SC^2: Satisfiability Checking and Symbolic Computation: www.sc-square.org

James Davenport
Hebron & Medlock Professor of Information Technology[1]

University of Bath (U.K.)
Fulbright Scholar (NYU)

12 May 2017

[1]Thanks to Erika Ábrahám, John Abbott, Bernd Becker, Anna M. Bigatti, Martin Brain, Bruno Buchberger, Alessandro Cimatti, Matthew England, Pascal Fontaine, Stephen Forrest, Alberto Griggio, Daniel Kroening, Werner M. Seiler & Thomas Sturm; H2020-FETOPEN-2016-2017-CSA project 712689

Notation

We are trying in this project to bridge two communities, that of

1. satisfiability checking (especially "satisfiability modulo theories") and

Notation

We are trying in this project to bridge two communities, that of

1. satisfiability checking (especially "satisfiability modulo theories") and
2. symbolic computation, also called computer algebra

Notation

We are trying in this project to bridge two communities, that of

1. satisfiability checking (especially "satisfiability modulo theories") and
2. symbolic computation, also called computer algebra

Notation

We are trying in this project to bridge two communities, that of

1. satisfiability checking (especially "satisfiability modulo theories") and
2. symbolic computation, also called computer algebra

The communities have their own technical terms, which we will distinguish as above

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most k literals is satisfiable.

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most k literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most *k* literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BHvMW09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most *k* literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BHvMW09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

- SAT solvers are in use throughout industry

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most *k* literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BHvMW09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

- SAT solvers are in use throughout industry
- In the UK, I put my life in the hands of SAT-solver verified software several times a week

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most *k* literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BHvMW09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

- SAT solvers are in use throughout industry
- In the UK, I put my life in the hands of SAT-solver verified software several times a week
- SAT-solving contests [JLBRS12] have driven much progress

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most k literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BHvMW09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

- SAT solvers are in use throughout industry
- In the UK, I put my life in the hands of SAT-solver verified software several times a week
- SAT-solving contests [JLBRS12] have driven much progress
- "Watched Literals" [MMZ$^+$01] is worth a factor of $(k-2)$ in the inner loop

Satisfiability Checking

k-SAT: checking whether a conjunction of disjunctions with at most k literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BHvMW09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

- SAT solvers are in use throughout industry
- In the UK, I put my life in the hands of SAT-solver verified software several times a week
- SAT-solving contests [JLBRS12] have driven much progress
- "Watched Literals" [MMZ$^+$01] is worth a factor of $(k-2)$ in the inner loop

#SAT (counting solutions) is a different problem from SAT

Complexity Theory

The 3-SAT problem is known to be NP-complete [Coo71]

The 3-SAT problem is known to be NP-complete [Coo71]
But what does this mean?

Complexity Theory

The 3-SAT problem is known to be NP-complete [Coo71]
But what does this mean?
(Assuming P$\neq$NP)

Complexity Theory

The 3-SAT problem is known to be NP-complete [Coo71]
But what does this mean?
(Assuming P≠NP)

1. There is no polynomial-time algorithm which will solve *all* SAT problems

Complexity Theory

The 3-SAT problem is known to be NP-complete [Coo71]
But what does this mean?
(Assuming P≠NP)

1. There is no polynomial-time algorithm which will solve *all* SAT problems

- But this doesn't necessarily imply exponential running time (though we don't know much better)

Complexity Theory

The 3-SAT problem is known to be NP-complete [Coo71]
But what does this mean?
(Assuming P≠NP)

1. There is no polynomial-time algorithm which will solve *all* SAT problems
   - But this doesn't necessarily imply exponential running time (though we don't know much better)
2. Any given SAT problem can be solved in polynomial time

SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans

SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans

- Substantial progress has been made when the theory is "easy" [BSST09, KS08]

SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans

- Substantial progress has been made when the theory is "easy" [BSST09, KS08]
- But even quantifier-free (i.e. purely existential) SMT for theories of non-linear arithmetic/algebra, real or integer, is still in its infancy

SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans

- Substantial progress has been made when the theory is "easy" [BSST09, KS08]
- But even quantifier-free (i.e. purely existential) SMT for theories of non-linear arithmetic/algebra, real or integer, is still in its infancy
- quantified (i.e. at least one alternation) SMT is currently a dream

SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans

- Substantial progress has been made when the theory is "easy" [BSST09, KS08]
- But even quantifier-free (i.e. purely existential) SMT for theories of non-linear arithmetic/algebra, real or integer, is still in its infancy
- quantified (i.e. at least one alternation) SMT is currently a dream

SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans

- Substantial progress has been made when the theory is "easy" [BSST09, KS08]
- But even quantifier-free (i.e. purely existential) SMT for theories of non-linear arithmetic/algebra, real or integer, is still in its infancy
- quantified (i.e. at least one alternation) SMT is currently a dream

"*Despite substantial advances in verification technology, complexity issues with classical decision procedures are still a major obstacle for formal verification of real-world applications, e.g. in automotive and avionic industries.*" [PQR09]

But isn't this standard computer algebra?

(at least over the reals)

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since
- it's in several computer algebra systems

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since
- it's in several computer algebra systems
- and it's even possible to eliminate a quantifier on an Android 'phone [Eng14]

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since
- it's in several computer algebra systems
- and it's even possible to eliminate a quantifier on an Android 'phone [Eng14]
- Of course, it's expensive, but we know the problem is doubly-exponential [BD07]

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since
- it's in several computer algebra systems
- and it's even possible to eliminate a quantifier on an Android 'phone [Eng14]
- Of course, it's expensive, but we know the problem is doubly-exponential [BD07]

But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since
- it's in several computer algebra systems
- and it's even possible to eliminate a quantifier on an Android 'phone [Eng14]
- Of course, it's expensive, but we know the problem is doubly-exponential [BD07]

Over the integers it's undecidable anyway, so what's the point?

But there's a fundamental difference

But there's a fundamental difference

Computer Algebra Begins with the polynomials, solves them completely (Cylindrical Algebraic Decomposition), then considers the Boolean structure

But there's a fundamental difference

Computer Algebra Begins with the polynomials, solves them completely (Cylindrical Algebraic Decomposition), then considers the Boolean structure

With some more recent flexibility, e.g. equational constraints.

But there's a fundamental difference

Computer Algebra Begins with the polynomials, solves them completely (Cylindrical Algebraic Decomposition), then considers the Boolean structure

With some more recent flexibility, e.g. equational constraints.

Hence we are essentially solving #SMT, rather than SMT

But there's a fundamental difference

Computer Algebra: Begins with the polynomials, solves them completely (Cylindrical Algebraic Decomposition), then considers the Boolean structure

With: some more recent flexibility, e.g. equational constraints.

Hence: we are essentially solving #SMT, rather than SMT

But see: single-cell constructions [Bro13, Bro15]

But there's a fundamental difference

Computer Algebra Begins with the polynomials, solves them completely (Cylindrical Algebraic Decomposition), then considers the Boolean structure

With some more recent flexibility, e.g. equational constraints.

Hence we are essentially solving #SMT, rather than SMT

But see single-cell constructions [Bro13, Bro15]

SMT Starts from the Boolean structure, and dips into the theory, adding and retracting theory clauses as required

There's also a question of strategy

There's also a question of strategy

Computer Algebra tends to have a fixed strategy

There's also a question of strategy

Computer Algebra tends to have a fixed strategy

at least in terms of what is documented: the pre-processing steps before one gets into the algorithm are rarely described

There's also a question of strategy

Computer Algebra
: tends to have a fixed strategy

at least
: in terms of what is documented: the pre-processing steps before one gets into the algorithm are rarely described

Quite often
: follows a general algorithm even when there's some "low hanging fruit"

There's also a question of strategy

Computer Algebra tends to have a fixed strategy

at least in terms of what is documented: the pre-processing steps before one gets into the algorithm are rarely described

Quite often follows a general algorithm even when there's some "low hanging fruit"

SAT tends to have lots of heuristics

There's also a question of strategy

Computer Algebra tends to have a fixed strategy

at least in terms of what is documented: the pre-processing steps before one gets into the algorithm are rarely described

Quite often follows a general algorithm even when there's some "low hanging fruit"

SAT tends to have lots of heuristics

SAT looks aggressively for low-hanging fruit [Spe15]

There's also a question of strategy

Computer Algebra tends to have a fixed strategy

at least in terms of what is documented: the pre-processing steps before one gets into the algorithm are rarely described

Quite often follows a general algorithm even when there's some "low hanging fruit"

SAT tends to have lots of heuristics

SAT looks aggressively for low-hanging fruit [Spe15]

SAT Frequently restarts [HH10], with some underpinning theory [LSZ93]

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order The most obvious one (also present in Gröbner bases, regular chains etc.)

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order
: The most obvious one (also present in Gröbner bases, regular chains etc.)

Often
: Crucial, in theory [BD07] and in practice

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order The most obvious one (also present in Gröbner bases, regular chains etc.)

Often Crucial, in theory [BD07] and in practice

Several heuristics suggested in the past: [HEW^{+}15] shows that no one heuristic is best, and a machine learning meta-heuristic outperforms all heuristics

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order The most obvious one (also present in Gröbner bases, regular chains etc.)

Often Crucial, in theory [BD07] and in practice

Several heuristics suggested in the past: [HEW+15] shows that no one heuristic is best, and a machine learning meta-heuristic outperforms all heuristics

Equational constraints We can only apply one for each variable, so need to choose

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order The most obvious one (also present in Gröbner bases, regular chains etc.)

Often Crucial, in theory [BD07] and in practice

Several heuristics suggested in the past: [HEW+15] shows that no one heuristic is best, and a machine learning meta-heuristic outperforms all heuristics

Equational constraints We can only apply one for each variable, so need to choose

No cheap heuristics: those available do all the projections then decide which one to lift

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order
: The most obvious one (also present in Gröbner bases, regular chains etc.)

    Often
    : Crucial, in theory [BD07] and in practice

    Several
    : heuristics suggested in the past: [HEW+15] shows that no one heuristic is best, and a machine learning meta-heuristic outperforms all heuristics

Equational constraints
: We can only apply one for each variable, so need to choose

    No cheap
    : heuristics: those available do all the projections then decide which one to lift

    TTICAD
    : "Truth Table Invariant CAD", i.e. trying to take account of the Boolean structure, has even more choices

Heuristics

In fact,there's a great deal of choice in CAD "algorithms".

Variable Order
: The most obvious one (also present in Gröbner bases, regular chains etc.)

Often
: Crucial, in theory [BD07] and in practice

Several
: heuristics suggested in the past: [HEW+15] shows that no one heuristic is best, and a machine learning meta-heuristic outperforms all heuristics

Equational constraints
: We can only apply one for each variable, so need to choose

No cheap
: heuristics: those available do all the projections then decide which one to lift

TTICAD
: "Truth Table Invariant CAD", i.e. trying to take account of the Boolean structure, has even more choices

Also
: No research in trying to make all the choices holistically.

Benchmarking, Problem Sets and Contests

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Specification
: Various different questions: [WBD12] is just CAD problems, not SMT problems

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Specification Various different questions: [WBD12] is just CAD problems, not SMT problems

Maintenance is a problem, see the PoSSo set of GB examples (only conserved in PDF of LaTeX)

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Specification Various different questions: [WBD12] is just CAD problems, not SMT problems

Maintenance is a problem, see the PoSSo set of GB examples (only conserved in PDF of LaTeX)

Language Not really a standard: we will extend the SMTLib standard — interested in volunteers/ interfaces; OpenDreamKit?; OpenMath; MathML-C;

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Specification Various different questions: [WBD12] is just CAD problems, not SMT problems

Maintenance is a problem, see the PoSSo set of GB examples (only conserved in PDF of LaTeX)

Language Not really a standard: we will extend the SMTLib standard — interested in volunteers/ interfaces; OpenDreamKit?; OpenMath; MathML-C;

but need a problem statement language as well as just formulae

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Specification Various different questions: [WBD12] is just CAD problems, not SMT problems

Maintenance is a problem, see the PoSSo set of GB examples (only conserved in PDF of LaTeX)

Language Not really a standard: we will extend the SMTLib standard — interested in volunteers/ interfaces; OpenDreamKit?; OpenMath; MathML-C;

but need a problem statement language as well as just formulae

Industry Not much current industrial use, so no industry problems, vicious circle

Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

Specification Various different questions: [WBD12] is just CAD problems, not SMT problems

Maintenance is a problem, see the PoSSo set of GB examples (only conserved in PDF of LaTeX)

Language Not really a standard: we will extend the SMTLib standard — interested in volunteers/ interfaces; OpenDreamKit?; OpenMath; MathML-C;

but need a problem statement language as well as just formulae

Industry Not much current industrial use, so no industry problems, vicious circle

Hard Problems? Quite a challenge for SAT [Spe15]

Hard Problems

CAD is known to be doubly-exponential (in n, the number of variables)

Hard Problems

CAD is known to be doubly-exponential (in n, the number of variables)

[DH88] Describing a single (non-trivial) solution needs polynomials of degree $2^{2^{n/5+O(1)}}$

Hard Problems

CAD is known to be doubly-exponential (in n, the number of variables)

[DH88] Describing a single (non-trivial) solution needs polynomials of degree $2^{2^{n/5+O(1)}}$

* So adding $\wedge 0 < x < 1$ makes describing a single solution doubly-exponentially more difficult

Hard Problems

CAD is known to be doubly-exponential (in n, the number of variables)

[DH88] Describing a single (non-trivial) solution needs polynomials of degree $2^{2^{n/5+O(1)}}$

* So adding $\wedge 0 < x < 1$ makes describing a single solution doubly-exponentially more difficult

[BD07] The solutions are all rational, describable with $2^{O(n)}$ bits. But there are $2^{2^{O(n)}}$ of them, so SMT might be $2^{O(n)}$ but #SMT $2^{2^{O(n)}}$

Hard Problems

CAD is known to be doubly-exponential (in n, the number of variables)

[DH88] Describing a single (non-trivial) solution needs polynomials of degree $2^{2^{n/5+O(1)}}$

* So adding $\wedge 0 < x < 1$ makes describing a single solution doubly-exponentially more difficult

[BD07] The solutions are all rational, describable with $2^{O(n)}$ bits. But there are $2^{2^{O(n)}}$ of them, so SMT might be $2^{O(n)}$ but #SMT $2^{2^{O(n)}}$

But There is symmetry, and we don't have to count the solutions one-by-one, so what is #SMT here?

Conclusions

We currently have two communities with different

Terminology Minor once you're aware of it

Conclusions

We currently have two communities with different

Terminology	Minor once you're aware of it
Approaches	Logic-first versus (historically) polynomials-first

Conclusions

We currently have two communities with different

Terminology	Minor once you're aware of it
Approaches	Logic-first versus (historically) polynomials-first
Also	incremental versus batch

Conclusions

We currently have two communities with different

Terminology	Minor once you're aware of it
Approaches	Logic-first versus (historically) polynomials-first
Also	incremental versus batch
Attitudes	Pragmatic contests versus worst-case complexity

Conclusions

We currently have two communities with different

Terminology	Minor once you're aware of it
Approaches	Logic-first versus (historically) polynomials-first
Also	incremental versus batch
Attitudes	Pragmatic contests versus worst-case complexity
Hence	problem sets, contests, standards etc.

Conclusions

We currently have two communities with different

Terminology Minor once you're aware of it

Approaches Logic-first versus (historically) polynomials-first

Also incremental versus batch

Attitudes Pragmatic contests versus worst-case complexity

Hence problem sets, contests, standards etc.

Industrial links (but currently not very strong for either: SMT can point to SAT).

Conclusions

We currently have two communities with different

- **Terminology** Minor once you're aware of it
- **Approaches** Logic-first versus (historically) polynomials-first
- **Also** incremental versus batch
- **Attitudes** Pragmatic contests versus worst-case complexity
- **Hence** problem sets, contests, standards etc.
- **Industrial links** (but currently not very strong for either: SMT can point to SAT).
- **So** We have a lot of work to do.

Gröbner bases: [MR13] versus [MM82]

Gröbner bases: [MR13] versus [MM82]

Let r be the dimension of the variety of solutions.

Gröbner bases: [MR13] versus [MM82]

Let r be the dimension of the variety of solutions. Focus on the degrees of the polynomials (more intrinsic than actual times)

Gröbner bases: [MR13] versus [MM82]

Let r be the dimension of the variety of solutions. Focus on the degrees of the polynomials (more intrinsic than actual times)

[MR13] modified both lower and upper bounds to show $\mathfrak{d}^{n^{\Theta(1)} 2^{\Theta(r)}}$

lower Essentially, use the r-variable [Yap91] ideal

Gröbner bases: [MR13] versus [MM82]

Let r be the dimension of the variety of solutions. Focus on the degrees of the polynomials (more intrinsic than actual times)

[MR13] modified both lower and upper bounds to show $\mathfrak{d}^{n^{\Theta(1)} 2^{\Theta(r)}}$

lower
: Essentially, use the r-variable [Yap91] ideal

which
: encodes an EXPSPACE-complete rewriting problem into a system of binomials

Gröbner bases: [MR13] versus [MM82]

Let r be the dimension of the variety of solutions. Focus on the degrees of the polynomials (more intrinsic than actual times)

[MR13] modified both lower and upper bounds to show $\mathfrak{d}^{n^{\Theta(1)} 2^{\Theta(r)}}$

lower Essentially, use the r-variable [Yap91] ideal

which encodes an EXPSPACE-complete rewriting problem into a system of binomials

note that these ideals are definitely not radical (square-free)

Gröbner bases: [MR13] versus [MM82]

Let r be the dimension of the variety of solutions. Focus on the degrees of the polynomials (more intrinsic than actual times) [MR13] modified both lower and upper bounds to show $\mathfrak{d}^{n^{\Theta(1)}2^{\Theta(r)}}$

- **lower** Essentially, use the r-variable [Yap91] ideal
- **which** encodes an EXPSPACE-complete rewriting problem into a system of binomials
- **note** that these ideals are definitely not radical (square-free)
- **upper** A very significant improvement to [Dub90], again using r rather than n where possible

What we would like to do (but can't)

What we would like to do (but can't)

Show radical ideal problems are only singly-exponential in n

What we would like to do (but can't)

Show radical ideal problems are only singly-exponential in n

This ought to follow from [Kol88]

What we would like to do (but can't)

Show radical ideal problems are only singly-exponential in n

This ought to follow from [Kol88]

Show non-radical ideals are rare (non-square-free polynomials occur with density 0)

What we would like to do (but can't)

Show	radical ideal problems are only singly-exponential in n
This	ought to follow from [Kol88]
Show	non-radical ideals are rare (non-square-free polynomials occur with density 0)
However	there seems to be no theory of distribution of ideals

What we would like to do (but can't)

Show radical ideal problems are only singly-exponential in n

This ought to follow from [Kol88]

Show non-radical ideals are rare (non-square-free polynomials occur with density 0)

However there seems to be no theory of distribution of ideals

Deduce weak worst-case complexity (i.e. apart from an exponentially-rare subset: [AL15]) of Gröbner bases is singly exponential

There's a catch [Chi09]

Theorem

$\forall n \geq n_0, d \geq d_0$ there are homogeneous $f_1, \ldots, f_\nu \in k[x_1, \ldots, x_n]$ ($\nu \leq n$, $\deg f_i \leq d$) and a prime ideal $\mathfrak{p}$ such that

1. *the zeros $\mathcal{Z}(\mathfrak{p})$ coincides with a component, defined over k, of $\mathcal{Z}(f_1, \ldots, f_\nu)$, and furthermore $\mathcal{Z}(f_1, \ldots, f_\nu)$ has exactly two components irreducible over $\overline{k}$: $\mathcal{Z}(\mathfrak{p})$ and linear space;*
2. *the Hilbert function of $\mathfrak{p}$ only stabilised after $d^{2^{\Omega(n)}}$;*
3. *the maximum degree of any system of generators of $\mathfrak{p}$ is $d^{2^{\Omega(n)}}$.*

There's a catch [Chi09]

Theorem

$\forall n \geq n_0, d \geq d_0$ there are homogeneous $f_1, \ldots, f_\nu \in k[x_1, \ldots, x_n]$ ($\nu \leq n$, $\deg f_i \leq d$) and a prime ideal $\mathfrak{p}$ such that

1. *the zeros $\mathcal{Z}(\mathfrak{p})$ coincides with a component, defined over k, of $\mathcal{Z}(f_1, \ldots, f_\nu)$, and furthermore $\mathcal{Z}(f_1, \ldots, f_\nu)$ has exactly two components irreducible over $\overline{k}$: $\mathcal{Z}(\mathfrak{p})$ and linear space;*
2. *the Hilbert function of $\mathfrak{p}$ only stabilised after $d^{2^{\Omega(n)}}$;*
3. *the maximum degree of any system of generators of $\mathfrak{p}$ is $d^{2^{\Omega(n)}}$.*

I don't fully understand the construction: it starts with [Yap91], as [MR13], but somehow builds a prime ideal inside this, with embedded high-multiplicity components

A technical complication, and solution

Making sets of polynomials square-free

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

but might leave the degree alone, or might replace one polynomial by $O(\sqrt{d})$ polynomials

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

but might leave the degree alone, or might replace one polynomial by $O(\sqrt{d})$ polynomials

hard to control from the point of view of complexity theory.

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

but might leave the degree alone, or might replace one polynomial by $O(\sqrt{d})$ polynomials

hard to control from the point of view of complexity theory.

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

but might leave the degree alone, or might replace one polynomial by $O(\sqrt{d})$ polynomials

hard to control from the point of view of complexity theory.

Solution [McC84] Say that a set of polynomials has the (M, D) property if it can be partitioned into M sets, each with combined degree at most D (in each variable)

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

but might leave the degree alone, or might replace one polynomial by $O(\sqrt{d})$ polynomials

hard to control from the point of view of complexity theory.

Solution [McC84] Say that a set of polynomials has the (M, D) property if it can be partitioned into M sets, each with combined degree at most D (in each variable)

This is preserved by taking square-free decompositions etc.

A technical complication, and solution

Making sets of polynomials square-free, or even irreducible,

- is computationally nearly always advantageous
- is sometimes required by the theory

but might leave the degree alone, or might replace one polynomial by $O(\sqrt{d})$ polynomials

hard to control from the point of view of complexity theory.

Solution [McC84] Say that a set of polynomials has the (M, D) property if it can be partitioned into M sets, each with combined degree at most D (in each variable)

This is preserved by taking square-free decompositions etc.

Can Define $(M, \mathfrak{D})$ analogously

Cylindrical Algebraic Decomposition for polynomials

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

Cylindrical Algebraic Decomposition for polynomials

Assume: All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However: there is no theory of distribution of CADs

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Note [MPP16] says this is no longer relevant

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Note [MPP16] says this is no longer relevant

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Note [MPP16] says this is no longer relevant

Then if A_n is the polynomials in n variables, with primitive irreducible basis B_n, the projection is

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Note [MPP16] says this is no longer relevant

Then if A_n is the polynomials in n variables, with primitive irreducible basis B_n, the projection is

$$A_{n-1} := \mathrm{cont}(A_n) \cup [\mathcal{P}(B_n) := \mathrm{coeff}(B_n) \cup \mathrm{disc}(B_n) \cup \mathrm{res}(B_n)]$$

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Note [MPP16] says this is no longer relevant

Then if A_n is the polynomials in n variables, with primitive irreducible basis B_n, the projection is

$$A_{n-1} := \mathrm{cont}(A_n) \cup [\mathcal{P}(B_n) := \mathrm{coeff}(B_n) \cup \mathrm{disc}(B_n) \cup \mathrm{res}(B_n)]$$

If A_n has (M, D) then A_{n-1} has $\left((M+1)^2/2, 2D^2\right)$

Cylindrical Algebraic Decomposition for polynomials

Assume: All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However: there is no theory of distribution of CADs

And: Bath has a family of examples which aren't well-oriented

And: rescuing from failure is doable, but not well-studied

Note: [MPP16] says this is no longer relevant

Then if A_n is the polynomials in n variables, with primitive irreducible basis B_n, the projection is

$$A_{n-1} := \mathrm{cont}(A_n) \cup [\mathcal{P}(B_n) := \mathrm{coeff}(B_n) \cup \mathrm{disc}(B_n) \cup \mathrm{res}(B_n)]$$

If A_n has (M, D) then A_{n-1} has $\left((M+1)^2/2, 2D^2\right)$
Hence doubly-exponential growth in n

Cylindrical Algebraic Decomposition for polynomials

Assume All CADs we encounter are well-oriented [McC84], i.e. no relevant polynomial vanishes identically on a cell

However there is no theory of distribution of CADs

And Bath has a family of examples which aren't well-oriented

And rescuing from failure is doable, but not well-studied

Note [MPP16] says this is no longer relevant

Then if A_n is the polynomials in n variables, with primitive irreducible basis B_n, the projection is

$$A_{n-1} := \mathrm{cont}(A_n) \cup [\mathcal{P}(B_n) := \mathrm{coeff}(B_n) \cup \mathrm{disc}(B_n) \cup \mathrm{res}(B_n)]$$

If A_n has (M, D) then A_{n-1} has $\left((M+1)^2/2, 2D^2\right)$

Hence doubly-exponential growth in n

The induction (on n) hypothesis is order-invariant decompositions

Cylindrical Algebraic Decomposition for propositions (1)

Suppose we are tryimg to understand (e.g. quantifier elimination) a proposition Φ (or set of propositions)

Cylindrical Algebraic Decomposition for propositions (1)

Suppose we are tryimg to understand (e.g. quantifier elimination) a proposition Φ (or set of propositions), and $f(\mathbf{x}) = 0$ is a consequence of Φ (either explicit or implicit), an equational constraint, and f involves x_n and is primitive

Cylindrical Algebraic Decomposition for propositions (1)

Suppose we are tryimg to understand (e.g. quantifier elimination) a proposition Φ (or set of propositions), and $f(\mathbf{x}) = 0$ is a consequence of Φ (either explicit or implicit), an equational constraint, and f involves x_n and is primitive

Then [Col98] we are only interested in $\mathbf{R}^n | f(\mathbf{x}) = 0$, not $\mathbf{R}^n$

Cylindrical Algebraic Decomposition for propositions (1)

Suppose we are tryimg to understand (e.g. quantifier elimination) a proposition Φ (or set of propositions), and $f(\mathbf{x}) = 0$ is a consequence of Φ (either explicit or implicit), an equational constraint, and f involves x_n and is primitive
Then [Col98] we are only interested in $\mathbf{R}^n | f(\mathbf{x}) = 0$, not $\mathbf{R}^n$
So [McC99] let F be an irreducible basis for f, and use
$\mathcal{P}_F(B) := \mathcal{P}(F) \cup \{\mathrm{res}(f, b) | f \in F, b \in B \setminus F\}$

Cylindrical Algebraic Decomposition for propositions (1)

Suppose we are tryimg to understand (e.g. quantifier elimination) a proposition Φ (or set of propositions), and $f(\mathbf{x}) = 0$ is a consequence of Φ (either explicit or implicit), an equational constraint, and f involves x_n and is primitive
Then [Col98] we are only interested in $\mathbf{R}^n | f(\mathbf{x}) = 0$, not $\mathbf{R}^n$
So [McC99] let F be an irreducible basis for f, and use
$\mathcal{P}_F(B) := \mathcal{P}(F) \cup \{\mathrm{res}(f, b) | f \in F, b \in B \setminus F\}$
This has $(2M, 2D^2)$ rather than $(O(M^2), 2D^2)$

Cylindrical Algebraic Decomposition for propositions (1)

Suppose we are tryimg to understand (e.g. quantifier elimination) a proposition Φ (or set of propositions), and $f(\mathbf{x}) = 0$ is a consequence of Φ (either explicit or implicit), an equational constraint, and f involves x_n and is primitive
Then [Col98] we are only interested in $\mathbf{R}^n | f(\mathbf{x}) = 0$, not $\mathbf{R}^n$
So [McC99] let F be an irreducible basis for f, and use
$\mathcal{P}_F(B) := \mathcal{P}(F) \cup \{\mathrm{res}(f, b) | f \in F, b \in B \setminus F\}$
This has $(2M, 2D^2)$ rather than $(O(M^2), 2D^2)$, but only produces a sign-invariant decomposition

Cylindrical Algebraic Decomposition for propositions (2)

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}^*_F(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}^*_F(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$
If $f(\mathbf{x}) = 0$ and $g(\mathbf{x}) = 0$ are both equational constraints, then $\mathrm{res}_{x_n}(f, g)$ is also an equational constraint

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}^*_F(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$

If $f(\mathbf{x}) = 0$ and $g(\mathbf{x}) = 0$ are both equational constraints, then $\mathrm{res}_{x_n}(f, g)$ is also an equational constraint

Suppose we have s equational constraints

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}^*_F(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$

If $f(\mathbf{x}) = 0$ and $g(\mathbf{x}) = 0$ are both equational constraints, then $\mathrm{res}_{x_n}(f, g)$ is also an equational constraint

Suppose we have s equational constraints

And (after resultants) we have a constraint in each of the last s variables

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}_F^*(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$

If $f(\mathbf{x}) = 0$ and $g(\mathbf{x}) = 0$ are both equational constraints, then $\mathrm{res}_{x_n}(f, g)$ is also an equational constraint

Suppose we have s equational constraints

And (after resultants) we have a constraint in each of the last s variables

And these constraints are all primitive

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}^*_F(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$

If $f(\mathbf{x}) = 0$ and $g(\mathbf{x}) = 0$ are both equational constraints, then $\mathrm{res}_{x_n}(f, g)$ is also an equational constraint

Suppose we have s equational constraints

And (after resultants) we have a constraint in each of the last s variables

And these constraints are all primitive

Cylindrical Algebraic Decomposition for propositions (2)

Generalised to $\mathcal{P}^*_F(B) := \mathcal{P}_F(B) \cup \mathrm{disc}(B \setminus F)$ [McC01], which produces an order-invariant decomposition, and has $(3M, 2D^2)$

If $f(\mathbf{x}) = 0$ and $g(\mathbf{x}) = 0$ are both equational constraints, then $\mathrm{res}_{x_n}(f, g)$ is also an equational constraint

Suppose we have s equational constraints

And (after resultants) we have a constraint in each of the last s variables

And these constraints are all primitive

Then [EBD15] we get $O\left(m^{s2^{n-s}} d^{2^n}\right)$ behaviour

Recent Developments

Recent Developments

CASC 2016[ED16] Under the same assumptions,

$O\left(m^{s2^{n-s}} d^{s2^{n-s}}\right)$ behaviour

Recent Developments

CASC 2016[ED16] Under the same assumptions,

$O\left(m^{s2^{n-s}} d^{s2^{n-s}}\right)$ behaviour

using Gröbner bases rather than resultants for the elimination, but multivariate resultants [BM09] for the bounds

Recent Developments

CASC 2016[ED16] Under the same assumptions,
$O\left(m^{s2^{n-s}} d^{s2^{n-s}}\right)$ behaviour

using Gröbner bases rather than resultants for the elimination, but multivariate resultants [BM09] for the bounds

ICMS 2016[DE16] The primitivity restriction is inherent: we can write [DH88] in this format, with $n-1$ non-primitive equational constraints

Recent Developments

CASC 2016[ED16] Under the same assumptions, $O\left(m^{s2^{n-s}} d^{s2^{n-s}}\right)$ behaviour

using Gröbner bases rather than resultants for the elimination, but multivariate resultants [BM09] for the bounds

ICMS 2016[DE16] The primitivity restriction is inherent: we can write [DH88] in this format, with $n-1$ non-primitive equational constraints

ISSAC2017 [BDE+17] Can do Cylindrical Algebraic Decomposition in 12 variables with 11 equational constraints

it's not $\mathbf{R}/\mathbf{C}$: it's quantifiers (and alternations)

[DH88, BD07] Are really about the combinatorial complexity of

it's not **R**/**C**: it's quantifiers (and alternations)

[DH88, BD07] Are really about the combinatorial complexity of

Let $S_k(x_k, y_k)$ be the statement $x_k = f(y_k)$ and then define recursively $S_{k-1}(x_{k-1}, y_{k-1}) := x_{k-1} = f(f(y_{k-1})) :=$

$$\underbrace{\exists z_k \forall x_k \forall y_k}_{Q_k} \underbrace{((y_{k-1} = y_k \wedge x_k = z_k) \vee (y_k = z_k \wedge x_{k-1} = x_k))}_{L_k} \Rightarrow S_k(x_k, y_k)$$

it's not $\mathbf{R}/\mathbf{C}$: it's quantifiers (and alternations)

[DH88, BD07] Are really about the combinatorial complexity of

Let $S_k(x_k, y_k)$ be the statement $x_k = f(y_k)$ and then define recursively $S_{k-1}(x_{k-1}, y_{k-1}) := x_{k-1} = f(f(y_{k-1})) :=$

$$\underbrace{\exists z_k \forall x_k \forall y_k}_{Q_k} \underbrace{((y_{k-1} = y_k \wedge x_k = z_k) \vee (y_k = z_k \wedge x_{k-1} = x_k))}_{L_k} \Rightarrow S_k(x_k, y_k)$$

We can transpose this to the complexes, and get zero-dimensional QE examples in $\mathbf{C}^n$ with $2^{2^{O(n)}}$ isolated point solutions, even though the equations are all linear and the solution set is zero-dimensional.

So let's not be mesmerised by the QE problem

Consider ([BDE+17]) a single semi-algebraic set defined by

$$f_1(x_1,\ldots,x_{n-1},k_1)=0 \land f_2(x_1,\ldots,x_{n-1},k_1)=0 \land \cdots$$
$$f_{n-1}(x_1,\ldots,x_{n-1},k_1)=0 \land x_1>0 \land \cdots \land x_{n-1}>0$$

So let's not be mesmerised by the QE problem

Consider ([BDE+17]) a single semi-algebraic set defined by

$$f_1(x_1,\ldots,x_{n-1},k_1)=0 \land f_2(x_1,\ldots,x_{n-1},k_1)=0 \land \cdots$$
$$f_{n-1}(x_1,\ldots,x_{n-1},k_1)=0 \land x_1>0 \land \cdots \land x_{n-1}>0$$

and ask the question "How does the number of solutions vary with k_1?"

So let's not be mesmerised by the QE problem

Consider ([BDE+17]) a single semi-algebraic set defined by

$$f_1(x_1,\ldots,x_{n-1},k_1)=0 \wedge f_2(x_1,\ldots,x_{n-1},k_1)=0 \wedge \cdots$$
$$f_{n-1}(x_1,\ldots,x_{n-1},k_1)=0 \wedge x_1>0 \wedge \cdots \wedge x_{n-1}>0$$

and ask the question "How does the number of solutions vary with k_1?" The f_i are multilinear ($d=1$ but $\mathfrak{d}=2,3,4$) and primitive, and are pretty "generic".

So let's not be mesmerised by the QE problem

Consider ([BDE+17]) a single semi-algebraic set defined by

$$f_1(x_1,\ldots,x_{n-1},k_1)=0 \wedge f_2(x_1,\ldots,x_{n-1},k_1)=0 \wedge \cdots$$
$$f_{n-1}(x_1,\ldots,x_{n-1},k_1)=0 \wedge x_1>0 \wedge \cdots \wedge x_{n-1}>0$$

and ask the question "How does the number of solutions vary with k_1?" The f_i are multilinear ($d=1$ but $\mathfrak{d}=2,3,4$) and primitive, and are pretty "generic".
Of course, this doesn't guarantee that all the iterated resultants in [EBD15], or the Gröbner polynomials in [ED16], are primitive, but in practice they are.

Bibliography I

D. Amelunxen and M. Lotz.
Average-case complexity without the black swans.
http://arxiv.org/abs/1512.09290, 2015.

C. W. Brown and J. H. Davenport.
The complexity of quantifier elimination and cylindrical algebraic decomposition.
In *Proceedings ISSAC 2007*, pages 54–60. ACM, 2007.

R.J. Bradford, J.H. Davenport, M. England, H. Errami, V. Gerdt, D. Grigoriev, C. Hoyt, M. Kosta, O. Radulescu, T. Sturm, and A. Weber.
A Case Study on the Parametric Occurrence of Multiple Steady States.
https://arxiv.org/abs/1704.08997, 2017.

Bibliography II

A. Biere, M. Heule, H. van Maaren, and T. Walsh.
Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*.
IOS Press, 2009.

L. Busé and B. Mourrain.
Explicit factors of some iterated resultants and discriminants.
Math. Comp., 78:345–386, 2009.

C.W. Brown.
Constructing a single open cell in a cylindrical algebraic decomposition.
In *Proceedings ISSAC 2013*, pages 133–140, 2013.

C.W. Brown.
Open Non-uniform Cylindrical Algebraic Decompositions.
In *Proceedings ISSAC 2015*, pages 85–92, 2015.

C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli.
Satisfiability modulo theories.
In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, 2009.

A.L. Chistov.
Double-exponential lower bound for the degree of any system of generators of a polynomial prime ideal.
St. Petersburg Math. J., 20:983–1001, 2009.

Bibliography IV

G. E. Collins.
Quantifier elimination for real closed fields by cylindrical algebraic decomposition.
In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

G.E. Collins.
Quantifier elimination by cylindrical algebraic decomposition — twenty years of progess.
In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 8–23. Springer Verlag, Wien, 1998.

S. A. Cook.
The complexity of theorem-proving procedures.
In *Proceedings STOC 1971*, pages 151–158. ACM, 1971.

Bibliography V

J.H. Davenport and M. England.
Need Polynomial Systems be Doubly-exponential?
In *Proceedings ICMS 2016*, pages 157–164, 2016.

J. H. Davenport and J. Heintz.
Real quantifier elimination is doubly exponential.
J. Symbolic Computation, 5:29–35, 1988.

T.W. Dubé.
The structure of polynomial ideals and Gröbner Bases.
SIAM J. Comp., 19:750–753, 1990.

Bibliography VI

M. England, R. Bradford, and J.H. Davenport.
Improving the Use of Equational Constraints in Cylindrical Algebraic Decomposition.
In D. Robertz, editor, *Proceedings ISSAC 2015*, pages 165–172, 2015.

M. England and J.H. Davenport.
The complexity of cylindrical algebraic decomposition with respect to polynomial degree.
In *Proceedings CASC 2016*, pages 172–192, 2016.

M. England.
Eliminating a Quantifier with SAGE/QEPCAD on Android.
Demonstration, 2014.

Z. Huang, M. England, D. Wilson, J. H. Davenport, and L. C. Paulson.
A comparison of three heuristics to choose the variable ordering for cylindrical algebraic decomposition.
ACM Communications in Computer Algebra, 48(3/4):121–123, 2015.

S. Haim and M. Heule.
Towards Ultra Rapid Restarts.
Technical Report Universities of New South Wales and Deflt, 2010.

M. Järvisalo, D. Le Berre, O. Roussel, and L. Simon.
The international SAT solver competitions.
AI Magazine, 33:89–92, 2012.

Bibliography VIII

J. Kollár.
Sharp effective nullstellensatz.
J.A.M.S., 1:963–975, 1988.

D. Kroening and O. Strichman.
Decision Procedures: An Algorithmic Point of View.
Springer, 2008.

M. Luby, A. Sinclair, and D. Zuckerman.
Optimal Speedup of Las Vegas algorithms.
Inf. Proc. Letters, 47:173–180, 1993.

Bibliography IX

S. McCallum.
An Improved Projection Operation for Cylindrical Algebraic Decomposition.
PhD thesis, University of Wisconsin-Madison Computer Science, 1984.

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

Bibliography X

S. McCallum.
On Propagation of Equational Constraints in CAD-Based Quantifier Elimination.
In B. Mourrain, editor, *Proceedings ISSAC 2001*, pages 223–230, 2001.

E. Mayr and A. Meyer.
The Complexity of the Word Problem for Commutative Semi-groups and Polynomial Ideals.
Adv. in Math., 46:305–329, 1982.

M.W. Moskewicz, Madigan.C.F., Y. Zhao, L. Zhang, and S. Malik.
Chaff: Engineering an Efficient SAT Solver.
In *Proceedings 38th Design Automation Conference*, 2001.

Bibliography XI

S. McCallum, A. Parusinski, and L. Paunescu.
Validity proof of Lazard's method for CAD construction.
https://arxiv.org/abs/1607.00264, 2016.

E.W. Mayr and S. Ritscher.
Dimension-dependent bounds for Gröbner bases of polynomial ideals.
J. Symbolic Comp., 49:78–94, 2013.

A. Platzer, J.-D. Quesel, and P. Rümmer.
Real world verification.
In *Proceedings CADE-22*, pages 485–501. ACM, 2009.

Bibliography XII

I. Spence.
Weakening Cardinality Constraints Creates Harder Satisfiability Benchmarks.
J. Exp. Algorithmics Article 1.4, 20, 2015.

D.J. Wilson, R.J. Bradford, and J.H. Davenport.
A Repository for CAD Examples.
ACM Communications in Computer Algebra 3, 46:67–69, 2012.

C.K. Yap.
A new lower bound construction for commutative Thue systems with applications.
J. Symbolic Comp., 12:1–27, 1991.