23 Computers and applied constructivism

Uri Leron and Orit Hazzan
Technion—Israel Institute of Technology
Haifa, Israel

Abstract

In this paper we discuss and demonstrate the nature of computational learning environments which support and encourage learners' constructions in a way which is compatible with constructivist learning theory. We highlight the process of learning by successive refinement as a way for the human mind to cope with complexity, and the essential role the computer can play in this process. Three examples of constructivist environments (ISETL, Dynamic Geometry and Logo), are used to describe the dynamics of how the computer can facilitate the process of successive refinement, as well as to delineate issues of classroom culture, assessment and time.

Keywords

Cognition, constructivism, human computer interface, learning models, problem solving.

INTRODUCTION

There is a lot of talk about constructivism these days (e.g., von Glaserfeld, 1991; Tobin, 1993). There is less talk about how to go about actually constructing constructivist learning environments (Pirie and Kieren, 1992) and considerably less talk about constructing such computational learning environments. Our intended contribution will be on this last issue.

Constructivism has fallen out of fashion in many circles. One almost feels a need to apologise before publicly endorsing constructivist principles. But just as having been a fad didn't make it right (or good), being out of fashion doesn't make it wrong (or bad). In fact, constructivism is neither right nor wrong: like all complex tools, it all depends on how it is used. In our presentation, therefore, we

Information and Communications Technologies in School Mathematics
J.D. Tinsley & D.C. Johnson (Eds.)

would like to describe and analyse some ways of using computers to implement constructivist principles in the classroom. We will discuss the implementation of these methods, their strengths and their weaknesses. In addition, we will use the specific examples discussed as a basis for making some theoretical observations and distinctions.

LEARNING BY SUCCESSIVE REFINEMENT

A central tenet of the constructivist approach is that learners construct new knowledge by rearranging and refining their existing knowledge. It follows that new knowledge is constructed gradually. The construction is done in steps, each elaborating on preceding ones, though there may of course be retreats and blind alleys. We will refer to this process as learning by successive refinement. It is closely related to the Piagetian mechanisms of assimilation and accommodation, and has also been discussed extensively by Lakatos (1976) and Papert (1980). The term itself is borrowed from computer science, where it refers to a methodology for the gradual elaboration of complex programs (Dijkstra, 1972). What is common to all these sources is the realisation that successive refinement is an especially effective way for the human mind, with its particular strengths and limitations, to deal with complexity. Here are some more recent quotes:

> "Constructivism ... characterises the process of learning as the gradual recrafting of existing knowledge that, despite many intermediate difficulties, is eventually successful." (Smith *et al.*, 1993, p.123).

> "The goal of instruction should be not to exchange misconceptions for expert concepts but to provide the experiential basis for complex and gradual processes of conceptual change." (*ibid*, p.154).

The standard model of dealing with complexity advocates decomposing a topic into a linear sequence of tiny 'atoms,' then proceeding along the sequence, mastering the atoms one piece at a time. The model of successive refinement offers a viable alternative for dealing with complexity. One starts with a simplified version of the concept or phenomenon under study, and refines it successively to include more and more details, subtleties and precision. Through the entire process, the learner constantly deals with the whole picture, though it may be vague or imprecise in the intermediate stages (Leron, 1994).

In what follows we elaborate on the mechanism of learning a complex topic by successive refinement, with special emphasis on the central role the computer can play in this process. Our discussion is based on several kinds of constructivist computational environments. These environments consist of objects and operations that can be performed on them, as well as means for constructing new objects and operations. In addition, each environment supplies some 'language' for conducting the interaction between the learner and the machine. This can be a

formal programming language (as in ISETL or Logo) or it can be an iconic language using menu commands and mouse clicks and drags (as in Dynamic Geometry environments).

Examples

The above ideas, especially learning by successive refinement and the role of the computer in facilitating it, are illustrated by two extended examples. Space limitations allow us only a brief outline here. In both examples, students are facing a task of constructing a mathematical object, where the meaning of 'constructing' is determined by the software tools available. In both examples, students start with some initial intuitive familiarity with the object to be constructed, which gives them the resources for starting off with their first versions of the construction. Then the computational environment enables them to manipulate the resulting object and see if it behaves according to their expectations. If not, they can scrutinise their construction and modify it to better fit their expectations. Eventually, after several steps of successive refinement, they reach an expert version of the construction, one which withstands all their tests and is therefore declared (by the students themselves!) to be satisfactory.

In the first example, two students are working on the task of constructing a definition for prime numbers. This takes the form of a Boolean function *is_prime* in ISETL (Interactive SET Language), which tests whether a given number is prime. The students have some previous experience in programming in ISETL and a rudimentary knowledge of prime numbers and divisibility (for example, they know that 7 is prime but 6 is not because "it can be factored"). Programming this function requires the students to construct a formal definition, consisting of quantifiers, logical expressions and mathematical relations such as divisibility. The product of their construction is a function in ISETL, which may then be applied to some inputs to see if it works properly. Their final product might look something like the following code:

```
is_prime := func(n);
                return
                    not exists i | i in {2..n-1} | i divides n;
            end;
```

and their tests might look like:

```
   is_prime(7);
true;
   is_prime(6);
false;
```

Note the similarity of this code to standard mathematical notation. As a result, learning mathematics via programming in ISETL requires very little programming 'overhead'.

Our second example[1] concerns a geometrical construction task in a 'dynamic geometry environment' (such as *Cabri*, *Sketchpad* or *SuperSupposer*). In particular, we look at young students' attempts to draw a square using a mixture of visual and analytical means of construction. Thus, angles may be made to look like right angles and sides may be made to look like they have the same length, or these relationships can be formally constructed, using the construction tools supplied by the software. The squares thus constructed may all look the same, but when one of the vertices is dragged by the student, only those relationships that were formally defined are preserved. For example, if the only explicit construction is that side CD is parallel to (the opposite) side AB, and all the rest is done by sight, the student has actually constructed a trapezium which happened to look like a square. Dragging the various vertices by the mouse shows clearly to the student what has really been defined: the dragging preserves only the parallelism of AB and CD and destroys all other relationship, thus showing those other relationships to be just apparent in the original construction. This gives the students a powerful feed-back mechanism to reveal what is a square and what only looks like a square, thus enabling them to reach a precise construction by successive refinement. Simultaneously with the computer construction, students may observe connections between mathematical concepts and relationships between geometrical objects. For example, they may observe that in such environments it is easier to construct a trapezium than a square. This observation may lead the student to the conclusion that fewer assumptions are required for a figure to be a trapezium than to be a square. Such analysis may lead to mental constructions of relevant mathematical knowledge, such as that squares are a subfamily of trapeziums.

CONSTRUCTIVIST COMPUTATIONAL ENVIRONMENTS

We now ask, by way of reflecting on the above examples, what is it that makes some computational environments particularly conducive to learning by successive refinement. This will highlight the characteristics of constructivist computational environments in contrast to other computational environments on the one hand, and to non-computational learning environments on the other.

The theoretical background for our discussion is what Papert (1980) called Piagetian learning, which incorporates Piaget's learning theory of young children interacting with their natural environments, as well as Papert's own extension of the theory to include interactions within suitable computational environments (called microworlds). Our use of the term microworld (henceforth abbreviated MW) includes the social and pedagogical environment as well as the hardware and software, but not necessarily the 'learning without teaching' component of Papert's characterisation of Piagetian learning. We will have more to say in the next section on the teacher's role in such MWs.

Very briefly, Piagetian learning works as follows. Children come to know the environment by acting on it and by noticing the 'feedback' coming from the (physical or computational) constraints which are built into the environment. Based on these experiences they construct theories, mental models and explanations which help them make sense of the environment and predict its behaviour in relation to their future actions. In cases where the behaviour of the environment is observably different from their predictions, the children are moved to adjust their theories and explanations to account for the new experience. This acting-theorising-predicting-testing-refining cycle, which may repeat many times, is the basic mechanism for learning by successive refinement.

From the above theoretical model we can derive three essential characteristics of MWs, which are necessary for this kind of learning process to occur. Piaget's studies have shown that these properties are present by default in natural environments, but in designing computational environments they have to be purposefully built into the environment. We will see that the MWs in our examples all share these properties.

- The MW, like natural environments, embodies through its built-in constraints certain knowledge elements. These are accessible to the learner via the objects of the computational environments and the operations that can be performed on them.
- The MW enables learners to construct new objects and shows them the results of their constructions. The fundamental assumption behind the use of the MW as a constructivist learning environment is that while learners are intensely engaged in computational constructions for a non-trivial length of time, some parallel *mental* constructions are hardly avoidable.
- The MW enables operations on the constructed object to be performed by learners, through which they can compare the results of their construction with their initial expectations. This comparison may create the disequilibrium which serves as both the engine and the guiding mechanism for the next round of successive refinement.

We now examine how these requirements are met by the MWs mentioned in our examples.

In the ISETL example, the built-in functions and operations of the language, as well as its syntax, are very nearly those of standard mathematics (sets, functions, quantifiers and the usual operations on them). Our example shows how the learners construct by successive refinement a new object, the function *is_prime*, while at the same time refining their understanding of prime numbers and their ability to express mathematical ideas in formal language. The operation available on the constructed object is in this case the substitution of various numbers (such as 6 or 7) to see whether the function properly tests for primality in known cases.

In the Dynamic Geometry example, the MW embodies objects and operations of Euclidean geometry. The learners construct a new object, a square, while at the same time refining their understanding of squares, the relations holding between their parts, and their relations to other families of quadrilaterals such as parallelograms and trapeziums. The operation available on the constructed object in this case is mouse dragging, to see what it is that highlights those properties which have been mathematically defined.

> "Feedback given by the computer must be interpreted: an elaborate interpretation of the drag mode and of its effects is certainly constructed by the students through a long process made of interactions with various problem situations on the computer." (Laborde, 1986, p134).

We look briefly at one more example, the construction of a graphical object (say a house) by programming with the Logo turtle. What is the operation on the constructed object in this case which enables learners to test it against their expectations and refine the construction if necessary? We suggest that in this case it is not a computational operation (in most versions of Logo, the resulting picture is not a computational object that cannot be operated upon), but a mental one: it is the comparison of the graphical image on the screen with the child's own mental image of the picture as he or she intended it to look like (Leron, 1986).

We elaborate a bit more on the specific part played by the computer and on its unique contribution to the constructivist process of learning by successive refinement. This contribution comes from the tangible and manipulable feedback given by the computer to the learner's input (the first two MW properties in the list given above). This feedback, can be a very powerful learning mechanism. In many mathematical activities it can achieve results that are hard to imagine without the computer. This is largely due to two specific capabilities of the computer. First, its ability to translate between representations, as in digital to analog conversion in turtle programming (Leron, 1986), or its ability to make formal mathematical expressions executable as in ISETL. Secondly, the consistent and non-judgmental nature of the feedback (Sfard and Leron, 1996). This is quite different from a feedback given by a human teacher, no matter how skilful. The computer's feedback doesn't judge or direct, it only shows learners the outcome of their constructions. In other words, it shows the *consequence* not the *value* (or correctness) of their work.

IMPLEMENTATION NOTES

Is this discovery learning?

Piagetian learning, i.e., the kind of learning by successive refinement in a MW that we have been describing, is different in important ways from discovery learning, because learners are not set out to 'discover' anything in particular. The purpose of the activity is to create an experiential basis for the later stages. For

this purpose, it matters little whether the learner has actually discovered the intended knowledge, or only experienced the situation in a way that created an initial familiarity. When 'answers' start coming in (from classmates or the teacher) they are likely to be rendered meaningful because of this initial experience.

Is this Socratic learning?

Piagetian learning is also different from learning by a Socratic dialogue. Here we differ from Arcavi and Shoenfeld (1992) who use a form of Socratic dialogue as their interpretation of constructivist learning. While it is clear that humans are far superior to computers in giving students empathy, emotional support and getting them unstuck when necessary, we claim that computational MWs are superior as feedback mechanism for the process of successive refinement. As noted above, the advantage of computational feedback lies exactly in its formal (i.e., non-human) interpretation of learners input, exhibiting in a consistent and neutral way the consequences of their constructions. It is our belief (and experience) that much of the *problematic* reported by Arcavi and Schoenfeld is alleviated when learners do not depend on a teacher for direct and regular interpretation of the their actions.

The social aspect

Papert (1980) said that "children learn best by doing and then reflecting on what they did". While computers enable a wide range of doing, it is the social context that enables most of the reflecting. Learners reflect as they talk about their activities, first in their teams during the activities, and then when the various insights are shared in the 'plenary' classroom discussion.

What activities?

Since we are interested in encouraging students' constructions, and not chiefly the finding of correct answers (these will gradually emerge as the process unfolds), it follows that the activities should be centred around open-ended problems, which allow exploring, conjecturing and testing on many levels and in many directions. The 'didactic contract' is for variety and richness of conjectures, explanations and insights rather than for getting 'the right answer'. In as much as it is desirable to also exhibit some standard mathematics (definitions, procedures, theorems, proofs), it is possible to let this emerge out of the activities and the discussion which follows. Successful activities are both engaging and mathematically rich. The creation of such activities, as well as encouraging and maintaining an atmosphere of openness and curiosity in the classroom, are the most demanding and at the same time the most rewarding parts of the teacher's role.

SOME DIFFICULTIES

We conclude by listing briefly some of the difficulties entailed in general by teaching towards understanding and conceptual development, and in particular, by the kind of teaching advocated in this paper.

- *Classroom culture:* First and foremost is the difficulty of adopting a different view of the teacher's role in such an environment. The teacher is no more the conveyor of knowledge but rather a creator of opportunity for action and reflection by the learners. This requires more flexibility and self confidence than conventional teaching.
- *Assessment issues:* If knowledge is constructed gradually over a long period, how do we evaluate partial knowledge in intermediate stages? Can we find ways of assessment that will show students' achievements in intermediate stages, rather than their deficiencies?
- *Time:* Learning via interaction in a MW is a lengthy process. A teacher cannot simply plan to 'cover' a certain amount of 'material' in a given amount of time. One must adopt an alternative view that knowledge and understanding emerge gradually over time and learn to value the process itself and the partial knowledge exhibited in intermediate stages.

Note

1 This example is taken from joint work with Paul Goldenburg, Educational Development Centre, Newton, Massachusets, USA.

REFERENCES

Arcavi, A. and Shoenfeld, A. (1992). Multiple Sense-Making: Tutoring Through a Constructivist Lens. Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco, CA.

Dijkstra, E. (1972). Notes on structured programming. In O. Dahl, C. Hoare and E. Dijkstra (eds.), *Structured Programming*. Academic Press, New-York.

Laborde, C. (1992). Solving problem in computer based geometry environments: The influence of the features of the software, *ZDM* **4**, 128–135.

Lakatos, I. (1976). *Proofs and Refutations*. Cambridge: Cambridge University Press.

Leron, U. (1986). The computer as mediator between analog and digital thinking, *Proceedings of the Tenth International Conference for the Psychology of Mathematics Education*, London.

Leron, U. (1994). Can we follow what we preach: Teaching according to constructivist principles. CMESG, Regina, Sask, Canada.

Papert, S. (1980). *Mindstorms—Children, Computers, and Powerful Ideas*, Basic Books, Inc. Publishers, New-York.

Pirie, S. and Kieren, T. (1992). Creating constructivist environments and constructing creative mathematics. *Educational Studies in Mathematics* **23**, 508–528.

Sfard, A. and Leron, U. (1996). Just Give Me a Computer and I Will Move the Earth: Programming as a Catalyst of a Cultural Revolution in the Mathematics Classroom. *International Journal for Computers in Math Learning* **1**(2), 189–195.

Smith, J., diSessa, A. and Roschelle, J. (1993). Misconceptions reconceived: A constructivist analysis of knowledge in transition. *The Journal of the Learning Sciences*, **3**(2), 115–163.

Tobin, K. (ed.) (1993). *The Practice of Constructivism in Science Education*. Lawrence Erlbaum Associates, Inc., Publishers, NJ.

von Glasersfeld, E. (ed.) (1991). *Radical Constructivism in Mathematics Education*. Dordrecht: Kluwer Academic Publishers.

Uri Leron received his Ph.D. in mathematics from the Hebrew University of Jerusalem in 1972, and has continued his research in ring theory at the University of Oregon, UCLA and the Israel Institute of Technology. Since 1980, he has moved to math and computer science education, with special interest in the teaching and learning of proofs and in the use of technology to enhance a more meaningful learning. He is the Current Executive Editor of the International Journal of Computers for Mathematical Learning, and the holder of the Churchill Family Chair of Education in Technology and Science at the Israel Institute of Technology.

Orit Hazzan received her Ph.D. degree from the Technion—Israel Institute of Technology in 1995. Her thesis is concerned with how undergraduate students understand concepts in Abstract Algebra. Her post-doctoral work in Education Development Center (Newton, MA) continued her research in undergraduate mathematics education. In addition, she has worked on the development of an epistemology involved in learning with Dynamic Geometry environments. Her current interests are the role of constructivist computational environments in the learning of mathematics in particular, and ways by which Information Technologies may enhance the learning of sciences in general.