Constructive and Instructive Representation

DECLAN O’REILLY
University of Sheffield, United Kingdom
DAVE PRATT
University of Warwick, United Kingdom
PETE WINBOURNE
South Bank University, London, United Kingdom

ABSTRACT Beginning teachers of mathematics are likely to have noted the development of powerful multiple representation software which offers children access to the many modalities through which mathematics is expressed. We argue that embedded, even hidden, within such software are many mathematics conventions, which the naive learner has to unravel in order to construct meaning for those representations. We contrast such representations, which we label as instructive, with those children construct through the use of expressive software; this contrast is seen as analogous to aspects of literacy. We identify various characteristics of these two distinctive forms of representation.

Introduction
There can be few, if any, beginning teachers of mathematics who do not believe that computers are changing and will continue to change the way that mathematics is learned and the way that they will teach it. Yet, many will be unaware of how different software can influence the pedagogical process. In this paper, we seek to highlight these differences by polarising the representations permitted by such software into two contrasting classes: instructive versus constructive representations. We acknowledge that this is an over-simplification: the reality is perhaps more akin to a continuum. Nevertheless, it allows us to focus on key characteristics of each – differences which we suggest are analogous to those between reading and writing.

Perhaps, the most fundamental difference lies in the intention of the software designer. At one pole, the priority is that of ease of access. At the
other, the stress is on giving the user control. Noss & Hoyles (1992) are critical of the former, arguing that such systems limit users to interpretation rather than construction. As they say:

Given that a primary aim of mathematical activity is for students to interact with mathematical objects and processes, it is often insufficient for students to see what happens, they need to know how it happens - and this is why we see programming as an important component of computer-systems. (p. 3)

On the other hand - as critics of Logo have pointed out - getting students to this point is no easy matter. There may be a long period during which the student is initiated into learning how to use the software. Moreover, there may be a greater need for teacher intervention, particularly in the early stages of learning. In this paper, we look at how relatively recent advances influence the two forms of representation. We present some vignettes: the first relating to the use of graphical calculators, the second to the use of Boxer. Boxer is a computational medium, which encourages children to express their ideas in a variety of forms, including the written language, graphical output and, crucially for our purposes, through programming in a Logo-like language. Boxer is being developed at Berkeley University, California. These vignettes are chosen then specifically to highlight the differences between what we term instructive and constructive representation.

**Instructive Representation**

We use the phrase instructive representation to indicate, simply, that a particular set of representations has been constructed with the aim of mediating mathematical instruction. Note that it is the intention of the producer of the representation which is the defining characteristic here; from a Vygotskian perspective, that intention will reflect a culturally mediated view of the function, as learning tool (Kozulin, 1990), of the available forms of representation. The producer (possibly a teacher, more likely a manufacturer of computer technology) will have as chief amongst their aims the induction of the intended users into the culture within which they are working.

Graphical calculators provide multiple representations of mathematical ideas. They are increasingly common in schools and, we believe, classroom activities that make use of graphical calculators generally exemplify instructive representation. Here we give a brief description of one such activity, then we present a vignette which, we feel, illustrates a potentially challenging secondary effect of instructive representation.

A class of students (12-13 years of age) were asked to focus on equivalence within and between numerical, symbolic and graphical representations of mathematical function. They were each ‘beamed’
electronic copies of files which included tables of values and accompanying notes (see Figures 1 and 2).[Technical note: The calculators the students were using were HP38Gs. These machines can send and receive files (aplets) which are collections of multiple representations (or views) of functions, but which can also include notes and pictures. The aplet the students were sent in this example contained pictures of the numerical or tabular representations of six functions; being simply pictures these did not themselves support any further interaction and so could be used to give only numeric clues to the other representations.]

Figure 1. Tables of values.
Figure 2. Accompanying notes.

The activity was judged to be successful against a number of criteria relating to prevailing curriculum objectives. For example:
- most of the students found symbolic and graphical representations of all six functions;
- one student devised a ‘method of differences’ for herself and used this directly to find symbolic forms of all but the quadratic function. Other students’ methods were less well-defined, but most could use their machines to reproduce them and so, in this sense, their methods could be articulated and inspected;
- the activity produced some useful discussion about symbolic equivalence; for example, three forms of symbolic representation of F6 were found to have ‘worked’: F6(X) = .5(X-1), F6(X) = .5X – .5, F6(X) = X/2 – .5

All representations have the capacity to create potential problems as well as learning opportunities (Goldenberg, 1988; Kaput, 1989). This activity reflected typical attributes of instructive representation. On the one hand, it reflected the traditional hierarchical view of the curriculum in that the aim was for students to conceptualise function as an abstract symbolic relation; their understanding of function as either numeric or graphical entity was intended to act as a stepping stone to this. On the other hand, using a
graphical calculator in instructive representation mode produces a secondary effect: this hierarchy becomes visible and thus questionable. As the following vignette shows, students can now ask why they need symbolic skills when they can develop their own well-understood graphical or numerical algorithms for understanding functions and use these effectively in associated activities such as solving equations.

**Vicky and Cecilia**

Vicky and Cecilia were both about 16 years old. They had just taken their GCSE examination and had both decided to continue with their mathematics next year at Advanced level. They, and the other students in their class, had been using graphical calculators in their mathematics lessons for 3 years now, although their curriculum and general experience of mathematics had actually been little changed in response to this. They had, however, made use of their graphical calculators to investigate graphical methods of solution of equations.

Extract 1 is from where Vicky and Cecilia were talking about their methods for using graphical calculators to solve quadratic equations: Extract 1.
The interviewer decided that it is worth prompting to see if and how the graphical calculator might be used as a lens to facilitate articulation and examination of the students' awareness both of their own awareness (Mason, 1992) and also of the activity as situated in the practice of school mathematics (Lave & Wenger, 1991) in Extract 2.

Extract 2.

This conversation reflected the students' experience of instructive representation. At the same time, it showed how the variety of representation offered by the graphical calculator had enabled us to see the mathematics in which they were engaged as a social practice, a social practice, moreover, whose valuing of particular algebraic skills originated within an identifiable community (Lave & Wenger, 1991).

Constructive Representation

We continue our characterisation of instructional representation through contrast with the following example. We will now describe in some detail an example of what may for most readers be a less familiar form of representation.
Context

The class of 10/11 year-old children had spent about two weeks learning about functions, during which time they had been constructing cardboard models of number machines. Figure 3 shows a typical example.

Some of the students then asked if they could draw their function machines in Boxer. The following episode features one of those children, Nora. Nora had made a cardboard model of a function machine in the shape of a spider (Figure 4), with the four left legs feeding in the inputs and the four right legs showing the outputs. She now wanted to reproduce this in Boxer.

Figure 3. A typical cardboard model of a Function Machine.
CONSTRUCTIVE AND INSTRUCTIVE REPRESENTATION

Figure 4. Nora’s cardboard spider function model for $x$:

$3.5x - 6$. 
Construction

Nora created a module called ‘spider’, obtaining the ‘body’ of the spider by using ‘stamp-hollow-circle 100’, a Boxer primitive. She now went on to draw all 8 legs encountering few problems in doing so. A combination of the repetitiveness of typing the same commands over and over, together with the space needed to accommodate her typing, made her susceptible to the suggestion that she group the commands for the leg in a do-it box which she could then repeat. Not unnaturally, she called this second module ‘leg’.

The next stage of the construction was concerned with making her spider machine operational, and this was where most researcher intervention was necessary. Typing the original numbers in the appropriate boxes was relatively unproblematic, but moving from there to using variables required assistance. She chose leg1, leg2, leg3 and leg4 to depict her 4 input numbers, and appeared to understand this. However, she thought that she could also get the output numbers by typing leg5, leg6, leg7, leg8. Intervention pointed out that she could obtain these using ‘function leg1’, ‘function leg2’ etc. By the end of the session, she had obtained a working function machine consisting of a large central circle and eight orbital circles (Figure 5).

In one sense, she had achieved as much as the other students – albeit with help. But Nora said she wanted to draw the spider’s eyes, nose etc. Each of the eyes required the drawing of arcs etc. for which there were no ready-made primitives. There was some concern that she would encounter both programming and mathematics difficulties in pursuing this goal. However, Nora was insistent on drawing a spider machine in Boxer that exactly resembled her original cardboard model.
Nora’s desire to continue her project development was likely of course to lead her into areas which may have seemed aesthetically imaginative but mathematically uninteresting, even counter-productive. There is here a common dilemma when working with representations constructively. Noss & Hoyles (1992) have referred to the play paradox. On the one hand, there is a desire to encourage play which may support appropriation by the child of a particular task or assimilation of some new ideas. On the other hand, such play-based activity may involve considerable time spent on aspects not central to the teacher’s, or researcher’s, agenda.

Each time this tension is encountered it will only be resolved satisfactorily by attention to both aspects of the dilemma (and not always even then). In this instance, it was felt that to prevent Nora from continuing her work in her own style would have been destructive. Modular programming now became the problem-solving tool through which she could achieve her goal.

Figure 6. Nora’s eye do-it box.

It is worth dwelling on one aspect of this process – the eye (Figure 6) – as it illustrates one of the differences between a constructive and instructive approach to representation. Nora’s desire to get the eye exactly where she wanted it led her to dissect her line ‘repeat 16 [fd 2 lt 5.5]’ and replace it
with ‘repeat 8 [fd 2 lt 5.5]’, ‘rt 75 fd 6 stamp-ellipse 12 20’, ‘bk 6 lt 75’, ‘repeat 8 [fd 2 lt 5.5]’. Later still, she made even finer discriminations changing the input to the first ‘repeat’ from 8 to 9 and the input to the second ‘repeat’ from 8 to 7. From a mathematical perspective, the eyes etc. were extraneous details – the importance lay in getting the machine operational – but for Nora the importance lay in getting an accurate picture of her spider. The drawing and the subsequent mathematical activity could not be separated out.

Interaction

In interacting with a constructive representation as opposed to an instructive representation, there are at least two key differences. The first is the sense of ownership which children feel over those objects and the second lies in the intimate knowledge that they have of their operation – having constructed them in the first place. In the following episodes, this can be seen both in the form of interviewer questions and in Nora's
responses. The interviewer was asking and she was answering questions about her function machine, and began by setting Nora the task of finding a function, that would map 1, 2, 3, and 4 to 0.1, 0.2, 0.3, and 0.4 respectively.

Extract 3.

In this episode (Extract 3), Nora’s efforts to find a function to map 1, 2, 3, 4 to 0.1, 0.2, 0.3, 0.4 led her to divide by 0.10, then 0.1, then 0.01 and finally 10. Her first two attempts yielded outputs of 10.0, 20.0, 30.0 and 40.0, and reveal her puzzlement at obtaining the same answers for 0.10 and 0.1. Nora clearly thought that these were different quantities at this point. Her next attempt using 0.01 produced a ‘worse’ answer, but nevertheless seemed to provide her with a clues to her final – and correct – strategy of dividing by 10.

Extract 4 shows how Nora was asked not merely to find a function which would map 1, 2, 3, 4 to 0.01, 0.02, 0.03, 0.04 but to justify it as if explaining to a younger child.

Nora’s initial articulation: “There’s like a tiny little bit at the end of the number. It’s hardly a number at all.” typifies the struggle which young students have in conceptualising decimal quantities. Her later identification of 0.01, 0.02, 0.03, and 0.04 with one hundredth, two hundredths, three hundredths etc. seemed to have been directly related to her interactions with the function machine. Moreover, it marked a key instance in her developing sense of decimal numbers. From this point on, she seemed to be able to generalise her discovery, as shown in Extract 5.

Extract 4.

These extracts illustrate how Nora set about creating functions to map 1, 2, 3, and 4 to decimals such as 0.01, 0.02, 0.03, and 0.04. To do so, she needed
to go inside the program ‘function’ do-it box. Initially, her choices led her in the opposite direction. However, from the trace on the machine’s graphic, she was able to create functions that not only gave her the desired mapping, but also gave her an operational means of generalising the relationship between whole numbers and decimals. The Boxer formalisation seems to have helped to structure Nora’s expression of decimals in that she articulated them in terms of operation in the do-it box.

In making this assertion, we do not wish to understate the importance of the nature of the task within which this structuring has taken place (indeed, see Pratt & Ainley, in press, for an account of how critical the design of the task can be in shaping the interaction between child and computer-based structures). Neither do we mean to suggest that the researcher’s questions were not an important factor in this structuring process. The relationship between the questioning, the tools and resources within the software, and the internal resources of the child is complex. However, we do see environments which encourage constructive representation as offering special opportunities for gaining insights into children’s construction of meaning as they try to articulate their thinking in that medium. In that respect the researcher’s or teacher’s questions are themselves shaped by the interaction in which both child and researcher forge new connections (webbing in the terminology of Noss & Hoyles, 1996) between the various resources within both computer-based and human-based sites.

Extract 5.

Two Types of Tools

When we discuss representation in a computer context, we probably think about a range of symbolic and iconic images made available by the programmer and instantiated on the screen by the user. In the mathematical and scientific domain, images of graphs, charts and tables come quickly to mind. We might even contemplate more dynamic representations where such modalities are linked so that changes in one trigger immediate changes in another. We probably conceptualise these representations as tools in the sense that they are intended to help us to make sense of and to visualise data or mathematical relationships.

Tools can be used in different ways. For example, in many schools in England and Wales, Logo has become one more tool for representing to students a fairly small part of the mathematics curriculum. There are some schools, however, in which Logo is used as what in Confrey’s terms (1995) we might call a tool for listening to students’ voices, which, also requires that students be encouraged to express themselves mathematically. Now, we certainly do not want to deny the possibility of describing similarly the creative use of spreadsheets in some classrooms as an example of a tool used to enable students’ voices to be heard. From the Vygotskian
perspective (Kozulin, 1990), tool use is seen to be culturally bound; so, clearly we must allow for a wide range of possible uses of any kind of computer software. But, there are also, we believe, useful distinctions to be made between the technical nature of the computational tools we have most identified with constructive and instructive forms of representation. These distinctions, we believe, become particularly important when we set out with the intention of providing students with opportunities for expressing themselves and teachers with opportunities for listening to their students.

Complexity

One characteristic of these tools is that they are complex. They contain coded information and conventions which the prevailing culture has agreed upon and into which the user has to be inducted. Dreyfus & Eisenberg (1990) expressed their concern with respect to diagrams in observing:

Reading a diagram is a learned skill; it doesn’t just happen by itself. To this point in time, graph reading and thinking visually have been taken to be serendipitous outcomes of the curriculum. But these skills are too important to be left to chance. (p. 33)

A number of research studies from both mathematical and scientific perspectives have emphasised the difficulty that children have in reading a graph. Kerslake (1981), reporting on the Concepts in Secondary Mathematics and Science project, gave evidence of children’s misconception with distance/time graphs, whilst two studies with lower secondary children in Britain and North America (Swatton & Taylor, 1994; Padilla et al, 1986) showed relatively low success rates in interpretative skills (interpolation and particularly reading relationships between variables), but higher rates in some construction skills (such as plotting points).

Nevertheless, there is a general recognition of the importance of a child developing these graphical reading skills and teaching methods have been developed to help children. Research evidence from studies of data-logging projects (Nachmias & Linn, 1987; Mokross & Tinker, 1987; Brasell, 1987) have aimed to bring into close proximity the experiment and its graphical representation. Pratt (1995) describes a pedagogical strategy...
called active graphing in which the child uses a spreadsheet’s graphing facility in order to construct meaning for the graph, the tabulated data and the ongoing experiment. In the active graphing approach, the graph is seen by the child as a tool to help decision making during the experiment and is used iteratively throughout the experiment. Again the aim is to encourage successful reading of the graph by relating it to the tabulated data and to the experiment itself.

Our aim here is to point up the essential complexity of these representations and the effort that is required in order to read the representation itself. We wish to contrast this with another type of representation, which is less complex and whose tool-like characteristic is that of a building brick, a primitive from which more complex representations can be constructed.

diSessa (1995) refers to this sort of tool as a ‘small dollop’ in order to emphasise its lack of complexity. He sees these ‘dollops’ as being dropped (by the programmer) somewhere near to the learner’s path, inviting their use in the construction of new products, new representations.

In Logo, for example, we can think of the forward command, and its associated graphic representation when the turtle moves, as one such dollop. The young child may need to use the command a little before she feels some mastery of it but she will soon be able to use the command in order to build her own representations, perhaps a house, perhaps later a graph.

This process of coming across small dollops and using them to build more complex representations is more akin to writing than reading. diSessa’s view, and ours, is that this process encourages appropriation of the representations by the child.

In a self-imposed experiment, we recently used Boxer to build as many representations of the notion of mathematical reflection as could imagine. We were struck by the variety of representations that could be built from a relatively small number of such dollops. For example, we built a reflecting tool rather like ones we remembered drawing with ourselves as children. Figure 7 illustrates how, with this tool, we could drag the black dot with the mouse, and the white dot would automatically draw a reflected image. We built a menu box that allowed us to control when the pens were up (and so not drawing) or down.

Figure 7. A complex representation built out of small dollops.

**Tuning**

Another characteristic of instructive representations is that they are finely tuned to a particular purpose (see diSessa, 1985). Consider the scatter graph, a representation which has been used extensively in the research on active graphing (see for example Ainley & Pratt, 1995; and Pratt, 1995) The scatter graph is a tool finely tuned to the interpretation of stochastic data. It
provides an image of how data in an experiment varies; it shows trends in data such as correlations or clustering. However, the scatter graph is not appropriate for a whole range of situations which may seem very similar to the learner. Children need to learn a different type of representation for each new context. They have to learn to discriminate between contexts in order to appreciate when one finely tuned representation is needed as opposed to another. This is a non-trivial process as evidence from the Primary Laptop Project (Pratt, 1995) has confirmed.

Constructive representations appear as de-tuned tools, which have wide applicability, but therefore do not match immediately any particular context. They are intended as constantly available objects to work with in order to create new products. Children using these de-tuned tools will spend time constructing their own representations, in contrast to the use of finely tuned instructive representations, when they will spend time reading those representations. diSessa (1995) expresses this contrast as follows:

... experts and software designers will supply the general populace with highly tuned and elegant tools and other pieces of software that we will learn to use in the niches for which they were intended – symbol manipulators, graphers, simulations and simulation tool kits, and the like.

versus

... the creation and modification of the dynamic and interactive characteristics of the medium, the very characteristics that define the medium as an extension and improvement of text in the first place. In
metaphorical terms, reading without writing is only half a literacy. Deep computational literacy means ‘writing’ in addition to ‘reading’, creating as well as using.

Triggers

The connections and links between representations lies at the heart of much mathematics. It is recognised as extremely important that a child has multiple aspects to their concept image (Tall, 1989). What do we mean when we say a child understands the concept of function? Has the child constructed a meaning for the notation, and does this meaning include an image of an input/output machine, a table of values and a parabolic graph? Can the child envisage function as both a procedure and an object in its own right, which can itself be manipulated through operations such as composition or differentiation? The concept of function, like all powerful mathematical ideas, contains many interconnected ideas.

Multiple representation software demonstrates these links explicitly. In such software, changes in one representation trigger automatic changes in another. Thus a change in algebraic representation of the function will
immediately promote a corresponding change in the table of values and the corresponding graph. It is straightforward to set up a similar process with a spreadsheet (Figure 8). The table of values is constructed from the formula in the column, \( f(x) \), which squares the number in the column, \( x \). The graph represents the data in these two columns.

Figure 8. Triggering graphical representations in a spreadsheet.

This method of construction means that changes in the data will automatically trigger changes in the graph as in Figure 9.

Figure 9. Changes in the data automatically trigger a new graph.

The spreadsheet does not however offer the option that changes to the graph can trigger changes in the table of values, or the algebraic description of the function. The example in Figure 10 illustrates a case where the triggering has been made two or even three way. In this example, the user could change \( f(x) \) in the top left box which would trigger changes in the table of values or the graph. Equally, changes to the graph, carried out by dragging points on the graph, will trigger changes in the other two boxes.

This latter example offers one other very important educational advantage over the spreadsheet. The child has control over where she makes the links. This begins now to feel a little more like a constructive representation. However, the child is constrained to using other people’s representations, and this carries with it the difficulties of reading described above. Indeed multiple representation software runs the risk that the difficulties of reading a representation are simply multiplied up by the number of modalities represented on the screen simultaneously. The child has to make sense of each modality in turn AND the links between them.

Figure 10. Changes in each modality can be triggered by changes in another.

Figure 11. Boxer allows the child to build triggers into her own representations.

The Boxer environment offers structures which allow the child to program the links for herself. When we worked together to build our own representations of reflection, we envisaged a relationship between object and image which was both numerical and graphical. We wished to express the link between these two modalities. Figure 11 illustrates the representation that we built. A change in the object immediately triggers a change in the image box and a change in the graphical representation of object and image.
However, we needed to build that link between our various representations. Figure 12 shows the do-it box (or procedure in Logo terms) which represents that trigger. This modified-trigger box lies inside the object box, so that the modified-trigger is executed whenever the object box is modified. The first line triggers a change in the image box. The second line triggers a change in the graphical image.

Figure 12. This procedure programs a change in the object to trigger a change in the image and in the graphics box.

We suggest that whereas multiple representations software tends to emphasise the reading of links between different representations of a mathematical concept, expressive software like Boxer has the potential to place emphasis upon children programming their triggers between their own representations.

Concluding Remarks

In the previous sections, we have picked out three major contrasting characteristics of constructive and instructive representations. We are not arguing that the use of instructive representations should be replaced by a pedagogy in which children spend all of their time building their own representations. However, it does seem to us that the prevailing pedagogy in mathematics is dominated by the use of instructive representations; although these have the potential to challenge existing curriculum paradigms, they limit severely the literacy which children can experience and develop. Environments such as Boxer offer opportunities for children to express their own mathematical ideas in a way that most other currently available software does not. Such usage carries with it implications about knowledge, about pedagogy, and about the nature of mathematical activity in schools.

Knowledge

The small, de-tuned building blocks provided by Boxer have little, if any, relation to the standard forms of representation of function. In this context, knowledge of function was in some sense open to reconstruction by the student and that is why we describe Nora's use of Boxer as an example of constructive representation.

The building blocks provided by the graphical calculator, in contrast, are large and finely tuned. In this activity knowledge is defined in a manner which is, we suggest, representative of prevailing classroom cultures. In such contexts, the calculator is used as tool to ‘uncover pre-existing truths’ and this use carries with it clear epistemological implications.
By definition, a learning tool cannot be used in constructive representation mode unless students are genuinely in control and know themselves to be. Thus, the pedagogy associated with this mode of use is necessarily open and challenging, even subversive. In general, and again by definition, for a tool to be used in instructive representation mode, the roles of student and teacher must correspond to those to be found in traditional classroom environments. It is the teacher’s knowledge which is to be represented and the variety of representation is an aid to the students’ acquisition of that knowledge. Thus there is no primary challenge to traditional pedagogy and the status quo is, in fact, reinforced.

The Nature of Mathematical Activity.

The two contrasting forms of representation that we discuss here both have the potential to provide us, also, with windows (Noss & Hoyles, 1996) on the nature of the mathematical activity in which students and teachers believe themselves to be engaged. There are, however, significant differences in the aspects of that activity to which the windows provide access. Instructive representation does not itself enable us to view the child’s perspective, although, as we have seen, it can enable the nature of the mathematical practice in which the activity is embedded to be made visible. Nora’s use of Boxer demonstrates how the window provided by this constructive representation can look into children’s thinking.

Noss (1985, p. 33) observes that: ‘It is the language which determines the nature of the interaction between learner (programmer), machine and the ideas being programmed’ and he argues that the essence of the case for learning to program rests on the process rather than the product. Gray & Tall (1994), in discussing the use of software to learn mathematics, write: ‘we have evidence that the use of the computer to carry out the process, thus enabling the learner to concentrate on the product, significantly improves the learning experience’ (p. 137). In the past, these were necessarily discrete domains for the novice programmer. Our argument is that with a medium like Boxer, this need no longer be the case. Boxer makes it possible for young students to construct their own interactive software, and in the process gain the kind of mathematical insight of the product normally available to the expert programmer. This is in essence the literacy of constructive representations.
Correspondence

David Pratt, Mathematics Education Research Centre, Institute of Education, University of Warwick, Coventry CV4 7AL, United Kingdom (dave.pratt@warwick.ac.uk).

References


