

Various lectures notes I have taken

James H. Davenport
J.H.Davenport@bath.ac.uk

April–May 2009¹
(While on Sabbatical at the University of Waterloo)

¹With a subsequent appendix describing lunch with Alfred Menezes on 1 June.

Contents

1	20 April 2009	4
1.1	Sparse LU Factorization using FPGAs — Jeremy Johnson (Drexel)	4
1.1.1	FPGA	4
1.2	High-performance code generation for polynomials and power series — Ling Ding	5
1.2.1	Multiplication code generation	5
1.2.2	differential equation code generation	5
1.3	Toward High-performance Polynomial System Solvers Based on Triangular Decomposition — Xin Li	6
1.3.1	Introduction	6
1.3.2	Fast polynomial arithmetic	6
1.3.3	Main result	6
1.3.4	Bivariate system solving	7
1.3.5	Regularity Test	7
1.3.6	Implementation	7
2	SCG Meeting 24.4.2009	8
2.1	Brainstorming	8
2.1.1	Easy	8
2.1.2	Unknownn	8
2.1.3	Hard	8
3	Cambridge (Mass) 27–30 April	10
3.1	MMM brainstorming with JHD and YX	10
3.1.1	RRC: RegularRealCounting	10
3.1.2	CTD	10
3.1.3	JHD on CTD	11
3.1.4	MMM on RRC	11
3.2	Sparse signals, sparse matrices and sparse algorithms – Anna Gilbert	11
3.3	Brainstorming continued	12

4	Future Directions of Symbolic Computation Research and Their Application to the Domain Sciences	13
4.1	Grand Challenges of Symbolic Computation	13
4.2	Symbolic Computation using Disk-Based Parallel Computation — Gene Cooperman	15
4.3	Tackling Parallelism With Symbolic Computation — Markus Püschel	15
4.4	Randomization and Real Solving — J. Maurice Rojas	15
4.5	Trace matrices and clusers of zeros — Agnes Szanto	16
4.6	Applicable Computer Science Foundations	17
4.7	Dependable Scientific Computing — Gabrielle Dos Reis	18
4.7.1	Programming Languages C++0x	18
4.7.2	Compiler Technologies	19
4.7.3	References	19
4.8	The Fine Art of Plumbing: Bringing New Advances in Computer Algebra into General Purpose Software — John May	19
4.9	20
4.10	Symbolic Computation: An (almost) Indispensable Tool for R&D — Daniel Lichtblau	20
4.11	Panel: What are the Great Successes and Failures of Computer Algebra	21
4.12	Representations of Positive Powers — Victoria Powers	21
4.13	Certifying the Rank of an Integer Matrix — Arne Storjohann	22
4.14	Symbolic-Numeric methods for Polynomial Algebra — Lee	22
4.15	Dense thoughts on sparse polynomials — Giesbrecht	22
5	East Coast Computer Algebra Day 2009	24
5.1	Successful and Surprising Appliations of Compter Algebra — Stan Wagon	24
5.1.1	π	24
5.1.2	Banach–Tarski Paradox	24
5.1.3	SIAM 100-digit challenge	25
5.2	High-Precision Arithmetic and Experimental Mathematics — David Bailey	25
5.2.1	Chicken McNuggets	25
5.3	Panel	25
5.4	How to Create Repeating Hyperbolic Patters — Doug Dunstan (Duluth)	26
6	29 May 2009	27
6.1	Jacques Carette — Modern Mechanized Mathematics	27
6.2	Boneh-Boyer Signatures and the Strong Diffie-Hellman Problem — David Jao	28
6.2.1	Closing Slides	28
6.2.2	Prior Discussion	28
6.2.3	Questions etc.	28
6.2.4	Subsequent Discussion	29

6.2.5	Looking at the slides subsequently	29
A	Colopha	33
A.1	Lunch JHD and Alfred Menezes 1.6.2009	33
A.1.1	Random Oracle Model	33
A.1.2	Crouch–Davenport	33
A.1.3	The rôle of Theorems	34
A.1.4	Other papers	34

Chapter 1

20 April 2009

A day of “High Performance Computer algebra” at University of Western Ontario.

1.1 Sparse LU Factorization using FPGAs — Jeremy Johnson (Drexel)

The problem is that LU decomposition of sparse matrices is very inefficient on general-purpose processors. Motivation: the electricity market. Load Flow is ubiquitous in power system analysis. Real case problems have ≈ 26000 nodes. There is a great deal of structure, and sparsity. Typically the equations are very well conditioned.

But for each vertex of edge, need to simulate failure. The sparsity structure does not change much, so can afford to do symbolic analysis *once*. A dense solver would be $O(n^3)$, whereas with sparsity-preserving we in practice get $O(n^{1.4})$. The symbolic analysis also predicts (bounds) the size of all intermediate data structures, so the FPGA hardware can be sized appropriately.

This much sparsity handling means that book-keeping dominates the actual floating-point. Hence go for an FPGA design to do row-side right-looking Gaussian Elimination with row pivoting.

1.1.1 FPGA

Essentially a bunch of look-up tables (4-input, 1 output), but also memory, and “DSP units”, which are essentially floating-point¹ units. Use on-chip memory as a cache for columns.

In practice, the “place and route” phase of VHDL→FPGA takes over an hour. One example has 22165 logic elements, 1606Kb memory, and 50 DP blocks. Future directions: GPU?

¹Single-precision IEEE in this case. Double could probably now be done, but turns out not to be necessary.

Major future direction: implement some of this as a class in Spiral, so that some of the optimisation, i.e. *when/where* to go into hardware, can be done in Spiral.

A development board is \$20K, but production boards would be less than \$1K. JJ claims that FPGAs are keeping pace with general-purpose hardware.

Subsequently, the following references were mailed: [JVT⁺05, JCV⁺08, NLNJ07, NJN⁺08].

1.2 High-performance code generation for polynomials and power series — Ling Ding

She is a Master’s student of Éric Schost. The aim is to produce symbolic solutions to first-order nonlinear differential equations by means of Newton iteration, and from a high-performance viewpoint.

1.2.1 Multiplication code generation

This is a key component. There are various algorithms, naïve, Karatsuba and Toom, and variants, such as plain, transposed and short (truncated?). By transposed she means that we can regard polynomial multiplication $a \cdot b$ as matrix-vector multiplication $M \cdot v$ where M is a matrix representing shifted copies of a and v is b . In this setting, what about $M^T \cdot v$. We can extend $M^T = \begin{matrix} a_0 & a_1 & 0 & 0 \\ 0 & a_0 & a_1 & 0 \\ 0 & 0 & a_0 & a_1 \end{matrix}$ to be

$$\begin{matrix} a_1 & 0 & 0 & 0 \\ a_0 & a_1 & 0 & 0 \\ a_0 & a_1 & 0 & 0 \\ 0 & a_0 & a_1 & 0 \end{matrix}$$

and this gives us various useful other terms such as the middle product (details unclear).

Define [vdH02] a “divide-and-conquer” paradigm to divide a multiplication into k components, requiring l multiplications of components, giving running time $O(n^{\log_k l})$, so that Karatsuba is $k = 2, l = 3$.

While there has been previous work (eg. [Mul00]), we claim that looking at these computations as linear graphs provides a more general approach for code generation. The code generator is a Java program that generates C code for machine data types (either `double` or `unsigned_long` for modular coefficients). Claims a factor of 5 over NTL’s ZZ, and 3 over Magma.

1.2.2 differential equation code generation

Note [Wat89] recasts the problem as a fixed-point problem. [vdH02] combines this with fast “relaxed multiplication”. Problem: Solve $G(x, f, f') = 0 \pmod{x^{2^n}}$, given a solution $f \pmod{x^n}$. Need to avoid recomputation, and computation of useless (too high-precision) components.

“Obvious” Newton iteration computes useless data, both because they are already known (the first n terms are the same as last time) or because they will

be changed (terms after $2n$). In particular, this motivates the need to compute the “middle product”.

This trick is pretty well-known for computing $1/f$, but less so for $\exp f$. We save a (pretty useful) constant factor this way.

Again produce a Java code generator, which goes from a DAG (need to avoid recomputation!) for $G(x, u, t)$ to C code for G , $\frac{\partial G}{\partial t}$ and $\frac{\partial G}{\partial u}$. Again, we say which terms we want, e.g. from x^n to x^{2n} .

1.3 Toward High-performance Polynomial System Solvers Based on Triangular Decomposition —Xin Li

He is a Ph.D. student of Marc Moreno Maza.

1.3.1 Introduction

We have practical, asymptotically fast, polynomial arithmetic routines these days, so we should use these in polynomial system solving.. For systems with finitely many solutions, the time complexity of Gröbner bases and triangular sets are the same [Laz92], but space complexity of triangular sets seems better [Dah06].

1.3.2 Fast polynomial arithmetic

Standard results.

1.3.3 Main result

Assume:

- $\text{Res}(P, q, \cdot) \in \text{sat}(T)$;
- $\text{init}(P), \text{init}(Q)$ are regular modulo $\text{sat}(T)$
- the S_i are the subresultants
- for all $1 \leq i \leq \deg(Q, y)$, $S_i \in \text{sat}(T)$

Then we can compute $\text{gcd}(P, Q) \pmod{T}$ by performing a bottom-up search. Write $x_{n+1} = y$. Let $d_i = \max(\deg(P, x_i), \deg(Q, x_i))$, $b_i = 2d_i d_{n+1}$ and $B = \prod (b_i + 1)$. Time

$$O(d_{n+1} B \log B + \dots).$$

1.3.4 Bivariate system solving

1.3.5 Regularity Test

Given Q and a regular chain T , either return OK if Q is regular modulo T , or split T .

1.3.6 Implementation

At the Maple level, use Maple's DAG and recursive dense. In C have own DAG corresponding to Maple's, but encoding SLPs. Also a C library using recursive dense polynomials and asymptotically fast arithmetic. For some benchmarks, Maple/C conversion can amount to 80% of the total. The C can also be integrated into Axiom.

Chapter 2

SCG Meeting 24.4.2009

2.1 Brainstorming

The following problems can be considered for sparse polynomials.

2.1.1 Easy

Addition, multiplication

differentiation

evaluation

low degree factors [CKS99, LJ99]

2.1.2 Unknownn

Univariate irreducibility (but see bivariate below)

divisibility (as yes/no, rather than producing the quotient): the Plaisted reduction does not work.

2.1.3 Hard

divisibility, i.e. producing the quotient

G.c.d. [Pla77, Pla84]

$\gcd(f, g) \stackrel{?}{=} 1$ [Pla84, reduction to SAT]

Therefore bivariate irreducibility, by looking at $f(x) - g(x)y$, which factors iff $\gcd(f, g) > 1$.

Finding the cyclotomic part

square-free factorization

factorization

Interpolation

sparsest shift

Example 1 (Reinhold) $x^p - 1 + 2(x - 1) = (x - 1)(\Phi_p(x) + 2)$.

Question 1 *Is f cyclotomic-free?*

Question 2 *Stronger: does f have any roots on unit circle?*

Both are unknown to JHD.

Assumption 1 *f has no roots on unit circle.*

What can we do under this assumption?

Chapter 3

Cambridge (Mass) 27–30 April

3.1 MMM brainstorming with JHD and YX

3.1.1 RRC: RegularRealCounting

RRC is given a set F of polynomials, P of positive inequalities, N of non-negative inequalities, H a set of inequations, s the prescribed number of solutions, and d which specifies which variables, the first d of them, are to be considered as parameters.

RRC returns a set of regular semi-algebraic sets, and a “border polynomial”. The border polynomial is an exceptional value, e.g. “ill-posed”, or a list of polynomials over the zeroes over which we need to do more work. A regular semi-algebraic set is a variable ordering (the ordering of the parameters, which may have been changed from that input), a regular chain, a quantifier-free formula in the parameters, and a specification, e.g. “the i -th root” of which function we are considering. We know that the non-parameter variables are delineable over the set defined by the quantifier-free formula.

3.1.2 CTD

Comprehensive Triangular Decomposition(F a set of polynomials, H set of inequations, $d = \#$ parameters) returns a list RS of regular systems and a list of [constructible set $\subseteq \overline{K}^d$, list of indices]. The constructible sets form a finite *partition* of \overline{K}^d . To each such CS a list L of indices is associated such that

$$V(F) \setminus V(H) \cap \Pi_u^{-1}(CS) = \bigcup_{l \in L} Z(RS_l).$$

A regular system is a pair $[rc, h]$ where rc is a regular chain and h is a polynomial, regular with respect to rc . Then $Z([rc, h]) = W(rc) \setminus V(h)$, where

$W(rc) = V(rc) \setminus V(\prod \text{initials}(rc))$ (essentially the ‘ordinary’ or ‘regular’ zeros of rc).

3.1.3 JHD on CTD

Since the cs form a partition of \overline{K}^d , let us regard them as the driver and imagine a construction of the form.

3.1.4 MMM on RRC

Assuming that the border polynomial is not an exceptional value, we can regard an RRC as a set of options with each $bp_i = 0$ corresponding to a semi-algebraic set, and each quantifier-free formula corresponding to the k -th root of a regular chain. Hence if `RootOf` could be extended to

- take a zero-dimensional regular chain, rather than just a polynomial;
- use the index counter to represent i -th *real* root, rather than complex roots

this structure could be represented. The `RootOf` would also need to be able to store the variable ordering appropriate to the regular chain.

In general it would be possible to trigger this sort of output in an upwards-compatible way with the additional parameter `output-piecewise`.

3.2 Sparse signals, sparse matrices and sparse algorithms – Anna Gilbert

This as an M.I.T. seminar, immediately after JHD give his seminar.

“A lot of you know about streaming algorithms, so I’ll talk about the flip side.”. For example, take a discrete wavelet transform, set all the ‘small’ coefficients to zero (permitting a sparse representation), and transform back, to get an “approximation” of the original. So we want to compress images, signals and data, in ways which are: accurate, concise and efficient.

However, “cutting edge” encoders *tend* (heckling from the audience) not to use a single orthonormal basis.

Definition 1 A dictionary D is a subset $\{\phi_i\}_{i=1}^d$ of unit-norm vectors. The elements are called atoms. If $\text{span}(\phi_i) = \mathbf{R}^d$, we say the dictionary is complete.

Example: Fourier–Dirac: union of complex exponentials and δ -functions. Various problems:

Sparse Given x , find sparsest c with $\Phi c = x$.

Error Given x, ϵ , find sparsest c with $\|\Phi c - x\| < \epsilon$.

Compression Given x, k find a c of weight $\leq k$ minimising $\|c - \Phi^{-1}x\|$

All three are NP-complete.

Claim sparse approximation crops up everywhere? A measurement matrix satisfies the Restricted Isometry Property if Many approximation algorithms, such as Iterative Thresholding, Sparse Matching Pursuit, will return “close” \hat{x} to x given $y = Ax$ if A satisfies a RIP property.

3.3 Brainstorming continued

We looked at John May’s e-mail. MM recalled that `Triangularize(F,H,R)` where $R = \mathbf{K}[x_1, \dots, x_n]$ returns a constructible set, which is a list of regular systems (interpreted as a union).

Then, for each such `RS:= [T:regularchain, h:Poly]`, we call

`LRSAG, BP := RealRootClassification(T, N, P, H ∪ {h} ∪ initials(T), d, 1 . . . k, R)`.

Then both the `RealSemiAlgebraicSets` and (made inert) the border polynomials need to be added to the solution.

MMM mentioned that there is a question of structuring the code so as to avoid repetition, since some of the classification work has been done already. However, since we are passing round regular chains, some of this is avoided since the `RegularChains` library “knows about” them.

There is also an issue: some of these questions came up when MMM was talking with “the Chinese people” (Bicom Xia¹, Rong Xiao², — both Peking University — and YX³, — M.I.T.), and they should probably be co-authors.

¹`xbc@math.pku.edu.cn`.

²`akelux@gmail.com`.

³`yxie@csail.mit.edu`.

Chapter 4

Future Directions of Symbolic Computation Research and Their Application to the Domain Sciences

An NSF sponsored workshop at the University of Rhode Island: 30 April–1 May. The meeting was co-organised by Erich Kaltofen and Lenore Mullin (NSF). Joan Peckham (URI) is on secondment to the NSF, and introduced CPATH as a programme to revitalise undergraduate computing education (not necessarily just for CS majors). The NSF notices that CS enrolments are down, and that graduates in other disciplines are not equipped to use computational skills effectively. The buzz-phrase is “Computational Thinking”.

Lenore Mullin, the NSF Director, posed the question, which will come up in the panels.

Can academically-developed and Open Source software such as La-Pack compete with commercially-developed software?

4.1 Grand Challenges of Symbolic Computation

Panellists: Mark van Hoeij (FSU), Hoon Hong (NCSU), Ilse Ipsen (NCSU), David Wood (Delaware).

1. **Is symbolic computation too broad?** (EK)

DW “As per Lady Lovelace”.

II When did one last hear any other area ask if it was “too board”?

MvH Agreed, especially the various sub-topics proposed all inter-relate.

HH Broad — yes; too broad — no.

Tim Daly We are seeing the growth of a new science of “computational mathematics”, and this apparent breadth is merely a question of growth pains.

2. **How can the latest algorithm advanced be made readily available?**

II We should look at what the numericians do, with BLAS etc.

DW Said by a numerician “You guys really need an export division”.

Dos Res This comes up in programming, and the Haskell community has a search tool which will search on parameter types. This could be done in a language like Axiom.

JHD Let’s not be too self-critical here: matrices have been around for 150 years, BLAS for 10–15, and sparse BLAS are still being argued about. The question is *indeed* a good one, but it’s not the case that others have solved the corresponding ones.

Singer Part of the problem is education: people don’t know the capabilities of computer algebra.

Watt But often the user does not *know* what they want.

3. Can OpenSource and academically developed systems such as LAPack and LinBox, and systems such as SAGE, compete with commencial mathematical software.

II Very surprised by this question, since, e.g. MatLab is based on LAPack. MatLab sells on its usability, not the underlying algorithms.

MvH “Compete in terms of user count?” — in that case it’s all about graphical interfaces.

May We should distinguish between libraries such as LAPack, which Maple is not equipped to develop in house, and systems such as Maple, much of the development work of which is in “boring” interface-related work.

Watt The ‘open source’ developer does what *he* thinks is right, the commercial developer what the customer thinks is right. The two are not compatible.

JHD I agree with May, apart from the labels: it’s really “core algorithmic functionality” versus “usability package”. John May nodded vigorously in agreement.

4.2 Symbolic Computation using Disk-Based Parallel Computation — Gene Cooperman

He first made a remark on checkpoint–restart, on which he has an ArXiv paper (reference illegible). `dmtcp.sourceforge.net`. They are aiming to get this validated as part of Condor.

The end of the world is nigh: Moore’s Law is changing, and memory chips are no longer twice as large every 18 months. Large shared-memory computers are expensive. We want to use *all* the discs of our cluster, in parallel, hence we *do* get the bandwidth necessary. But what about latency? We have five years experience of doing this on a case-by-case basis, across applications such as Gröbner bases, rewriting, verification.

Essentially he is emulating a 200-node 25TB shared memory computer on a 50-machine cluster with discs on each. Claim (experimental): every program generating TB of data is essentially doing it in parallel.

For Baby Monster, the disc is 2 days, CPU/RAM 10 days, and the network 2 days. In fact it is much more the RAM speed than the CPU which dominates the 10 days, and hence the total. This is a *good thing*, as individual CPUs are not getting faster. He showed code, using his library code with data types such as `roomy_list`. One current application is in coset enumeration.

4.3 Tackling Parallelism With Symbolic Computation — Markus Püschel

Example — DFT. The code from Numerical Recipes, compiled with the best compiler shows about 1Gflop, independent of problem size. The best on the net is anything from 12–30 times faster, but this *does* depend on problem size. But 2004 was the end of machines getting faster in terms of clock speed, though total power available is still following Moore’s law, because of SIMD, multi-core etc. Of his 12–30 factor, a factor of 5 comes from better C code (making better use of the memory hierarchy), 3 from SIMD instructions (which compilers cannot use in the case of DFT) and 2 from genuine multi-core operation. But this rewriting is expensive in terms of labour.

Spiral should *automatically* generate good code. The algorithms are specified in a high-level language, on which the transformers work.

4.4 Randomization and Real Solving — J. Maurice Rojas

Application 1: Multi-Impulse orbit Transfer Problem (for Satellites), Mortari & Avendaño, 2008. Application 2: Flow Feature Extraction, Peby & Thompson, 2004. Application 3: Probabilistic Modeling of Shock Response Spectrum data.

Fitting the right probabilistic model relies on algebraic statistics, which tends to want the roots on $[0, 1]$.

What's the right tool: suppose I wanted the *exact* number of real roots of a sparse high-degree polynomial, e.g. $1 - 2x^{196418} + x^{307841}$. By f_{26} , the Sturm sequence has 674000 digit numbers. Can we take account of the sparsity in this sequence? Claim: Yes. If we look at $f(x) = 1 - cx^d + x^D$, then for small c we have no real root, and for large c we have two. This can produce what he calls the Archimedean Newton Polynomial. This ends up in an equation in c , d and D , but is very expensive to evaluate. Take logs and use Baker's theorem.

Theorem 1 *Suppose $A \subset \mathbf{Z}^n$ has cardinality m , maximum bit-size τ for any coordinate of A , and f has support A . Then we can decide whether f lies in a unique chamber cone (and correctly detect if not) in time polynomial in m , n , τ .*

By a "chamber cone", we mean a region in which the number of roots (and the homotopy, JHD assumes) is constant.

Corollary 1 *For fixed n , we can find canonical lower binomial system for any $n \times n \dots$*

4.5 Trace matrices and clusters of zeros — Agnes Szanto

Problem 1 *Given s polynomials in n variables generating I , with infinitely many roots in \mathbf{C}^n . Suppose I has roots with multiplicities, or roots in clusters, then we want to compute an "approximate radical", a polynomial system with one root in each cluster, corresponding to the arithmetic mean of the cluster.*

Univariate example: $f(x) = (x - 1)^8(x + 2)^5(x - 3)^3(x + 4) + 10^{-5}R(x)$. Perturbed clusters, and the higher the multiplicity the wider the cluster is. The corresponding 17×17 eigensystem is ill-conditioned. Instead, we compute what is essentially the companion matrix of the *square-free part* of the original problem.

In general, the companion matrix is the univariate equivalent of the more general multiplication matrix. In the case of exact roots with multiplicities we have the following facts.

1. The rank of the trace matrix is equal to the number of *distinct* roots.
2. Any full-rank submatrix will give the multiplication matrix of the radical.

In general, if ϵ is the perturbation away from genuine clusters, and we try to do Gaussian elimination in the multiplication matrix, then, where we would have had 0 in the multiplicity case, we get ϵ^2 entries. Hence it *should* be easy to spot the clusters.

There is a generalisation to the over-determined case. However, the bottleneck is the computation of the Sylvester matrix.

The generic degree bound is much worse than Lazard’s worst-case degree bound — can we find and exploit this? Ongoing work.

A second ongoing task is to make the whole process fraction-free.

4.6 Applicable Computer Science Foundations

Panellists: Lenore Mullin (moderator) Bruce Char, Ed LaMacchia, Jeremy Johnson. All questions were put up first, so there is not a match between questions and answers.

1. How can programming languages provide computational reproducibility?
2. Can “Production languages” have symbolic prowess? symbolic/numeric?
3. How can we easily translate from one language to another?
4. Is there semantic equivalence of an algebra
5. Can we abstract machines for some domain, whihc one?

BC In practice reproducibility is very difficult, and the problem isn’t the language as the “hacker culture” [JHD’s words]

LM How do we know that the “translation into C from MatLab” is faithful.

Tim Daly We should look at OpenMath/MathML 3 work on semantics, as per JHD.

JHD Spoke to OpenMath/MathML convergence, pointing out that Rob Miner is the U.S. lead in MathML content.

LM How do we bring this into the U.S.?

Kaltofen This makes no sense: MathML is international.

LM When NSF hears “it’s being done in Europe”, NSF understands “we don’t need to do it”.

Watt The U.S. editors of MathML do *not* work at universities, but this is true more generally: himself and Kohlhasse are the only academic editors of the MathML standard.

LM Can there be an abstract machine which could be efficient and do these computations (JHD had some difficulties with understanding this question).

JJ We need ways of performing translations from high-level descriptions that are semantically correct.

Watt In designing Maple, we had to make many compromises.

Cooperman Working with CERN on 1,000,000+ lines of C++. There is a major issue of reproducibility here, as they move to multi-core architectures, how can they have reproducibility of codes involving random numbers.

LM Should there be a list of standard problems, or algorithms, or whatever.

Kaltofen Berkeley has a list of the 13 dwarves of computation¹, which does not mention computer algebra. Yellick agrees that this was a mistake. Kaltofen has his own list of the seven dwarves of symbolic computation.

LM How do we bring more visibility to this area, which NSF is probably going to be calling symbolic/numeric.

EL Computer science has changed dramatically over the last ten years. Look at *Comm. ACM*: most of the articles are now about applications.

LM Is that because we have decided to ignore fundamental problems?

HH It's what NSF funds.

LM But the community drives NSF directions (some hollow laughter).

BC Often a problem is intractable in general, but one can see a version, in a specific application area, which is tractable.

LM Technology is getting smaller, ...

4.7 Dependable Scientific Computing — Gabrielle Dos Reis

4.7.1 Programming Languages C++0x

C++ is widely used, available on virtually any system, efficient and expressive, (now) has an ISO definition². C++98, the first ISO standard, had the Standard Template Library, which got a (certain amount of) generic programming into the main stream.

```
namespace algebra {  
  //  
  concept Monoid<Regular T> {  
    T operator+(T,T);  
    T neutral_value();  
    axiom identity(T x):  
      x + neutral_value() <=> x;  
      neutral_value() + x <=> x;  
  }  
  ...  
}
```

¹Apparently one can just Google this.

²He represents France on the ISO committee, and therefore has one vote, just like the U.S.!

But so far the axioms are essentially only comments³.

To make `string` into a monoid, we would need:

```
concept_map algebra:: Monoid<string> {  
    // As it happens + works on strings  
    // need code to define the entry string as neutral value.  
    // need code to define the empty string as neutral value.
```

In subsequent conversion, Dos Reis explained that he has managed to get `<=>` into the C++ standard, as an operator meaning abstract equality, and inspired by JHD's [Dav02]. JHD congratulated him.

4.7.2 Compiler Technologies

Ran out of time, but did mention that one of the problems of STL with C++98 technology was that one error could produce pages of error messages.

4.7.3 References

Many, including Self plus Stroustrup in POPL 2006; Self + many in OOPSLA 2006.

4.8 The Fine Art of Plumbing: Bringing New Advances in Computer Algebra into General Purpose Software — John May

These software tools are (supposed to be) user driven. We can solve a quadratic in x , but suppose the user has the same equation in $\cos x$. Is it $\arccos(-3/2) \pm \frac{1}{2}\sqrt{517}$ or $\arccos(-3/2) \pm \frac{1}{2}\sqrt{517} + 2k\pi$, $2k\pi - \arccos(-3/2) \pm \frac{1}{2}\sqrt{517}$? And what happens if they write 2.31 rather than 2?

Watt Why is 2.31 a floating point number? I thought it was a number?

May Good question. The language takes the decimal point as an indicator that approximate numbers are implied. In practice the worst is 0.5, which “equals” $\frac{1}{2}$.

The worst case is when experimentally-determined floats appear in exponents.

Rojas But one can extend the theory of discriminantal varieties to real exponents.

May But it *is* a non-trivial extension.

³JHD has a conversation later in the conference with him. These axioms were inspired by his ++ comments in Axiom, but Dos Reis did not know that JHD *et al.* had originally intended these to be executable specifications, but of course the state of the art in theorem-proving in 1978 was rather different than today.

He showed the call graph of `solve`, nearly all of which was “plumbing” to turn the user’s problem into things that computer algebra (ISSAC etc.) describe can solve with algorithms. Both are 10+KLOC, but the algorithms can be proved correct, the plumbing can only be verified against regression tests etc.

Most papers about polynomial system solving assume an ordering is given, but most users don’t know this, have no idea of its importance. Quoted [BD07] that orderings *may* make a massive difference⁴, or may not. In Maple’s regression tests, the commonest number of variables is 2, and also for equations, and for total degree. It also matters that small systems are solved fast, and sophisticated analysis can eat up the savings from a sophisticated algorithm.

4.9

Areas where computers have been used:

- four-colour theorem
- Kepler’s conjecture
- Birch–Swinnerton-Dyer.

But more and more papers are using proprietary systems to verify detail: he sees several MAGMA cases. Neubüser describes this as a violation of the tenets of mathematics: that everything should be published and available for checking.

- Commercial software is analogous to a mathematics journal that only published statements, not proofs.
- Implementation of mathematical algorithms whose development was funded by the taxpayers is sometimes absorbed into commercial programs.
- If mathematics were done this way, NSF would probably refuse to fund it.

* Fractious debate with LM ensued.

SAGE is actively supported, there are 850+ people on the `sage-dev` list, about 150 of whom are “significant” developers. Everything is open source. uses `pseudotty` as one way of invoking other systems, also the `pexpect` system for passing strings to/from other systems. Many people prefer GUIs, and the “SAGE Notebook” (based on Ajax) provides this.

4.10 Symbolic Computation: An (almost) Indispensable Tool for R&D — Daniel Lichtblau

The speaker admitted to have been at Wolfram Research for 17 years, and this defined/coloured his perspectives.

⁴Polynomial versus doubly exponential.

He asked how many people, other than Erich Kaltofen, would have thought in 1982 that lattice reduction was symbolic computation, and speculated that the answer was none. There was a debate about the presence of numerics in symbolic computation, but that is largely history.

The GUI, and the graphics, are an important part of the “packaging” of symbolic computation. An important trend is writing more and more in the system itself, and this is happening to Mathematica as well. Availability of quality libraries, such as GMP as well as LAPack are important. The “plumbing” that John May mentioend is very important, even though it tends to be less exciting than the pure algorithmic development.

4.11 Panel: What are the Great Successes and Failures of Computer Algebra

Panel: Stephen Watt (Moderator), Erich Kaltofen, David Saunders, James Danenport

4.12 Representations of Positive Powers — Victoria Powers

Always in $\mathbf{R}[x_1, \dots, x_n]$.

PSD Positive Semi-Definite

SOS Sum of Squares.

PSD implies SOS in a few special cases ($n = 1$, degree 2 and one other) and not elsewhere (Hilbert 1888).

$$g := y^6 + x^6 - x^2y^4 - y^4x^4 + 3x^2y^2 + \dots$$

is PSD not not SOS, but has a certificate

Given a basic closed semi-algebraic set $S = \{g_1(X) \geq 0, \dots, g_k(X) \geq 0\}$ and $f \in \mathbf{R}[X]$, we want a representation in terms of g_i and sums of squares from which it is “obvious” that $f \geq 0$ on S .

Theorem 2 (Hilbert’s 17th problem: Artin) *If f is GSD, then $\exists g$ such that $g^2 f$ is SOS.*

Theorem 3 (Pólya) *If p is homogeneous and $p > 0$ on the n -simplex Δ_n , then $\exists N \in \mathbf{N}$ such that $(x_1 + \dots + x_n)^N$ has positive coefficients.*

Recently Powers and Resnick can bound the N here, and can deal with cases when $p = 0$ at places on the boundary of the simplex.

Problem 2 *If $f \in \mathbf{Q}[X]$ is SOS in $\mathbf{R}[X]$, is it SOS in $\mathbf{Q}[X]$?*

Conjecture 1 (BMV) *The polynomial $p = \text{tr}(A + tB)^m \in \mathbf{R}[t]$ has only non-negative coefficients for all m whenever A and B are square psd matrices over \mathbf{R} .*

Proved true for $m \leq 13$ (Klep and Schweighofer).

4.13 Certifying the Rank of an Integer Matrix — Arne Storjohann

Expression swell with matrices over \mathbf{Z} or worse.

Smith form: deterministic (thesis) $\tilde{O}(n^4d)$, uncertified $\tilde{O}(n^{3.2}d)$ Monte3 Carlo — Kalfoten *et al.*, 2004.

Frobenius form: $\tilde{O}(n^5d)$ Giesbrecht 1995, $\tilde{O}(n^4d)$ Storjohann & Giesbrecht (2004) and uncertified $\tilde{O}(n^{3.2}d)$.

Lattice computation $O(n^6d^3)$, Kalfoten $O(n^6d^2 + n^5d^3)$ and Nguyen & Stehlé (2005/2009) $O(n^6d + n^5d^2)$, even with classical arithmetic.

4.14 Symbolic-Numeric methods for Polynomial Algebra — Lee

Largely about sparse interpolation, but JHD was distracted.

4.15 Dense thoughts on sparse polynomials — Giesbrecht

Essentially computer algebra understands (systems of) dense polynomials. Human beings write polynomials sparsely, and systems will comply with the writing, the adding and multiplying, and not much else. Producing

$$(x^{6363}y^{16929} + 2)^2(x^{6969}y^{13365} - 5)^2$$

from the expanded form is *very* hard. The factorization of $x^{1000000} - 1$ is very long, and tells you nothing.

Kalfoten & Nehring are working on sparse interpolation. Note that there are various questions about which basis to choose.

[Pla84] has NP-hardness results, e.g. about gcd. Also squarefreeness [BS99]. However, we can do integer root finding [CKS99], small degree factors [LJ99], perfect powers [GR09], polynomial decomposition. Many open problems, e.g. sparse division.

De Prony (*J. Ecole Poly.* 1795) had a strong precursor to [BOT88]. Work by Garg & Schost on Chinese Remainder in exponent spaces. Giesbrecht *et al.* working on evaluations at powers of a random root of unity, which involves showing that Vandermonde are well-conditioned with high probability (depending on the root of unity chosen).

Sparse shifts: if f is sparse in $x - r$ for *unknown* r , can we find this. Can be done in polynomial time *if* I view f as a black box allowing modular evaluations. [LS94] show that the sparsest shift is unique and rational.

Roche (MICA 2008) has some results on fast multiplication of “chunked” sparse polynomials.

Chapter 5

East Coast Computer Algebra Day 2009

5.1 Successful and Surprising Applications of Computer Algebra — Stan Wagon

The speaker announced that Mathematica had changed his life. He runs a Mathematica course in Colorado every July.

5.1.1 π

The remarkable formula

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

is due to Bailey *et al.*. But

$$\pi = \sum_{k=0}^{\infty} \frac{(-1)^k}{4^k} \left(\frac{2}{4k+1} + \frac{2}{4k+2} + \frac{2}{4k+3} \right)$$

is an easier formula, due, subsequently, to the author and Adamchik, and discovered by the method of undetermined coefficients.

5.1.2 Banach–Tarski Paradox

But note Hausdorff was really the originator. The real starting point is a group $\langle s, t \rangle$, which can be divided into A , B and C , with $tA = B$, $t^2A = C$, but $sA = B \cup C$, so A is both $\frac{1}{2}$ and $\frac{1}{3}$ of the whole.

A different tiling is $\langle s, t, u \mid \dots \rangle$ which produces an Escher-like tessellation.

5.1.3 SIAM 100-digit challenge

Problem 3 (4) Find the extrema (probably minimum) of

$$\sin(60e^y) + \sin(70 \sin x) + \sin(\sin(80y)) - \sin(10(x + y)) + e^{\sin(50x)} + \frac{1}{4}(x^2 + y^2).$$

The URI team was the only one to use interval analysis (now re-named “reliable computing” — smart move! — who wants “unreliable computing”), and solved the problem. It was important here that ∞ was a number in Mathematica, since he began by dividing x, y space into a finite number of boxes.

In the process of thinking about interval arithmetic, he re-discovered the dependence problem (JHD notes that it is described in [DF90], but this is rarely cited).

5.2 High-Precision Arithmetic and Experimental Mathematics — David Bailey

Concerned with highly-accurate arithmetic, particularly with highly non-linear or ill-conditioned problems. Emphasis on the PSLQ integer relation algorithm: given $\mathbf{x} \in \mathbf{R}^n$, find a vector $\mathbf{n} \in \mathbf{Z}^n$ such that $\mathbf{n} \cdot \mathbf{x} = 0$, or $\mathbf{n} \cdot \mathbf{x} \approx 0$. The algorithm is by Ferguson, who is apparently now better known as a sculptor than a mathematician. PSLQ requires very high precision input, but detects a “true” relation by a sharp increase, say by 10^{20} or more, in the accuracy of the approximation. The example demonstrated had a drop of 10^{200} , essentially to ∞ since he had 250-digit accuracy.

While this has been described as “a solution in terms of a problem”, it has found new equations for numbers in Chaos Theory. What else (see π above) remains to be discovered.

Euler-Maclaurin

5.2.1 Chicken McNuggets

Used to come in boxes of 6, 9 and 20. This means that the largest number one *cannot* buy is 43: the Frobenius Number of $\{6, 9, 20\}$. This is basically a question of Integer Linear Programming, and area closely related to computer algebra via lattices.

5.3 Panel

Panelists: Jeremy Johnson (Moderator), JHD, Hoon Hong, daniel Lichtblau (Wolfram Research), John May (Maplesoft), Victoria Power.

My notes are clearly incomplete. Johnson pointed out that JHD was “one of the pioneers of the field”. JHD was to start, on the topic “What are the grand challenges of symbolic computation”. JHD stated that he had some

difficulties with this, since he didn't think that computer algebra had distinct "grand challenges" in the same sense as experimental sciences. Rather, our "grand challenge" is to create tools that others can use to solve their grand challenges.

One good point that came up, from Stan Wagon, was that, if he types an ODE involving 5 into Mathematica, he gets one result, but using 5.3 gets a completely different result. In later discussion with him, he admitted that 53/10 would have given a result more like 5. JHD therefore identified three different dichotomies that computer algebra systems tend to confuse.

1. That imposed by hardware: integers or floating point.
2. That imposed by notation: is there a . in the input?
3. That imposed by intention: is this *precisely* the number $\frac{53}{10}$, or rather "some number near" 5.3.

He also point the different between Mathematica's software floats, which are essentially "poor man's intervals", with some indication of precision, and hardware floats, which are just that. JHD pointed out that Maple's software floats are not like Mathematica's.

5.4 How to Create Repeating Hyperbolic Patterns — Doug Dunstan (Duluth)

Gave some good precursors of Escher's diagrams, from Klein and Coxeter.

Escher's replication process for "Circle Limit I-IV" was based on hyperbolic geometry. There were some good illustrations of this, based on his computer-based reconstructions of Escher's prints.

Then JHD has to leave for Dubai.

Chapter 6

29 May 2009

Two lectures at the University of Waterloo, the first in the context of the Waterloo/Western Ontario Joint Laboratory Meeting.

6.1 Jacques Carette — Modern Mechanized Mathematics

If you were building a new system now, and had the premise that correctness and performance were equally important, what would you do?

Statistics 100–200 million

MatLab 30–40 million

CA 5–6 million

Theorem Provers 400,00

Theorem Provers 30,000

Shows various user interfaces: points out that theorem provers have pretty non-visual user interfaces. Gives a “family tree” of major CA systems, and, and of major theorem provers. States that de Bruijn, with Automath, was far ahead of his time, especially about dependent types.

His new system is Chiron. It is based on biform theories, and the programming model is λ -calculus with dependent types.

Points out that LU decomposition is really a family of algorithms, with at least 25 different decisions. There are actually > 80 in Maple. Hence he has an algorithm generator for this, which will generate code that is *identical* to the human-written code.

Has “little theories” as a paradigm, each theory containing *one* idea, and so it takes him 20 theories to get to commutative monoid, but each is a single step.

6.2 Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem — David Jao

A seminar in the Combinatorics and Optimization Department (where cryptography lives here) series. JHD was very late, due to a crown falling out. These notes are a combination of his last slides, my conversations with him, his e-print [JY09] and his slides which he subsequently e-mailed to me.

6.2.1 Closing Slides

- q -SDH is *equivalent* to Boneh-Boyen Signatures.
- Unless p is specifically chosen¹, Boneh-Boyen *keys* can be recovered on $O(p^{2/5+\epsilon})$ time, with $O(p^{1/5+\epsilon})$ signature queries.
- Bottom line: “avoid protocols based on q -SDH”.

6.2.2 Prior Discussion

It transpired in previous discussions that $\frac{2}{5}$, a number that does not often occur in complexity exponents, is essentially because $\frac{2}{5}$ is the harmonic mean of $\frac{1}{2}$ and $\frac{1}{3}$, both well-known numbers. The $\frac{1}{5}$ is (logically) derived from the $\frac{2}{5}$, not *vice versa*

6.2.3 Questions etc.

- What is we have more than $q \approx p^{1/5}$ signature queries — doesn't that speed things up? No — the partial fraction process ([JY09, Proposition 4.1]) takes time $O(q^2)$, so this phase costs you more.
- What would you use for short signatures — answer BLS².

JHD “specifically chosen”? Neither $p-1$ nor $p+1$ must have *any* (not necessarily prime) divisor “near” $p^{1/5}$.

Menezes And furthermore, the curve must be “pairing-friendly”. BN curves [BN06] are themselves rare.

DJ True, but examples do exist, for example [JY09, p. 11] there is a BN curve with 1461501641662054988059088728056207736278975404329 points. JHD subsequently wrote to DJ as follows.

I'm impressed by the number on page 11 of the preprint: $\log(\text{factor}(p-1))/\log(p)$ jumps from 0.06 to 0.24, neatly avoiding 0.2.

For $p+1$, though, it doesn't seem so good: the jump around 0.2 is [0.1983844679,0.2012310607]. Does this mean that the window around $p^{1/5}$ is really very tight?

¹See later.

²Presumably Boneh, Lynn, Shacham.

He replies as follows.

However, one thing that I didn't mention in the talk (but is written in the paper) is that the $p + 1$ algorithm requires comparable (exponential) amounts of time and space, whereas the $p - 1$ algorithm gets by with constant space. So in practice, if I am choosing a prime p , I am a lot more worried about divisors of $p - 1$ than $p + 1$.

6.2.4 Subsequent Discussion

Apparently Boneh-Boyen Signatures have been used in practice because of their shortness, and the application was so “close to the edge” that, unusually, the change in exponent from $\frac{1}{2}$ to $\frac{2}{5}$ is significant.

6.2.5 Looking at the slides subsequently

BLS signatures [BLS04] are equivalent to the computational Diffie-Hellman problem *under the random oracle assumption*. However, this latter assumption is false [CGH04]: as Jao's slide says:

There exist cryptographic protocols which are provably secure using random oracles and provably insecure using any implementation of a function in place of the random oracle.³

Boneh-Boyen Signatures [BB04] are secure in the “standard model”, in the sense that they are secure if the “ q -Strong Diffie-Hellman (SDH) Problem” is hard., Note that this is currently only a one-way relationship, and DJ proves equivalence.

³JHD, DJ and AM had an interesting discussion about the practical significance of this. Subsequently, AM and JHD had lunch — see Appendix.

Bibliography

- [BB04] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Proceedings EUROCRYPT 2004*, pages 223–238, 2004.
- [BD07] C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In C.W.Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.
- [BLS04] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *J. Cryptology*, 17:297–319, 2004.
- [BN06] P.S.L.M Barreto and Naehrig.M. Pairing-friendly elliptic curves of prime order. *Selected areas in cryptography*, pages 319–331, 2006.
- [BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings 20th. Symp. Theory of Computing*, pages 301–309, 1988.
- [BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Proceedings EUROCRYPT '94*, pages 92–111, 1994.
- [BS99] A. Bernasconi and I. Shparlinski. Circuit Complexity of Testing Square-Free Numbers. In *Proceedings STACS 99*, pages 47–56, 1999.
- [CD01] P.A. Crouch and J.H. Davenport. Lattice Attacks on RSA-Encrypted IP and TCP. In B. Honary, editor, *Proceedings 8th. IMA Conf. Cryptography and Coding*, pages 329–338, 2001.
- [CGH04] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51:557–594, 2004.
- [CKS99] F. Cucker, O. Koïran, and S. Smale. A Polynomial Time Algorithm for Diophantine Equations in One Variable. *J. Symbolic Comp.*, 27:21–29, 1999.
- [Dah06] X. Dahan. Sur la représentation des systèmes polynomiaux : triangulation, méthodes modulaires, évaluation dynamique. *Doctorat Thesis*, 2006.

- [Dav02] J.H. Davenport. Equality in computer algebra and beyond. *J. Symbolic Comp.*, 34:259–270, 2002.
- [DF90] J.H. Davenport and H.-C. Fischer. Manipulation of Expressions. *Improving Floating-Point Programming*, pages 149–167, 1990.
- [GR09] M. Giesbrecht and D.S. Roche. Detecting lacunary perfect powers and computing their roots. <http://arxiv.org/abs/0901.1848>, 2009.
- [JCV+08] J.R. Johnson, T. Chagnon, P. Vachranukunkiet, P. Nagvajara, and C. Nwankpa. Sparse LU Decomposition Using FPGA. In *Proceedings PARA 2008: 9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, 2008.
- [JVT+05] J.R. Johnson, P. Vachranukunkiet, S. Tiwari, P. Nagvajara, and C. Nwankpa. Performance analysis of load flow computation using FPGA. <http://www.montefiore.ulg.ac.be/services/stochastic/pscc05/papers/fp560.pdf>, 2005.
- [JY09] J. Jao and K. Yoshida. Boneh-Boyen signatures and the Strong Diffie-Hellman problem. <http://eprint.iacr.org/2009/221.pdf>, 2009.
- [KM07a] N. Kobitz and A. Menezes. Another look at generic groups. *Adv. Math. Commun.*, 1:13–28, 2007.
- [KM07b] N. Kobitz and A. Menezes. Another Look at "Provable Security". *J. Cryptology*, 20:3–37, 2007.
- [KM08] N. Kobitz and A. Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *J. Math. Cryptology*, 2:311–326, 2008.
- [Kob07] N. Kobitz. Another look at automated theorem-proving. *J. Math. Cryptology*, 1:385–403, 2007.
- [Laz92] D. Lazard. Solving Zero-dimensional Algebraic Systems. *J. Symbolic Comp.*, 13:117–131, 1992.
- [LJ99] H.W. Lenstra and Jr. Finding small degree factors of lacunary polynomials. *Number theory in progress*, pages 267–276, 1999.
- [LS94] Y.N. Lakshman and B.D. Saunders. On computing sparse shifts for univariate polynomials. In *Proceedings ISSAC 1994*, pages 108–113, 1994.
- [McD81] D. McDermott. Artificial Intelligence Meets Natural Stupidity. *Mind Design*, pages 143–160, 1981.
- [Mul00] T. Mulders. On Short Multiplications and Divisions. *AAECC*, 11:69–88, 2000.

- [NJN⁺08] C. Nwankpa, J. Johnson, P. Nagvajara, Z. Lin, M. Murach, and P. Vachranukunkiet. Special purpose hardware for power system computation. In *Proceedings Power and Energy Society General Meeting 2008*, pages 1–7, 2008.
- [NLNJ07] P. Nagvajara, Z. Lin, C. Nwankpa, and J. Johnson. State Estimation Using Sparse Givens Rotation Field Programmable Gate Array. In *Proceedings Power Symposium NAPS '07*, pages 421–427, 2007.
- [Pla77] D.A. Plaisted. Sparse Complex Polynomials and Irreducibility. *J. Comp. Syst. Sci.*, 14:210–221, 1977.
- [Pla84] D.A. Plaisted. New NP-Hard and NP-Complete Polynomial and Integer Divisibility Problems. *Theor. Comp. Sci.*, 31:125–138, 1984.
- [Sho02] V. Shoup. OAEP Reconsidered. *J. Cryptology*, 15:223–249, 2002.
- [vdH02] J. van der Hoeven. Relax, but Don't be Too Lazy. *J. Symbolic Comp.*, 34:479–542, 2002.
- [Wat89] S.M. Watt. A Fixed Point Method For Power Series Computation. In P. Gianni, editor, *Proceedings ISSAC 1988*, pages 206–217, 1989.

Appendix A

Colopha

A.1 Lunch JHD and Alfred Menezes 1.6.2009

Various topics were discussed.

A.1.1 Random Oracle Model

AM continued the discussion from Friday about the rôle of the random oracle model and [CGH04]. As far as he is concerned, that paper says little more than “you have to be cautious when you instantiate with a real hash function”. For example, if you instantiate the random oracle with SHA-1, you may have problems with message m if $\text{SHA-1}(m) = 0$.¹

Nevertheless, the issue continues to divide the community, with several people rejecting, or marking lowly, papers that are proved in the random oracle model. JHD pointed out that “instantiate the random oracle with SHA-1” is, in engineering-speak, putting a system together out of components.

A.1.2 Crouch–Davenport

In this context, JHD described [CD01], where a weak system is built from TCP/IP and RSA. In particular, the system has the property that *increasing* the key length can make it *less* secure, e.g. in Figure 3, going from 512-bit RSA to 1024-bit decreases the break time from 8068 seconds to 177. AM says he’s never encountered this phenomenon before.

JHD subsequently forwards him the paper, saying

I also (still) like the last sentence of the paper.

[More generally, we have shown that a cryptosystem built according to standard principles of protocol layering with “standard” compo-

¹JHD subsequently re-read [CGH04]. Indeed, the attack they describe in the paragraph “Using this notion” (p. 560) is little more than this.

nents displays unexpected, and in some cases computationally trivial, vulnerabilities.

He replies

The last sentence is indeed very nice (as is the second sentence of the Conclusions) [which makes the point about the decrease].

A.1.3 The rôle of Theorems

AM admits the paradox that, though much of his work is in “provable security”, he does not agree that “proofs” are the be-all and end-all of cryptography. He quotes the case of RSA-OEAP, whose history he sees as follows.

1. Invented at IBM, but only uses one round of Feistel.
2. Taken up in [BR94], who use two rounds of Feistel, and have a proof. They do not have the IBM team as co-authors “because they didn’t have a proof”.
3. A hole is discovered in the proof of OEAP-schemes [Sho02], even though it is “essentially by accident, rather than by design” that RSA-OEAP is secure: “however, this fact relies on special algebraic properties of the RSA function, and not on the security of the general OAEP scheme.”
4. In fact, one round of Feistel *is* sufficient.

Nevertheless, Bellare & Rogaway still get the credit, even though, by the own logic (2 above) they don’t deserve it.

A.1.4 Other papers

He draws my attention to, and subsequently forwards me a copy of a forthcoming paper by himself and the Koblitzes² describing how elliptic curve cryptography came to be regarded with “almost unquestioned acceptance”.

He also sends me a copy of [Kob07]³. Apparently there is a fashion (fad?) for automated theorem proving in cryptography. He (Menezes) is fairly disparaging of this, saying that “all the hard steps have to be done by hand”. JHD is too polite to question his host, but notes subsequently that Koblitz actually writes

Note: Here I am not referring to attempts to formally verify proofs that have already been written out in detail by mathematicians.

Koblitz also questions whether this is “theorem proving”, and writes:

As in other areas of the Artificial Intelligence field, extravagant terminology has outpaced actual accomplishments.

²A.H. Koblitz *et al.*, Elliptic curve cryptography: The serpentine course of a paradigm shift, *J. Number Theory* (2009), doi:10.1016/j.jnt.2009.01.006.

³Part of a series [KM07a, KM07b, KM08].

Although JHD did not know of this particular quote at the time, the conversation was very similar. JHD drew attention to [McD81] and the tendency to wishful thinking in AI. McDermott discusses the following (JHD's recollection).

Consider the following statements.

1. Jumbo is an elephant.
2. Elephants have tusks.
3. Elephants live in Africa.

How does one deduce that Jumbo has his *own* tusks, but lives in the same Africa as all other elephants.

One might be tempted to argue that it's the capital letter, but then how does a German-speaker make that inference in German?

This, JHD opined, is a key cryptographic issue, since much formal proof in cryptography is about "who knows, or can deduce, what when", and it can be very hard to formalise this.