HAL
open science

Constructing structure in number sequences

Yishay Mor, Richard Noss, Ken Kahn, Celia Hoyles, Gordon Simpson

To cite this version:

Yishay Mor, Richard Noss, Ken Kahn, Celia Hoyles, Gordon Simpson. Constructing structure in number sequences. 7th International Conference on Technology in Mathematics Teaching (ICTMT7), 26 - 29 July 2005, 2005, Bristol, United Kingdom. 9 p. hal-00190138

HAL Id: hal-00190138

https://telearn.hal.science/hal-00190138v1

Submitted on 23 Nov 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Presented at the 7th International Conference on Technology in Mathematics Teaching, Bristol, UK, 26 - 29 July 2005

CONSTRUCTING STRUCTURE IN NUMBER SEQUENCES

Yishay Mor, Celia Hoyles, Ken Kahn, Richard Noss, Gordon Simpson

London Knowledge Lab, Institute of Education, University of London

This paper reports on a design experiment in the domain of number sequences conducted in the course of the WebLabs project[1]. In this study, we designed and tested a set of activities in which 13-14 year old students use the ToonTalk programming environment to construct models of sequences and series, and then use the WebReports web-based collaboration system to share these models and their observations about them. We utilise a design pattern (programming method) called "Streams" which enables students to make sophisticated arguments regarding the sequences' mathematical structures.

INTRODUCTION

In many countries pattern recognition and generalisation are considered fundamental to mathematical thinking, and a fruitful pathway into algebraic thinking (Sassman et al, 1999; Zazkis & Liljedahl, 2002). Yet at the same time, a number of researchers have pointed to the difficulties students encounter in shifting from pattern spotting to structural understanding (Radford 2000; Noss et al, 1997). Students erroneously base their conclusions on superficial or incidental patterns they observe in the sequence, rather than on arguments referring to its structure. Although the use of structural reasoning increases modestly with age, empirical reasoning remains widespread (Küchemann & Hoyles, 2005). In the case of number sequences, some of the aforementioned researchers have suggested that one of the obstacles to developing an understanding of structure is students' tendency towards a recursive view. That is, identifying the relationship between consecutive terms rather than its general rule of the sequence. By contrast Weigand (1991) posits that iteration sequences offer rich mathematical experiences that should be exploited in activity design.

Several attempts have been made to explain these difficulties. Cottrill et al (1996) use the APOS theory (Dubinsky, 1992), while others propose co-variation, correspondence, or a property-oriented view (Confrey & Smith, 1994; Salvit, 1997). Regardless of the interpretative framework, two observations are universal: first, that number-pattern spotting is a predominant solution strategy, and second that the recursive form is a predominant description strategy. Indeed, pattern spotting lacks the definitiveness of a formal argument, and the recursive form does not generalise easily to the real numbers. Nevertheless, we conjecture that it is better to work *from* them than *against* them, to design learning activities that allow students to start from

[1] We acknowledge the support of the European Union, Grant # IST-2001-32200, directed by Prof. Richard Noss and Prof. Celia Hoyles. (http://www.weblabs.eu.com.)

intuitive forms, formalise them, and then hopefully develop alternative ways to explore problem situations.

The work reported here was done in the course of the *WebLabs* Project, which aims to explore new ways of constructing and expressing mathematical and scientific knowledge in communities of your learners. Our approach brings together constructionist and knowledge building perspectives. We use ToonTalk as our primary tool for construction, while supplementing it with any other tool that the students find useful (e.g. Excel). We have developed a web-based collaboration system called *WebReports*[2] for sharing and discussing these constructions. This system allows students to seamlessly embed their models in free form text documents they publish. The details of the project have been described elsewhere (Simpson et al, 2005; Mor, Tholander, & Holmberg, 2005), and in a symposium at this conference (Hoyles et al).

Our activities follow a common pattern. We begin each activity by discussing an intriguing mathematical theme. We encourage students to propose conjectures or derive concrete questions to explore, which are then formulated by us into programming tasks; Students complete these tasks individually or in pairs and publish their models (ToonTalk programs) along with their observations about them. These models are used as input to an instructor-led group discussion. The product of this discussion is a webreport which represents the concerns of the group. By engaging students in a discussion they are provoked to reflect on their work. Mistakes are exposed and explained by peers, rather than a figure of authority. Students acknowledge the need to construct rigorous arguments for their claims, and negotiate socio-mathematical (Yakel & Cobb, 1995) and socio-technical norms.

Ideally, at this point the webreport would be reviewed by another group, perhaps in another country, and an inter-group discussion would ensue, using the WebReports "comment" mechanism[3]. In reality, we rarely succeeded in orchestrating such a discussion due to pragmatic limitations. Alternatively, where possible, the class was split into two or more groups for the concluding discussion and webreport authoring. Each group then elected a representative to present their webreport to the whole class using the electronic whiteboard.

The above pattern of activities emerged from the iterative design and evaluation of this and other experiments (Simpson et al, 2005). It attempts to combine the power of individual constructionist learning (Papert & Harel, 1991) and collaborative knowledge-building (Scardamalia & Bereiter, 1994). Our design also aims to make the most of the new media at our disposal, without losing the established power of traditional educational methods. In a way, the students are mimicking the practice of a

---

[2] http://www.weblabs.org.uk/wlplone

[3] The comment mechanism allows learners to react to each others ideas not only by textual remarks but also by posting alternative ToonTalk models.

research and development team, the only difference being that their broad research path is mapped out by us beforehand.

This paper reports on a set of activities designed for students to investigate number sequences, which also introduces students to the WebLabs programme as a whole. They are designed assuming a minimal programming competence. After completing them the students proceed to activities which explore notions of rate and graphing, convergence and divergence, randomness, cardinality, or patterns in the Fibonacci sequence. The main aim of this initial set of activities is for students to learn to reason and argue about the structure of number sequences. This is achieved by allowing students to develop an alternative, non-algebraic language for mathematical discussion. This new language emerges from their collaborative discourse and stems from their experience in constructing programs which generate and modulate sequences.

Students start by modelling the most basic sequence: the natural numbers. However, the way we encourage them to model it affords easy generalisation to any arithmetic sequence, and later to any iterative sequence. The second activity focuses on summing the sequences created. Students program a generic component which produces the sequence of partial sums for any given sequence, thus engaging with the idea of summation as an *operation* which transforms one sequence to another. These activities provide the foundations, both in terms of tools and of knowledge, for further sets of activities that explore topics such as convergence and divergence, or the Fibonacci sequence.

DESIGN OF THE TOOLS AND ACTIVITIES

The first activity with sequences is the "Add-a-number challenge". Its motivating question was posed more as a ToonTalk puzzle than a mathematical one. We asked the students "*how would you train a robot* [the ToonTalk equivalent of "program a procedure"] *to count (1, 2, 3, 4, and so on)*"? As expected, students would propose a construction similar to the one in Figure 1, and we would follow their instructions on the interactive whiteboard. However, this solution has a serious flaw: the robot does not maintain a record of the numbers it generates. Since all computations are done "in place", the only term of the sequence we can access is the last.

This problem provides a motivation for introducing *birds*: ToonTalk's message-passing mechanism. Whenever a bird is given an object, it will carry it to its *nest*. If we provide our robot with a bird, and train it to hand the sequence term over to it, we will have them stacked on the nest as the robot runs.

	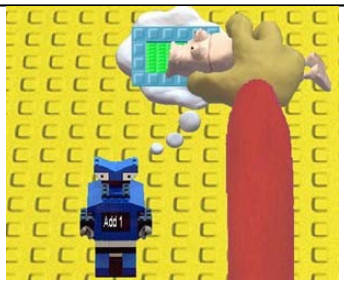	
Train the robot to take a number 1 from the toolbox and drop it on the input, to increment it.	Generalise the program by erasing the value of the input from the robot's memory.	Give the robot its input box. The robot will continually repeat the actions it has been taught.

Figure 1: Training a robot to count

Now we are ready for the programming task: train a robot to generate the natural numbers, and send them to a nest. To scaffold students' work, we provide an *active worksheet*: a webreport template that includes the instructions for the task and questions related to it. Students create a webreport of their own by clicking a button on this page, and fill in the answers as they go along. The unique feature which makes the worksheet "active" is that the ToonTalk tools required for the task are embedded in it, and at the click of the mouse students can load them into their programming environment. In this particular case, the worksheet contains a *task-in-a-box* (Figure 2): a ToonTalk box containing task instructions, an untrained robot, an input box, and output nest. The task-in-a-box serves several purposes at once. First, it helps students overcome the shift in medium from a (mainly textual) web page to the animated programming environment. More important, it scaffolds their work by providing the input box to be used in training. Last, it implicitly sets a standard for packaging and sharing ToonTalk models.

Training *Add number*

Train a robot to repeatedly add 1 to produce the numbers 0, 1, 2, 3.... Call the robot *Add number*.

(Click the box to get started, post your robot here when you're done)

Figure 2: `Add-a-number` task-in-a-box

The input box contains two holes with numbers: an `increment` and the `current` value. A third hole contains the `output` bird. The robot needs to be trained to perform two actions: drop a copy of the `increment` over `current` (thus adding them), and then hand a copy of `current` to the `output` bird. This is the first occurrence of the *Stream pattern*: the numbers are sent out to the nest, one after the other, *"ad infinitum"*.

From a mathematical point of view, the *streams* method generates the natural numbers by repeated application of the *successor* function. By constructing this procedure,

manipulating it and using it as a building block in larger constructions, students reify the natural numbers as the product of this process; perhaps not as strong a formalisation as Peano's axioms, but we believe, a step in that direction. A second important mathematical concept is prompted by a unique affordance of ToonTalk. Most programming languages distinguish clearly between constants and variables. Code is written for the general case ("*any n*") and tested for specific cases (or written for a singular setting). ToonTalk employs *programming by example*. This means that robots are trained for specific values, which can then be generalized by "erasing" the specific value from the robot's memory. In the case of the `Add-a-number` robot, the first generalization is required immediately: after the robot runs once, the value of `current` is no longer 0, and needs to be generalized if we want the robot to continue counting past 1. However, by generalising the `increment` as well, students can use the program to generate any arithmetic progression! The next part of the worksheet asks students to predict which sequences can be generated by their robot and which cannot. These questions aim to promote students' mathematical conjecturing and argumentation, and specifically raise their awareness to the relationship between the *procedural* and the *structural* facets of sequences. After reflecting on these examples, students are asked to provide one more sequence that their robot can generate and one it cannot. The latter question is probably the hardest, in order to say that the robot *cannot* produce a sequence one has to argue about the nature of the *class* of sequences it *can* produce.

Once students have posted their `Add-a-number` robot and answered the questions, they are introduced to the next task: train a robot to add up the terms of a sequence. We refer to it as the `Add-up` robot. Mathematically, this robot embodies the concept of a partial sum series, and implements it as a function on the domain of sequences: for any given sequence, it will produce the sequence of its partial sums. In concrete terms, we give the nest of the first sequence to the `Add-up` robot, which sums the numbers coming in to that nest, and sends the results out to its output nest (Figure 3).

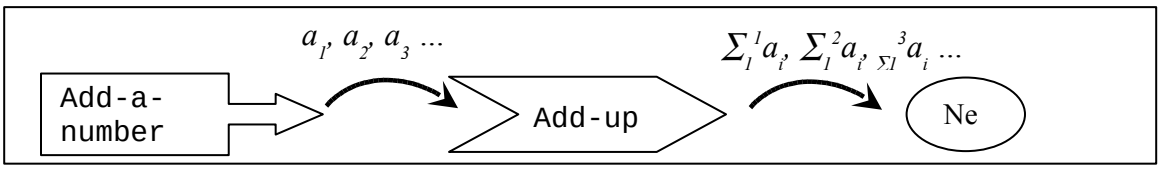

Figure 3: "Chaining" `Add-a-number` to `Add-up`

Again, we rely on ToonTalk's features and utilise the streams pattern. ToonTalk is a concurrent language, which means that several programs (robots) can run concurrently. This allows us to generate a sequence and add up its terms at the same

time, while keeping the two processes clearly distinguishable. Directing students to this pattern addresses two aims. First, our initial experiments have shown – as suggested by the literature – that students tend to become confused between source sequences and the corresponding sequences of partial sums. This confusion causes difficulties in reasoning about limits, and sequence behaviour in general. Secondly, by using one sequence as an input to a process which generates another one, we address both process and object perspectives and encourage students to construct links between them.

Students are asked to construct the `Add-up` robot and post it on their webreport. They then chain it with the `Add-a-number` robot, and experiment with summing different sequences. Next, they are asked to answer some questions regarding the chain of robots. Observing the patterns in and between these examples can lead to conjectures regarding the rules governing the co-variance of the source sequence and the corresponding sequence of partial sum. The `Add-a-number` and `Add-up` phase of activities was concluded by a group discussion. The discussion revolved around the goal of composing a consensus webreport based on the individual webreports for `Add-a-number` and `Add-up`. The teacher used an active worksheet displayed on an interactive whiteboard. First, the `Add-a-number` robot was constructed by the group, with the teacher acting as a proxy. One of the students instructed the teacher how to train the robot, and where others disagreed, they discussed their solutions until a consensus was reached. After the robot was trained and posted, the students continued to discuss the answers to the questions in the worksheet. At this point the teacher displayed students' individual webreports to refresh their memory. This process was iterated for the `Add-up` robot.

RESULTS

The results reported here are from an experiment conducted in London in autumn 2004. This experiment involved a group of 10 boys, age 13-14, for 6 hourly sessions and a full day workshop. The activities have also been tested concurrently and in the two preceding years and in Bulgaria, Cyprus, London and Portugal.

Most students found the activities engaging. They completed the tasks successfully, and then refined their answers through collaboration. They identified the natural numbers as a case of arithmetic progression, and then expanded that class to a more general one. They used formalisations derived from their ToonTalk experience to make sophisticated mathematical arguments.

During a group discussion about the `Add-a-number` robot, several students expressed the generalization of the natural numbers to arithmetic sequences:

> Yishay: The next thing is, what about this sequence? -1, -2, -3, -4? Yeah, Morris?
>
> Morris: Change the current to 0 and change the 'add this' to -1.

Moreover, they saw no distinction between positive and negative numbers – as long as the structure is the same. In the words of Andy, a minute later:

> Gordon: So what is it that stays the same and what changes to create this sequence compared to 1, 2, 3, 4, ...?
>
> Andy: Um, its just, it ... what it does stays the same but just the numbers it uses are changed.

ToonTalk terminology became part of the students' repertoire, allowing them to develop a formalism which is situated in their activity. In several groups, students consistently referred to the natural numbers as "*the add-one sequence*". Once the formalism is established, the natural numbers become an instance of the class of arithmetic sequences.

Collaboration through writing and commenting on webreports took the students one step further. Regarding the `Add-a-number` robot, one group wrote:

you cannot do: - square numbers - anyting where you times or divide - in can't go up in prime numbers - any sequence with two stages - tringular numbers	It can only go up (or down) in the same number each time. That's why it can't do all those sequences on the left.

To which the other group commented:

> You assumed that you cannot mulitply *[sic]* or divide, but this can be done. You can also do the square numbers, by using ^2. We disagree with your statement "any sequence with two stages" because you could use advanced formulas (*2; +1).

This second group of students had discovered that ToonTalk allows them to substitute the additive for any function, and in the act had re-formalised arithmetic sequences as a special case of iterative sequences.

Constructing the `Add-up` robot inspired further sophistication of students' arguments. Again, when asked if a particular sequence can be generated by the chain they remarked:

2, 6, 16, 20, 30,...	No. Because you cant retrain the robot. You would need two add this boxes (4 and 10) or retrain the robot to alternate between adding 4 and 10 (change the value by typing).

This is not a programming claim – it is a mathematical one. `Add-a-number` can generate any arithmetic sequence. Chain it with `Add-up`, and you get a corresponding sum series. The sequence at hand is the result of alternately adding 4 and 10 – and is neither of the first form nor the second. The other group responded:

> You could do it if you could get two birds feeding into one nest.

i.e. you could express this as a sum of two alternating sequences. But when Allen presented their response in a group discussion, a second observation emerged:

> We believe if you change the 16 to a 12 it would be fine. If you um… started with um… with the 'in' as 2 'cause each it'll go up by 2: [*pointing at the spaces between the sequence terms*] 2, 4, 6, 8, 10, 12 and so on. So you get the answer, uh, and that would be a way without actually having two birds, which is impossible.

He agreed that the sequence was not a sum series of an arithmetic progression, but if you changed the third term to 12 it would be. Not only did he note the structure of the sequence, he also saw how a new structure could be constructed from it.

CONCLUSIONS

Our main conclusion is that, under rather carefully controlled circumstances and with a great deal of design effort, the modelling approach – in which students construct and share programs that model rich phenomena – can assist in developing students' understandings of structure and consistency in mathematical situations. Certainly our students did not yet have access to the armoury of algebraic tools that would be necessary for a detailed study of these phenomena, but the tools we designed did provide an interim solution to this difficulty that at least led to engagement with highly non-trivial ideas in mathematics. In our design, these formalisms emerged out of a combination of constructive and collaborative activities in which the "streams" design pattern allowed students to mould their intuitions into a situated formalism with which they could explore quite complex ideas, and argue convincingly and with commitment for their hypotheses.

REFERENCES

Hoyles, C., Kahn, K., Mor, Y., Noss, R., Sendova, E., Sacristán, A. I. and Simpson, G.: this conference, The WebLabs Project: Building New Formalisms for Mathematical and Scientific Ideas.

Mor, Y., Tholander, J. and Holmberg, J.: 2005, Designing for cross-cultural web-based knowledge building, *Proceedings of The 10th Computer Supported Collaborative Learning (CSCL) conference*, Taipei, Taiwan.

Noss, R., Healy, L. and Hoyles, C.: 1997, The Construction of Mathematical Meanings: Connecting the Visual with the Symbolic, *Educational Studies in Mathematics* 33 (2), pp. 203-233.

Papert, S. and Harel, I.: 1991, Situating Constructionism, in (Ed.), *Constructionism* Ablex Publishing Corporation.

Radford, L.: 2000, SIGNS AND MEANINGS IN STUDENTS' EMERGENT ALGEBRAIC THINKING: A SEMIOTIC ANALYSIS, *Educational Studies in Mathematics* 42 pp. 237–268.

Salvit, D.: 1977, AN ALTERNATE ROUTE TO THE REIFICATION OF FUNCTION, *Educational Studies in Mathematics* 33 pp. 259-281.

Sasman, M., Olivier, A. and Linchevski, L.: 1999, DEVELOPING AND STIMULATING GENERALISATION THINKING PROCESSES AND SKILLS.

Scardamalia, M. and Bereiter, C.: 1994, Computer support for knowledge-building communities, *Journal of the Learning Sciences* 3 (3), pp. 265–283.

Simpson, G., Hoyles, C. and Noss, R.: 2005, Designing a programming-based approach for modelling scientific phenomena, *Journal of Computer Assisted Learning* 21 (2), pp. 143-158.

Zazkis, R. and Liljedahl, P.: 2002, GENERALIZATION OF PATTERNS: THE TENSION BETWEEN ALGEBRAIC THINKING AND ALGEBRAIC NOTATION, *Educational Studies in Mathematics* 49 pp. 379–402.