Case-based Reasoning Approach to Adaptive Modelling in Exploratory Learning

Mihaela Cocea[1,2], Sergio Gutierrez-Santos[2] and George D. Magoulas[2]

[1] School of Computing, University of Portsmouth,
Buckingham Building, Lion Terrace, Portsmouth, Hampshire, PO1 3HE, UK
mihaela.cocea@port.ac.uk
[2] London Knowledge Lab, Birkbeck College, University of London,
23-29 Emerald Street, London, WC1N 3QS, UK
{sergut, gmagoulas}@dcs.bbk.ac.uk

Abstract. Exploratory Learning Environments allow learners to use different strategies for solving the same problem. However, not all possible strategies are known in advance to the designer or teacher and, even if they were, considerable time and effort would be required to introduce them in the knowledge base. We have previously proposed a learner modelling mechanism inspired from Case-based Reasoning to diagnose the learners when constructing or exploring models. This mechanism models the learners' behaviour through simple and composite cases, where a composite case is a sequence of simple cases and is referred to as a strategy. This chapter presents research that enhances the modelling approach with an adaptive mechanism that enriches the knowledge base as new relevant information is encountered. The adaptive mechanism identifies and stores two types of cases: (a) inefficient simple cases, i.e. cases that make the process of generalisation more difficult for the learners, and (b) new valid composite cases or strategies.

Key words: user modelling, knowledge base adaptation, exploratory learning environments, case-based reasoning

1 Introduction

Exploratory learning environments (ELEs) are built upon a constructionist pedagogical approach [1], which is characterised by two core ideas: (a) learning is seen as a reconstruction of knowledge rather than as a transmission of knowledge and (b) learning is most effective when it is part of an activity in which learners feel they are constructing a meaningful product [1]. The constructionist approach is inspired by Piaget's constructivist theory [2] which states that learners construct mental models to understand the world around them. Consequently, based on these principles, exploratory learning environments allow learners a high degree of freedom and encourage learners to explore and experiment with different models within the particular learning system. Therefore, these environments are radically different from Intelligent Tutoring Systems in

which the learning activities are highly structured and the learner is guided in a stepwise manner.

Exploratory learning environments provide activities that involve constructing [2] and/or exploring models, varying their parameters and observing the effects of these variations on the models' behaviour. When provided with guidance and support ELEs have a positive impact on learning compared with other more structured environments [3]; however, the lack of support may actually hinder learning [4]. Therefore, to make ELEs more effective, intelligent support is needed, despite the difficulties arising from their open nature.

To provide intelligent support, a mechanism for diagnosing the learner is needed, which in Intelligent Learning Environments is done through user/learner modelling. The typical approach is based on concepts of the domain: learners are required to study materials about a concept and then their knowledge level is assessed through testing. In ELEs the emphasis is on the process of learning by means of constructionist activities rather than on the knowledge. Therefore, the focus is on the actions the learners perform in the educational system than on answers to tests, and, consequently, the learner modelling process should focus on analysing the learners' interactions with the system.

To address this, we have proposed a learner modelling mechanism for monitoring learners' actions when constructing/exploring models by modelling sequences of actions that reflect different strategies in solving a task [5]. An important problem, however, remains: only a limited number of strategies are known in advance and can be introduced by the designer/teacher. In addition, even if all strategies would be known, introducing them in the knowledge base of a system would take considerable time and effort. Moreover, the knowledge about a task evolves over time - students may discover different ways of approaching the same task, rendering the knowledge base suboptimal for generating proper feedback, even if initially it had a good coverage. To address this issue, we employ a mechanism for adapting the knowledge base in the context of *eXpresser* [6], an exploratory learning environment for mathematical generalisation.

The knowledge base adaptation involves a mechanism for acquiring inefficient simple cases, i.e. cases which include actions that make it difficult for students to create a generalisable model, and a mechanism for acquiring new strategies. The former could be potentially useful to enable targeted feedback about the inefficiency of certain parts of a construction, or certain actions of the student; this approach could also lead gradually to creating a library of inefficient constructions produced by students that could be analysed further by a researcher/teacher. Without the latter a new valid strategy will not be recognised as such, and, consequently, the learner modelling module will diagnose the learner to be still far from a valid solution and any potential feedback will be confusing as it will guide the learner towards the most similar strategy stored in the knowledge base.

The rest of the chapter is structured as follows. The next section briefly introduces *eXpresser* and the problem of mathematical generalisation. Section 3 describes the case-based reasoning cycle for *eXpresser* and gives a brief overview

of the knowledge representation and the identification mechanism employed. Section 4 presents our proposed approach for adapting the knowledge base. Section 5 describes the validation of this approach and, finally, Section 6 concludes the chapter and presents some directions for future work.

2 Mathematical Generalisation with *eXpresser*

Mathematical generalisation has been defined or described in several ways, varying from philosophical views that could be applied to any type of generalisation to views very specific to mathematics. Examples from the first category are: (a) "an object and a means of thinking and communicating" [7](p. 63), and (b) "applying an argument in a broader context" [8](p. 38). An example from the second category is: "Generalizing problems, also known as numeric sequences or geometric growing sequences, present patterns of growth in different contexts. Students are asked to find the underlying structure and express it as an explicit function or 'rule'." [9](p. 442).

Mathematical generalisation is at the centre of algebraic expressions, as "algebra is, in one sense, the language of generalisation of quantity. It provides experience of, and a language for, expressing generality, manipulating generality, and reasoning about generality" [10](p. 105). This relation, however, together with the idea of recognising and analysing patterns and articulating structure, seems to be elusive to students who fail to understand algebra and its purpose [11]. Students are unable to express a general pattern or relationship in natural language or in algebraic form [12].

Students, however, are able to identify and predict patterns [10] and there are claims that it is not the generalisation problems that are causing difficulties to students, but the way these are presented and the limitations of the teaching approaches used [9]. Typically, "generalising problems are usually presented as numeric or geometric sequences, and typically ask students to predict the number of elements in any position in the sequence and to articulate that as a rule" [9](p. 443). A common strategy is "the construction of a table of values from which a closed-form formula is extracted and checked with one or two examples" [13](p. 7), introducing a tendency towards pattern spotting and emphasising its numerical aspect [14], [15]. This approach obscures the variables involved, "which severely limits students ability to conceptualise the functional relationship between variables, explain and justify the rules that they find, and use the rules in a meaningful way for problem solving" [9](p. 444).

Another approach that affects students' understanding of generalisation is the focus on mathematical products rather than mathematical processes [16], [17]. Malara and Navarra[17] argue that students should be taught to distance themselves from the result and the operations needed to obtain that result, and to reach a higher level of thinking by focusing on the structure of a problem.

Another difficulty encountered in teaching mathematical generalisation is the students' difficulty to use letters that stand for the unknown [18] and to realise that letters represent values [19]. Secondary school students also tend to

lack a mathematical vocabulary for expressing generality [11] and their written responses lack precision [16].

Taking this aspects into account, a system called *eXpresser* [6] was developed using an iterative process that involved designing with students and teachers. The main aim was to develop an environment that provides the students with the means for expressing generality rather than considering special cases or spotting patterns.

eXpresser enables constructing patterns, creating dependencies between them, naming properties of patterns and creating algebraic-like rules with either names or numbers. It is designed for classroom use and targets 11 to 14 years old pupils. Each task involves two main phases: building a construction and deriving an algebraic-like rule from it.

Fig. 1 illustrates the system, the *properties list* of a pattern (linked to another one) and an example of a rule. The screenshot on the left includes two windows: (a) the students' world, where the students build their constructions and (b) the general world that displays the same construction with a different value for the variable(s) involved in the task, and where students can check the generality of their construction by animating their pattern (using the Play button).

We illustrate here a task called 'stepping stones' (see Fig. 1) displayed in the students' world with a number of 3 red (lighter colour) tiles and in the general world with a number of 8 red tiles; the task requires to build such a construction and to find a general rule for the number of blue (darker colour) tiles needed to surround the red ones. The construction for this task can be built in several ways that we call *strategies*. Here we illustrate the 'C strategy', named after the shape

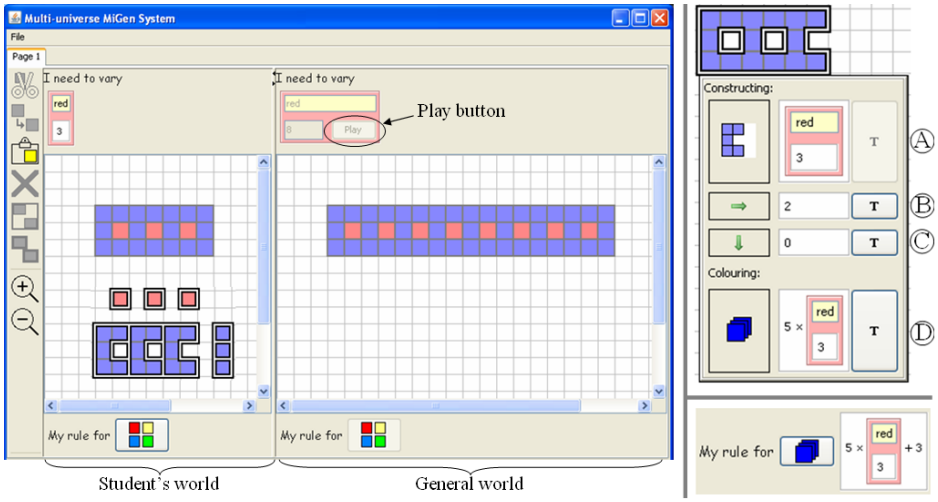

Fig. 1. *eXpresser* screenshots. The screenshot on the left includes a toolbar, the students' world and the general world. The screenshot on the top right shows the property list of a pattern. The bottom right screenshot illustrates a rule.

of the building-block, i.e. the basic unit of a pattern. The components of this strategy are displayed separately in the students' world for ease of visualisation: a red pattern, having 3 tiles, a blue one made of a C-shape pattern repeated 3 times, and 3 blue tiles.

The property list of the C-shape pattern is displayed in the screenshot on the top right. The first property (Ⓐ) specifies the number of iterations of the building-block; the value for this attribute is set to the value of the iterations of the red pattern by using a T-box (that includes a name and a value); by using a T-box, the two (or more) properties are made dependent, i.e. when the value in the T-box changes in one property, it also changes in the other one(s). The next properties are *move-right* (Ⓑ), which is set to 2, and *move-down* (Ⓒ), which is set to 0. The last property (Ⓓ) establishes the number needed to colour all the tiles in the pattern - in our case 5 times the iterations of the red pattern. The bottom right screenshot displays the rule for the number of blue tiles: $5 \times red + 3$, where red stands for the T-box displayed in Ⓐ (a T-box can be displayed with name only, value only or both).

The construction in Fig. 1 and the rule in the bottom-right corner constitute one possible solution for the 'stepping stones' task. Although in its simplest form the rule is unique, there are several ways to build the construction and infer a rule from its components. Thus, there is no unique solution and students follow various kinds of strategies to construct their models (i.e. construction and rule). Two examples of such different constructions and rules are illustrated in Fig. 2. The following section presents our approach for modelling and identification of strategies.

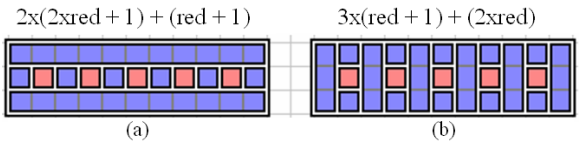

Fig. 2. (a) 'HParallel' Strategy; (b) 'VParallel' Strategy.

3 Modelling Learners' Strategies Using Case-based Reasoning

In case-based reasoning (CBR) [20] knowledge is stored as cases, typically including the description of a problem and its solution. When a new problem is encountered, similar cases are retrieved and the solution is used or adapted from one or more of the most similar cases. The CBR cycle typically includes four processes [20]: (a) *Retrieve* cases that are similar to the current problem; (b) *Reuse* the cases (and adapt) them in order to solve the current problem; (c) *Revise* the proposed solution if necessary; (d) *Retain* the new solution as part of a new case (see Fig. 3).

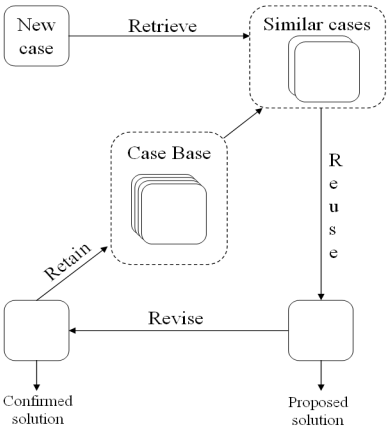

Fig. 3. CBR cycle.

In exploratory learning the same problem has multiple solutions and it is important to identify which one is used by the learner or whether the learner has produced a new valid solution. To address this for *eXpresser* each task has a case-base (or knowledge base) of solutions (i.e. strategies). When a learner is building a construction, it is transformed into a sequence of simple cases (i.e. strategy) and compared with all the strategies in the case-base for the particular task that the learner is working on; the case-base consists of strategies, i.e. composite cases, rather than simple cases. To *retrieve* the strategies that are most similar to the one used by the learner, appropriate similarity metrics are employed (see below). Once the most similar strategies are identified, they are used in a scaffolding mechanism that implements a form of *reuse* by taking this information into account along with other information, such as the characteristics of the learner (e.g. knowledge level, spatial ability), completeness of solution and state within a task. The *reuse*, *revise* and *retain* steps are part of the knowledge base adaptation described in Section 4: simple cases are modified and then stored in a set of inefficient cases; new strategies are stored without modifications.

We use the term knowledge base adaptation in the sense that the knowledge base changes over time to adapt to new ways in which learners approach tasks - ways that could be either efficient or inefficient. This is referred to as 'adaptation to a changing environment' [21]. It is not, however, the same as adaptation in the CBR sense, although this is present to a certain degree in the acquisition of inefficient cases, as it involves the processes of *reuse* and *revise* which are generally referred to as case adaptation [22]. The acquisition of new strategies corresponds to case-base maintenance in CBR terminology [20], as it involves adding a new case for which no similar case has been found.

The following paragraphs briefly present the knowledge representation and the similarity metrics used for strategy identification.

3.1 Knowledge Representation

In our approach, strategies for building a construction are represented as a series of simple cases with certain relations between them. A simple case is defined as $C_i = \{F_i, RA_i, RC_i\}$, where C_i represents the case and F_i is a set of attributes. RA_i is a set of relations between attributes and RC_i is a set of relations between C_i and other cases respectively.

The set of attributes of a given case C_i is defined as $F_i = \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_N}\}$, where N represents the number of attributes. It includes three types of attributes: (a) variables (the first v attributes), (b) numeric (attributes from $v+1$ to w) and (c) binary (attributes from $w+1$ to N). The numeric attributes correspond to the values in the property list and the variables correspond to the type of those properties: number, T-box, expression with number(s) or expression with T-box(es). The binary attributes refer to the membership of a case to a strategy and is defined as a $PartOfS$ function which returns 1 if the case belongs to the strategy and 0 if it does not. There are S binary attributes, where S is the number of strategies in the knowledge base.

The set of relations between attributes of a given case C_i and attributes of other cases (as well as attributes of C_i) is represented as $RA_i = \{RA_{i_1}, RA_{i_2}, \ldots, RA_{i_M}\}$, where M represents the number of relations between attributes and at least one of the attributes in each relation $RA_{i_m}, \forall m = \overline{1, M}$, is from F_i, the set of attributes of C_i. Two types of binary relations are used: (a) *dependency relations* such as the one illustrated in Fig. 1 where the number of the iterations of the blue pattern depends on the iterations of the red pattern through the use of a T-box; these relations are formally represented as $\alpha_{i_k} = DEP(\alpha_{j_l})$, where $\alpha_{i_k}$and α_{j_l} are variables of cases i and j and means that α_{i_k} depends on α_{j_l}; (b) *value relations* such as the fact that the value of the colouring property of the blue pattern in Fig. 1 is 5 times the value of the iterations of the red pattern. A case is considered *specific* when it does not have dependency relations and is considered *general* when it has all the dependency relations required by the task.

The set of relations between cases is represented as $RC_i = \{RC_{i_1}, RC_{i_2}, \ldots, RC_{i_P}\}$, where P represents the number of relations between cases and one of the cases in each relation $RC_{i_j}, \forall j = \overline{1, P}$ is the current case (C_i). Two time-relations are used: (a) $Prev$ relation indicates the previous case and (b) $Next$ relation indicates the next case, with respect to the current case. Each case includes at most one of each of these two relations.

A strategy is defined as $S_u = \{N_u(C), N_u(RA), N_u(RC)\}$, $u = \overline{1, S}$, where S represents the number of strategies in the knowledge base, $N_u(C)$ is a set of cases, $N_u(RA)$ is a set of relation between attributes of cases and $N_u(RC)$ is a set of relations between cases.

To illustrate how a learner's construction is transformed into the knowledge representation detailed above, we use the 'stepping stones' task introduced in Section 2, which requires to find the number of tiles that surround a pattern like the red one displayed in Fig. 1. There are several strategies for constructing the

surrounding for that pattern as illustrated in Fig. 1 (the 'C strategy') and Fig. 2 ('HParallel' and 'VParallel' strategies).

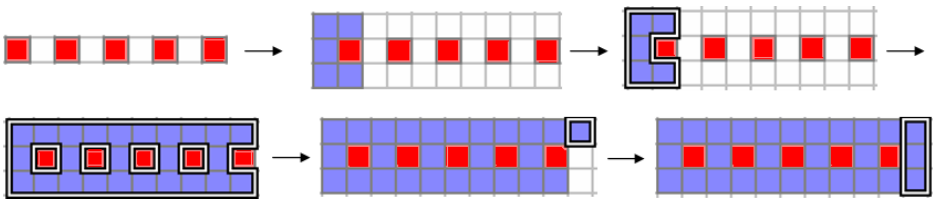

Fig. 4. Possible steps for 'C strategy'.

Besides multiple possible constructions, there are several ways of reaching the same construction. A possible trajectory for the 'C strategy' is illustrated in Fig. 4. The learner may start with the footpath (the red tiles) and then build a group of five blue tiles around the leftmost red tile having the form of a 'C'. Next, the group is iterated five times (the number of red tiles) and, finally, a vertical pattern of three tiles is added at the right of the footpath. The details for most steps of this particular strategy are displayed in Table 1. This table includes a list a patterns, the relations between attributes and the relations between cases.

The first step includes only one case: the red tiles pattern. After some intermediate steps, not illustrated here, the second step includes 6 cases, i.e. the red pattern and five single blue tiles, which are in a given order as expressed by the set of $Prev$ and $Next$ relations. In the third step, the 5 blue tiles are grouped in one pattern which now becomes C_2; consequently, at this point there are 2 successive cases. In the fourth step, the second case, i.e the group of 5 blue tiles, is repeated 5 times (the number of red tiles), so now there is also a value and a dependency relation. In the fifth step a new blue tile is added, becoming C_3 and

Table 1. S_u definition for each step of the 'C strategy'.

S_u	$N_u\,(C)$	$N_u\,(RA)$	$N_u\,(RC)$
Step 1	C_1	-	-
Step 2	$C_1, C_2, C_3,$	-	$Prev(C_{i+1}) = C_i$ for $i = \overline{1,5}$
	C_4, C_5, C_6		$Next(C_i) = C_{i+1}$ for $i = \overline{1,5}$
Step 3	C_1, C_2	-	$Next(C_1) = C_2$
			$Prev(C_2) = C_1$
Step 4	C_1, C_2	$\alpha_{2_3} = \alpha_{1_3}$	$Next(C_1) = C_2$
		$\alpha_{2_3} = DEP(\alpha_{1_3})$	$Prev(C_2) = C_1$
Step 5	C_1, C_2, C_3	$\alpha_{2_3} = \alpha_{1_3}$	$Next(C_i) = C_{i+1}$ for $i = \overline{1,2}$
		$\alpha_{2_3} = DEP(\alpha_{1_3})$	$Prev(C_{i+1}) = C_i$ for $i = \overline{1,2}$
Step 6	C_1, C_2, C_3	$\alpha_{2_3} = \alpha_{1_3}$	$Next(C_i) = C_{i+1}$ for $i = \overline{1,2}$
		$\alpha_{2_3} = DEP(\alpha_{1_3})$	$Prev(C_{i+1}) = C_i$ for $i = \overline{1,2}$

in the sixth step this tile is iterated 3 times; in the last two steps, the relations between attributes and between cases are the same as in step 4.

The attributes for each pattern were not included in Table 1 as the focus is on the representation of the strategy and a list of attributes for each pattern in every step would hinder the understanding of the high level representation. The difference between Step 5 and Step 6, however, is not clear without knowing that the difference lies in the attribute values, i.e. for C_3, the iterations attribute has changed from 1 to 3 and the move down attribute has changed from 0 to 1, which is not shown in Table 1.

3.2 Similarity Metrics

Strategy identification is based on scoring elements of the strategy followed by the learner according to the similarity of their attributes and their relations to strategies previously stored. Thus, to identify components of a strategy, four similarity measures are defined:

(a) *Numeric attributes* - Euclidean distance: $D_{IR} = \sqrt{\sum_{j=v+1}^{N}(\alpha_{I_j} - \alpha_{R_j})^2}$ (I and R stand for input and retrieved cases, respectively); attributes from $v+1$ to N are used, i.e. the numeric and binary attributes described in the previous section.

(b) *Variables*: $V_{IR} = \sum_{j=1}^{v} g(\alpha_{I_j}, \alpha_{R_j})/v$ (attributes from 1 to v are variables), where g is defined as: $g(\alpha_{I_j}, \alpha_{R_j}) = 1$ if $\alpha_{I_j} = \alpha_{R_j}$ and $g(\alpha_{I_j}, \alpha_{R_j}) = 0$ if $\alpha_{I_j} \neq \alpha_{R_j}$.

(c) *Relations between attributes* - Jaccard's coefficient: $A_{IR} = \frac{|RA_I \cap RA_R|}{|RA_I \cup RA_R|}$. A_{IR} is the number of relations between attributes that the input and retrieved case have in common divided by the total number of relations between attributes of the two cases;

(d) *Relations between cases* - Jaccard's coefficient: $B_{IR} = \frac{|RC_I \cap RC_R|}{|RC_I \cup RC_R|}$, where B_{IR} is the number of relations between cases that the input and retrieved case have in common divided by the the total number of relations between cases of I and R.

To identify the closest strategy to the one followed by a learner during construction, cumulative similarity measures are used for each of the four similarity types:

(a) Numeric attributes - as this metric has a reversed meaning compared to the other ones, i.e. a smaller number means a greater similarity, the following function is used to bring it to the same meaning as the other three similarity measures, i.e. a greater number means greater similarity:

$$F_1 = \begin{cases} \frac{z}{\sum_{i=1}^{z} D_{I_i R_i}} & \text{if } \sum_{i=1}^{z} D_{I_i R_i} \neq 0 \\ z & \text{if } \sum_{i=1}^{z} D_{I_i R_i} = 0, \end{cases}$$

(b) Variables: $F_2 = (\sum_{i=1}^{z} V_{I_i R_i})/z$;

(c) Relations between attributes: $F_3 = (\sum_{i=1}^{z} A_{I_i R_i})/y$;
(d) Relations between cases: $F_4 = (\sum_{i=1}^{z} B_{I_i R_i})/z$,

where z represents the minimum number of cases among the two compared strategies and y represents the number of pairs of cases in the retrieved strategy that have relations between attributes; for example, the 'C strategy' has three cases and only one relation between an attribute of case C_1 and an attribute of C_2 (see Table 1); therefore there is only one pair of cases that have a relation between attribute, i.e. $y = 1$.

As the similarity metric for numeric attributes has a different range from the other metrics, normalisation is applied to have a common measurement scale, i.e. $[0, 1]$. This is done using linear scaling to unit range [23] by applying the following function: $\overline{x} = \frac{x-l}{u-l}$, where x is the value to be normalised, l is the lower bound and u is the upper bound for that particular value. The range of the values that can be taken by the similarity metric for the numeric attributes, i.e. F_1, is $[0, z]$. Consequently, to transform the values so that they are within the $[0, 1]$ range, the following normalisation function is applied: $\overline{F_1} = F_1/z$.

Weights are applied to the four similarity metrics to express the central aspect of the construction, the structure. This is mostly reflected by the $\overline{F_1}$ metric and, to a lesser extent, by the F_3 metric. Therefore, we agreed on the following weights: $w_1 = 6, w_2 = 1, w_3 = 2, w_4 = 1$. Consequently, the similarity metric for strategies is: $Sim = 6 * \overline{F_1} + F_2 + 2 * F_3 + F_4$, which can take values in the range of $[0, 10]$.

The metrics have been tested for several situations of pedagogical importance: identifying complete strategies, partial strategies, mixed strategies and non-symmetrical strategies. The similarity metrics were successful in identifying all these situations (details can be found in [5]).

4 Adaptation of the Knowledge Base

Adaptive systems refer to systems that change over time to respond to new situations. There are three levels of adaptation depending on the complexity and difficulty of the adaptation process, with the first level being the least difficult and the third being the most complex and difficult [21]: (a) adaptation to a changing environment; (b) adaptation to a similar setting without explicitly being ported to it; (c) adaptation to a new/unknown application. Our adaptive modelling mechanism involves adaptivity at the first level, meaning that the system adapts itself to a drift in the environment by recognising the changes and reacting accordingly [21].

Before going into the details of our approach, we would like to point out the structure of the knowledge base. As mentioned in Section 3, for each task, there is a corresponding knowledge base which consists of strategies. The strategies are represented as a list of simple cases; each case is represented as a list of attributes, a list of relations between attributes and a list of relations between cases. We are not using indexing as for our purpose the similarity matching is

not computationally expensive; moreover, because there is a separate knowledge base for each task, the size of the knowledge bases is relatively small.

Our proposed approach for adapting the knowledge-base of *eXpresser* includes acquiring inefficient simple cases and acquiring new strategies. Fig. 5 shows some examples from the 'stepping stones' task introduced previously; the constructions in Fig. 5a and 5c have been broken down into the individual components used by the students for ease of visualisation. These examples, with the adaptation rationale and mechanism are discussed below.

4.1 Acquiring inefficient simple cases

The goal of this mechanism is to identify parts of strategies constructed in inefficient ways and store them in a set or library of 'inefficient constructions', i.e. constructions that pose difficulties for the learners in their process of generalisation. The library could be further used for automatic generation of feedback or could be analysed by a researcher or teacher. The results of such an analysis could be then used to design better interventions or make other design decisions for the current system, could be presented as a lesson learned to the scientific community of mathematics teachers and researchers, or even discussed further in class (e.g in the case of an inefficient construction that is frequently chosen by the pupils of that class).

The construction in Fig. 5a illustrates an inefficient pattern within the "HParallel" strategy of the 'stepping stones' task: the middle bar of blue tiles is constructed as a group of two tiles repeated twice - this can be seen in the property list of this pattern displayed in Fig. 5b. The efficient way to construct this component is one tile repeated four times or, to make it general, one tile repeated the number of red tiles plus one. The efficient and the inefficient way of constructing the middle row of blue tiles lead to the same visual output, i.e. there is no difference in the appearance of the construction, making the situation even more confusing. The difficulty lies in relating the values used in the construction of the middle row of blue tiles (C_i) to the ones used in the middle row of red tiles (C_j). If the learner would relate the value 2 of iterations of C_i to the value 3 of iterations of C_j, i.e. the value 2 is obtained by using the number of red tiles

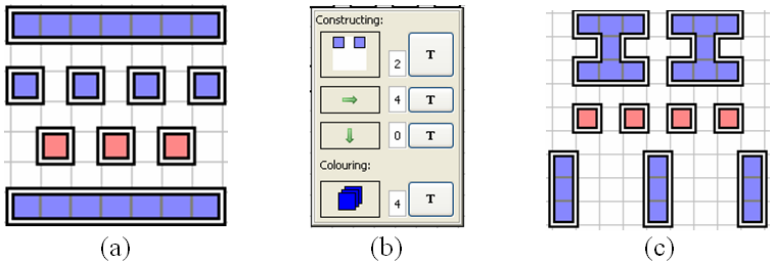

Fig. 5. (a) HParallel strategy with one inefficient component (blue middle row) ; (b) property list of the inefficient component; (c) a new strategy

(3) minus 1, this would work only for a 'stepping stones' task defined for 3 red tiles. In other words, this will not lead to a general model.

Algorithms 1, 2 and 3 illustrate how inefficient simple cases are identified and stored. First, the most similar strategy is found. If there is no exact match, but the similarity is above a certain threshold θ, the process continues with the identification of the inefficient cases; for each of these cases, several checks are performed (Alg. 2). Upon satisfactory results and if the cases are not already in the set of inefficient cases, they are then stored (Alg. 3).

Algorithm 1 Verification(*StrategiesCaseBase*, *InputStrategy*)

```
Find most similar strategy to InputStrategy from StrategiesCaseBase
StoredStrategy ← most similar strategy;
if similarity > θ then
  Find cases of InputStrategy that are not an exact match to any case of
  StoredStrategy
  for each case that is not an exact match do
    InputCase ← the case that is not an exact match
    Compare InputCase to all cases of the set of inefficient cases;
    if no exact match then
      Find the most similar case to InputCase from the cases of StoredStrategy
      StoredCase ← the most similar case
      if Conditions(StoredCase, InputCase) returns true then // see Alg. 2
        InefficientCaseAcquisition(StoredCase, InputCase) // see Alg. 3
      end if
    end if
  end for
end if
```

Algorithm 2 Conditions($C1$, $C2$)

```
if (MoveRight[C1] ≠ 0 and Iterations[C1] * MoveRight[C1] = Iterations[C2] *
MoveRight[C2]) or
(MoveDown[C1] ≠ 0 and Iterations[C1] * MoveDown[C1] = Iterations[C2] *
MoveDown[C2]) then
  return true
else
  return false
end if
```

What is stored is actually a modification of the most similar (efficient) case, in which only the numerical values of *iterations*, *move-right* and/or *move-down* are updated together with the value and dependency relations. These are the only modifications because, on one hand, they inform the way in which the pattern has been built and its non-generalisable relations, and, on the other hand, it is im-

Algorithm 3 InefficientCaseAcquisition(*StoredCase*, *InputCase*)

```
NewCase ← StoredCase
for i = 4 to v − 1 do // attributes from iterations to move-down
  if value of attribute i of NewCase different from that of InputCase then
    replace value of attribute i of NewCase with the one of InputCase
  end if
end for
for all relations between attributes do // value and dependency relations
  replace relations of NewCase with the ones of InputCase
end for
add NewCase to the set of inefficient cases
```

portant to preserve the values of $PartOfS$ attributes, so the researcher/teacher knows in which strategies these can occur. The colouring attributes and the relation between cases are not important for this purpose and, therefore, they are not modified. This has also the advantage of being computationally cheaper.

4.2 New strategy acquisition

The goal of this mechanism is to identify new strategies and store them for future use. New strategies could be added by the teacher or could be recognised as new from the learners' constructions. In the later case, after the verification checks described below, the decision of storing a new strategy is left with the teacher. This serves as another validation step for the detected new strategy.

Fig. 5c illustrates the so-called "I strategy", as some of its building blocks resemble the letter I. When compared to all stored strategies, this strategy is rightly most similar to the 'VParallel' one (see Fig. 2b), as some parts correspond to it. However, the similarity is low, suggesting it may be a new strategy. Without the adaptation mechanism, the learner modelling module will infer that the learner is using the 'VParallel' strategy, but is still far from having completed it. This imprecise information could be potentially damaging as it could, for example, lead to inappropriate system actions, e.g. providing confusing feedback that would guide the learner towards the 'VParallel' strategy. Conversely, identifying the new construction as a new valid strategy will prevent generating potentially confusing feedback, and storing the new strategy will enable producing appropriate feedback in the future - automatically or with input from the teacher/researcher.

Algorithms 4, 5 and 6 illustrate the process by which an input strategy could be identified and stored as a new strategy (composite case). If the similarity between the input strategy and the most similar strategy from the case-base is below a certain threshold θ_1 (Alg. 4), some validation checks are performed (Alg. 5) and upon satisfaction, the new strategy is stored in the case-base (Alg. 6). If the input strategy has been introduced by a teacher and the similarity is below θ_1, the teacher can still decide to go ahead with storing the new strategy, even if it is very similar to an existing one in the database.

Algorithm 4 NewStrategyVerification(*StrategiesCaseBase*, *InputStrategy*)

```
Find most similar strategy to InputStrategy from the StrategiesCaseBase
if similarity < θ1 then
  if ValidSolution(InputStrategy) returns true then // see Alg. 5
    NewStrategyAcquisition(InputStrategy) // see Alg. 6
  end if
end if
```

Algorithm 5 ValidSolution(*InputStrategy*)

```
if SolutionCheck(InputStrategy) returns true then // checks if InputStrategy
'looks like' a solution
  if the number of cases of InputStrategy < θ2 then
    if InputStrategy has relations between attributes then
      RelationVerification(InputStrategy) // verifies that the numeric relation cor-
      responds to the task rule solution
      if successful verification then
        return true
      end if
    end if
  end if
end if
```

Algorithm 6 NewStrategyAcquisition(*NewStrategy*)

```
add NewStrategy to the strategies case-base
adjust values of PartOfS
```

In Algorithm 5 the SolutionCheck(*InputStrategy*) function verifies whether *InputStrategy* 'looks like' a solution by examining if the mask of *InputStrategy* corresponds to the mask of the task. The following check takes into consideration the number of simple cases in the *InputStrategy*. Good solutions are characterized by a relatively small number of simple cases; therefore, we propose for the value of θ_2 the maximum number of cases among all stored strategies for the corresponding task, plus a margin error (such as 3). If this check is satisfied, the RelationVerification(*InputStrategy*) function derives a rule from the value relations of the cases and checks its correspondence to the rule solution of the task. For example, in the construction of Fig. 5c, the rule derived is $3*(\frac{red}{2}+1)+7*\frac{red}{2}$ which corresponds to the solution $5*red+3$. If all checks are satisfied, the new strategy is stored in the case-base and the $PartOfS$ values are adjusted.

5 Validation

The validation of our proposed adaptive modelling mechanisms includes: (a) identifying the boundaries of how far a pattern can be (inefficiently) modified and still be recognised as similar to its original (efficient form); (b) correct identification of inefficient cases within these boundaries and (c) correct identification

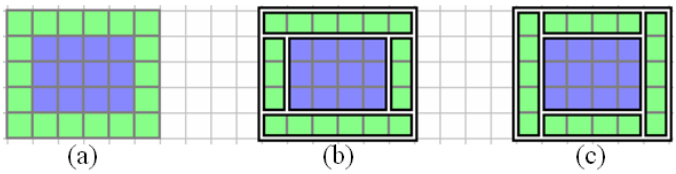

Fig. 6. (a) the construction for the 'pond tiling' task ; (b) 'I Strategy'; (c) 'H strategy'.

of new strategies. This low-level testing of the system shows how the adaptation of the knowledge-base and the learner modelling module function together to improve the performance of the system.

To this end, experiments have been conducted using real data produced from classroom use of *eXpresser* as well as artificial data that simulated situations observed in the classroom sessions. Simulated situations were based on varying parameters of models produced by learners in order to provide more data.

First, a preliminary experiment using classroom data was conducted to identify possible values for the threshold θ in Algorithm 1 and threshold θ_1 in Algorithm 4. Since our main aim was to test the adaptive modelling mechanism we decided not to seek optimal values for these thresholds, but only to find a good enough value for each one. Two possibilities were quickly identified - for θ: the minimum *overall similarity* (4.50) minus an error margin (0.50) or value 1.00 for the *numerical similarity*; for θ_1: the maximum *overall similarity* (3.20) plus an error margin (0.30) or value 1 for the *numeric similarity*.

Experiment 1: identifying the boundaries of how far a pattern can be inefficiently modified and still be recognised as similar to its original efficient form. As mentioned previously, we consider changes in a pattern that can lead to the same visual output as the original one but use different building-blocks. More specifically, these building-blocks are groups of two or more of the original efficient building-block. This experiment looks for the limits of changes that a pattern can undergo without losing its structure so that it can be still considered to be the same pattern.

For this experiment we used 34 artificial inefficient cases from two tasks: (a) 'stepping stones' that was defined earlier and (b) 'pond tiling' which requires to find the number of tiles needed to surround any rectangular pond. Fig. 6 illustrates the construction for the 'pond tiling' problem and two strategies frequently used by students to solve this task. Our adaptive mechanism was build to work for any task in *eXpresser* rather than for particular tasks. For the two tasks we used in our experiments, the tests were conducted using their corresponding user data and their (separate) knowledge bases; the results were collated.

From the 34 cases, 47% were from the 'stepping stones' task and 53% were from the 'pond tiling' task. Using these cases, the following boundaries were identified: (i) groups of less than 4 building-blocks; (ii) groups of 2 building-blocks repeated less than 6 times and (iii) groups of 3 building-blocks iterated less than 4 times.

Experiment 2: correct identification of inefficient cases within the previously identified boundaries. From the total of 34 inefficient cases used in *Experiment 1*, 13 were outside the identified boundaries and 21 were within. From the 21 cases within the boundaries, 62% were from the 'stepping stones' task and 38% were from the 'pond tiling' task.

Using the previously identified values for θ, we obtained the following results: out of these 21 cases, 52.48% had the overall similarity greater than 4.00 and 100% had the numeric similarity above 1.00. These results indicate that a small modification of a pattern can drastically affect the identification of the strategy the learner is following; hence almost half the cases have an overall similarity less than 4.00. The results obtained using the numeric similarity are much better and consistent with the fact the modifications are just numerical.

Experiment 3: correct identification of strategies. The data for this experiment included 10 new strategies: 7 observed in trials with pupils and 3 artificial. Out of the 10 new strategies, 4 were from the 'pond tiling' task; all of them were observed in trials with pupils. The remaining 6 new strategies were from the 'stepping stones' task, with 3 of them observed and 3 artificial. The knowledge base for the two tasks included originally 4 strategies for the 'stepping stones' task and 2 strategies for the 'pond tiling' task.

Using the previously identified values for θ_1, we obtained the following results: out of the 10 new strategies, 100% had the overall similarity below 3.50 and 70% had the numeric similarity below 1.00. As opposed to Experiment 2, the overall similarity performs better, being consistent with the fact that the overall similarity reflects better the resemblance with the stored strategies than the numeric similarity alone. Given the range that the overall similarity has, i.e. 0 to 10, values below 3.50 indicate a very low similarity and therefore, rightly suggest that the learner's construction is considerably different from the ones in the knowledge base.

6 Conclusions

In this chapter we presented research on modelling users' behaviour in *eXpresser*, an exploratory learning environment for mathematical generalisation. We adopted a case-based formulation of strategies learners follow when building constructions in *eXpresser* and employed similarity metrics to identify which strategy is used by each learner. Due to the open nature of the environment, however, not all strategies are known in advance. Moreover, learners use the system in inefficient ways that lead to difficulties in solving the given tasks. To overcome these problems, we developed an adaptive modelling mechanism to expand an initially small knowledge base, by identifying inefficient cases (i.e. cases that pose additional difficulty to the user's learning process) and new strategies.

For both inefficient patterns and new strategies, the principle is the same: they are compared with data from the knowledge base and if they are not already stored, some task-related checks are performed and upon successful verification,

they are added to the knowledge base. With this mechanism, new data can be added to the knowledge base without affecting the recognition of existing data.

To evaluate our proposed adaptive modelling mechanism three experiments were conducted: (a) identifying the boundaries of how far a pattern can be inefficiently modified and still be recognised as similar to its original efficient form; (b) correct identification of inefficient cases within these boundaries and (c) correct identification of new strategies. The evaluation of the proposed approach showed that it is capable of recognising new inefficient patterns within certain boundaries and of recognising new strategies. The boundaries for recognising inefficient patterns are related to the similarity metrics' ability to identify how much they have been modified from their original initial form. When looking at the modifications that learners tend to make, we notice that they take the form of using repetitions of the basic building-block, which modify the structure of the pattern. The similarity metrics, however, were defined to recognise *structural* similarity. Therefore, to improve the metrics' ability to recognise modifications of efficient patterns, they should be enhanced with the capacity to recognise sub-patterns.

Our adaptive modelling mechanism ensures that the learner diagnosis will be accurate even when the researcher or teacher authors only one or two strategies for a new task. Also, it ensures that the learner diagnosis will be accurate when learners' behaviour changes over time even if initially there is a large knowledge base.

The adaptive mechanism that we developed was tailored for *eXpresser* and the domain of mathematical generalisation. We believe, however, that the high level idea can be used in other exploratory learning environments and for domains where several approaches are possible for the same problem.

Future work includes using the information on inefficient cases and new strategies in an automatic way to either incorporate this information in the feedback and/or inform the teachers and allow them to author feedback.

Acknowledgements

This work is partially funded by the ESRC/EPSRC Teaching and Learning Research Programme (Technology Enhanced Learning; Award no: RES-139-25-0381).

References

1. Papert, S.: Mindstorms: children, computers and powerful ideas. BasicBooks, New York (1993).
2. Piaget, J.: The Psychology of Intelligence. New York: Routledge (1950).
3. de Jong, T., van Joolingen, W.: Scientific discovery learning with computer simulations of conceptual domains. Review of Educational Research **68** (1998) 179–202.
4. Kirschner, P., Sweller, J., Clark, R.: Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential and inquiry-based teaching. Educational Psychologist **41**(2) (2006) 75–86.

5. Cocea, M., Magoulas, G.D.: Task-oriented modeling of learner behaviour in exploratory learning for mathematical generalisation. In: In Proceedings of the 2nd ISEE workshop, in conjunction with AIED'09. (2009) 16–24.
6. Pearce, D., Mavrikis, M., Geraniou, E., Gutiérrez, S.: Issues in the design of an environment to support the learning of mathematical generalisation. In Dillenbourg, P., Specht, M., eds.: EC-TEL. Volume 5192 of Lecture Notes in Computer Science., Springer (2008) 326–337.
7. Dorfler, W.: Forms and means of generalisation in mathematics. In Bishop, A., Mellin-Olsen, S., van Dormolen, J., eds.: Mathematical Knowledge: Its Growth through Teaching, Kluwer Academic Publishers, Dordrecht (1991) 63–85.
8. Harel, G., Tall, D.: The general, the abstract and the generic in advanced mathematics. For the Learning of Mathematics **11**(1) (1991) 38–42.
9. Moss, J., Beatty, R.: Knowledge building in mathematics: Supporting collaborative learning in pattern problems. International Journal of Computer-Supported Collaborative Learning **1**(4) (2006) 441–465.
10. Mason, J. L. Haggarty, Aspects of Teaching Secondary Mathematics: Perspectives on Practice. In: Generalisation and algebra: Exploiting children's powers. Routledge Falmer and the Open University (2002) 105–120.
11. Geraniou, E., Mavrikis, M., Hoyles, C., Noss, R.: A constructionist approach to mathematical generalisation. In Joubert, M., ed.: Proceedings of the British Society for Research into Learning Mathematics. Volume 28 (2) of BSRLM Proceedings. (2008).
12. Hoyles, C., Küchemann, D.: Students understanding of logical implication. Educational Studies in Mathematics **51**(3) (2002) 193–223.
13. Bednarz, N., Kieran, C., Lee, L.: Approaches to algebra: Perspectives for research and teaching. In Bednarz, N., Kieran, C., Lee, L., eds.: Approaches to algebra: Perspectives for research and teaching, Kluwer Academic Publishers, Dordrecht (1991) 3–12.
14. Noss, R., Healy, L., Hoyles, C.: The construction of mathematical meanings: Connecting the visual with the symbolic. Educational Studies in Mathematics **33**(2) (1997) 203–233.
15. Noss, R., Hoyles, C.: Windows on Mathematical Meanings: Learning cultures and computers. Kluwer Academic Publishers, Dordrecht (1996).
16. Warren, E., Cooper, T.J.: Generalising the pattern rule for visual growth patterns: actions that support 8 year olds' thinking. Educational Studies in Mathematics **67**(2) (2008) 171–185.
17. Malara, N., Navarra, G.: ArAl Project: Arithmetic pathways towards favouring pre-algebraic thinking. Pitagora Editrice, Bologna (2003).
18. Küchemann, D.: Childrens Understanding of Mathematics :11 - 16. John Murray, London (1991).
19. Duke, R., Graham, A.: Inside the letter. Mathematics Teaching Incorporating Micromath **200** (2007) 42–45.
20. Kolodner, J.L.: Case-Based Reasoning. 2nd edn. Morgan Kaufmann Publishers, Inc. (1993).
21. Anguita, D.: Smart adaptive systems: State of the art and future directions of research. In: Proceedings of the 1st European Symposium on Intelligent Technologies, Hybrid Systems and Smart Adaptive Systems, EUNITE 2001. (2001) 1–4.
22. Mitra, R., Basak, J.: Methods of case adaptation: A survey: Research articles. International Journal of Intelligent Systems **20** (June 2005) 627–645.
23. Aksoy, S., Haralick, R.: Feature normalisation and likelihood-based similarity measures for image retrieval. Pattern Recognition Letters **22**(5) (2001) 563–582.