EXPRESSING GENERALITY: A COMPUTATIONAL APPROACH

An outline paper to the Asian Technology Conference in Mathematics

Richard Noss

London Knowledge Lab, Institute of Education, University of London

This paper is derived from work by the MiGen project team. I am particularly grateful to Eirini Geraniou, Sergio Gutiérrez, Ken Kahn, Manolis Mavrikis & Darren Pearce who have been kind enough to let me cite their various contributions. In particular, I refer the reader to Pearce, D., Mavrikis, M., Geraniou, E., Gutierrez, S. (2008) and Pearce, D., Geraniou, E., Mavrikis, M., Gutierrez, S., Kahn, K. (2008).

I will present work arising from the MiGen project[1], which aims at designing, building and evaluating a pedagogical and technical environment for improving 11-14 year-old students' learning of mathematical generalisation. This project is in its relative infancy, and is very ambitious, involving novel strands of both educational and computation design. In this outline paper therefore, I will concentrate largely on the educational design questions, and I will only give a flavour of what is intended: I will demonstrate some of the real and projected features of the system at the conference. Before I do so, I would like to refer to the title of this outline and explain what I do and do not mean by 'computational approach'. At its simplest, I could mean using a computer to assist in the expression of generality. Viewed in this way, the expression of generality is a given, remaining more or less unchanged in the process of being 'approached' via a computer. We did indeed start this way! But as the project evolved, we came to realise that while we could express old things in new ways, *there were also new things to express.* This outline, therefore, will sketch the process by which we have achieved our current implementation (about a third of the way into the project timeline) and focus on these, essentially epistemological rather than merely pedagogical, issues.

Accommodating the idea of mathematical generalisation, recognising and analysing pattern and articulating structure are key aspects of mathematical thinking and a central rationale for algebraic expression. These have always been elusive ideas for students, who routinely find themselves unable to understand what generalisation is for, or what its classical language of expression – algebra – is all about. Even though students are able to identify and predict patterns (Mason, 2002), they are much less able to articulate a general pattern or relationship in either natural language or – more demanding still – in algebraic symbolism (Hoyles and Küchemann, 2002). The challenge, therefore, is to design situations that are rich in affording opportunities for the construction and analysis of patterns, and provide both a rationale and computational support for expressing generality.

The system (by which we mean both the technical and pedagogical elements) consists of three components. First, we are designing a microworld that supports students in their reasoning and problem-solving of generalisation tasks. Two further components aim to provide personalised support adapted to students' construction processes and foster an effective online learning community by advising learners and teachers as to which other students' constructions could

[1] See http://www.migen.org/ for more details. Funded by the Teaching and Learning Research Programme, Technology Enhanced Learning Phase-I; grant number: RES-139-25-0381.

fruitfully be viewed, compared, critiqued and built upon: these components will be the subject of other papers in due course.

Before I describe the system, we will briefly outline the rationale in the light of the relevant educational literature (the review of the technical literature, while equally important, will not be discussed here).

A SKETCH OF SOME RELEVANT EDUCATIONAL LITERATURE

Moss and Beaty (2006) claim that generalising problems are not themselves the cause of students' difficulties, but rather the way they are presented and the constraints of teaching approaches. In school, a standard approach leads towards pattern spotting, where students are taught techniques of finding the nth term by comparing consecutive terms usually presented in tables, and divorced as soon as possible from any structural features of the object being modelled. The difficulty is that this approach tends to generate correct answers (and therefore achieve requisite performance on tests) but fails to generate a motivation for generalisation, and, more importantly, does so by encouraging a breakage between symbolic (or arithmetic) representation and how it is represented (Hoyles and Healy, 1999).

Emphasizing the numeric aspect of patterning tends to lead to the variables becoming obscured (Noss et al., 1997; Noss and Hoyles, 1996) and students' ability to conceptualise relationships between variables, justify the rules and use them in a meaningful way becomes limited (Moss and Beaty, 2006). Teachers tend to teach "the abstracted techniques isolated from all context" or alternatively "the technique as a set of rules to be followed in specific contexts" (Sutherland and Mason, 2005) to help their students find the rule. Yet, as Mason (2002) points out, students have 'natural' powers to generalise in all kinds of contexts, a basic psychological substrate that is often ignored (or worse, destroyed) in routine mathematical teaching.

A further difficulty faced by secondary school students is the use of letters to stand for unknowns (see Küchemann, 1981 for seminal work in this regard). They struggle to grasp the idea of letters representing any value (Duke and Graham, 2007) and tend to lack the mathematical vocabulary needed to express generality. Even though it is a reasonable strategy to introduce algebra early, there is still the issue of how to introduce it so that students can make the transition from simple arithmetic to algebra. Other researchers (Warren and Cooper, 2008) report how students' written responses lack precision, which supports the view of primary school students' inexperience with mathematical language. Even if students succeed in expressing generality, their first attempts, naturally enough, are in natural language. Motivating an alternative way (and one that is seen by the students as more effective) is a crucial concern (see Redden, 1996).

Taking these concerns into account in the design of the system led us to try to find ways that would encourage students to write expressions in a general form, but without presupposing that they had already mastered the point of (let alone the mechanisms) of algebra. This is a challenge indeed: designing and implementing a quasi-algebra with both expressive power and learnability has taken the best part of a year. We also intend to rise to the challenge of allowing students to see different representations, such as symbolic, iconic, or numeric, and realise the relationships and the equivalence of these different representations. The following discussion and presentation at the conference will focus mainly on the design and implementation of the microworld, but I ask the reader to bear in mind that this is only the educational component of an effort based largely in the arena of computer science.

ITERATIVE DESIGN

We start by noting that a key aspect of our design methodology has involved working alongside teachers at every available point, trying ideas with (their) students, and involving teacher expertise as a central element of our decision-making process. This is worth stressing, as at least part of our intention is to develop a system that can support teachers in their efforts, and there are plenty of examples of failed computational systems that do not take sufficient account of the aspirations and methodologies of those they are intended to help.

Naturally, we have iteratively tested prototype components of the system in numerous ways, interleaving software development phases with small-scale pilot studies with individual or groups of students of our target age (11-14 years). We have also integrated feedback from teachers and teacher educators as well as the students who participated in the studies.

Our starting point was to develop a microworld in which students could express generalities of a kind very familiar to all English teachers. An archetypical example would be "Given a rectangular pond with an integer length and breadth, how many 1×1 tiles are needed to surround it?". The software (which we called ShapeBuilder) that aimed to assist in this process – the microworld - allows students to build various shapes (in the first instance, rectangles), by providing appropriate tools to build shapes *using expressions*. Once a shape is defined, the student can move it and attach it to other shapes. An important feature of the software was the ability to define expressions using properties of existing constructed shapes. For example, the student was able to obtain *icon-variables* which evaluate to the current values of the dimensions of the shape (see Figure 1).

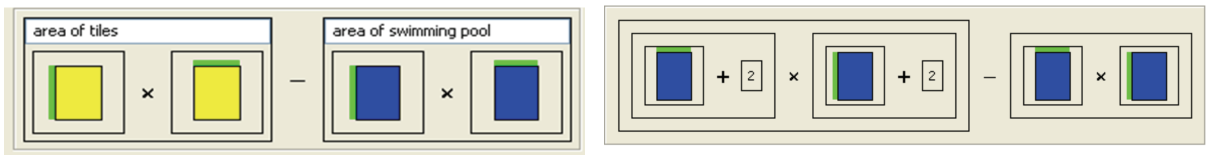

Figure 1. Some rules using icon-variables for (L)ength and (B)readth of a tiling round a rectangular 'swimming pool'. The left-hand rule expresses L_1B_1 – LB and the right-hand rule expresses (L+2)(B+2) - LB

These icon-variables – replacing the idea of *x* but also offering a much more dynamic representation of what is being generalised by representing an up-to-date small-scale ('thumbnail') image – act as components of an intermediate language for students: the idea was to ease the transition from a specific construction, and therefore expression, to a general construction represented by a general rule. So icon-variables are a good example of our first attempt at trying to find new ways to express classical generalities.

From a pedagogical point of view, we had to find a rationale for students to think about the particular with an eye on the general. This is, perhaps, the crucial pedagogic challenge: the student can *only* work with a particular example, and yet the solution to a given task is supposed to be for an infinite number of invisible and unspecified cases. No wonder the expression of generality is hard! One solution we came up with borrowed from previous research in dynamic geometry (Healy et al, 1994), in which students were challenged to construct a solution that is impervious to "messing-up": i.e. a construction that would be valid even if the values for either the width or the breadth of the pond were changed. So, for constructions that are not entirely general, changing the number of (independent variable) tiles 'breaks' the pattern. Thus changing various parameters of the problem provides a simple yet powerful mechanism for the student to judge whether a pattern is general or not, and provides students with the opportunity to realise that there is an advantage to thinking in terms of abstract characteristics of the task rather than specific values.

ShapeBuilder underwent many transitions and prototypes, many discussions and trials with students. Fundamentally, however, while we were reasonably content that the 'tiling' metaphor was a good starting point, the environment was constrained to the simple iteration of tiled shapes. This meant that – unless the student was exceptional in her mathematical orientation – the idea that it is the *pattern* that needs to be generalised is very hard to keep in focus; it is hard to work with the particular while keeping an eye on the general.

We needed, in fact, some alternative ways to think about the problem, and ShapeBuilder was clearly problematic, as the students required considerable support to ascertain just what was the pattern to be generalised. We required an exploratory environment that allowed the user to construct their patterns freely and analyse these constructions so as to obtain interesting general rules. To do so, we had to address a wide variety of points; below we point to four ways in which we extended the ShapeBuilder idea to create its successor: the *eXpresser*. I include these simply to whet the appetite of the reader: without demonstration and considerable further description, the following outline should be seen as merely indicative.

Basic Patterns
When the user creates a block of tiles, or looks at an already-existing block of tiles, it is very hard to see a simple example as an instance of a more general class. We struggled with this problem: it is, after all, another example of the mathematician's need to keep an eye on the general, even when (as is necessarily the case most of the time) one's attention is mainly on the particular. We found a way to address this problem (which I will elaborate at the conference) in which the student is able to create a 'unit pattern' which looks like a single block of tiles. However, inspecting its properties reveals that it is in fact something more complex: a collection of attributes (Figure 2a). Each of these attributes has an icon: the cogs icon represents the 'element count' of the pattern and the cogs icons with arrows indicate how far to the right/down each shape will be from its predecessor. Figures 2b-2e show various examples of basic patterns that can be obtained through editing these properties.

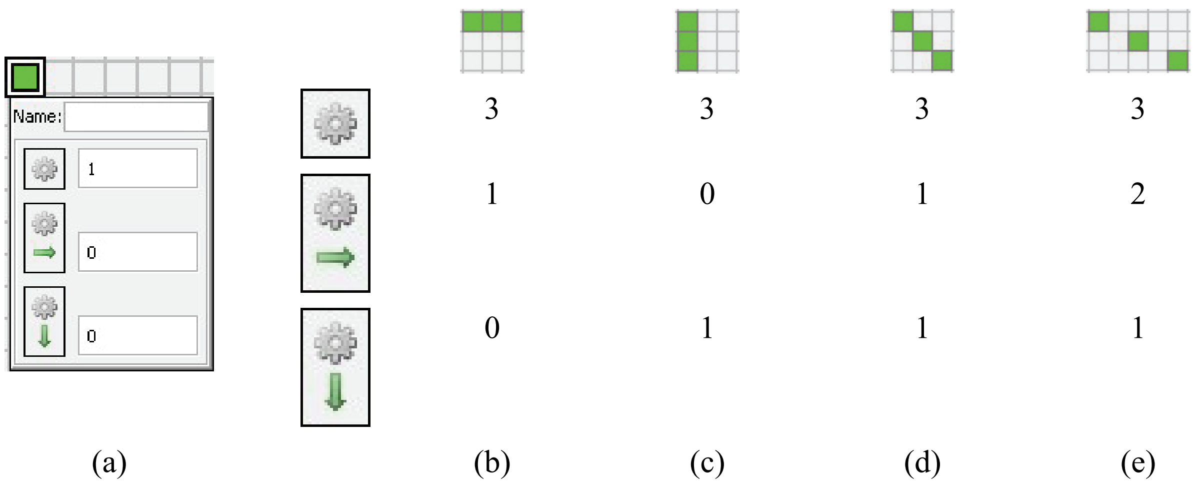

Figure 2: Basic Patterns

Generalising Patterns rather than (merely) shapes

Although interesting patterns can be created using such a basic mechanism, we required a way to step beyond basic shapes – in fact, beyond the idea of generalising shapes to generalising pattern. Currently the interface provides three ways in which to access this generic facility: creation of a horizontal sequence, creation of a vertical sequence and creation based on a ‘demonstration’ of the first two elements. These first two elements implicitly indicate what happens from one element of the pattern to the next. Using this information, the system is able to create the pattern.

Creating Dependencies

With the facilities described so far, the user is able to build complex patterns. However, there is also a need to make patterns depend on one another. Again, we struggled with a solution, and finally came up with a consistent way (i.e. consistent with the system as a whole) for the student to name patterns, and to describe the properties of new patterns in terms of attributes of existing, named, patterns. Again, more at the conference!

Allocations

The features above provide the user with mechanisms for constructing general pattern but no explicit way to analyse them so as to obtain general rules such as the number of tiles needed to surround a shape. This issue is addressed by allowing the user to specify the number of squares of each colour that they need for their construction, so that ultimately, the “allocation” of tiles drives the new pattern. This allows a rigorously interleaved process of construction and analysis. The advantage here – we have not tested it sufficiently at the time of writing - is that through this interleaving, the user may gain a deeper understanding of their construction. There are disadvantages too, that will be demonstrated, as well as our current attempts at overcoming them.

CONCLUDING REMARKS

The fundamental idea of the eXpresser is that patterns can be built and analysed so that the general case – “n” – is held in mind throughout the process, rather than being a final step. In traditional classrooms, it is easy for the teacher to delude his/herself into thinking that a sequence of carefully designed tasks will ‘lead’ the student from the particular to the general: if only it were quite so easy! Our continuing attempt to build a system that tries simultaneously to work with the particular and general is ongoing: progress will be reported at the conference.

Among the outstanding questions we are facing, is how best to provide the student with access to the full expressive power of the underlying engine. Since patterns can be built out of patterns and any attribute can be modified from one instance to the next (including modification of attributes themselves), the way in which the student should interact with this power poses an interesting interface design challenge. We have experimented with various possibilities in previous versions of the software but have yet to produce something intuitive and fully expressive.

Throughout this outline, we have ignored a fundamental aspect of our work: how to provide assistance to learners and advice to teachers based on analyses of individual students and the activities of the group overall. A crucial aspect of this in the coming months, will be to strike a balance between the need to take into account pedagogical and HCI considerations, or between the benefits of an open microworld for learning and constraints that assist in developing the intelligent analysis of student strategies by the machine. I will make some preliminary remarks on this ongoing work at the conference.

REFERENCES

[1] Duke, R. & Graham, A. (2007) Inside the letter, Mathematics Teaching Incorporating Micromath, 200, 42-45.

[2] Healy, L., Hoelzl, R., Hoyles, C. & Noss, R. (1994) Messing up, Micromath, 10, 14-17.

[3] Hoyles, C. & Healy, L. (1999) Visual and symbolic reasoning in mathematics: Making connections with computers. Mathematical Thinking and Learning, 1, 59-84.

[4] Hoyles, C. & Küchemann, D. (2002) Students' Understanding of Logical Implication, Educational Studies in Mathematics, 51(3), 193-223.

[5] Küchemann, D. (1981) Algebra. In Hart, K. M. (Ed) Children's Understanding of Mathematics: 11–16, 102–119 (London: John Murray).

[6] Mason, J. H. (2002) Generalisation and Algebra: Exploiting Children's Powers. In Haggerty, L. (Ed) Aspects of Teaching Secondary Mathematics: Perspectives on Practice, 105-120 (London, Routledge Falmer).

[7] Moss, J. & Beaty, R. (2006) Knowledge Building in Mathematics: Supporting collaborative learning in pattern problems, Computer-Supported Collaborative Learning, 1, 441-465.

[8] Noss, R., Healy, L. & Hoyles, C. (1997) The construction of mathematical meanings: Connecting the visual with the symbolic, Educational Studies in Mathematics, 33(2), 203-233.

[9] Noss, R. & Hoyles, C. (1996) Windows on Mathematical Meanings: Learning cultures and computers (Dordrecht: Kluwer).

[10] Papert, S. (1993) Mindstorms: children, computers and powerful ideas (BasicBooks, New York).

[11] Pearce, D., Mavrikis, M., Geraniou, E., Gutierrez, S. (2008) Issues in the design of an environment to support the learning of mathematical generalisation. In: Proceedings of the Third European Conference on Technology Enhanced Learning (EC-TEL08).

[12] Pearce, D., Geraniou, E., Mavrikis, M., Gutierrez, S., Kahn, K. (2008) Using Pattern Construction and Analysis in an Exploratory Learning Environment for Understanding Mathematical Generalisation: The Potential for Intelligent Support. In:Proceedings of the First International Workshop in Intelligent Support for Exploratory Environments; held in conjunction with the Third European Conference on Technology Enhanced Learning (EC-TEL08).

[13] Redden, T. (1996) Patterns language and algebra: A longitudinal study. In Clarkson, P. (Ed) Teachnology in Mathematics Education. Proceedings of the 19th Annual Conference of the Mathematics Education Research Group, 469-476 (Rotorua: MERGA).

[14] Sutherland, R. & Mason, J. H. (2005) Key aspects of teaching algebra in schools (QCA, London).

[15] Warren, E. & Cooper, T. (2008) The effect of different representations on year 3 to 5 students' ability to generalise, ZDM Mathematics Education, 40, 23-37.