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

22–25 September 2014

Contents

Chapter 1

22 September 2014

1.1 Tools for Online Technical Collaboration: Watt

One issue in Computer Algebra is that of getting the mathematics *into* the computer, hence my past interest in handwriting recognition. Software-based collaboration is not a new idea. Slide "I think you should be more explicit here" showing collaboration in front of a blackboard. Note the shared blackboard, which is really missing from most of the collaborative software solutions, of which there are many (shows Wikipedia page "collaborative software").

Note that sharing images of a shared whiteboard is *not* the same as sharing the mathematics: Magritte "ceci n'est pas un pipe". For example Einstein's blackboard from his 1931 Rhodes House lecture isn't live mathematics.

Most current research in handwriting is *off-line* recognition (cheques, addresses etc.), but our interest is *on-line* recognition, hence "digital ink". Hence InkML [Wor11].

Discrete dots are not the right solution: instead use orthogonal series. To compare two curves, people in recognition normally use "elastic matching" as an artefact of discretisation: in orthogonal series we can use compute the area (actually area squared to allow for signs). Then we get the Euclidean distance between the coefficient vectors as the answer, which can be done in real time in JavaScript (!). Small distance is not always the right answer: compares "c" with "ϵ". The solution is to work in jet space, and insist the derivatives (first derivatives will do) be close as well. In practice linear separability works on his training set.

Symbolic clustering based on 20,000 arXiv articles. Note also that "baseline recognition" is a far more complex problem in mathematics (super/sub scripts). Compute the 'average" shape of a symbol from a training set.

In 2002 on "Pocket PC" he had expression recognition by inflating balloons round characters. Then using Java had InkChat. But this was hard to install, had limited portability, incompatible software bases, and moving APIs.

JavaScript is *not* the right language for large software applications, but

it's ubiquitous, and all the APIs are there. Hence current generation runs in browsers/telephones/tablets, with auto-sizing menus, the ability to annotate and save documents[1]. Has a Google Hangout embedding.

Gives a line demo of `InkChat.org`, room `synasc`.

Q Classroom collaboration?

A Certainly. But more and more our students will not be in the same physical room as us — see Open University.

1.2 Using the distribution of cells by dimension in a cylindrical algebraic decomposition: Davenport

See `http://staff.bath.ac.uk/masjhd/Slides/SYNASC-2014slides.pdf`.

1.3 Branch differences for Lambert W: Jeffrey

Many people have worked on branch cuts and differences for elementary functions [BCD+02]. These have simple relations (which people have therefore tended to ignore!).

Lambert W was unnamed for much of its history. Now that we've named it, it appears everywhere, and CA software has shown people this. Square root singularity at $-1/e$. The branches tend, asymptotically as $\Re(z) \to \infty$, to the branch cuts of log. However, the principal branch doesn't exist as $\Re(z) \to -\infty$, and branch 1 is adjacent to branch -1.

People have also been using the differences between branches of W without knowing it.

The entire part of the principal difference lies in the ++ quadrant of $\mathbf{C}$.

Q Is this phenomenon of the principal branch "disappearing" new?

A Yes!

Q Numerical evaluation near branch cuts?

A Hard. The branch cuts are at π, so precision of computation can affect what's happening

1.4 On corank 2 edge-bipartite graphs and simply extended Euclidean diagrams: Zając

Bigraphs have two kinds of edges: solid and dotted.

[1] Anyone ever marked up a thesis for a student?

Want connected loop-free bigraphs. These are useful in representation theorem, and in $q_\Delta(x) = \sum x_i^2$+dotted edges − solid edges. The Gram matrix $\check{G}_\Delta$ of a bigraph is upper-triangular. Let $G_\Delta = \frac{1}{2}(\check{G}_\Delta - \check{G}_\Delta^T)$.

We want to produce the finite set of all connected non-negative edge-bipartite graphs of rank n. Grows rapidly, $n = 6$ has 333,400 of them! There's a connection to simply-laced Dynkin diagrams. The simply-laced Euclidean diagrams are one-point extensions of these. These don't map directly, but we need a further extension.

There are forms of **Z**-equivalence, and we need to make use of these for a better algorithm.

1.5 Solving Parametric Sparse Linear Systems by Local Blocking II: Sasaki

Note that version I was presented as CASC 2014. "The algorithm is too *ad hoc*". Very common in industry (electrical networks).

Want a "block triangular" system. The determinants of these blocks are the degeneracy conditions. A block is a strongly connected maximal subgraph (SCC), whereas we want local blocks, (SCsubG) which aren't necessarily maximal.

Theorem 1 *Parametric system $Lx' = xc'$ is F-solvable if the graph G_L associated with L is strongly connected.*

We want to use the graph structure to find the "bets" subgraphs. However, this didn't work out: no characterisation of good systems, and inclusion checking was hard. In CASC 14 [SIK14] we tried to combine local blocks, while now we want user support. Note that industrial systems have a natural block structure, with weakly-connected blocks, so we should take advantage of this. Given this user input, then finding G_{char} is easy, as we can ignore back-edges, so the complexity is linear.

Q–JHD Do you need true user input, or rather from their tools as used to build the systems?

A Could be tools.

Q–SMW How closely do these come from the automotive industry.

A I have a student working there.

1.6 Point compression and coordinate recovery for Edwards curves over Finite Fields: Justus

This is about cryptography and symbolic computation. ECM is a method using elliptic curves to factor integers. Weierstraß and Edwards forms (he seemed to be saying that Edwards were genus 2).

ECDPL: gives P, Q find k such that $k \cdot P = Q$. Note no polynomial-time algorithm known. Note Certicom ECC challenge, with p being 109 bits being solved, and 131 believed to be insoluble. "For binary curves the record is 131 bits" (JHD wasn't clear if he meant curves over $GF(2^{131})$). Also mentioned Montgomery curves $By^2 = x^3 + Ax^2 + x$. Note that there is differential addition (using only x coordinates) as well as regular addition. implications for bandwidth and storage requirements. Hence represent (x, y) by $(x \pmod 2, y)$. Can decompress via $\sqrt{}$.

Theorem 2

$$dxyx_n y_n \equiv -1 \pmod{(dBy_n^2 + A)/D}.$$

Q–FW Queried 'genus 2'.

A There is such an addition law here.

Q–SMW You chose $c = 1$ — does it work generally?

A Yes: ease of presentation.

1.7 Hierarchical reasoning on local theory extensions and applications: Sofronie-Stokkermans

Want to use computers as 'intelligent assistants', but there are problems of description.

Theories from analysis "The sum of two Lipschitz functions" becomes

$$\mathbf{R} \bigcup L^f_{c_1} \bigcup L^g_{c_2} \models L^{f+g}_{c_1+c_2}.$$

algebras

Verification and embedded software.

But FOL is undecidable. We need to identify (sub-)theories which are decidable (and with low complexity).

Their AVOCS project solves ECTS case study (2009). We were surprised how many data structures are "comparatively easy".

Q You've narrowed some logic problems to be soluble. How about programming.halting problem.

A Good point. We generally don't reason about termination, for this reason. There is related work, but not time to explain it.

1.8 Implementing Reasoning Modules in Implicit Induction Theorem Provers

Ax is a set of (conditional) equational axioms. Equational Inductive Theories $Ax \models_{ind} C$: any ground instance of C is valid.

For reasoning over $|N$ we have induction, Presburger arithmetic [AotoStratulat2014], or a combination.

We want to separate logic from computation [Str01]. Reasoning modules show how to built Φ from ϕ (JHD didn't follow the notation here). The SKIPE inference system `https://code.google.com/p/spike-prover/` has

- conditional specifications
- instances of abstract systems
- automatic proofs
- non-trivial applications using equality and arithmetic reasoning [Barthe-Stratulat2003].

1.8.1 Decision procedure for T_{la}

`linearize`: only one conjunction of equalities is preserved by avoiding disjunction operators. `arith` uses Fourier–Motzkin over **Q** [LM92]. This is incomplete, since consistency over **Q** does not imply consistency over **N**. There is an A2L routine that moves deduced equations into the arithmetic component (I think).

1.8.2 Z3 [DMB08b]

Uses the Model Based Theory Combining approach [DMB08a]. SPIKE in BM took under 10 seconds on a benchmark, whereas Z3 took over 3 minutes. Conclusion: SPIKE interacting the two solvers is more effective: BM is faster but incomplete, Z3 is slower but complete. We would like to try others, since as CVC3, Yices.

1.9 Automated Synthesis of Target-Dependent Programs for Polynomial Evaluation in Fixed-Point Arithmetic

Context: French ANR DEFIS project. Want optimised code with accuracy certificates (hence proof, not simulation).

Polynomial evaluation is key to approximations of trigonometric functions. Note that degree 5 univariates have 2334244 different schemes. Horner is one options, but it's very sequential (not good on modern hardware, even embedded systems).

The goal is univariate and bivariate polynomial approximations. Unsigned fixed point only. Targeting modern processors (he cited one ST231, with a combined "shift and add" instruction, which apparently is responsible for 8.7% reduction in cycle count).

1. Computation step: generate many DAGs from a scheme
2. Filtering scheme (for speed and accuracy)
3. Gathering scheme: generated C code and Gappa certificates.

Previous restriction: unsigned only, and didn't support shifts (a major problem for fixed point!). The description of the processor didn't allow for modern (multiple ALU) features.

Note that instruction selection is NP-complete on DAGs. FMA, or shift/add, are the sort of problems we have. [VoronenkoPüschel] have a proof of optimality, but the FMA is hard-wired into algorithm. We accept an XML-file describing the instruction set: incorporating both C and Gappa descriptions. We use the first step of the NOLTIS procedure for this. [KoesGoldstein,CGO2008].

Get a 1-bit increase in accuracy (31-bit FP). For degree-7 polynomial approximating cos on [0,2], we reduce Horner from 41 to 34 cycles, Estrin from 16 to 14, and best scheme from 15 to 13.

Intend to extend to floating-point, where FMA is pervasive.

1.10 Random Test Generation

Testing is hard. Have to meet coverage criteria (all-uses intra-class def-use pairs is his example). Needs automation. OO-testing also needs to satisfy the object constraints to produce meaningful tests. Example tool: Randoop [PLB08]. However, this does not record coverage information direct test generation. So we capture information information during the Randoop test execution. If pairs are uncovered, schedule additional tests (typically 40% more).

Q How many samples.

A 3, but note that Randoop uses a fixed seed for its PRNG, which we didn't change.

1.11 Proof generation from δ-decisions: Gao

dReal is an SMT solver for nonlinear theories over the real. Returns either UNSAT or δ-SAT, where δ is user-specified. [GKC13]. Apparently $\sin(-2.437592) > 10^{53}$ in `eglibc` shipped with Ubuntu 12.10. This caused many false answers for a while. 'sat' generally has a certificate: what about proofs of unsatisfiability? Note this is key, since $\forall x P(x)$ is proved by UNSAT of $\exists x \neg P(x)$.

Works via Interval Constraint Propagation. Contract big initial interval boxes to small ones that cover solutions. Branch and Prune loop. Return sat when intervals are small enough. Have interval axioms (splitting) and constraints axioms (

$$x \in I \Rightarrow f(x) \in f(I) \tag{1.1}$$

). Turn each solving step into a transition step, and the transition step into a subtree. Note that backtracking requires a proof tree as well. The hard part is the constraint axioms (1.1). Interval extension, Taylor proofs and branch0and-bound loops. The more we understand a numerical algorithm, the easier we can verify these axioms. In the end we need to formalise the numerical algorithms completely.

Main set of benchmarks in the Flyspeck project [Hal12]. This has hundreds of lemmas, such as

$$\forall \mathbf{x} \in [4.0, 6.3504]^3 \left(2 \arctan(\frac{\cdots}{\cdots}) - \ldots\right) < 0.$$

We can get 2GB proofs for these.

Q Difference with MetiTarski?

A He converts these into polynomial constraints. We are also using δ-satisfiability. MetiTarski relies on the correctness of the CAD procedure.

Q 2GB proof in what representation?

A currently a text file.

1.12 Reducing Formal Equivalence to Formal correctness

Example: imperative and recursive $\sum_{i=1}^{n} i$. Partial equivalence would be equivalence if both terminate. Let A_L and A_R be the two languages for the left and right programs.

$$A_L \otimes A_R \vdash \ldots$$

A programming language is syntax and semantics (precise definition given)

To reason about configurations we have matching Logic$(\gamma, \rho) \vDash \pi$ where π is a basic pattern, of $\rho(\pi) = \gamma$, extended across logic in the usual way. This gives reachability Logic. $\phi \Rightarrow \phi'$ means any terminating γ that matches ϕ will move into γ' matching ϕ'. These reachability rules are essentially the semantics.

The claim is that equivalence is reachability in the aggregated language $A_L \otimes A_R$. The syntax of the aggregated language is done via pushouts etc. This paper explains how to aggregate the semantics We add

$$(h'_L(\phi), x) \Rightarrow \ldots (h_L(\phi'), ??)$$

for each rule $\phi \Rightarrow \phi'$ in A_L (and similarly for A_R.

Are working on total equivalence, but this is a hard problem.

1.13 A Heuristic-based Approach for reducing the power consumption of real-time system

So we have a set of processors and a set of tasks. tasks is $T = (c, d)$ where c is computation cost (worst-case) and d is deadline. Scheduling could be pre-emptive or not, and task occurrences could be single, periodic or sporadic. We consider single non-pre-emptive.

Effective switch capacitance is not constant, and power is not linear with speed. Then can state the problem as a multivariate optimisation. There was no analytic solution. Hence looking at a genetic algorithm, representing a problem by y_i which is the rate of operating of task i. Also differential evolution.

Genetic algorithm seems to show slightly better improvements.

Q–JHD How does this differ from [YDS95]

A Speaker didn't really understand: to be discussed off-line.

1.14 Catamorphism Generation and Fusion using Coq

Coq is a functional language with dependent types. We can write programs in Coq, prove them correct, then extract to OCaml/Hasjkell/Scheme. here is an OCaml API for Coq; tactics etc., which cannot threaten the language. We use this to build a tool based on Bird–Meertens formalism.

For any inductive type T, there is a polymorphic higher-order function $T \to A$, the catamorphism of T. Showed how to construct these for various type constructors.

```
Definition plus (n:nat) := cata_nat (Succ )
```

Similarly `mult`, other recursive operations. Note that there are also paramorphisms (which use the value), which allows to compute using the value (e.g. $n!$) and anamorphisms, which produce an inductive data structure. There is a fusion theorem for catamorphisms: the fused version is generally more efficient. The problem is that there a `cata_t_fusion` for each data type, with its own proof, while abstract Category Theorem has one fusion theorem. The `SyDRec` pug-in will generate the fusion theorem.

Showed how to use the catamorphism/fusion framework to generate an efficient solution for “max segment sum”.

Chapter 2

23 September 2014

2.1 Can tools from symbolic computation help analysing iterative solvers: Takacs

Certainly — quantifier elimination: see [HLS97] and [PT11, PT14]. Mentioned other applications.

Define QE, CAD.

Example 1

$$f(\tau) = \sup_{-1 \leq c \leq 1} 1 - \tau(1+c) \tag{2.1}$$

Rewrite as

$$\forall_{-1 \leq c \leq 1} c 1 - \tau(1+c) \leq \lambda$$

and eliminate c.

Example 2 $-\Delta u = f$ *in* Ω*, and* $u = 0$ *on* $\partial\Omega$*, then go to a variational formulation. Lebesgue norm* $\int_\Omega u^2(x)dx$*, Sobolev norm ....*

Discretise and finite element method. Find u_h such that $K_h u_h = f_h$. This is "only" a linear system, but very large.

K_h is symmetric and positive definite, but very large and condition number C is $O(h^{-2})$. Jacobi iteration has #iterations $O(C)$, but conjugate gradient is $O(\sqrt{C})$, however nonlinear.

$$u_h^{(k+1)} = h_h^{(k)} + \hat{A}_h^{-1}(f_h - K_h u_h^{(k)})$$

If u_h^* is the true solution, the errors decay by $||I - \hat{H}_h^{-1} K_h||_{K_h}$.

There are two ways to analyse the convergence: classical and local "Fourier Analysis": compute the convergence rate for a special case, and "extrapolate". The latter should have realistic values rather than upper bounds.

Assume $\Omega = \mathbf{R}^d$. For $K_h = tridiag(2; -1)$, we can explicitly multiply by a Fourier vector and calculate the convergence. $q(t) = \sup_{\theta\in[0,2\pi]} (1 - \tau(1 - \cos\theta))$. Involves trigs, but express via $c = \cos\theta$ and reduce to (2.1).

Have high-frequency and low-frequency components, hence multigrid methods.

1. Apply pre-smoothing (Jacobi)
2. Apply coarse-grid correction
   (a) Compute defect and restrict to coarser grid
   (b) solve of coarser
   (c) prolongate and add result

   If realised exactly to a two-grid method
3. Apply post-smoothing(Jacobi)

With this hierarchy of grids, we have to solve.

$$\sup_{\theta\in[0,2\pi)} \left|(\tau - 1)^2 + \tau(3\tau - 2)\cos^2\theta\right|$$

Again this could be solved by hand, but we have taken a very special case.

For a multigrid method for a system of PDEs, we have a question of choosing the smoother.. The convergence is now given by a symbol which is a 4×4 matrix. We then have[1] a $3 \times$ sup construct: messy by hand but the same process.

Moving to 2D gives 8×8, and this has problems with CAD.

Which has better constant: Courant element (piecewise linear) or splines.

Similar, but also there's a telescoping argument to handle the multigrid convergence issue.

Courant $\sqrt{1/3}$ and splines $\sqrt{2/5}$.

Q Which CAD?

A Mathematica. Note that this doesn't give certificates.

Q What about rounding errors?

A A problem with Gaussian elimination, but iterative methods are self-correcting.

Q–FW Note that there are comparatively few variables.

2.1.1 JHD subsequent notes

1. Not only does every DE generate a new CAD problem, but every solver does as well.
2. Although they always involve trigs, they are never mixed (never t and $\cos t$) and hence can always be algebraicised.
3. Mathematica code via the JSC paper.

[1] JHD observes that there is no alternation in this construct.

2.2 Lipschitz bounds for noise robustness

ℓ^1 minimisation is most widely used, but the computational time can be high. Through the construction so specific vectors known as the dual certificate, we can guarantee success of ℓ^1 minimisation without actually doing it.

Given image $x^0 \in \mathbf{R}^n$, and $A \in \mathbf{R}^{m \times n}$ a linear operator and a noise vector $w \in \mathbf{R}^m$. The data vector is $y = Ax^0 + w$.

Definition 1 *The basis pursuit is* $\min \ldots$

Noise robustness is expressed in terms of Lipschitz bound C relating the ℓ^2 recovery error to the ℓ^2 norm of the noise. The support l of x^0 is the indices of the nonzero elements. x_l is the restriction of x to this. Similarly A_l.

Definition 2 (Identifiability) *A signal x^0 is identifiable by ℓ^1-minimisation of*

- *(noiseless) x^0 is the unique minimiser of (2)*
- x^δ
- *or ...*

Lemma 1 (Source condition) *x^0 is identifiable by BP is $\exists \eta \in \mathbf{R}^m$ such that ...*

Then η is called a dual certificate.

We have a cross-polytope $P = AB_1 = Conv(\{\pm a_i : i = 1 \ldots n\})$.

Lemma 2 (Donoho) *x_0 is identifiable iff ...*

We look for a dual certificate by a sequence of transformations. Greedy algorithm appears to outperform projection by $\times 3$ in terms of time.

2.3 Fast methods for locating critical values of ζ

A new algorithm RSpeak isolated peak values on the critical line. Riemann-Siegel formula computes in $O(t^{1/2})$. In 2011 methods (Ghaith Ayesh Hiary) were presented $O(t^{2/5})$, $O(t1/3)$ and $O(t^{4/13})$. In the main sum of R–S, the initial segment dominates. [Kotnik2004] observes a good way of isolating peaks There is an L^3 approach to find numbers with appropriate approximations.

We have a small-n replacements, which is 10,000 times faster. Used a 2.9GHz laptop 16GB Ivybridge. Verified with ATLAS using $O(t^{1/3})$ at Előtvős Loránd University: 44 dual-CPU Ivybridge nodes. 10775 candidates found on laptop in two hours with 95.6% success rate for $Z(t) > 500$. See `www.riemann-sieegel.com`. Record is $Z(t) = -2624.76$.

Let $\phi(t) = \frac{\log|Z(t)|}{\log t}$. From Huxley, $\zeta(1/2 + it) = O(t^{32/205+\epsilon})$, so ϕ is interesting.

2.4 A lowest-level rule push-relabelling for submodular flows and matroid optimisation

For matroid optimization, our strategy needs no lexicographical order on the elements.

2.5 Very large sparse matrices

XY or CSR are slow and space-inefficient. We proposed quadtree-base formats. But how to convert? Assume $1 \ll N \ll n^2$.

We use a minimal QT format $4 \cdot \left(\frac{N}{3} + \cdots\right)$ space[2]. The usual conversion algorithm has a bottleneck in `INES`, so propose `INES2`. Can also run in parallel with OpenMP barrier calls. Time complexity $O(N \log n + n)$ for the biggest step.

We tested on 11 matrices from Florida Sparse Matrix Collection. Sequentially, factor of nearly 300 for `circuit5M` and faster for almost all examples. With 8–12 threads, often see a speed up of $\times 6$, but not quite as good for the new algorithm as first step is not parallelisable. Also the sort phase is not balanced: some lists are long and others short.

Q Why not use work-stealing rather than straight barrier?

A Possibly, but the first phase is the main issue.

Q What about oct-trees?

A We haven't gone here (don't really see the application)

Q On a real application, is the whole process faster?

A See previous papers, which show improvements for quad-tree methods.

2.6

Consider a nonlinear DE

$$F^{'''}(\eta) + (n-1)F(\eta)F^{''}(\eta) - e\eta(F'(\eta))^2 = 0$$

where n is a known parameter. The investigation of boundary layer effects was Sakiadis. There are analytic solutions (? special cases) in terms of incomplete Gamma functions.

Produce a homotopy ...

[2] Very compressed, but not directly suitable for matrix-matrix operations

2.7 Analysis of paper fold methods by geometric algebra: Ida

Recently became interested in 3D origami: applications in robotics and 3D printing techniques. Hence trying to extend my Eos system to 3D. Shows "piano origami", demonstrating that the 3D-ness normally comes at only the last stage of folding. Also "flapping crane".

The current Eos has restricted 3D modelling for both verification and visualisation, and no theory for motion as in the flapping crane.

The Geometric Algebra (GA) is a Clifford Algebra of a vector space over $\mathbf{R}$ endowed with a quadratic norm. Originally due to Grassman. [Hestenes1986,Doran2003]. It's a unital ring.

GA in Isabelle/HOL. Called g3. Inhabited by elements of type mvec: scalar, vector etc. Inner/Outer product are the operations that should connect to geometric intuition. Reflection is a very important operation in origami, but reflection in arbitrary lines is poor in Cartesian coordinates.

[Huzita1989] shows that the fold operation (in six flavours) is basic for 2D Origami.

Q Quadratic forms?

A This is the link back from vectors to $\mathbf{R}$.

2.8 Accelerators, GPUs and other hardware: Adrian Jackson (EPCC)

Simulation is now the fourth pillar of science (Observation, Theory, Experimentation being the others). His own work is in fusion, where it's hard to observe. Usual comments about "need more processors, not faster". Graph of $\log R_{\max}$ against time 1940–2010 looks linear. Claims that Tianhe-2 is being used to run game webservers because "the way it's put together is not useful". Chip images, BlueGene-Q (16 1.1GHz processors, capable of 4 threads) and GPU.

2.8.1 Issues

- processor affinity (where are caches/memory shared?)
- Initialisation optimisation
- usual optimisation tricks
- Minimising communication

Note that a lot of what is on a usual processor, branch prediction, out-of-order execution etc. is not that necessary for HPC (JHD might doubt this) hence GPUs have less of this. Graph showing performance of GPUs, but the headline

figure is SP (DP figures also given). Note that scientifically-oriented GPUs have, as well as DP, ECC, which is not really necessary for graphics, but errors in HPC persist/infect other cells. Note also that a GPU can't run an O/S: we need to connect the GPU to a CPU.

Picture of AMD Barcelona, 10% of area is FP, and maybe 20% is cache. NVidia Fermi is over 40% FP.

CPUs use DRAM, while GPUs use graphics memory, which is faster (but more expensive and more energy-consumptive).

Compares Fermi with Magny Cours: DP 515/101 Gflop, memory bandwidth 144/27.5 GB/sec. Hence a rough 5× factor.

2.8.2 Xeon Phi

7–8 years ago Intel were worried about GPUs. Initial answer was Larrabee (never worked). Around 60 physical cores (240 threads). Typically 1GHz. Based on an *old, simpler) Pentium core plus 512-bit AVX SIMD capability. Therefore you can take your standard program, recompile and it will run (poorly, but . . .). Tianhe-2 has 2 CPUs and 3 Phis/node.

2.8.3 Using GPUs

98% of our programs are C or Fortran (50/50). Hence we need to run, or interoperate, with these. Neither of these really support GPUs, and all the complex multi-threading, memory management. We've been used to MPI. CUDA (nVidia proprietary) allows GPUs to run C/C++. The key concept is a 'kernel' — a piece of code to run on a GPU. Note that memory management to make data available to the kernels is explicit (and slow). PGI[3] provide a commercial Fortran version of CUDA. There's actually a two-level hierarchy. 1 GPU = several SMs (streaming multiprocessors) each of which has multiple cores. CUDA abstracts this as a grid of thread blocks.

CUDA works by offloading functions, so the *body* of the lop has to become a function. Use the `__global__` specifier to indicate that it's a kernel. Then

```
vectorAdd <<<blocksPerGrid, threadsPerBlocks>>>(arguments)
```

In practice we would want to use multiple blocks. `blockIdx.x` is unique to every block, as opposed to `threadIdx.x`. `blockDim.x` is the number of threads/block. Can also have 2D and 3D allocations of threads and blocks.

There's also OpenCL, an open standard for Homogeneous Parallel Processing. Uses keyword `global` and functions for `get_global_id` etc. Kernel launching is more complicated.

OpenACC attempts to do this via directives: `#pragma acc` in C and `!$acc` in Fortran. PGI and Cray tend to support these; GCC is "real soon now". The `parallel` directive says "run on GPU" (i.e. make kernels out of it), but you will need `loop` (similar to OpenMP `loop`) to actually get parallelism. By default,

[3]Now owned by nVidia

`loop` will use its knowledge of the target to decide how many blocks/threads to use.

By default, anything used inside a parallel region is copied to/from the GPU — easy programming, but not necessarily efficient. The `data` directive, with `copy`, `copyin`, `copyout`, `create`, `private` etc. can control this.

Instead of `parallel`, there's `kernel` (note that OpenACC was a merger of Cray's and PGI's efforts) which is similar but different.

Note that "deep copy" is generally not supported in any model: has problems for CASTEP.

2.8.4 Using Phis

Three modes of use.

Offload using kernels or MKL routines to get specific parts run on Phi.

```
#pragma offload target(mic:0) in (A:length(N)
```

There's also a "virtual shared memory" model, where the user declares memory to be "shared" (actually copied to/from) between the two.

Symmetric (running on both using MPI)

Native Recompile code to run on Phi (only)

Various views.

> The performance improvement to be gained by parallelising is limited by the proportion of the code which is sequential. [Amdahl]

Note that solving bigger problems tends to decrease the sequential size of the code, weak scaling.

> The performance improvement to be gained by parallelising is limited by the proportion of the code which is sequential, and the cost of that parallelism. [Adrian]

Heterogeneity impacts load balancing (and we really need to use the processors and the accelerators). We've also got heterogeneity of communication: PCIE bus or Infiniband.

VASP is 17% of EPCC,and Gromacs is second. 70% in total is "big packages". CASTEP simulating 64 atoms on 64 cores has 13.5% of execution time in MPI alltoall, and doesn't really move to GPUs with performance. Currently 1 (entire) GPU does about the same performance as 8 MPI processes (i.e. at most one chip).

Alternatively, have been trying a preprocessor approach, which is showing good speedup on both IvyBridge and GPUs — the speedup on IvyBridge was due to the fact the loops had been split up differently — tiling/compiler not doing well.

2.8.5 Future

Knight's Landing. This *may* finally remove the memory bottleneck.

Chapter 3

24 September 2014

3.1 Genetic Improvement of programs: Langdon

When should one automatically improve software:

- small glue between systems, e.g. mashups. "Grow and Graft GP" — small additions to big systems.
- GP as a tool, especially when there are conflicting objectives, where the GP can generate the alternatives, leaving a human to choose.

Start with a population of randomly-created programs, define a fitness function by running them, better programs are selected to be parents, and offspring are produced by merging/mutating, and repeat.

In NASA, there was an urgent need for an antenna: the humans took two months but the GP took two weeks, exceeded all the specifications, and NASA actually flew the GP one.

In general, the GP system can produce a solution specific to the task, and therefore can outperform the generic solution. This has been demonstrated on *small* systems. [White2011] was able to generate low-power code.

Example [LH14]: Bowtie2: 50,000 lines of C++, used in BioInformatics. This is the "test oracle", uses the (slow) Smith–Waterman algorithm. The aim was to tailor the system to specific tasks. Run the GP on parsed programs, and actually evolve patches, rather than programs. Better half are chosen, each gets two children, one by crossover and one by mutation: these are (locally) syntactically correct, but may fail to compile (variable not in scope, typically). Grammar of this program (5792 statements) apparently had 2252 `if` but 24 `else`. basically, only 15% compile. Moving a line < 100 lines has 82% compile. Pragmatically, "only move within same source file" has 45% compiling, which is good enough. Even if programs compile, they may well fail the tests, often spectacularly.

There are some lines of code that are executed often, independently of size of output, some are linearly dependent on size, some quadratic and some cubic. Ignoring C++ preprocessor lines, and focusing on heavily used, cuts program from 50745 lines to 2744. GP finds a 39-line change spread across 6 files which improves performance (by a factor of 70: see below) on the examples we care about.

Population size was 10 (should be much larger, but a run took a day, so project lifespan and use of a single machine dictated this!). Over half the elapsed time goes in compilation, use `make` to compile patches only, precompile headers, `-O0`, only use five tests (but a different five each generation). Fitness is based on

- number which succeed
- Smith-Waterman score
- number of instrumented C++ lines of code executed (least important).

During evolution, 74% compile, 6% fail at runtime. 200 generations took 25 hours.

Then have a "clean-up" phase to reduce the 39 changes to 7: (did the change make a noticeable difference). Biggest change replaced a loop iteration from 200M to 84 times (`i<offsLenSampled` to `i<this->_nPat`. Two more of same ilk. These three are all independent[1]. There's then a group of four patches that must go together to do with C++/machine code/vector instructions interface.

Then compile with same flags as original, run on 200 DNA sequences (same DNA scanner but different people as samples), taking 4 hours rather than 12.2 days. 89% identical, 9% output better (higher mean S–W score, median improvement 0.1), 0.5% the same and 1.5% worse (in 4th and 6th decimal place). Enormous speedup and slight *improvement* in quality!

Conclusion — based on a *population* of changes, software isn't fragile — bend it, break it etc., and see what evolves.

Q–SMW How do you get "bad for a while then better' changes?

A We don't: that's evolution. There's a real risk since our population was very small (10).

JHD discussed afterwards. The "cleanup" removed all changes accounting for $< 1\%$, so the coupled group contributes at least that much — his feeling is that it was a major component.

3.2 Spiking Neural P-Systems: Păun

See `http://ppage.psystems.eu`.

[1] And, independently, are now in the GitHub source, 700 changes later. WL/JHD discussed this. These changes could have been discovered by traditional profiling. WL doesn't actually know how the changes made their way independently into the new version

Membrane computing started 17 years ago, whose goal was to abstract computing models/ideas from living cells. It's original goal did not include providing models for biologists, "though this is now a tendency". The idea is that DNA works *in vivo*, not *in vitro*, as in the earlier "DNA Computing".

No-one knows how the brain becomes the mind.

Note that the axon is not a wire, and there are cells around the neurons, which feed the neurons depending on traffic along the axon.

A computing extended spiking neural P system of degree $n \geq 1$ is a construct

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, sgn, in, out)$$

where

$O = \{a\}$ is the singleton alphabet (a is called *spike*)

σ_i are neurons, $\sigma_i = (n_i, R_i)$, where $n_i \geq 0$ is the number of spikes in σ_i; R_i is a finite set of runs of the form

1. $E/a^r \to a_o; d$ where E is a regular expression over a
2. $a^p \to \lambda$

Example 3 (SN system module) *If register r holds n, then the neuron σ_r holds $2n$ spikes.*

Therefore Turing-complete. Also an example that computed any Boolean function of three variables.

Theorem 3 *1. $NFIN = N_2SNP_1(rule_1, cons_1) = N_2SNP_2(rule_2, cons_2)$.*

2. $N_2SNP_n(rule_k, cons_k) = NRE$ for all $l \geq 2, p \geq 3$

3. $NSLIN = N_2SNP_k(rule_k, cons_k, bound_s)$ for all $k \geq 3, l \geq 3, s \geq 3$).

A key question is "when does the output neuron spike".

Various generalisations, e.g. $\overline{a}$ as an anti-spike with $a\overline{a} \to \lambda$ as a cancellation rule.

3.3 The Rise of Cognitive Computing: IBM Watson and Beyond: Manaila (IBM)

Characterisation of "Big Data".

Volume

Velocity

Variety

Veracity

80% of the world's data is unstructured (video etc.); 90% of the world's data was created in the last two years.

Claims that Watson *understands* natural language, *generates* hypotheses, *learns* from users. Claims that cognitive systems are the third era, after tabulating and programming.

Watson's decision advisor works when there is a large body of available data to draw from. Watson engagement advisor when one needs a stronger tie with constituents, better automated or agent-facilitated conversations.

Watson (Jeopardy) per se is 2880 cores (Power 7, 90 nodes) 16 TB RAM and 20TB disk. 80 Tflops. Moving from this to thousands of users is a challenge. For example, input might be text, tables and images.

Medical Journal concept annotations. There's a "DeepQA" architecture architecture (with a complicated, messy diagram!).

Problems with exporting the Watson technology to other applications and architectures (e.g. non-vN).

Vision "True North", 50mW 4096 core, 10^6 neuron, $2.56 \cdot 10^8$ synapses "chip" as a field-programmable neural network.

Chapter 4

24 September HPCRes workshop

4.1 Performance Improvements for LNAWENR Model: Tirsa

Started with a Fortran code, translated into C for OpenCL reasons. Asteroseismology: can we look at the interior of pulsating stars. Run the same piece of sky over and over, looking at changes. LNAWENR: Linear NonAbdiabatic NonRadial WavEs. Part of the ... suite. Not a great deal of data (200MB at most), but a lot of calculations.

Looking at pressure waves of mode i ($0 \leq i \leq 3$). Frequency $0.01 \ldots 60$, split into 100,000 mini-intervals. Adiabatic uses the first four equations and finding the roots roughly, then move to full set of equations and BCs to refine.

Want to keep source code readability high (even for the ex-Fortran guys).

Once one root is found, use this to refine where to look for the next. Good use of loop invariants (the equations were written naturally). Over 80% was spent in Gaussian elimination, so changes here got $\times 2$ in sequential performance. Only keeping the data from the most recent measurement made the program much more cache friendly.

4.1.1 OpenMP version

1. Outer loop (`i`) only
2. Inner loop only
3. both

Get 12-fold speedup with 4 outer loop threads and 6 inner.

4.1.2 OpenCL

Clearly only go for inner loop! Private data was too large, which needed an explicit linearisation of the multi-dimensional arrays. The GPUs (chip/chip compare) bought ×4.22

2×6-core Intel Nehalem, 2×228 core M2070 Tesla. Results gathered on inputs of 1000 measurements.

OPT0 base case in C, very similar to Fortran.

OPT1 `-O3`×2.84

OPT2 Take out unnecessary complex variables ×2+

OPT3 etc.

OPT4

OPT5

OPT0

Total gain from `-O0` to GPU/OpenCL/OpenMP was ×*over*200, just under ×100 starting at `-O3`.

4.2 NSGA-II implementation and Performance Metrics Extraction for CPU and GPU: Pădurianiu

We are using multi-objective genetic algorithms. The cost is the evaluation time, and hence we wish to parallelise. Pareto Dominance (all elements $\geq$) and Pareto Front. Concept of crowding distance of $u \in PF$: that space dominated by $u$but not any other elements of PF. Her GA merges parent and child populations, ranking these $2N$ individuals, and selecting the best N.

Use one thread per individual. No interaction when it comes to evaluation of objective function. Also execute the "nondominated sorting" step on GPUs Also genetic operators.

Test problems were ZDT1 and similar. Both CPU and GPU versions give good convergence (in terms of #generations) and diversity. Speed-up in range 3.8–7.3, typically ×4.52.

Q Did you look at individual kernels.

A Sorting gives the best speedup. Used the same code for the different fronts.

WL asked afterwards: apparently the speed-up is computed as one core/whole GPU.

4.3 A Parallel distributed HP Architecture for simulating particle-based models: Sabou

Realistic simulation requires complex models (large numbers of particles). Complicated by adding interactivity. So we are using the mass/spring model. Structural springs, shear springs and bend springs for the cloth model they are looking at.

Around the model, we have Numeric Integration, Collision detection/response; Distributed rendering and remote visualisation/interaction.Hence client/server model. Do we do explicit or implicit integration? Explicit is simpler code, but needs small time steps. Implicit is opposite, and the large time steps tend to lose fine features in time. We wish to evaluate in terms of "fastest visualisation". Gets $\times 1.5$ on two nodes CPU+GPU; uses both), and up to $\times 2.7$ on four, provided the problem sizes (#particles) are big enough. For big problem sizes, the matrices for implicit integration overflow the GPU memory.

Chapter 5

24 September HydroGIS Workshop

5.1

Hydrodynamic Modelling has requirements analysis etc giving mathematical models. These pus boundary and initial conditions go to numerical models, and hence results. The aim is to find the minimal flooding in polder operation. Showed an image of river, dams and fields, some grouped in two polders.

Based on real data from her workplace.

5.2 Implementing a Hi-Res Weather Prediction for Romania + surrounds:Oană

Part of CBIES (Cross-Border Romanian-Serbian Emergency response).. 1024×4core BlueGene. Developed in 2000 based on old MM5 model (open-source, Fortran). Non-hydrostatic, This ARW and its variants became popular from 2008. From 50.2N to 39.8, and 11 to 33 East, 28 vertical layers. Use two hours of actual data and low-res data from global models. Dewpoint errors showed an error of up to 3C. It took 12 hours for the MSL Pressure error to stabilise, with a messy saw-like behaviour before stabilisation. Hence needed to change the initialisation methodology. Have a 1-hour pre-forecast period, but changed this to 12 hours. Note that the distribution of weather stations available to us is very varied (low in Romania). Details for 24/5/2014 22UTC: storms arose not predicted by V1. V2 did predict precipitation, though not at the actual intensity.

5.3 Indexing of Geographic data in Distributed Environments

Objects w.r.t. position: lakes, streets etc. Generic raster,points, line, polylines, polygons, regions. Note might want "area flooded in 1998" and "area flooded in 2010", which will overlap. The index is the database structure to improve retrieval. Can be grid/quad tree/R[ectangular]-tree. This used to be mainframe, now consider smartphones. As an alternative to traditional DB, what about Hadoop. Claim that moving processes is faster than moving data. HDFS architecture diagram. But HDFS doesn't fit with traditional spatial indices. Note also HDFS is write-once read-many. There is a Spatial Hadoop project using R-tree. One master file and many data files.

Used 2-core 800MHz 1GB VMs ina private cloud. Various datasets from US Government supply. Largest is 2.6GB.

5.4

Chapter 6

25 September 2014

6.1 Symbolic Summation in Difference Rings — With Applications in Combinatorics, Numerics and Physics: Schneider

Slides at `http://www.risc.jku.at/home/cschneid` and [SB13].

6.1.1 Indefinite Summation

Consider

$$\sum_{k=1}^{n} S_1(k) \text{ where } S_1(k) = \sum \frac{1}{i}$$

Demonstrates in his package[1], and gets $-n + (1+n)S_1(n)$.

Telescoping is the summation equivalent of indefinite integration. To sum $F(k)$, we want a G such that $F(k) = G(k+1) + G(k)$. We want a G that is "as simple as' F.

Definition 3 *$F(k)$ is called an* (indefinite) nested product-sum expression w.r.t. *k is it can be build by*

- *k and a finite number of constants*
- *arithmetic operations*
- *indefinite nested sums and products $\sum_{i=1}^{k} f(i)$ or $\prod_{i=1}^{k} f(i)$ where $f(i)$ is free of k and a similar w.r.t. i.*

We use binomials etc. purely as shorthand in this grammar.

[1] `SigmaSum` is an inert summation, and he has an explicit simplify: `SigmaReduce`.

We want to use the language of difference fields. We want an automorphism $\sigma : F \to F$. Over $\mathbf{Q}$ we have no choice: $\sigma = id$. In $\mathbf{Q}(k)$ we have $\sigma(k) = k + 1$. Then $\sigma(S_1(k)) = S_1(k) + \frac{1}{k+1}$ in $\mathbf{Q}(k, S_1(k)) = \mathbf{Q}(k, s)$. Basic theory in [Kar81], extended in [Sch04]. As usual we need the "no new constants" rule.

When solving $\sigma(g) - g = s$, the key requirement is a denominator bound (in the last extension variable). Find d such that $\sigma(\frac{g'}{d}) - \frac{g'}{d} = s$. After this, we need a degree bound. Once we have this, we find the coefficients. The equations are triangular, solves by induction on the extension degree. Note that it is necessary (as JHD says when speaking at RISC) that we then have to unwind the algebraicisation and put back in the boundary conditions for the summation.

Can also solve $\sum_{k=1}^{n} S_1^2(k)$, but $\sum_{k=1}^{n} S_1^3(k)$ needs the harmonic numbers of the second kind. For $\sum_{k=1}^{n} S_1^4(k)$ he cannot avoid nested sums, but can produce one that is less nested.

One problem is building the "appropriate" $\Pi\Sigma$-field: see [Sch08]. he has an algorithm which, given a nested summation, will produce an extension of minimal depth. For

one would naïvely get no simplification, but he can express it entirely in terms of non-nested sums.

Want the output b, given input A, requite $A(k) = (k)$ (correctness), all sums in B to be simplified as above, and the arising sums must be algebraically independent (hence we get zero recognition (modulo constants, JHD assumes).

Consider the ring $\mathbf{Q}^{\mathbf{N}}$ of indefinite sequences. We say that two sequences are equivalent $\sim$ if they agree from a certain point on. This is an equivalence relation, let $[\ldots]$ denote equivalent class. Then we work in $\mathbf{Q}^{\mathbf{N}}/\sim$, and identify sequences with equivalence classes. A rational function has only finitely many poles, so $ev : \mathbf{Q}(n) \to \mathbf{Q}^{\mathbf{N}}/\sim$ is well-defined, and the images are the *rational sequences*. Then I can embed $\mathbf{Q}(n)[h_1, h_2, \ldots]$ with $h_k \mapsto \left\langle \sum \frac{1}{i^k} \right\rangle$, i.e. the (evaluations of) harmonic numbers. Note we only use h_i in numerators, as he has problems with the poles if they are allowed in denominators.

How does one guarantee that the constants are unchanged. Given (F, σ) with constant field $\text{const}_\sigma(F)$. Adding t with$\sigma(t) = t + f$ for some $f \in F$. Then [Kar81] $\text{const}_\sigma F(t) = \text{const}_\sigma(F)$ iff $\nexists g \in F : \sigma(g) = g + f$ (i.e., we can't solve the problem without extending).

Products are subtler. Consider $\prod^k i + \prod^k (-i)$. We can't solve the first in $\mathbf{Q}(k)$, so we need to add t_1 (which is $k!$) as a new variable. There is no solution to the second even in $\mathbf{Q}(k)(t_1)$ so we add t_2 (which is effectively $(-1)^k k!$. But

$$\sigma\left(\frac{t_1^2}{t_2^2}\right) = \frac{t_1^2}{t_2^2},$$

and we have new constants.[2]

[2] On further discussion, this constant $(-1)^n$ is essentially the only constant we get with one level of summation over $\mathbf{Q}$. A larger number field can introduce other roots of unity. Also, a

Can solve this by adding x where $x = (-1)^k$. However $(x+1)(x-1) = 0$, so we now have a ring with zero divisors — ouch!

Nevertheless we can extend [Kar81] to this setting, providing summations are only in the numerator.

6.1.2 Definite Summation

Example from Pemantle: analysing simplex algorithm. These had sums to infinity: dangerous. Replace by a. However, these indefinite sums don't simplify. There is no $g(n,k)$ such that $g(n.k+1) - g(n.k) = f(n,k)$. Try to solve

$$g(n.k+1) - g(n.k) = c_0 f(,n,k) + c_1 f(n+1,k) + c_s f(n+2,k)$$

where the c_0 are undetermined constants (i.e. free of k). This gives a recurrence relation. *Now* we can let $a := \infty$ and solve the recurrence equation. It's a second order so we find two solutions which span the space. This expressed Permantle's inner sum (we get some ζ_2 values.

General methodology in [Sch07]. Note that it is not guaranteed to find a recurrence relation if sums exist in denominator, but Zeilberger's theory will show that they always exist. We can find these if the operator factorises.

Application in particle physics: where the degree is 61, and coefficients have 100s of digits. The actual solution space fits into a couple of screens, though *after simplification*. Note that the base objects are indefinite summations.

6.1.3 Particle Physics

This is a long project on evaluations of Feynman diagrams with Blümlein. he transforms Feynman integrals to Feynman sums, which CS can solve, using this machinery recursively.

Shows a triple sum. He (using 1990s technology) can solve the innermost sum. This then decoupled, and the expressions reduces to a single sum. In practice, he has 6-fold sums, which he can reduce by using a human-selected combination of his tools.

6.2 Cloud

Formal verification of cloud-based systems has the "state space explosion" problem. On the other hand., "Big Data" manipulates very large data sets. We therefore want a framework, independent of the precise verification formalism, using `MapReduce`. We use a previously-developed state-space builder. Also a distributed model checker of CTL formulas. CTL (Computational Tree Logic) models the evolution of a system as a tree.

---

double summation can introduce $(-1)^{n(n-1)/2}$, and so on, but this is as bad as it gets. CS believes, work in progress, that this is also true for q-hypergeometric, with the extra constants being $-1)^{n/2}$ etc.

MaRDiGraS (ouch!) is his formalism. This gives a partition of the state space, such that each state s stores all incoming transitions arranged by processor they are coming from, hence we have a local tree.

Shows a set of MapReduce algorithms for the evaluations. Run on three models, 10^6–10^8 states). Claims (though JHD couldn't read the graphs) that he gets good scaling for the large models. Apparently used 16 Amazon instances. At some points he has superlinear speedup, which he suspects is due to cache growth.

6.3 Views and Updates on Distributed Databases: Ravve

Assume that the computation is sent (by the central site) to where the data are. Feferman–Vaught reductions are used when a relational structure is pieced together from various components. let $\mathbf{R}$ and $\mathbf{S}$ be a database scheme, with instances $I(\mathbf{R})$ etc.

Definition 4 Φ *is* k-feasible *for* $\mathbf{S}$ *if* ϕ *has exactly* k *distinct free FOL variables and each* ϕ_i *has ...*

6.4

6.5

6.6

Q

A

6.7

-
-

Bibliography

[BCD+02] R.J. Bradford, R.M. Corless, J.H. Davenport, D.J. Jeffrey, and S.M. Watt. Reasoning about the Elementary Functions of Complex Analysis. *Annals of Mathematics and Artificial Intelligence*, 36:303–318, 2002.

[DMB08a] L. De Moura and N. Bjørner. Model-based theory combination. *Electronic Notes in Theoretical Computer Science*, 198:37–49, 2008.

[DMB08b] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, 2008.

[GKC13] S. Gao, S. Kong, and E.M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. *Automated Deduction — CADE-24*, pages 208–214, 2013.

[Hal12] T.C. Hales. Dense sphere packings: a blueprint for formal proofs. *Cambridge University Press*, 2012.

[HLS97] H. Hong, R. Liska, and S. Steinberg. Testing Stability by Quantifier Elimination. *J. Symbolic Comp.*, 24:161–187, 1997.

[Kar81] M. Karr. Summation in Finite Terms. *J. ACM*, 28:305–350, 1981.

[LH14] W. Langdon and M. Harman. Optimising existing software with genetic programming. *To appear in IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2014.

[LM92] J.-L. Lassez and M.J. Maher. On Fourier's Algorithm for Linear Arithmetic Constraints. *J. Automated Reasoning*, 9:373–379, 1992.

[PLB08] C. Pacheco, S.K. Lahiri, and T. Ball. Finding errors in net with feedback-directed random testing. In *Proceedings of the 2008 international symposium on Software testing and analysis*, pages 87–96, 2008.

[PT11] V. Pillwein and S. Takacs. Smoothing analysis of an all-at-once multigrid approach for optimal control roblems using symbolic computation. *Numerical and Symbolic Scientific Computing: Progress and Prospects*, 2011.

[PT14] V. Pillwein and S. Takacs. A local Fourier convergence analysis of a multigrid method using symbolic computation. *J. Symbolic Comp.*, 63:1–20, 2014.

[SB13] C. Schneider and J. Blümlein. Computer algebra in quantum field theory: Integration, Summation and Special Functions. *Springer Publishing Company*, 2013.

[Sch04] C. Schneider. A Collection of Denominator Bounds to Solve Parameterized Linear Difference Equations in ΠΣ-Extensions. *An. Univ. Timisoara Ser. Mat.-Inform.*, 42:163–179, 2004.

[Sch07] C. Schneider. Symbolic Summation Assists Combinatorics. *Sém. Lothar. Combin.*, 56:1–36, 2007.

[Sch08] C. Schneider. A refined difference field theory for symbolic summation. *J. Symbolic Comp.*, 43:611–644, 2008.

[SIK14] T. Sasaki, D. Inaba, and F. Kako. Solving Parametric Sparse Linear Systems. In V.P. Gerdt *et al.*, editor, *Proceedings CASC 2014*, pages 403–418, 2014.

[Str01] S. Stratulat. A General Framework to Build Contextual Cover Set Induction Provers. *J. Symbolic Comp.*, 32:403–445, 2001.

[Wor11] World-Wide Web Consortium. Ink Markup Language (InkML) W3C Recommendation 20 September 2011. `http://www.w3.org/TR/2011/REC-InkML-20110920/`, 2011.

[YDS95] Frances Yao, Alan Demers, and Scott Shenker. A scheduling model for reduced cpu energy. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 374–382. IEEE, 1995.