Teaching Theoretical Computer Science to Would-Be Hackers

Hans J. Schneider
Lehrtsuhl für Programmiersprachen, Universität Erlangen-Nürnberg,
Martensstraße 3, D-91058 Erlangen
Electronic Mail: `schneider@informatik.uni-erlangen.de`

Entering mathematics in 1956, I enjoyed the constructive approach of Bourbakism that proceeds from simple structures to more complicated ones by applying a small set of design principles again and again. Computer science is even more suited to be taught in such a way since it is constructive by its objectives: Its ultimate goal is to build tools of some kind or other, may be in hardware or in software.

A usual programming course, however, is mainly concerned with presenting syntactic sugar and semantic tricks to solve more or less small, standardized problems. In the end, the students know a lot of details how to make a computer run, and at the best, they are familiar with good design practice and some software-engineering tools. But, do they understand what computer science is and what it can bring about? To remedy this, they need not only be taught the theory, but also be supplied with the relationship between fundamentals and practice.

The main task of both programming and theory is making definitions, proving properties of the objects introduced by the definitions, and taking advantage of these properties. We can derive profit from this analogy by teaching theory from a programmer's point of view. A topic, which is very closely related to programming and at the same time, is at the heart of theoretical computer science, is computability theory. Therefore, it is well-suited as an entrance to theory. Even freshmen perceive the clarity of Kleene's theory of recursive functions if the definitions are translated one-to-one into higher-order functions of about five lines using some LISP-dialect. The students can then apply these functions to create computable functions one after the other, make them run and can convince themselves that the theory works. It is nice to see that even hackers can be tempted into studying the theory of recursive functions, efficiency problems, denotational semantics, etc., if they can do this sitting in front of their beloved computers. I have never understood why lectures in theoretical computer science must be difficult to follow and restricted to using paper and pencil.

Contribution to the Millenium Issue of the Educational-Matters Column in the Bulletin of the European Association for Theoretical Computer Science (EATCS)
(Bulletin EATCS vol. 70, pp. 164-166, Feb. 2000)