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 <u>mathematical maturity and problem solving</u>.

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 <u>mathematical maturity and problem solving</u>. 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: $\sim(\forall x \exists y\ P(x,y)) \Longleftrightarrow (\exists x \forall y \sim P(x,y))$. 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 **Shella Grelbach** 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 **Ell Gafnl** 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

1.........Introduction.......1
2.........The Course Reputation and Typical Student Background.......2
3.........Some Teaching Techniques.......2
  3.1.........Using Colored Chalk.......2
  3.2.........Giving Students a Copy of Slides.......3
  3.3.........Cartoons as a Tool to Improve Morale.......3
  3.4.........Reaching the Students Through Familiar Examples.......3
  3.5.........Calling on the Class and on Individual Students.......4
  3.6.........Informal Course Evaluations.......4
  3.7.........Homework and Examination Solutions.......4
  3.8.........Giving the Students My Home Phone Number and Address.......5
  3.9.........Encouraging Students to Exchange Information.......5
  3.10.......Public Speaking.......5
  3.11.......Extra Review Sections.......6
  3.12.......Off-the-wall Questions.......6
  3.13.......Open Book Examinations.......6
    3.13.1......Selling Examinations Hints vs."Double Jeopardy".......6
  3.14.......Extra-Credit Problems.......7
  3.15.......The Lack of Initiative and Curiosity in Students.......7
4.........A Self-Printing Program and Other Extra-Credit Problems.......8
5.........The Textbook Used in this Course.......8
6.........Some Proposals to Improve the Status Quo.......8
  6.1.........Keeping the Students Informed.......8
  6.2.........Permanent Student Computer Accounts.......9
  6.3.........A Proposal for Splitting Into Two Courses.......9
  6.4.........A Proposal for a Brand New Course.......10
7.........Summary.......10
8.........Acknowledgements.......10
9.........Bibliography.......10
10.......Table of Contents.......12

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

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 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: a UCLA graduate in computer science who has never heard of NP-completeness certainly does not help the department in achieving its stated goal of permanently establishing itself as first-rate in the nation.

Based on observations I made while being involved in teaching this course at UCLA for several quarters, I have developed some general 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 many 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 lists some good exam questions, section 6 describes the textbook used in this course as well as possible alternatives, section 7 makes several suggestions to improve the status quo, and section 8 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", "useless", "contrived", "impractical", and "too abstract". Interestingly, those very few students (usually those who excel in the material) describe it as "elegant", "fascinating", "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.

Often when I solicit questions from an undergraduate class, the most predominant question happens to be "do we have to know this for the final exam?" It greatly disturbs me as a teacher to witness such student apathy. Once I announced to a class that although I will grade them very leniently, the final I composed for the class was "designed to make them think"; this was immediately followed by groans and complaints from many of the students, whereupon I almost had to apologize to the class for not giving instead questions that can instead be answered mechanically by rote.

On another occasion I asked a class of fifty students how many have attended a departmental distinguished-speaker-series seminar talk given by Edsger Dijkstra just the previous week. Only two people raised their hands, and both were of course graduate students. When I next asked how many even knew who Dijkstra was, only one additional person responded. Given that a department goes through considerable effort and expense to invite such an influential figures to speak on campus, why don't more students attend these talks? In fact, even among graduate students seminar attendance is so poor, that our department finds it necessary to resort to making such attendance mandatory via treating these talks as a required course-for-credit. This a rather sad state of affairs.

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.

With respect to whether the burden of work in a course falls on the teacher (to teach the material) or the student (to learn the material), I found that most undergraduates feel it is primarily the burden of the teacher to "run in circles" around the students, catering to the students' every whim, while forcefully injecting the course material into the students' minds. I also know some teachers who display quite the opposite sentiment, namely that it is mainly the responsibility of the students to keep up with the teacher no matter what, do considerable reading and thinking on their own, while energetically absorbing the course material into their own minds.

I believe that while the first view borders on laziness (on the part of the student), the second is rather egotistical (on the part of the instructor); therefore, as a teacher, I am willing to meet my students halfway: if a students shows effort, patience, and a certain amount of initiative, I will go out of my way to guide them and make their learning task as easy and painless as possible. This philosophy stems from my belief that although learning new concepts is one of the most difficult jobs that exist, most of the effort expended towards this end can only come from the student's side of the teacher-learner team. That is, giving a lecture about material that one is already familiar with is much easier than understanding a lecture concerning new material; the former is mainly a mechanical process of slide-preparation followed by an exercise in public speaking, while the later entails the rapid assimilation and analysis of new ideas, a process that is cognitively very taxing.

3. Some Teaching Techniques

In this section we 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 certain diagrams in order to highlight certain relationships among the concepts discussed, thereby enhancing the students' comprehension of the material. For example, when proving that the cardinality of the set of points of the real number line is the same as that of the set of points of the real X-Y plane, one must show how to map a pair of real numbers into a single real number; this is usually accomplished via the "interlacing" the two real numbers, digit by digit, a process that can be instantly grasped if two different colors are used to represent the respective digits of the two original numbers.

The conventional alternatives to using colors usually consist of drawing arrows, pointing to pairs of related objects with the fingers, or using special characters or symbols. I found that these methods are not very satisfactory since a multitude of arrows tends to clutter up a diagram, finger-pointing fails when discussing more than a couple of objects, and the liberal usage of special characters often serves only to confuse the students. In addition to helping in the highlighting of relationships, a color diagram or picture is much more pleasant to look at (if not, then why do people prefer a color-television to a black-and-white television?), thereby increasing the students' attention. 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. I resorted to buying my own colored chalk out of my own pocket, an investment that I found well-worth the effort. 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

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, and they have thanked me for it on several occasions. A slide presentation tends to proceed rather quickly since the speaker has prepared the material off-line; if the students are expected to take notes of the presentation, it follows that they will not be able to pay as much attention to what the speaker is saying. This problem is therefore alleviated if they already have a complete copy of the slides, which also serves to help them relax, 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. For example, suppose a two-hour lecture consists of 15 overhead transparencies, and there are 40 people in the class. The photocopying costs in this situation would amount to approximately 15*40*0.04=$24; conducting private office hours, on the other hand, costs the department much more, and therefore if disbursing copies of the slides to the class would save at least one office-hour worth of confusion to the students, as it often does in fact, this scheme constitutes a more cost-effective usage of departmental resources.

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. The evidence in practice for these observations is that after a confusing lecture has taken place, the number of students that come into my office hours greatly increases, and with that, inadvertently also increase the student requests for special tutoring/help sessions in addition to my regular office hours.

3.3. Cartoons as a Tool to Improve Morale

At the beginning of every 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. Some students also shown these cartoons to their friends and family, inducing some additional excitement about section 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. My second favorite cartoonist is Matt Groening, who has put out some very amusing anthologies about the life of students. The photocopying cost here is negligible (say one or two dollars per class meeting), while the psychological benefits are numerous. For example, I included the cartoon on the left on a handout sheet containing an extra-credit problem, while the cartoon on the right was placed in the middle of the written midterm examination:

Right side! One two, one two, one two, leftside! One Two, one two, one Two, one.. C'mon! Keep those cerebellums up!.. One two, one two-..

Brain Aerobics

Halfway through the exam, Allen pulls out a bigger brain.

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 carried the class through the following discussion: "if I wanted to convince you that I am a millionaire, it would suffice to show you that my bank account has a seven-figure balance; on the other hand, suppose that I wanted to convince you that I am not a millionaire, how can I convince you of it?"

Their first instinct was to say "show us a low balance on your bank account" but they soon though better of it when they started to recognize the problem. I next said "even if I show you a low balance, maybe I still own very expensive real estate; and even if I can manage to prove that I don't own any real estate, perhaps I keep 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 all possibilities had to be individually considered and disproved in turn, and there are usually an infinity of scenarios one must defeat.

A second example entails logical implication: some students were under the impression that if A implies B then automatically B implies A. In order to convince them that their reasoning was fallacious, I prompted the class to consider the statement "if it rains, then the sidewalk is wet" and asked whether they can infer from it that "if the sidewalk is wet, then it rains." A couple of students replied in the affirmative, until I pointed out that if the lawn sprinkles were running (or similarly for any number of other familiar phenomena), that could wet the sidewalk without the falling of rain, so if the sidewalk is wet, it does not necessarily follows that rain has fallen. This has established firmly (albeit informally) in the mind of the students the difference between logical implication and logical equivalence.

As a third 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: $\sim(\forall\ x\ \exists\ y\ P(x,y)) \Longleftrightarrow (\exists\ x\ \forall\ y \sim P(x,y))$. The last form seemed rather cryptic to the students until I said "please replace x by a person, y by a house, 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[every person owns a house]". When I next asked for an interpretation of the right-hand-side, now they all correctly replied "some person does not own any 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 section 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.

I often pause in mid-sentence and wait for the class to complete it. This works quite well in soliciting class participation, and also keeps me informed with respect to the general progress of the class. I also ask the class many questions, often directing a question that came from a student back towards the rest of the class; this gives the students a feeling that they are helping each other, and thus serves to develop a strong sense of camaraderie between them.

On a lighter note, I sometimes take a mock-vote on a question; for example, I may ask "how many would say that the complement of a regular language is regular?", whereupon many people would raise their hands. Next I would ask "how may think it isn't?" and observe several hands inadvertently go up. Finally I would declare "its settled then, complementation preserves regularity - proof by consensus!" This gimmick never fails to get a good laugh out of the students. Needless to say, I would later ask for a volunteer to prove that statement rigorously.

3.6. Philosophy Presented During the First Class Meeting

During the first class meeting, I would announce to the students that the purpose of the section is to answer questions, review the lectures, work out problems, and monitoring their progress. I would emphasize that there is no such thing as a stupid question (only stupid answers, and I promise the students that I shall do my best to avoid giving those in reply...) I would explain to the students that many problems in theoretical CS are of a "puzzle-nature": the solutions are often short once a certain "AHA!" insight is noticed. These are a lot of fun to solve and I try to challenge the students with as many of these as I can find.

I implore the students not to postpone questions to the last-minute before an exam or before an assignment is due, but rather to ask for my help as soon as they get lost or have a pressing question. I emphasize to the students that the material of this course builds on previous material in a very strong sense; for example, if a couple of definitions are missed, the next lecture may not make much sense at all. So therefore it is crucial that the students do not fall behind; the consequences of doing so will in all likelihood be more severe than in other courses that they may have had.

I instruct the students to pay special attention to definitions and technical terms, telling them to memorize the important definitions thoroughly, because subsequent material will make numerous references to terms defined previously; lack of knowledge of these will seriously impede their understanding of subsequent material.

I caution them that "cramming" does not work very well in this class, so they shouldn't bet their grade on it; the ideas presented in this course sometimes have to "simmer on a low flame" in the back of their minds for a long time before an effective understanding of them can be achieved. I encourage the students to type up their assignments. My favorite tool for document processing is the MacIntosh computer running the MS Word 3.02 word-processing editor; I highly recommend both. This reduces reader hours expended in grading the homeworks, as well as forces the students to better scrutinize the correctness of their solutions, which results in better scores for them. Since some students do not have access to a computer or a laser printer, I am careful to mention that typing the homework solutions is not absolutely mandatory, just very advisable.

I explain to the students that certain problems sometimes require many days of wrestling before they are solved satisfactorily; this means that the students should not give up if they had thought about a problem for half an hour or so without solving it. In fact, they shouldn't give up even after several hours of attacking the same problem. I suggest to them that sometimes it is helpful to leave a problem alone and go do some other activity, later returning to the problem, perhaps solving it much more quickly.

My experience has shown that students in a first-course in theory often do not see the point of much of the material or the exercises. The usefulness, relevance, and richness of much of the topics covered in this course is therefore lost on many individuals. To these skeptics among the students, although it would be rather difficult to convince them of it at the beginning of the course, I ask to take on faith the assertion that the concepts they will learn in this course constitute an elaborate infra-structure upon which hinges and rests a very large and significant subset of computer science; so even at times when it seems difficult to accept this premise, an open mind should be kept.

In order to become better acquainted with the students, I ask the students to give me some information about themselves in writing, including their name, E-Mail address, major, year in school, degree objective, previous relevant courses, interests, and why they are taking this course.

3.7. Informal Course Evaluations

Since normal TA evaluations are conducted too late in the quarter for either the TA 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. Running such an evaluation impresses upon the student how important improvement is to me, both theirs and my own.

I make the evaluation a take-home one, since I would like the students to think carefully about what they would like to communicate to me; I don't want to press them into filling out an evaluation quickly for its own sake, so I urge them to take their time and give it some thought: these evaluations may affect the content and structure of the rest of my presentations and office-hour consultations with my students. Another purpose served by this evaluation is that it gives the students an rare opportunity to vent out their frustrations and complaints, and has been quite effective in this respect.

The evaluation is an anonymous one, in order to give the students complete freedom in expressing their comments. When they wish to turn it in, they may either leave it on my desk or mail it directly to my P.O. Box. The informal evaluation I have my students fill out typically includes questions such as the following:

- Are you enjoying my presentations?
- 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?
- How much do I seem to care about your progress/proficiency in the material?
- 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?
- What do you think about the professor? Do you enjoy the lectures?
- 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?
- Any other comments about the department? UCLA? Life? The universe?
- 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!

3.8. Homework and Examination Solutions

Solutions to all homework assignments are worked out in detail, neatly type-set, and distributed to the students when they receive their graded assignments back. This entails more effort than simply reviewing the solutions on the board during section (although I do that also), but the advantage is that the student now have a precise written record of what the correct solutions look like. Moreover, this enables me to easily distribute homework problems-and-solutions from previous quarters to the students, so that they may be exposed to a variety of problems and solution techniques. Since every quarter new homework problems are assigned, giving the students solutions to homework problems from previous quarters does not create a conflict of interest. Students have commented on numerous occasions that they really appreciate this service.

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. As with homework solutions from previous quarters, 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 on campus 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 courses 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.

3.9. 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.

The reason I give out a postal address to my students is so that they can mail me their assignments in case they are out of town or otherwise not able to come into school when an assignment is due in class. My students have actually used this option on numerous occasions. If an instructor is reluctant to give out his residence address, a P.O. box address will suffice and serve the same purpose equally well. On a couple of occasions students have used this address to send me "thank-you" cards for all the help they received from me. This practice of enabling the students to reach me anytime either by phone or mail is another example of how a nominal effort on the part of the instructor can result in substantial benefits as far as the psychology of the students is concerned.

3.10. 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; this is really handy when a student misses a lecture and needs some notes or other information on what transpired during his/her absence, or when no progress can be made towards the solution of a particularly difficult homework problem. The main obstacle in the way of such exchanges is that students are usually too shy to exchange addresses with people they do not know very well, and when they urgently need to obtain information from fellow-students (typically right before an exam, or the night before a difficult assignment is due in class), it is often too late to try to contact anybody. The other problem is that some students prefer to protect their own privacy to such an extent that they do not wish to give out their phone number or address to anyone, a sentiment with which I can completely sympathize. How then is this problem to be resolved?

The scheme I developed is as follows: 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. This scheme, and in what I perceive as a very fair manner, preserves the rights of the very private individuals to maintain their privacy, while allowing the rest of the students to corroborate throughout the course. It was observed empirically that about two-thirds of a typical class would participate in this mechanism by submitting their phone number. This device has proven quite successful in enhancing the camaraderie between classmates as well as in encouraging the free exchange of information and ideas, a property that is a necessary condition in every successful research environment.

3.11. 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 was taught 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.

For example, eye-contact is a very important aspect of capturing the audience's attention; in particular, talking to an audience without looking at them is not very effective, and is even considered by many to be impolite. Another major device that I use to establish rapport with an audience is humor, which can be extremely effective in creating a certain closeness between a speaker and an audience. 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. As explained elsewhere in this article, I often disburse cartoons to my students. I also 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 <u>never</u> reach each other, but in practice, they would very soon be <u>close enough for all practical purposes</u>." This anecdote will fit nicely into a discussion of power series or limits.

Another classic anecdote involves Sir Arthur Eddington, who was a renowned expert on general relativity. In the early 1920's a reporter once asked Eddington whether it was true that at that time there were only three people in the entire world who understood general relativity. When instead of replying, Eddington paused and frowned, the reporter was quick to ask him what was the matter, to which Eddington replied: "nothing, I'm just trying to figure out who the <u>third</u> person is."

Invariably, many similar anecdotes involve the famous German mathematician John von Neumann, who was supposedly faster at numerical calculations than any of the early-generation computers. Sometimes I find it amusing to make up my own (fictional) anecdotes; for example, after explaining the "big O" asymptotic order-of-magnitude notation which disregards multiplicative constants, I like to present the following amusing scenario: suppose special relativity was invented by a theoretical computer scientist instead of by a physicist. Then after some lengthy computations, the computer scientist would derive the equation $E=O(M)$, depicting the linear relationship between matter and energy. But when it comes the time to calculate the constant, namely c^2, the theoretical computer scientist would dismiss this task as trivial and unimportant, leaving atomic weapons never to be invented. This amusing "alternate reality" drives home the importance of the numerical constants which theorists often tend to ignore.

3.12. 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; typically the consensus establishes the meeting time for these extra sessions to be 6:00 or 7:00 in the evening. These sessions are open-ended, in the sense that the session will go on as long as the students remain awake. Students have often thanked me for conducting these extra sessions, and I have found that the students that attend these review sessions tend to perform better on the pending examination. Oftentimes interesting revelations are made during these sessions regarding certain gaps in the students' understanding of the material, more frequently so than in normal class or section meetings; perhaps this is because as it gets later in the day, students become less inhibited and are therefore less embarrassed to confess ignorance regarding a certain topic with which they are experiencing difficulty.

3.13. Off-the-wall Questions

Sometimes students ask the strangest questions; for example, once while explaining to a student how to accept a certain particular context-free language using a push-down automaton, the student interrupted my explanation and asked why not also use an array in addition to the stack. It took the student quite a while to understand that it is preferable to use a model that is as simple as possible when characterizing the complexity of some phenomenon - a principle more generally referred to by philosophers as Occam's razor. In our example, we need not use an array since ANY context-free language may be recognized using a one-stack push-down automaton; in fact, the class of context-free sets corresponds exactly to the class of languages accepted by push-down automata.

Usually students defer their questions to the privacy of the instructor's office hours, not wishing to appear unintelligent in front of their peers during class meetings. But in any case, it is important for the instructor to encourage such 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.

3.14. 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. Under this scenario, I am able to walk away from the course having secured knowledge of more relevant concepts, and experiencing less anxiety in the process of learning, which ideally should be a pleasurable experience anyway. I believe that the majority of students share my opinion in this issue.

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.

Once when conducting a review session for a final examination one quarter, I started the session by saying "I just finished composing the final exam, and there are good news and bad news." Having captured the attention of the class I continued: "the bad news is that that you'll have to think" This announcement started a wave of groans propagating through the room. This made me smile as I continued: "the good news is that if your answer even remotely resembles something that can conceivably be extended into a correct solution, you will receive most of the credit for that answer." At this point the whole class was laughing. In other words, I believe that any difficulty introduced via making an examination an open-book one, me be mitigated via some degree of leniency in grading.

3.14.1. 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.

As concrete examples of such "real-time" hints, we give the following, which refer to the examination questions given in another section of this paper:

<u>30% hint for #2</u>: $\{ww^R \mid w \in \{a,b\}^*\}$ <u>cannot</u> be recognized in linear time on any one-tape TM.

<u>60% hint for #2</u>: Apply Rice's theorem.

<u>20% hint for #3</u>: Can a write-once tape be simulated by an ordinary tape?

70% hint for #3: Simulate an ordinary tape using a write-once tape by copying the entire contents of the tape used so far to a fresh new section of tape (every time a symbol need to be overwritten).

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. Giving the hints to students in an ad hoc fashion during and examination provides for more flexibility and saves the instructor some preparation, but is a less uniform method.

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.

3.15. Extra-Credit Problems

I found that students are rather docile during section, and their minds often tend to drift from the material being discussed; part of the reason for this is that our section usually met Friday at 8:00 a.m., with many students being very late to class or half-asleep. To help combat the lateness and the low energy level of the students, each time when I came into the lecture room the first thing I would do is to 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 section, knowing that otherwise they would miss out on those extra-credit problem sessions; in addition, several students who rarely bothered to show up for section, started instead to attend section 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.

3.16. 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). For general reading on up-to-date advances in all of the sciences, an excellent source is the journal Scientific American; it is very accurate and reliable while not too technical, and due to its style and colored diagrams also makes very enjoyable reading. Other interesting periodicals include Science, Spectrum, Science News, and the American Mathematical Monthly.

Other, more technical/specialized publications, should also be examined by students on a regular basis, but if such an investment in time proves too prohibitive, at least the table-of-contents of several technical journals should be inspected periodically. The advantages of such a practice often more than repays for the time investment it requires; for example, last year a friend of mine was working quite intensely on a particular problem in parallel complexity theory, when I informed him that I have seen a recent technical paper in the European Journal of Theoretical Computer Science which already solved the very same problem. Although my friend already spent considerable time on that problem (and in fact made some good progress towards a solution), my pointer to the published paper saved him a substantial amount of time, as well as some potential embarrassment had he tried to publish his work independently.

The moral of this discussion is that 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. To those critics who would say "you can bring a horse to water, but you can't make him drink," my reply is "yes, but maybe he will get thirsty eventually..."

3.17. Other Readings

My personal curiosity extends into physics, astronomy, and cosmology; I am fascinated by current theories regarding the origin and evolution of the universe, the life-cycles of stars and galaxies, and modern theories of matter and energy. For the benefit of those readers who would like to pursue such reading (yet at the risk of loosing the interest of other readers), some excellent (non-technical) recent books on these topics are [Riordan], [Pagels], [Gribbin1], [Gribbin2], [Glashow], [Hawking], and [Einstein].

Regarding initiative and resourcefulness, qualities also notably lacking in many students, I believe a most appropriate quotation comes from billionaire entrepreneur Donald Trump: "Most people think small, because most people are afraid of success, afraid of making decisions, afraid of winning. And that gives people like me a great advantage." [Trump] Although Trump makes this observation with respect to the business world, I found that it applies to the academic world as well. I find it very gratifying to put my thoughts into writing, and and the present paper is an example. Sometimes even the process of writing itself leads to further insights and results.

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, I often assign 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].

The informed reader may at this point wonder what does this problem have to do with theoretical computer science. Indeed, the connection becomes obvious if the same problem would have been stated in a different guise: "prove that there exists a one-tape Turing machine that when running on the null input, prints out exactly its own description (with respect to any fixed acceptable encoding scheme) and halts." The existence of such a Turing machine is easily shown using an application of the recursion and the S-M-N theorems [HU]. The former problem simply asks for a specific instance (implementation) of such a machine.

Other take-home extra-credit questions that I have assigned include the following:

- Let $L \subseteq \Sigma^*$ be an arbitrary regular language, and $L' = \{w \mid w \in \Sigma^*$ and $ww^R \in L\}$. Is L' necessarily regular? Prove your answer.
- Show that if L is any language over a one-letter alphabet, then L^* is regular.
- Define **Half(L)**=$\{v \mid$ for some w such that $|V|=|W|$, vw in L$\}$. Show that if L is regular, then so is **Half(L)**.
- Show that neither of K_5 or $K_{3,3}$ is a planar graph.

- Define **Sqrt(L)**=$\{v \mid$ for some w such that $|V|=|W|^2$, vw in L$\}$. Show that if L is regular, then so is **Sqrt(L)**.
- Is $\{w_1\$w_2 \mid w_1,w_2 \in \{0,1\}^*, w_1 \neq w_2\}$ a context-free language?
- Give a linear-time algorithm to sort the ratios of N pairs of integers between 1 and N.

Some of these problems come out of research papers and are given as double-starred (i.e., difficult) exercises in [Hopcroft, and Ullman]. The problem about graphs relies on Euler's formula and its solution may be found in any standard text on graph theory. The last problem regarding sorting ratios in linear time was raised by Bob Tarjan when I was a student at Princeton. In any case, I consider none of these problems to be trivial, and so I typically give students the entire quarter to work on them; correct solutions count up to half of one homework assignment. The interested reader may find it amusing to solve some of these problems.

I inform students ahead of time that some of the solutions to these problems may be found in the literature of other textbooks, and that they may actually look these up and still be eligible to receive the credit. This is designed to motivate the students into looking up some books and papers which they would otherwise never knew existed.

Problems which I give in class for extra-credit in "real-time" are obviously easier than ones above. I list here some of the problems I gave out as extra-credit in class, along with the relative point-values given for correct solutions. Typically I allow students about 20 minutes to work on such a "quiz"; the point value associated with these questions are intended to reflect their relative difficulty level:

- Let $L_1=\{0^n1^n \mid n>0\}$. Is the complement of L_1 a regular language? [4 point]
- Is $L_2 = \{0^i1^j \mid 1 \leq i \leq j \leq 2i\}$ a context-free language? [8 point]
- Is $L_3 = \{0^i1^j \mid i \neq j\}$ a context-free language? [12 point]
- Is the complement of L_1 a context-free language? [16 point]
- Is L_3 regular? [20 point]
- Is $L_p=\{a^p \mid p \text{ prime}\}$ a regular set? [3 point]
- Is $L_c=\{a^c \mid 4 \leq c \text{ composite}\}$ a regular set? [5 pts]
- Is L_pL_c a regular set? [10 pts]
- Give an example of two non-regular languages whose concatenation is regular. [5 pts]
- Give countably-infinite different examples of two non-regular languages whose concatenation is regular. [10 pts]
- Are there an uncountable number of different examples of two non-regular sets whose concatenation is regular? Explain why or why not. [15 pts]

- Show that a countable union of countable sets is countable. [5 pts]
- Show that in any group of six people, there are either 3 mutual strangers or 3 mutual acquaintances. [10 pts]
- Show that the intersection of two uncountable sets can be empty, finite, countably infinite, or uncountably infinite. [5 pts]
- Let $L = \{w \in \{a,b\}^* \mid w$ contains an equal number of a's and b's$\}$. Show that L is not context-free, or else give a CFG for L. [15 pts]
- Show that in any group of people, there are at least two people with the same number of acquaintances within the group. Assume that the "acquaintance" relation is symmetric but non-reflexive. [10 pts]
- Show that the difference of an uncountable set and a countable set is uncountable. [9 pts]
- Prove or give a counter-example: a countably-infinite union of regular sets is a regular set. [8 pts]
- Is the transitive closure of a symmetric closure of a binary relation necessarily reflexive? [5 pts]
- Show that a countable union of countable sets is countable. [7 pts]
- Show that if T is countable, then the set $\{S \mid S \subseteq T, S$ finite$\}$ is also countable. [10 pts]
- Give a simple bijection from the natural numbers, and the rationals crossed with the integers. [8 points]
- Show that $n^4 - 4n^2$ is divisible by 3 for all $n \geq 0$. [7 points]
- How many distinct Boolean functions on N variables are there? In other words, what is the value of $|\{f \mid f:\{0,1\}^N \rightarrow \{0,1\}\}|$? [8 points]
- How many distinct N-ary functions are there from finite set A to finite set B? Does this generalize the previous question? [12 points]

As the reader will notice, these problem sets include material from graph theory, Ramsey theory, combinatorics, and trasfinite arithmetic, as well as from formal language theory. I strongly believe undergraduates majoring in computer science should at the very least be made aware of the existence of each one of these mathematical areas. Some of these problems come from textbooks or are simple corollaries to well-known theorems; others I have made up myself. The interested reader may find it amusing to solve some of these problems.

Many of the proofs in theoretical computer science require a mathematical intuition and a strong common sense; when the students regularly solve such problems, especially under a time constraint, they become better able to think "on their feet."

5. Sample Midterm and Final Examination Questions

For the record, I now list some of the problems that appeared on examinations during the quarters that I taught this course:

1) Language classification

Characterize each of the following languages as tightly as possible (and explain your answer), by stating whether it is:

- finite
- regular <u>but</u> not finite
- deterministic context-free <u>but</u> not regular
- context-free <u>but</u> not deterministic context-free
- recursive <u>but</u> not context-free
- recursively enumerable <u>but</u> not recursive
- not recursively enumerable

a) $L_1 = \{w \in \{a,b,c\}^* \mid 2\#a\text{'s}(w)=3\#b\text{'s}(w)=6\#c\text{'s}(w)\}$
b) $L_2 = \{a^n b^n c^n w \mid 0 \leq n,\ w \in \{a,b,c\}^*\}$
c) $L_3 = \{(guess)^n (my)^n (type)^n \mid n>0\}$
d) $L_4 = \{a^n b^m \mid$ n is an integer that encodes a TM that halts on the input encoded by the integer m$\}$
e) $L_5 = \Sigma^* - L_4$
f) $L_6 = \{www \mid w \in \{x,y,z\}^*,\ |w| < 1000\}$
g) $L_7 = \Sigma^* - L_6$
h) $L_1 = \{w \in \{a,b,c\}^* \mid$ w has equal number of a's, b's, and c's$\}$
i) $L_2 = \{a^n w b^n w^R \mid n>0,\ w \in \{a,b,c\}^*\}$
j) $L_3 = \{a^n \$ a^n \$ a^n \mid n>0\}$
k) $L_4 = \{a^n a^n a^n \mid n>0\}$
l) $L_5 = \{1^m \mid$ TM m halts when running on the blank tape$\}$
m) $L_6 = \{1^m \mid$ TM m loops forever when running on the blank tape$\}$
n) $L_7 = \{0^n 1^n \mid n > 0\} \cup \{0^n 1^{2n} \mid n > 0\}$
o) $L_8 = \{(w^R w)^p \mid w \in \{x,y,z\}^{999}$ and p is a prime such that $p < |w|^2\}$

2) Linear-time-language recognition

Is it decidable whether a given language can be recognized on a one-tape TM in linear time?

3) Write-once Turing machines

Would having only write-once tapes reduce the power of Turing machines? (That is, if TMs were <u>not</u> allowed to overwrite a non-blank tape symbol with a *different* symbol, how would this restriction affect the class of r.e. languages?)

4) Which properties get "lost in the shuffle"?

We define the SHUFFLE of two strings $v,w \in \Sigma^*$ as:

$$\text{SHUFFLE}(v,w) = \{v_1w_1v_2w_2\ldots v_kw_k \mid v=v_1v_2\ldots v_k,\ w=w_1w_2\ldots w_k,\ \text{some } k \geq 1,\ v_i,w_i \in \Sigma^*,\ 1\leq i\leq k\}$$

and extend the definition of SHUFFLE to two languages $L_1,L_2 \subseteq \Sigma^*$ as follows:

$$\text{SHUFFLE}(L_1,L_2) = \{w \in \text{SHUFFLE}(w_1,w_2) \mid w_1 \in L_1, w_2 \in L_2\}$$

a) is the SHUFFLE of two context free languages necessarily context free?)
a) is the SHUFFLE of two context sensitive languages necessarily context sensitive?)
c) is the SHUFFLE of two recursively enumerable sets necessarily recursively enumerable?

5) Variations on a TM

A Turing machine is said to **visit** the i<u>th</u> tape square during a computation, if its read-write head enters the i<u>th</u> tape square, and sometime later leaves this square. A tape square is said to be **revisited** during a computation if it is visited more than once. A **revisitless TM** is a TM such that during no computation is any tape square revisited. A **revisit-once TM** is a TM such that during no computation is any tape square revisited more than once. A **revisit-twice TM** is a TM such that during no computation is any tape square revisited more than two times.

Determine whether each of the following restricted classes of Turing machines has less "power" than ordinary (unrestricted) TM's. Try to characterize precisely as you can the reduction in power in terms of the class of sets recognized by such TM's:

a) revisitless TM's
b) revisit-once TM's
c) revisit-twice TM's

6) Sorting it out

Given an arbitrary alphabet $\Sigma = \{a_1,a_2,\ldots,a_n\}$, we can impose a total ordering on it in the sense that we can define $<$ so that $a_1<a_2<\ldots<a_n$. We can now proceed to define the SORT of a string $w = w_1w_2\ldots w_k \in \Sigma^*$ (where $w_i \in \Sigma$) as:

$\text{SORT}(w) = w_{\sigma(1)}w_{\sigma(2)}\ldots w_{\sigma(k)}$ so that $w_{\sigma(i)} < w_{\sigma(i+1)}$ for $1 \leq i \leq k-1$ and σ is a permutation (i.e., a 1-to-1 onto mapping $\sigma:[1..k]\rightarrow[1..k]$)

and extend the definition of SORT to languages $L \subseteq \Sigma^*$ so that $\text{SORT}(L) = \{\text{SORT}(w) \mid w \in L\}$. For each one of the following statements, state whether it is true or false and explain:

a)* $\text{SORT}(\Sigma^*)$ is not regular.
b) $\text{SORT}(L) \subsetneq L$

c) SORT(SORT(L))=SORT(L)
d) SORT preserves regularity
e) SORT preserves context-freeness
f) SORT preserves recursive enumerability
g) SORT(SHUFFLE(L_1,L_2)) = SORT(L_1L_2)
h) There exists no language L and an alphabet Σ such that SORT(L)=SHUFFLE(L,L)=L.

7) Undecidability

For each one of the following questions, determine whether it is decidable or not:

a) Given a TM, does it halt when running on the input "computo ergo sum"?
b) Given a TM, does it accept a recursively enumerable set of inputs?
c) Does the decimal expansion of π contains 99 consecutive 9's?

8) Subsequences and Supersequences

Define the SUBSEQ of a language as:

$$\text{SUBSEQ}(L) = \{w_{i(1)}w_{i(2)}\ldots w_{i(j)} \mid w\in L,\ w=w_1w_2\ldots w_k,\ \text{some } k\geq 1,\ w_h\in\Sigma^* \text{ for } 1\leq h\leq k,\ i(m)\leq i(m+1) \text{ for } 1\leq m\leq j-1,\ 1<j\}$$

and similarly the SUPERSEQ of a language as:

$$\text{SUPERSEQ}(L) = \{v_1w_1v_2w_2\ldots v_kw_kv_{k+1} \mid w\in L,\ w=w_1w_2\ldots w_k,\ \text{some } k\geq 1,\ w_i\in\Sigma^* \text{ for } 1\leq i\leq k,\ v_j\in\Sigma^* \text{ for } 1\leq j\leq k+1\}$$

Examples: SUBSEQ({abc})={ϵ,a,b,c,ab,ac,bc,abc}, SUPERSEQ({abc})=$\Sigma^*a\Sigma^*b\Sigma^*c\Sigma^*$.
For each one of the following statements, state whether it is true or false and explain:

a) SUBSEQ preserves regularity
b) SUBSEQ preserves context-freeness
c) SUPERSEQ preserves regularity
d) SUPERSEQ preserves context-freeness
e) SUBSEQ(L) $\subsetneq$ SUPERSEQ(L)
f) SUPERSEQ(L)=SHUFFLE(L,Σ^*)
g) L$\subsetneq$SUPERSEQ(L)
g) L$\subsetneq$SUBSEQ(L)
i) There exists no language L and an alphabet Σ such that SUBSEQ(L)=SUPERSEQ(L)=L.

9) NP completeness

a) explain briefly what it means for a language to be NP-complete.
b) Suppose you heard a rumor that someone showed that sorting is NP-complete; would you believe this rumor? Why or why not?

10) Horrible Sets Over One-Letter Alphabets

Give an example of a non-R.E. language over a one-letter alphabet, or prove that non exist.

6. 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: [Papadimitriou, and Lewis] is more formal and its notation is a little more difficult to read, although it does devote special chapters to the propositional and predicate calculi, respectively, something that is lacking in other texts. [Harrison] gives a most comprehensive treatment of context-free languages and grammars, and hence is more appropriate for a graduate-level course. [Cohen] gives a very coherent presentation, with diagrams on almost every page. Although it should be very accessible to undergraduates, this text is rather verbose - in over 800 pages Cohen manages to cover only about half the material contains in [Hopcroft, and Ullman]'s 400 pages. [Savitch] is a rather concise (200 pages) treatment, but does not address complexity at all, nor does it even mention NP-completeness. In addition to covering the standard topics, [Salomma] gives a balanced intermediate-depth coverage of some novel material, including cryptography and Petri nets, topics that are conspicuously missing in other texts. [Davis, and Weyuker] has a novel order of presentation, treating computability first and only later the Chomsky hierarchy; on the other hand, it covers logic, the undecidability hierarchy, and advanced set theory.

[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].

6.1. Further Reading

Students often ask me to refer them to books covering various other topics in more depth. For the record, my recommendations follow: for recursive function theory, the authoritative text is [Rogers], with a less rigorous (nor thorough) treatment in [Cutland]. A good introductory logic text is [Boolos, and Jeffrey] or [Andrews]. A reasonable graph-algorithms text is [Even]. An encyclopedic coverage of NP-completeness is given in [Garey, and Johnson]. A good introduction to combinatorics is presented in [Polya, Tarjan, and Woods]. The state-of-the-art in data structures is covered in [Tarjan].

An excellent text on sequential algorithms is [Sedgewick]. The mathematical analysis of sequential algorithms is explained in great depth in [Purdom, and Brown]. Two very good texts on computational geometry are [preparata, and Shamos] and [Edelsbrunner]. An understandable account of concurrency may be found in [Ben-Ari]. A representative sample of research in distributed algorithms is collected in [Gafni, and Santoro]. A classic exposition on problem-solving in general is given in [Polya]. The functional programming paradigm was pioneered by [Backus]; a good introduction is given in [Eisenbach], while various examples and discussions appear in [Robins2].

The first twenty Annual Turing Award lectures may be found in [Ashenhurst]. [Gamow] gives an amusing introduction to transfinite arithmetic. Numerous clever programming problems may be found in [Bentley1] and [Bentley2]. A fascinating and artistic perspective of recursion and incompleteness appears in [Hofstadter1]. Many interesting recreational mathematical problems appear in [Hofstadter2], [Gardner1], [Gardner2], [Gardner3] and [Gardner4]. A detailed analysis of numerous mathematical games and strategies appears in the two volumes of [Berlekamp, Conway, and Guy]; the second volume also contains a fascinating proof that the game of "life" can simulate a universal Turing machine. A delightful anthology of cartoons is printed in each of [Larson1], [Larson2], [Larson3], [Larson4], [Larson5], [Larson6], [Larson7], [Larson8], [Unger], [Davis], and [Groening].

7. Some Proposals to Improve the Status Quo

7.1. 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 Computer Science Department Quarterly publication, a bound booklet that is published four times a year (in over one-thousand copies) by our 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.

7.2. 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. So far in our department, only graduate students and professors hold permanent computer accounts; why not afford undergraduates the same luxury? For example, I use electronic mail almost exclusively for most interactions with co-researchers, faculty, and students, and this works out extremely well, especially when the person I wish to contact is in a different state or country.

Sometimes even graduate students are required to "renew" their computer accounts; why? Does a department really believe that an enrolled graduate student will cease to use his/her computer account? Not likely! Enrollment-long accounts will eliminate the superfluous administrative overhead entailed in account renewal procedures. 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, suppose that each student requires about 5 megabytes of storage on average (a generous estimate). Then the combined files of approximately 100 students could be accommodated on a single half-gigabyte drive; the price of disk drives has sufficiently dropped so that the money involved is no longer a major hurdle. For example, I have 70 megabytes of storage connected to my MacIntosh Plus, and this storage had cost me about $2,500 in 1986; presently in 1988, one can buy a 100 megabyte disk for just a little over $1,000.

7.3. A Proposal for Breaking CS181 Into Two Courses

I believe that the main reason that CS181 (University of California's 10-week course in theoretical computer science) is difficult to teach (and learn), is that too much material is packed into one course, or equivalently, that the course duration is too short. Only with a tremendous effort can an instructor manage to squeeze into a 10-week quarter the first 8 (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 CS181 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 finish CS181 while never having heard of NP-completeness, or other equally important ideas.

My proposed solution to this problem is to break CS181 into two separate courses, CS181A and CS181B. CS181A 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.

CS181B will go over the above topics in much greater depths; in particular, CS181B 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 CS181A, but only the pure computer science majors should be made to complete CS181B. 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.

7.4. Is Infinity an Integer?

Once during a special final-review section, I posed the following question: "is a countable union of regular languages necessarily regular?" The answer to this simple question is of course no, since for example, the non-regular language $\{0^n1^n \mid n>0\}$ can be represented as the countable union $\{01\} \cup \{0011\} \cup \{000111\} \cup \ldots$ In fact, any language (even a horribly non-r.e. one) may be similarly represented a countable union of its elements taken as singleton sets.

Nevertheless, as a result of a raise-of-hand poll, it appeared that many of the students believed that the answer to the above question is "yes". In fact, even when presented with the above counter-example, some of the students still insisted that the class of regular sets is closed under countably-infinite union; the counter-example served only to confuse them, to the point that they could not even see where the fallacy of their belief lay. One particularly assertive student adamantly insisted that in the text [Lewis, and Papadimitriou] it was stated that the union of any number of regular sets is regular, and that since infinite unions were not explicitly excluded, then it follows that the assertion also applies to them as well.

At this point I realized that the problem the students were having was not with the "union" operator on regular sets, but rather with operators in general, and their extension from a finite number of applications to an infinite number of applications. I therefore asked them whether it was true that the addition of two integers necessarily yields an integer; they all nodded in agreement. Next I asked whether the addition of a million, or any other fixed number, of integers necessarily yields an integer; again everyone agreed. Finally, I inquired whether it follows that a countable infinity of integers, when added together, results in an integer; to my astonishment, many of the students insisted that this is still true!

While trying to hide my shock, I wrote on the board the following form:

$$\sum_{i=1}^{\infty} i$$

and asked again whether the value of this form is equal to some integer; three or four students still firmly maintained that it was! Moreover, none of my further arguments seemed to convince them otherwise. At some points I raised my hands in frustration, and gave up on this point, not wishing to waste any more valuable class time on it. It is amazing that college seniors, after having completed a four-year curriculum of courses in mathematics and computer science, do not know exactly what an integer is! I believe that this incident reflects on a fundamental flaw in the education system, in that it allows students to pass through many years of course-work while retaining fundamental gaps in the understanding of basic concepts.

7.5. A Proposal for a Brand New Undergraduate 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.

8. Summary

Theoretical computer science is a difficult subject to teach at the undergraduate level, and has gained a universal reputation of being a "tough course." 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. Based on observations that I have made while being involved in teaching this course at UCLA for several quarters, 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 CS181 be split into two separate courses, namely CS181A and CS181B, and while all engineering-related students would be required to pass CS181A, only the pure computer science majors should be made to complete CS181B. 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". I believe that instituting this course into the current computer science curriculum will significantly increase the professional competence of our graduates, and thus may help to improve and maintain the national ranking of our department.

9. 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.

10. Bibliography

Andrews, P., An Introduction to Mathematical Logic and Type Theory: to Truth Through Proof, Academic Press, Orlando, 1988.

Ashenhurst, R., ACM Turing Award Lectures: the First Twenty Years, ACM Press Anthology Series, Association for Computing Machinery, New York, 1987.

Backus, J., Can Programming Be Liberated from the Von Neumann Style? A Functional Style and its Algebra of Programs, Communications of the ACM, Vol 21, No. 8, August, 1978, pp. 613-641.

Ben-Ari, M., Principles of Concurrent Programming, Crawley, England, 1982.

Bentley, J., Programming Pearls, Addison-Wesley, Reading, MA, 1986.

Bentley, J., More Programming Pearls, Addison-Wesley, Reading, MA, 1988.

Berlekamp, E., Conway, J., Guy, R., Winning Ways for Your Mathematical Plays, Volume 1: Games in General, Academic Press, Orlando, Florida, 1985.

Berlekamp, E., Conway, J., Guy, R., Winning Ways for Your Mathematical Plays, Volume 2: Games in Particular, Academic Press, Orlando, Florida, 1985.

Boolos, G., and Jeffrey, R., Computability and Logic, Cambridge University Press, Cambridge, 1980.

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

Cutland, N., Computability: an Introduction to Recursive Function Theory, Cambridge University Press, London, England, 1980.

Davis, J., Herman, Garfield at Large, Ballantine Books, New York, 1980.

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.

Edelsbrunner, H., Algorithms in Combinatorial Geometry, Springer-Verlag, Germany, 1987.

Einstein, A., Relativity: the Special and General Theory, Crown Publishers Inc., New York, Fifteenth Edition, 1952.

Eisenbach, S., Functional Programming: Languages, Tools, and Architectures, Ellis Horwood Limited, England, 1987.

Even, S., Graph Algorithms, Computer Science Press, Potomac, Maryland, 1979.

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

Gafni, E., and Santoro, N., Distributed Algorithms on Graphs, Carleton University Press, Ottawa, Canada, 1986.

Gamow, G., One, Two Three, Infinity!, ?.

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.

Garey, M., and Johnson, D., Computers and Intractability: a Guide to the Theory of NP Completeness, W. H. Freeman and Company, San Francisco, California, 1979.

Glashow, S., and Bova, B., Interactions: A Journey Through the Mind of a Particle Physicist and the Matter of This World, Warner Books, New York, 1988.

Gribbin, J., In Search of Schrodinger's Cat: Quantum Physics and Reality, Bantam Books, New York, 1984.

Gribbin, J., In Search of the Big Bang: Quantum Physics and Cosmology, Bantam Books, 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.

Hawking, S., A Brief History of Time: From the Big Bang to Black Holes, Bantam Books, New York, 1988.

Hofstadter, D., Godel, Escher, Bach: An Eternal Golden Braid, Bantam Books, New York, 1979.

Hofstadter, D., Mathemagical Themas: Questing for the Essence of Mind and Pattern, Bantam Books, New York, 1985.

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.

Larson, G., Beyond the Far Side, Andrews, McMeel, and Parker, Kansas City, 1983.

Larson, G., Bride of the Far Side, Andrews, McMeel, and Parker, Kansas City, 1984.

Larson, G., In Search of the Far Side, Andrews, McMeel, and Parker, Kansas City, 1984.

Larson, G., The Far Side Observer, Andrews, McMeel, and Parker, Kansas City, 1984.

Larson, G., Valley of the Far Side, Andrews, McMeel, and Parker, Kansas City, 1985.

Larson, G., It Came From the Far Side, Andrews, McMeel, and Parker, Kansas City, 1986.

Larson, G., Hound of the Far Side, Andrews, McMeel, and Parker, Kansas City, 1987.

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

Pagels, H., Perfect Symmetry: The Search for the Beginning of Time, Bantam Books, New York, 1985.

Polya, G., How to Solve It.

Polya, G., Tarjan, R., and Woods, D., Notes on Introductory Combinatorics, Birkhauser, Boston, 1983.

Preparata, F., and Shamos, M., Computational Geometry: an Introduction, Springer-Verlag, New York, 1985.

Purdum, P., and Brown, C., The Analysis of Algorithms, Holt, Rinehart, and Winston, New York, 1985.

Riordan, M., The Hunting of the Quark: A True Story of Modern Physics, Simon and Shuster, Inc., New York, 1987.

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

Robins, G., On Style, Expressibility, and Efficiency in Functional Programming Languages, The UCLA Computer Science Department Quarterly, UCLA, Fall-Winter, pp. 105-121, 1988.

Rogers, H., Theory of Recursive Functions and Effective Computability, McGraw-Hill, 1967.

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

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

Sedgewick, R., Algorithms, Addison-Wesley, New York, 1988.

Tarjan, R., Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, 1983.

Trump, D., and Schwartz, T., The Art of the Deal, Random House, New York, 1987.

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

Unger, J., Herman, You Can Get in the Bathroom Now, Andrews, McMeel, and Parker, Kansas City, 1987.

11. Table of Contents

1.........Introduction........1
2.........The Course Reputation and Typical Student Background........2
3.........Some Teaching Techniques........3
3.1...........Using Colored Chalk........3
3.2...........Giving Students a Copy of Slides........4
3.3...........Cartoons as a Tool to Improve Morale........5
3.4...........Reaching the Students Through Familiar Examples........5
3.5...........Calling on the Class and on Individual Students........7
3.6...........Philosophy Presented During the First Class Meeting........7
3.7...........Informal Course Evaluations........8
3.8...........Homework and Examination Solutions........9
3.9...........Giving the Students My Home Phone Number and Address........10
3.10..........Encouraging Students to Exchange Information........10
3.11..........Public Speaking........10
3.12..........Extra Review Sections........11
3.13..........Off-the-wall Questions........12
3.14..........Open Book Examinations........12
3.14.1...........Selling Examinations Hints vs."Double Jeopardy"........13
3.15..........Extra-Credit Problems........14
3.16..........The Lack of Initiative and Curiosity in Students........14
3.17..........Other Readings........15
4.........A Self-Printing Program and Other Extra-Credit Problems........15
5.........Sample Midterm and Final Examination Questions........18
6.........The Textbook Used in this Course........21
6.1...........Further Reading........22
7.........Some Proposals to Improve the Status Quo........22
7.1...........Keeping the Students Informed........22
7.2...........Permanent Student Computer Accounts........23
7.3...........A Proposal for Breaking CS181 Into Two Courses........23
7.4...........Is Infinity an Integer?........24
7.5...........A Proposal for a Brand New Undergraduate Course........25
8.........Summary........25
9.........Acknowledgements........26
10.........Bibliography........26
11.........Table of Contents........30