Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

TEACHING THEORETICAL COMPUTER SCIENCE AT THE UNDERGRADUATE LEVEL: EXPERIENCES,OBSERVATIONS, AND PROPOSALS TO IMPROVE THE STATUS QUO

Gabriel Robins

August 1989
CSD-880063

Teaching Theoretical Computer Science at the Undergraduate Level: Experiences, Observations, and Proposals to Improve the Status Quo

Gabriel Robins

Computer Science Department
University of California, Los Angeles
Los Angeles, California 90025

Abstract

Theoretical computer science is a difficult subject to teach at the undergraduate level for several reasons. Although it is often a required course for graduation, theoretical computer science has the reputation of being a "tough course," so most undergraduates postpone taking it until absolutely necessary, namely, during their senior year. To compound the problem, many students who enter the course have very little theoretical or mathematical background. If the material is not motivated enough in its presentation to the students, the students quickly drown in the terminology and the abundant technical notation, loosing their interest and patience in the process. Since theoretical models constitute an extensive infra-structure upon which rests much of computer science, it is crucial that undergraduates acquire an appreciation of these concepts before they leave school. Based on observations I made while being involved in teaching this course at UCLA for several quarters, I have developed and used some teaching techniques which have been quite successful, both in increasing student interest, as well as in enhancing their understanding of the material. Finally, to help combat declining academic standards, I propose and describe a new course to be added to existing computer science curricula, namely mathematical maturity and problem solving.

1. Introduction

Theoretical computer science is a difficult subject to teach at the undergraduate level for several reasons. Although in most computer science departments it is a required course for graduation, theoretical computer science (formally CS181 at the University of California) has the reputation of being a "tough course," so most undergraduates postpone taking it until absolutely necessary, namely, during their senior year. To compound the problem, many students who enter the course have very little theoretical or mathematical background; for example, some students have never constructed an inductive proof prior to entering this course, a rather disturbing state of affairs.

If the material is not motivated enough in its presentation to the students, the students usually drown in the terminology and the abundant technical notation, loosing their interest and patience in the process. Since theoretical models constitute an extensive infrastructure upon which rests much of computer science, it is crucial that undergraduates develop an appreciation of these concepts before they leave school.

Based on observations I made while being involved in teaching this course at UCLA for several quarters, I have developed and used some teaching techniques which have proven successful both in increasing student interest, as well as in enhancing their understanding of the material. The techniques for improving the quality of this course range from general presentation aids (e.g. using multi-colored chalks to highlight certain relationships), to giving common-sense intuitions behind major theorems and logical formulae (e.g. why regular languages have a pumping property, or why negation switches existential and universal quantifiers), to various creative techniques to boost the "morale" of the class (e.g. distributing cartoons at the beginning of each session).

In this paper I share my experiences regarding the teaching of theoretical computer science, and argue that most of the techniques I used are general enough to be applicable in teaching other technical courses. Finally, I propose and describe a new course to be added to existing computer science curricula, namely mathematical maturity and problem solving. This course would induce students to exercise their common-sense and simple logic while improving their problem-solving skills and enhancing their mathematical sophistication.

The organization of the rest of this paper is as follows: section 2 describes the typical background of undergraduates entering the theoretical computer science course, section 3 describes some general teaching techniques, section 4 describes teaching techniques that I specialized for this particular course, section 5 describes the textbook used in this course as well as possible alternatives, section 6 makes several suggestions to improve the status quo, and section 7 summarizes and concludes the present discussion.

2. The Course Reputation and Typical Student Background

The vast majority of the undergraduates who enter a theoretical computer science course simply have no wish to attend it; rather, this course is imposed upon them by the standard graduation requirements for obtaining a degree in computer science. Theoretical computer science has an awful reputation among undergraduates, and students therefore postpone enrolling in it until the very last quarter prior to graduation. I have heard many resentful undergraduates describe this course using adjectives such as "dry", "boring", "unmotivated", "contrived", "impractical", and "too abstract". Interestingly, those very few students (usually those who excel in the material) describe it as "elegant", "challenging", "practical", and "stimulating". To what is owed this discrepancy of opinions?

Surely there exists a human tendency for those who understand the material better to automatically think more highly of it, but there is more to this situation. I believe that the decline in the level of undergraduate education as a whole is manifested here. Undergraduates are allowed to sail through a four-year degree while doing relatively little abstract thinking or problem-solving; they are forced to learn by rote and rarely are assigned tasks which require serious resourcefulness or insight. Every passing year marks a noticeable average decline in the mathematical maturity of college seniors, and an average increase in their general apathy towards their chosen fields.

Richard Feynman, the famous Nobel Prize-winning physicist once said: "the power of instruction is seldom of much efficacy except in these happy dispositions where it is almost superfluous" [Feynman]. I found this analysis to be very accurate. Of a 50-student undergraduate class, there are usually 2 or 3 individuals who get near-perfect scores; I never see these individuals during office hours, and they rarely solicit my help; instead they rely on their ability to read the text books independently at their own pace, investing a lot of effort in the learning process. On the other hand, another 4 or 5 other individuals every quarter seem to monopolize about 90 percent of my office hours, and for all the help I gave them they barely manage to pass the course.

The students in this course consist mainly of computer science and engineering majors, but other majors often take this course as well (for example, mathematics, anatomy, architecture, and cognitive science majors). Although most of the students attending this course should have had considerable exposure to mathematics, this is not at all apparent during discussions with them. In fact, sometimes I got the distinct feeling that the vast majority of the students entering this course have never seen simple summation formulae or know even the simplest of theorems about graphs. Not that such knowledge necessarily constitutes a prerequisite for this course, but rather it is my experience that the nebulous quality known as "mathematical maturity" (or at least "mathematical curiosity") is seriously lacking in these students.

3. Some Teaching Techniques

In this section I outline several teaching techniques and practices that I have found to be valuable towards increasing student interest in the material, as well as in enhancing their understanding of same.

3.1. Using Colored Chalk

I found that when writing at the board, it is extremely helpful to use colored chalk instead of plain white chalk. Using several colors, I am able to color-code diagrams in order to highlight certain relationships among the concepts discussed, thereby enhancing the students' comprehension of the material. Color-coding also works quite well in overhead-transparency slide presentations.

One drawback to using colored chalk is that one rarely finds colored chalk available in standard classrooms, perhaps because it is somewhat more expensive than ordinary white chalk. The second drawback of using colored chalk in the classroom is that unless the students are also taking notes in colored ink, some of the information will be lost when the students copy down colored diagrams from the board. This problem is easily solved by advising the students during the first meeting to obtain a multi-colored pen if they do not own one already (a standard 4-colored pen may be purchased for less than two dollars). This scheme has proved quite successful in practice.

3.2. Giving Students a Copy of Slides

3.2.1. When giving a presentation involving overhead transparencies, I always photocopy my slides ahead of time so I can distribute the copies at the beginning of my lecture. Students find this practice extremely helpful, since they don't have to worry about "missing" anything while taking notes. The disadvantage of this scheme is of course the cost involved in the photocopying.

I believe that this is a very small price to pay, however, since this scheme increases student comprehension and therefore reduces the amount of help they will require outside of class. Note that I am not arguing that office hours do not have their place, but rather that if the comprehension of the students is increased, then less effort would have to be spent by the instructor in addition to the lectures themselves.

3.3. Cartoons as a Tool to Improve Morale

Often at the beginning of a lecture I distribute to the students a sheet containing some cartoons. This serves to break the tension, alleviate the boredom, and encourage students to attend section. One student has literally admitted that the only reason she attended the discussion section is that she knew she would get to see some new cartoons each time. I found that these cartoons also improve the morale of the class, especially when a particular cartoon is somewhat relevant to a topic being presently discussed in class. Many students have also shown these cartoons to their friends and family, inducing some additional excitement about class meetings.

My favorite cartoonist is Gary Larson, because his cartoons exhibit insight, present novel points of view, and possess a certain "pseudo-intellectual" quality, much like does Woody Allen humor [Larson]. Another favorite cartoonist is Matt Groening, who has put out some very amusing anthologies about the life of students [Groening]. The photocopying cost here is negligible (say one or two dollars per class meeting), while the psychological benefits are numerous.

3.4. Reaching the Students Through Familiar Examples

Often students find it difficult to understand very fundamental logical relationships and propositions. In such situations I find it remarkably useful to relate the material to the students in a manner familiar to them, couching the logic in terms they can easily grasp. For example, once the students had some trouble understanding why exhibiting an efficient algorithm is usually much more difficult than proving than none exist, so I gave the following analogy: if I wanted to convince someone that I am a millionaire, it would suffice to show that my bank account has a seven-figure balance; on the other hand, suppose that I wanted to convince you that I am <u>not</u> a millionaire, how can I convince someone of it? Showing a low balance on my bank account does not suffice, for maybe I still own very expensive real estate, or perhaps some gold bullion in a numbered Swiss bank account, etc., etc. Now everyone in the class had realized why an non-existence proof had to be so powerful/exhaustive kind of a proof - since <u>all</u> possibilities had to be individually considered and disproved in turn, and there are usually an infinity of scenarios one must defeat or argue about.

As a second example, numerous students had trouble understanding why logical negation switches existential and universal quantifiers in a proposition, in the sense that if P(x,y) is a logical formula over two variables, "$\forall$" denotes universal quantification, "$\exists$" denotes existential quantification, and "$\sim$" denotes negation, then in general we may write the following: $\sim(\forall\ x \exists\ y\ P(x,y)) \Leftrightarrow (\exists\ x \forall\ y \sim P(x,y))$. The last form seemed rather cryptic to the students until I said "please replace x by a <u>person</u>, y by a <u>house</u>, and interpret P(x,y) to mean 'person x owns house y.' What does the left-hand-side mean to you now?" To this everyone correctly replied "NOT[<u>every</u> person owns <u>a</u> house]". When I next asked for an interpretation of the right-hand-side, now they all correctly replied "<u>some</u> person does not own <u>any</u> house", the equivalence of the two forms now being obvious to everyone.

As a last example, some students had difficulty grasping the meaning of the pumping lemma for regular languages, which appears quite technical at first (and even second) glance. I wanted the students to remember not only the lemma itself, but be able to derive it from first principles at any time! To this end I constructed the following intuitive explanation: "suppose you have a regular language accepted by some finite automaton. Then take a long input string in the given language, and start 'feeding' it to the automaton, one symbol at a time. The finite automaton is of fixed size, and it changes states at every symbol it 'digests'; but the input is longer than the number of states, so if you keep feeding it symbols, the poor finite automaton must eventually cycle back onto itself and re-enter some previously visited state. Now look at the sequence of symbols that caused this cycle, and lo-and-behold it can be fed to the helpless beast over and over again, as many times as one wishes, followed by the ending sequence of our original string, so that the entire input sequence is also a string in our original language. In this manner, every sufficiently long string in a regular language induces an infinite family of strings, all of which must also be in the language!" After hearing this intuitive explanation, properly illustrated with color-coded diagrams, almost none of the students forgot the proof for the pumping lemma, even many weeks later.

When explaining technical formulae and relationships in an informal manner as above, one must make certain that the students are aware of the informality inherent in the explanation. These examples illustrate only how an instructor may appeal to student intuition utilizing notions from everyday life; while intuitive arguments are never a substitute to mathematical rigor, in my experience they can greatly amplify comprehension of the latter. For an amusing set of arguments of why intuition will never be completely substituted by rigor, the reader is encouraged to examine the classic paper of [De Millo, Perlis, and Lipton].

3.5. Calling on the Class and on Individual Students

Students who managed to obtain extra-credit points during class meetings were very proud of it, especially when I would ask them to explain their answers to the rest of the class or even present their solution on the blackboard. This would also give them a chance to practice public-speaking, and one of the students even thanked me for being given such an opportunity. When a student presents a solution on the board, however, it is important not to humiliate or ridicule him/her, even if they are totally wrong; instead, I might say "nice try, but not quite..." All such interactions should be geared towards encouraging them to speak up and be assertive.

3.6. Informal Course Evaluations

Since normal teaching evaluations are conducted too late in the quarter for either the teacher nor the students to benefit out of the revelations resulting from such evaluations, I conduct an early informal evaluation of myself by my students. I care very much about my students' progress and comments, and I use their feedback to further improve my style, presentations, and the areas of material on which I should concentrate more. These evaluations are take-home and anonymous; they typically includes questions such as the following:

- How do you feel about having extra-credit problems at the beginning of each section?
- Do you feel the problems I am giving you are too easy? Too hard?
- Am I going over the material too slow? Too fast?
- Am I helpful in answering questions? During office hours?
- Am I too formal? Not formal enough? Too serious? Not serious enough?
- Do I appear organized? Not organized? Is my style confusing?
- How do you feel about having a chance to go to the board and present your solutions? Do you welcome it? Does it intimidate you? Do you like it?
- What should I do more of?
- What should I do less of?
- Do I seem to direct my discourse at the bulk of the class? The top half? The top forth? Only a few people? Why does it seem that way?
- What do you think about the homeworks? Too many? Too few? Too hard? Too easy? Too boring? Anything else?
- What do you think about the textbook? Too formal? Too informal? Interesting? Easy to read and follow?
- How do you feel about theoretical computer science at this point?
- Any other comments about myself? My style?
- Any other comments about the subject? The problems? The course?
- Just for fun, write some comment here that you would like me to read in front of the class out loud; remember, this is anonymous!

3.7. Homework and Examination Solutions

Solutions to all homework assignments are worked out in detail, neatly type-set, and distributed to the class, so that the students have a precise written record of what the correct solutions look like. The only drawback I see in this practice is the photocopying cost involved in the duplication of old assignments. If this cost becomes prohibitive, these solution sets may either be placed on reserve at the various libraries so that the students can duplicate them at their own expense, or else these sets may be reproduced as official course-notes and sold to the students. My personal opinion is that the cost-to-benefit ratio associated with this practice is extremely favorable.

Solutions to examination problems are treated in a similar manner: they are worked-out in detail by either myself or the grader, neatly type-set, and distributed to the students. I also distribute to the students several old examinations from previous quarters. Some instructors are reluctant to do this, but it came to my attention that several fraternity houses regularly provide their members with old examinations as a "standard service." Some students are also able to receive these materials from their friends who have attended the same course in previous quarters. This selected trickle of information puts the rest of the students at a distinct disadvantage. My practice of distributing old exams to everyone eliminates this unfair advantage by giving everyone the same competitive edge.

3.8. Giving the Students My Home Phone Number and Address

I always give my students my home phone number, as well as a postal mailing address. Many instructors are apprehensive about having their students know their private home phone number, but my experience has shown that the students usually exercise excellent judgement in such matters: they only call about urgent matters, such as regarding missing an exam, etc. During an average quarter, I only receive about half-dozen phone calls at home from my students, and always at very reasonable hours of the day. Meanwhile, all the rest of the students may take comfort in the knowledge that if they have to contact me, all they need to do is simply pick up the phone and call me. Again, I use this device to put their minds at ease and establish a certain rapport of closeness with them.

3.9. Encouraging Students to Exchange Information

Just as students would like to have access to my own phone number and address, they would like to have a way of reaching each other. To overcome student shyness, while still respecting the privacy of others, I developed the following scheme: I pass a sheet of paper around the room, instructing students that towards enhancing the information transfer potential with respect to this course, anybody who puts their name and phone number on that sheet will be given a list of everybody else who did the same. It was observed empirically that about two-thirds of a typical class would participate in this mechanism by submitting their phone number.

3.10. Public Speaking

Speaking in front of a crowd is naturally a stressful task, but if a lecturer feels nervous in front of the audience, the resulting awkwardness is not particularly conducive to learning. Several years ago I took a course in public-speaking, in order to improve my effectiveness as a lecturer. I learned various techniques of establishing rapport with an audience, preparing lectures, and delivering speeches. This knowledge has helped me a great deal, and I would highly recommend that every lecturer should be exposed to some sort of formal training in public speaking.

Eye-contact is a very important aspect of capturing the audience's attention. Another major device that I use to establish rapport with an audience is humor: everybody likes to laugh, and when I get the students to laugh and enjoy themselves, classroom moral is higher all around, which in turn makes for a more positive learning environment. I like to use various anecdotes, especially ones which are (supposedly) true and involve famous people; such anecdotes can serve to make the students remember the material better, especially if it is somehow tangentially related to the topic under discussion.

For example, one of my favorite anecdotes from professional folklore involves the great physicist Niels Bohr: one day Bohr explained to a class certain aspects of subatomic particle interactions, when he happened to use the phrase "close enough for all practical purposes." When someone from the audience asked him to elaborate on what that meant, Bohr explained: "suppose all the men in this room lined up along one side, while all the women lined up along the opposite side of the room, and with every passing minute, these two parallel rows of individuals would move towards each other in such a manner as to halve the distance between them. Well, in theory, the men and the women would never reach each other, but in practice, they would very soon be close enough for all practical purposes." This anecdote will fit nicely into a discussion of power series or limits.

3.11. Extra Review Sections

I make it standard practice to schedule additional review sections before the midterm and the final examinations, in order to give the students additional opportunity to ask questions and practice the course material. The meeting times of these additional sections is established by vote, in order to maximize the number of students that will be able to attend, and I have found that the students that attend these review sessions tend to perform better on the pending examination.

3.12. Off-the-wall Questions

Sometimes students ask the strangest questions; it is important for the instructor to encourage all questions, and never make the student feel incompetent or stupid for asking the question. It is very easy, and indeed enticing, for a professor to develop an openly condescending attitude towards the students, but I view this as a serious flaw in a teacher. I go to great lengths to impress upon the students that there is no such thing as a stupid question, and I try to treat each query from the class with the seriousness and respect it deserves.

Human psychology is such that the humiliation of others may prove (albeit unconsciously) to be a source of elation; this is unfortunate, and I believe that insulting or intimidating the students is a poor practice which fosters resentment, is not at all conducive to learning, and only mirrors problems with the personality of the teacher him/herself. A simple antidote to looking down upon students is the realization that most of them lead complex and interesting lives, and that some of them are truly experts at certain areas about which the professor knows literally nothing about (such as martial arts, team sports, business, arts, music, weapons, cars, etc.) Keeping this attitude in mind, it would be easier to respect the students, even when they do not appear particularly well-versed in the course material.

3.13. Open Book Examinations

I much prefer open-book examinations, both as a student, and as an instructor. As a student, knowing that an exam is open-book puts my mind at ease and relaxes me, because I am assured that I need not memorize every trivial detail of the material, but rather concentrate on the important high-level ideas.

Since in an open-book exam students are able to respond to certain questions simply by copying the appropriate paragraphs out of the book, it increases the work on the part of the instructor to come up with questions the solutions for which will not be readily found in the text; however, I think this is an effort well-spent. Of course, some simple definition-like questions may still be included in the exam, just to make sure the students know the basic concepts (or at least where they may be found in the textbook...)

A small number of students, on the other hand, dislike open-book examinations, believing that this kind of an exam is automatically more difficult than a closed-book examination. These students much prefer to memorize whole textbooks rather than try to be insightful. Although I can sympathize with these individuals, I still maintain that testing students on how resourcefully they can apply the concepts gives a much better indication of their mastery of the material than does a simple check of their memorization potential. In any case, any difficulty introduced via making an examination an open-book one, me be mitigated via some degree of leniency in grading.

3.13.1. Selling Examinations Hints vs."Double Jeopardy"

Selling hints-for-points during an examination is a device that can be used to increase the benefit of examinations and reduce student anxiety. That is, if a student becomes "stuck" on the first part of an exam question but needs the answer to that part in order to solve a subsequent part, the student may be willing to give up a few points from their total exam score in order to be given the correct answer on the spot, either in full or in part. It is completely up to the discretion of the instructor what is the point price of a given fraction of an answer.

The hints may either be constructed,

"priced", and duplicated by the instructor ahead of time and disbursed to the students upon request during the examination, or else be given either verbally or scribbled on the student's exam paper when requested. The former method (of preparing written hints ahead of time) is more uniform and assures that all students will be treated equally and fairly with respect to the hints given; however, it involved more work on the part of the instructor.

My experience has been that if the students know that they can buy hints during an examination, they are more relaxed since they can cease to worry about "double jeopardy" situation where the solution to each question in a set depends upon the previous question. In this sense the availability of hints is more of a psychological crutch than a physical aid; but since a calmer state of mind may all by itself help improve student performance, this scheme on the average offers considerable benefit, at only a minimal cost in effort to the instructor. For example, my experience has shown that during 3-hour examinations on a class of fifty students, a total of half-dozen or so hints will be requested by the class during the exam.

3.14. Extra-Credit Problems

I found that students are rather docile during section, and their minds often tend to drift from the material being discussed. To help combat the declining attendance and the low energy level of the students, the first thing I would do each time when I came into the lecture room is put a couple of problems on the board, and then announce that anybody who solves any of these problems within 15 minutes will receive a few extra-credit points towards the next homework assignment. The students would then scramble to solve these problems as fast as they can.

This scheme has had several positive effects. First, many students stopped being late to class, knowing that otherwise they would miss out on those extra-credit problem sessions; in addition, several students who rarely bothered to show up for class, started instead to attend class meetings regularly. Secondly, the energy level of the students, as well as student participation has risen dramatically. When I would come into the room I would notice the students alert, pen-in-hand, and ready to solve problems for extra-credit.

I found that the same small group of students would get the extra-credit points each time, so from then on I included some easy problems as well, increased the number of problems I gave each time (to say, about five), made the point-value of the problem proportional to its difficulty, and announced that any one individual may solve at most two problems. This scheme insured that the extra-credit points would not be monopolized by the same small set of students, causing frustration to the less-abled or slower individuals; in other words, everybody had a fair shot at gaining extra-credit points during section.

Sometimes in the middle of a lecture, after asking a rhetorical question and noting the many blank stares from the students, I would write the question down on the board and ask "if this question was on the next exam, could you solve it?" Usually this prodding still did not illicit a response from the class, so I would then proceed to ask "for twenty extra-credit points, would you solve it now?" At this point the students would spring into action and many of them very quickly came up with a solution. The moral of these incidents is that if you want something done, put a reward on it; this is classroom-capitalism at its best.

3.15. The Lack of Initiative and Curiosity in Students

Initiative and curiosity are qualities that are visibly lacking in the majority of undergraduates. Too many students blunder through the required courses while expending just enough effort to obtain passing grades; they typically are not interested in any topic that is not going to be covered in the examinations, and remain disturbingly ignorant of even the existence of entire (significant) subdisciplines of their major field. Lest my critique of undergraduates should appear too harsh, I do not expect every undergraduate to concern themselves with current research problems, but on the other hand I strongly believe that students of any scientific discipline should strive to familiarize themselves with, at least in outline, the state of the art and the general research trends in their field.

In particular, students of computer science should glance regularly at general professional publications such as Communications of the Association for Computing Machinery (CACM). It pays to be familiar with the literature, even if one only has the time to only skim through tables-of-contents and abstracts. I believe that if a department undertakes the practice to photocopy the tables of contents of various technical journals and post them on bulletin boards or in other designated locations (or even hand them out to the students), the students would be much better informed of their chosen field.

4. A Self-Printing Program and Other Extra-Credit Problems

In order to give the students a chance to earn additional points towards improving their grade, I usually assign some take-home extra-credit problems, which may be turned in anytime before the end of the quarter. I try to make these problems challenging and at the same time amusing; for example, one of my favorite problems to assign for extra-credit is the following:

Problem: write a program that when executed, prints out **exactly** itself and stops. No run-time input whatsoever is allowed to be used by the program (i.e., no reading the keyboard, files, pipes, etc.) Any programming language may be used, but note that the program must print itself out exactly, right down to the last punctuation mark, tab, and carriage return.

Although at first glance this task sounds impossible, it is quite possible; moreover, there is no "dirty trick" required, such as a special command, or an obscure construct in a particular language, since this problem is meant to be essentially language-independent. I consider this an elegant and a subtle problem, which despite its short solution, often eludes experienced, professional programmers. I also offer a few extra points to the individual who finds the shortest solution. The reader is encouraged to try to solve this problem sometime.

The shortest solution I have yet seen to this problem (only 66 characters long in C) is based on one actually turned in by a resourceful student. I would be very curious to see any shorter solutions in C, or a proof that none exist. A natural extension of this problem is to write a program that prints itself backwards. A rather amusing (yet sinister) application of the idea of self-replicating programs is described by Ken Thompson, one of the two original inventors of UNIX, in his 1983 ACM] Award Lecture: using self-replication it is possible to embed a particularly devious type of Trojan horse in operating systems [Thompson].

5. The Textbook Used in this Course

The textbook used in this course is typically Introduction to Automata Theory, Languages, and Computation, by Hopcroft and Ullman. This is altogether a good textbook, being both concise (less than 400 pages), up-to-date (1979), and well-written (Aho, Hopcroft, and Ullman are one of the most prolific team of authors in all of computer science). The main problem with this book is that not enough of it can be covered in one quarter: out of fourteen chapters, only the first six are usually covered, and very rarely chapters seven and eight are also introduced. This is rather discouraging because it means that in a typical quarter there is hardly any time to discuss Turing machines or undecidability. A second problem is that students complain that this textbook is too formal; this is a less serious problem, as this complaint is likely to exist no matter how the material was presented, and besides, I have heard certain other students complain that this text is not formal enough!

Other recent comparable texts exist, notably [Papadimitriou, and Lewis], [Harrison], [Cohen], [Savitch], [Salomma], [Davis, and Weyuker], and [Harel]. Many of these texts follow the same general format as does [Hopcroft, and Ullman] modulo some peculiarities. [Harel] is by far the most unconventional text in this lot. It is very informal, which would make it quite accessible to freshmen, and even non-majors, yet it manages to cover advanced topics such as algorithms, complexity, lower bounds, NP-completeness, Turing machines, universality, undecidability, recursive function theory, transformations, parallelism, concurrency, and probabilistic algorithms. It is full of clever diagrams and amusing (but relevant!) quotations from the Old Testament; in addition, it contains a detailed annotated bibliography for more in-depth reading.

[Harel] is a pleasure to read, and I believe would also be a pleasure to teach from. Naturally, this text would have to be supplemented by some additional material on regular and context-free sets, but with this caveat, I would highly recommend that [Harel] be used as a basic text in this course, supplemented perhaps by selected sections from [Hopcroft, and Ullman].

6. Some Proposals to Improve the Status Quo

6.1. Keeping the Students Informed

I often found that undergraduates are sometimes extremely uninformed as to what goes on in the department. For instance, on one occasion

I discovered that several computer science seniors had never heard of the UCLA Computer Science Department Quarterly publication, a bound booklet that is published four times a year (in over one-thousand copies) by the UCLA computer science department. This publication details the official policy, course, degree requirements, and program information of the computer science department, as well as faculty biographies and research interests. To hear that some students have not been aware of even the existence of this publication is disturbing. Other times I found that world-class speakers had given talks in our department, while many of our students remained informed of these events.

To whom may the responsibility here be attributed? While some students will never find large amounts of initiative, they should nevertheless be kept informed of the department's professional activities. I would recommend that all computer science students be given a copy of the Quarterly upon its publication, and be mailed a monthly schedule of computer science talks, seminars, and other special events. I believe that the postage/overhead costs involved with this practice could be easily overshadowed by the corresponding increase in student interest and participation. Of course much of the initiative in such matters must come from the students themselves, but the department would do well to endeavor to meet the students half-way.

6.2. Permanent Student Computer Accounts

In order to keep students informed and in contact, both with the department as well as with each other, I suggest that they be given permanent computer accounts when they are first enrolled, to be cancelled only when they graduate or drop out. This will enable anybody to reach everybody via electronic mail, and will help establish a greater sense of cohesiveness among the students.

The usual argument in favor of account deactivation is that old accounts hog too much disk space; this could be mitigated by a proper tape-archive migration policy for aging/unused files. Even without such a facility, mass-storage device prices have sufficiently dropped in recent years as to make such schemes financially viable.

6.3. A Proposal for Splitting Into Two Courses

I believe that the main reason that theoretical computer science is difficult to teach (and learn), is that too much material is packed into one course. Only with a tremendous effort can an instructor manage to squeeze into one quarter the first 7 (out of 14) chapters of [Hopcroft, and Ullman], and even then, many of the topics will be left inadequately covered (or completely neglected altogether). The serious computer science majors (and graduate students) who enroll in this course are often held back by less initiated non-computer science majors, since the latter tend to greatly slow down the pace of the course due to their lack of mathematical sophistication. The result is that all too often graduate students complete this course while never having heard of NP-completeness, or other equally important ideas.

My proposed solution to this problem is to break this course into two separate undergraduate courses. The first course will introduce to the students the various basic definitions, mathematical abstractions, and proof methods involved in theoretical computer science. Next, the Chomsky hierarchy will be discussed, as well as simple examples of languages of the various common types, along with discussions of the various machine models. The course will conclude with a brief introduction to undecidability, NP completeness, and a shallow discussion of complexity theory.

The second course will go over the above topics in much greater depths; in particular, it will challenge the students with more difficult examples of languages having (or not having) certain properties, present a more refined partition of the hierarchy of formal languages, discuss in detail various restrictions and generalizations of computation models, present numerous NP-completeness proofs, and elaborate on some results of complexity and lower-bound theory.

I would recommend that all engineering-related students would be required to pass the first course, but only the pure computer science majors should be made to complete the second course. This would ensure that non-computer science majors will receive a solid exposure to all of the important concepts of computer science theory yet without drowning in rigor and notation, while computer science majors would have an opportunity to acquire a greater in-depth understanding of selected relevant topics. In any case, the problem with the status quo is that few topics are discussed in very great detail, while other topics are left completely unmentioned, and I believe that it is this lack of balance that is primarily responsible for many of the problems entailed in teaching theoretical computer science at the undergraduate level.

6.4. A Proposal for a Brand New Course

To combat mathematical apathy at the undergraduate level, I would recommend adding to the standard curriculum a course named Mathematical Maturity and Problem Solving. This course would expose students to a collection of problems selected from basic mathematics, introductory logic, riddle/puzzle books, and the "Mathematical Themas" and "computer recreations" sections of Scientific American. Any problem which requires a certain "Aha!" insight to solve (or is otherwise fun to solve) would be a good candidate for inclusion in this course.

This course would induce students to exercise their common-sense and logic while improving their problem-solving skills and enhancing their mathematical sophistication. A secondary goal of this course would be to illustrate to the students that computer science and mathematics could be a fascinating field of inquiry, one in which problem-solving is a most gratifying activity. Supplementary texts for this course may include [Polya], [Gardner1], [Gardner2], [Gardner3], [Gardner4], [Harel], [Bentley1], [Bemtley2], plus a selected few others from a large number of recreational mathematics books.

Problems showcased in this course may include ones that impinge upon the areas of graph theory, Ramsey theory, combinatorics, trasfinite arithmetic, formal language theory, distributed computing, lower-bound theory, recursive function theory, undecidability, and basic logic. I strongly believe that undergraduates majoring in computer science should at the very least be made aware of the existence of each one of these areas of study.

7. Summary

Theoretical computer science is a difficult subject to teach at the undergraduate level: many students who enter the course have very little theoretical or mathematical background, and if the material is not motivated enough in its presentation to the students, the students quickly drown in the terminology and the abundant technical notation, loosing their interest and patience in the process. Since theoretical models constitute an extensive infra-structure upon which rests much of computer science, it is crucial that undergraduates acquire an appreciation of these concepts before they leave school. I have developed and discussed some teaching techniques which have proven successful both in increasing student interest, as well as in enhancing their understanding of the material.

One of the problems with the status quo in teaching theoretical computer science to undergraduates is the disbalance that is created when few topics are discussed in very great detail, while other topics are left completely unmentioned. I made a recommendation that the standard undergraduate theoretical computer science course be split into two separate courses. This would ensure that non-computer science majors will receive a solid exposure to all of the important concepts of computer science theory yet without risking being drowned in numerous technical details, while computer science majors would have an opportunity to acquire a greater in-depth understanding of selected relevant topics.

Finally, to help combat declining academic standards, I proposed and described a new course to be added to existing computer science curricula, namely mathematical maturity and problem solving. This course would expose students to a diverse collection of problems, riddles, puzzles, and proof methods selected from basic mathematics and introductory logic, and will be designed to cultivate within students the nebulous quality of "mathematical maturity".

8. Acknowledgements

Many thanks go to Professor **Sheila Greibach** for encouraging me to become involved in the teaching of theoretical computer science at UCLA, and for some insightful comments and suggestions regarding this paper. I thank Professor **Eli Gafni** for enabling me to gain valuable additional experience by allowing me to prepare and deliver numerous lectures in his classes. Finally, to the hundreds of students whose learning I had the pleasure of assisting: I thank you for improving me as a teacher.

9. Bibliography

Cohen, D., Introduction to computer Theory, John Wiley, ans Sons, Inc., 1986.

Davis, M., and Weyuker, E., Computability, Complexity, and Languages: Fundamentals of

Computer Science, Academic Press, New York, 1983.

De Millo, R., Lipton, R., and Perlis, A., Social Processes and Proofs of Theorems and Programs, Communications of the Association for Computing Machinery, 22, pp. 271-280, 1979.

Feynman, R., Leighton, R., Sands, M., The Feynman Lectures on Physics, Addison-Wesley, Volume II, p. 5., 1963.

Gardner, M., New Mathematical Diversions, The University of Chicago Press, Chicago, 1966.

Gardner, M., Aha! Gotcha: Paradoxes to Puzzle and Delight, W. H. Freeman and Company, New York, 1982.

Gardner, M., Wheels, Life, and Other Mathematical Amusements, W. H. Freeman and Company, New York, 1983.

Gardner, M., Knotted Doughnuts and Other Mathematical Entertainments, W. H. Freeman and Company, New York, 1986.

Groening, M., School is Hell, Pantheon Books, New York, 1987.

Harel, D., Algorithmics: the Spirit of Computing, Addison-Wesley, 1987.

Harrison, M., Introduction to Formal Language Theory, Addison Wesley, 1978.

Hopcroft, J., and Ullman, J., Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, Massachusetts, 1979.

Larson, G., The Far Side, Andrews, McMeel, and Parker, Kansas City, 1982.

Lewis, H., and Papadimitriou, C., Elements of the Theory of Computation, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

Polya, G., How to Solve It.

Robins, G., Class Notes for Theoretical Computer Science, Unpublished Manuscript, UCLA computer Science Department, 1987-1988.

Salomma, A., Computation and Automata, Cambridge University Press, 1985.

Savitch, W., Abstract Machines and Grammars, Little, Brown and Company, 1982.

Thompson, K., and Ritchie, D., 1983 ACM A.M. Turing Award Lecture, Communications of the ACM, Volume 27 Number 8, August, 1984, pp. 757-763

10. Table of Contents