COMPUTING IN SCHOOLS

Michal Armoni

Teaching CS in Kindergarten:

How Early Can the Pipeline Begin?

IN AN INTERESTING TALK, given in the last ITiCSE conference [2], Paul Gibson described a unique outreach initiative that starts as early as kindergarten, with children as young as five years old. This ambitious program touches theoretical abstract computer science (CS) concepts such as graph connectivity and graph isomorphism. While a methodological study on the effectiveness of this program is yet to be conducted, Gibson's impressions indicate that even at this early age, children are capable of working with abstractions and use computational reasoning. This project continues a series of projects introducing formal computing methods [3] and programming (in Java) [4] to children of various ages, though in previous projects the youngest children were seven years old.

The rationale and motivation behind the development of such programs is clear. It is well known that enrollment to undergraduate CS studies does not meet the requirements of the job market, and that among other reasons affecting students' decision not to major in CS are students' misconceptions regarding the nature of CS and of the work in CS and their negative attitudes towards CS. Therefore, the recommendation is to expose students to CS before they form these misconceptions and attitudes, and before decisions affecting enrollment are taken. Many argue that high school is too late for such an exposure and indeed many attempts are reported, which start at junior high school and even primary school.

So why not start earlier?

I don't have an answer for this question, but I do believe it is not a trivial question and one that deserves some serious thinking. Of course, a straightforward answer to this question would be:

Why not, as long as no harm is done?

I am not sure we can automatically conclude that no harm is done, not without deeply considering it, and I believe that we should also consider the issue of effectiveness, that is – whether any good comes out of it.

An inherent component of any discussion of or introduction to CS is abstraction. It is always there, to some extent. The basic idea of a solution to a problem encapsulates abstraction in it, since such a solution is always universal, one solution for all possible inputs. The simplest instructions in any programming language encapsulate abstraction, since such instructions are actually patterns that become specific instructions once values are incorporated into it. As noted above, in the project presented by Gibson the children are introduced to concepts in graph theory, which are no doubt abstract concepts.

Can very young children understand abstraction, even in its basic forms?

According to Piaget, they cannot, not in its real sense [5]. Before the age of seven, children are at the preoperational stage of their cognitive development. Piaget noted that children in this stage do not yet understand concrete logic, and cannot manipulate information mentally, only physically. Even older children (seven to eleven years of age) who are at the concrete operational stage, can only solve problems that apply to actual (concrete) objects or events, and not abstract concepts or hypothetical tasks. At the concrete operational stage, the child develops an ability to think abstractly and rationally, but only about concrete or observable phenomena.

Bruner [1] argued for spiral teaching, that is, fundamental ideas and central concepts should be revisited again and again throughout the curriculum, but at each age they should be taught at a level that corresponds to the current developmental stage of the students. Following Bruner's

Even older children (seven to eleven years of age) who are at the concrete operational stage, can only solve problems that apply to actual (concrete) objects or events, and not abstract concepts or hypothetical tasks.

recommendations, perhaps these abstract ideas can be taught to younger children, if connected to concrete objects and if (at the earlier ages of five through seven) information is manipulated physically.

Actually, this is exactly the didactic approach in Gibson's projects. Students search and sort bits of strings; they reason on primality by organizing sweets in rectangles, they use sticky colored bricks to reason about parity. They "prove" properties by looking at specific concrete cases. We are familiar with this approach from math education, where concrete objects like different kinds of blocks are used to teach very young children about the abstract concepts of numbers and number operations. Of course, we should keep in mind that according to Bruner's framework, the educational goal is not to teach children how to add and subtract blocks, or how to sort bits of strings, or how to check if sweets can be organized in rectangles. The goal is that this concrete knowledge will in due time evolve or transfer to more general and abstract contexts. So, let us try to look at math education research regarding this approach, and perhaps we can import from their body of knowledge into CS education.

In a very interesting essay, included in a book on conceptual (abstract) and procedural (concrete) knowledge of mathematics, Schoenfeld [6] discussed teaching of abstract concepts using concrete reference systems, such as concrete block systems for teaching numbers and number operations. Schoenfeld points at a few factors and obstacles that complicate such didactic learning processes. Here are three of these.

- First, the more natural a representation is, the harder it might be to abstract the underlying ideas.
- Second, in most cases, the concrete world does not completely map into the abstract world, which might cause confusion, difficulties, and even misconceptions.
- Third, children fail to connect the concrete world and the "real" abstract world. In a sense, for them the two worlds can live side by side, with no connection between them.

For example, they might try to take actions in the concrete world (like doing a certain geometrical construction) ignoring results that they know and understand in the abstract world (of Euclidean geometry). They will "prove" correctness of procedures performed in the concrete world, by looking at concrete examples and concrete features (like accuracy of drawing), ignoring proof tools they are familiar with in the abstract world.

Teaching abstraction in early ages necessitates concrete reference systems. Even if the children are capable of reasoning mentally in these concrete systems, which can probably happen only if they are at least seven years old, the transfer to the abstract world, which is the ultimate teaching goal, is questionable. In addition, if some transfer is achieved, but it carries along with it some inaccuracies of the concrete representation, it may result in confusion and misconceptions.

This discussion is by no means thorough or exhaustive, and it brings no bottom line regarding a question like "teaching CS in kindergarten – good or bad?" My objective here is to emphasize that there are uncertainties, and many question marks, and that these questions deserve deep consideration, with theoretical and empirical treatment. **Ir**

References

[1] Bruner, J. S. (1960). The Process of Education. Cambridge, MA : Harvard University Press.

[2] Gibson, J. P. (2012). Teaching graph algorithms to children of all ages. In *Proceedings of the 17th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE'12), July 3–5, 2012, Haifa, Israel. T. Lapidot, J. Gal-Ezer, M. E. Caspersen, and O. Hazzan, Eds. New York: ACM Press, 34-39.

[3] Gibson. J. P. (2008). Formal methods — never too young to start. In Z. Istenes, Ed., *Proceedings of Formal Methods in Computer Science Education* (FORMED 2008), Budapest, Hungary, 151–160.

[4] Gibson. J. P. (2003). A noughts and crosses Java applet to teach programming to primary school children. In *Proceedings of the 2nd International Symposium on Principles and Practice of Programming in Java* (PPPJ 2003), volume 42 of ACM International Conference Proceeding Series, Kilkenny City, Ireland. J.F. Power and J. Waldron, Eds. New York: ACM Press, 85–88.

[5] Santrock, J. W. (2004). Life-Span Development (9th Ed.). Boston, MA: McGraw-Hill College - Chapter 8.

[6] Schoenfeld, A. H. (1986). On having and using geometric knowledge. In J. Hiebert, Ed., Conceptual and Procedural Knowledge: The Case of Mathematics. Hillsdale, NJ: Lawrence Erlbaum, 225-264.

Michal Armoni
Weizmann Institute of Science
Rehovot 76284 Israel
Michal.Armoni@weizmann.ac.il

DOI: 10.1145/2381083.2381091