EXPLORING STUDENTS' UNDERSTANDING OF THE RELATIONSHIP BETWEEN RECURSION AND ITERATION

Mojtaba Ammari-Allahyari

Institute of Education, University of Warwick

In this study, I examine students' appreciation of the relationship between recursion and iteration, including how they perceive the intra-relationship of the components within those processes. In this environment, the students model trees and fractal-shape objects. These early results show that having a clear understanding of the relation between recursion and iteration and of the flow of control are important in understanding the recursion itself. In addition, functional abstraction is a key concept in dealing with recursion. In the light of these results, I am planning to open up the software such that students will be able to engage with recursion at two levels, namely "functionality" and "functionnings".

INTRODUCION

The relations between mathematics and the real world have always been central in mathematics education. Perhaps the main reasons for this fact is that almost all problems concerning human learning and teaching of mathematics affect and are affected by relations between mathematics and the real world. Therefore it seems that emphasizing the functional aspect of mathematical knowledge in different situations and contexts is very important. In this research I have been trying to figure out the role and effects of using a LOGO-based environment to investigate students' difficulties in dealing with recursion through the modelling of trees and fractal-shape objects.

WHAT IS RECURSION?

In recent years, a number of researchers in mathematics and computer science have tried to provide a satisfactory and convincing definition of the concept of recursion. Unfortunately we do not have a comprehensive and perfect definition for recursion yet. Harvey (1997) regards recursion as involving a procedure that calls itself. In its simplest form, recursion is a process or a function, which is able to re-call itself, or use itself as its sub-procedure. Each recursive process contains two crucial parts: base case(s) and recursive call(s). The base case is the trivial part of the original problem and the recursive call is that part of the procedure, which shifts the flow of the control to the original procedure and suspends the rest of the commands in the procedure. Haberman & Averbuch (2002) studied base cases as an essential component of the concept of recursion from two different stances. Base cases are the smallest instances (declarative approach) and Base cases are the stopping condition (procedural approach). In a declarative approach, the smallest instance of the problem is one, for which we know the answer immediately, without any effort. A procedural approach represents the end of decomposing the problem into the smaller similar problems. In

Logo, recursion can be categorized into two major kinds as follows: 1-Tail recursion, 2-Embedded recursion. The recursive call in tail recursion appears in the last line of the procedure while in embedded recursion, recursive call(s) will appear perhaps multiply, in the middle of the procedure.

UNDERSTANDING RECURSION

Many students find recursion difficult to understand and to apply in their problem solving activities (Harvey & Wright, 1993; Segal, 1995). There are some conceptual obstacles, which have been explained by Pirolli & Anderson (1985), Kurland & Pea (1985), Wiedenbeck (1988), Anazi & Uesato (1983), and Sooriamurthi (2002). Firstly, there is a lack of everyday life analogies for recursion. It is really hard to find examples of recursion in our everyday life. Secondly, there is a difficulty in tracking the flow of control between levels of recursion. The related third problem is the concept of functional abstraction. The functional abstraction doctrine basically is that, the way something works is not necessarily the same way that we have to work with it. Sooriamurthi (2002) stated that one of the major student's difficulties arises from focusing on the *how*. In contrast he believes that they should focus on *what* first, and then focus on *how*. By functional abstraction, I refer to the "separation of what needs to be done from how it will be done" (Sooriamurthi, 2002). Finally there appears to be a difficulty in distinguishing recursion situations from the more familiar iterative problems. The main focus of this article is on the mutual effects of iteration and recursion. Perhaps one of the most challenging aspect of understanding recursion lies in blending this concept with that of iteration. In the process of performing an iteration procedure, each step is completely worked out before the next step, in order to solve a bigger problem. An iteration process can be considered as an accumulative process, one stage starts after its previous stage ends. In contrast, recursion is a process where one procedure (as a sub-procedure of itself) begins, and ends before its previous procedure ends.

A number of researchers worked on the influence of learning iteration on the subsequent learning of recursion and its converse. Anazi & Uesato (1983) reported how 88 students studied the factorial function in an iterative version and a recursive version. They concluded, "Recursive procedures may be acquired based on learning of the corresponding iterative procedure" (p. 100). Their work was in fact focused on the factorial function as a mathematical definition rather than as a computer program. Wiedenbeck (1988) criticized the results of their study by arguing that they only concentrated on these aspects: iterative-recursive and recursive-iterative. Wiedenbeck extended that study by adding iterative-iterative and recursive-recursive groups. Wiedenbeck (1988) believed that knowing how recursion and iteration together influence each other is central to finding out new pedagogic. She repeated the task again but this time by focusing on a computer program aspect of factorial function instead of mathematical definitions. Her findings were 1) having previous experiences with the iteration could facilitate understanding of recursion, to some extent; 2) Learning by examples is quite effective. Further support for Wiedenbeck's results has been presented in Kessler & Anderson (1986). They focused on transferring skills between performing iterative and recursive procedures. They concluded that although writing procedures on iterative and recursive functions does not facilitate writing procedures on recursion functions but having prior experience on similar iterative procedures enabled increased sophistication in dealing with the "flow of control" which is needed for understanding recursion. Ginat & Shifroni (1999) argued that discovering iteration is much easier than a recursion. They also explained that, although decomposition of a problem into its sub-problems of the same kind is logically coherent, it is not easily understood by learners to perform.

Previous studies focused mainly on a pre-prepared recursive procedures or a mathematical function such as factorial function and the subjects were tested to see if they were able to figure out the values of the given function recursively (Anazi & Uesato, 1982; Wiedenbeck 1988) or if they were able to recognize the recursive structure (Kurland & Pea, 1985; Sooriamurthi, 2001). In addition, Textbooks explain what recursion is, how it works and also they give examples of recursive functions but they never explain how to go from a problem specification to a recursive function. Although the same is true for iteration, but the students' prior experiments in real life facilitates dealing with iterative processes. (Pirolli et al, 1988)

In contrast, my focus is on real world phenomena such as trees or fractal-shape objects, which can only be described recursively through using the Logo-based microworld environment. My aim was to explore students' understanding of iteration and recursion as it evolves within interactive computational environment. In light of the above, the research questions are as follows: 1-How do modelling of trees and fractal-shape objects in Logo environment shape the construction of the concept of recursion? 2- How do having the prior experiments on iteration influence understanding recursion?

METHOD

This research adheres to a design-based methodology, which is based on a cycle of designing, testing, analysing, and modifying (Cobb, diSessa, 2003; Barab and Squire, 2004). Theoretically, the study is based on a constructionist paradigm (Papert, 1980). I report on the first iteration of a design based research approach for creation of a Logo-based microworld called TREEMENDERS. In this environment, I intended to facilitate the problematic aspects of the recursion concept by modelling of trees and fractal shape objects and I speculate on possible education benefits. The study carried out with five volunteered students, one first year computer science student, group of two first year mathematics students, and a group of two third year computer science students. Although much of my approach was based around the use of a Logo-based microworld, I also carried out some initial paper and pencil tasks through semi-structured interviews. I took part as a participant observer. The interviews were audio taped. Each of the interview sessions lasted about one hour. Tapes and written works of students plus transcriptions were treated as data and they were analysed. The microworld put the students in the situation where they can play with recursion freely and at the same time they were able to model their own trees. The students were both in positions of apprenticeship and tutor (Lave & Wenger, 1991). It gives opportunity to the students to work with a recursive procedure interactively and produce their own trees recursively. In the first version of this software students were able to change the size of the branches and also the size of new branches which will be created as a result of the procedure, besides they were able to change the angle of branching to the either left or right sides. Another thing which has been predicted in the tool and students were expected to work with was the stopping condition in terms of the size of new branches. Students were required to generate their own model of trees by having control over various sliders, which implicitly reflect the cornerstones of recursion.

FINDINGS

During the pen and paper tasks, most of the students used iterative methods rather than recursive. In the task on the Koch curve, I asked, how the different levels of the Koch curve are related to each other?

Sarah: Each level becomes the one of the parts of the others. Umm, level one *repeating* four times to build the structure of the level two and level two is going to build level three and so on.

Jin added: If we call level one X, the level four will be *4 times X* and the other levels can be constructed similarly.

Feng: It is a For-Loop.

Although when I asked him, is he able to calculate the factorial of a natural number, He immediately responded "Factorial of a natural number is a recursive process."

Based on his prior knowledge in real life, he was looking for an iterative way of describing the Koch curve by counting the vertices or implementations of a For-loop, which was clearly an iterative strategy. All descriptions appeared to tend an iterative description for the Koch curve. And particularly Feng pointed to one of the familiar Loop commands. In contrast Koroush (the third year Computer science student) is response to the same questions gave me a good description of recursion because of previous familiarity with it.

Koroush: You start with the first level and reproduce the whole thing on each of the smaller segments. And on the each edge you did the same.

And when I asked him for programming of it he answered as follows.

Koroush: Start with the straight line and split it into three parts recursively repeat it into new parts.

His response seemed to suggest that the prior experiments could be helpful to use and apply the recursion concept in problem solving situations.

In working with paper and pencil tasks I gave them two photos of two different trees, one of them with a big trunk and then branches and the other one has no main trunk. I asked them to describe them for me.

> Sarah: I think this one is a little bit harder to program (She points to the photograph with no main trunk[1]), because the first one has a trunk and then branches, whereas in this photo we have not got trunk and just branching.

It is interesting that Sarah responded my question by explaining the starting point, with a main trunk or with branches. It seemed to suggest the role of base case in dealing with recursion and its relationship with iterative processes. Another third year student Yasaman also gave an interesting separation for iteration and recursion.

> Yasaman: Basically, *recursion is different with iteration because of the base case*. In recursion we start the process and then we get to the base case at the end. But in recursion we start with base case.

During work with TREEMENDERS it became visible that after doing some trials and errors the subjects gradually were getting familiar with the concept of base case as a stopping condition (Procedural view). Perhaps, because of their prior knowledge and experiments with Loop-commands and necessity of a stopping condition in loop-commands for avoiding infinite loop. This is one of the points that I am going to explore in more details in the next iterations. In addition, throughout working with TREEMENDERS the students appeared to develop an understanding of the relationship between the procedural aspect of base case and having control over the number of calling of the recursive calls.

> Koroush: …I want to get more iteration, because it gives us to get more branches.

Feng was looking for more branches from another perspective. He started to work with the slider of recursive call rather than stopping condition slider. He kept the stopping condition fixed and increased and decreased the size of new branches in the recursive calls but there was no difference in the number of new branches. Soon he realized that the reason for that was keeping fixed the stopping condition slider. He continued with changing the stopping condition slider and after some trials and errors he increased the number of new branches. He was also curious to see why in each recursive call we will have only two new branches and he answered as follows:

> Feng: … Because there are only two recursive calls.

It seemed that work with the microworld provided an environment in which he could find some new ideas on recursion. During pen and paper tasks Feng believed that recursion was a For-Loop. After his work with TREEMENDERS, I asked:

> Me: Do you still think that recursion is a For-Loop?
>
> Feng: No, recursion is a While-loop… here each time we are changing the size of the new branches, so it should be While-loop rather than For-loop.

[1] The text inside of the brackets is added by me

For me, his answer is quite promising for the further work on the microworld. Because his answer shows as a result of working with TREEMENDERS he has found that there are some differences between recursion and For-Loop. Nevertheless, he appeared to have some fundamental problems.

CONCLUSION

I think these results which are grasped by the subjects throughout using the microworld are very promising results for the future work in the next iterations. Although these results at this stage seemed to be rough to some extent but reaching them by using pen and paper tasks is really difficult if not impossible. In the next iterations I would concentrate on these invisible aspects of the recursion to make them more visible in the microworld environment. These early results show that having a clear understanding of the relation between recursion and iteration and the flow of control is important in understanding the recursion itself. In addition, functional abstraction is a key concept in dealing with recursion. In the light of these results, I am planning to open up the software such that students will be able to engage with recursion at two levels, namely "functionality" and "functionnings". It means that, the students can switch in their thinking between functionality of recursion and functionnings (mechanistic) level of recursion.

REFERENCES

Anazi, Y. & Uesato, Y.: 1982, Is Recursive Computation difficult to Learn?, PA: Psychology Department, Carnegie-Mellon University, Pittsburg.

Cobb, P., diSessa, A.: 2003, "Design Experiments in Educational Research", Educational Researcher, Vol. 32, No. 1, PP. 9-13.

Ginat D. & Shifroni E.: 1999, "Teaching Recursion in a Procedural Environmental- How much should we emphasize the Computing Model?", Technical Symposium on Computer Science Education, pp. 127-131.

Haberman, B., Averbuch, H.: 2002, "The case of Base Cases: Why are They so Difficult to Recognize? Student Difficulties with Recursion", ITiCSE'02, June 24-26, Aarhus, Denmark, pp. 84-88.

Harvey, B.: 1997, Iteration, Control Structures, Extensibility. Computer Science Logo Style, volume 2: Advanced Techniques 2/e, MIT Press.

Harvey, B. & Wright, M.: 1993, Simply Scheme, introducing computer science, University of California Press.

Kessler, C. M. & Anderson J. R.: 1986, "Learning flow of control: Recursive and iterative Procedures", Human-Computer Interaction, Vol. 2, pp. 135-166.

Kurland, D. M. & Pea, R. D.: 1983, "Children's mental model of recursion LOGO programs", Proceedings of the 5th Annual Conference of the Cognitive Science Society, Rochester, NY, pp. 1-5.

Papert, S.: 1980, Mindstorms, New York: Basic Books.

Sooriamurthi, R.: 2001, "Problems in comprehending recursion and suggested solutions", Proceedings of the 6th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, pp. 25-28.

Wiedenbeck, S.: 1988, "Learning Recursion As a concept and As a Programming Technique", Proceedings of the 19th SIGCSE technical symposium on computer science education, pp. 275-278.