BCTCS 2013

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

25–27 March 2013

Contents

Chapter 1

Abramsky: Invited

1.1 Introduction

The basic scenario is that we have two or more agents, traditionally Alice, Bob etc. These can make measurements, and observe the values. These give us tuples of values. The experiments can be repeated (not necessarily giving identical results!), getting a sequence of tuples.

In the quantum case, we can be performing spin measurements on a single qubit. These can be visualised as unit vectors in $\mathbf{C}^2$. Spin can be measured in any direction, so there is a continuum of measurements possible. There are two possible outcomes "spin up" and "spin down". Hence a measurement is a pair of antipodal points.

1.2 Quantum Entanglement

This occurs when Bob's qubit depends on Alice's (and *vice versa*). Compound systems are represented as $\mathcal{H}_1 \otimes \mathcal{H}_2$, where a state is

$$\sum_i \lambda_i \cdot \phi_i \otimes \psi_i.$$

Theorem 1 (Bell) *Quantum Mechanics is essentially non-local*

1.3 The Bell Model

A	B	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
a_1	b_1	0	$\frac{1}{2}$	$\frac{1}{2}$	0
a_1	b_2	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{3}{8}$
a_2	b_1	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{3}{8}$
a_2	b_2	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{3}{8}$

What Alice sees should be independent of Bob's measurements, and *vice versa*. This is known as **non-signalling**. Hence QM is consistent with SR.

Also Einstein wanted locality/non-contextuality. The probability of a joint outcome (Alice,Bob) should be a product of the probabilities observed by Alice and by Bob. This is what is not possible.

1.4 Possibilistic Views

Only consider zero/non-zero probabilities.

$$\langle p \rangle(x, y|a, b) = M(x, y|a, b) > 0.$$

A local deterministic model is a function

$$f : \{a_1, a_2, b_1, b_2\} \to \{0, 1\}.$$

It assigns definite outcomes to all measurements simultaneously, independently of the context in which they appear. In the language of sheaf theory, this is a global function.

	(R,R),	(R,G),	(G,R),	(G,G)
(X_1, Y_1)	1			
(X_1, Y_1)	0			
(X_1, Y_1)	0			
(X_1, Y_1)				0

cannot be completed.

It has to be

	(R,R),	(R,G),	(G,R),	(G,G)
(X_1, Y_1)	1			
(X_1, Y_1)	0	1		
(X_1, Y_1)	0			
(X_1, Y_1)	1			0

There is then a contradiction at .... But this system can be built.

1.5 Strong Contextuality

This would say that ...

1.6 Logical Description of GHZ States

The GHZ situation in dimension $n > 2$ is a n-partive systems with two incompatible measurements $X^{(i)}, Y^{(i)})$. We can show that there is no compatible set of assignments to all values, both X and Y.

1.7 Conclusion

Such observation is inherently contextual. Robust CSP — can every consistent partial assignment be extended to a solution? Main result, Robust 2-SAT and Robust 3-colourability are NP-complete.

Chapter 2

Monday 25 March

2.1 Using Patterns to infer Temporal Logic Specifications

Claims that formal specification are **useful** for all sorts of things, *but* are difficult and costly to write. We would like to infer specifications from (correct) programs. Therefore more precisely

> Given a set of **dynamic** traces, **passively** infer a **temporal** specification.

Last time I was testing specifications against a trace, this time I am trying to infer them.

$$\left.\begin{array}{r}\text{trace}\\ \text{pattern library}\end{array}\right\} \overset{\text{generate}}{\Rightarrow} \text{pattern} \overset{\text{test}}{\Rightarrow} \text{pattern} \overset{\text{infer}}{\Rightarrow} \text{specification}$$

If we have behaviours for two files, say, we want to separate them.

Patters need to be predicates on traces that can be combined. Patterns become "open automata", which have a special $\bullet$ "hole" symbol.

Example 1 (Java Standard Library) $\forall c : Collection \forall i : Iterator$ *if we update the collection, we can no longer use the iterator.*

Note that we didn't use automata learning/regular inference, since this constructs an automaton that accepts *exactly* these traces, which is not the point.

Note that the whole technique relies on the pattern library in use. We need a methodology for finding these.

2.2 Lawrence: Program Extraction in Action

"Extraction" means from a constructive proof. We obtain a proof term in the lambda calculus via Curry–Howard correspondence.

2.2.1 Extracting a DPLL Algorithm

DPLL has been verified in Coq and Isabelle, but this was done by starting form the algorithm.

Definition 1 *A literal l is a positive variable $+v$ or a negative $-v$. We use CNF. A* valuation *is a set of literals.*

DPLL consists of five rules. The expected statement of completeness is

$$\forall |Gamma, \Delta(incompatible(\Gamma, \Delta) \to \Gamma \vdash \Delta)$$

The original constructive proof took many seconds, compiled extracted program milliseconds.

2.2.2 Resolution and DPLL Equivalence

We can formalise the equivalence between DPLL and resolution

$$\Gamma \dots$$

$$\forall \Delta . \exists n \dots$$

2.2.3 Clause Learning

The standard DPLL can be seen as brute force, but using unit clauses as heuristics to choose a nice variable ordering. Many SAT solvers make use of clause learning to reduce the search space. When we learn a clause we may need to do non-chronological backtracking, via a conflict graph. Need to carry "decision levels" around with literals. Experimenting with how this modifies DPLL: has one version.

There's a Unit Resolution proof system, which essentially does this. Has a more complicated completeness statement.

We'd like to capture the back-tracking in the resolution proof search,using sets of resolution proofs at each level.

Also, can we optimise the Haskell code further using the LLVM optimiser.

2.3 David Wilson: Advances in Cylindrical Algebraic Decomposition

Introduced by Collins [Col75] for quantifier elimination. Note that we need an order on variables.

Definition 2 *An F-invariant CAD has*

1. *F-sign invariance*
2. *cylindricity (projections equal or disjoint)*

3. semi-algebraic definition.

Example 2 *Two intersecting circles — 55 cells.*

Doubly-exponential complexity [BD07]. Critical point theory might be only singly-exponential, but in practice far worse [Hon91].

- Improved projection [McC85, Bro01]
- Partial CAD [CH91]
- Variable Ordering [DSS04]
- Preconditioning
- Alternative Algorithm [CMMXY09]
- Equational Constraints [McC99]

Our work generalises the last. We input a list of clauses, each of which has an equation constraint (currently must be explicit). Only worries about inequalities when the *corresponding* equations are true.

Example 3 *Spheres and Hyperboloids. Projection CAD 5639 cells, (Maple CAD [CMMXY09] 3xxx) and a comparable number of seconds. TTI CAD 105 cells and 7 seconds.*

Example 4 *Branch cuts. These are transformations of half-lines, so are naturally suited to CAD. For Kahan, a full CAD is 409/1143 cells (depending on ordering), TTI CAD is 55/39. Note the saving in verification time.*

2.3.1 Full-dimensional cells

[McC93, Str94] originally proposed this. We can compute multiple layers, directly or recursively. Again, we can compute just the 25 full-dimensional cells for Kahan.

2.4 Cionca: Path Dependency Analysis in Complex Systems

A complex system will have many agents that interact. These agents can reason, and have their own goals. We model a complex system with business rules amd express the behaviour and interactions between agents. Theya re expressed in a declarative fashion. Agents can be wrapped as web services in distributed transactions. Choreography describes the interaction between agents. Example of booking a trip. Orchestration has no central pint of control, and describes many agents.

Concurrency can be interleaving or non-interleaving. See Π-calculus and CSP.

Example 5 (Food supply chain) *Based on Leontief's Input Output Analysis.*

2.5 Gorry: Faster Communicationless Agent Location Discovery on the Ring

Swarms can be deployed for exploration or monitoring tasks. Many control problems for swarms. Most depend on access to a global picture. Many tasks require agents to know the location of other agents, a non-trivial problem in itself. Our model is a ring with unit circumference, populated by n uniform autonomous agents. Agents that meet just bounce off each other. They don't exchange messages or leave marks on the circle. Each agent should use this to discover the start locations of all other agents. Was $O(n \log^2 n)$, but now $O(n + \log^2 n)$. (Audience objected that this is just $O(n)$! Subsequent conversation shows that he means $n + O(\log^2 n)$: a rather different beast.)

Define a round to be the time it takes an agent to traverse the circle. Virtually, each agent carries a baton, which is swapped when agents collide. Therefore at the end of each round, the starting position p_i of agent a_i is occupied by baton b_i carried by *some* agent. The probability of a successful round in the original algorithm was $\frac{1}{\log n}$ (Prime Number Theorem!).

Chapter 3

Albers: Energy-Efficient Computing

3.1 Motivations

- Mobile devices
- Sensor networks.
- Data centres.

> What matters most at Google is not speed but low power, because Data centres can consume as much electricity as a city — Eric Schmidt, 2002.

Most papers/research discusses

- Power-down mechanisms
- Dynamic speed sensing — both Intel and AMD now offer dynamic speed scale.
- Networks

3.2 Power down

Suppose our system has

active r units/time unit

sleep 0 units/time unit

wake up costs W units (JHD assumes the cost of sleep is rolled into this).

- We ignore transition time, and focus totally on power.

The optimal off-line strategy OPT is therefore to power down if an idle period of length T has $rT > W$. But the real problem is on-line!

Definition 3 *A n on-line algorithm A is c-competitive, if, for all input sequences S, the cost of* $A(S) \leq cOPT(S)$.

Theorem 2 *The algorithm which powers down after* W/r *time units is 2-competitive.*

Theorem 3 *The algorithm RALG which powers down which powers down after a probabilistic (given, JHD didn't catch formula) time units is* $\left(\frac{e}{e-1}\right)$*-competitive.*

Suppose we know the probability distribution Q. Let $ALG - Q$ be the minimal deterministic algorithm for Q.

Theorem 4 *For all* Q*, the algorithm* $ALG - Q$ *is* $\left(\frac{e}{e-1}\right)$*-competitive for* Q.

There are also generalisations to multi-state systems.

3.3 Multi-speed

Definition 4 *Assume the power requires at speed* s *is* $P(s) = s^\alpha$ *with* $\alpha > 1$. Typically $2 \leq \alpha \leq 3$.

Recent work also works with arbitrary convex P.

[YDS95] looks at scheduling with deadlines. Job J_i arrives at a_i, needs to be completed by b_i, has a volume v_i. We allow arbitrary pre-emption.

Lemma 1 *In an optimal solution, a given job is always processed at a constant speed.*

[YDS95] have a polynomial-time solution for the off-line problem. For the on-line setting with an average rate, [YDS95] has $c \leq (\alpha)^\alpha/2$ (and a lower bound pretty close).

In a general setting, there's a lower bound of $e^{\alpha-1}/\alpha$. Proof by convexity of P.

Definition 5 *Let the density of an interval* I *be*

$$\Delta_I = \frac{1}{|I|} \sum_{[a_i, b_i] \subset I} v_i$$

i.e. the work that **has** *be to e done in interval* I.

Then the optimal [YDS95] algorithm is to take the densest interval, solve its jobs, take the interval out (re-formulating jobs that aren't totally in I) and repeat.

3.4 Race to Idle

My work, integrating the two. Speed 0 is more expensive than sleep! [AA12] — execute jobs at high speed then idle. Suppose J_i is executing at speed s_i (constant by Lemma 1) Then the processing costs in

$$E_p(S) = \sum \frac{v_i}{s_i} P(s_i)$$

Note that a job now has a critical speed s_{crit} where the line through the origin of slope $P(s_{\text{crit}})/s_{\text{crit}}$ meets the P curve.

[ISG03] Had a 2-approximation, using s_{crit} where feasible.

[ISG03] Transformation to allow sleep states.

[HLL$^+$10] $\alpha^\alpha + 2$-competitive, where $P(s) = s^a lpha + \gamma$

BCD ...

We showed in the off-line setting that it was still NP-complete. Also no s_{crit}-algorithms can do better than 2-approximation.

For upper bounds we have an an algorithm combining YDS (down to speed s_0), then BCD. This can be $\frac{4}{3}$-approximating.

Best result has $c = \frac{eW_{-1}(-e^{-1-1/e})}{eW_{-1}(-e^{-1-1/e})+1} < 1.211$ (W_{-1} really is the lower branch of the Lambert W function — [BCDJ08]).

Given a schedule $\mathcal{J}$.

- Choose s_0 such that $\frac{P(s_0)}{s_0} = \frac{4}{3}\frac{Ps_{\text{crit}}}{s_{\text{crit}}}$.

1. execute YDS until $\Delta_{\max} \leq s_0$.
2. Let $\mathcal{J}_0 = \mathcal{J} \setminus \mathcal{J}_{YDS}$. Schedule each $J \in \mathcal{J}$ such that ....

* Jobs requiring more than s_{crit} run at their designated speed. The rest are averaged to s_0

Chapter 4

26 March

4.1 Torrini: Parametric Polymorphism, Value Restriction and Resource Logic

Parametric Polymorphism is widely used (SML, OCAML, Haskell etc.). ML-style (Hindley–Milner) polymorphism restriction of F, allows for type inference without annotation. In this talk we will focus on a declarative view of polymorphism. In system F, we have a typing context. It is very expressive, but type inference without type annotation is undecidable. We have the substitution (i.e. cut) rule

$$\frac{\Gamma \vdash c' : \tau' \quad \Gamma, x : \tau' \vdash e : \tau}{\Gamma \vdash e[e'/x] : \tau}$$

Two restrictions in ML-style:

1. Predicative: type parameters can only be instantiated with mono types;
2. Prenex: function arguments can only be mono: never $\forall$ on the left of $\rightarrow$.

In our mini-ML, type parameters range over mono types, and we have a specialisation order $\prec$ on type schemes, so that if $\phi = \phi'[\tau]$ then $\phi' \prec \phi$. `let` is no longer just syntactic sugar, since it gives the defined name its most general type.

References (and exceptions) are a problem. The problem seems to be that our rules are too static. Done carefully, type preservation still works (after Tofte, but with a case distinction in the semantics of `let`). We wish to make this distinction syntactic.

See `www.plancomps.org`.

4.2 McDonald: A Substructural Logic of Layered Graphs

Layered graphs turn up in many contexts:

- Transport systems
- Business systems
- Security

But there's no accepted rigorous definition — much treatment is non-rigorous, or pre-defined layerngs.

Definition 6 *Take a graph $\mathcal{G}$ with a distinguished set of edges $\mathcal{E}$. Write $G_1 \to_{\mathcal{E}} G_2$ if we can get from subgraph G_1 of $\mathcal{G}$ to G_2 using $\mathcal{E}$ (only, JHD thinks).*

Definition 7 *We define a* layered algebra. *Carrier A, Boolean algebra on A, and a bifunctor with left and right adjoints. All scaffolds $(\mathcal{G}, \mathcal{E})$ generate layered magmas. Also can show that layered magmas generate layered algebras.*

4.3 Revell: Relational Semantics of Type Systems

[A Bath Maths undergraduate]

Simply typed λ-calculus as a Cartesian Closed Category. Types and objects, and terms as morphisms. We account for free type variables as a functor: $[[X_1, \ldots, X_n \vdash T type]] : [C^n] \to C$, where $[[\ldots]]$ handle co/contra-variance issues. With the trivial CCC Set, have semantics $[[\_]]_0$

Use the CCC Rel (A, B, R), with $[[\_]]_1$ for its semantics.

$$[[T \to Y]]_1 = [[T]]_1 \to [[U]]_1$$

etc. There's a commuting diagram relating $[[\_]]_1$ and $[[\_]]_0$. This can be done with fibrations. One fibration is $u : (A, B, R) \to (A, B)$.

Theorem 5 (Fundamental Theorem of Logical Relations) *Statement is an enormous commuting diagram.*

Then $([[\_]]_1, [[\_]]_0)$ is a fibred functor.

Theorem 6 (Fundamental Theorem of Logical Relations (again)) *Every Simply-typed Lambda Calculus term defines a fibred natural transformation.*

Note that this allows us to move from Set to terms, or ωCPOs.

I am trying to formalise these for arbitrary fibrations, such as system F. Note that we don't necessarily want to work in Set, because of problems of quantification.

4.4 Lignos: Reconfiguration of colourings of chordal graphs

A configuration graph of a graph problem has

- The vertex set if the set of all feasible solutions
- The edges connect (minimally different) neighbours,

Example 6 *3-colourings of C_5: the configuration graph in fact has two components.*

This gives various questions.

- Is it connected
- Are two elements connected?
- Shortest Path?
- Diameter?

Theorem 7 *Some (stated) problems are coNP.*

Definition 8 *A graph is* chordal *if it has no induced cycle of length more than 3.*

Theorem 8 *For a chordal graph which can be minimally k-coloured, the configuration graph of $k+1$-colourings is ....*

Lemma 2 *Every chordal graph which is not a clique contains maximal cliques C and D where every vertex in C either*

- *belongs to no other maximal clique or*
- *is also in D.*

Theorem 9 *Some problem has diameter $2n^2$.*

Theorem 10 *There is a polynomial-time algorithm for the reconfiguration problem for Hamiltonian cycles for graphs of maximum degree 4.*

Also look as satisfiability reconfiguration.

If a decision problem is in P, can the reconfiguration problem be solved in polynomial time? Also, what is the complexity of finding a path between a pair of solutions of the TSP?

4.5 Chen: Computability of Hybrid Systems

$X \subset \mathbf{R}^n$ is the continuous space, and Q is a set of discrete locations. $f : Q \to (X \to \mathbf{R}^n)$ assigns a continuous vector field on X to each discrete location. I_0 is the initial condition. $Inv : Q \to 2^X$ is the *invariant* of each location. δ is a set of transitions of the form (l, g, γ, l') where g is a transition guard and γ is a reset relation. Simple example is a thermostat with ON/OFF control. Various variants: Timed Automata, Rectangular Automata, and n-dimensional Piecewise Constant Derivative systems (PCD). Also Piecewise Affine Maps (in 1-D) Dom(f) is the union of disjoint intervals I_i. $\forall x \in I_i, f(x) = a_i x + b_i$ for constant a_i, b_i.

Reachability for 1-D PAMs is an open problem. However, for 2-D PCD is known to be decidable, and for 3D PCD known to be undecidable. It is also known that Timed Automata reachability is PSPACE-complete.

Reachability for 2D initialised RA is decidable, but 2D uninitialised is undecidable.

Chapter 5

Mahboubi: Checking Mathematical Proofs with a Computer

5.1 Introduction

Proofs of correctness are important: real-world example is line 14[1] of the Paris Metro. Hard proofs involving lots of computation include

- Four-colour theorem: heavy computation in [AH77];
- Kepler Conjecture [Hal05].

My tool is Coq. A very low-level language compared to the mathematics we write, hence the need for a "proof assistant" to build the 'candidate' proof. Note that this assistant need not be reliable, since it's only building candidate proofs.

The average textbook features

- A set-theoretic flavour
- A common belief that first-order logic is enough
- A shameless use of the Axion of Choice, contradiction etc.

However, Coq proposes

- A type-theoretic formalism
- Higher-order logic
- Constructive logic.

Demonstration of Coq, e.g. building `seq` and `map` (system noted that `map` is decreasing on the list argument).

[1] St. Lazare — Olympiades.

5.2 Odd-Order Theorem

Human proof in [FT63], reworked proof is two complete volumes of LMS Lecture Notes: our proof announced in [GT12].

There is analogy between prime numbers and simple groups, but there isn't a uniqueness result. Classifying Finite Simple Groups is a major process: several (depending on whom you ask) thousand results. The key building block is the following.

Theorem 11 (Odd Order) *A finite simple group of odd order is* $\mathbf{Z}/n\mathbf{Z}$.

Proof outline: let G be a minimal counterexample. "Local Analysis" (the first LMS book) of its maximal subgroups shows there are five types I–V. Character Theory eliminates all but II. This is they attacked by Galois Theory. This gives a fairly formidable list of ingredients, but there is no heavy computation — hence unlike the Four-Colour or Kepler. Consider (probably from [Lan65])

> Let A be a square matrix. Then
>
> $$\det(A) = \sum_{\sigma \in S_n} \prod_i a_{\sigma(i),i}.$$

Note that we don't actually say that A is of size $n \times n$. This would tell us about S_n and the type of i. We have to declare which operations $\sum$ and $\prod$ are instantiated over etc. Would like the Coq to be of the same length as the LaTeX.

Further demo, using first-class type system with dependent types. Shows a class diagram with about 40 classes.

5.3 Conclusion

This is not a super spelling-checker for LaTeX. The process is not actually more difficult for a more difficult proof.

The appropriate data structure emerges from the proof patterns the object is involved in. A single textbook may not help, and there is a lot of trial-and-error and refactoring involved. Programmable type inference decreases the amount of information to be provided by the user.

There is ongoing work in the Coq logic, the proof assistant and the library. `http://www.msr-inria.inria.fr/....`

The length of the files corresponding to the LMS books is about twice the length of the LaTeX source of the books, roughly implying a de Bruijn factor of 2. Of course, we really ought to compile, ZIP etc. first, and the speaker hasn't done that.

5.4

Definition 9

Definition 10

Chapter 6

27 March

6.1 Bak: Rooted Graph Programs

We all know about graphs, and graph transformation. A *rule* is applied to a *host graph* to modify it. A *rule* is (L, K, R): replace L by R where K is the "interface", i.e. between L and the rest of the graph. We assume a fixed grammar, so $|G|^{|L|}$.

We allow a *set* of root nodes, and insist that the transformations don't change the root nodes, and root nodes in a rule must map to root nodes in the graph..

Our GP2 language allows conditional rule schemata, i.e. this transformationcan only be applied if these labels satisfy preconditions. Simple rules for labels in L, which make matching easy:

- no arithmetic expressions
- at most one list variable
- at most one string variable

`if C then P else Q` means if the program C succeeds on G, then execute P on G else execute Q on G. Also a `try` statement

Definition 11 *A conditional rule schema is* fast *if every node is reachable from a root and ....*

We then have a matching algorithm for fast rule schemata. If the number of roots and the degree is bounded then a fast schema is applied in constant time. Then 2-colouring runs in linear time with certain constraints (bounded node degree), or quadratic time otherwise.

6.2 Saleh: Ordered Gauss Paragraphs

Gauss Paragraph is a concept from knot theory.

Definition 12 *An* Euler Diagram *is a set of closed curves in the plane. Curves represent sets, and zones represent intersections.*

We don't allow tangent points etc., so the intersections all consist of transverse double points. We do allow disconnected zones such as $B \cap \overline{A}$, where A and B are congruent ellipses rotated with respect to each other.

We label each crossing point, choose an orientation and starting point for each curve. [Car91].

6.3 Collins: The Men who Stare at Graphs

Need visualisation to look at graph algorithms. Our sources tend to be social media graphs, such as Twitter data about the papal election[1], Facebook data from the CS Department. Data are either harvested or given, and stored in a MySQL database. Sometimes we have GPS data, which gives a starting point for location in terms of drawing the data. Can also get travel graphs as people tweet from new locations. But otherwise how do we draw the graph?

The ellipse layout is effective for graphs where most vertices have low degree. Best idea is "force directed layout": vertices repel and edges pull together. Tends to converge to *local* minima. Detected some spambots, e.g. `anonymous`. For the papal election, there was a clear split between "catholic influencers" and "general media" (speaker's labels). JHD observed "Huffington Post Religion" as a large-ish[2] vertex on the frontier.

Also "hashtag" graphs, N-gram graphs etc.

[1] peaked at 75 tweets/seconds.

[2] The speaker drew vertex size as a function of page rank

Bibliography

[AA12] S. Albers and A. Antoniadis. Race to idle: new algorithms for speed scaling with a sleep state. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1266–1285. SIAM, 2012.

[AH77] K.I. Appel and W. Haken. The Solution of the Four-Color-Map Problem. *Scientific American 4*, 237:108–121, 1977.

[BCDJ08] M. Bronstein, R. Corless, J.H. Davenport, and D.J. Jeffrey. Algebraic properties of the Lambert W function from a result of Rosenstein and Liouville. *J. Integral Transforms and Special Functions*, 18:709–712, 2008.

[BD07] C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In C.W. Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.

[Bro01] C.W. Brown. Improved Projection for Cylindrical Algebraic Decomposition. *J. Symbolic Comp.*, 32:447–465, 2001.

[Car91] J.S. Carter. Classifying immersed curves. *Proc. A.M.S.*, 111:281–287, 1991.

[CH91] G.E. Collins and H. Hong. Partial Cylindrical Algebraic Decomposition for Quantifier Elimination. *J. Symbolic Comp.*, 12:299–328, 1991.

[CMMXY09] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing Cylindrical Algebraic Decomposition via Triangular Decomposition. In J. May, editor, *Proceedings ISSAC 2009*, pages 95–102, 2009.

[Col75] G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

[DSS04] A. Dolzmann, A. Seidl, and Th. Sturm. Efficient Projection Orders for CAD. In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 111–118, 2004.

[FT63] W. Feit and J.G. Thompson. Solvability of Groups of Odd Order. *Pacific J. Math.*, 13:775–1029, 1963.

[GT12] G. Gonthier and L. Théry. Formal Proof — The Feit–Thompson Theorem. `http://www.msr-inria.inria.fr/events-news/feit-thompson-proved-in-coq`, 2012.

[Hal05] T.C. Hales. A proof of the Kepler conjecture. *Ann. Math.*, 162:1065–1185, 2005.

[HLL+10] X. Han, T.W. Lam, L.-K. Lee, I.K.-K. To, and P.W.H. Wong. Deadline scheduling and power management for speed bounded processors. *Theoretical Computer Science*, 411:3587–3600, 2010.

[Hon91] H. Hong. Comparison of several decision algorithms for the existential theory of the reals. Technical Report 91-41, 1991.

[ISG03] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 37–46, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[Lan65] S. Lang. Algebra. *Addison-Wesley*, 1965.

[McC85] S. McCallum. *An Improved Projection Operation for Cylindrical Algebraic Decomposition*. PhD thesis, Univ. Wisconsin at Madison Computer Science Tech. report 548, 1985.

[McC93] S. McCallum. Solving polynomial strict inequalities using cylindrical algebraic decomposition. *Computer J.*, 36:432–438, 1993.

[McC99] S. McCallum. On Projection in CAD-Based Quantifier Elimination with Equational Constraints. In S. Dooley, editor, *Proceedings ISSAC '99*, pages 145–149, 1999.

[Str94] A.W. Strzeboński. An algorithm for systems of strong polynomial inequalities. *The Mathematica Journal*, 4:74–77, 1994.

[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.