Spreadsheets in Education (eJSiE)

Volume 2 | Issue 1 Article 3

2-16-2006

Recursion and Spreadsheets

John Baker
Natural Maths, john@naturalmaths.com.au

Jozef Hvorecký
Vysoka skola manazmentu/City University Bellevue, Slovak Republic, jhvorecky@vsm.sk

Stephen J. Sugden
Bond University, ssugden@bond.edu.au

Recursion and Spreadsheets

Abstract
This article provides a definition of recursion1 based on the facilities offered by the spreadsheet. The recursive paradigm is contrasted with the functional form, whereby a table of values is generated by reference to the values of neighbouring cells rather than by means of context-independent direct formulae. The examples given in this article show that the application of recursive thinking need not be restricted to conventionally recognized areas, but that in many cases recursive thinking allows fresh insight into normally 'hard' areas.

Keywords
spreadsheets, recursion, algorithms

Recursion and Spreadsheets

John E Baker
Natural Maths, Australia
john@naturalmaths.com.au

Jozef Hvorecký
Vysoka skola manazmentu/City University Bellevue, Slovak Republic
jhvorecky@cutn.sk

Stephen J Sugden
School of Information Technology, Bond University, Australia
ssugden@bond.edu.au

February 16, 2006

Abstract

This article provides a definition of recursion[1] based on the facilities offered by the spreadsheet. The recursive paradigm is contrasted with the functional form, whereby a table of values is generated by reference to the values of neighbouring cells rather than by means of context-independent direct formulae. The examples given in this article show that the application of recursive thinking need not be restricted to conventionally recognized areas, but that in many cases recursive thinking allows fresh insight into normally 'hard' areas.

Submitted February 2005; revised and accepted October 2005.

Keywords: spreadsheets, recursion, algorithms.

1 Introduction

For centuries, mathematical computation was made possible through the use of tables: powers, square roots, logarithms, sines, cosines, and many other very useful quantities. Such tables had their origins in the need for accurate maritime navigation, for which tables were first developed in the 16^{th} century. In 1569, Mercator published his famous projection map, and Wright followed with a set of corrected tables for this projection

[1] By *recursion*, we refer to *primitive recursive functions* that are described formally as recursive functions that can be implemented programmatically by iteration (loops) [27]. This level of formality is not appropriate to this paper, and we give a less formal definition of recursion later in the paper.

at the close of that century [12]. One can have nothing but admiration for those whose efforts went into the creation of this information form.

Four centuries later, with the advent of the modern computer, the need for tables of data was superseded, but a similar effort was channeled into the development of programming languages in which information processing needs could be expressed. While not the only programming paradigm, the implementation of recursion as a way of expressing a problem and hence its solution by program became one of the main tools available to programmers.

In this paper, we explore how the spreadsheet provides an environment in which both information can be quickly laid out in tabular form, and how this layout so elegantly supports the recursion paradigm. The term *elegantly* is used here to capture the notion that a recursive approach, when used on a spreadsheet, is not only the simplest, but most often the most transparent way of describing a process and of generating a table of values within which the solution to a problem can be found.

Recursion is a computational method based on (usually simple) relationships between values of a sequence: a small number of initial elements being separately defined (otherwise the calculation could not start). The remaining elements are calculated from their predecessors using a specified calculation method. The (much overworked) factorial function is defined in such a way, and is shown in eq (1).

$$n! = \begin{cases} 1 & \text{if} \quad n = 0 \\ n \times (n-1)! & \text{if} \quad n > 0 \end{cases} \tag{1}$$

Recursion is seldom taught in schools except where LOGO is used to enhance understanding of geometry. The main reason for this seems to be the complexity of evaluation, or finding the value of a recursively defined function for a given value of x. It is not always easy to understand the definition of a recursive function as we tend to expect that a function will be defined as a formula that enables the function output to be calculated directly from its input. However, given that the Church hypothesis [16] asserts that every computable function can be expressed as a recursive function, we should not feel that the ground that can be covered by recursive functions is limited.

Recursive functions or procedures are also not very popular among computer programmers. The main reasons for this would seem to be an apparent complexity in their definitions and a reputation of poor run-time performance (inefficiency).

But we need to define what we mean by the use of recursion on a spreadsheet and indicate what the inferior alternatives are. Let us begin then with an example from school mathematics, the arithmetic sequence. The sum of the first n terms of an arithmetic sequence is given by:

$$S = na + \frac{1}{2} n\,(n-1)\,d \tag{2}$$

In eq (2), a is the initial term and d the constant difference. While this formula can certainly be used to calculate the nth sum, the process by which the sum is obtained is completely obscured by the formula. On the other hand, with a little understanding of

Recursion and Spreadsheets

	A	B	C	D
1				
2		**a**	12	
3		**d**	3.5	
4				
5		**n**	**term**	**sum**
6		1	12	12
7		2	15.5	27.5
8		3	19	46.5
9		4	22.5	69
10		5	26	95
11		6	29.5	124.5
12		7	33	157.5
13		8	36.5	194
14		9	40	234
15		10	43.5	277.5
16		11	47	324.5

	A	B	C	D
1				
2		**a**	12	
3		**d**	3.5	
4				
5		**n**	**term**	**sum**
6		1	=a	=C6
7		2	=C6+d	=D6+C7
8		3	=C7+d	=D7+C8
9		4	=C8+d	=D8+C9
10		5	=C9+d	=D9+C10
11		6	=C10+d	=D10+C11
12		7	=C11+d	=D11+C12
13		8	=C12+d	=D12+C13
14		9	=C13+d	=D13+C14
15		10	=C14+d	=D14+C15
16		11	=C15+d	=D15+C16

Figure 1: Values and recursive formulas for an arithmetic sequence.

spreadsheet mechanics, the table of Figure 1 can be constructed in such a way that both a and d can be given values that the user might choose and all intermediate calculations are on display as well.

In Figure 1, we see one of the hallmarks of a recursive, rather than functional, relationship. The *term* column expresses the relationship that successive terms are made by adding d, the constant difference, at each stage. The *sum* column first establishes the initial value, and the remaining formulas are based on close neighbor values, expressing the recursive definition that each sum is made by adding the current *term* to the previous *sum*. Many readers will be familiar with the key combination[2]: Hold down <Ctrl> and press <~> as the way to switch between the left-hand part and right-hand part of Figure 1. It is a key combination that the authors use frequently in the classroom to uncover the calculations that a spreadsheet uses.

The expressions used for the *term* and *sum* column are those that are closest to our interpretation of recursion within the spreadsheet. The formula for *term* cam be recursively expressed as

$$\text{term}_n = \begin{cases} a & \text{if} \quad n = 1 \\ \text{term}_{n-1} + d & \text{if} \quad n > 1 \end{cases} \tag{3}$$

The recursive equivalent of *sum* is

$$\text{sum}_n = \begin{cases} a & \text{if} \quad n = 1 \\ \text{sum}_{n-1} + \text{term}_n & \text{if} \quad n > 1 \end{cases} \tag{4}$$

Their analogs can be easily found in the columns of Figure 1.

[2] This key combination applies to Excel spreadsheets and may be different in other spreadsheet programs.

	A	B	C	D
1				
2		**a**	12	
3		**d**	3.5	
4				
5		**n**	**term**	**sum**
6		1	12	12
7		2	15.5	27.5
8		3	19	46.5
9		4	22.5	69
10		5	26	95
11		6	29.5	124.5
12		7	33	157.5
13		8	36.5	194
14		9	40	234
15		10	43.5	277.5
16		11	47	324.5

	A	B	C	D
1				
2		**a**	12	
3		**d**	3.5	
4				
5		**n**	**term**	**sum**
6		1	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
7		2	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
8		3	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
9		4	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
10		5	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
11		6	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
12		7	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
13		8	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
14		9	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
15		10	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d
16		11	=a+(n-1)*d	=n*a+0.5*n*(n-1)*d

Figure 2: Using non-recursive formulae to create a table of values.

Whilst programmers might define recursion in terms of a routine that calls itself recursively until the terminating condition is met, recursion within the spreadsheet is somewhat simpler.

Definition 1 *A recursive process within a spreadsheet is one in which a table of values is formed sequentially by means of formulae that refer to nearby cells that have been evaluated earlier in the calculation sequence.*

To complete our introduction, let us generate the same table of values, but this time using the functional form of the expressions for *term* and *sum*[3].

In Figure 2, we give values and formulas for the same table, and invite the reader to make a comparison between that and Figure 1, from a learner's point of view. Noss [18] has suggested that we may need to rethink about which is more natural: direct formulas or recursive ones. The present authors are in complete agreement.

2 Recursion at the foundation of mathematics

The concept of recursion is one of the fundamentals of mathematics as it is the method by which both sequences and the Axiom of Mathematical Induction are defined. Indeed, it could be argued that only sets, functions and relations are more fundamental than sequences and induction, as the normal definition of an infinite sequence is "a function whose domain is the natural numbers" [17]. Since the natural numbers are the

[3]Note: In this example, we have made extensive use of Names, a feature of Excel spreadsheets that greatly simplifies the appearance of formulae. We should remind readers that the only letters that are reserved by Excel are C (for columns) and R (for rows) and that simple lettering for variable names, such as that used in this example, should be used wherever possible.

rock-bottom foundation of the hierarchy of number systems, and fundamental to the development of all mathematical structures, it is not hard to see that sequences (and hence recursion) assume an essential and fundamental role in the development of mathematics. The natural numbers themselves are defined by the Peano Axioms [20] where the smallest natural number is defined to be 1. A more modern definition appears on a page of the Math World site [26], where, according to modern consensus, zero is now granted membership as follows.

1. Zero is a number.
2. If a is a number, the successor of a is a number.
3. Zero is not the successor of a number.
4. Two numbers of which the successors are equal are themselves equal.
5. Induction axiom. If a set, S, of numbers contains zero and also the successor of every number in S, then every number is in S.

Our interest here is in use of recursion in Axiom 2, and in the principle of mathematical induction, embodied in Axiom 5. Notorious among students for its difficulty (see, for example, Abramovich and Pieper [2] and Sugden [23]), the fifth axiom is associated with an extremely powerful technique of proof, wide-ranging in its potential scope of application: any true proposition concerning the set of natural numbers is, in principle at least, a candidate for inductive proof. The process whereby mathematical induction often uses a formula associated with n to show the correctness of the successor formula associated with $n + 1$ is closely related to recursion, and their structures may be usefully compared. For IT majors, a fundamental unit is discrete mathematics, and will typically include both mathematical induction and elementary recurrence relations, but perhaps not recursive programming, which is usually left to a programming unit. Nevertheless, one of the authors has found it beneficial to point out the many connections between recursion and induction, via such seemingly mundane fundamentals as truth-tables, relations, binomial coefficients, sets and power sets (Sugden [23]).

3 How does a table express a recursive relationship?

The spreadsheet is a tool, available on essentially every PC, that provides strong support for tabular information. Indeed, the *Fill Handle* offers features that make the construction of tables both natural and rapid. When combined with the ability to highlight numbers that meet predetermined criteria by the use of conditional formatting [3], the tabular layout provides more that just a list of calculated values. Patterns may be easily observed, giving a strong support to constructivist learning, and solutions to equations located by visual means. For example, if a table of function values is created and solutions to $f(x) = 0$ are sought, then applying a conditional format to table such as that shown in Figure 3, enables roots to be located visually rather than by inspection.

To slightly paraphrase Hamming, *computing is about insight, not numbers!* [8]. When we have tables or lists (sequences), it is of interest, from the constructivist viewpoint, to find and explain patterns within them. Mathematicians look for patterns, and would encourage students at school to do the same when learning mathematics. Consider the following statement by Hardy [9]:

> A mathematician, like a painter or a poet, is a maker of patterns. If his patterns are more permanent than theirs, it is because they are made with ideas. ... The mathematician's patterns, like the painter's or the poet's, must be beautiful; the ideas, like the colours or the words, must fit together in a harmonious way. Beauty is the first test: there is no permanent place in the world for ugly mathematics.

We have the spreadsheet at our disposal: a near-ideal tool for looking for patterns. Have we overlooked the power at our fingertips in a spreadsheet environment?
But just because a table of values can be readily formatted to expose patterns does not necessarily imply that it should be constructed in recursive, rather than functional, form. The many examples given later in this article are intended to make the case that the recursive method has wide application as well as providing a simplifying view of the processes that underlie a model. But before we expand into these examples, we give a further rationale in support of the use of recursion within a table of values.

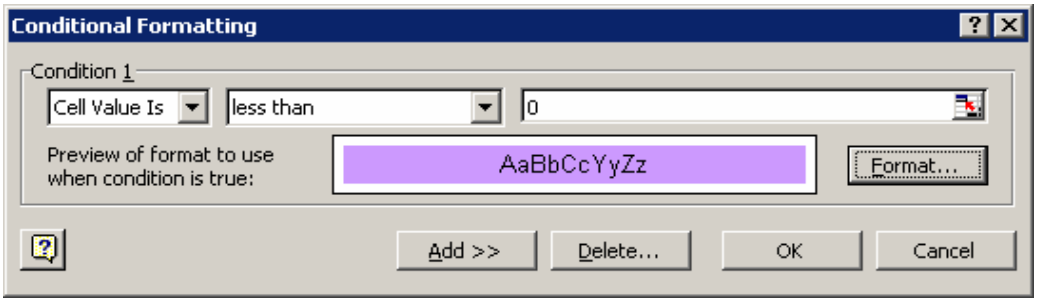

Figure 3: Applying conditional formatting to a table of values.

Recursion, and recurrence relations, are pervasive concepts in mathematics and computer science. Indeed, entire programming languages are based on recursion (for example, LISP [14], Scheme [22]), so it is perhaps surprising that recursive programming traditionally is very tricky for students. Recurrence relations (also called difference equations) and their continuous analog, differential equations, are pervasive in mathematics and the physical sciences and yet are usually perceived by students as being complex and confusing. Later, we present examples to show that recursive relationships are naturally and easily represented in a spreadsheet; so much so that we might be inclined to underestimate the relatively high information content of a seemingly simple-looking recursive formula that may arise in programming and mathematics. For example, the

Recursion and Spreadsheets

	A	B	C
1			
2		**n**	**Fib**
3		1	1
4		2	1
5		3	2
6		4	3
7		5	5
8		6	8
9		7	13
10		8	21
11		9	34
12		10	55
13		11	89

	A	B	C
1			
2		**n**	**Fib**
3		1	1
4		2	1
5		3	=C3+C4
6		4	=C4+C5
7		5	=C5+C6
8		6	=C6+C7
9		7	=C7+C8
10		8	=C8+C9
11		9	=C9+C10
12		10	=C10+C11
13		11	=C11+C12

Figure 4: The Fibonacci sequence.

Fibonacci sequence is nearly trivial to generate on a spreadsheet (see Figure 4) but the corresponding formula is inaccessible to all but the top echelon of high-school mathematicians.

In contrast to the simplicity of Figure 4, the formula for the n^{th} term of a Fibonacci requires a substantial understanding of mathematical notation.

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n \tag{5}$$

Through the formulas of Figure 4, we can see how a recursive relationship is expressed in a table by means of reference to neighboring cells. But the table goes one step further in providing understanding and insight into formulas that are expressed in recursive terms. A typical computer science exercise would be to ask the student to generate a trace table of values that the execution of a routine implies. By generating a trace table, the student may be able to locate where an error in coding has occurred, or alternatively, to verify that a process is executing correctly. This is where one of the great benefits of computing recursive expressions appears: a table of values emerges. Normally, there is a considerable overhead to the generation of a trace table, but when the table of values in a spreadsheet is created, the trace of many iterations is part and parcel of the process [13].

Thus the spreadsheet not only makes it appropriate to express relationships in their recursive form, it also provides a complete trace of the correctness of the processes used—a form of feedback that in computer science normally requires considerable overhead.

Can a recursive reference also address cells to the right or down? Yes, in some special cases—when then size of the referenced area can be restricted by a constant—such calculations are possible. Hvorecký and Trencanský [10] describe a calculation of this type. An unlimited growth to the right or down would certainly cause problems as

sooner or later a cell with an undefined content or a non-existing cell (a cell outside the sheet) will be referenced.

4 Power in simplicity

Consider this: even without using any mathematical functions, that is, just by keeping to the four operations of arithmetic, we can produce models of great power, yet of very low complexity. We offer our compound interest/superannuation or mortgage payment models in Excel. These each contain one addition and one multiplication, yet, with the recursive power of fill-down, generate a complete payment schedule, and are able to cope with residuals and superannuation rollover from a previous fund into a new one. Our spreadsheet will double up for Excel's built-in financial functions, of which the big five are `PV`, `FV`, `PMT`, `RATE` and `NPER`, but it requires no understanding of these complex formulas. We start with an example of *superannuation*, also termed a *future value annuity*.

Consider the following example (a slightly modified examination question from the 1990s in Discrete Mathematics at Bond University):

Example 2 *Al Gorithm has a superannuation account in which \$200 is deposited at the end of each month. His account earns 12% per annum compounded monthly, and was opened with a one-off deposit of \$200,000, being a lump-sum payout from a previous fund after switching jobs. The first monthly payment of \$200 was made at the end of the first month. For each nonnegative integer, n, let $\$x_n$ be the amount in the account at the end of n months.*

1. *Write down a linear, first-order recurrence relation for x_n*
2. *Find the general solution for your recurrence relation of part 1.*
3. *Find the particular solution satisfying the initial condition $x_0 = 200,000$.*
4. *Assuming no \$200 payment is missed, and the interest rate never changes, how long will Al have to wait for his account to be worth \$1,000, 000?*

These questions invite an answer in functional form, and after a moderate algebraic effort, we obtain the solution of a recurrence relation. We could also have used the Excel intrinsic such as `NPER` to solve this problem. While it is useful for the student to know how to use such functions, that approach obscures the structure of the problem. If our goal is not just to get the right answer, but also to show maximum understanding of the structure of the problem, then the recurrence approach in Excel is strongly recommended. Numerical analysts may argue that round-off errors may accumulate and affect the accuracy of the answer, but this is not a significant problem, even for hundreds or even thousands of periods. A side benefit of the recursive method in Excel is that a complete schedule of net worth is generated.

To show how the model is created, we give the following steps that produce the spreadsheet shown in Figure 5.

	A	B	C	D
1	month	net worth at end of month	payment	rate
2	0	200000	200	0.01
3	1	=C3 + rate*C3 + payment		
4	2	=C4 + rate*C4 + payment		
5	3	=C5 + rate*C5 + payment		
6	4	=C6 + rate*C6 + payment		
7	5	=C7 + rate*C7 + payment		
8	6	=C8 + rate*C8 + payment		
9	7	=C9 + rate*C9 + payment		

Figure 5: The Financial Model

1. In `A1:D1`, respectively type the headings “month”, “net worth at end of month”, “payment” and "rate".
2. Fill the range `A2:A202` with the series $0, 1, 2, 3 \ldots$ by typing 0 in cell `A2`, holding down <Ctrl> and dragging the Fill Handle down to `A202`.
3. Click in cell `C2`. Then go to the Name Box and type “payment” followed by <Enter>. Now click in D2 and name it in the same way as “rate”.
4. Click on Column `B` and format it to $. Do the same with cell `C2`.
5. In `C2`, type 200 and in `B2`, type 200,000.
6. Now comes the recurrence: in cell `B3`, put a formula which shows that after one month, our net worth is the sum of three quantities:
   - what we had last month ($200,000),
   - the interest on that sum, and finally,
   - the monthly payment.

Thus, our formula is: `(B3) = B2 + rate*B2 + payment`

1. Double-click on the fill-handle of `B3`. The recurrence now occupies cells `B3:B202`.
2. Scrolling down to `B157` yields our answer, which is 155 months (in `A157`).

Suppose now that 155 months (almost 13 years) is a bit too long for Al to have to wait. He wants to retire on $1,000,000 (or thereabouts) in 10 years instead of 13. How do we solve this one?

We assume he can increase his payment from $200 sufficiently to allow early retirement. We can now use *Goal Seek* to determine the payment so that `B122` (net worth after 10 years) is 1,000,000. The steps are:

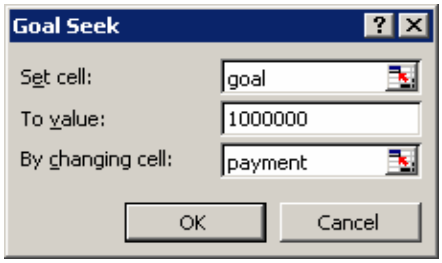

Figure 6: Using Goal Seek to find the required payment

1. Click in `B122`; then click-in the Name Box and type "goal" followed by <Enter>.
2. Select *Tools: Goal Seek* and fill out its dialog box as shown in Figure 6.
3. Click OK, and Excel will do the rest. A surprisingly substantial increase from $200 to $1477.68 is seen to be necessary, even with a relatively high interest rate of 1% per month! Al may wish to reconsider his plan and just work for another three years!

In summary, we should point out that a minimum of effort is needed to establish this superannuation model. It includes rollover from a previous fund, and the ability to alter payments at any stage should we want to investigate a different time period. It would be very easy to extend to make the interest rate another parameter, and likewise for the initial rollover value. These answers may be checked with Excel intrinsics such as `PMT`, `FV` or even by algebra using sums of appropriate geometric series. A similar Excel model may be made to produce a schedule for mortgage payments; in fact we need to change *just one character!* The formula in `B3` and its descendants in the same column just needs to be changed from `(B3) = B2 + rate*B2 + payment` to `(B3) = B2 + rate*B2 - payment`.

The potential that such a deceptively simple model offers business students for learning the structure of elementary finance in a compound interest environment is considerable indeed, and many more variations on this theme can be quite easily done. For example, we may ask what new interest rate would be necessary for Al to retire in 10 years, if he is prepared to increase his monthly contribution to $500. Such a question can easily be accommodated, and answered, by defining the rate as a parameter (just like payment was) and then invoking *Goal Seek*.

5 A range of applications

We conclude this article with a range of applications of the recursive paradigm with the intention of showing that the method is not restricted to replacing numerical formulas with recurrence relations. For a modern, general treatment of the recursive paradigm in the spreadsheet environment, the work of Kreith and Chakerian [11] is worth reading.

Table 1: The recurrence for Pascal's triangle.

	A	B	C	D	E
1	$\binom{0}{0}$	$\binom{1}{1}$	$\binom{2}{2}$	$\binom{3}{3}$	$\binom{4}{4}$
2	$\binom{1}{0}$	$\binom{2}{1}$	$\binom{3}{2}$	$\binom{4}{3}$	$\binom{5}{4}$
3	$\binom{2}{0}$	$\binom{3}{1}$	$\binom{4}{2}$	$\binom{5}{3}$	$\binom{6}{4}$
4	$\binom{3}{0}$	$\binom{4}{1}$	$\binom{5}{2}$	$\binom{6}{3}$	$\binom{7}{4}$
5	$\binom{4}{0}$	$\binom{5}{1}$	$\binom{6}{2}$	$\binom{7}{3}$	$\binom{8}{4}$

This book illustrates that such an environment is well-suited to investigations into dynamic modeling and chaos theory. For an even more recent treatment of a wide-range of applications, we recommend articles that have appeared in earlier issues of *Spreadsheets in Education*; in particular, in the domain of geosciences, the excellent paper of Fratesi and Vacher [7].

The reader might like to open the attached spreadsheet, *recursion.xls*, where we give working models of each of the applications discussed below.

5.1 Combinatorial numbers

The foregoing examples showed some of the simplest recursive functions. Their definitions contained only one trivial and one recursive case. There are many functions with more complex definitions. One well-known example is the collection of combinatorial numbers forming Pascal's triangle (see Abramovich and Brantlinger [1] for a discussion of the pedagogical aspects of using spreadsheets to explore this topic). Their most common definition is

$$\binom{m}{n} = \begin{cases} 1 & \text{if} \quad n = 0 \\ 1 & \text{if} \quad n = m \\ \binom{m-1}{n} + \binom{m-1}{n-1} & \text{if} \quad 0 < n < m \end{cases} \tag{6}$$

It consists of two trivial cases (for the extreme values of n) and one recursive (for the values in between).

Our first problem is to position the calculation into the sheet. Let the values of m grow with the column numbers. Then the values of n also grow with the column numbers. One can see that all the pairs in the sheet are different because the difference between m and n grows in each row. In the first row, it equals zero. In the second row, it is one, in the second it is two, and so on. Table 1 shows our proposed organization of combinatorial numbers in Pascal's triangle.

Notice that the Pascal triangle is slightly rotated. Usually, it is formed by rows of m, whereas in Table 1, its rows are the diagonals of the sheet. The first standard row corresponds to $m = 0$. Here it is just the cell `A1`. The second row (for $m = 1$) is formed by `A2` and `B1`. The third row is made by `A3`, `B2`, and `C1`. Thus, the k-th row of the Pascal triangle starts at the k-th row of the sheets and occupies its corresponding diagonal.

It is easy to see that first row and the first column must contain just 1's. These are our boundary conditions, when $n = 0$ and $m = n$. In this representation, the numbers referred to in the recursive case are the upper and left neighbor of the calculated value so that `(B2) = A2 + B1.`

Filling the formula into all empty cells of the sheets computes Pascal's triangle, as shown in Figure 7.

	A	B	C	D
1	1	1	1	1
2	1	2	3	4
3	1	3	6	10
4	1	4	10	20
5	1	5	15	35
6	1	6	21	56

	A	B	C	D
1	1	1	1	1
2	1	=A2 + B1	=B2 + C1	=C2 + D1
3	1	=A3 + B2	=B3 + C2	=C3 + D2
4	1	=A4 + B3	=B4 + C3	=C4 + D3
5	1	=A5 + B4	=B5 + C4	=C5 + D4
6	1	=A6 + B5	=B6 + C5	=C6 + D5

Figure 7: Pascal's triangle calculated by recursion.

5.1.1 Strips of Pascal's triangle

The same set of numbers can also be defined by other recursive relationships. The first one is a rarely used but simple formula:

$$\binom{m}{n} = \begin{cases} 1 & \text{if} \quad n = 0 \\ \frac{m}{n}\binom{m-1}{n-1} & \text{if} \quad 0 < n \leq m \end{cases} \tag{7}$$

Notice that the elements bound by the recursive case form the rows in our representation of the triangle. On the other hand, the cell must "know" its position in the sheet. Otherwise m and n must be specified as constants somewhere in the sheet—our readers are welcome to form such a solution as an easy exercise.

The position of the cell can be calculated using `ROWS` and `COLUMNS` functions. For given m and n, its corresponding combinatorial number $\binom{m}{n}$ is located in the cell for which the next relationships hold:

m = number of rows + number of columns -2

n = number of columns -1

The implementation starts with the trivial case. The number 1 is typed in the first cell of the addressed row. For example, if our goal is the value of $\binom{8}{4}$, we work in the fifth row e.g. start with `A5`. Into `B5` we build the recursive formula using the `ROWS` and `COLUMNS` functions in the form:

```
(B5) = ((ROWS(B$1:B5)+ COLUMNS($A5:B5)-2)/(COLUMNS($A5:B5)-1)) * A5
```

	A	B	C	D	E
1					
2					
3					
4					
5	1	5	15	35	70
6					
7					

Figure 8: Pascal's triangle calculated by rows

	A	B	C	D	E
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	70
6	1	6	21	56	126
7	1	7	28	84	210
8	1	8	36	120	330

Figure 9: Pascal's triangle calculated by columns

where `ROWS(B$1:B5)` calculates the number of rows between the beginning of the strip and the given cell (inclusive). `COLUMNS($A5:B5)` calculates in a similar way the number of columns. Spreading the formula along the fifth row produces the section of the triangle shown in Figure 8.

In the chapter on recursion in Polya [21], one can find inspiration for forming similar selected portions of Pascal triangle.

5.1.2 Complex recursive formulas

Another recurrence defining of Pascal's triangle is quite complex. It goes as follows:

$$\binom{m}{n} = \begin{cases} 1 & \text{if} \quad n = 0 \\ \binom{m-1}{n-1} + \binom{m-2}{n-1} + \ldots + \binom{n+1}{n-1} + \binom{n}{n-1} + \binom{n-1}{n-1} & \text{if} \quad n > 0 \end{cases} \tag{8}$$

In our rectangular representation, the long summation contains a strip in the nearest column to the left of the calculated cell – starting from the neighboring cell up to the top as shown in Figure 9. To calculate the value of `D6`, all values from `C1` to `C6` must be summed. Thus, the method of calculation becomes obvious: we fill the first column with 1's. Then we use the formula `(B1) = SUM(A$1:A1)`in `B1` and spread it into all remaining cells of the sheet.

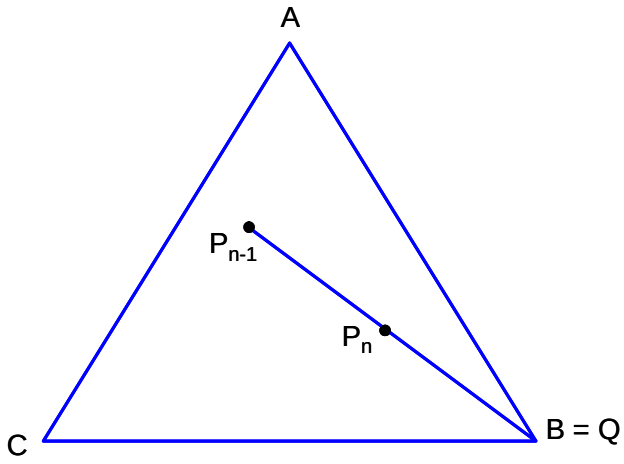

Figure 10: Locating the next point

5.2 Chaos theory

We will return to Pascal's triangle in a moment, but let us first digress to consider the *Chaos Game* of Barnsley. The *Chaos Game* provides a fascinating introduction to fractals and demonstrates how order can be derived from an apparently random activity. The activity itself can only be described in recursive terms.

Points are positioned in triangle ABC according to the following rules.

1. Point 1 is a randomly chosen point inside the triangle.

2. Point n is derived from point $n-1$ by randomly choosing Q as point A, B or C and making P_n the midpoint of QP_{n-1}.

Figure 10 shows this for the case when Q is chosen to be B, but it could equally well have been A or C.

The behaviour of the *Chaos Game* can be simulated in Excel by recursive formulas to make a table of values such as those shown in Figure 11.

In Figure 11, the initial values of the point $P1$ are shown as $(2, 2)$ and names are created for the x and y coordinates of triangle ABC. The formulas[4] used are:

```
(B3) = RANDBETWEN(1, 3)
(C3) = OFFSET(base, B3, 1)
(D3) = OFFSET(base, B3, 2)
(E3) = scale*C3 + (1 - scale)*E2
(F3) = scale*D3 + (1 - scale)*F2
```

[4]The RANDBETWEEN function is part of the Excel Analysis toolkit, which may need to be installed before the function can be used.

	A	B	C	D	E	F	G	H	I	J
1	Point	Random	chosen-x	chosen-y	x-value	y-value		Scale	0.5	
2	1	3	4	0	2	2				
3	2	2	2	4	2	3		Base	x	y
4	3	2	2	4	2	3.5		A	0	0
5	4	3	4	0	3	1.75		B	2	4
6	5	2	2	4	2.5	2.875		C	4	0
7	6	2	2	4	2.25	3.4375				
8	7	2	2	4	2.125	3.71875				
9	8	2	2	4	2.0625	3.859375				
10	9	3	4	0	3.03125	1.929688				
11	10	3	4	0	3.515625	0.964844				
12	11	1	0	0	1.757813	0.482422				
13	12	1	0	0	0.878906	0.241211				
14	13	3	4	0	2.439453	0.120605				
15	14	1	0	0	1.219727	0.060303				
16	15	3	4	0	2.609863	0.030151				
17	16	1	0	0	1.304932	0.015076				

Figure 11: Spreadsheet values for the chaos game.

To make a reasonable representation of positions that P_n can take, at least 3000 points need to be plotted, but given that there is space to calculate some 63000 points, the 3000 are easily accommodated. With this spreadsheet it is good to turn off the automatic calculation feature on the **Tools: Options: Calculation** tab.

As shown in Figure 12, the resulting image is that of the Sierpinski Gasket, although not yet complete. By augmenting the *Chaos Game* spreadsheet, the resulting pattern can be turned into a gasket for a square, pentagon, etc., depending on how many points in the original figure there are to choose from.

O'Sullivan shows how Pascal's triangle can be modified to demonstrate the same pattern. Using the diagonal form, make the following change: `(B2) = MOD(B1 + A2, 2)`.

This picks out where the elements of the triangle are odd, and then condition formatting can be used to expose the Sierpinski Gasket (see O'Sullivan [19]). Choose **Format: Conditional Formatting** and set the first condition to be equal to 1. After some rescaling of the cell size, the result is as shown in Figure 13.

5.3 Simulation

There are many situations where a simulation model relies on a recursive definition. A very modern example is O'Sullivan [19], while a more comprehensive treatment in that of Kreith and Chakerian [11]. Typically, the use of the Fibonacci sequence to model a rabbit population is used in the classroom. A similar model, but one that is less deterministic than the Fibonacci model, can be described in terms of tossing coins. The growth of a rabbit population is defined by the following rules:

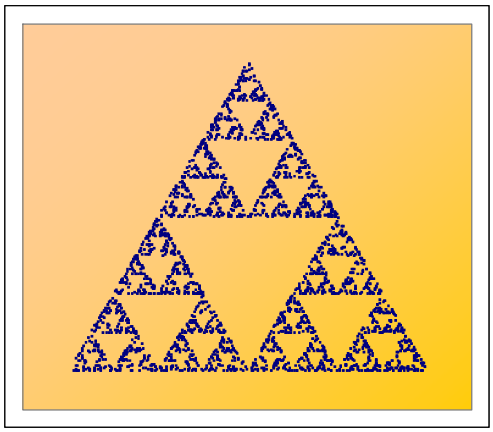

Figure 12: Chaos game chart

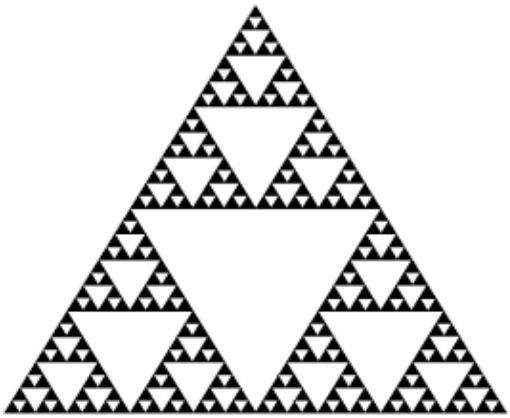

Figure 13: Sierpinski's gasket

1. Initially there is a population of 10 rabbits. Using normal ecology conventions, we consider only female rabbits as they are able to breed.
2. Each breeding season, the females have a 50% chance of breeding and have three offspring when they do.
3. Also, predators take their toll on the population, with each rabbit having an even chance of survival to the next breeding season.

The process can be modeled by tossing the coins on the table and counting the number that have Heads showing, an event that has an even chance of occurring.

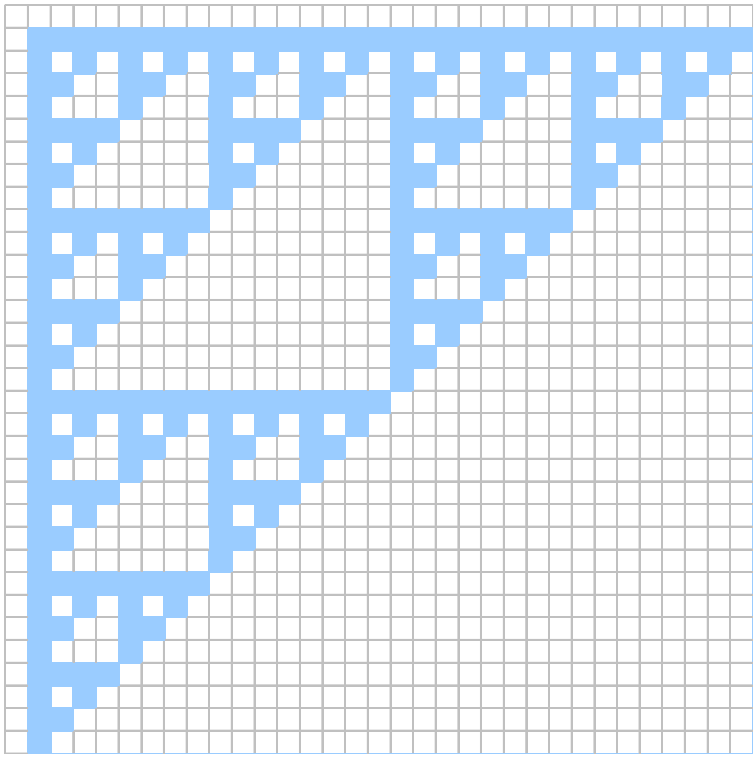

Figure 14: The Sierpinski gasket and Pascal's triangle

Students can be asked to both derive a deterministic model and run a simulation with coins (recording only Heads as a Birth) and to comment on the results. To run the simulation requires a large number of coins, as the population can quickly get quite large, and as a result the data gathering part of the exercise is likely to be very time-consuming.

The same simulation can be set up in Excel by basing the coin simulation on random numbers and using the `CRITBINOM` function[5]. This function returns the number of suc-

[5] For some reason, there appears to be an unspecified limit on the number of trials allowed with the `CRITBINOM` function, thus in the simulation given, a sample size of 100 was used and then scaled to the actual size of the rabbit population.

cesses of a Bernoulli trial needed achieve a given probability or criterion. Thus if there were 100 rabbits with a 50% chance of breeding, a simulation of this is obtained by: `= CRITBINOM(100, 0.5, RAND())/100`.

Suppose RAND() evaluates to 0.3, then this formula returns the proportion of breeding rabbits that would have a 0.3 chance of happening. This function can be used for predation as well as breeding, and produces the data shown in Figure 16.

Year	Random 1	Random 2	Breed	Predate
0			10	10
1	55	48	26	12
2	42	48	27	12
3	52	49	30	14
4	54	54	36	19
5	51	50	48	24
6	42	53	54	28
7	53	58	72	41
8	52	51	104	53
9	52	46	135	62
10	44	54	143	77

Figure 15: Simulation of rabbit breeding

The randomized model needs to be compared with the deterministic model in which the number of rabbits that breed in season n is $0.5 \times 3 \times P_{n-1}$and this is added to the previous season's population, while the predation reduces this by half. Thus the recursive definition of P_n is given by $P_n = 1.25P_{n-1}$. The deterministic and simulated data can be compared on a graph as that shown in Figure 17.

When the calculation mode has been set to Manual, by choosing the **Tools: Options: Calculation** tab, it is possible to run many simulations and to explore the way in which the deterministic model is occasionally matched by the simulation model, but often the two vary considerably ... particularly on those occasions when the predators get the upper hand and the population is wiped out entirely, but on other occasions, the two models agree quite well, as is the case in the simulation run on which Figure 16 was based.

Before leaving the topic of simulation, we note that the modern spreadsheet is not just useful for teaching simulation modeling, but also allows "industrial-strength" real-world simulation; just two examples are those of Noble and Sugden [24], and Thiriez [25].

5.4 Mazes

The recursive concept lies at the heart of many maze-threading algorithms. Who has not followed this rule:

Keep your hand on the left-hand wall.

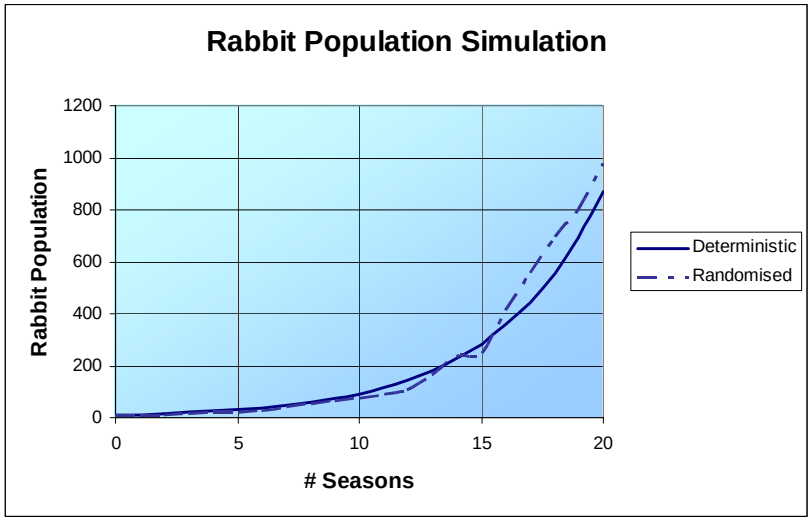

Figure 16: Graphs of the rabbit population simulation

In the same vein, there is the *greedy algorithm* concept, defined rather precisely by McMahon [15] as *those non-backtracking algorithms in which irrevocable decisions of global significance are made on the basis of local information.* We summarize this as: *do the best you can from where you are.*

However, to write such programs in a programming language often requires a hefty overhead of data input and output and the simplicity of the recursive definition of the process can be lost. To demonstrate the power of recursion in this context, the spreadsheet can greatly simplify data input and output and at the same time clearly exemplify how and why the recursive process works.

Suppose the maze-threading problem is posed as:

A grid of numbers is given. Starting at the top left corner and moving only to the right and downwards, find the path with the smallest total.

First, a random grid can be established in cells `C4:N15` with the `RANDBETWEEN` function.

```
(C4) = RANDBETWEEN(1, 10)
```

This formula is dragged over the range `C4:N15` to give a 12 by 12 grid of random numbers. We will denote this table of random numbers by $R(i, j)$.

The first stage of threading the maze is to make a *lowest-score* grid of the same size in `R4:AC15`. The values in this grid are based on recognising that the lowest total that can be in any position in the *lowest-score* grid is found by the recursive formula:

$$S(i+1, j+1) = R(i+1, j+1) + MIN\left(S(i+1, j), S(i, j+1)\right)$$

This recursive formula translates into cell formulae such as:

```
(R4) = C4
```

```
(S5) = D5 + MIN(R5, S4)
```

That is, apart from the top left cell, the value in a cell is found by adding the lesser of the values above it and to its left to the current value of the cell. To enable the `D5 + MIN(R5, S4)` formula to be used in every cell of *lowest-score* grid, we create a wall of high numbers around the outside of `R4:AC15`. This ensures that along the edges, the total is always calculated from numbers inside the grid.

If a trace of the actual path is needed, the formulas are a little more complicated, but still fully recursive in nature (see Figure 16). The path into cell X is from either of cell A or B, and from A, you only choose to go to cell X if its total-so-far is less than that of cell Z. There is the additional requirement, of course, that the path goes through A in the first place. Thus the journey path is created with the formulas:

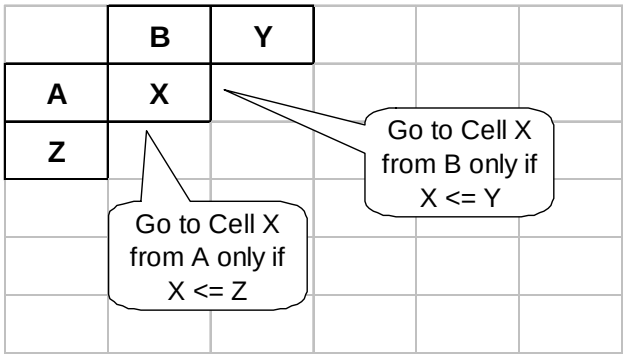

Figure 17: The conditions that apply to choosing cell X

```
(AG4) 1
(AH5) = IF(AND(S5<=T4, AH4 = 1), 1, IF(AND(S5<=R6, AG5=1), 1, 0))
```

This time a wall of 0 values is placed round the grid to indicate the path doesn't use a cell that is outside the grid and conditional formatting can be used to highlight the cells in the *journey* grid that the path uses.

6 Conclusion

The examples given in Section 5 underline the inherent simplicity of a recursive formulation of a problem. When, in the 19^{th} century, Fourier [6] explained the use of infinite series to represent functions, he did more than open up a very fertile area of mathematics. He challenged the conventional thinking that a function could only be expressed as a finite combination of algebraic symbols. In doing so, Fourier changed the mathematicians' view of the function and the benefit of the changes was widespread. In a similar way, the recursive definition of a function or process represents a changing view of the function and it is one that still has to find its place in the educational context.

The paradigm of recursive thinking differs from functional thinking in that it represents the rules that determine the transition between states rather than a fixed functional

form. Our definition of recursion in terms of the spreadsheet emphasizes this distinction by referring to formulas based on the values of neighboring cells that will alter depending on the position of the formulas in the spreadsheet. As a result, we find that the use of recursive formulas relies on the power of the spreadsheet to generate tables of information, but this is a positive feature of their use rather than an overhead, because spreadsheets are designed to support this type of layout and provide support through facilities such as the *Fill Handle* and *Conditional Formatting*. Another benefit of the use of recursive thinking that we would claim lies in the simplicity of the formulas that express such relationships by comparison with formulas expressed in functional form.

We do not expect to learn how close we might have come to convincing the reader that the recursive method, when combined with the table features of the spreadsheet, provides us and our students with an unusually powerful tool for learning, exploring and doing mathematics. But we can hope that the impression that the reader takes away will sustain interest enough to continue exploration into how simple formulas, combined with a tabular layout really does open up new possibilities for students to gain insight into mathematical concepts. While we have pointed to a number of indicators suggesting that others recognize and have researched the value of recursive thinking within the spreadsheet context, we know that the impact of such insights on mainstream mathematics teaching have yet to be felt.

References

[1] Abramovich, S. and Brantlinger, A. (1998). Tool Kit Approach to Using Spreadsheets in Secondary Mathematics Teacher Education. In S. McNeil, J.D. Price, S. Boger-Mehall, B. Robin, J. Willis (Eds), *Technology and Teacher Education Annual*, 1998, 573–577. Charlottesville, VA.

[2] Abramovich, S. and Pieper, A. (1996). Fostering Recursive Thinking in Combinatorics through the Use of Manipulatives and Computing Technology. Mathematics Educator **7**(1): 4–12.

[3] Abramovich, S. and Sugden, S. J. (2004). Spreadsheet Conditional Formatting: An Untapped Resource for Mathematics Education. *Spreadsheets in Education* **1**(2): 85–105. Available online at http://www.sie.bond.edu.au/articles/1.2/AbramovichSugden.pdf.

[4] Baker, J. E., and Sugden, S.J. (2003). Spreadsheets in Education: The First 25 Years. *Spreadsheets in Education* **1**(1): 18–43. Available online at http://www.sie.bond.edu.au/articles/1.1/bakersugden.pdf.

[5] Dubisch, R. (1965) *Introduction to Abstract Algebra*, p15. Wiley, New York.

[6] Fourier, J. B. J. (1822). *La Theorie analytique de la chaleur*, Academie des Sciences.

[7] Fratesi S. E. and Vacher, H. L. (2004). Using Spreadsheets in Geoscience Education: Survey and Annotated Bibliography of Articles in the Journal of Geoscience

Education through 2003. *Spreadsheets in Education* **1**(3): 168–194. Available online at http://www.sie.bond.edu.au/articles/1.3/FratesiVacher.pdf.

[8] Hamming, R. W. (1962) *Numerical Methods for Scientists and Engineers*, page (v), preceding preface. McGraw-Hill, New York.

[9] Hardy, G.H. (1940). *A Mathematician's Apology*, Cambridge University Press, ISBN 0521427061.

[10] Hvorecký, J. and Trencanský, I. (1998). *Recursive Computations in Spreadsheets.* In Wei-Chi Yang, Kizoshi Shrirayanagi, Sung-Chi Chu, Gary Fitz-Gerald (editors): Proceedings of the Third Asian Technology Conference in Mathematics, Springer, Tokyo 1998, pp. 290–299.

[11] Kreith, K. and Chakerian, D. (1999) *Iterative algebra and dynamic modeling : a curriculum for the third millennium.* Springer, New York.

[12] Introduction to Marine Navigation, Chapter 1, www.marineplanner.com/bowditch/chapt-01.pdf.

[13] Lovászová, G. and Hvorecký, J. (2003). On Programming and Spreadsheet Calculations. *Spreadsheets in Education* **1**(1): 44–51. Available online at http://www.sie.bond.edu.au/articles/1.1/hvorecky.pdf.

[14] McCarthy, J. (1960) Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part 1. *Communications of the ACM*, April 1960.

[15] McMahon, G.B. (1989). *A structural taxonomy for algorithms.* Working Paper 1989 3 007, July 1989. School of Information and Computing Sciences, Bond University.

[16] Mendelson, E. (1998) *Introduction to Mathematical Logic.* Chapman & Hall.

[17] Mizrahi, A. and Sullivan, M. (2000). *Finite mathematics: an applied approach.* New York, Wiley.

[18] Noss, R. (2001) For a learnable mathematics in the digital culture. *Educational Studies in Mathematics* **48**(1): 21–46.

[19] O'Sullivan, B. (2003). *What lurks under Pascal's Triangle. Teaching Mathematics*: *Journal of the Queensland Association of Mathematics Teachers*, **28**(1): 4–8.

[20] Peano, G. (1889) *Arithmetices principia, nova methodo exposita,* Pamphlet published by the University of Turin.

[21] Polya, G. *Mathematical Discovery: On Understanding, Learning and Teaching Problem Solving.* Wiley Text Books, New York, 1981.

[22] Steele, G. L. Jr. and Sussman, G. J. (1978). *The Revised Report on Scheme, a Dialect of Lisp.* Massachusetts Institute of Technology. MIT AI Memo 452. January 1978.

[23] Sugden, S. J. (2001). Bits, Binary, Binomials and Recursion: Helping IT Students Understand Mathematical Induction. *Quaestiones Mathematicae, Journal of The South African Mathematical Society*, Supplement #1.

[24] Noble C, Sugden, S.J. (2002). Stochastic Recurrences of Jackpot KENO. Computational Statistics & Data Analysis 40: 189-205.

[25] Thiriez, H. (2001). Improved OR Education Through the Use of Spreadsheet Models. *European Journal of Operational Research*, **135**: 461–476.

[26] Weisstein, E. W. (2005a) "Peano's Axioms." From MathWorld, a Wolfram web resource. http://mathworld.wolfram.com/PeanosAxioms.html.

[27] Weisstein, E. W., (2005b) "Primitive Recursive Function." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/PrimitiveRecursiveFunction.html