GEOMETRY

LECTURE 6

BETTER PROGRAMMING

by

Harold Thimbleby
Gresham Professor of Geometry

13 May 2004

BETTER PROGRAMMING

Harold Thimbleby
Gresham Professor of Geometry

Introduction

This year's series of Geometry Lectures have looked at all aspects of computer science, from how it works, how we can teach it better, what the exciting issues are, and what the future research questions should be. We've explored some very exciting areas, and we have covered some of the areas that are exciting, but often misunderstood. Now, in this final lecture of the year, we turn to how the computer science community itself works. Where do all these ideas come from? How reliable are they? And, how can we do even better?

The underlying idea through the series of lectures is that things can be better when we know and understand things objectively about them. For everyday devices, there are theories. My lectures explained some of the theories, and how they can be applied to help make a better world: how they draw attention to design problems, and how they suggest fixes.

A scientific idealist might argue that the best science is objective and from that firm ground we can then build a world more similar to the way we might like it to be. Surely, then, we want firm foundations? We need reliable knowledge. For example, I am interested in how computers can be made easy to use; to work in my area, we need to check the things that are claimed to be 'easy to use,' and we need to know why they are easy to use (or not, as the case may be) so that we can improve the ideas they are based on. If I was interested in the speed of programs, we would need the claims about speeds to be clear, honest, and to be clear about which factors led to which speed concerns. And so on; whatever we are interested in, we need reliable knowledge — which goes behind just what is said in a paper.

But problems come when understanding is subjective, but is still presented as universal, and benefits fewer people than its mask of objectivity might suggest. Subjectivity can be exploited for advantage in many ways, some deliberate, some accidental. The ideal, then, is to approach understanding objectively to make the knowledge of universal value.

What does objectivity mean? At its simplest, objective knowledge does not depend on who or where it is said. It is true any where, any time, any place. It is invariant under a change of context. In contrast, we say things are *relative* if they vary under context change, and more specifically if the context change is a change of person, we call them *subjective*. Thus I like apples, but you may not. So my likes are subjective. If everybody liked apples, it would be an objective fact about humanity, but it is far from that!

I can jump a few feet in the air. So can you. But you can jump higher or not so high, as the case may be. So "how high one can jump" is not objective knowledge. But if you ask a physicist, they will show you an objective fact. It is expressed through an equation. I can jump $h=E/mg$ metres, where E is the energy I have for jumping, m my mass, and g is a constant which is the same for everyone. I can go around and measure E and m and predict how high people can jump. Or I might measure h and m, and work out E to see how physically fit people are. What is g? It varies from place to place. It is lower on the moon, and that allows me to predict that I could jump higher on the moon (in fact, mg is my weight).

What is the difference between measuring my mass m and claiming $h=E/mg$ is objective, whereas — what sounds superficially the same — measuring "my like of apples" (π, say) and (wrongly) claiming that measure is objective? The difference is that π says nothing about anyone or anything other than me; it is not objective knowledge. In contrast, with my weight, I can work out how strong a safety harness should be, I can work out

what sort of parachute to use, and so on — there are even applications for things I have not thought of yet. It is objective knowledge that transfers to many other areas.

So, to summarise, objective knowledge is "invariant under transformation of context," as physicists and some philosophers might like to say. It is true for me, for you, and even for people on the moon. Because of its objective nature, such knowledge allows predictions to be made (such as how high I could jump on the moon), and is thereby very useful. It could even predict how high cats and fleas can jump. On the other hand, that I like apples is not knowledge that is very useful to you for any purpose, unless you invite me round for tea.

Objective knowledge can only be useful to you if you *know it*, of course. Science therefore has a simple system — called peer reviewed publication — that spreads objective knowledge around the community. Scientists publish their knowledge so that it can be checked and used in new contexts. Having other people check it is thought to increase the chances that no mistakes have been made, although of course it doesn't if everybody is prone to make exactly the same mistakes, as has happened. In fact, this happens with programming a lot, and is one of the things that makes the task of building reliable software so very much harder than it looks.

Ideally what scientists publish is objective, or can be checked and refined to improve its objectivity. For example, if I published I can jump 1.5 metres, that is not exciting science, because it is so specific. It is even specific to things I have not told you about, such as how heavy I am. If you want to obtain objective knowledge from the things I say, I have to be very careful to tell you a lot, to enumerate the context as far as I can. Worse, I may not know exactly what to tell you! You might be interested in whether men or women can jump higher; I forgot to say I am a man. Had you noticed? Would you notice?

Finally, objective knowledge is an *ideal*. We can only do our best, and in hindsight — maybe years later — we will realise what we thought was objective was mistaken or too narrow a conception in one way or another. For example, my formula $E=mgh$ gives you the wrong answer for multistage rockets (the invention that made space flight possible), and of course it is also a Newtonian rather than a modern Einsteinian formulation. In computer science, we have Huffman Codes (see my Gresham Lecture, 'Designing mobile phones,' 28 November, 2001) which were thought for 20 years to be minimum codes until an assumption was spotted — an observation that opened up a new spurt of exciting research.

In short, to do good science, we need to balance four key factors:

- Publishing all the facts that determine the knowledge.
- Providing sufficient methodological information that the knowledge can be reproduced. (If it cannot be reproduced, we have a discovery: there is additional context that needs to be controlled — or possibly I omitted something accidentally or fraudulently.)
- Because objective knowledge is an ideal, and we make mistakes, we must try hard to be clear what we are doing and to communicate it accurately. Otherwise people waste time using wrong knowledge.
- And doing all this concisely. To say what the specific knowledge is, part of my contribution is to cull irrelevant knowledge. Otherwise, I might as well post you my video diary and let you work everything out for yourself.

To do all this requires a very peculiar mind, at once both committed and detached. A scientist has to be dedicated to the pursuit of knowledge, yet despite that commitment they have to be able to change their minds when reality has other ideas. However, when we are challenged, it's possible we may not be wrong, but that somebody else is (or that we are both wrong!). We have to balance vigorously defending our ideas against

mistaken attack, yet still be able to graciously admit defeat if necessary. Inevitably, there are some horrible stories of scientists pursuing their own agendas rather than the truth.

If we are lucky, we end up with a community of scientists agreeing on the shared knowledge. Following my simple $E=mgh$ example a bit further, we may eventually come up with even more general ideas of objective knowledge, such as the conservation of energy, rather than its specific applications to human jumping.

We now have a framework to discuss how science progresses. Richard Feynman spoke eloquently on these issues when he talked about the dangers of what he called *Cargo Cult Science* — seeming to do the right things (having a cargo cult), but missing out on the deeper understanding of what science is really about (not getting the airplanes to land). He summarised achieving the key points above as requiring *radical integrity* [see further reading for reference details]. It is so easy to be more relaxed, and talk about knowledge that you wished was true, or talk not quite accurately, rather than talk about knowledge you are certain is true.

Before proceeding, I want to mention two ways that computer science is different from natural science.

First, in a natural science such as physics, we have a sense in which the world is out there, fixed, and full of laws and facts we can discover. In computer science, which is mostly a science of artifacts, there is no world out there until we build it. The 'facts' change daily. Computers are faster every day, and the time it takes to do something is not an objective fact as such. My programming language (and my programming skills), operating system, hardware, disc capacity, and so on, are all different from yours. If we are not careful, many or most results in computer science will be contingent and subjective, and thereby not generally very useful.

Secondly, because computer science creates new things for us, there is also an important role for envisionment. I may have a great idea about how things could be. This is surely not objective knowledge! But it may serve as an inspiration to you or someone else to make it into knowledge. A lot of work is required, and it may be beyond me to achieve my dreams. But telling you my dreams is worthwhile, and may give you the hope of reproducing them objectively. The danger arises when I am not completely clear where objectivity ends and my dreams begin. Again, it takes a radical honesty. Envisionment is *not* science, but it may motivate scientists. I think research journals should clearly distinguish between articles that are envisionment and articles that exhibit work that should really work.

If we want to do better as scientists, building on objective knowledge, we need to be more careful than natural scientists need to be. When I do a computer science experiment, it is intrinsically harder to reproduce than when I do a physics experiment. On the other hand, fortunately, much of what I do as a computer scientist is written down as instructions to computers. In principle, then, what I do is *very* easily reproduced — it's just a matter of getting hold of the program source code (and a computer to run it on).

Computer scientists write programs and explain programs — whether to document them for other programmers, to explain them in the computer science literature, or to write manuals, or to provide help for users. As Hal Abelson and the Sussmans put it in their classic book *Structure and Interpretation of Computer Programs*, "programs must be written for people to read, and only incidentally for machines to execute." Alan Perlis (who wrote the preface for their book) wrote much earlier in 1966 of a "firm conviction that fluency in writing algorithms for one another and reading those written is a fundamental property of a professional…"

Publishing code reliably means that people can use the code directly and benefit from its correctness; it also means that people can independently check its correctness, efficiency, portability and so on. Informal means of publishing code, particularly using pseudo-code, are inadequate; errors are spread in the literature, work must be unnecessarily duplicated, and when an error is detected one does not know whether this is caused by a fault in the original work, a fault in authoring the paper, a fault in its printing, or a fault in understanding and implementing its ideas.

Evidence

After this philosophical introduction, I am going to turn to some evidence about how computer science research is done. We will then return to the question how we can make computer science more objective.

Computers are magical devices, that inspire and stimulate our imagination. You only have to watch *Star Trek*, or any movie, to see how we think computers will become. They can do anything. Certainly, today's research is aiming in that sort of direction, but an important question is how much of today's work is fact and how much of it has been creative with the facts.

There is an interesting anecdote in the excellent little book *Some Time with Feynman* [see further reading], which recounts how a physicist had exaggerated his computer results:

Constantine's claim to fame was his computer calculation … but there was a rumour going around that Constantine did not translate the problem to the computer in an honest way. "What's the big deal?" Constantine said. "I used what I knew to improve my computer model. Everybody does that." … I told Richard Feynman. He just shrugged. I thought he'd say, "What a louse! He did it because he thought what was important was success, not discovery." But Feynman replied, "Hell no. I'm not going to psychoanalyze the guy. But what should bother you as much as whether or not your friend fudged his work is that a lot of people read it and couldn't tell the difference. There are so many people out there not being skeptical, or not understanding what they are doing. They're all just following along. That's what we have — too many followers, too few leaders."

A survey

How much of our research in computing is "fudged" and how much is real? Feynman thinks there are not enough people being skeptical, and Constantine thinks everyone "improves" their results. Does it matter? — computers are hugely successful, whether or not you side with Feynman or with Constantine.

In 2003 the *Times Higher Education Supplement* asked me to write a review for the *Journal of Machine Learning Research*, which had been founded in 2001. The *Times Higher* is a widely read, British non-technical weekly newspaper, but the *Journal of Machine Learning Research* is highly technical. What was I to do? The journal in its own words is "as an international forum for the electronic and paper publication of high-quality scholarly articles in all areas of machine learning." Machine learning is a fascinating and important area of computer science: basically, can computers learn? It is both a theoretical and practical area, where computer programs are developed and run on data to see what they can learn. Although there are enormous and widespread practical applications for machine learning, from labour negotiation, medical treatment, agriculture, to generally making computers more 'intelligent.' There are applications in machine learning for countering terrorism. Machine learning makes a difference, and makes a lot of money worldwide.

But what is the general interest for *Times Higher* readers? I wondered how I could review a technical journal even in such an interesting subject for the broad audience of the *Times Higher*.

The bulk of the journal's papers are devoted to discussing and evaluating computer based learning methods. I thus became interested to see how ideas talked about in the journal actually worked, because that's really the whole point. So, as the journal is available on-line, I looked at every paper and then emailed all the authors to ask them about their systems. Maybe I would get some ideas for my review. After a few weeks I had 111 replies. The enthusiasm of authors for their work was impressive; in fact, I had replies covering every paper published. I drafted a review once I had something to talk about from all the feedback I got, and I then bounced the draft review off the editorial board and the authors again.

I had asked whether the system described in each paper was successful. Of course, some papers were theoretical; so some replies said my question was irrelevant. Some people were very enthusiastic about their work and sharing it with me. However, over a third specifically said their systems were unavailable for some reason. Their systems were private, commercial confidential, or incomplete in some way.

Here are some quotes from replies I got:[1] "Unfortunately, I do not have the system in a state where I can give it away right now ››› We don't have the data ready to be published." (Both quotes are years after the publication of the papers.) Further quotes are quite revealing about the authors' attitudes: "The system is a research prototype developed in my group, and is not appropriate for public dissemination ››› The implementations we had were very much 'research code,' and are not suitable for public consumption." Somehow research, even stuff published in a refereed research journal, isn't always considered public! (By 'public' these authors meant other researchers.)

My informal survey thus suggested some authors have a relaxed regard for the standard scientific norms and virtues: reproducibility, testability, and availability of data, methods and programs — basically, the openness and attention to detail that supports other researchers.

Since some authors pointed out that it might have been unfair to 'pick' on *JMLR* (implying that other journals might be worse!), more recently, for this lecture, I did a survey of papers in the two highly-regarded journals *Software—Practice & Experience*, and the *Computer Journal*. As its name implies, *Software—Practice & Experience* tends to publish articles about programs that work (so I thought this journal should do much better than the *JMLR*) and the *Computer Journal* publishes one of the widest ranges of articles, covering all aspects of computer science. By picking the *Computer Journal*, I hoped to get a more balanced view than *Software—Practice & Experience* might result in (*SP&E* presumably selects papers on some criteria of how effective they are).

I surveyed from 2001 for *Software—Practice & Experience* (218 papers) and from 2000 for the *Computer Journal* (201 papers), both to the current issue in 2004. I excluded a couple of special issues and papers, which for some reason did not provide email contacts. I did this by writing a Perl script that trawled the publisher's web pages. I then read a sample of papers, composed a questionnaire (this time saying what the survey was about, which my questions to *JMLR* authors had not), and mailed it to a sample of 20 authors. From their responses, I refined the survey questions and mailed to another 20, and refined it again.[2]

To reduce email clog, I only emailed one author (the corresponding author if I could tell who it was, otherwise the first author) from each paper, whereas for *JMLR* I had emailed all authors.

I analysed the responses, some of which were more-or-less along the lines of "I won't answer this question." Of the responses I could safely classify: for the *Computer Journal*, 26% replies said the program code was relevant to the work, but it was not available as described; for *Software—Practice & Experience* the answer was, surprisingly, 45%. These results compare with the *JMLR* figure of 38%. A χ^2 test suggests there is no significant difference between these results; they are samples from the same population.[3]

There're dangers in my method. For example, perhaps respondents are confessional, and there is a tendency for me receive more responses from guilty parties? Or perhaps people who write papers and have never had any interest shown in them are overjoyed to get my query: if so, I'd tend to get replies over-representing bad papers. But I think if an author believes in reproducibility they would be *more* likely to be the

[1] I have fixed typos and anonymised all quotes by paraphrasing identifying remarks. For readability, quotes are separated by '›››' marks.

[2] As an aside, I am interested in how inconsistent the publishers' web sites are: they obviously generate them from databases, yet the format changes from year to year. I wonder why the entire web site is not generate consistently — it seems harder to me to do it in bits and pieces! There are also simple errors like Latex code remaining in the HTML, which a proper database, properly done would have sorted out.

[3] The χ^2 test (and the term standard deviation, etc) was invented by Karl Pearson, who was Gresham Professor of Geometry in the 1890s (coincidentally, he was also a professor at UCL, where I am now a professor too). Before Pearson, people would draw graphs (if at all) and merely assert their results were as expected. Pearson's paper on the χ^2 test effectively founded modern statistics, bringing objectivity into empirical work.

sort of person who would reply clearly to any query about their work. Overall, I guess I am under-estimating the problem. The fact that three different journals are involved, and two quite different surveys give the same results suggests the results are robust. Nevertheless, there is scope for further work on the issue — I've put a brief list of thoughts at the end of these lecture notes.

Rather than pick over the statistics, then, it is probably more insightful to review some of the respondents' comments: "We were explaining some situations that were "imaginary" … the system only copes with simple classes, and not with the whole problem. ››› The system is not publicly available. We were thinking about solving some bugs, developing a stable version and making it available. Unfortunately, other responsibilities have distracted us. ››› It's no longer supported and was developed as a proof-of-concept software engineering prototype. ››› Actually the system is not implemented! I simply applied the method by-hand. ››› It's buggy. ››› The code may or may not still exist. ››› It's hard to set up and use — it was never intended for release. ››› We are no longer supporting it … I believe it is no longer downloadable. ››› I'm not sure the version will even compile."

There were a few comments about obsolescence: "The paper was published quite a long of time ago, please, leave me some days to see if I can find the code. ››› The work is not reproducible because it was run on a specific machine. That equipment is now obsolete." These are comments for papers published in January 2001 and February 2002 respectively!! I'm not sure whether computer science really moves on this fast, or whether it's just an excuse. Nevertheless it is a potential problem that all authors face.

And a comment I liked: an author who replied to me had obviously been prompted by my survey to get his student to make the code available. He copied this email to his student to me "The code for your thesis prototype should be put into <a web site>"

But we work under pressures: "Like everyone else, we are under pressure to publish regularly; inevitably therefore, quality suffers, and it is hard to ensure that academic honesty does not suffer as well." Then there is the problem of who owns the work. "We have not had the time to turn our experimental code into something other people can use (and anyway our employers wouldn't like to see things given away). ››› All of the code is a proprietary trade secret and is not available to be published. ››› The models and codes will be available [but they aren't now] at the project web site or via ftp after the project finishes and is approved by the state commission committee. ››› Some of the "tricky" stuff is deliberately not being published as some of the work is being patented." Certainly there needs to be a balance between science and protecting intellectual property; it's a big problem, as turning research ideas into code that really works might involve a company that then owns it.

Here's another problem: when papers are published based on work done by a student. "Since my former student (and the co-author) was the primary "do-er" for the tool and experiments, I asked him to provide replies to your questions." (I haven't heard from the student yet; I hope he or she has got a job!) A more interesting problem of the same ilk is: "The algorithms in the paper are available on the web in a student's thesis written in <an obscure language>" — well, that's helpful when the paper was written in English for an English audience! Please don't get me wrong; I understand why it is impractical to translate a thesis into English. But why is a paper published as if it was objective knowledge (with all that that means), when the underlying objective knowledge is inaccessible?

Quotes in favour

Here are some quotes about the state of the art: "The paper we published relies on an academic open-source research project. The code still works and is maintained. Newer versions are published regularly. I think that ideas and envisionment are important, but it should always be possible to the interested reader to get the details and code supporting the main ideas discussed in the research paper. ››› While we do release all our sources, I believe availability of good ideas is more important. ››› In my experience as a researcher, overall real

reproducibility of published computer science research is depressingly low. There are probably at least two primary reasons for that. First, many publications do not reveal the magnitude of details necessary to reproduce results from scratch. When you try to reproduce such results, you end up with different answers or you realise that their hidden assumptions do not match yours. Second, there is an issue with raw data availability. Quality traces and logs are very difficult to obtain and nearly impossible to share. ››› It was less important that it was correct, than that it made one think about the system specification. ››› There are far too many personal opinions and there is too little acceptance of scholarship for its own sake. Referees slate papers on grounds that are either irrelevant or based on personal prejudice. ››› I believe that researchers should release the code so that field can progress much faster. ››› I'd like to see more articles that come with publicly available high-quality code that is reusable, that is portable to other environments than the exact configuration of the researcher's desktop has, and which comes with documentation. ››› It is actually disturbing that in the computer architecture community, a lot of tools are publicly available for researchers. This is good, because it promotes reproducibility and it avoids lots of people implementing the same tools. The downside is that especially students use the tools as a black box and fail to gain insight into what the tools do. ››› I think this is a great question, and that making programs that really work is so important. Yes, my code did really work and still does. One of the main motivations behind my paper was that there were algorithms out there in the literature to supposedly do the same things, but the code for them (at least as far as I could find) didn't actually work, or if it did run, it didn't work well at all. ››› As an example, recently I was involved in a piece of collaborative work that aimed to extend a published method. As a sanity check we decided to replicate the published experimental results. We could not since: a) The publication described work on an adapted version of a published program but did not list the ways in which this had been adapted. b) Some of the parameters involved were not given. ››› As a result of your query we have thought about the issue of reproducibility and accessibility that you propose and we have noticed that we have missed being able to access some other people's work that were not available. We agree it would be very desirable to extend that idea and we have promised to meet that commitment."

Quotes against

Is it too strict to ask for programs to be reproducible? Some of the people who objected to my draft *Times Higher* article objected to stringent standards I seemed to be applying; but my issue was "if there is a runnable system described in the paper it ought to be reproducible" not that "everything published should be reproducible as a program," which is impractical. But one author wrote that "Asking for source code is indeed, as someone wrote to you, equivalent to asking a natural scientist to use their lab equipment. In both cases, such requests are actually counterproductive to science, since they undermine the principle of independent verification of results. No computer code of any complexity is bug-free, so reimplementation (rather than blind reuse of buggy code) is essential to establish trust in some new algorithm." Independent verification is important, certainly, but we can only do that properly when we know what was done in the first place.

Some people have pointed out that 'reproducibility' is a term that needs closer inspection. Merely copying a program may still mean you can't reproduce the same results: for example, results may depend on networks and the exact results may be irreproducible. This is true, but you'd be even less likely to be able to reproduce the method without the program source code; worse, you might think you were reproducing the work, but actually you were doing something quite different! The program code must have existed to do the experiment reported in the first place, so why not make it available?

"It would be nice to have all the software to play around with. It would be great for the other researchers. But the effort required is considerable, too, and I'm not sure your position is the correct one. ››› Claiming that journal results are irreproducible or wrong because 95% of authors are reluctant to hand over source-code is like claiming that results published in *Nature* are suspect because 95% of *Nature* authors won't let him into their

labs to run the experiments with their equipment." Lots of people expressed similar sentiments, some more forcefully; but my question had been about papers on systems. It seems to me more like if somebody creates a special breed of mouse and claims it can do something useful, then if a paper is published about it, the mouse ought to be available for inspection, possibly at cost. But the people who replied negatively to me didn't say they didn't want to release the mouse, they said it simply was not available and in a few cases had never worked properly in the first place.

The bigger picture

It seems, then, that non-reproducibility is a cause for concern in computer science — though I am aware that concern begs questions: is there too much, is it just right, is it positive, bad, or what? Whatever! I'm guilty, too — although everything I've published worked at the time I published it, I've lost track of most of it (some stuff is on paper tape, which I still have but can't be read any more!). None of the research journals or conferences I have ever submitted my work to has ever required me to make my work reproducible.[4] We programmers tend not to keep the equivalent of lab books, and reconstructing what we have done is often unnecessarily hard. Some of that may not really be the author's fault (like my obsolete paper tape) and, on the other hand, sometimes we could certainly do better — many of the quotes above showed that a proportion of authors do not even try to make their work reproducible at the time of publishing.

What can be done?

Some people reading my notes and listening to my lecture will probably dismiss the issue. As one correspondent to me wrote, science progresses anyway, even if it is not done perfectly. Of course, in the limit, we have to be able to progress because life isn't perfect. On the other hand, my position is that we have taken a far too relaxed approach to publishing and disseminating computer science. We give the impression that computer science is easy and everything works nicely.

But when we look more closely rather a lot of computer science is imagination! I guess that's fine when the end result is another consumer product, like a DVD player — it will just be harder to use than need be, and anyway will be obsolete faster. But when the final product is a safety related device, from a heart pacemaker, an aircraft cockpit or a nuclear power station control room, or even a financial package paying my mortgage, then this relaxed attitude that permeates the community is wrong and misleading. Good computer science is hard, but it is too easy to fake.

What can we do to make things better? Here are some suggestions.

a) Open source

There is no reason why open source code cannot be made freely and immediately available, at least to the depth the ideas are discussed in the papers. And it is possible: look at sites like the GNU-licensed open source *Weka* machine learning project (www.cs.waikato.ac.nz/ml), which provides a framework people can give and take shared work. Many other sites have papers, code, demos and data too.

The *Journal of Machine Learning Research* does try to encourage authors to add electronic appendices with source code, data, demonstrations: anything, as the journal puts it, that will make life easier or more interesting for readers and researchers who follow in the authors' footsteps. Some authors do an excellent job, but spreading the good practice is an uphill struggle!

[4] The same worry applies to funding research, but examining this topic would be going into territory beyond a single lecture. A report I had from the USA recalled a comment made during a funding decision, "He got *n* papers in good journals and conferences; it doesn't matter if the system worked. It is our job to count, not to evaluate."

The journal *Transactions on Mathematical Software* (*ACM TOMS*) is one that *does* make a distinction between papers and papers that contribute programs or algorithms, and takes it very seriously. So it is possible! See http://math.nist.gov/toms/AlgPolicy.html for more details of the *ACM TOMS* policies.

b) LISP

My previous Gresham lecture ('Computer Circles,' 26 February, 2004) used a dialect of the programming language LISP I had designed specially for the purpose. One of the features I wanted in the LISP was the ability to present it running in lectures and to be able to merge writing about it and using it. Being able to do both at once, which most programming languages do not permit, meant that I could write more honestly — the lecture notes, for instance, required no work to prepare direct from the working programs.

My LISP allowed interleaving of writing, comments, program code, and program output. However, everything had to be visible. *Mathematica*, which I describe next, allows any stuff to be optionally hidden from view. Thus programs can be seen as much or as little is necessary to write a good paper, but built on the entire program code.

c) Mathematica

Many of my Gresham lectures have used *Mathematica*, for much the same reason. It allows you to write programs, explanation, slides, and so on, and mix them all up — but always keeping within the same document. *Mathematica* might be described as a powerful programming language inside a full-featured word processor (a bit like Microsoft Word), which has an outliner and the ability to show or hide any details, as is best for explanatory or other purposes. The advantages are that what you are writing about (or explaining using slides) is *exactly* what you have done.

As an example, my paper "Analysis and Simulation of User Interfaces" [see further reading] was written in *Mathematica*. I used *Mathematica* both as the 'laboratory' where I could do the experiments and programming, but also as the complete environment where I could write the paper about the work. In effect, *Mathematica* was my lab notebook. I then wrote a few lines of *Mathematica* program to automatically — and therefore reliably — transform my notebook into the paper I actually published (I will demonstrate this in the lecture). I am therefore certain that the published results were really generated by the programs I had used and talked about. Moreover, I have put the paper and lab notebook on the web, and anyone can download them and reproduce the results — as indeed some people have.

I have often wondered why systems like *Mathematica* are not more widely available to support other languages. I guess it's because the basic need to write up working programs is not a priority.

Although *Mathematica* works on any computer (a huge advantage for reproducibility) it is a special purpose programming language (and an expensive one) that not everyone will fancy. The idea I describe next, warp, I wrote as a versatile and simple tool that can be used with any programming language (more or less) to get the advantages.

d) Warp

Most programming languages are nothing like as supportive as either my LISP or *Mathematica* for writing papers about them or programs written in them. Although programming languages allow 'comments,' they are designed as if the whole point of a program is to tell a computer what to do, rather than to explain it to another human.

There have been several attempts at getting around this problem, most notably 'literate programming.' I've built a literate programming system, but — like all of them — it is a bit heavy-weight and puts off people from using it. Instead, a light-weight and easy to use approach is needed. This is where warp comes in.

I decided that describing warp, its design and motivation, would take us too far beyond this lecture, however, the lecture itself will briefly exhibit warp. Readers of these notes should refer to the further reading

for more information. Briefly, warp allows ordinary program comments to be extended into XML (or HTML) mark up, so that anything can be written in a program as part of its explanation, which is then automatically embedded into the paper about the program. The point is 'automatically' guarantees what you write about in the paper is an accurate reflection of the actual program, thanks to warp.

Warp is a special case of *literate programming* — a topic relevant but beyond the scope of this lecture. My paper on warp provides further information.

e) Editorial policy

Some journals have editorial policies that encourage reproducibility. I've already mentioned *ACM TOMS*, but others (especially biomedical journals) request that one or more authors, referred to as "guarantors," are identified as the persons who take responsibility for the integrity of the work as a whole, from inception to published article, and publish that information [see the *Uniform Requirements for Manuscripts Submitted to Biomedical Journals: Writing and Editing for Biomedical Publication* at http://www.icmje.org/]. Another approach is taken by the US National Academy of Sciences, which requires that authors must make unique materials (specifically including computer programs) promptly available on request by qualified researchers for their own use. And they provide sanctions: failure to comply precludes future publication in the journal. It is reasonable for authors to charge to cover the cost of preparing and shipping requested material.

Conclusions

There are good reasons to increase objectivity. We want ideas, and in particular programs, that work in one place to work everywhere. One form of objectivity is that published science must work elsewhere than just in the author's laboratory or even just in the author's imagination; this requirement is called *reproducibility*. Computer science depends on programming, and programs in themselves are very easily reproduced. Indeed, Don Knuth, knowing the importance of reproducibility to science, wrote that "Science is what we understand well enough to explain to a computer" because once it is explained to a computer you know what you've done, and it is in principle reproducible.

However, there is much scope to turn this 'in principle' reproducibility to effective reproducibility. Research relying on programs should take more care to explain those programs more reliably to the scientific community. Tools such as my warp are one way to do this; using program and data repositories or open source approaches are other ways.

Based on the evidence this lecture discussed, I don't think enough computer science papers are explaining programs well enough, nor are they promoting a rigorous attitude in their readers, the next generation of programmers and scientists. A change in attitude is required.

Richard Feynman's inspiring talk on Cargo Cult Science is a fitting place to end: "I'm talking about a specific, extra type of integrity that is not lying, but bending over backwards to show how you're maybe wrong, that you ought to have when acting as a scientist. And this is our responsibility as scientists, certainly to other scientists, and I think to laymen."

Computer science changed the world more than any other science in the last century. As we know from the expensive fiasco that was the millennium bug and the occasional out-of-control rocket, the wonders that are computers are precariously near to catastrophe. We've been lucky so far! While much of computing is fun, more of it needs to be done with this extra type of integrity. I hope that my series of lectures has communicated that fun, and also shed light on the underlying principles and the even greater possibilities — and fun — that being critical, rigorous and mathematical can achieve.

Further reading

S. M. Burke, *Perl & LWP*, O'Reilly & Associates Inc., 2002. [How to do web surveys.]

R. P. Feynman, "Cargo Cult Science," Caltech commencement address given in 1974. Also in R. P. Feynman and R. Leighton, *Surely You're Joking, Mr. Feynman! Adventures of a Curious Character*, Vintage, 1992.

A. Finkelstein, "Y2K: A retrospective view," *Computing & Control Engineering Journal*, **11**(4), pp156–159, 2000.

D. S. Johnson, "A Theoretician's Guide to the Experimental Analysis of Algorithms," in *Proceedings of the 5th and 6th DIMACS Implementation Challenges*, M. Goldwasser, D. S. Johnson, and C. C. McGeoch, Editors, American Mathematical Society, Providence, 2002. [I recommend this paper; it has lots of practical recommendations. It is *ACM TOMS* recommended reading for prospective authors.]

L. Mlodinow, *Some time with Feynman*, pp150–152, Penguin, 2003.

H. A. Simon, *The Sciences of the Artificial*, 2nd ed., MIT Press, 1996.

H. Thimbleby, "Analysis and Simulation of User Interfaces," *Human Computer Interaction 2000*, BCS Conference on Human-Computer Interaction, edited by S. McDonald, Y. Waern and G. Cockton, **XIV**, pp221–237, 2000.

H. Thimbleby, "Explaining Code for Publication," *Software — Practice & Experience*, **33**(10):975–908, 2003. See also www.uclic.ucl.ac.uk/harold/warp

H. Thimbleby, O. Nevalainen and T. Raita, "An Improved Insert Sort Algorithm," *Software — Practice & Experience,* **33**(10):909–1001, 2003. [An example paper I wrote to 'show off' warp in action.]

Possible further work

My survey was quick, simple and superficial. There are many ways to improve it and the methodology behind it. For example, with appropriate ethical clearance, you could write to request getting the program (perhaps one way to do this would be to *really* build a program repository for several journals, you then have an honest reason to ask for program code). Independent people could recheck my classifications.

My survey only addressed program code availability. There are wider issues in computer science, particularly the way experiments are done — I believe we routinely confuse anecdotes for theories, trial runs for experiments, and data for experimental results. Computer science is a new discipline and its approach to experimental methods is weak, at best, and the relation of experiment to theory is (I believe) far too often tenuous. Having a clear view of experimental methods to underpin doing a critical survey of what is done in the scientific community would be very interesting. Better, as we are in the business of making the world a better place, is not to survey what *is* happening, but to find ways to improve the effectiveness of the discipline in ways that are effective. Copying the policies and concerns from medicine, mathematics or the natural sciences may not be ideal.

I deliberately avoided issues of fraud in this lecture. We know there is plenty of fraud in other disciplines, and I see no reason why computer science should be immune — in fact, I can think of reasons for it to be blighted, since it is closer to market than any other science.

Finally, these lecture notes must be viewed as a draft (e.g., some more responses have come in, which I have not had time to process), and I will develop them further — in particular I will make my data available. Watch my web site, www.uclic.ucl.ac.uk/harold/warp, or email me. I hope they stimulate constructive debate.

Acknowledgements

Tim Bell, Jason Brotherton, Paul Cairns, Jon Crowcroft, Tony Hoare, Peter Ladkin, David Parnas, and the numerous respondents.

GRESHAM COLLEGE

Policy and Objectives

An independently funded educational institution, Gresham College exists

- to continue the free public lectures which have been given for 400 years, and to reinterpret the 'new learning' of Sir Thomas Gresham's day in contemporary terms;
- to engage in study, teaching and research, particularly in those disciplines represented by the Gresham Professors;
- to foster academic consideration of contemporary problems;
- to challenge those who live or work in the City of London to engage in intellectual debate on those subjects in which the City has a proper concern; and to provide a window on the City for learned societies, both national and international.

Gresham College
Barnard's Inn Hall
Holborn
London EC1N 2HH

www.gresham.ac.uk

Tel: 020 7831 0575 Fax: 020 7831 5208
e-mail: enquiries@gresham.ac.uk