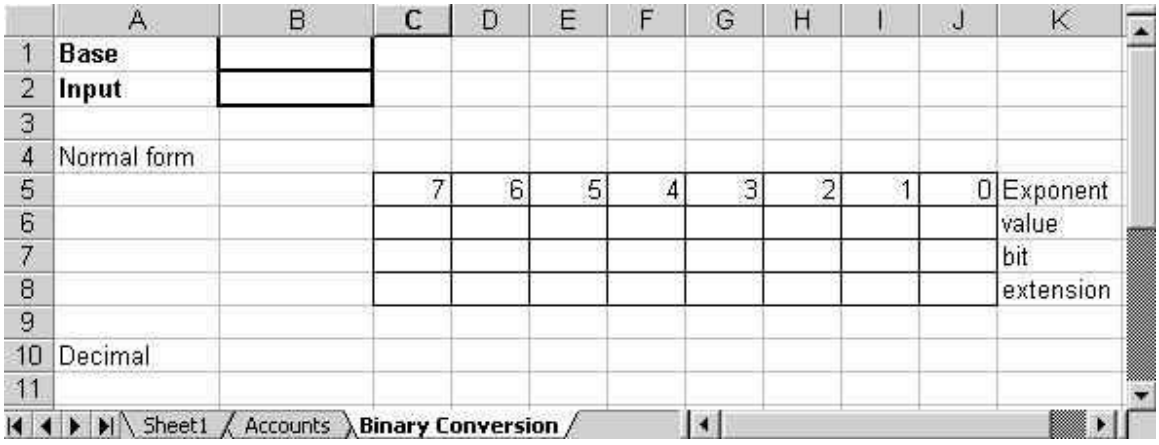
Switch to the Accounts sheet. It lists Surnames and Given Names for a group of fictitious people. Your task is to create another column in which to calculate each person's user identification number. The rule for these IDs is to join the first letter of the Given Name to the whole Surname, and convert the resulting string to UPPERCASE.

The TEXT and VALUE functions are used to convert data from one type to another. Notice that this is very different from simply applying a format. A format changes the way the data is displayed, but these functions actually change the data. If you display a number as text it can still be used in calculations, but if you change a number into a text string it CANNOT be used to compute further values – unless you convert it back.

Ex 1.3

Switch to the sheet named Binary Conversion. It should look like Figure 4.1.

Figure 4.1 – the Binary Conversion Worksheet



The Base cell is used to specify the base of the number that will be Input. When values are entered in the cells the sheet converts from the base specified into Decimal. To do this you will need several formulae.

Normal form is required because Excel will ignore any leading zeroes in the Input. To guarantee that the number has 8 digits it will be necessary to convert it to a string and pad the left end with 0's as necessary. This can all be accomplished with the TEXT function.

TEXT (value, format_text)

value *is a numeric value, a formula that evaluates to a numeric value, or a reference to a cell containing a numeric value.*

format_text *is a number format in text form from the* ***Category*** *box on the* ***Number*** *tab in the* ***Format Cells*** *dialog box.*

For our purposes we can specify exactly 8 digits using the format specification: "00000000". The formula for Normal form then is:

= TEXT (Input, "00000000")

Now that the data is in Normal_form we can create a single formula to extract each digit and place it in the appropriate cell in the range named bit. This formula requires some thought. How can Excel copy characters from a string into different cells? The key to the answer is in the fact that the digits are **characters**. We can therefore use a TEXT function. The formula will be located in cells C7 through J7. In C7 it needs to select the

first character from Normal_form, while in D7 it needs to select the second character, etc. The MID function looks like the best choice.

We know the name of the string and the number of characters needed so the formula will resemble this one:

=MID (Normal_form, *start_position*, 1)

Of course, the calculation of *start_position* is still the key.

You might be tempted to simply replace *start_position* with an integer: 1 in cell C7, 2 in D7, etc. But this is clearly NOT the best way to handle the task. Hard coding literal values in formulae is generally a bad idea. What's needed is an expression that will generate the appropriate sequence of digits, namely 1 in C7, 2 in D7, etc.

As it turns out, we have a descending sequence of integers just above in the **Exponent** row. Subtracting a descending sequence from the length of the string will produce the numbers we need. This part can be accomplished with

LEN (Normal_form) – Exponent

So substitute this expression for *start_position* to complete the formula.

You will also need to write a formula to calculate the cells in the value range. These values are simply the Base raised to the Exponents.

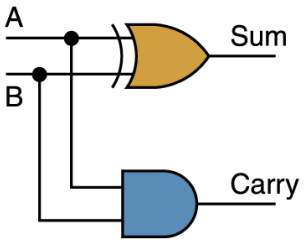
With the value and bit ranges defined you can multiply them together to get the extension values and SUM these to get Decimal.

Binary Addition

Figure 4.2 illustrates a half adder. Its results can be expressed in Boolean algebra as follows:

```
Sum =  A XOR B
Carry = A AND B
```

Figure 4.2 – a half adder



To express these in Excel requires a little more work. Carry is easy:

Carry =AND(A,B)

but there is no XOR function so you will need to construct a formula to perform the task. An exclusive OR is TRUE when either one of its arguments is TRUE, but not when both are TRUE. The "not both" part is simply NOT(AND( )), so an expression for Sum could look like this:

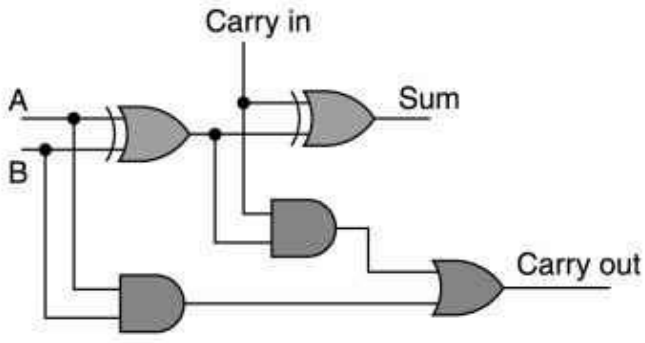
Sum =AND(OR(A,B),NOT(AND(A,B)))

Here is another way to look at this: An exclusive OR is TRUE when its arguments are not the same and FALSE otherwise. So, a formula to perform the task of the XOR function can also be constructed by simply using the NOT EQUAL operator (<>):

Sum =A<>B

A full adder (shown in Figure 4.3), of course, has a few more gates, one of which is OR, but this shouldn't pose any problems.

Figure 4.3 – a full adder



While it's certainly possible to describe each output in a single Boolean expression (and therefore a single Excel formula) these will be very complex. For example, they can be expressed as follows:

```
Sum         =AND(OR(AND(OR(A,B),NOT(AND(A,B))),CarryIn),
                NOT(AND(AND(OR(A,B),NOT(AND(A,B))),CarryIn)))

Carry out   =OR(AND(A, B),AND(AND(OR(A, B),NOT(AND(A,B))),CarryIn))
```

or, alternatively, assuming the inputs are TRUE and FALSE, rather than 1 and 0, respectively:

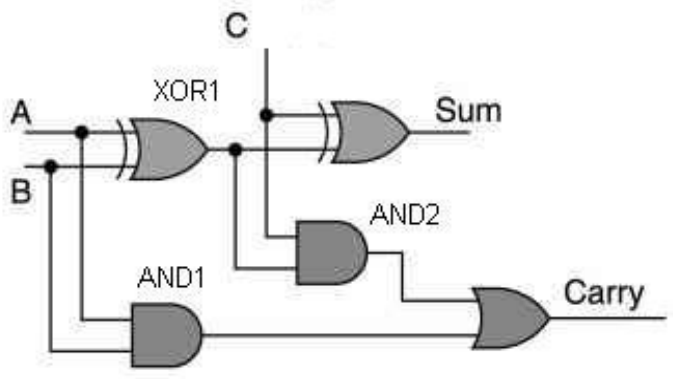
```
Sum         =(A<>B)<>CarryIn

Carry out   =OR(AND(A, B),AND(A<>B,CarryIn))
```

This is because comparisons such as <> (not equal) and logical functions such as AND and OR will accept 0 and 1 as inputs, interpreting them as FALSE and TRUE, but output only TRUE and FALSE regardless of the format of the inputs. Inputs of 0 and 1 will work, provided the result of each comparison is converted back to a number by using the IF function: IF(*comparison*,1,0).

These are very hard to understand! A better way to implement these formulas is to name and define the output for each logic gate. That way downstream gates can simply use the results of other gates by name. Figure 4.4 shows one way to label the gates.

Figure 4.4 – a full adder



In the case above there are 5 gates so we need 5 formulae:

```
XOR1    = AND (OR (A, B), NOT (AND (A, B)))
AND1    = AND (A, B)
AND2    = AND (C, XOR1)
Sum     = AND (OR (XOR1, C), NOT (AND (XOR1, C)))
Carry   = OR (AND1, AND2)
```

or, alternatively, again assuming inputs of TRUE and FALSE rather than 1 and 0,

XOR1 = A <> B
AND1 = AND (A, B)
AND2 = AND (C, XOR1)
Sum = XOR1 <> C
Carry = OR (AND1, AND2)

Exercise 2

Open the file called Exercise 2 (ch4_Ex2.xlsx) in Support Files (Chapter 4) on the course website. The adder sheet is a skeleton for a 4-bit adder. Note that the sizes of the rows and columns have been adjusted so that they line up with the diagram as it is currently placed. Of course it can be moved without affecting the operation of the worksheet in any way. You will also notice that names have been defined for each gate. The cells to which these names are applied are behind the picture, so you will need to slide it out of the way to see them. Your task is to make the adder work. You will need to extract the bits from the numbers entered by the user and define formulas for each of the gates. Your final draft might look like Figure 4.5.

Figure 4.5 – a full adder implemented in Excel