Sequence Detection for Adaptive Feedback Generation in an Exploratory Environment for Mathematical Generalisation*

Sergio Gutierrez-Santos, Manolis Mavrikis, and George Magoulas

London Knowledge Lab
{sergut,m.mavrikis}@lkl.ac.uk, gmagoulas@dcs.bbk.ac.uk

Abstract. Detecting and automating repetitive patterns in users' actions has several applications. One of them, often overlooked, is supporting learning. This paper presents an approach for detecting repetitive actions in students who are interacting with an exploratory environment for mathematical generalisation. The approach is based on the use of two sliding windows to detect possible regularities, which are filtered at the last stage using task knowledge. The result of this process is used to generate adaptive feedback to students based on their own actions.

1 Introduction

Detecting repetitive actions in users' behaviour with interactive applications has several advantages. First, automating execution of regular tasks can increase users' productivity. This observation led to the creation of script languages for the automatic execution of repetitive tasks in commercial applications. Subsequently macros or programming by demonstration systems [1] emerged, and recent advantages in machine learning allowed the development of user interface learning agents (e.g., [2,3]) that attempt to take away the burden of explicit demonstration and by observing users' actions and identify repetitive patterns and provide suggestions to automate them (e.g., [2]). Also, identifying regularities in users' interactions can help system designers and developers understand better how a system is used, identify unnecessary actions, problems and design flaws [4].

A somehow overlooked area for the identification of regularities in actions is supporting learning in technology-enhanced learning scenarios. In particular, our interest in identifying patterns of actions stems from the requirement to support young learners while interacting with eXpresser, an Exploratory Learning Environment (ELE) for mathematical generalisation, in classroom conditions. This is a domain where repetition and commonality is the basis of hypothesis formation and subsequently generalisation [5].

ELEs are virtual spaces that provide learners with relatively open tasks and therefore have been shown to be useful tools for discovery learning [6,7]. However,

* This work is funded in part from the MiGen Project, grant TLRP RES-139-25-0381. Thanks to the rest of the MiGen team for support and ideas.

as with other constructivist approaches (c.f. [8]), research in the learning sciences suggests that students require explicit support and guidance to learn from their experience [9]. For example, students often pursue unproductive learning paths, or just adopt a playful but unproductive interaction. Our work responds to the need to provide learners with the right information at the right time when interacting with the eXpresser.

In particular, the work presented in this paper focuses on automating the detection of latent repetitive actions while students are constructing models in an ELE. With the ability to detect repetitive actions, the system is able to provide personalised feedback that draws the attention of the learner to what they have been repeating (i.e. making the implicit structure explicit) and supporting them in their efforts to find the general rule.

2 Background

2.1 The eXpresser: An Exploratory Learning Environment

The eXpresser is a microworld (i.e. a special kind of ELE where students can create their own objects and construction to think with) that engages students in generating figural patterns. This is quite central to the British Mathematical curriculum, and is considered one of the major routes to algebra [10,5,11]. For a detailed description of the system, the pedagogical rationale behind its design and its potential in supporting student autonomy in their learning, and the difficulties that students face in mathematical generalisation, the reader is referred to [12]. In relation to the work presented in this paper, one of the objectives behind the design of the microworld is to support students to develop an orientation towards looking for the underlying structure of patterns by identifying their building blocks, a central 'habit of mind' in algebraic thinking [13].

As an example, consider one of the tasks students undertake in the system (see Figure 1). Through exploration of the environment, students are encouraged to 'see' the structure of the pattern in a multitude of different ways and make explicit the rule that pertains their construction using the metaphor of colouring the pattern. In order to do that, students can use building blocks of square tiles to make patterns.

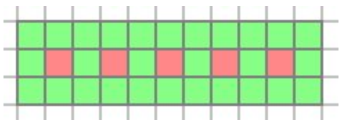


Fig. 1. The footpath task: How many green tiles do you need to surround n red tiles?

2.2 The Problem of Detecting Structure

Many learners in our target age (11–12 years old) find it challenging to express the general case of how many green tiles are needed to surround the red tiles (e.g. $n_g = 5 \cdot n_r + 3$). The crucial determinant of progress in this task is the ability to perceive a structure within the pattern and articulate it to themselves and subsequently to the system.

Although the system provides patterning and grouping facilities so that any structure can be expressed explicitly, many learners fail to be precise about the building block that they perceive in their patterns. They often revert to constructions that are specific to numerical cases. One pedagogic strategy used to support students in this situation capitalises on the 'rhythm' of their action [5], by drawing attention to repeated sequences of actions that highlight some recognition of structure by the student. More often than not, the perceived structure is implicit both to the system and to the learner in that they themselves often do not realise that they are working systematically. Findings from early studies with the system were in line with the literature in mathematics education [14,10,5], suggesting that prompting students to observe the rhythm in their actions has the potential to make the implicit structure evident, to enable them to identify the variants and invariants of a pattern and capture them in their constructions and expressions.

In the classroom, identifying and reinforcing the rhythm of actions is usually part of the teacher's role. However, the overall aim of our project, which is to facilitate integration of the ELE in classroom, introduced the technical challenge of delegating some of the responsibilities of the teacher to an intelligent system. One of the requirements therefore is to identify students' latent structure and provide personalised feedback that draws the learners' attention to what they have been repeating. In other words, by making the implicit structure explicit, the system can support students in their efforts to find the general rule.

3 Detection of Repetitive Actions

This section describes the algorithm designed to detect "rhythm" in the actions of the students, and how it is used to provide feedback to them.

3.1 General Description

Our approach consists of three stages (see Figure 2). First, sequences of events (i.e. in the case of eXpresser, tiles' positions) are analysed using two sliding windows. The distances between the windows are calculated, using a string metric that is appropriate for the problem according to a pedagogical analysis done in advance. These distances are stored in a table. Those windows that exhibit a higher similarity to other subsequent windows are selected as possible student rhythms, i.e. inherent structures in their minds. A final step makes an additional fine-filtering according to specific knowledge about the task to discard patterns that would not be adequate from a pedagogical point of view.

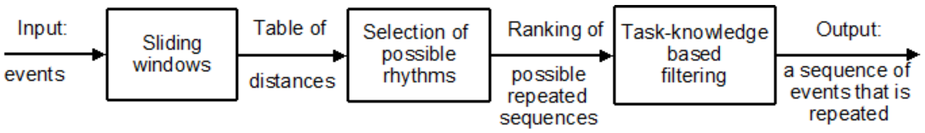

Input:
events
Sliding windows
Table of
distances
Selection of possible rhythms
Ranking of
possible repeated sequences
Task-knowledge based filtering
Output:
a sequence of events that is repeated

Fig. 2. Detection of repeated structures in users' actions in three stages

In the first stage the two sliding windows contain sub-sequences from the total sequence of events. The windows are moved to cover all the events and never overlap as it does not make sense in this case to look for similarities between a sequence of events and a particular copy of itself. For each pair of positions, the distance between both windows is calculated (a specific example is shown later in Table 1). Distance calculation is application-dependent, being determined by the different kinds of "noise" or "errors" that are commonly observed in the sequences of events for a particular application: e.g. when rhythm sequences are expected to differ because of insertion errors, a Levehnstein metric could be used. A survey that studies several distance functions between strings can be found in [15].

The movement of both sliding windows is described using pseudo-code in Figure 3. It should be noted that the window size is not fixed, but varies between two thresholds $T1$ and $T2$. These thresholds are also application-dependent, and must be set beforehand according to the characteristics of the task. It is important to strike a balance between low thresholds, which may result in too many spurious results, and high thresholds, which may result in no findings at all. One of the two windows (*main_window*) is moved from the beginning of the sequence of events to the end. The other (*auxiliary_window*) is moved from the last position of the main window to the end of the stream. The windows never overlap because it does not make sense in our case to look for similarities between a sequence of events and a partial copy of itself.

A simple example illustrates these concepts in Figure 4, where a sequence of events (represented as letters) is shown. The four pairs of lines below represent examples of sliding windows, the main one on the left. The string distance

```
FOR_EACH window size (between two thresholds T1 and T2)
  FOR_EACH position (from beginning to end of stream)
    get main_window
    FOR_EACH window size (between two thresholds)
      FOR_EACH position (from the end of the main window until end)
        get auxiliary_window
        calculate and store distance between both windows
      END_FOR
    END_FOR
  END_FOR
END_FOR
```

Fig. 3. Sliding windows

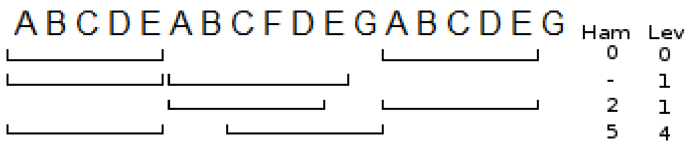


Fig. 4. Sliding windows. The distance between both windows depends on the distance used: e.g. if a Hamming distance is used, the distances are 0, N/A, 2, 5; if a Levehnstein distance is used, the distances are 0, 1, 1, 4.

between each pair of windows is calculated using both a Hamming and Levehnstein distance (cf. [15]).

3.2 Detection of Rhythm in eXpresser

This section describes the application of this process in the context of eXpresser and the footpath task. In this case, the events are generated by tiles placed on the canvas of eXpresser, and a rhythm –when detected– indicates some hidden structure that can be further used to provide adaptive feedback.

Preparation. The first step is choosing the two thresholds on which the sliding window algorithm operates. This usually involves some level of knowledge elicitation from the experts in the domain. In our case, this elicitation process produced a series of possible "rhythmical structures" or possible "building blocks" that can be used to construct the footpath pattern. These are depicted in Figure 5. The shortest ones have 3 tiles, while the longest ones have 5 tiles. Therefore, we chose $T1 = 3$ and $T2 = 6$, to allow insertion errors in the longer rhythms.

The next design decision is choosing which distance will be measured between windows. In our case, after observing several recorded sessions with students and discussing the most common typical errors on the part of the students with the pedagogical team, we came to the conclusion that most would be in the form of insertions/deletions, e.g. putting some tiles out of place in the sequence of positions. Therefore, we decided to use the Levehnstein distance in the first phase. Allowing the possibility of a certain Levehnstein distance between the windows permits the detection of rhythm in the presence of noise: e.g. in our case, tiles out of place.

In the next section, the process is executed step by step for a particular scenario.

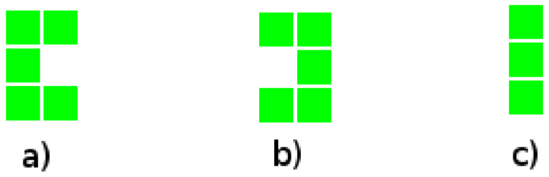


Fig. 5. Typical rhythmical structures in the footpath task: the "C", the "inverted C", and the "column"

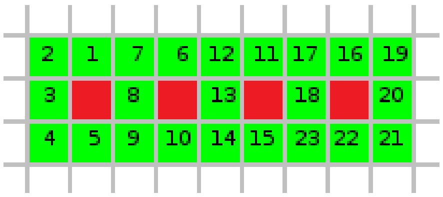


Fig. 6. Tiles position on a footpath-like task. Numbers show the order in which individual tiles have been placed on the canvas.

Rhythm of a student. In this section we illustrate all the steps in the process with an example. We start from Figure 6, that shows the order in which the tiles were placed in eXpresser. As it has been explained in Section 2, the goal of this task in eXpresser is to find how many green tiles are there for any given number of red tiles (i.e. $5 \cdot reds + 3$), and students are encouraged to construct the patterns themselves to see the relationships between both types of tiles. However, some students just put tiles on the eXpresser canvas "drawing" the pattern with colour tiles.

In this example, we see that the student has dropped tiles with high regularity until the end, where the characteristic "C" shape is abandoned to put first the tiles at the end. This and other similar behaviours have been observed by our team in pilot studies with the eXpresser as well as sessions in the context of a mathematical lesson in classrooms [12].

The output of the sliding window phase is stored in a table. Some sample rows of this table are shown on Table 1 to illustrate the result of this phase. In the table, p denotes the position of a window and s denotes its size.

The full table is analysed to find those windows that have a higher number of repetitions with lower distances. In this case, there are four windows that are repeated (i.e. reproduced with distance zero) four times. They are shown in Figure 7.

Table 1. Some distances for the example in Figure 6 (p denotes the position of a window and s denotes its size). Notes: † This corresponds to Figure 7c. ‡ This 3-tile-rhythm is discarded in favour of the longer rhythms (of size 5), in which it is embedded. ◇ This corresponds to a copy of the "C" (Figure 7a), and is ignored in favour of the first instance (window of size 5 on position 1).

p_{main}	s_{main}	p_{aux}	s_{aux}	*distance*	comments
1	5	6	5	0	
3	5	8	5	0	†
2	3	19	3	0	‡
1	5	7	5	2	
6	5	11	5	0	◇
1	5	16	5	4	

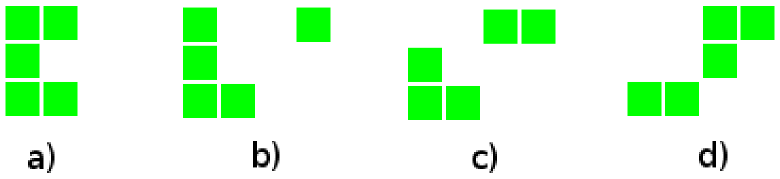
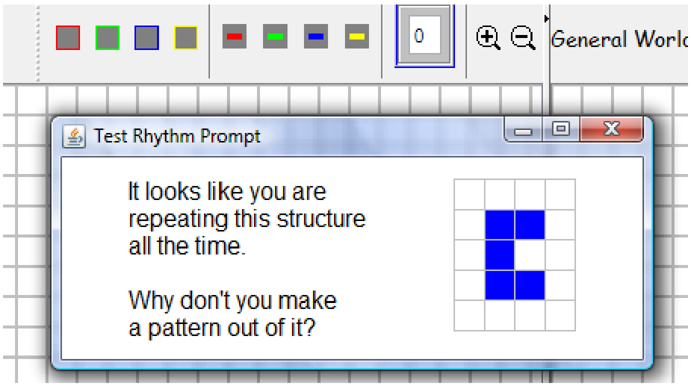


Fig. 7. Possible rhythms detected in the tiles position shown in Figure 6. All of them are repeated four times, i.e. there are four entries for each of them that show a distance of zero. The windows from which (a) and (c) are derived have been highlighted in Table 1.

It should be noted that the result of the former two stages might have been only option a) in Figure 7, if only the student's rhythm had been regular until the last tiles. In that case, the "C" structure would have been repeated five times and would have been the only output at this stage. However, this kind of small irregularities are very common among the students. This introduces the need for additional filtering using task knowledge. Although the four options are equivalent from a mathematical point of view (i.e. all of them have five green tiles), the expertise distilled from the pedagogical team in the project showed that for our target group building blocks with connected tiles facilitate children's reasoning process and realisation of structure. For this reason, we employ a filter that accepts only those rhythms in which all tiles are connected. This results in the rejection of options 7b, 7c, and 7d, and choosing 7a as the only possible rhythm in the actions of the student. The result is the same as a human teacher would have chosen.

Once the rhythm has been detected, feedback is provided to the student suggesting the use of such a building block to create the figure in a more structured way: repeating the building block to create a pattern. An example of such a form of feedback is shown in Figure 8. This feedback can be scaffolded in different ways in the architecture of the system (the interested reader is referred to [16]).



It looks like you are repeating this structure all the time.

Why don't you make a pattern out of it?

Fig. 8. Example of "rhythm" feedback

4 Evaluation

Detection accuracy. We have tested our algorithm with two different tasks in eXpresser, choosing from a a set of tasks that lend themselves to the appearance of "rhythm" in the action sequences of the student. Both tasks have been taken from the British National Curriculum, and are tasks that are commonly used in classrooms for the learning of mathematical generalisation. We have generated a series of scenarios for each task using five simulated students [17]. Each scenario corresponds to a student laying tiles on eXpresser, one by one. In five of these scenarios, simulated students show some rhythm (from those identified by the pedagogical team), but with some errors (e.g. spurious tiles among the ones that form the rhythm sequence); in the other cases, the laying of tiles did not follow these rhythmical patterns.

The algorithm was set up according to the characteristics of each task, i.e. the thresholds according to a pedagogical analysis as explained in Section 3.2, and the appropriate task filters are used. The algorithm was able to detect the rhythm in all test scenarios that included rhythm sequences. In the cases in which there was no rhythm in the actions of the artificial student, no rhythm was detected.

Computational cost and responsiveness. The highest computational cost of this approach comes from the first phase, where the algorithm's complexity is quadratic. In our application in eXpresser this is not an important handicap, since most of the students' interactions rarely involve more than a hundred tiles. Therefore, the response time of the algorithm is adequate for providing timely feedback when needed. As an extreme example, Figure 9 shows the response time for up to 300 single tiles generated by simulated students.

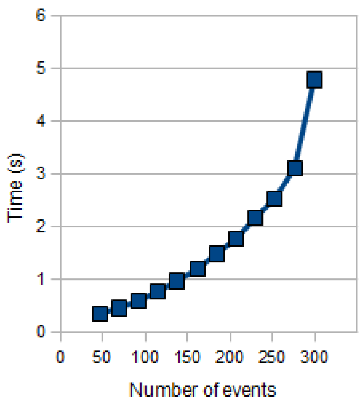


Fig. 9. Time used by our implementation to detect rhythm as a function of the number of tiles. For the scope of tasks in eXpresser (i.e. less than 100 tiles) the response time of the feedback is less than a second.

5 Related Work

There is a variety of sequential pattern mining algorithms for identifying repetitive patterns in sequential data (see [18]). Previous research suggested that an efficient approach to address our problem would be to mine for sub-sequences of sequential patterns over data stream of sliding windows (e.g. [19,20,21]).

Such approaches have been traditionally applied in online situations. However, challenges remain and they tend to cluster around the need to tolerate noise in the sequence [22] and to produce results with small amounts of data. The approach we presented here attempts to address these issues by combining two sliding windows with additional task knowledge filtering at the end, thus avoiding taking into account spurious effects.

6 Conclusions and Future Work

This paper has described a novel application of a method for detecting repetitive patterns in user actions on interactive systems. Detection of regularities in exploratory learning environments can be used to help students reflect on their actions. In our case, we use this to detect "rhythm" in the actions of students as they construct patterns in eXpresser tile by tile.

When detected, this personal "rhythm" is used to provide adaptive feedback to students that demonstrates to them the inherent structure in their actions. This feedback is crucial in helping them to articulate their implicit structure and derive a rule for the pattern they are constructing. Although any kind of related feedback would be useful, being able to detect the students' own inherent structure gives them a sense of ownership and contributes to helping them relate and internalise the feedback.

After the technical evaluation of the approach, we are in the process of validating the pedagogical relevance and appropriate timing of the feedback generated by this rhythm–detection component of our system. We are interacting with the pedagogical team comparing the feedback generated by the system and the interventions of the teacher related to rhythm in the actions of the students (i.e. gold–standard validation).

Currently, our approach relies on the task–specific window sizes. However, as more learners are interacting with the environment and data are collected from realistic interactions, our future efforts will concentrate on the design on an ensemble classifier for the selection of rhythms on the second stage. This would enable the automatic selection of window size.

References

1. Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A. (eds.): Watch what I do: programming by demonstration. MIT Press, Cambridge (1993)
2. Caglayan, A., Snorrason, M., Jacoby, J., Mazzu, J., Jones, R., Kumar, K.: Learn sesame - a learning agent engine. Applied Artificial Intelligence 11, 393–412 (1997)

3. Ruvini, J., Dony, C.: Learning Users' Habits to Automate Repetitive Tasks, pp. 271–295. Morgan Kaufmann, San Francisco (2001)
4. Ramly, M.E., Stroulia, E., Sorenson, P.: Mining system-user interaction traces for use case models. In: IWPC 2002: Proceedings of the 10th International Workshop on Program Comprehension, p. 21. IEEE Computer Society, Los Alamitos (2002)
5. Radford, L., Bardini, C., Sabena, C.: Rhythm and the grasping of the general. In: Novotná, J., Moraová, H., Krátká, M., Stehlíková, N. (eds.) Proceedings of the 30th Conference of the International Group for the Psychology of Mathematics Education, vol. 4, pp. 393–400 (2006)
6. Noss, R., Hoyles, C.: Windows on Mathematical Meanings: Learning Cultures and Computers. Kluwer Academic Publishers, Dordrecht (June 1996)
7. van Jong, D.T., Joolingen, W.R.: Discovery learning with computer simulations of conceptual domains. Review of Educational Research 68, 179–201 (1998)
8. Mayer, R.E.: Should there be a three-strikes rule against pure discovery learning? - the case for guided methods of instruction. American Psychologist, 14–19 (2004)
9. Kirschner, P., Sweller, J., Clark, R.E.: 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), 75–86 (2006)
10. Mason, J., Graham, A., Wilder, S.J.: Developing Thinking in Algebra. Paul Chapman Publishing, Boca Raton (2005)
11. Orton, A.: Pattern in the teaching and learning of mathematics. Cassell (1999)
12. Noss, R., Hoyles, C., Mavrikis, M., Geraniou, E., Gutierrez-Santos, S., Pearce, D.: Broadening the sense of 'dynamic': a microworld to support students' mathematical generalisation. Zentralblatt fur Didaktik der Mathematik 41, 493–503 (2009)
13. Cuoco, A.E., Goldenberg, E.P., Mark, J.: Habits of mind: An organizing principle for mathematics curriculum. Mathematical Behavior 15(4), 375–402 (1996)
14. Noss, R., Healy, L., Hoyles, C.: The construction of mathematical meanings: Connecting the visual with the symbolic. Educational Studies in Mathematics 33(2), 203–233 (1997)
15. Navarro, G.: A guided tour to approximate string matching. ACM Computing Survey 33(1), 31–88 (2001)
16. Gutierrez-Santos, S., Mavrikis, M., Magoulas, G.: Layered development and evaluation for intelligent support in exploratory environments: the case of microworlds. In: Intelligent Tutoring Systems, ITS 2010 (to appear 2010)
17. Vanlehn, K., Ohlsson, S., Nason, R.: Applications of simulated students: an exploration. J. Artif. Intell. Educ. 5(2), 135–175 (1994)
18. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining & Knowledge Discovery 15(1), 55–86 (2007)
19. Chang, L., Wang, T., Yang, D., Luan, H.: Seqstream: Mining closed sequential patterns over stream sliding windows. In: IEEE International Conference on Data Mining, pp. 83–92 (2008)
20. Zhou, A., Cao, F., Qian, W., Jin, C.: Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems 15, 181–214 (2008)
21. Gama, J., Rodrigues, P.P., Sebastian, R.: Evaluating algorithms that learn from data streams. In: SAC 2009: Proc. of the 2009 ACM Symposium on Applied Computing, pp. 1496–1500. ACM, New York (2009)
22. Wang, W., Yang, J., Yu, P.S.: Mining patterns in long sequential data with noise. SIGKDD Explor. Newsl. 2(2), 28–33 (2000)