School Algebra and the Computer[1]

David Tall

Mathematics Education Research Centre
University of Warwick
COVENTRY CV4 7AL

How are we to use the computer in the teaching and learning of algebra? In the long-term the new technology is introducing new possibilities that may radically change the algebra curriculum. However, in the short-term we already have the National Curriculum placing its template on the development of algebra in school. The recent regrouping of topics into five attainment targets has integrated number pattern with the development of algebraic symbolism. It seems natural to build from expressing patterns in words to expressing them in a shorthand algebraic notation, but, although this proves a sound tactic for the more able, there are subtle difficulties for the majority of children. Instead I shall advocate introducing algebraic symbolism by using it as a language of communication with the computer, through programming in a suitable computer language. This has two distinct benefits – it develops a meaningful algebraic language which can be used to describe number patterns, and it gives a foundation for traditional algebra and its manipulation.

"Fruit Salad Algebra"

Traditional algebra often begins by introducing children to letters standing for numbers and teaching them to manipulate and simplify expressions such as $3a+4b+5a$. To give success in the operation of simplification it is a common practice to explain that "$3a+4b$" stands for "three apples plus four bananas". Cyril Quinlan of the Australian Catholic University calls this "fruit salad algebra". It helps children cope with simplification of "$3a+4b+5a$" because "three apples plus four bananas plus five apples" is "eight apples plus four bananas" or "$8a+4b$". This interpretation of "letter as an object" fits with the previous experience of 5m as "five metres" or 10p as "ten pence", so it has a measure of familiarity and can lead to initial success. But it founders badly at a later stage when attempting to interpret $3a–5a$ or $(a–b)(a+b)$. Some authorities try to "explain" the first of these in terms of "adding 3 apples and taking away 5 apples" which is equivalent to "taking away 2 apples", but such a meaning will not stretch to cope with the second expression. How does one given meaning to $a^2–b^2$ in terms of "apples and bananas"? What does a square apple or a square banana look like? The "fruit salad" meaning of the symbolism must be changed to accommodate these new

Published *in Micromath*, 9 (1), pp. 38–41, 1993.

notions and this proves difficult for the vast majority of children who dissociate the symbolism from any real meaning, producing a sense of abstract alienation. It may very well increase short-term success but at the expense of wide-spread long-term failure for many.

Learning algebra through generalising pattern

Pattern has rightly become a central focus in the National Curriculum. But what was a separate arithmetic/algebraic strand for pattern has now been absorbed into a single algebra attainment target (AT3) in a way which has muddied the waters. Because pattern starts earlier in school than formal algebra, the use of pattern in the earlier levels of AT3 may give the impression that pattern is an appropriate foundation of formal algebra. Indeed it is a persuasive argument to see the development of algebra through an exploration of patterns, the description of these patterns in words, and the translation of these descriptions into algebraic notation, prior to moving into the traditional symbolic skills. Despite the attractiveness of this sequence – apparently embodied in the National Curriculum – there is solid research evidence that it proves difficult for he majority of children.

Consider the problem of placing paving slabs round a square pond. If the pond is 2 ft by 2 ft and 1ft square slabs are used, there will be 2 slabs on either side, making 8 in all, plus 4 in the corners – a total of 12. A 3 ft square pond requires 16 slabs, four by four requires 20 and so on. In an experiment in Australia[1], over seven hundred children were given this problem, supported by the image of a representation on a peg-board so that they could count round the various squares and gain experience of the numbers involved after the fashoin of figure 1.

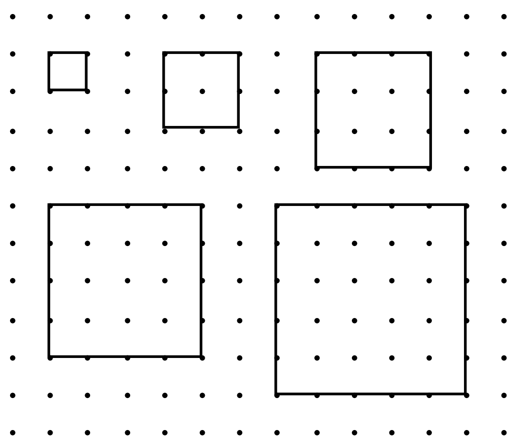

Figure 1 : putting square paving slabs round a square pond of various sizes

Most children were able to count the squares : 8 for a 1 by 1, 12 for a 2 by 2, 16 for a 3 by 3 and so on, and many could see that this requires "four more" at each stage. But less than 20% were able to express it in a way that led to the formula $4n+4$ for a side of length n. The pattern seen by most related one size to the next one up, rather than the relationship between the number of sides and the number of slabs. The further difficulty of using the letter n to express the pattern placed an even greater cognitive strain on them. All but the most able seemed to see little reason for using a letter to stand for a number.

Try thinking about this from the viewpoint of a child who sees arithmetic as a way of getting an answer to a problem. It is possible to *calculate* with numbers and get an answer. But until the value of n is known it is not possible to calculate the value of the expression $4n+4$ and, if the value of n *is* known, why not just use arithmetic anyway? Without being to rationalise this situation, children are faced with a dilemma of meaning when they first meet algebra.

Algebra with a computer

If we think of the symbolism of algebra as part of a "language of mathematics", then there is a powerful way of making it "less foreign" to those meeting it for the first time. This is to speak it in a context where the algebraic language is seen to make sense, such as communicating with the computer in a suitable programming language. Here I am *not* talking about the full gamut of learning to program, but simply to use that part of a programming language which uses variables to get into the idea of a letter standing for a number. BASIC and Logo are both suitable for this purpose, but whilst Logo is a better language for children freely exploring ideas, BASIC symbolism is closer to traditional algebraic notation and proves highly successful for a structured approach to algebra. Logo has a slightly more sophisticated way of dealing with variables. In Logo it is necessary to distinguish between the *name* of a variable and its *value*, with the notation **"n** being used for the name and the notation **:n** for its value. Thus if it is necessary to give the value 3 to the Logo variable **n**, the command required is **MAKE :n 3**. This subtle distinction has great value in more sophisticated usage, but for the purpose of giving children insight into the traditional algebraic idea of a letter standing for a variable, the BASIC commands are quite satisfactory.

A BASIC command such as

```
a=3
```

followed by the the command

PRINT a+1

will lead to the output **4**. It becomes easy for children across virtually the whole ability range to predict what happens with the command

PRINT a+2

The results of a such a command can be *predicted* before it is carried out, and then *tested* by typing the command. This environment for building and testing ideas helps the child to construct the ideas that a letter can stand for a number and an arithmetical expression involving that letter can be evaluated in the usual arithmetic way.

A practical game which encourages children to develop a mental picture of storage of numbers in named locations within the computer may be played using a "cardboard computer". This involves two large pieces of cardboard (figure 2). One of them represents the screen, on which an operator would place commands, and the other represents the computer memory with rectangular boxes as storage spaces which can be named by letters and in which numbers can be placed. By this method a group of children can carry out the internal workings of the computer, receiving a command on the screen, storing numbers in named locations, and carrying out computations to print results back on the screen.

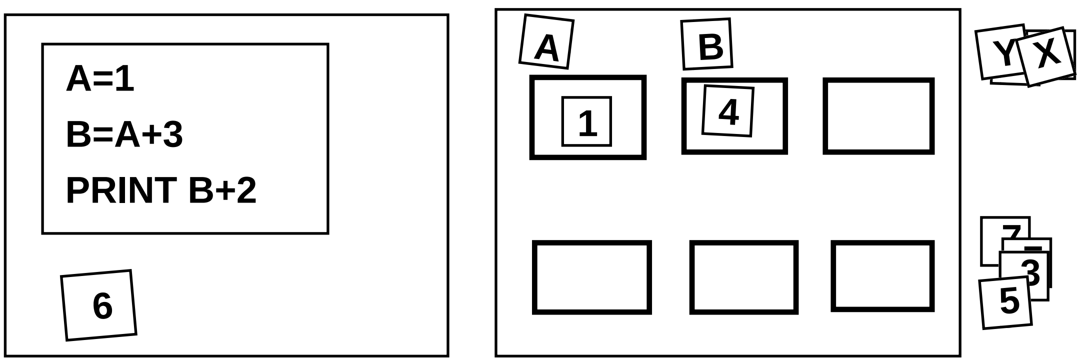

figure 2: A calculation using the **cardboard computer** – put1 in a store labelled **A**, put this value plus 3 in the store labelled **B**, then print the value in **B** plus two.

When Michael Thomas and I considered using this in a Y8 classroom, we were concerned that secondary school children might consider it too juvenile. This did not happen. Many actually preferred the action game using the cardboard computer to typing commands into the real computer. On reflection we saw that the two activities were complementary in a very subtle way. In the practical game the children had to carry out the *process* of evaluation for themselves. In this way they would see that the process of calculating the value of **2*(a+b)** and **2*a+2*b** involved *different* sequences of arithmetic operations. Using the computer, the commands **PRINT**

2*(a+b) and **PRINT 2*a+2*b** give instant responses, so that the focus of attention is on the *result*, that they always both give the same value. In this way it is possible to consider the equivalence of the expressions without being distracted by having to carry out the arithmetic every time.

Although BASIC is quite close to traditional algebra, there are differences, such as the fact that the multiplication sign must be explicit, writing **2*x** instead of $2x$, and the power x^2 must be written as **x^2**. To give experience in communicating with the computer in standard algebra requires the use of special software which accepts both algebraic and computer notation.

Two pieces of software that allow standard input of algebra and evaluate numerical expressions are *The Maths Machine*[2] (figure 3) and *The Function Calculator*[3] (figure 4). The *Maths Machine* allows the user to specify numerical values of variables and to calculate numerical values of expressions. Figure 3 shows an investigation to see if **2(x+y)** always gives the same numerical value as **2x+2y**. The *Function Calculator* allows more sophisticated calculations, with a column for the letter name of a variable, a column for its current numerical value and a column for an optional formula to calculate the value of the variable. Figure 4 has a variable **a** whose value has been specified as 4, and variables **b**, **c**, **d** are assigned the values given by the formulae **2+3a**, **5a** and **2+3*a**. This allows children to investigate how the computer interprets these expressions, to see that the first and the third always give the same values but the second does not.

In a practical experiment using the *Maths Machine* as part of an integrated teaching scheme, children found it easy to use and to relate to the other activities. As might be expected, with more emphasis on meaning and less on manipulation, when compared with other children following a traditional approach, just after the experiment they were better at sophisticated questions (such as "what is the perimeter of a rectangle *D* by 4?") and less successful on manipulation exercises (such as "simplify $4a+3b+2a$"). But a term later, after a brief revision (without computers) for all concerned, those using the computer retained their conceptual skills and actually improved in their manipulation to reach a level better those with a traditional background[4].

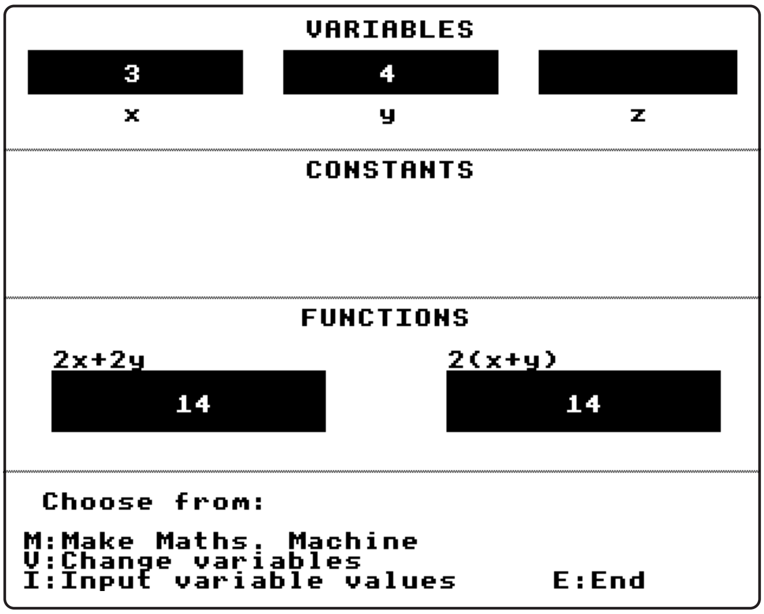

Figure 3: **The Computer Maths Machine** here calculates that **2x+2y** and **2(x+y)** both give the value 14 when **x** is given the value 3 and **y** is 4.

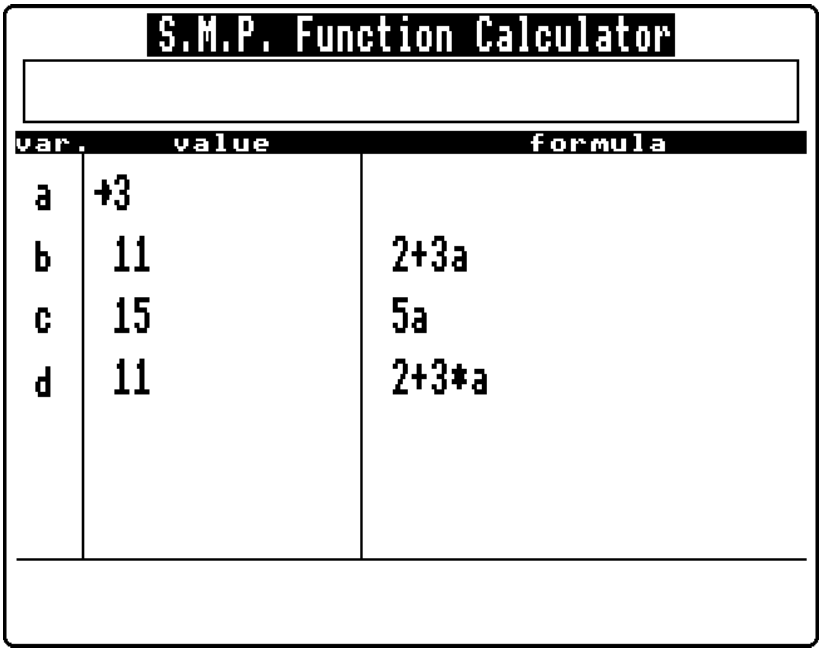

Figure 4: **The Function Calculator** is here set up to investigate the values of **2+3a**, **5a** and **2+3*a** when **a** is given various values. In what sense are the expressions "the same"?

Other Activities with a Computer

In advocating an approach to algebraic notation using programming, the main idea is to use algebraic language in a context where the child can try out expressions and predict, then test, their meaning. There is a variety of software available that enables children to explore number patterns and some of it involves the use of algebraic notation which can be used for communication in a predictable way..

Spreadsheets – originally written for business use – have educational uses. For instance, the very way in which spreadsheets operate by calculating the values of one location from the values in others using a formula involves an act of programming not unlike programming in a computer language.

Graph-plotting is an important aspect of algebra in the National Curriculum. Using the computer can give meaning to formulae used for plotting graphs. For instance, a "top maths group" in a middle school (Y7) easily compiled a table of values for the two sides x, y of a rectangle with perimeter 20, because they knew the values of x and y added up to 10. So if x is 3, then y must be 7. But, although this *seemed* to be like algebra, it was not, because the children were just using the letters x and y as *names* for the sides, not for their numerical values. When the table of values for x against y was plotted using a graph-plotter, the points clearly lay in a straight line. But it was only when the software required the user to "express y as a formula in x" that one of the most able in the class suggested "10 take away x" and the graph of y=10–x was drawn in its entirety. Any entered value of x, say x=3.5, then led to the corresponding y=6.5 being calculated and the point plotted on the graph. The fact that the formula 10–x could be used by the computer to calculate y gave meaning to the symbolism which was not apparent when algebra is used in everyday communication between people. Once again "speaking to the computer in algebra" proved its special value.

In all these activities, communication with the computer using algebraic notation in a way which is predictable and testable proves to be invaluable for giving meaning. It can pay dividends in exploring number patterns too. For if one has a language in which to express generalities, then this can be immediately used to describe number patterns as they occur. This gives a viable alternative sequence of activities to introducing algebraic notation as a shorthand for expressing pattern. It is easier to express the meaning if the shorthand is meaningful first.

[1] reported by John Pegg of the University of New England, Australia at a conference on "New Directions in Algebra Research", Queensland University of Technology, September 1992.
[2] available for the BBC Computer in *Secondary Mathematics with Micros – A Resource Pack*, from the Mathematical Association.
[3] available for BBC, Nimbus & Archimedes computers as part of *Real Functions and Graphs*, Cambridge University Press.
[4]Tall D. O. & Thomas M. O. J., 1991: "Encouraging Versatile Thinking in Algebra using the Computer", *Educational Studies in Mathematics*, **22** 2, 125–147.