SYNASC 2011

Notes by J.H. Davenport — J.H.Davenport@bath.ac.uk

26–28 September 2011

Contents

Chapter 1

26 September

1.1 What can Symbolic Computation do for Mathematics? — Franz Winkler

What is Symbolic Computation? Quotes the definition from Wikipedia. Notes this cites [Buc85] as a description. His view is that symbolic computation is characterised by the fact that the input *and output* are mathematical expressions, normally exact rather than approximate. A non-comprehensive list of topics would be

- Computer algebra
- Formal proof
- computational and symbolic geometry.

An exemplar is the work in the 1960–70s on computation of greatest common divisors [Col67, Bro71]. Relies on resultants. Resultants are *in* elimination ideals, but do not necessarily *generate* them. Therefore Gröbner bases [Buc65], which do compute the *generators* of the elimination ideal. Example: a GB consisting of four polynomials in three variables (and not equiprojectable).

1.1.1 Equational Theories

First-order theories are those whose only predicate is `=`, and where axioms are universally quantified. Examples: groups, rings, modules (but not fields). Many examples in computer science. In such a theory, can we automatically prove or disprove results here? Examples in groups and queues.

1.1.2 Symbolic Computation as a Holistic View

Evolution lets us see that the same structure can evolve independently: wings in birds, insects, bats etc. Claims that "completion algorithms" are a similar

instance. Gaussian elimination, Euclid's algorithm, Buchberger's algorithm, Knuth–Bendix.

We are given a structure $\mathcal{S}$, and a congruence relation $\widetilde{=}$ on it. We introduce $\rightarrow_G \subset \mathcal{S} \times \mathcal{S}$ (but viewed as a rewrite left→right), with the properties that $\rightarrow_G$ is

- Compatibility with $\widetilde{=}_G$
- Noetherian (terminating)
- Church–Rosser (confluent) — or at least we hope so.

If so, we can check $a \widetilde{=}_G b$ by reducing both (terminates) and checking equality.

- **Vector spaces**. $\mathcal{S} = V$, $W = Span(B)$ is a subspace, and $a \widetilde{=}_G b$ iff $a - b \in W$. We can define $\rightarrow_B$ (elimination w.r.t the first relevant non-zero of $b \in B$), which is Noetherian, but not Church–Rosser. Gaussian elimination is transforming B to make $\rightarrow_B$ Church–Rosser.
- **Euclid's Algorithm**. $\mathcal{S} = K[x]$. $F = \{f_1, f_2\}$ generates an ideal I. $f \widetilde{=}_I g$ iff $f - g \in I$. Can define reduction as taking the remainder, but this is again not Church–Rosser. Euclid's algorithm transforms the set to make the reduction Church–Rosser.

* Both of these can throw away an old element every time we add a new one, but this is *not* inherent.

- **Buchberger's Algorithm**. A sort of Euclid for multivariate polynomial ideals. This doesn't have (*), which may account for the difficulty in discovering it.
- **Knuth–Bendix** [KB70]. Note that this does not always terminate. In Group Theory, it does, and produces 10 rules which are a confluent system for Group Theory.

1.1.3 Rational Solutions of algebraic ODEs

[NW10] — uses differential equations, differential algebra and polynomial theory.

1.1.4 Epistemology

What is "knowing" an object — existential or constructive, and, if so, how? He quoted [Her26] as an example of a constructive view which was ignored for many years because the demand wasn't there.

1.2 Tracking method for reparametrized geometrical constraint systems – Imbach *et al.*

We have a set of geometric objects and constraints relating them (collinearity etc.), known as a GCS (Geometric Constraint System). Essentially: unknowns, parameters and constraints (relating unknowns and parameters). A 'valuation' gives values to the unknowns, and a solution is a valuation that satisfies the constraints. There are algebraic methods (Ritt–Wu, Gröbner), numeric methods (Newton, homotopy), geometric knowledge-based methods, and the Locus Intersection Method (LIM). This is the aim of this talk.We construct a hypergraph, where vertices and objects and edges are constraints. Then backward-propagation of degrees of freedom. This gives us a Construction Plan (CP). We then evaluate the CP, noting that there may be branches in the CP. Apparently, a bad example is the hexagon. Therefore, given a GCS G, we add and remove constraints to give G' which is soluble by LIM, which we solve and then process to get a construction of G. The constraints added/removed are defined by the back-propagation.

Let d be the number of removed (or added) constraints. $\mathbf{k} = (k_0, \ldots, k_{d-1})$ is the parameters of these constraints. Want an evaluation $F : \mathbf{R}^d \to \mathbf{R}$, but there is no analytic expression for this. We try to find a solution by a continuation method. Given this, we find others by a relaxation method: remove one constraint, and follow the curve from the known solution to find other points where the removed constraint is still satisfied. For larger problems (e.g. icosahedron, 12 parameters and 30 constraints), this method is much faster than sampling: 17 seconds against 767.

Q–JHD It's not obvious that the relaxation will find all solutions.

A We can't guarantee that. Especially if the curve has multiple real branches, relaxation won't get us from one to another. In practice, we find many solutions.

1.3 Long Integers and Polynomial Evaluations with Estrin's Scheme — Zanoni & Bodrato

Evaluate $p(x) = \sum_{i=0}^{d} a_i x^i$ at a long integer x. Assume a_i and x both have about n bits. Costs$A(m, n)$ and $M(m, n)$ for adding and multiplying, and assume $A(m, n) \approx \min(m, n)$.

Ruffini–Horner $d + O(1)$ multiplications.

Pan $d/2$ multiplications

Stockmeyer $O(\sqrt{d})$ operations [PS73].

but all in the classical definition of M. We have faster methods known as Karatsuba [KO63], CookToom [Coo66, Too63], FFT [SS71], all implemented,

Figure 1.1: last substantive slide

In the world of (generalized) power series, the branch cuts

- Only appear in the expansions about the singular points
- Their directions are coded in the arguments

a) $\log z$ = branch cut heading west, adhering north

b) $\log(iz)$ = branch cut heading north, etc.

- Their adherence is coded similarly

c) $-\log(1/z)$ = branch cut west, adhering south

and [FÖ07], of only theoretical interest. Note that Ruffini–Horner produces unbalanced multiplications. Hence $E_{RH} \approx M(N)\frac{D(D-1)}{2}$. Let $\alpha = \log_k(2k-1)$. Then $E_E \approx O(M(n)D^\alpha)$, which is roughly $O(M_{fast}(Dn))$, whereas E_{RH} is roughly $O(M_{classical}(Dn))$. The analysis for small coefficients differs in details, but not in principle. With $d = 63$ and large coefficients, we are 20% of the cost of Ruffini–Horner.

For $d = 2$ $E_{RH} = M(2n, n) + M(n,) + 2A(N)$, whereas $E_E = M(2n, n) + M(n) + 2A(N) + S(n) + A(n)$.

We have shown that Estrin can be efficient, and faster in some cases. This, and our variants, should be considered whenever fast multiplication is being considered.

1.4 JHD on Branch Cuts

As I was speaking, there are no real notes. Questions focused on the last substantive slide (Figure 1.1), which let JHD explain the connection with BS's previous work, and "what do we do about branch cuts", which let JHD speak about [BBDP07].

Chapter 2

26 September — HPC

2.1 Communications Schemes of a Parallel Fluid Solver for Multi-Scale Environmental Simulations — Frisch

Environmental simulations with CFD require enormous computational effort. Large geometry domains and necessary for sufficient numerical resolution of physical phenomena. Hence parallel computation. The geometry of a large power plant contains more that 12M triangles. Therefore we need adaptive grids.

Each $n \times m \times l$ data cells is surrounded by a layer of pseudocells, holding links to neighbouring sub (? or super) grids. In the parallel case we use MPIv2 to communicate between cells. Typical equations are Navier–Stokes $\nabla.u = 0$; $\frac{\partial}{\partial t}u + (u.\nabla)u = -\frac{1}{\rho}\nabla\rho + \nu\Delta u$.

We must be careful with synchronicity. Each cell issues a non-blocking `MPIreceive` for each refinement required. Then `MPIwait_all` is used to block for all needed values. We then communicate, and execute `MPI_send`. This satisfies the bottom-up requirement. We then do 'horizontal' communication, via ghost cells. Finally top-down to propagate to lower levels.

2.2 Gaining Experience in BlueGene/P Application Development. A Case Study in Remote Sensor Analysis — Pianica

2.2.1 Computer

Compute cards is 1 chip (4 processors) I/O cards have 10Gb optical links. Each processor has its own L1, and prefetching L2. There are 2 4Mb L3 caches, with each pair of processors connected to a multiplexing switch with connections to

both L3. Each L3 is connected to a certain amount (?) of memory.
There are various modes

? One process per processor, spawning up to four threads.

VN Four processes per processor, shared memory between the cores.

Dual Mode Two process per processor with up to two threads each.

850MHz processors, 32K L1/core, 14 strand L2 prefetch, 8MB L3, Main memory 4G/node, Main store bandwidth 16GB/sec. The interconnect is a 3D-torus, which interconnects all compute nodes. Adaptive cut-through routing. 0.5μsec latency between adjacent nodes.

2.2.2 Use Case

Using unsupervised classification (clustering) of satellite images to identify regions with similar characteristics. Pixels may contain spectral information corresponding to several ground components. We assign possibly several values (fuzzy clustering). Because of noise, we also make use of spatial information. Increase in both spectral and spatial resolutions leads to huge datasets. Hence spatial, or spectral, domain partitioning, and parallel implementation of clustering. Input is n pixels, with a number of spectral bands in each, number of desired clusters c. Output is

- Matrix of classifications $n \times c$
- ...

[KSLT02].

2.2.3 Old and New Systems

- Intragrid Xeon:16 nodes, 8 cores/node 1.25GB/core
- Blue Gene (see above)

We had used MPICH-2 on Intergrid, but BlueGene comes with its own MPI. A simple port did not work well: changing compilation flags `-O3 -qhot -qipa=level=2` brought a factor of 10. Square partitioning did $\times 8$ better than either vertical or horizontal. Our algorithm was efficient on Intergrid (max 128cores); on BlueGene fine up to 256 cores, then tailed off rapidly to 30% efficiency at 1024 cores.

2.2.4 Conclusions

- We needed to make serious modifications to make true use of the 3D torus topology.
- Compilation flags are vital.

Q–EK Why didn't IBM just tell you to use these flags?

***** Apparently it's not in the documentation, but is discussed in the training courses.[1]

A Well, it depends on the application. There is some kind of auto-profiling, but this only got us half-way to where we got by hand-tuning.

2.3 A Scalability Study of Two Meso-scale Weather Prediction Models — Slusanschi

2.3.1 Hardware

32 nodes of dual quad-core Xeon 2GHz; 20 nodes of dual hex-core AMD Opteron 2.6GHz. 16GB/Node, Xeon was Gb Ethernet, Opteron Infiniband. Storage 36TB.

***** Both software systems come from Germany

HRM High-resolution Regional Model. Fortran-90 with C I/O. Hydrostatic model. C-horizontal grid with second-order centered differencing. 301×201 grid with 32 vertical layers. 20km resolution. Split semi-implicit time stepping, with FFT or Gauss Solver. Parallelization is OpenMP and MPI. Although it's a 3D problem, the vertical layer is all on one processor, and we have a $p_1 \times p_2$ horizontal grid (for us 2×8 or 2×16). Main conclusion was that it was I/O bound in the post-processing phase. 8 processors gave $4.5\times$ with post-processing, $\times 7.5$ without.

COSMO Consortium for Small-scale Modelling. 81×73 grid with 14km resolution (can be refined to 2.8km). Based on the primitive thermo-hydrodynamical equations describing compressible flow in a moist atmosphere. Uses a generalised terrain following height coordinates. MPI parallelization strategy. 2D horizontal domain decomposition, with subdomains surrounded by two halo gridlines belonging to neighbouring processors. Observed a lot of small (1KB) messages.

Proposed solutions were dedicated I/O processes, and tune inter-processor communication. The dedicated I/O processor wasn't a great help on the Ethernet machine.

For COSMO, original method was `SendReceive`. Immediate send/blocking receive was about the same up to 8 processors, then on 16 showed $11\times$, whereas `SendReceive` was $7\times$. Blocking send/immediate receive was also better than `SendReceive`, marginally worse than the converse.

The Opterons scale marginally better, $\times 5$ for 10 processors for HRM, versus $\times 4$ for the Xeon.[2]

[1] Cynically: "We guarantee your code will run faster after you've been on the training course. The course teaches `-dinvoice` option!"

[2] JHD notes that the Opterons were Infiniband, rather than Gb Ethernet.

Chapter 3

26 September — plenary

3.1 Searching Simulation Scenarios on the Grid with ELSIG Explorer — Muntean

Searching for relevant simulations and scavenging the results gives resource savings. But there are many such simulations, small semantic differences between scenarios and the distributed nature of the grid is an additional challenge. Existing tools are too coarse. We are interested in computational neuroscience. Hodgkin–Huxley, Abbott–Kepler etc. are used for biological neural microcircuits, which leads to Spike Response Model, Skip-time dependent Plasticity etc. Hence various systems such as Brian, NEURON, GENESYS and (our choice) Neutron.

There is IR work — GIR is infrastructure for distributed IR using grid services, and GRACE project is an IR system for search and categorization. These are aimed at documents, not scenarios. Our Approach, ELSIG, uses a vector space model.

We use the GRIDSFEA (Grid Simulation Framework for Engineering Applications) tool to build end-user grid applications. Developed originally for GT4 at TU München, and now mainly at UTC-N. Enhanced Latent Semantic Indexing on the Grid (ELSIG) is a prototype for searching simulation scenarios on the grid. Uses an LSI algorithm and a linguistic processing model. Meta description of scenarios gives a large number of existing simulations. LSI is used for small semantic differences between scenarios. Application plugins for GRIDSFEA are used for the distributed nature of the grid environment.

ELSIG currently runs on the end-user machine, but the aim is to have data collection and index building done on the grid itself. The Latent Semantic Indexing was enhanced: weighting functions etc. Attributes of simulation metadata include:

- scenario identifier
- identifier of the computation phase

- location of checkpoints
- list of results and post-processing files
- human-readable description of the computer scenario.

Our definition of simulation services was based on WS-GRAM (GT4) which is now obsolete.

Typically the top-5 hits contain all the relevant scenarios, for our tests on the MED benchmark. Index-building is an expensive operation (minutes) but query response is fast (below 0.5 seconds).

3.2 Pattern Detection Model for Monitoring Distributed Systems — Dinu

Distributed systems are important in scientific computation, content distribution networks and wireless sensor networks. Claims that pattern detection and classification is the basis of automated management. Our tool is called MonALISA. There is a choice between statistical and structured approaches. Both approaches tend to be univariate, while the real world has many variables. Our model is Bio-inspired (organization of biological neural networks), multivariate, and AI-friendly (example-based learning). We take various input signals (load, memory, Ethernet traffic etc.) and apply functions like linear regression. We used synthetic and real data, testing on technically-literate subjects, and got 90% success rate. Hence we deduce that our model is general and expressive, and is automatically learnable[1]

3.3 Grid-based services and tools for Hydrological Model Processing and Visualisation — Bacu

Envirogrids is a project to gridify the Black Sea catchment to support its sustainable development. We use SWAT — Soil Water Assessment Tool. It's a hydrological model operating on a daily time spent. Used to predict water resources, sediment and chemical yields in a specific watershed. Needs lots of input data: weather, soil (1–3 GB of data), topography, vegetation etc. ArcSWAT is the uncalibrated model. From running this and comparing with reality, we produce a calibrated version with gSWAT.

The calibration process consists of several iterations, and an iteration of several simulations. Ganga is used to manage the jobs involved. There is a graphical interface to create the project. Presented performance data, typically with 150 simulations and 30 compute nodes. At this level, more nodes give greater throughput, practically 1:1.

[1] Meaning, I expect, that the model is learning, rather than being learned about.

3.4 A Data Dissemination Algorithm for Opportunistic Networks — Dobre

Opportunistic Networks are networks composed of mobile nodes that may not have a direct connection between them. Routes are built dynamically, and nodes act on a store-carry-forward paradigm. Previous solutions have been Socio-Aware, DTN, ContentPlace (henceforth CP) etc. In Social Dissemination (henceforth SD), we take advantage of the social grouping of nodes in communities. Each node stores

- Required Objects — information required by the user
- Data memory — information received from other nodes
- Cache memory
- Published data.

So when two nodes are in range:

- advertise stored data
- analyse received data (advertisement) to find needed objects
- if needed object aren't found, analyse received data description to find objects that can be disseminated (utility function)
- send back the hash of these
- actually collect the data

$$u = \sum_{i=1}^{n} p_i^c(1 - p_i^e) + freshness + c_v$$

where n is the number of communities, p_i^c is the percentage of nodes from community encountered so for, p_i^e is the percentage of nodes from community i that the encountered node has been in contact with before. Based our model on HCMM. We compared with data from ContentPlace. 45 nodes into 3 communities. 1000×1000 gid divided into 16 cells. 3 channels, each generated 99 data objects We looked at hit rate, fairness (Jain's index), resource consumption and latency. Various policies: SD and seven from ContentPlace: MLN, Future, present, Greedy, Uniform, MFV, Uniform. In terms of resource utilisation, SD is outperformed only by greedy. The highest hit rate is obtained when the number f channels is equal to the number of communities. SD's fairness is very close to 100%. No CP policy outperforms SD on all four metrics. SD seems to scale well with number of nodes and number of communities.

Chapter 4

27 September

4.1 What is Hybrid Symbolic–Numeric Computation? — Kaltofen

Example of model discovery. What is the best fit for "random curve". Colella's "7 Dwarfs" of computation. Structured Grids, unstructured grids, FFT, Dense LA, Sparse LA, Particles, Monte Carlo. `http://view.eecs.berkeley/...` — Berkeley's 13 dwarfs. EK's 7 dwarfs of symbolic computation (SNSC 2008, [Kal10]). To appear in SFB-book (Linz).

4.1.1 Probabilistic Analysis with Numeric Data

[gfenLeeYangatSNC2011] By sampling a black box, compute a t-sparse representation $f(x) = \sum_{j=1}^{t} c_j x^{d_j}$ with $c_j \neq 0$. [GLL09][1] — numeric sparse Ben-Or/Tiwari. Rational functions $\frac{f(x)}{g(x)}$ by [KaltofenYangZhi2007] — numeric Zippel Lemma, sparse Zippel interpolation.. Applies to sparse signal processing [Leeetal2011].

For the exact problem, w have early termination [KaltofenLee2003]. Pick a random ω, evaluate f at $\omega^i : 1 \leq i \leq 2k-2$. Write $h_i = f(\omega^i)$. Make the h_i into a $k \times k$ Hankel. Terminate when this gets singular, and with high probability this is right.

What to do in the presence of noise? Can we identify whether $\Delta(z_1, \ldots, z_s) \stackrel{?}{=} 0$? Let $\zeta - j = \exp(\frac{2\pi i}{p_j})$, p_j prime, then for random r_i, the expected value of $|\Delta(\zeta_i^{r_1}, \ldots, \zeta_s^{r_s})| \geq 1$. Doesn't work, since the problem is ill-conditioned. [Rump2003] tells us the distance to the nearest singular Hankel matrix: $||(H^{[t+1]})^{-1}||_2^{-1}$. So how input-sensitive is $\det(H^{[t+1]})$?

Claims that algorithms can be unstable, but only problems are ill-conditioned. So the determinant of the zero-matrix is a well-conditioned problem, but Gaussian elimination is an unstable method for it. We also need to ask about the

[1] Also 2003 conference paper.

Zippel Lemma with floating-point ζ_i.

The Gohberg–Semencul Formula tells us the whole of H^{-1} in terms of the last row and column. there is an open problem of computing in $O(t^2)$ the actual condition numbers of the leading submatrices. Note that noise does not cause explosion of terms, as it would in the exact case. Very sparse signals occur in medical signal processing `http://sparsecare.be`

4.1.2 Exact Certificates

$O(n \log n \log \log n)$ certificate for $\gcd(a, b)$. Store Bezout coefficients. A certificate is an input-dependent data structure and an algorithm that computes from that input and its certificate the specified output and that has lower computational complexity than any known algorithm that does the same when only receiving the input. Note that correctness of the data structure is *not* assumed, but validated by the algorithm.

How to certify $C = A \cdot B$ [Fre79]: check (Cy)=A(By)). Monte Carlo $n^{2+o(1)}$ and the certificate is C alone.

Rank certificates [KNS11]. There are three kinds of balls: truths, undetectable lies, and detectable lies, and the number of undetectable lies is bounded. Then take a ball[2], and check it's not a detectable lie, then with high probability it's truth. $x^4y^2 + x^2y^4 + z^6 - 3x^2y^2z^2$ is Motzkin's polynomial. Positive semi-definite, but *not* a sum of squares. It is if you multiply by $x^2 + z^2$.

Hence we have a hard case of finding sums-of-squares representation [KLYZ08, KLYZ11], which requires rank certification. We have $O(n^{2+o(1)}$ complexity, by running Storjohann's Las Vegas algorithm, recording all random choices and intermediate results (except matrix multiplies, for which we store inputs/outputs only), and to verify, re-run, verifying the matrix multiplications via Freivalds [Fre79]. This is equivalent to running a rank algorithm with quadratic matrix multiplication.

4.1.3 JHD/EK

JHD quizzed EK about the rank certificates. A certificate presented to us might be any of

- A genuine certificate, which is the equivalent of a "lucky prime"
- An undetectable lie, which is the equivalent of a "bad prime", in the usual modular sense, and there are only finitely many of these, since these are the primes that divide a certain non-zero determinant. By choosing a large enough sample region, we can make the ratio of these two types sufficiently close to 1.
- A detectable lie, i.e. a certificate that doesn't certify. In this case, we reject the certificate (**not** our sample).

[2] JHD has doubts about the analogy, at least. See section 4.1.3

4.2 A Theory and an Algorithm for Approximate Gröbner Bases — Sasaki

EK's introduction — Sasaki is about the first person to work in SNC.

Floating point GB: with conventional definition and algorithm, compute a GB. Naïve methods are very unstable, and we get $\{1\}$ very often. Hence we need to understand cancellations.

Systematic/Exact Harmless

Systematic/Inexact Intrinsic — the system has an approximate linear dependency.

Accidental/Exact Tractable

Accidental/Inexact Not yet.

[KatoSasaki1997] introduces an efloat, which lets us detect approximate syzygies: $P = \sum A_i F_i$ with $||P|| < \epsilon||A_i||$ means there is an approximate syzygy amongst the F_i. Note, however, that reduction can cause a loss of accuracy, so introduce 'accuracy-guarding reduction": in the reduction of P by an intermediate basis.

Shows an Approximate GB algorithm, and a theorem about approximate reduction, essentially the standard ones with clauses on error bounds to ensure fidelity.

4.3 A computer-aided software for nonlinear digital control — Tanasa

There are various control design strategies, depending on the considered time-domain:linear dynamics(generally numeric), nonlinear dynamics (symbolic packages in Macsyma and specialised ones) and sampled-data systems, where we propose SymNLSys. This has an I/O Matching module.

$$\dot{x}(t) = f(x(t)) + u_c(t, x(t))g(x(t)); \qquad y_c = h(x(t))$$

which we discretise.

[MonacoNormand-CytorCIFA2008] gives a theorem for existence of a solution. But in practice the solution of the linear system is not always possible. example: 2-link planar manipulator, shows that his method produces a much better fit to the continuous-time solution.

4.4 Experiences in computing mesh root systems for Dynkin diagrams using Maple and C++ —

Dynkin diagrams. Examples from $\mathcal{S} = \{A_n, D_n, E_6, E_7, E_8\}$. Let $\Delta = (\Delta_0, \Delta_1)$ with $\Delta_i \in \mathcal{S}$. The matrix Morsification of was introduced by Simpson. $Mor_\Delta \times W_\Delta \to Mor_\Delta$.

These are important for Diophantine equations (Hilbert 10th), classification of finite Tits geometries, complex simple Lie algebras etc.

The main tools are Maple, Eigen, GiNaC (GiNaC is Not a CAS) and GAP. All systems fast us to $n = 5$, when $n = 6$, Lu beat QR, and both seriously beat GiNaC. Examples from A_5

Q–EK Why C++?

A Speed for the linear algebra.

4.5 A Distributed Approach for solving systems of nonlinear equations — Mocano

SPICE, the bible of simulation, was built around newton–Raphson. The problem with this is the Jacobian. We therefore generate a linear approximation, solve this, and either access or iterate. Note that we have a pivoting problem, which affects data distribution (or vice versa).

- Row distribution — search for pivot in the row, by column swaps.
- Checkerboard distribution — generally faster if no pivoting takes place, but pivoting is more complex.

3.5 seconds on 1 processor, 1 on 4 processors, 0.9 on 8 and 1.5 on 16 (of 1.5, 1 is taken in pivoting, whereas negigeable in 1 or 2 processors). Each processor is 2 quad-core Xeons.

4.6 Parallel Migration Model in Competitive Differential Evolution

Interested in global optimization for $F : \Omega \to \mathbf{R}$, where $\Omega = [a_1, b_1] \times \cdots \times [a_D, b_D]$. We use differential evolution, by

- mutation, with a control parameter $F \in (0, 2)$
- crossover, with a control parameter $CR \in (0, 1)$

- selection

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(nj + n_0)}$$

  where n_h is the number of successes of strategy h.

Migration divides the individual strategies into sub-populations each on its "island" evolving via DE. There is a topology of a migration model,a nd after several generations, selected individuals migrate.

We compare two sequential DE against 6 parallel versions. The two sequential were `b6e6rl` and `b9e9rl`. Parallel

- star topology
- population is divided uniformly among the islands
- Each island is linked only with mainland
- One best individual is **copied** to mainland after every node(=10) generations, but only if better than all (? JHD's reading of formula — confirmed in questions) mainland individuals.

In general, parallel is faster, but has less reliability — see proceedings. On difficult test problems, 3 of the six parallel methods had very little success. `b9e9rl` was the best sequential one to use inside the parallel evolution. We could reduce costs by a factor of 5.

4.7 Self-organising context-aware agent systems — Florea

Agents may be both software and hardware. In both cases, they are in, and interacting with, and affecting, the environment, which is complex and dynamic. Therefore the design of a system with *pre-determined* behaviour is a significant challenge.

4.7.1 Self-organising

Definition 1 Self-organization *is the process in which a pattern at the global level of a system emerges sole;y form the numerous interactions among the lower-level components of a system.*

Definition 2 Emergence *is defined by its effect as the resulting properties of a system that are not captured the properties of the parts, such as the formation of patterns in the structure or behaviour of the overall system.*

Definition 3 (Viscount Ilya, Nobel prize-winner) *Self-organization can be caused by external or internal pressures.*

S we need a two-level description: micro- and macro-level. Can we have emergence without self-organisation or *vice versa*? The micro-description is usually unique. The macro-description is generally not unique, and needs an observer to observe the behaviour, and the macro-description can depend on the observer.

The anticipated behaviour of an artificial system with emergent behaviour is a cornerstone in the design of such a system, as emergent properties may be desired or undesired ones. When we have cognitive agents, there is both the cognition at the level of the designer, and at the level of the cognitive agent. Hence such an agent is *also* an observer: albeit internal rather than external. By changing its behaviour, such an agent can change the emergent behaviours.

An example of a Reactive MAS with self-organisation is Conway's 'Life'. This is easily encoded in her formalism. The simple form of emergence is the glider. Behaviour depends on the rules and the initial state, but also on the rule selection strategy..

For cognitive MAS, agents have some competencies, and may therefore solve tasks themselves or delegate to other agents. Agents have belief about their own competencies `SelfCap` and also beliefs about the capabilities of other agents. After each encounter, an agent updates belief about the other agents.

4.7.2 Context-aware

Definition 4 Context *is any information about entities that is relevant to the interaction between a user and the system.*

Context-aware systems are able to adapt their operations to the current environment without explicit user intervention. Mobile systems are a good example. Context can be external, internal or computational (available resources, network quality etc.). Typically graph-based representations and/or ontologies.

4.8 IBM Supercomputers — Energy efficiency with iDataPlex cluster — Manaila

iDataPlex is an innovative x96 solution. It's a half-depth server. Cabling is run at the front, to make maintenance easier. Compared to a standard racking, it needs 3 PDU rather than 4 (notes that many co-location facilities charge by the power feed!).

Interconnect can be 1Gb, 01Gb Ethernets, Infiniband or Fibre Channel. Compute node is (2?) quad- or hex-core Xeons, with $\leq$ 128GB memory. There is also a GPU-card (nVidia) option: Tesla or Quadro 4000/5000 versions. Claims 1 server + 2$\times$Tesla is a 1Tflop system (JHD suspects this may be single precision).

Windows 2008, VMware, SUSE and Red Hat (EL4/5/6) are the supported operating systems.

Next generation of this (warm water-cooling to the rack) has been chosen for the Leibniz RZ 3Pflop system, under the name of Aquastar.

4.9 First-Order Theorem Proving with Vampire — Voronkov (& Kovács) (T)

The original purpose of theorem proving was to prove theorems.

Theorem 1 (example) *In group theory, $x^2 = 1$ implies the group is commutative.*

So we need to include the axioms of group theory (standard 3).

TPT syntax: variables begin with upper case letters, constants are nullary functions, and the syntax is `~ & | => <=> ![X1,...Xn]:F` and `?[X1,...Xn]:F` (the last two are universal and existential quantifiers). . Vampire is a totally non-interactive prover. Main applications are software and hardware verification.

Proof by refutation of $f_1, \ldots, f_n \Rightarrow g$ is unsatisfiability of $f_1, \ldots, f_n, \neg g$, but Vampire, like others, treats this last specially for efficiency. General scheme is

1. Read problem
2. determine proof-search options
3. preprocess the problem
4. convert to CNF
5. Run a saturation algorithm

An inference system is a set of inference rules. Example: inference system for (unary) arithmetic, with ϵ denoting 0, and $|n$ denoting $n + 1$.

Binary resolution BR has two rules: BR $\frac{p \vee C_1 \quad \neg p \vee C_2}{C_1 \vee C_2}$ and factoring $\frac{L \vee L \vee C}{L \vee C}$ [C can be an empty clause, in response to JHD's question] (in response to a question from TJ: we assume that $\vee$ is associative and commutative). This is a sound system of inferences (he gave a definition of "sound" in terms of "logical consequence", which seems somewhat circular).

A literal selection function σ selects a literal from a (non-empty) clause. Then BR_σ has BR only if p is selected, and Fact only if p is selected and positive. Then there is a selected set which is unsatisfiable, and proved by BR, but not provably so by BR_σ.

Take a well-founded ordering $\succ$ on atoms. extend it by $p \succ q$ implies $p \succ \neg q$ and $\neg p \succ q$. Also $\neg p \succ p$. A literal selection function is well–behaved w.r.t $\succ$ if, whenever all selected literals are positive, all maximal literals are selected. In fact BR_σ is complete for any well-behaved σ: the example above had a non-well-behaved σ.

How do we establish unsatisfiability? Initially $S = S_0$, keep taking inferences and add to S. If at any point we find the empty clause, we report unsatisfiability.

Definition 5 *A set$S$ is said to be* saturated*with respect to I if every inference by I from the set S is already in S.*

Hence if we reach a saturated set, we can report 'satisfiable'.

If we have an inference $S_0 \Rightarrow S_1 \Rightarrow S_2 \ldots$, then the limit is $\bigcup_i S_i$. Suppose that we have an infinite inference process such thatS_0 is unsatisfiable, we use use binary resolution. Question: does completeleness imply that we must reach the empty clause?

Definition 6 *An inference process $S_0 \Rightarrow S_1 \Rightarrow S_2 \ldots$ is said to be*fair *if, whenever*

$$\frac{F_1 \ \ldots \ F_n}{F}$$

is an I-inference, and $\{F_1, \ldots, F_n\} \subset S_\infty$, then $\exists i : F \in S_i$.

Theorem 2 *The following conditions on I are equivalent*

1. *I is complete*
2. *For every unsatisfiable set of formulas S_0 and any fair I-inference process with the initial set S_0, the limit of this inference process contains the empty clause.*

Robinson already noted that C subsumes $C \vee D$ and we can remove subsumed clauses. How do we prove that completeness is preserved if we remove subsumed clauses? We need to extend $\succ$ to clauses.

Definition 7 *A clause C is* redundant *in a search space if it is a logical consequence of clauses strictly smaller than C.*

This leads to a new kind of inference process:

1. Add an inference
2. delete a redundant clause

This causes problems with the definition of fairness. A clause is persistent if, once present, it will never be deleted.

Theorem 3 (Completeness) *Let $\succ$ be a simplification ordering and σ a well-behaved selection rule. Let also*

1. *S_0 be a set of clauses*
2. *$S_0 \Rightarrow \ldots$ be a fair BR_σ inference process*

Then S_0 is unsatisfiable if and only if some S_i contains the empty clause.

4.10 Task scheduling

The general problem is NP-hard. We assume

- non-preemptive scheduling
- tasks are single-instance
- multi-processor target architecture

$T = (s, c, d)$ where s is a (can't start before) start time, c is cost and d is deadline.

We previously considered the case where all $s_i = 0$. Here we sorted $d_1 \leq d_2 \leq \ldots$, and $\ldots$, we got a lower bound.

But now it is possible that CPU utilisation is $< 100\%$ since tasks may not be ready yet. Let

$$R_i = \underbrace{\sum_{j=1}^{i} c_i}_{\text{what has to be finished}} + \underbrace{\sum_{j=i+1}^{n} (c_j + d_i - d_j)}_{\text{what has to be started}}$$

This is what we **must** do, and this drives our algorithm.Then we combine EDF (Earliest Deadline First) and LLF (Least laxity First). In fact, we use LF, sometimes switching to EDF, essentially as a tie-breaker.

4.11 Reachability of TimeBasic Petri nets

A petri net has places, transitions, arcs and tokens, to which we add TimeStamp, attached to Tokens, TimeFunctions, attached to transitions, and TimeSemantic, which may be weak or strong. There's a classic GasBurner example. A *symbolic state* is a way of representing infinitely many state which differ only in timestamps. There is a state inclusion relationship. Absolute-time references can generate infinite states in cases when relative times do not (shows a simple example). If a timestamp will never be used in order to establish how a net starting from a marking will evolve, it is possible to anonymise such a timestamp, which simplifies many cases.

Using their parallel version (I think the application was the GasBurner, but not sure), instead of 7.5 hours, it took 1.5 with any of

- 6-core machine and multithreaded
- 50-machine cluster (old machines)
- 150 MapReduce Amazon EC2 nodes.

4.12 Program Synthesis

Starting from the specification, by applying a certain method, **find** an algorithm which solves the problem. This should

- let us obtain natural-style proofs
- build a theory
- by applying different induction principles we obtain different synthesised algorithms.

Given input constraints $I_F[X]$, and an output constraint $O_F[X, Y]$ fund an F such that $\forall X : I_F[x] : O_F[X, F(X)]$. We state our rules in prolog style.

For sorting, I_S is "true" and $O_S[X, Y]$ is "Y has the same elements as X and is sorted". `IsSorted` is defined by induction. She introduced a proposition that "concatenation is merging".

She also discussed the synthesis of merging. The cases where A or B are empty fall out, but for the remaining (the non-trivial!) case she has to use "special induction": the details were too hard to follow.

4.13 RangeLab: a static analyser

This is based on interval arithmetic and static analysis. Computes the ranges of variables in simple programs. This bounds the errors in simple fixed/floating-point programs. Related tools are Gappa (Melquiond), Fluctuat (CEA) and IntLab (Rump) in MatLab.

Output is an interval *and* an error of the interval. So for a bad trapezium, the interval is wide, but precisely known. It allows many f.p. formats, vectors and matrices, and standard control structures. The programming technology is abstract interpretation [CousotCousot1977].

Mathematically equivalent expressions are different w.r.t. "wrapping" effect. Hence we compute a set of equivalent expressions, evaluating the deepest nodes first, and also use interval slicing to reduce the interval spread.

Explanation of how `while` loops can be made to terminate. In the case of Gaussian elimination (the code seemed to include no pivoting) the intervals are wide, but with small errors.

4.14 CFG extraction and unfolding — Asavoae

[It turned out that CFG=Control Flow Graph]. "Abstraction by example". This is based on the **K** framework. Configurations represent the necessary entities to capture the program senatics denoted in **K** as (nested) bags of cells. Actual states are ground terms of the configurations. We are interested in assembly language configurations. He gave a concrete rule for the `beq` instruction:

$$\frac{\texttt{beq}(V_1, V_2, Addr)}{setPC(V_1 == V_2, Addr)}.$$

He appears to want to recognise/handle loops (JHD wasn't quite sure which: he thinks the compiler was generating pseudo-instructions to tell where the loop was).

Hence we can use a formal semantics of machine code (via **K**) in order to define abstract executing for the CFG.

Chapter 5

28 September

5.1 Hybrid fuzzy rule-based classification

Determine CF_j = degree of certainty = difference/sum. We classify patterns by $\max_q \mu_{i,q} CF_q$. On a very standard breast cancer database, we get the results in Table 5.1

A more recent technique is thermal imaging (differences between the two breasts). We have 38 features/thermogram. 146 cases, of which 29 were malignant and 117 benign. This then looks like a 38-dimensional problem: curse of dimensionality. So we limit the number of attributes in each rule to two. With 10–15 partitions, we get 75–8% accuracy.

A further question is gene expression levels. Here we have even higher dimensionality (1000s of genes) and fewer training samples (dozens). Colon data: 40 malign/22 normal; leukaemia: a 3-way classification and a third case (lymphoma), also binary. We did some data thresholding and exclusion of outliers, then a $\log_{10}$ transform. Colon — $L = 3$ beat alternatives, Leukaemia "nearest neighbour" won, and lymphoma $L = 4$. This contradicts earlier studies that $L = 2$ was always best.

A further development is to introduce costs of misclassification: in medical ones we don't want to miss cases of malignancy. This leads to *cost-sensitive fuzzy classification*. Now CF_j = degree of certainty = difference of weighted/sum of weighted. We get Table 5.2 if we weight mis-classifying a case which was actually malign as twice as serious as the other error.

Still not happy, so now suggests a learning algorithm for adjusting the CF_i. See Table 5.3.

Table 5.1: Breast cancer data

	Correct	Incorrect
Benign	438	6
Malignant	233	6

Table 5.2: Weighted breast cancer data

	Correct	Incorrect
Benign	437	7
Malignant	235	4

Table 5.3: Post-learning breast cancer data

	Correct	Incorrect
Benign	442	2
Malignant	239	0

Next phase is to use the fuzzy if-then rules as a population in a Michigan-style genetic algorithm. The training set determines a 'fitness' function. On the thermography data set, we get very comparable performance, but with 20 final rules, rather than thousands.

Or we can use a Pittsburgh-style genetic algorithm: individuals are rule sets, rather than rules. For colon , Michigan was slightly better (83% versus 82%) than Pittsburgh, for the other two, Michigan was significantly better.

- A learning algorithm can improve performance.
- A generitic algorithm can produce a classifier with similar accuracy, but *many* fewer rules.

Q–JHD Am I right in seeing that, in Pittsburgh, crossover only moves rules around between individuals, and it is only mutation that changes the contents of rules?

A Correct. Mutation is the GA operation that guarantees you will eventually find the optimum. Of course, we may not have run Pittsburgh long enough.

5.2 Program Assertion Synthesis using Symbolic Computation — Kovács (T)

5.2.1 Introduction to Verification

Verification means that a program satisfies its requirements. Note that test cases are equivalent to specifying a finite set of correct results only. Therefore we need to have the program (in today's tutorial, largely imperative programs) *and* the requirements.

Example 1 *Given natural numbers x and y, with $y \neq 0$ to compute the quotient* `quo` *and remainder* `rem` *of x/y.*

precondition P $x \geq 0 \wedge y > 0$

postcondition Q $\texttt{quo} \times y + \texttt{rem} = x \wedge 0 \leq \texttt{rem} < y$.

program code S We take the following

```
quo:=0;
rem:=y;
while y <= rem  do
      quo:=quo+1;
      rem:=rem-y;
     end while
```

Note the distinction between `:=` in programs, which implicitly has $\texttt{quo}_{old}$ and $\texttt{quo}_{new}$, and $=$ in the specification.

Hoare triple $\{P\}S\{Q\}$ The program S started in P, if it terminates, will satisfy Q (partial correctness).

5.2.2 Weakest precondition

Definition 8 *P is* weaker *than R of $R \to P$*

Weakest precondition $\text{wp}(S, Q)$ for S with Q, is such that for any R with $\{R\}S\{Q\}$, $R \to \text{wp}(S, Q)$. Therefore $\{\text{wp}(S, q\}S\{q\}$. Therefore for a sequence $s_1, \ldots, s_n$, we compute $\text{wp}(s_n, Q)$, and this becomes the postcondition for s_{n-1}, and so on.

$$\text{wp}(x := \texttt{expr}, Q) = Q_{x \leftarrow \texttt{expr}} \quad \text{(wp:assign)}$$

$$\text{wp}(s_1; s_2, Q) = \text{wp}(s_1, \text{wp}(s_2, Q)) \quad \text{(wp:seq)}$$

$$\text{wp}(\texttt{if } c \texttt{ then } s_1 \texttt{ else } s_2, Q) = (c \Rightarrow \text{wp}(s_1, Q)) \wedge (\neg c \Rightarrow \text{wp}(s_2, Q)) \quad \text{(wp:cond)}$$

What about loops:

$$\text{wp}(\texttt{while } c \texttt{ do } s \texttt{ end while}, Q) = W \quad \text{(wp:while?)}$$

Applying an unfold to the loop, we get

$$W = (c \Rightarrow \text{wp}(s, W)) \wedge (\neg c \Rightarrow Q).$$

W is known as a *loop invariant*. More formally

Definition 9 *I is a* loop invariant*, or* inductive assertion*, for* $\{P\}\texttt{while } c \texttt{ do } s \texttt{ end while}\{Q\}$ *iff*

0. $P \to I$
1. *Iterative condition:* $\{I \wedge c\}S\{I\}$
2. *Final condition:* $I \wedge \neg c \Rightarrow Q$

Look at Example 1. Here we can take

$$\mathrm{wp}(loop, Q) = (\texttt{quo} \times y + \texttt{rem} = x) \wedge (0 \leq \texttt{rem}) \wedge (0 < y) \wedge (x \geq 0).$$

Hence the main challenge in verification of programs is getting the right invariant.

Example 2 *Cube root program*

Q–EK What about verifying floating-point programs

A We can prove things about the states the program ends up in, but the numeric properties of these states are hard to reason about.

EK Maybe we should use intervals instead of floats.

AV That is what in done in practice.

5.2.3 Synthesis of Invariants

Even if it's not present in the program, we should introduce the loop counter n. Then, rather than talking about $\texttt{quo}_{old}$ and $\texttt{quo}_{new}$, we can talk about $\texttt{quo}[n]$ and $\texttt{quo}[n+1]$. Now we essentially have recurrence equations. What we need are closed form expressions for the solutions of the recurrence equations. which in our case are

$$\texttt{quo}[n] = \texttt{quo}[0] + n \text{ and } \texttt{rem}[n[= \texttt{rem}[0] = ny. \tag{5.1}$$

But, of course, these contain n.

Eliminating n (Gröbner bases on (5.1) and taking an elimination ideal) and replacing $\texttt{rem}[n]$ by rem etc., we get

$$\texttt{rem} = \texttt{rem}[0] - (\texttt{quo} - \texttt{quo}[0])y. \tag{5.2}$$

Note that any other consequence will lie in the ideal generated by this, since we have done Gröbner bases, hence this is essentially a completeness result.

Example 3 *A program with loops.*

```
x:=1; y:=0;
while ... do x:=2*x; y:=(1/2)*y+1; end while
```

Hence

$$x[n] = 2^n x[0]; y[n] = \frac{1}{2^n} y[0] - \frac{2}{2^n} + 2. \tag{5.3}$$

Gröbner bases don't directly apply here, but we can use Kauers' theory to deduce $xy - 2x = x_0 y_0 - 2x_0$, i.e. $xy - 2x + 2 - 0$.

Q–EK You ignore the tests in the body of the loop

A I'm taking all possible combinations of the `then` and `else` parts.

Bibliography

[BBDP07] J.C. Beaumont, R.J. Bradford, J.H. Davenport, and N. Phisanbut. Testing Elementary Function Identities Using CAD. *AAECC*, 18:513–543, 2007.

[Bro71] W.S. Brown. On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors. *J. ACM*, 18:478–504, 1971.

[Buc65] B. Buchberger. *Ein Algorithmus zum Auffinden des basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Math. Inst., 1965.

[Buc85] B. Buchberger. Symbolic Computation: Editorial. *J. Symbolic Computation*, 1:1–6, 1985.

[Col67] G.E. Collins. Subresultants and Reduced Polynomial Remainder Sequences. *J. ACM*, 14:128–142, 1967.

[Coo66] S.A. Cook. *On the minimum computation time of functions.* PhD thesis, Ph.D. dissertation, 1966.

[FÖ7] M. Fürer. Faster integer multiplication. In D. S. Johnson and U. Feige, editors, *Proceedings STOC 2007*, pages 57–66, 2007.

[Fre79] R. Freivalds. Fast probabilistic algorithms. In J. Becvár, editor, *Proceedings Mathematical Foundations of Computer Science 1979*, pages 57–69, 1979.

[GLL09] M. Giesbrecht, G. Labahn, and W. Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. *J. Symbolic Comp.*, 44:943–959, 2009.

[Her26] G. Hermann. Die Frage der Endlich vielen Schritte in der Theorie der Polynomideale. *Math. Ann.*, 95:736–788, 1926.

[Kal10] E. Kaltofen. The seven dwarfs of symbolic computation. `http://www.math.ncsu.edu/~kaltofen/bibliography/10/Ka10_7dwarfs.pdf`, 2010.

[KB70] D.E. Knuth and P. Bendix. Simple Word Problems in Universal Algebras. *In: Computational Problems in Abstract Algebra*, pages 263–297, 1970.

[KLYZ08] E. Kaltofen, B. Li, Z. Yang, and L. Zhi. Exact Certification of Global Optimality of Approximate Factorizations Via Rationalizing Sums-Of-Squares with Floating Point Scalars. In D.J.Jeffrey, editor, *Proceedings ISSAC 2008*, pages 155–164, 2008.

[KLYZ11] E. Kaltofen, B. Li, Z. Yang, and L. Zhi. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. *J. Symbolic Comp.*, ??, 2011.

[KNS11] E. Kaltofen, M. Nehring, and B.D. Saunders. Quadratic-Time Certificates in Linear Algebra. In *Proceedings ISSAC 2011*, pages 177–185, 2011.

[KO63] A. Karatsuba and J. Ofman. Multiplication of multidigit numbers on automata. *Sov. Phys. Dokl.*, 7:595–596, 1963.

[KSLT02] T. Kwok, J. Smith, S. Lozano, and D. Taniar. Parallel Fuzzy c-Means Clustering for Large Data Sets. In *Proceedings Euro-Par 2002*, pages 27–58, 2002.

[NW10] L.C.X. Ngô and F. Winkler. Rational general solutions of first order non-autonomous parametrizable odes. *J. Symbolic Comp.*, 45:1426–1441, 2010.

[PS73] M.S. Paterson and L.J. Stockmeyer. On the Number of Nonscalar Multiplications Necessary to Evaluate a Polynomial. *SIAM J. Comp.*, 2:60–68, 1973.

[SS71] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:282–292, 1971.

[Too63] A.L. Toom. The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers. *Soviet Mathematics-Doklady*, 4:714–716, 1963.