

Knowing how to finish by Gerald Fitton

*It's not where you start, it's where you finish
It's not how you go, it's how you land*

*From "Seesaw", the 1973 Broadway Musical
Lyrics by Dorothy Fields (1905 - 1974)*

It seems a lifetime ago now, but once upon a time I used to teach mature, Management Students all sorts of vaguely Mathematical things, one of which was Logic. My opening comment to these students on this topic was, "When you're given a legal contract to consider, then the first thing you should look at are the termination clauses". This provoked the highly fruitful discussion that I had hoped for.



As an aside to you, my very intelligent and worldly wise readers, I would add that you will realise that you are unlikely to consult the fine print of a contract when things are going well. It is always possible to terminate a contract by mutual agreement and then write another one which will take account of any changed circumstances. The time when you do need to invoke clauses from a contract is when it is all going badly wrong; if this happens then your attention will focus on that part of the contract which deals with termination.

Knowing how to finish anything, from a relationship to writing a mystery novel or an opera is, arguably, something you ought to get sorted out in your mind before you begin.

Deductive Logic

I think that, even before my interest in mathematics, I was attracted to Logic.

You will be familiar with the syllogism which starts with the major premise, "All cows eat grass", followed by the minor premise, "This animal is a cow" and then arrives at the conclusion, "Therefore this animal eats grass". What struck me at a very early age was that Deductive Logic can never increase the extent of human knowledge. Why? Because in order to make with total certainty the statement, "All cows eat grass", we must have investigated previously the grass eating habits of every individual cow, including our cow! During our extensive investigations we will have discovered that every animal which we call a cow (including 'our cow') eats grass; we do not need the syllogism of Deductive Logic but only the record of our observations on 'our cow' to know that it eats grass.

Inductive Logic

Inductive logic is a different matter entirely. Anyway, from whence came our major premise, "All cows eat grass"? Probably we used Induction; we first noticed that all those things we called cows ate grass and we took a leap of faith (or 'belief' if you prefer).

Often you do get to discover something new if you start with a set of particular examples and then work towards a general principle. This is a form of Induction rather than anything at all to do with Deductive Logic. The process of learning by using Induction owes more to our inbuilt acceptance of Bayesian Epistemology than it does to Aristotelian Logic. The Deductive Logic of Aristotle is not an Epistemological tool whereas the Inductive method of the Reverend Thomas Bayes will increase our knowledge of the world in which we live.

Mathematical Induction

Mathematical Induction depends on the same principles as Inductive Logic (it is slightly different but I won't go into the differences right now). In its most general form, Mathematical Induction is a process in which you start with something complicated that you wish to prove; then you show that this complicated thing is true if something a bit simpler is true. In turn the simpler thing is reduced to something even simpler ... and so on but not forever! Eventually you arrive at something which you are willing to regard as an axiom or which you can prove to be true some other way.

An inductive chain of reasoning, like so many things in life - including personal relationships and motor cars, is much easier to start than to stop. One of the golden rules when setting up a chain of inductive reasoning is that, before you start using it, you must know with certainty the condition under which you will abandon the recursive definition.

Why?

When he was about four years old, my grandson discovered that asking, "Why?" generates an inductive chain of answers. What Craig didn't appreciate at that time, but he does now, is that at some point he has to be satisfied with whatever 'unsatisfactory final answer' he was going to be given. We must accept that there are some questions for which there is not and never will be an answer.

In my youth when I worked at Vickers, the Managing Director used to classify his technical staff as 'One Why Men', 'Two Why Men' or 'Three Why Men'; the MD never asked "Why?" four times in a row. It took me quite a while to notice this. He wasn't asking "Why?" in order to receive an understandable answer. He kept asking "Why?", but only 3 times, in order to assess who he could trust with the problem. If you could answer his "Why?" 3 times then he accepted that you knew your stuff and let you get on with it.

The Exit Condition

Often it is by studying the 'Exit Condition' of an inductive sequence that we gain that insight and understanding we call Wisdom! So study the 'Exit Condition' not the sequence.

The Recurrence Relationship

One manifestation of Mathematical Inductive Logic is the recurrence relationship, a simple example of which is the factorial function. The meaning of factorial n (written as $n!$) is best demonstrated with an example such as factorial 7, written as $7!$.

By the way, n must be a member of the set, \mathbb{N} . It has to be a Natural Number, the sort of number such as 1, 2 or 3 that you count with. The factorial function, $n!$, exists only when n is a Natural Number.

This is what $7!$ means when it is fully expanded.

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1.$$

A second example is the expanded form of $6!$

$$6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1$$

We can use these two examples to define $7!$ as a recurrence relationship.

$$7! = 7 \times (6 \times 5 \times 4 \times 3 \times 2 \times 1) = 7 \times 6!$$

In more general terms, for any value of n ,

$$n! = n \times (n - 1)!$$

If you accept that $(n - 1)!$ is just a bit simpler than $n!$ then you will agree that I have defined the factorial function in terms of a simpler version of itself.

When to Stop

It is knowing when to stop using the recursive definition of $n!$ that is all important.

If you go one step too far then you'll finish up with nothing.

Have a go using the recurrence relationship (not the full, expanded definition) a couple of times to find $2!$ and you'll find that you're multiplying by zero because $(1 - 1) = 0$.

Anything multiplied by zero is the nothing you will finish up with.

Let me repeat something which applies not only to the factorial function but to (nearly) all functions which are defined using a recurrence relationship. It is knowing when and how to stop defining the function as a recurrence relationship and defining a particular value in some other way which is totally essential to the process of Mathematical Induction.

You have to accept that the recurrence relationship $n! = n \times (n - 1)!$ is not and cannot be true for every value of \mathbb{N} . If you do not accept this then not only will your sequence never end but also you will finish up with total rubbish such as the zero of the example above.

Does $0!$ exist?

Those of you still in possession of that obsolescent tool, a calculator, may find that hidden away somewhere amongst its many specialist functions, your member of this dying breed of mathematical tools sports a factorial function.

The number 0 is regarded by mathematicians as a member of \mathbb{N} ; it is a Natural Number with which you can count (it was the Mathematician and Philosopher Bertrand Russell who proved this) and, therefore, 0 must have a factorial. Type into your ancient machine $0!$ and you will be able to discover for yourself that built into it, is the value of $0! = 1$.

This is the exception to the recurrence relationship formula for the factorial function. $0!$ is not 0 times something (and therefore nothing) but just the plain and simple (but, you might think, illogical) number, 1 . It is this seemingly anomalous value of $0!$ which gives us the Exit Condition from our otherwise endless and indeterminate sequence, $n! = n \times (n - 1)!$

Of course, right now you will be inclined to think that this choice of 1 rather than 0 for the value of $0!$ is a bit arbitrary. I assure you that it isn't. This value, $0! = 1$, is 'logical' (meaning that it is consistent with a set of mathematical rules) when the factorial function is seen within the wider context of Regular (ie smoothly varying) Functions.

Calculating $n!$ using $n! = n \times (n - 1)!$

The screenshot below shows the PipeDream file I have used to calculate the value of $n!$ for 9 values of n using the recurrence relationship definition of the factorial function. You will see in the cell **B3** of the screenshot below the custom function `[c_fact]factorial(cellref)`.

	A	B
1	n	n!
2		
3	0	1
4	1	1
5	2	2
6	3	6
7	4	24
8	5	120
9	6	720
10	7	5040
11	8	40320
12		

The formula in **A4** is **A3+1**. This formula and the custom function in **B3** are replicated down to the bottom of the spreadsheet. These are the only formulae in the spreadsheet.

So what is this custom function thing called from every cell in the block **B3B11**?

It is a bit like a BASIC program. Look at the screenshot of the spreadsheet `[c_fact]`. Custom function spreadsheets are a special kind of PipeDream spreadsheet. They are characterised by the definitive three dots at the beginning of every line.

In line 5 the parameter " n " is passed to this function from the main spreadsheet. The answer is returned to the cell in the main spreadsheet by the instruction at line 14 .

```

4 A
5 ...function("factorial","n")
6 ;
7 ...if(@n<0|int(@n)<>@n,result("Invalid Entry"),);
8 ;
9 ...set_name("answer",B9)
10 ...set_value(answer,@n)
11 ;
12 ...set_value(answer,if(@n>1,@n*factorial(@n-1),1));
13 ;
14 ...result(answer)

```

Line 7 checks whether the number “ n ”, which has been passed to it as a parameter, is a member of \mathbb{N} or not. If “ n ” is not a Natural number then the custom function returns the error message, “Invalid Entry” to the main spreadsheet.

You can see this happen (if you load these files from the website) when you enter a number such as 0.5 or -1 into cell **A3** of the main spreadsheet.

It is line 12 that contains the recursive procedure:
`...set_value(answer,if(@n>1,@n*factorial(@n-1),1))`

What I want to draw your attention to is that the recursive call in this line is to the same custom function, *factorial(value)*, but with the argument reduced from n to $n-1$.

What line 12 does is evaluate $n!$ by using the recursive definition $n! = n \times (n - 1)!$.

When to stop

I can’t say this often enough. If you set up a recursive procedure to do anything then you must specify an Exit Condition otherwise your recursive procedure will fail.

In Line 12 you will see that there is an *if(condition, true, false)* for which the condition is that $n > 1$ (ie 2 or more). This ‘if’ ensures that recursive procedure exits when $n = 2$.

I could have used $n = 1$ rather than $n = 2$ for the exit condition because anything times 1 is itself; both exit conditions give the same result but $n = 2$ is just a little faster.

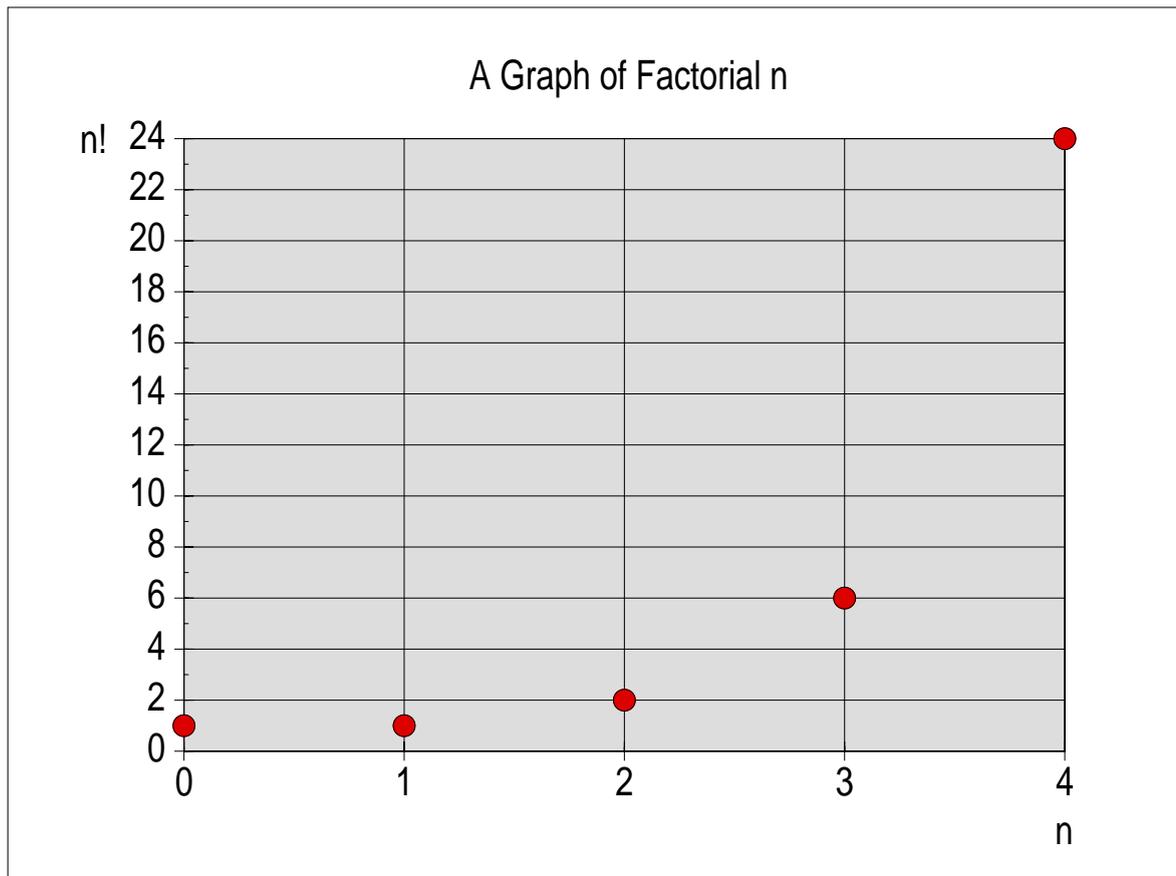
You will find that this custom function does give the correct answer not only for any natural number, \mathbb{N} , which returns an answer within the capacity of the computer, but also for the factorials $1!$ and $0!$.

The graph of $n!$

I have used PipeDream to draw a graph of $n!$ for values of n from 0 to 4 inclusive.

You will see that there are values for both $n = 0$ and $n = 1$ as well as values for 2, 3 and 4.

This graph is drawn by selecting Scattergraph from the PipeDream Gallery and the option from the submenu which displays just points with no lines joining the points. Just as you cannot have 'half a transistor' or 'half a hole' so you cannot have a Natural number, \mathbb{N} , which takes a value between these points.



The factorial function is not a continuous function which uses all the Real numbers, \mathbb{R} , but a discrete function which jumps from one 'quantum value' to another with nothing in between. Of course, Mathematicians do wonder if there is something in between (there is!) but that is another story for another day (yes, that's a promise). For now let me say that it is from considerations of the 'bits in-between' these Natural numbers, \mathbb{N} , that we find ourselves choosing $0! = 1$ rather than anything else because this choice is consistent with 'rules' about the in-between bits of this graph which extend $n!$ to values of \mathbb{R} .

The Tower of Brahma

I shall recount to you an ancient fable as the introduction to my second example of recursion. It is written in BBC BASIC rather than as a spreadsheet custom function. You will find a copy of this BASIC program on the Archive website.

In the Indian city of Benares, beneath a dome that marks the center of the world, is to be found a brass plate in which are set three diamond needles, each a cubit high and as thick as the body of a bee. At the time of Creation, Brahma placed 64 discs of pure gold on one of these needles. Each disc has a different diameter. Every gold disc on the diamond needle is placed so that it rests on top of another gold disc of greater diameter. The largest of these pure gold discs rests on the brass plate at the bottom and the smallest gold disc is at the top.



Within the temple are holy men whose job it is to transfer all these pure gold discs from the original diamond needle to a second diamond needle without ever moving more than one disc at a time. The Brahmin holy men can never place any golden disc on top of a smaller one, nor anywhere else except on one of the diamond needles.

When the task is done and all 64 golden discs have been successfully transferred to another diamond needle, then will the tower, the temple, the Brahmins and all of us alike, crumble into dust. With a divine thunder clap, our world, created by Brahma, will vanish.

BASIC Needles

It was a long time ago in 1987 that I wrote this short recursive program in BBC BASIC. It calculates and prints to the screen a set of instructions for carrying out this task without making any mistakes. I have included a screenshot of that BASIC program.

There is a lot of snobbery about recursion.

It is often portrayed as a programming feature which can be used only by an expert.

Don't believe it!

There are many useful mathematical functions such as $!$, the factorial function, and nC_r , the function used to find the number of combinations of n different things taken r at a time (eg the 14 million lottery possibilities), which are capable of simple expression as a recurrence relationship but which, in explicit form, are overwhelmingly difficult to understand.

In these cases recursion makes the program easier to write and easier to understand. When a program is easy to understand then improving or expanding it is much easier.

```
hostfs::Sharing$.MyFiles.GoldLine.Gold.Arc.2010.GC1003.Files.Basic.Needles2
10REM > Needles      :
20REM Author        : G L Fitton
30REM Copyright     : ABACUS TRAINING
40REM Date          : 23rd November 1987
50:
60:
70ON ERROR PROCerror
80REM MODE 3
90:
100:
110:
120PROCintro
130PRINT "Number of discs to transfer. "
140INPUT "Try a small number, eg 3, first. " discs$'
150discs%=VAL(discs$)
160REM Move a pile of discs from Needle 1 to Needle 2.
170PROCmove(discs%,"Needle 1","Needle 2","Needle 3")
180END
190:
200DEF PROCerror
210:
220:
230UDU 26,12
240REPORT
250PRINT " at line ";ERL
260STOP
270:
280ENDPROC
290:
300DEF PROCmove(disc%,from$,to$,spare$)
310REM Moves a pile of discs (from disc 1 down to disc%) from from$ to to$.
320:
330:
340IF disc%(<)0 THEN PROCmove(disc%-1,from$,spare$,to$)
350IF disc%(<)0 THEN PRINT "Move Disc ";disc%;" from ";from%;" to ";to%;"
360IF disc%(<)0 THEN PROCmove(disc%-1,spare$,to$,from$)
370:
380ENDPROC
390:
400DEF PROCintro
410:
420CLS
430PRINT "There are three needles. On the first needle is a number of discs."
440PRINT "You must transfer the discs to the second needle."
450PRINT "You must not put a low number disc below one of a higher number."
460PRINT "Running this programme gives instructions for moving the discs."''
470:
480ENDPROC
490:
```

The BASIC procedure, PROCmove, is called at line 170 and is defined in lines 300 to 380 inclusive. What I would like you to notice about this rather interesting procedure is that it contains two recursive calls to the same procedure at lines 340 and 360 but with different arguments. This recursive call moves not one disc at a time but a whole pile of discs from the 'from' needle to the 'spare' needle at line 340 and then, at line 360 this same pile is transferred from the 'spare' needle to the 'to' needle. Note also that the Exit Condition is a totally logical one; there are no discs left on the needle because they have all been moved.

Using two recursive functions in this way seems to me to be much more understandable than any attempt to try to design a BASIC procedure which moves only one disc at a time.

Dorothy Fields

Dorothy Fields wrote the lyrics of hundreds of songs including, “I can’t give you anything but love, baby”, “Don’t blame me”, “On the sunny side of the street”, “A fine romance”, “I’m in the mood for love”, “Big spender”, “If my friends could see me now”, “Pick yourself up” and “I won’t dance”.

My favourite lyrics of hers are to be found in the Jerome Kern melody, “Just the way you look tonight”. I recall that a very long time ago I was dancing to this foxtrot in the Town Hall at Cheltenham with Jill, the lady who has been my dearest friend - as well as my wife. As we danced together she sang the words of the chorus softly in my ear. It was in response to something I had said a week or so earlier (about me changing) that Jill insisted that I should, “Never, ever change ...’cause I love you, Just the way you look tonight.”

Bio-Bit

I thought you’d all be a bit fed up with seeing pictures of me each month so, for a change, here’s a picture of the person without whom I would not be the person I am. She is my dearest friend, Jill, who once sang to me, “Never, ever change ...”.



The Uncaused Cause

Many people believe, as my grandson once did, that they are being logical when they claim that the recursive nature of the classic series of questions about causes, which eventually leads to the question, “Who created the creator of the universe?”, negates the claim that a creator must exist. I suggest that this logic is flawed and that we must look again at the basic premise of the recursion inherent in Inductive Logic which, in this case, can be stated as, “You must know when to stop asking for a cause!”

See if you can follow my next statement: “It is the recursive nature of ‘cause’ which forces us to accept that if causality exists at all, there has to be a first cause, an Exit Condition from the causal chain, namely an uncaused cause”. What is needed to overcome the feeling we have of the logical inelegance of an uncaused cause is a new and wider perspective. In this wider perspective the uncaused cause will take its natural place in the scheme of things just as $0! = 1$ takes its proper and natural place as a special point on a smooth curve - more about smooth curves and why in another article.

It is the more enlightened and more elegant concept of smoothness, used by mathematicians to avoid the quantum jumps of the natural numbers, \mathbb{N} , that allows us to develop the factorial function, $n!$, (but with a different name) to include, negative as well positive numbers, \mathbb{Z} , all the real numbers, \mathbb{R} , and even the complex numbers of \mathbb{C} . Within this smoothly enlightened world we shall find that it is completely natural for $0! = 1$.

To repeat myself, as soon as you begin to believe in cause and effect (causality) then, because of the recursive nature of causes, you have to accept the existence of an uncaused cause. In the wider scheme of things ‘cause’ implies a time dependence and, speaking personally, I reckon that I’ll have to understand a lot more about the nature of Eternity before I can see how the uncaused cause breaks the causal chain in a totally natural way.

Defining concepts such as ‘cause’ and ‘effect’ as well as mathematical functions recursively, has its place in philosophical as well as mathematical epistemology.

We have to accept that an intrinsic and essential part of every inductive process is an inelegant and seemingly illogical Exit Condition. Although it might be difficult to decide when and how to stop, we must stop somewhere. Furthermore, before we start any inductive chain of reasoning we must have a good idea where we want to stop and how to do it.

I shall go further and state that there is more Wisdom to be gained by studying Exit Conditions than by blindly following the well trodden pathway of induction until we drop from mental exhaustion.

Perhaps later, when I understand more about the nature of Eternity, I shall be able to put aside my need for the childish concept of causality. Until the time comes when I cease to, “see through a glass darkly”, causality has its place in my working model of the universe and, call it faith or if you prefer, call it the pursuit of wisdom, I accept the necessity for an exceptional, seemingly illogical, uncaused cause.