# COMPUTING IN SCHOOLS

Michal Armoni

## On Teaching Topics in

# Computer Science Theory

**THIS COLUMN,** as its title suggests, is devoted to teaching computer science in schools. By computer science, I do not mean computer literacy or computer fluency, and not even just programming. Rather, I talk about teaching the scientific discipline of computer science. By schools, I mean mainly (but not solely) high schools and junior high schools. This is a broad area, and in this column, I will try to touch upon various issues such as …

- Why should we teach CS in schools?
- Is it possible to teach CS to young, sometimes very young students?
- What should be taught and how?

This column intends to be international and written from an international, global perspective, telling also about various computer science programs in schools around the world.

Since this is the first column, I would like to talk about an issue with which I was and still am personally involved and for which I deeply care:

Teaching topics of the theory of CS to high school students.

When thinking about teaching CS to young students, many would not consider the theory of CS as an obvious choice. In fact, in more than one occasion, CS educators shared with me their views that these topics are not teachable to young students. Well, our own experience in my country proves otherwise. A course that deals with the theory of CS is a part of our CS high school program for more than ten years. Our experience shows that high school students can not only deal with the concepts and ideas of the theory of CS in a meaningful way, but many of them also find them challenging and interesting, sometimes even refreshing. I do not believe the case should be different in other countries. Therefore, it can be done, and the two obvious, important questions are, therefore, – why and how?

**Since CS is much more than programming, the introduction should touch various areas, characterized by various skills and thinking patterns.**

Let me start with the fundamental "why" question. After answering this question, it will be easier to discuss the second one. The decision to include topics of the theory of CS in our high school CS program was taken by the committee that designed this program [1]. In a nutshell, the goal of a high school CS program, as seen by this committee, is not to produce computing professionals; rather, it is to introduce a scientific discipline, in its depth and breadth, to the young students. Since CS is much more than programming, the introduction should touch various areas, characterized by various skills and thinking patterns.

The theory of CS serves as an example for an area that involves formal, rigorous thinking patterns, mathematical working habits (such as rigorous definitions and formal proofs), an area that does not focus on design (like algorithmic design, system design, logic design or programming), but rather on asking questions about designs and their expression abilities. It serves as an example of a meta-discussion, that examines algorithmic problems and theirs solutions from above, and aims at characterizing them and thus at a better understanding of their power and limitations. The goal is to enable the students a glimpse to the work of theoretical computer scientists: What kind of questions do they ask themselves? Which tools they use? What skills do they need and which thinking patterns do they employ?

Some students are disappointed in discovering that CS involves such material and is more than programming, while others are delighted in and motivated by discovering that it is so. For all of them, a proper introduction of this discipline, representing all the colors of its rainbow, is important and just.

Once this goal is clear, the "how" question can be dealt with, but not before we think for a moment about another important question – "what?". What should we teach in order to achieve such an introduction that induces meaningful learning?

Since the goal is to introduce an area that reasons about design rather than doing designs, it follows that this should not be a design course. It is not about designing automata of various kinds. After

## Computing In Schools

*continued*

all, constructing automata is not much different from designing an algorithm or a program. The language is probably much more primitive than the programming languages with which we usually work; yet, essentially constructing an automaton is like programming a solution to a problem. The students experience with design in other parts of the CS high school program, algorithmic design and program design in various paradigms.

This course is not a design course. It is an opportunity to introduce other aspects of CS. It is an opportunity to expose students to abstraction (e.g., through definitions of abstract models), abstract flexibility and the ability for abstract generalization (e.g., by changing or generalizing definitions of computational models, changes which are not motivated just by modeling "real-life" situations, but sometimes by scientific theoretical "curiosity"), and the ability to reason about abstract objects. It is also an opportunity to introduce students to some general important CS concepts and terms related to formal language theory such as nondeterminism.

This can be done, but how? After all, in the college and university level this course it has a (not unjustified) reputation of an advanced, difficult, and challenging course. How can we adapt it to the high school level, keeping the principles mentioned above? I will elaborate on the "how" question in the next column. Meanwhile, I hope I succeeded in convincing at least some readers of the importance of teaching these topics to high school students and of the interesting challenge it bears for high school CS educators and curricular designers. **Ir**

**References**

[1] Gal-Ezer, J., Beeri, C., Harel, D. and Yehudai, A. A High School Program in Computer Science, Computer, Vol. 28, No. 10, 1995, pp. 73-80.

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

**DOI:** 10.1145/1721933.1721941



# CS EDUCATION RESEARCH

Raymond Lister

## The Naughties in CSEd Research:

# A Retrospective

**THIS PARTICULAR COLUMN** on CSEd Research is both the first in the new form of the *ACM Inroads*, and the first of a new decade. It therefore seemed appropriate to reflect on CSEd Research in the years 2000–2009 (the "naughties"), and to speculate a little on the coming decade (the "teenies"). My goal is to convey some subjective impressions, and not to review the last decade systematically. The publications I cite are to illustrate a point, and those publications are most certainly not my personal "hit pick" of the decade.

I would like to see more research work that focuses on the drop in student numbers as an inter-generational issue.

### Recruitment

The very first year of the decade was a heady time. Student numbers were booming. In those days, I do not recall many people talking about how to attract even more students to computer science. Some of us complained about there being too many students, and as the old saying goes, *"Be careful what you wish for, as you may get it."* By SIGCSE 2001, the Tech Crash had started, and I recall watching television in my hotel room, shocked to see even Cisco stock falling. Even then, few of us predicted a precipitous drop in student numbers.

At SIGCSE 2003, I recall the Saturday lunch, which was the book launch of *"Unlocking the Clubhouse"* (Margolis and Fisher, 2003). That book has since become an influential study of women in computing. I assume that Margolis and Fisher started their work before the drop in student numbers, and my impression is that the research work on recruitment in the naughties has largely been a continuation of the work on why women tend to stay away from computing. Now, women in computing is a worthwhile research area in its own right, and I suspect that anything that attracts women to computing will also attract minorities, and even attract more white males. However, I would like to see more research work that focuses on the drop in student numbers as an inter-generational issue. There has been some work on that front, such as that by Sarita Yardi (2007), but I would like to see more. Furthermore, I'd like to see more studies of both sides of the generational divide – not just studies of students, but studies of the teachers. A recent blog entry by Mark Guzdial (2009) illustrates eloquently why we need to study ourselves as much as study our stu-