Chapter 9

Theory and Practice, IV

[SLIDE 0 to be shown during introduction of the speaker]

Slide 0.

Good morning! I want to welcome you all to the San Francisco Bay area and to nearby Silicon Valley, where I live at Stanford University. [SLIDE 1] In recent years the people around here have been taking advantage of an idea that originated, I think, in the international road signs that have spread from Europe to the rest of the world: the idea of *icons*, as graphic representations of information. Icons have now become so pervasive, in fact, that I think people might soon be calling this place Silly Icon Valley.

The title of my talk this morning is "Theory and Practice," and in order to be up-to-date I want to begin by showing you two icons that might make suitable pictographs for the notions of theory and practice.

149

the given word, as shown here; and then we calculate the maximum of all the numbers that occur between each pair of adjacent letters. If the resulting number is odd, it represents a place to break the word; but if it is even, we don't insert a potential hyphen.

$$h \; y \; p \; h \; e \; n \; a \; t \; i \; o \; n$$

$$_0h_0y_3p_0h_0$$
$$_0h_0e_2n_0$$
$$_0h_0e_0n_0a_4$$
$$_0h_0e_0n_5a_0t_0$$
$$_1n_0a_0$$
$$_0n_2a_0t_0$$
$$_1t_0i_0o_0$$
$$_2i_0o_0$$

$$_0h_0y_3p_0h_0e_2n_5a_4t_2i_0o_0n_0$$
$$h \; y\text{-}p \; h \; e \; n\text{-}a \; t \; i \; o \; n$$

Slide 16.

The beauty of Liang's method is that it is highly accurate, it runs fast, and it takes up very little space inside a computer. Moreover, it works with all languages, not just English: Successful sets of patterns have already been found for French, German, Italian, Spanish, Portuguese, Swedish, Icelandic, Russian, and other languages. Thus, it is a uniform method able to support international communication. Liang discovered this unified method only after considerable theoretical study of other techniques, which solved only special cases of the problem. And his practical work also had a theoretical payoff, because it led him to discover a new kind of abstract data structure called a dynamic trie, which has turned out to be of importance in other investigations. I think it's reasonable to compare this with some of Gauss's work; Gauss worked with masses of numerical data while Liang worked with masses of linguistic data, but in both cases there was an enrichment of practice that would have been impossible without the theory and an enrichment of theory that would have been impossible without the practice.

[NO SLIDE] That was an example from TEX; let me give another example, this time from METAFONT. One of the key problems of discrete geometry is to draw a line or curve that has approximately uniform thickness although it consists entirely of square black-or-white pixels. The obvious way to solve this problem is to draw a solid line of the desired thickness, without thinking about the underlying raster, and then to digitize the two edges of that line separately and fill in the
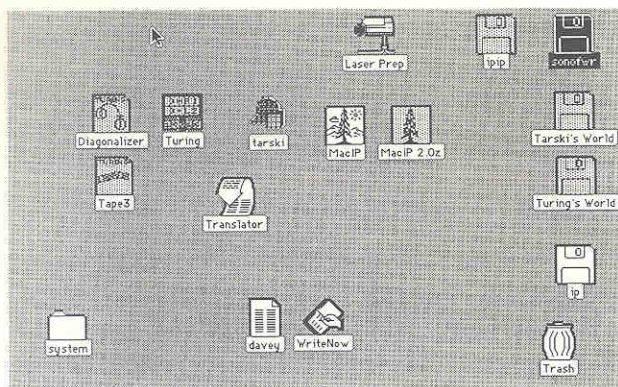
Slide 1.

I didn't have any trouble finding such images, because the reference section of our local telephone directory contains lots of icons these days. Looking at those pages, I immediately spotted an image that seems just right to depict theory [SLIDE 2]: A light bulb of inspiration. And what

Slide 2.

about practice? Right next to that light bulb in the phone book was another suitable image [SLIDE 2 + 3]: A hand carrying a briefcase.

Slide 2 + 3.

Theory and *Practice*. Both of these English words come from the Greek language, and their root meanings are instructive [SLIDE 4]. The Greek $\theta\epsilon\omega\rho\iota\alpha$ means seeing or viewing, while $\pi\rho\alpha\kappa\tau\iota\kappa\eta$ means doing, performing. We owe to ancient Hellenic philosophers the revolutionary notion of theory as the construction of ideal mental models that transcend concrete physical models. They taught us systems of logic by which intuitive assumptions and rules of inference can be made explicit; therefore significant statements can be rigorously and conclusively

Θεωρία πρακτική

seeing   doing

Slide 4.

proved. Throughout the ages, practitioners have taken such theories and applied them to virtually every aspect of civilization. [SLIDE 4+5] Thus, we can say that theory is to practice as rigor is to vigor.

Θεωρία πρακτική

seeing   doing

rigor   vigor

Slide 4 + 5.

Theory *and* Practice [SLIDE 6 + 7]. The English word 'and' has several meanings, one of which corresponds to the mathematical notion of 'plus'. When many people talk about theory and practice, they are thinking about the sum of two disjoint things. In a similar way, when we refer to 'apples and oranges', we're talking about two separate kinds of fruit.

Slide 6 + 7.

Slide 6 + 8.

[SLIDE 6 + 8] But I wish to use a stronger meaning of the word 'and', namely the logician's notion of 'both and', which corresponds to the *intersection* of sets rather than a sum. The main point I want to emphasize this morning is that both theory and practice can and should be present simultaneously. Theory and practice are not mutually exclusive; they are intimately connected [SLIDE 6 + 8 + 9]. They live together and support each other.
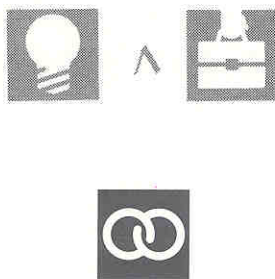
Slide 6 + 8 + 9.

This has always been the main credo of my professional life. I have always tried to develop theories that shed light on the practical things I do, and I've always tried to do a variety of practical things so that I have a better chance of discovering rich and interesting theories. It seems to me that my chosen field, computer science — information processing — is a field where theory and practice come together more than in any other discipline, because of the nature of computing machines.

I came into computer science from mathematics, so you can suspect that I have a soft spot in my heart for abstract theory. I still like to think of myself as a mathematician, at least in part; but during the 1960s I became disenchanted with the way mathematics was going. I'll try to explain why, by saying a few things about the history of mathematical literature [SLIDE 10]. The first international journal of mathematics was founded in 1826 by a man named August Leopold Crelle. I think its title was significant: *Journal für die reine und angewandte Mathematik*, a journal for pure and applied mathematics. In many people's

eyes, 'pure mathematics' corresponds to 'theory' and 'applied mathematics' corresponds to 'practice'; so there we have it, theory and practice. This venerable journal is still being published today, currently in volume number 398. [SLIDE 11] Another journal with the equivalent title in French began publication ten years later. This one too has continued to

Journal für die reine und angewandte Mathematik

gegründet 1826 von

August Leopold Crelle

fortgeführt von
C. W. Borchardt, K. Weierstrass, L. Kronecker, L. Fuchs, K. Hensel, L. Schlesinger, H. Hasse, H. Rohrbach

gegenwärtig herausgegeben von

Simon Donaldson · Willi Jäger
Martin Kneser · Horst Leptin · Samuel J. Patterson
Peter Roquette · Michael Schneider

unter Mitwirkung von
J. Arthur (Toronto), T. tom Dieck (Göttingen), O. Forster (München), P. R. Halmos (Santa Clara), F. Hirzebruch (Bonn), R. Howe (New Haven), Y. Ihara (Tokyo), H. Koch (Berlin), J. Lindenstrauss (Jerusalem)

JRMAA8

Band 398

Walter de Gruyter · Berlin · New York 1989

TOME 68 FASCICULE 1 1989
NEUVIÈME SÉRIE

JOURNAL
DE
MATHÉMATIQUES
PURES ET APPLIQUÉES

gauthier-villars

Slide 10.   Slide 11.

the present day, and both journals still mention both pure and applied mathematics in their titles. But there was a time when the only applied mathematics you could find in these journals consisted of applications to pure mathematics itself.

[NO SLIDE] When theory becomes inbred — when it has grown several generations away from its roots, until it has completely lost touch with the real world — it degenerates and becomes sterile. I was attracted to computer science because its theory seemed much more exciting and interesting to me than the new mathematical theories I was hearing about in the 60s. I noticed that computer science theory not only had a beautiful abstract structure, it also answered questions that were relevant to things I wanted to do. So I became a computer scientist.

History teaches us that the greatest mathematicians of past centuries combined theory and practice in their own careers. For example, let's consider Carl Friedrich Gauss, who is often called the greatest

Slide 12.

Slide 13.

Slide 14.

mathematician of all time because of the deep theories he discovered. [SLIDE 12] Here is an excerpt from one of his diaries; Gauss left behind thousands of pages of detailed computations. His practical work with all these numbers led him to discover the method of least squares and the so-called Gaussian distribution of numerical errors. [SLIDE 13] He also made measurements of the earth and drew this map as a basis for land surveys in parts of Germany, the Netherlands, and Denmark. [SLIDE 14] His study of magnetism led him to publish a series of world maps such as this one. Thus Gauss was by no means purely a theoretician. His practical work went hand in hand with his theoretical discoveries in geometry and physics.

One of the main reasons I've chosen to speak about Theory and Practice this morning is that I've spent the past 12 years working on a project that has given me an unusual opportunity to observe how theory and practice support each other. [NO SLIDE] My project at Stanford University has led to the development of two pieces of software called TEX and METAFONT: TEX, a system for typesetting, and METAFONT, a system for generating alphabets and symbols. [SLIDE 15] Here are the icons for TEX and METAFONT.
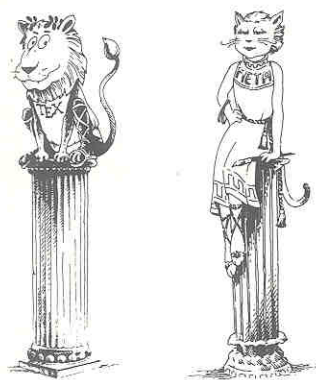
Slide 15.

Throughout my experiences with the TEX project, I couldn't help noticing how important it was to have theory and practice present simultaneously in equal degrees. One example of this is the method for hyphenating words that was discovered by my student Frank Liang. [SLIDE 16] Suppose we want to find permissible places to break up the word 'hyphenation'. Liang's idea is to represent hyphenation rules by a set of patterns, where each pattern is a string of letters separated by numerical values. We find all the patterns that appear as substrings of

region inside. But this obvious approach doesn't work. [SLIDE 17] For example, here are two straight lines of slope 1/2 and thickness 1 that were drawn by the obvious method. When we digitize the two edges and fill the inner region, [SLIDE 17 + 18] the lower line comes out 50% darker than the upper one, because it happens to fall in a different place on the raster.
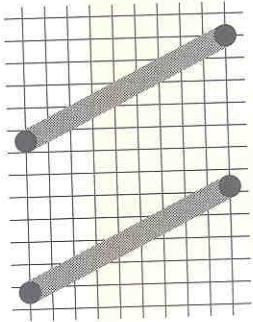
Slide 17.
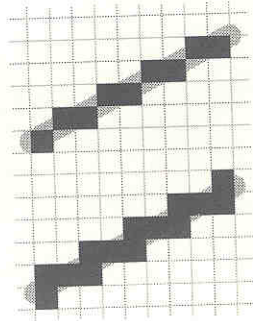
Slide 17+18.

There's a better way, which I'll call the diamond method. Imagine a diamond-shaped pen tip, one pixel tall. [SLIDE 19] Draw a line or curve with this pen, and then digitize the edges. Now you get a line or curve
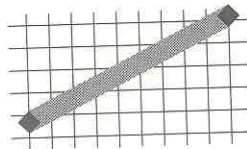
Slide 19.

that has nearly uniform thickness, regardless of where it falls on the raster. The "obvious" method I mentioned before corresponds to lines that you would draw when the tip of the pen is a circle of diameter 1 instead of a diamond. The track of a circular pen nib does not digitize well, but the track of a diamond-shaped pen nib does.

[SLIDE 20] Here's another example, using circular and diamond-shaped pens to draw a circle whose radius is 7.5. In this case the circular pen gives a digital track [SLIDE 20 + 21] that's noticeably heavier when it travels diagonally than when it is travelling horizontally or vertically. The diamond pen gives a much nicer digital circle without such glitches.

My student John Hobby found a beautiful way to extend these ideas to curves of greater thickness. [SLIDE 22] Here, for example, is an octagon-shaped pen nib that turns out to give the best results when
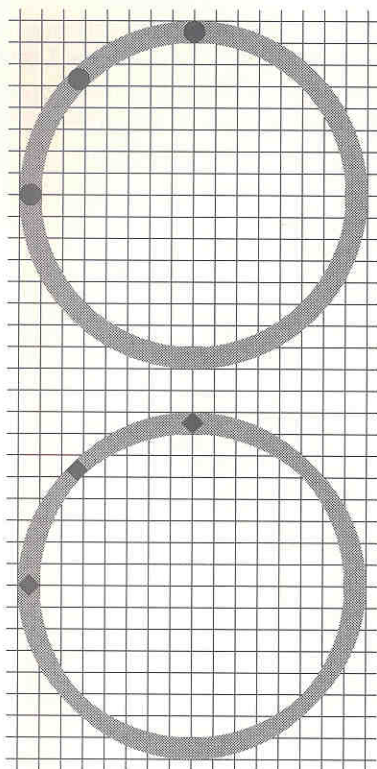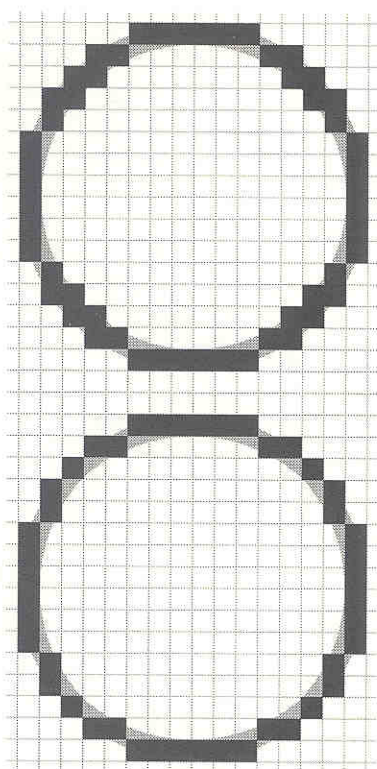
<div align="center">Slide 20.   Slide 20+21.</div>

you want to draw curves that are slightly less than 3 pixels thick. Hobby developed METAFONT's polygonal method of curve drawing by creating a truly elegant combination of number theory and geometry. His work is one of the nicest blends of theory and practice I have ever seen: It's

<div align="center">Slide 22.</div>

a case where deep theoretical results have made an important contribution to a practical problem, and where the theory could only have been discovered by a person who was thoroughly familiar with both the practice of digitization and with mathematical theories that had been developed for quite different problems.

[NO SLIDE] I want to mention also a third example, This one isn't as important as the other two, but I can't resist telling you about it because I just thought of it four days ago. I decided last week to make some extensions to TEX so that the system will be more useful for languages other than English. The new standard version of TEX will support 8-bit character sets instead of only the 7-bit ASCII code. Furthermore it will allow you to hyphenate words from several different languages within the same paragraph, using different sets of patterns for each language. One of the new features will be an extension of the mechanism by which TEX makes ligatures in the text, and that's the method I want to explain now.

Suppose two letters occur next to each other in a word that is to be typeset by the computer; I'll call those letters α and ω. [SLIDE 23] The present version of TEX allows the font designer to say that the letters α and ω should be replaced by a ligature, say λ. This is the way, for example, that an 'f' followed by an 'i' is converted into a symbol for 'fi' that looks better.

$$\alpha\omega \to \lambda$$
$$\mathrm{fi} \to \text{fi}$$

Slide 23.

The new version of TEX will extend this mechanism as follows. A new letter λ will be inserted between α and ω, and the original letters might still remain. [SLIDE 24] There are nine cases, depending on what letters are kept and depending on where TEX is instructed to look next for another possible ligature. (The little caret between letters in this picture shows where TEX is focusing its attention.) The first case here shows the old ligature mechanism; the middle seven cases are new; and the bottom case is the normal situation where no ligature is to be inserted.

$$\alpha_{\wedge}\omega \to \lambda_{\wedge}$$
$$\alpha_{\wedge}\omega \to \alpha_{\wedge}\lambda$$
$$\alpha_{\wedge}\omega \to \alpha \ \lambda_{\wedge}$$
$$\alpha_{\wedge}\omega \to \lambda_{\wedge}\omega$$
$$\alpha_{\wedge}\omega \to \lambda \ \omega_{\wedge}$$
$$\alpha_{\wedge}\omega \to \alpha_{\wedge}\lambda \ \omega$$
$$\alpha_{\wedge}\omega \to \alpha \ \lambda_{\wedge}\omega$$
$$\alpha_{\wedge}\omega \to \alpha \ \lambda \ \omega_{\wedge}$$
$$\alpha_{\wedge}\omega \to \alpha \ \omega_{\wedge}$$

Slide 24.

This new mechanism has a potential danger: A careless user can now construct ligature instructions that will get TₑX into an infinite loop. [SLIDE 25] For example, suppose we have the four rules shown at the top of this illustration. Then when 'a' is followed by 'z', the rules set off a chain reaction that never stops.

$$a_\wedge z \to a\ b_\wedge z$$
$$b_\wedge d \to b\ c_\wedge d$$
$$b_\wedge z \to b_\wedge d\ z$$
$$c_\wedge d \to a_\wedge$$

$$a_\wedge z \to a\ b_\wedge z$$
$$\to a\ b_\wedge d\ z$$
$$\to a\ b\ c_\wedge d\ z$$
$$\to a\ b\ a_\wedge z$$
$$\to a\ b\ a\ b_\wedge z$$
$$\to \quad \cdots$$

Slide 25.

To minimize this danger, I need an algorithm that will take a given set of ligature rules and decide if it can spawn an infinite loop. And that's where computer science theory comes to the rescue! [SLIDE 26] We can define a function f on letter pairs according to the nine ligature possibilities, as shown here. This definition is recursive. It's not hard to see that f is well defined if and only if there are no infinite ligature loops; we can understand this from the theory of deterministic pushdown automata. (The value of $f(\alpha, \omega)$ represents the letter just preceding the cursor when the cursor first moves to the right of the original ω.) And we can check whether or not f is well defined by using a small extension of an important algorithm called depth-first search.

$$\alpha_\wedge \omega \to \lambda_\wedge \qquad\qquad f(\alpha, \omega) = \lambda$$
$$\alpha_\wedge \omega \to \alpha_\wedge \lambda \qquad\qquad f(\alpha, \omega) = f(\alpha, \lambda)$$
$$\alpha_\wedge \omega \to \alpha\ \lambda_\wedge \qquad\qquad f(\alpha, \omega) = \lambda$$
$$\alpha_\wedge \omega \to \lambda_\wedge \omega \qquad\qquad f(\alpha, \omega) = f(\lambda, \omega)$$
$$\alpha_\wedge \omega \to \lambda\ \omega_\wedge \qquad\qquad f(\alpha, \omega) = \omega$$
$$\alpha_\wedge \omega \to \alpha_\wedge \lambda\ \omega \qquad\qquad f(\alpha, \omega) = f\big(f(\alpha, \lambda), \omega\big)$$
$$\alpha_\wedge \omega \to \alpha\ \lambda_\wedge \omega \qquad\qquad f(\alpha, \omega) = f(\lambda, \omega)$$
$$\alpha_\wedge \omega \to \alpha\ \lambda\ \omega_\wedge \qquad\qquad f(\alpha, \omega) = \omega$$
$$\alpha_\wedge \omega \to \alpha\ \omega_\wedge \qquad\qquad f(\alpha, \omega) = \omega$$

Slide 26.

I like this example not only because it gives an efficient, linear-time algorithm for testing whether or not a ligature loop exists. This practical problem also showed me how to extend the theory of depth-first search in a way that I hadn't suspected before. And I have a hunch the extended theory will have further ramifications, probably leading to additional applications having nothing to do with ligatures or typesetting.

What were the lessons I learned from so many years of intensive work on the practical problem of setting type by computer? One of the most important lessons, perhaps, is the fact that SOFTWARE IS HARD [SLIDE 27]. From now on I shall have significantly greater respect for every successful software tool that I encounter. During the past decade

<p style="text-align:center">Software</p>

<p style="text-align:center">is</p>

<p style="text-align:center">HARD</p>

<p style="text-align:center">Slide 27.</p>

I was surprised to learn that the writing of programs for TEX and for METAFONT proved to be much more difficult than all the other things I had done (like proving theorems or writing books). The creation of good software demands a significantly higher standard of accuracy than those other things do, and it requires a longer attention span than other intellectual tasks.

My experiences also strongly confirmed my previous opinion that the best theory is inspired by practice and the best practice is inspired by theory [SLIDE 28]. The examples I've mentioned, and many others, convinced me that neither theory nor practice is healthy without the other.

<p style="text-align:center">The best theory
is
inspired by practice</p>

<p style="text-align:center">~</p>

<p style="text-align:center">The best practice
is
inspired by theory</p>

<p style="text-align:center">Slide 28.</p>

But I don't want to give the impression that theory and practice are just two sides of the same coin. No. They deserve to be mixed and blended, but sometimes they also need to be pure. I've spent many an hour looking at purely theoretical questions that go way beyond any practical application known to me other than sheer intellectual pleasure. And I've spent many an hour on purely practical things like pulling weeds in the garden or correcting typographic errors, not expecting those activities to improve my ability to discover significant theories. [SLIDE 29] Still, I believe that most of the purely practical tasks I undertake do provide important nourishment and direction for my theoretical work; and I believe that the hours I spend contemplating the most abstract questions of pure mathematics do have a payoff in sharpening my ability to solve practical problems.

Slide 29.   Slide 29 + 30.

When I looked for an icon that would be appropriate for 'practice', I was tempted to use another one instead of the briefcase — a symbol for money! [SLIDE 29 + 30] It seems that people who do practical things are paid a lot more than the people who contribute the underlying theory. Somehow that isn't right. The past decade has, in fact, witnessed a very unfortunate trend in the patterns of funding for basic, theoretical research. We used to have a pretty well balanced situation in which both theory and practice were given their fair share of financial support by enlightened administrators. But in recent years, greater and greater amounts of research dollars have been switched away from basic research and earmarked for mission-oriented projects. The people who set the budgets have lost consciousness of the fact that the vast majority of the crucial ideas that go into the solution of these mission-oriented problems were originally discovered by pure scientists, who were working alone, independently, on basic research stimulated by intellectual curiosity. At the present time the scientific community faces a crisis in which

a substantial number of the world's best scientists in all fields cannot get financial support for their work unless they subscribe to somebody else's agenda telling them what to do. We need to go back to a system where people who have demonstrated an ability to devise significant new theories are given a chance to set their own priorities. We need a lot of small projects devised by many independent scientists, instead of concentrating most of our resources on a few huge projects with predefined goals. [SLIDE 29 + 30 + 31] In other words, we need a balance between theory and practice in the budgets for scientific research, as well as in the lives of individual scientists. Otherwise we'll face a big slump in our future abilities to tackle new problems.
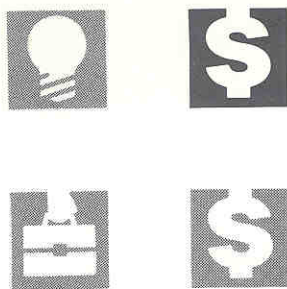
Slide 29 + 30 + 31.

These comments hold true for industry as well as for the university community. Many of the graduates of Stanford's Computer Science Department who have written Ph.D. theses about theoretical subjects have now taken jobs in Silicon Valley and elsewhere; and they have in most cases been able to work with enlightened managers who encourage them to continue doing basic research. I think it's fair to state that these so-called theoreticians are now considered to be among the key employees of the companies for which they work.

Slide 32.

[SLIDE 32] Speaking of key employees reminds me that this is a keynote speech; indeed, this morning is surely the only time in my life when I'll be able to give the keynote address to an IFIP Congress. So I would like to say something memorable, something of value, something that you might not have expected to hear. I thought about David
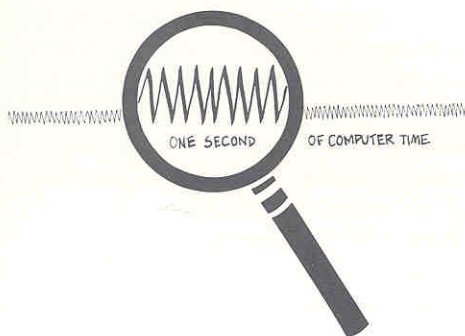
Slide 33.

Hilbert's famous address to the International Congress of Mathematicians in 1900, when he presented a series of problems as challenges for mathematicians of the 20th century. My own goals are much more modest than that; but I *would* like to challenge some of you in the audience to combine theory and practice in a way that I think will have a high payoff.

[SLIDE 33] My challenge problem is simply this: *Make a thorough analysis of everything your computer does during one second of computation.* The computer will execute several hundred thousand instructions during that second; I'd like you to study them all. The time when you conduct this experiment should be chosen randomly; for example, you might program the computer itself to use a random number generator to decide just what second should be captured and recorded.

Many people won't be able to do this experiment easily, because they won't have hardware capable of monitoring its own activities. But I think it should be possible to design some tracing software that can emulate what the machine would have done for one second if it had been running freely.

Even when the machine's instructions are known, there will be problems. The sequence of operations will be too difficult to decipher unless you have access to the source code from which the instructions were compiled. University researchers who wish to carry out such an experiment would probably have to sign nondisclosure agreements in order to get a look at the relevant source code. But I want to urge everyone who has the resources to make such a case study to do so, and to compare notes with each other afterwards, because I am sure the results will be extremely interesting; they will tell us a lot about how we can improve our present use of computers.

I discussed this challenge problem with one of the botanists at Stanford, since I know that biologists often make similar studies of plant and animal life in a randomly chosen region. [SLIDE 34] She referred me to a recent project done in the hills overlooking Stanford's campus, in which all plants were identified in several square blocks of soil. The researchers added fertilizer to some of the plots, in an attempt to see what this did to the plant life. Sure enough, the fertilizer had a significant effect on the distribution of species.

Study site

Studies were carried out on Jasper Ridge Biological Preserve in San Mateo County, Northern California. A serpentine outcrop bisects a low-lying ridge (max. elevation 189 m) situated on the eastern side of the Santa Cruz mountains. The climate is a Mediterranean-type with a mean annual rainfall of 480 mm and a virtually rain-free period from May to September. The serpentine soil is extremely nutrient poor (0.192% N, 0.029% P; C. Chu, unpublished work) and has a Ca Mg ratio of 0.034 and a high Ni concentration (216 ppm, acetic acid extraction; B. Streit, unpublished work). The vegetation has been described by McNaughton (1968) and studied in detail by Hobbs and Mooney (1985). The main annual species present at the study site are *Lasthenia californica* and *Plantago erecta* (authorities given in Table 1). A diverse collection of other annual species is also present, together with several species of perennial bunch grasses, notably *Sitanion jubatum* and *Stipa pulchra*.

Experimental procedure

In November 1982, we set up grids in the serpentine grassland which formed part of a long-term experiment on herbivore exclusion. Each grid measured 1 × 3 m and was subdivided into 12, 50 × 50 cm quadrats. Three sets of 2 adjacent grids were used for the present study, and each set was fenced using 1 m high chicken wire (2 cm mesh) to exclude rabbits and deer. Within each grid 2 adjacent 50 × 50 cm quadrats were chosen at random and supplied with 56 g Osmocote slow release fertiliser (14.0% N as 7.4% NO3, 6.6% NH4; 14.0% P; 14.0% K) in December 1982 and again in 1983.

At the beginning of April in the years 1983–86 we recorded the species present in each fertilised (50 × 50 cm) quadrat and in an equivalent number of unfertilised quadrats (i.e. 12 quadrats each in fertilised and unfertilised). The unfertilised quadrats analysed were located one quadrat away from the fertilised areas to obviate cross contamination.
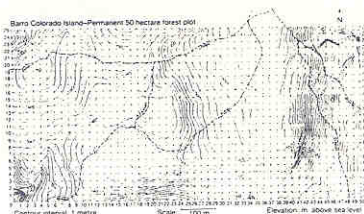

Fig. 3.1. Topographic map of the 50-hectare plot on Barro Colorado Island, Panama (BCI). Contour lines are drawn for each metre of elevational change. Dotted lines mark permanent trails through the plot. Grid marks indicate the corners of 1250, 20×20 metre quadrats. A stream in the lower left corner drains a seasonal swamp (left of centre). Grades on the steeper slopes range between 10 and 20%.

3.4   A neotropical forest case study

In 1980 we began a long-term study of a half-square kilometre plot of tropical forest on Barro Colorado Island (BCI), Panama (Hubbell & Foster, 1983), a 15 km² nature reserve in Gatun Lake. In this plot (Fig. 3.1), we have completely identified, measured, and mapped to the nearest metre all free-standing woody plants with a 1 cm or greater stem diameter at breast height (dbh). There are nearly a quarter of a million trees, saplings, and shrubs in the 50-hectare plot, representing approximately 300 species. The plot is located in 'old-growth' forest (Foster & Brokaw, 1982). Recent palaeoecological surveys of the site reveal that the forest has never been clearcut for agriculture: fossil corn phytoliths and weedy pasture grasses are absent from plot soil cores, in contrast to permanent pre-Columbian settlement areas along the Rio Chagres several kilometres away (D. Piperno, pers. comm.). However, radiocarbon dating of cores indicate that from 500 A.D. until 545 years ago, small temporary clearings were occasionally made in the forest, probably for seasonal hunting camps. Thus the forest has been essentially undisturbed by man for the past 500 years or more.

Slide 34.   Slide 35.

[SLIDE 35] My colleague also told me about another recent experiment in which British researchers identified and counted each tree in a tropical rain forest. About 250,000 trees were enumerated altogether. I imagine a typical computer will execute something like that number of instructions every second, so my specification of exactly one computer second seems to be reasonable in scale.

Here are some of the questions I would like to ask about randomly captured seconds of computation: [SLIDE 36]

- Are the programs correct or erroneous? (I have to report reluctantly that nearly every program I have examined closely during the past thirty years has contained at least one bug.)

- Do the programs make use of any nontrivial theoretical results?

○ Are the programs correct?

○ Do they use existing theory?

○ Would existing theory help?

○ Would new theory help?

Slide 36.

- Would the programs be substantially better if they made more use of known theory? Here I am thinking about theories of compiler optimization as well as theories of data structures, algorithms, protocols, distributed computation, and so on.

- Can you devise new theoretical results that would significantly improve the performance of the programs during the second in question?

In a sense, I'm asking questions something like the botanists considered. I'm asking to what extent computer programs have been "fertilized" by theory, and to what extent such fertilization and cross-pollination might be expected to improve our present situation. I hope many of you will be inspired to look into questions like this.

[SLIDE 37] In conclusion, let me encourage all of you to strive for a healthy balance between theory and practice in your own lives. If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice.

The theme of this year's IFIP Congress is Better Tools for Professionals. I believe that the best way to improve our tools is to improve the ways we blend Theory with Practice. Thank you for listening.

Acknowledgment
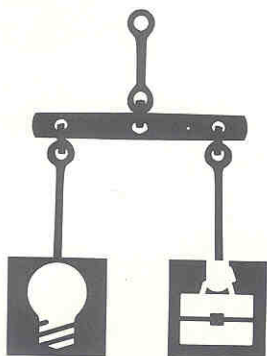
BETTER TOOLS FOR PROFESSIONALS

Slide 37.

References

[1] Pacific Bell SMART Yellow Pages for Palo Alto, Redwood City and Menlo Park, May 1989–1990, page A47.

[2] C. F. Gauss's *Werke*, volumes 10^1 and 12.

[3] Franklin Mark Liang, "Word hy-phen-a-tion by com-pu-ter," Ph.D. dissertation, Stanford University, September 1985.

[4] John Douglas Hobby, "Digitized brush trajectories," Ph.D. dissertation, Stanford University, September 1985.

[5] Donald E. Knuth, *Computers & Typesetting*, five volumes (Reading, Massachusetts: Addison–Wesley, 1986).

[6] Richard J. Hobbs, S. L. Gulmon, V. J. Hobbs, and H. A. Mooney, "Effects of fertilizer addition and subsequent gopher disturbance on a serpentine annual grassland community," *Œcologia* **75** (1988), 291–295.

[7] Stephen P. Hubbell and Robin B. Foster, "Canopy gaps and the dynamics of a neotropical forest," in *Plant Ecology*, ed. by Michael J. Crawley (Blackwell Scientific, 1986).