Theoretical Issues in Computer Engineering

Feodor Vainstein, Mark Rajai
Georgia Institute of Technology/ Northern Kentucky University

Abstract

In this paper we present our experience on teaching Theoretical Issues in Computer Engineering. We emphasize on the importance of abstract algebra for practical applications. In particular we show that algebra is the appropriate mathematical tool for many problems in minimization, fault tolerance, digital communications, random number generators etc.

Introduction

Computer engineering has been formed as a distinct discipline only recently. It is significantly different from traditional engineering areas like mechanical, chemical or electrical engineering. By its nature, computer engineering is discrete and structural.

We believe that the maturity of a research area is mostly defined by the level at which mathematics is used in this area. There is an opinion that mathematics is 300 years ahead of other sciences, since most mathematical tools that are used now have been developed in 17th century. Computer engineering to some extent challenges this point of view. It is only 100-200 years behind mathematics (in some cases the gap is only 50 years wide).

In this paper, we present our experience on teaching Theoretical Issues in Computer Engineering. We emphasize on the importance of abstract algebra for practical applications. In particular we shall show that algebra is the appropriate mathematical tool for many problems in minimization, fault tolerance, digital communications, random number generators etc. Some textbooks on discrete mathematics cover the abstract algebra very shallow. There can be two explanations of this. First, it is believed that this subject is too tough for an engineering student. Second, many authors, being professional mathematicians, are not aware of all numerous applications of algebra in computer engineering and do not realize importance of teaching it. We strongly disagree with these points of view.

We give the list of topics that should be, in our opinion, covered in a course on Theoretical Issues in Computer Engineering. Two versions, undergraduate and graduate are considered.

Teaching Theoretical Issues in Computer Engineering

Strangely enough simple elements of set theory like set notations, set operations, definition of functions is not taught well enough neither at high school nor at freshman math classes at many universities. Students usually understand the notations like $A \bigcup B$ or $A \bigcap (B \bigcup C)$, but the notation $f : A \to B$ has to be explained. Also image and preimage of a subset seems to be new notations for majority of students. Cardinality of a set should be introduced (at least for finite sets) prior to introduction of the power set 2^X to explain this notation. Indeed the notation becomes natural from the equality $|2^X| = 2^{|X|}$.

Since engineering students tend to dislike the theory they need to be constantly motivated by "practical" examples. It is not always easy to find such examples for every mathematical notion covered in class but an instructor should really value them. For instance, to illustrate the composition of a function, cascade of combinatorial circuits can be used. Characteristic function of subject of a finite set can be represented by a binary string making the proof of the formula $|2^X| = 2^{|X|}$ just a simple corollary of the fact that there exist exactly 2^n binary numbers represented by a binary string of length n. The later statement is well known to all computer engineering/science students and the proof creates a bridge between computer engineering and abstract mathematics. The existence of such "bridge" is hard to overestimate. It is especially interesting when engineers present better proofs that those obtained by using traditional mathematical approach. For instance an identity $A \bigcap (B \bigcup C) = A \bigcap B \bigcup A \bigcap C$ can be proved naively by Venn's diagrams or by reasoning like: Let $x \in A \bigcap (B \bigcup C)$, then… Engineering approach will be to write expressions for switching functions for both parts: $A \cdot (B + C)$ and $AB + AC$ and compare their truth tables.

Boolean algebras are traditionally taught in courses in Discrete Mathematics[1] or Digital Design[2]. Many years of our teaching experience show that switching (Boolean) functions can effectively replace Boolean algebras. This is particularly important since switching functions can be represented by combinational circuits.

Undergraduate topics

For the undergraduate course the leading applications are minimization of finite state machines, Myhill Nerode theory, capabilities of computers, and linear machines.

Finite state machines can appear in the very beginning of the course as illustrations of mappings and Cartesian product. Optionally, nondeterministic finite automata can illustrate the notion of a power set. Finite state machines will reappear a few times in the course as application of equivalence relations (Myhill-Nerode theory) for minimization and as examples of semigroups.

Graph theory can be introduced in undergraduate level class but we do not think that more than 2 lectures should be devoted to cover it. Major application of Graph theory is that it provides convenient language to describe some engineering problems. Beyond this, a good example of its usage is VLSI design. It also makes sense to consider some famous problems: like planarity and

four-color problems. Simplicity of formulation and toughness of solution of these problems may interest and challenge the students.

We insist on using mathematical notations for operations. Many books on Discrete Mathematics use the notations like *, □, or ∘ to denote an operation. Although occasionally one can use these notations, we want to point out that the major idea of algebra is that there are many objects of various nature that can be treated as numbers and therefore "$\cdot$" and "$+$" should be used to denote operations. It would be a good practice if a mainstream textbook on Algebra (written for mathematicians) like[3] is used as guidance for preparation in this area. Surely, it has to be appended with examples of applications that appeal to engineers, but the definitions, formulations of theorems and proofs have to be on the strict mathematical level. We understand that this requirement is very challenging for an instructor with traditional engineering education. However cutting corners will undermine the major advantages of mathematical language: clearness and completeness.

Graduate topics

For graduate level course the leading applications are error correcting codes, fault tolerant computing, and cryptography. We consider abstract algebra the major mathematical tool for solving problems in Computer Engineering. We suggest starting with semigroups (Finite state machines as examples). Then proceed with groups, rings and fields. There are numerous examples of applications for these notions:

- Binary arithmetic modulo 2^n and computer arithmetic
- Block codes
- Generator and Parity-check Matrices
- Fast adders
- Polya Enumeration theory

Coverage of algebra should not be too shallow. We suggest covering linear algebra and field theory for a few various reasons. First of all it provides an appropriate mathematical tool for studying linear machines, shift registers, random number generator, and algebraic codes. As a more advanced example (appropriate for graduate level class) we can consider using the notion of transcendental degree of field extension for fault tolerant computations[4,5]. Secondly, broad coverage of algebra will give an opportunity to demonstrate the depth of the subject and avoid the situation when we have many definitions and very few results.

Cardinality of an infinite set should be introduced and the theorem $|2^X| > |X|$ has to be proved. This theorem is a vehicle to showing the limitation of capability of computers, first with finite control and finite memory (Finite State Machines) and then with finite control but infinite memory (Turing Machines).

Graph theory is a good candidate for an example if an instructor chooses to give definition of categories and factors in a graduate level course. Error detecting/correcting codes have become an integral part in many computer Engineering/Science programs. The subject is too broad to be covered appropriately in a class on Theoretical Issues in Computer Engineering. We suggest covering the following topics in a graduate class:

- Codes over finite fields

- Cycle codes
- BCH codes
- Reed-Solomon codes

We recommend using two textbooks[6,7]. Although[6] was published over 20 years ago, it is still one of the best books on applied algebra. [7] is a more recent book. It is shorter and more specialized. We also suggest using[8] as the source of examples of practical applications of error-control coding.

Conclusions

We have presented our experience in teaching Theoretical Issues in Computer Engineering. We strongly believe that there is nothing more practical than a good theory. We suggested a list of topics to be included and emphasized the importance of both high mathematical level of coverage and rich variety of practical examples.

Bibliography

[1]Kenneth H. Rosen "Discrete Mathematics and its Applications", 2002, McGraw-Hill.
[2]Morris M. Mano, Charles R. Kime "Logic and Computer Design Fundamentals", 1999, Prentice Hall.
[3]Serge Lang "Algebra", 1993, Addison-Wesley.
[4]F. S. Vainstein "Low Redundancy Polynomial Checks for Numerical Computations," *Applicable Algebra in Engineering, Communication and Computing*, vol. 7, No. 6, pp. 439-447, 1996.
[5]F. S. Vainstein "Self-Checking Design Procedure for Numerical Computations," *VLSI Design,* vol. 5, No. 4, pp. 385-392, 1998.
[6]Larry L. Dornhoff, Franz E. Hohn "Applied Modern Algebra", 1978, Macmillan Publishing.
[7]Oliver Pretzel "Error-Correcting Codes and Finite Fields", 1992, Oxford University Press.
[8]T.R.N. Rao, E. Fujiwara "Error-Control Coding for Computer Systems", 1989, Prentice Hall.

FEODOR VAINSTEIN
Dr. Vainstein is Professor of Electrical and Computer Engineering, College of Engineering, at Georgia Institute of Technology. Dr. Vainstein's current research interests include fault-tolerant computing, computer hardware and software testing, computer hardware design, digital communication, error correcting codes, applied mathematics and control.

MARK RAJAI
Dr. Rajai is Associate Professor in MST Program at Northern Kentucky University. He also serves as editor-in-chief of an international journal and is member of editorial board of several national and international journals. He has published several books and more than thirty articles and is recipient of several major grants and contracts. He is a nationally recognized researcher and major TV networks-including CNN, ABC, and BBC – and numerous newspapers and radio stations have interviewed him.