Using Pattern Construction and Analysis in an Exploratory Learning Environment for Understanding Mathematical Generalisation: The Potential for Intelligent Support

Darren Pearce[1], Eirini Geraniou[2], Manolis Mavrikis[2], Sergio Gutierrez-Santos[1], and Ken Kahn[2]

[1] Birkbeck College {darrenp,sergut}@dcs.bbk.ac.uk
[2] Institute of Education {e.geraniou,m.mavrikis,k.kahn}@ioe.ac.uk

Abstract. Few systems exist that support learners explicitly in the process of learning mathematical generalisation. This paper presents the eXpresser, part of a new system that seeks to address this issue by providing the user with a microworld for the construction and analysis of general patterns. The design includes the provision of sophisticated intelligent support that assists learners and teachers throughout their various interactions with the system. Given the open and exploratory nature of the environment and the resultant freedom it affords, integrating such intelligent support poses a significant research challenge. This paper describes the system in detail and discusses a variety of ways in which we intend to meet our research goals for providing intelligent support.

Key words: microworld, generalisation, exploratory learning environments

1 Introduction

This paper presents the eXpresser, a mathematical microworld in which users can build and analyse general patterns. Within this microworld, the user can explore, experiment, and, through developing their own models, actively construct their own understanding of the idea of mathematical generalisation. The eXpresser is part of a larger system that is being designed and developed by the MiGen Project.[3] The aim of the project — to improve the mathematical generalisation abilities of 11–14-year-old children — addresses a real need in secondary mathematics education since many students in this age-range perceive generalisation and indeed algebra as difficult and unnecessary with no relevance to the way in which they approach problem-solving (as discussed in [1]).

Despite this significant problem, a paucity of systems exist to help children embrace algebra and understand its power. The MiGen Project is seeking to

[3] See http://www.migen.org/ for more details. Funded by the TLRP, e-Learning Phase-II; grant number: RES-139-25-0381

address this issue and, in the light of research that suggests that significant pedagogic support is needed for the student during their interaction with such environments (see [2]), the design of the system also includes two intelligent components: the eGeneraliser which will model the user's understanding of generalisation and scaffold their further exploration; and the eCollaborator which will intelligently assist users in discussing their ideas and (mis)conceptions with other users.

As important (and perhaps more importantly), the philosophy behind the development of the system is not to replace teachers but to *support* them. As a result, all components of the system will provide appropriate potentially-intelligent interfaces for the teacher to assist them in their goals for the lesson and to integrate the system into the classroom context.

After a brief discussion of related work (Section 2), Section 3 presents the current version of the system in detail as a basis for discussion in Section 4 of the potential for intelligent support, both for the student *and* the teacher.

2 Related Work

Several learning environments have been developed and integrated in classroom contexts over the last few years that attempt to help students in algebra and problem-solving. However, most of them expect students to have a clear notion of the concept of variables and an understanding of algebraic notation. Therefore, they do not deal explicitly with the difficulties students face when they are involved in generalisation (for a review of these difficulties see [1]) particularly because they usually provide ready-made abstractions and representations instead of allowing students to construct them themselves.

Closest to our work is research on mathematical microworlds [3, 4]. However, the majority of these revolve around geometric concepts or are usually targeted to students who already have a level of understanding of algebra. These tend not to focus on algebra directly but on developing an understanding of various representations such as tables and graphs (e.g. [5]).

One successful approach, in relation to generalisation from patterns enables students to engage with models that they are constructing themselves. For example, in Mathsticks [6], students work on patterns and regularities constructed out of matchsticks using a subset of Logo commands in order to explore how variables relate to each other and make relationships explicit using different representational forms. Despite some successes, difficulties remain, and these tend to coalesce around the need for significant pedagogic support from the teacher to provide a bridge to algebraic symbolism and generalisation.

The aim of our project is to develop tools that provide assistance to learners and advice to teachers based on analyses of students' interactions with the system. The rationale behind this aim is that the freedom of exploration in microwords entails the risk that the student may not 'accept or notice the educator's agenda' [4] or that they do not always manage to take advantage of the expressive environment that the microworld provides [7]. This is a serious issue in a classroom situation where it is difficult to attend to the interactions and the understandings that students develop over the course of these interactions.

Despite the existence of intelligent tutoring systems that attempt to help students with algebra (e.g., [8]), they are again targeted to problem-solving and are suitable for students who are following a particular procedural method of problem solving. The closest related work is the Motions microworld [7] developed in the early 1990s. Although it does not deal with algebra explicitly, there was an attempt to incorporate aspects of intelligence in order to address 'misconceptions of basic concepts and of generalisation' [7]. The design proposed for the 'intelligent microworld' gives the student control to decide when, and what for, to invoke the tutor asking questions such as 'why', 'predict', 'generalise' and 'challenge-me'. However, the system was not developed further. Recently developed intelligent exploratory learning environments (e.g. [9, 10]) are still within the realm of investigating specific concepts by providing a well-defined open learning environment that usually allows exploration of a model or a simulation. Such environments usually allow students to work with a pre-determined list of variables, restricting their reasoning [11] rather than engaging learners in the construction of their *own* models.

3 System Description

Earlier iterative development towards the eXpresser developed ShapeBuilder, a prototype microworld for exploring generalisation. Although many of its features were pivotal to our thinking, ultimately, the underlying engine was not applicable to many typical generalisation tasks within the National Curriculum. This is due to the fact that many such tasks make use of the notion of *repetition* within patterns. Figure 1 shows two typical problems, the 'footpath' which is a repeated tiling and the 'pyramid' which not only makes use of repetition but repetition with *changes* since each layer increases in size relative to the previous layer.

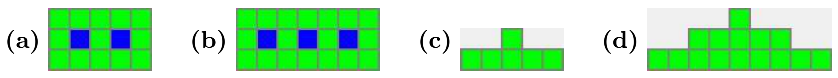

Fig. 1. Example patterns. **(a)** the footpath pattern with 2 (dark) blue squares; **(b)** with 3 blue squares; **(c)** the pyramid pattern with 2 levels; and **(d)** with 3 levels.

When investigating such tasks, students would typically be asked to work out its general structure so that they could describe other instances of the pattern. In the case of the footpath, for example, they would be able to describe what it would look like if it had 10 (dark) blue squares surrounded by (light) green squares and, similarly, in the case of the pyramid, what it would look like if it had 6 layers. In addition to this, students are also usually asked to investigate questions such as the general rule for the number of green squares needed to surround any given number of blue squares in the footpath pattern.

Following from these observations, the eXpresser provides an exploratory environment that allows the user to construct their patterns freely and analyse these constructions so as to obtain interesting general rules. Figure 2 shows the current interface and briefly describes each of the available tools.

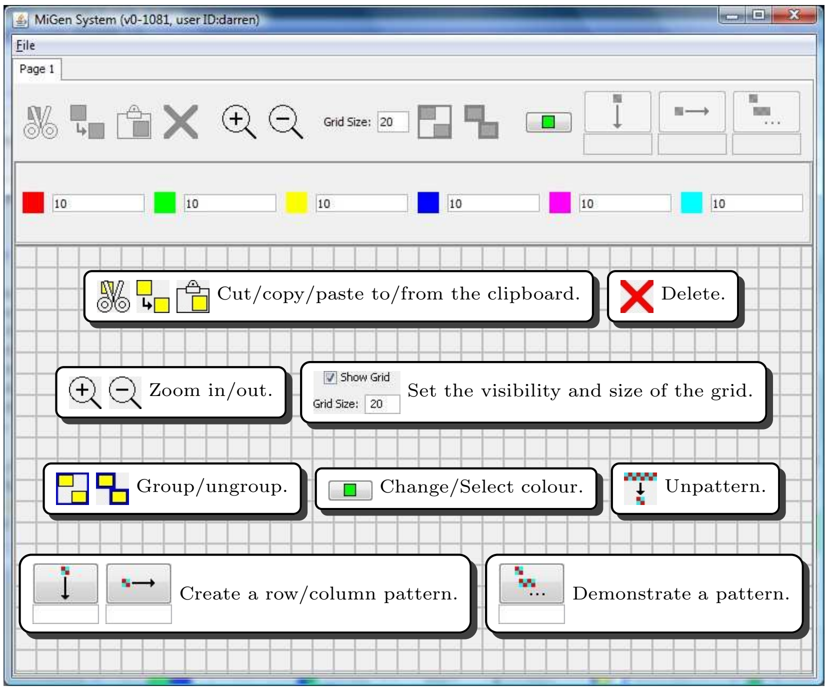
MiGen System (v0-1081, user ID:darren)
File
Page 1
Grid Size: 20
10
Cut/copy/paste to/from the clipboard.
Delete.
Zoom in/out.
Show Grid
Grid Size: 20
Set the visibility and size of the grid.
Group/ungroup.
Change/Select colour.
Unpattern.
Create a row/column pattern.
Demonstrate a pattern.

Fig. 2. The interface of the current version of the eXpresser annotated with descriptions of the available tools.

3.1 Basic Patterns

The user is able to create a 'unit pattern' which *looks* like a single block. However, inspecting its properties reveals that it is in fact something more complex as illustrated in Figure 3a. The property box in this figure shows three numeric attributes, each of which has an icon. The cogs icon specifies the *element count* of this pattern. The cogs icon with the right arrow specifies how far to the right each shape should be from its predecessor and, similarly, the cogs icon with the arrow downwards specifies how far down a shape should be from its predecessor. Initially, the element count of a pattern is 1 and the other attributes 0 but by setting these values appropriately, various basic patterns can be obtained. Some of these are shown in Figure 3b–e.

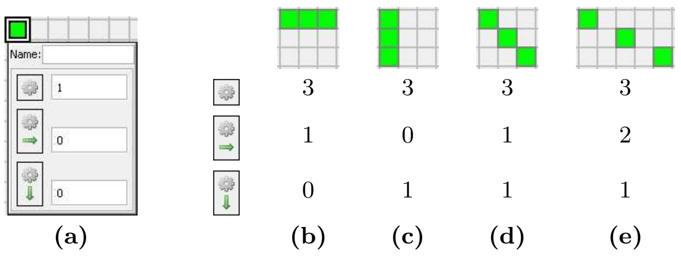

Fig. 3. Basic patterns. **(a)** A ‘single block’. Editing its attributes can result in patterns such as **(b)** a row; **(c)** a column; **(d)** a diagonal; or **(e)** an ‘expanded’ diagonal.

3.2 Using the Pattern Tools

Although interesting patterns can be created using such a basic mechanism, the underlying engine is far more powerful since a pattern can be based on *any* shape whether it is a pattern itself or a group. 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. For example, given two shapes with attributes as shown in Figure 4a and Figure 4b, specifying 4 to the demonstrator tool (Figure 4c) results in a triangle pattern (Figure 4d).[4]

3.3 Creating Dependencies

With the facilities described so far, the user is able to build complex patterns with ease. However, there is also a need to make patterns depend on one another. For example, consider the case where the user is constructing the footpath pattern. If they decide to work row by row, they may first create a pattern of blue squares as shown in Figure 5a. The number of green squares needed on that row is one more than the number of blue squares. By naming the first pattern, this relationship can be specified iconically as shown in Figure 5b. The other rows of the pattern depend in a similar way on the number of blue squares as shown in Figure 5c.

[4] Note that the two shapes used in this example also implicitly specify a movement downwards from one element to the next. At present, this is not shown explicitly in the attribute list so as to prevent user confusion.

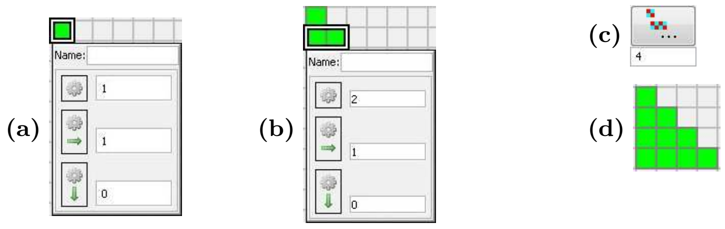

Fig. 4. The demonstrator tool. **(a)** The first element in the pattern; **(b)** the second element; **(c)** The number of elements to be in inferred pattern; **(d)** The resulting pattern with 6 elements.

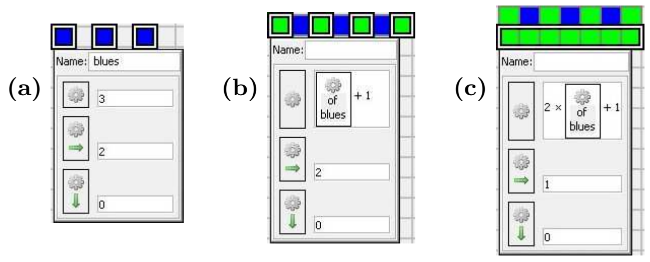

Fig. 5. Constructing the footpath pattern using dependencies.

3.4 Messing-up

Being able to relate attributes of shapes in this way is fundamental to the construction process and through it, the power of generalisation becomes apparent to the user.[5] Figure 6a shows how the pattern (as constructed in Figure 5) changes when the number of blue squares is changed to 5. As expected, the layout of the green squares changes appropriately, thus demonstrating the generality of the construction. In contrast, the construction in Figure 6b does not look correct since the number of squares in its top and bottom rows were specified in absolute terms when there were originally 3 blue squares. In the case of Figure 6c, no aspects of the construction are general; all parts of the pattern were created specific to when there were 3 blue squares.

For constructions that are not entirely general, changing the number of blues in this example 'breaks' the pattern. This potential for 'messing up' [13] by changing various parameters of the problem provides a simple yet powerful mechanism for the student to judge whether a pattern is general or not. It also provides a pedagogic strategy for the teacher (or potentially the system) to challenge students to construct a solution that is impervious to 'messing up' in this way. This 'incentive to generalise' [14] provides students with the opportunity to realise

[5] In some of our earlier work [12], this was referred to as 'building with n'.

that there is an advantage to thinking in terms of abstract characteristics of the task rather than specific numbers.

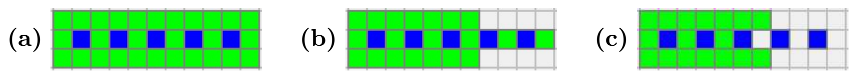

Fig. 6. Messing-up. **(a)** A general construction; **(b)** A partially-general construction; **(c)** An entirely non-general construction.

3.5 Allocations

The features above provide the user with mechanisms for constructing general patterns but no explicit way to analyse them so as to obtain general rules such as the number of greens needed to surround the blues. This issue is addressed by allowing the user to specify the number of squares of each colour that they need for their construction. As a concrete illustration of this feature, Figure 7 shows the step-by-step construction of the footpath interleaved with updating such colour allocations. In this example, the user starts off by considering the instance of the footpath pattern in which there are 3 blue squares. Since they know that (currently) they need that many blue squares, they specify this explicitly (Figure 7a). At this stage, since the number of green squares needed is unknown, they leave this at its initial value of 0 (Figure 7b). Figure 7c shows their first step in the construction process in which they create the pattern of blue squares.[6]

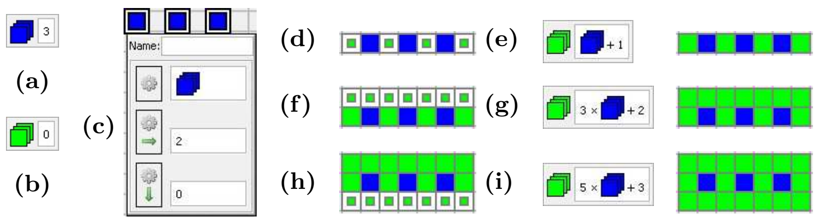

Fig. 7. Using allocations for the footpath pattern. **(a)** Initial blue allocation; **(b)** Initial green allocation; **(c)** Using the blue allocation for pattern construction; **(d)** Adding the enclosing green squares pattern; **(e)** Increasing the allocation; **(f)** Adding the top row pattern; **(g)** Increasing the allocation to cover the top row; **(h)** Adding the bottom row; **(i)** Increasing the allocation to cover the bottom row.

The user next adds the surrounding green pattern for the middle row. Since the allocation for green is currently zero, these squares appear differently as

[6] Note that the specification of the element count here is in terms of the blue allocation. This differs from Figure 5a in which the element count was specified explicitly.

shown in Figure 7d. The user can correct this by specifying the number of green squares that are now needed[7] (Figure 7e). The remainder of the example continues similarly for the top row (Figure 7f-g) and the bottom row (Figure 7h-i). In this way, the user iteratively refines the expression for the green allocation, interleaving construction and analysis. Ultimately, the blue allocation now 'drives' the pattern in that changing its value results in a different instance of the footpath. Moreover, the green allocation *is* now the general rule for the number of green squares needed to surround the blue squares.

This example shows a very rigorously interleaved process of construction and analysis. The advantage here is that through this interleaving, the user may gain a deeper understanding of their construction. However, this comes at a cost in terms of requiring quite complex interaction with the task on two different levels. In view of this, the system is being designed so that if the user, teacher or task designer so desires, construction can proceed without a concern for allocations. Then, once comfortable with how to create the general pattern, the user can, in a subsequent phase, re-build their construction paying attention to the allocations needed. So, rather than interleaving construction and analysis, construction happens first followed by interleaved *re*construction and analysis.

In summary, just as 'messing up' provides the user with an incentive to generalise, allocations provide the user with an incentive to *analyse*.

3.6 Multiple Solutions

One of the design criteria for the eXpresser and one closely related to the constructionist approach we have adopted is that the microworld should be expressive enough so that any given pattern could be constructed in a variety of ways. Such scope for alternatives allows the learner to realize how multiple valid solutions to a problem lead to different — but equivalent — expressions. There is then an incentive to develop intuitively some of the basic rules of algebra such as commutativity and associativity as well as provocation to collaborate with other children who have found correct but different solutions. Figure 8 shows two further constructions of the footpath.

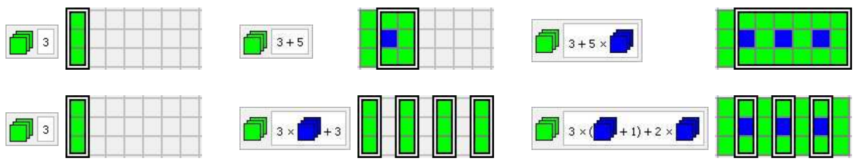

Fig. 8. Different constructions of the footpath pattern. Each row shows a different construction step by step with the most recently added shape selected.

[7] We are in the process of elaborating this feature so that excessive allocations are indicated to the user appropriately.

4 Discussion

As Section 3 explains, the system affords a variety of ways for users to interact with tasks. These interactions between the system and the students are shown diagrammatically in Figure 9 by the edges annotated with ① and ②. Given the inherently open and exploratory nature of these interactions, integrating intelligent support within this process is a significant challenge (see [15]) but one that holds great potential. Possibilities that we are actively exploring include highlighting inconsistencies, automatically prompting the user to simplify their constructions in various ways and providing them with alternative representations so as to assist their understanding of the tasks.

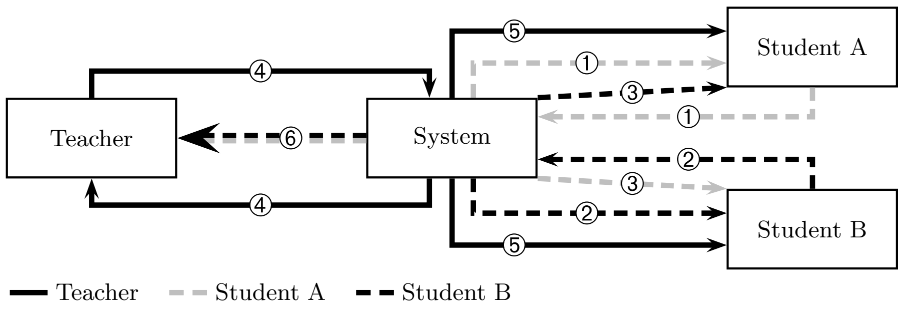

Fig. 9. Information flow between the system, the teacher and the students.

Students will be able to collaborate both during and after the construction and analysis processes (③) and, through intelligent modelling of the collaborative behaviour of the students, the system will be able to recommend effective pairings/groupings of students. Furthermore, there is also significant scope within the system for the effective use of dynamic variation of the collaborative affordances of the interface [16], allowing users to co-construct their patterns while reducing dominance and encouraging high quality collaboration.

The teacher will interact with the system (④) in order to create, design and deploy tasks and task sequences. They will also be able to affect the experiences of the students directly (⑤) by recommending next steps, intervening or allocating further tasks. Most importantly, it will also be possible for the teacher to obtain summary information across all their students (⑥). At its simplest, this can provide the teacher with an overall impression of class progress and, with the use of appropriate representations, can facilitate attending first to those who are most in need of help. Both these services can be greatly enhanced by the appropriate integration of intelligent support. For example, a summary of class progress could take into consideration all previous user experience and provide a much more accurate view. Moreover, with such a facility in place, it is possible to avoid situations in which students misunderstand certain concepts and leave a lesson with a false sense of achievement. This, in combination with other aspects of intelligent support, balances freedom and feedback, allowing students to play, explore, make mistakes, see the immediate outcome of their actions and learn.

References

1. Geraniou, E., Mavrikis, M., Hoyles, C., Noss, R.: Towards a constructionist approach to mathematical generalisation. Proceedings of the British Society for Research into Learning Mathematics (in press)
2. Noss, R., Hoyles, C.: Windows on mathematical meanings: Learning cultures and computers. Dordrecht: Kluwer (1996)
3. Balacheff, N., Kaput, J.: Computer-based learning environments in mathematics. In Bishop, A.J., Clements, K., Keitel, C., Kilpatrick, J., Laborde, C., eds.: International Handbook on Mathematics Education, Dordrecht: Kluwer (1996)
4. Hoyles, C.: Exploratory software, exploratory cultures. In diSessa, A., Hoyles, C., Noss, R., eds.: Computers for Exploratory Learning. Berlin: Springer-Verlag (1995) 199–219
5. Roschelle, J., Kaput, J.: SimCalc MathWorlds: Composable components for calculus learning. Communications of the ACM **39** (1996) 97–99
6. Hoyles, C., Healy, L.: Visual and symbolic reasoning in mathematics: Making connections with computers. Mathematical Thinking and Learning **1**(1) (1999) 59–84
7. Thompson, P.W.: Mathematical microworlds and intelligent computer-assisted instruction. In: Artificial intelligence and instruction: Applications and methods, Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc. (1987) 83–109
8. Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. Proceedings of the 7th World Conference on AIED (1995)
9. Bunt, A., Conati, C.: Assessing effective exploration in open learning environments using bayesian networks. In: Intelligent Tutoring Systems, Springer (2002) 698–707
10. Stathacopoulou, R., Magoulas, G., Grigoriadou, M., Samarakou, M.: Neuro-fuzzy knowledge processing in intelligent learning environments for improved student diagnosis. Information Sciences **170** (2005) 273–307
11. Dimitracopoulou, A.: Design principles for the support of modelling and collaboration in a technology-based learning environment. International Journal of Continuous Engineering Education and Lifelong Learning **15** (2005)
12. Pearce, D., Mavrikis, M., Geraniou, E., Gutierrez, S.: Issues in the design of an environment to support the learning of mathematical generalisation. In: European Conference on Technology Enhanced Learning (EC-TEL08). (2008)
13. Healy, L., Hoelzl, R., Hoyles, C., Noss, R.: Messing up. Micromath **10** (1994)
14. Mason, J., Graham, A., Johnston-Wilder, S.: Developing Thinking in Algebra. Paul Chapman Publishing (2005)
15. Cocea, M., Magoulas, G., Gutierrez, S.: The challenge of intelligent support in exploratory learning environments: A study of the scenarios. In: Proceedings of the 1st International Workshop in Intelligent Support for Exploratory Environments on European Conference on Technology Enhanced Learning (EC-TEL08). (2008)
16. Pearce, D., Kerawalla, L., Luckin, R., Yuill, N., Harris, A.: The Task Sharing Framework: A generic approach to scaffolding collaboration and meta-collaboration in educational software. In: Proceedings of the 13th International Conference on Computers in Education (Singapore). IOS Press (2005) 338–345