# SC-Square: Past Successes and Future Progress with Machine Learning

**Matthew England** (Coventry University, UK)

**The 6th SC-Square Workshop (SC$^2$ 2021)**
SIAM Conference on Applied Algebraic Geometry (AG21)
Texas, USA (Online) 19–20 August 2021

Author supported by EPSRC grant EP/T015748/1 (The DEWCAD Project).

# Outline

# Outline

# SC-Square Definition



**SC**: **S**atisfiability **C**hecking Community
Interested in algorithms to check satisfiability of logic problems with variables from a variety of mathematical domains. Implement solutions in SAT/SMT solvers.

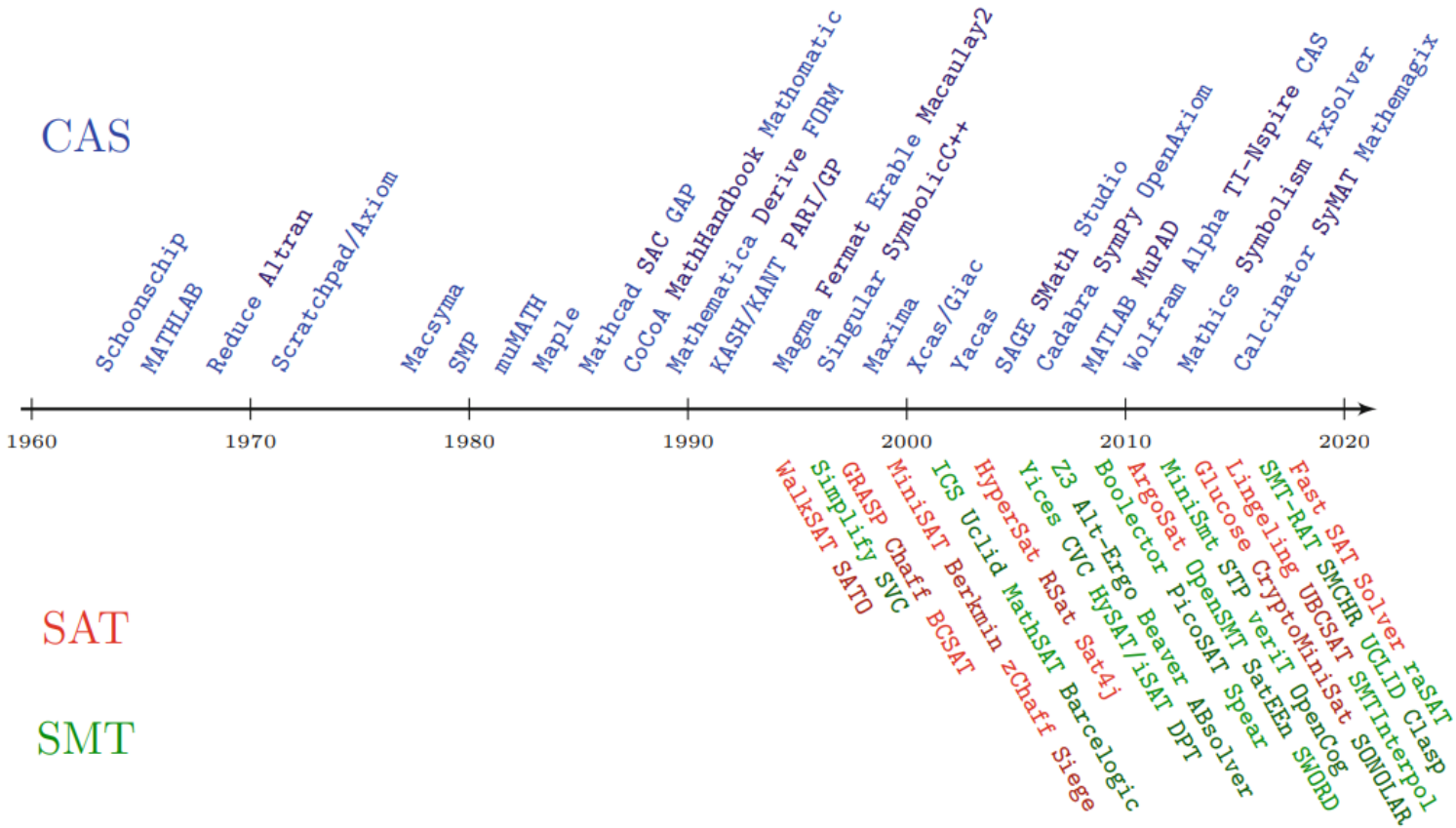
**SC**: **S**ymbolic **C**omputation Community
Interested in algorithms to perform algebraic computations, such as polynomial computations over real and complex numbers. Implement solutions in Computer Algebra Systems.

**SC$^2$**: SC-Square Community
Interested in both SCs above!

# Separate Histories



Until the last decade the two communities rarely interacted.




Timeline by E. Ábrahám Ábrahám [2015]

# Why Interact?



**Satisfiability Modulo Theories** (SMT) refers to the SAT decision problem where the atoms need not be Boolean variables but can instead be predicates over variables from other domains (e.g. arithmetics, data structures, uninterpreted functions).

A common framework for solving such problems separates out the reasoning in the logic (performed by a SAT solver) from the reasoning in the theory. The theory reasoning is done with algorithms / software from the domain of interest. For many theories the domain of interest is traditionally studied as Symbolic Computation.

In the opposite direction; the integration of SAT solvers into computer algebra systems can allow these tools to offer more powerful logical reasoning.

# Outline

# EU Project



The EU funded **The SC-Square Project** from 2016-2018 [Ábrahám et al., 2016]. The aim was to bridge the gap between the communities to produce individuals who can combine the knowledge and techniques of both fields to resolve problems currently beyond the scope of either.

The project funded new collaborations, new tool integrations, proposals on extensions to the SMT-LIB language standards, new collections of benchmarks, two summer schools (in 2017 and 2018) and this SC-Square Workshop Series.

Although the project finished, the workshop series continues, and the website and mailing lists remain active. www.sc-square.org

Volume 100 of the Journal of Symbolic Computation was an SC-Square themed special issue [Davenport et al., 2020].

# EU Project



The EU funded **The SC-Square Project** from 2016-2018 [Ábrahám et al., 2016]. The aim was to bridge the gap between the communities to produce individuals who can combine the knowledge and techniques of both fields to resolve problems currently beyond the scope of either.

The project funded new collaborations, new tool integrations, proposals on extensions to the SMT-LIB language standards, new collections of benchmarks, two summer schools (in 2017 and 2018) and this SC-Square Workshop Series.

Although the project finished, the workshop series continues, and the website and mailing lists remain active. www.sc-square.org

Volume 100 of the Journal of Symbolic Computation was an SC-Square themed special issue [Davenport et al., 2020].

# SC-Square Workshop Series (Slide 5/19)

2016 Timisoara, Romainia (as part of SYNASC 2016).

2017 Kaiserslautern, Germany (alongside ISSAC 2017).

2018 Oxford, UK (as part of FLoC 2018).

2019 Bern, Switzlerland (as part of SIAM AG19)

2020 Paris, France (online) (alongside IJCAR 2020)

2021 Texas, USA (online) (as part of SIAM AG21)

Take place as part of / alongside established conferences (alternate between computational algebra and logic).

Each year there are two chairs, one from each SC.

Now as many conferences outside the EU project as inside.
Funding obtained to support workshops in 2022-23.

# Non-linear real arithmetic I



Algorithms for working with systems of non-linear polynomials are a core topic for symbolic computation but also key for SMT in the [QF_]NRA logic. This area has seen a lot of work in SC$^2$:

NLSAT Algorithm of Jovanovic and de Moura [2012] re-purposed the theory of cylindrical algebraic decomposition by Collins [1975] to use in the MCSAT proof framework [de Moura and Jovanović, 2013].

The SMT-RAT Toolbox has implementations of a variety of computer algebra tools for use in SMT [Kremer and Ábrahám, 2018].

The algorithms must be adapted for the purpose (incremental by constraint and produce conflicting cores). E.g. Cylindrical Algebraic Decomposition [Kremer and Ábrahám, 2020], Groebner Bases [Junges et al., 2013].

# Non-linear real arithmetic I (Slide 6/19)

Algorithms for working with systems of non-linear polynomials are a core topic for symbolic computation but also key for SMT in the [QF_]NRA logic. This area has seen a lot of work in SC$^2$:

NLSAT Algorithm of Jovanovic and de Moura [2012] re-purposed the theory of cylindrical algebraic decomposition by Collins [1975] to use in the MCSAT proof framework [de Moura and Jovanović, 2013].

The SMT-RAT Toolbox has implementations of a variety of computer algebra tools for use in SMT [Kremer and Ábrahám, 2018].

The algorithms must be adapted for the purpose (incremental by constraint and produce conflicting cores). E.g. Cylindrical Algebraic Decomposition [Kremer and Ábrahám, 2020], Groebner Bases [Junges et al., 2013].

# Non-linear real arithmetic II



Conflict Driven Cylindrical Algebraic Coverings of Ábrahám et al. [2021] repurpose CAD technology like NLSAT.

Like NLSAT they produce coverings instead of a decomposition but unlike NLSAT this is a single algebraic procedure which may be used in a DPLL(T) framework.

The NuCAD algorithm of Brown [2015] is also inspired by NLSAT to construct one cell by cell using local information.

Can be used also for Quantifier Elimination [Brown, 2017].

Redlog+VeriT combination of computer algebra with heuristics based on interval constraint propagation and subtropical satisfiability [Fontaine et al., 2018].

Incremental Linearlization of techniques Cimatti et al. [2018].

# Non-linear real arithmetic II



Conflict Driven Cylindrical Algebraic Coverings of Ábrahám et al. [2021] repurpose CAD technology like NLSAT.

Like NLSAT they produce coverings instead of a decomposition but unlike NLSAT this is a single algebraic procedure which may be used in a DPLL(T) framework.

The NuCAD algorithm of Brown [2015] is also inspired by NLSAT to construct one cell by cell using local information.

Can be used also for Quantifier Elimination [Brown, 2017].

Redlog+VeriT combination of computer algebra with heuristics based on interval constraint propagation and subtropical satisfiability [Fontaine et al., 2018].

Incremental Linearlization of techniques Cimatti et al. [2018].

# Non-linear real arithmetic II



Conflict Driven Cylindrical Algebraic Coverings of Ábrahám et al. [2021] repurpose CAD technology like NLSAT.

Like NLSAT they produce coverings instead of a decomposition but unlike NLSAT this is a single algebraic procedure which may be used in a DPLL(T) framework.

The NuCAD algorithm of Brown [2015] is also inspired by NLSAT to construct one cell by cell using local information.

Can be used also for Quantifier Elimination [Brown, 2017].

Redlog+VeriT combination of computer algebra with heuristics based on interval constraint propagation and subtropical satisfiability [Fontaine et al., 2018].

Incremental Linearlization of techniques Cimatti et al. [2018].

# Other SC-Square Successes I



**Maple** can now read to and from SMT-LIB [Forrest, 2017] and ships with the both the Z3 and MapleSAT.

**CoCoALib** C++ Library that underpins CoCoA [Abbott and Bigatti, 2014] used by MathSAT and SMT-RAT.

**MathCheck** has made significant progress on a variety of combinatorics problems from a combination of SAT-solvers and computer algebra, enumerating new cases and verifying conjectures. E.g. Williamson Matrices [Bright et al., 2020]; Golay Pairs [Bright et al., 2018]; Good Matrices [Bright et al., 2019].

# Other SC-Square Successes I



**Maple** can now read to and from SMT-LIB [Forrest, 2017] and ships with the both the Z3 and MapleSAT.

**CoCoALib** C++ Library that underpins CoCoA [Abbott and Bigatti, 2014] used by MathSAT and SMT-RAT.

**MathCheck** has made significant progress on a variety of combinatorics problems from a combination of SAT-solvers and computer algebra, enumerating new cases and verifying conjectures. E.g. Williamson Matrices [Bright et al., 2020]; Golay Pairs [Bright et al., 2018]; Good Matrices [Bright et al., 2019].

# Other SC-Square Successes II



**Boolean SAT** has also benefited from symbolic computation via Boolean Groebner Bases and parallel computation on both the conjunctive and algebraic normal forms of a problem [Horáček and Kreuzer, 2020].

**Circuit Verification** Algebraic techniques key for circuit verification [Kaufmann, 2021], and in combination with SAT-solvers [Kaufmann et al., 2019].

**Proofs** work on natural style proofs [Jebelean, 2018], [Ábrahám et al., 2020].

**New Applications** in Economics [Mulligan et al., 2018], Dynamic Geometry [Vajda and Kovács, 2020], knot theory [Lisitsa and Vernitski, 2017, Meesum and Prathamesh, 2018].

# Other SC-Square Successes II



**Boolean SAT** has also benefited from symbolic computation via Boolean Groebner Bases and parallel computation on both the conjunctive and algebraic normal forms of a problem [Horáček and Kreuzer, 2020].

**Circuit Verification** Algebraic techniques key for circuit verification [Kaufmann, 2021], and in combination with SAT-solvers [Kaufmann et al., 2019].

**Proofs** work on natural style proofs [Jebelean, 2018], [Ábrahám et al., 2020].

**New Applications** in Economics [Mulligan et al., 2018], Dynamic Geometry [Vajda and Kovács, 2020], knot theory [Lisitsa and Vernitski, 2017, Meesum and Prathamesh, 2018].

# Outline

# ML for SC$^2$ Technology? (Slide 10/19)

**Machine Learning** tools use statistics and large quantities of data to learn how to perform tasks not explicitly programmed.
They offer only probabilistic guidance, but both SC communities produce software that prizes exact results. Is there interest for SC$^2$?

**Yes**. ML techniques can make non-critical choices and guide searches. Used this way, the ML performance has no affect on the mathematical correctness of the final result, but can make a substantial contribution to both computational efficiency and presentation of the end result.

Often, ML can make such choices better than the user, developer, or currently employed human-made heuristics.

Note that this approach differs from say the Facebook Research work of Lample and Charton (2020) which used ML to predict the end result of a mathematical process directly.

# ML for SC$^2$ Technology? (Slide 10/19)

**Machine Learning** tools use statistics and large quantities of data to learn how to perform tasks not explicitly programmed.
They offer only probabilistic guidance, but both SC communities produce software that prizes exact results. Is there interest for SC$^2$?

**Yes**. ML techniques can make non-critical choices and guide searches. Used this way, the ML performance has no affect on the mathematical correctness of the final result, but can make a substantial contribution to both computational efficiency and presentation of the end result.

Often, ML can make such choices better than the user, developer, or currently employed human-made heuristics.

Note that this approach differs from say the Facebook Research work of Lample and Charton (2020) which used ML to predict the end result of a mathematical process directly.

# ML for SC$^2$ Technology? (Slide 10/19)

**Machine Learning** tools use statistics and large quantities of data to learn how to perform tasks not explicitly programmed.
They offer only probabilistic guidance, but both SC communities produce software that prizes exact results. Is there interest for SC$^2$?

**Yes**. ML techniques can make non-critical choices and guide searches. Used this way, the ML performance has no affect on the mathematical correctness of the final result, but can make a substantial contribution to both computational efficiency and presentation of the end result.

Often, ML can make such choices better than the user, developer, or currently employed human-made heuristics.

Note that this approach differs from say the Facebook Research work of Lample and Charton (2020) which used ML to predict the end result of a mathematical process directly.

# Examples from the literature (Slide 11/19)

- Xu et al. [2008] won SAT competitions with their portfolio solver which selected the best SAT solver for the instance.
- Liang et al. [2016] used a machine learnt branching heuristic in their MapleSAT solver for formulae in Boolean logic.
- Wu [2017] used a regression model to predict the satisfiability of formulae after fixing some variables; to select initial values.

---

- Kuipers et al. [2015] used a Monte-Carlo tree search to find the representation of polynomials that are most efficient to evaluate in their CAS.
- Huang et al. [2016] used a support vector machine to decide whether or not to precondition CAD with Gröbner Bases.
- Brown and Daves [2020] used a neural network to select the order of constraints to process in NuCAD.

# Examples from the literature (Slide 11/19)

- Xu et al. [2008] won SAT competitions with their portfolio solver which selected the best SAT solver for the instance.
- Liang et al. [2016] used a machine learnt branching heuristic in their MapleSAT solver for formulae in Boolean logic.
- Wu [2017] used a regression model to predict the satisfiability of formulae after fixing some variables; to select initial values.

---

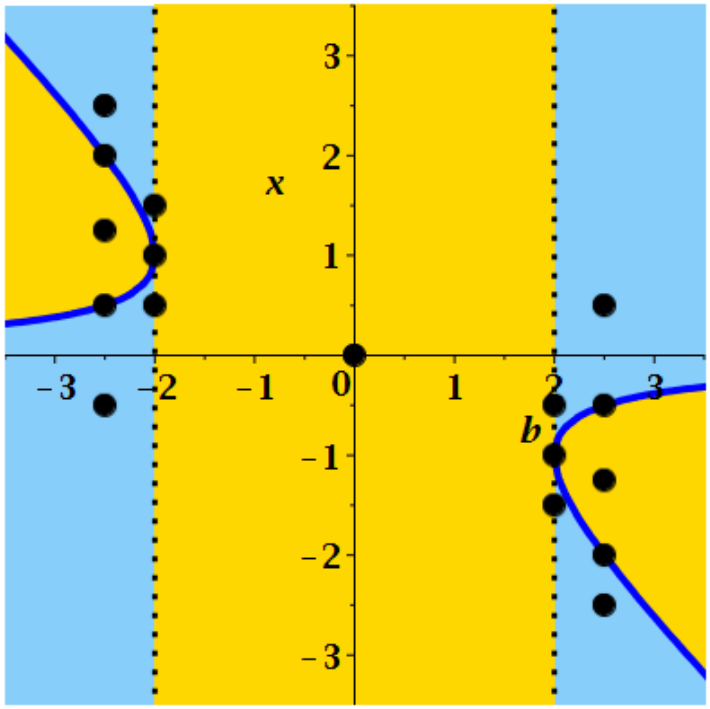
- Kuipers et al. [2015] used a Monte-Carlo tree search to find the representation of polynomials that are most efficient to evaluate in their CAS.
- Huang et al. [2016] used a support vector machine to decide whether or not to precondition CAD with Gröbner Bases.
- Brown and Daves [2020] used a neural network to select the order of constraints to process in NuCAD.

# Outline

# CAD and QE



A CAD is a decomposition of $\mathbb{R}^n$ that may be algorithmically produced from a set of polynomials such that each polynomial has invariant sign on each cell. E.g. CAD below for $\{x^2 + bx + 1\}$. The invariance means we can answer questions on polynomials by testing a finite number of sample points.



# CAD and QE

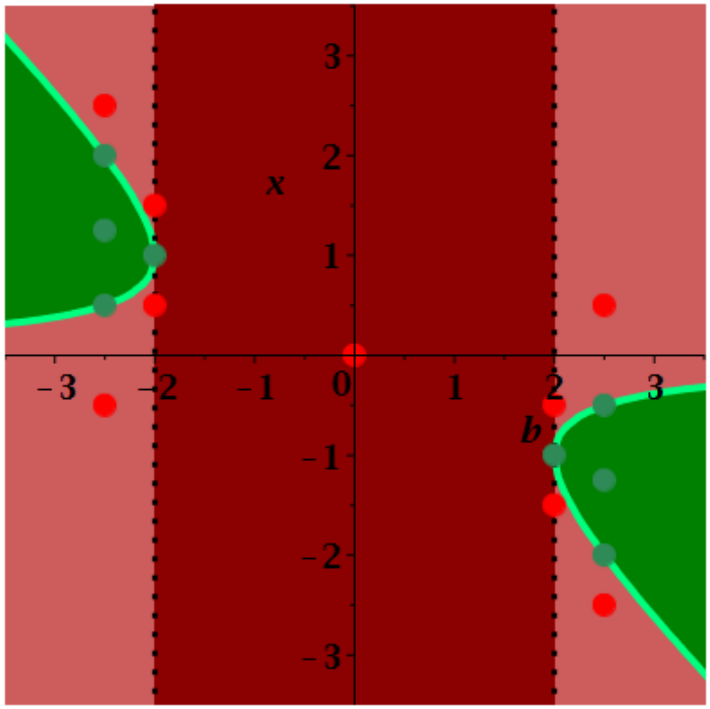


A CAD is a decomposition of $\mathbb{R}^n$ that may be algorithmically produced from a set of polynomials such that each polynomial has invariant sign on each cell. E.g. CAD below for $\{x^2 + bx + 1\}$. The invariance means we can answer questions on polynomials by testing a finite number of sample points.

A key application is quantifier elimination. E.g.

$$\exists x, x^2 + bx + 1 \leq 0$$

Tag cells in the CAD true or false according to polynomial sign.



# CAD and QE

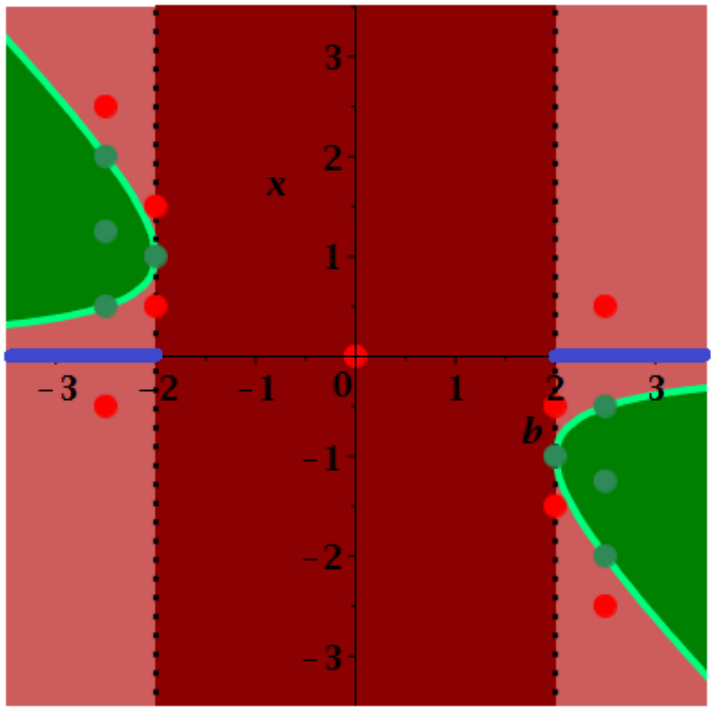


A CAD is a decomposition of $\mathbb{R}^n$ that may be algorithmically produced from a set of polynomials such that each polynomial has invariant sign on each cell. E.g. CAD below for $\{x^2 + bx + 1\}$. The invariance means we can answer questions on polynomials by testing a finite number of sample points.

A key application is quantifier elimination. E.g.

$$\exists x, x^2 + bx + 1 \leq 0$$

Tag cells in the CAD true or false according to polynomial sign. Then project true cells and take disjunction. $\implies b \leq -2 \vee b \geq 2$.



# CAD Variable Ordering Choice



CAD has doubly exponential complexity in the number of variables! The complexity is felt in practice. But careful optimisations and pre-processing can *push back the doubly exponential wall* and bring new applications in scope.

A particularly important optimisation is the variable ordering. For QE one must order variables as they are quantified; but there is no restriction on free variables and adjacent quantifiers of the same type may be swapped.

The ordering is well known to dramatically affect both the number of cells and the time to compute them.

# CAD Variable Ordering Choice



CAD has doubly exponential complexity in the number of variables! The complexity is felt in practice. But careful optimisations and pre-processing can *push back the doubly exponential wall* and bring new applications in scope.

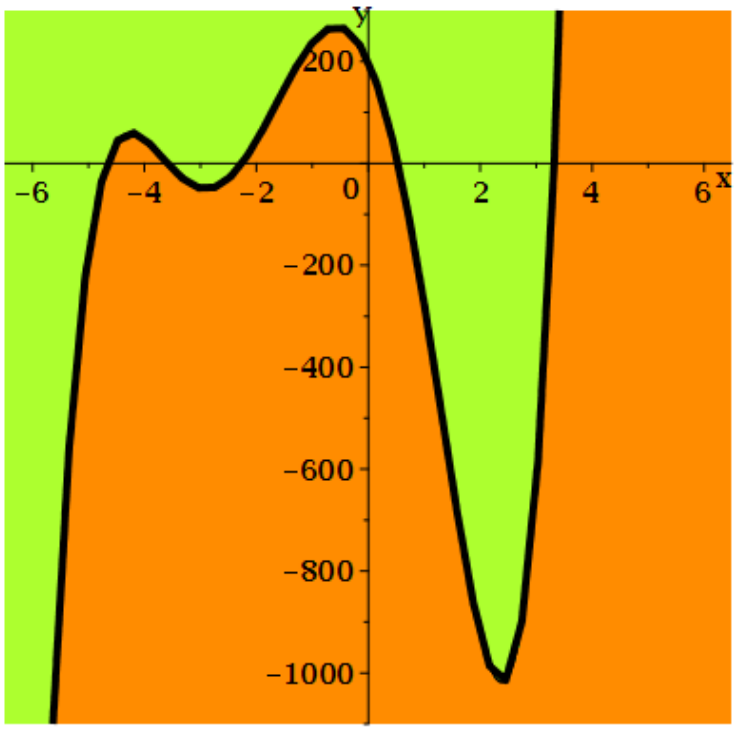
A particularly important optimisation is the variable ordering. For QE one must order variables as they are quantified; but there is no restriction on free variables and adjacent quantifiers of the same type may be swapped.

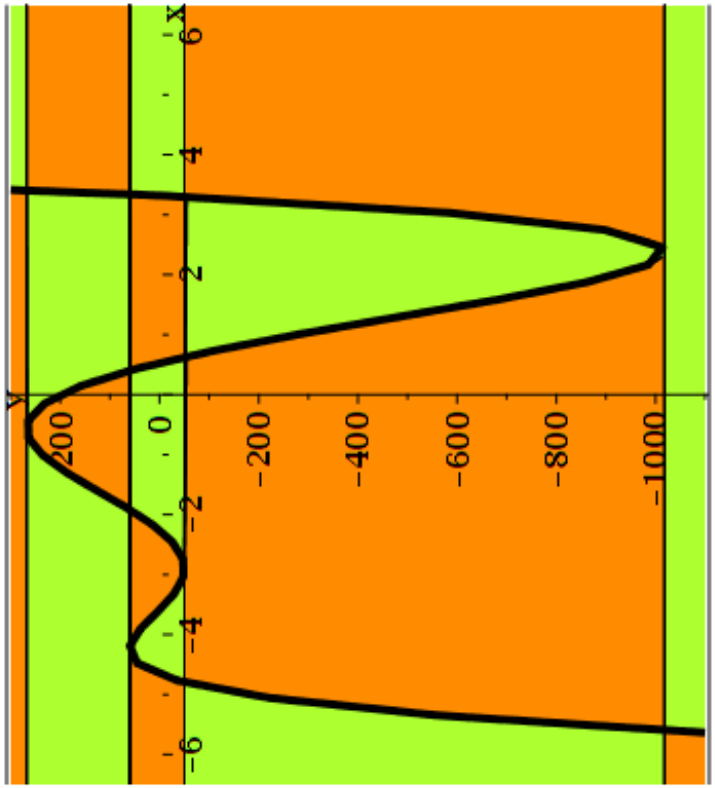
The ordering is well known to dramatically affect both the number of cells and the time to compute them.

# CAD Variable Ordering Example



CAD for polynomial $y - 3x^5 + 20x^4 - 10x^3 - 240x^2 - 250x + 200$.
With $y \succ x$ a sign-invariant CAD has 3 cells, with $y \prec x$ it is 59.



# Prior Work on CAD Variable Ordering (Slide 15/19)

There have been a succession of human designed heuristics: [Brown, 2004], [Dolzmann et al., 2004], [Bradford et al., 2013], [Wilson et al., 2014]. They are increasingly expensive and have variable performance: none of them are perfect.

Huang et al. [2014] used a support vector machine to pick which of the above heuristics to follow: this meta-heuristic did better than any one heuristic individually. The experiments also identified substantial subsets on which each human-designed heuristic was superior to the others.

Recently Li et al. [2021] demonstrated that CAD can preserve chordality. This informs a heuristic to preserve sparsity.

# Prior Work on CAD Variable Ordering (Slide 15/19)

There have been a succession of human designed heuristics: [Brown, 2004], [Dolzmann et al., 2004], [Bradford et al., 2013], [Wilson et al., 2014]. They are increasingly expensive and have variable performance: none of them are perfect.

Huang et al. [2014] used a support vector machine to pick which of the above heuristics to follow: this meta-heuristic did better than any one heuristic individually. The experiments also identified substantial subsets on which each human-designed heuristic was superior to the others.

Recently Li et al. [2021] demonstrated that CAD can preserve chordality. This informs a heuristic to preserve sparsity.

# Our Recent Work I



EPSRC Project EP/R019622/1 (2018-2020) *Embedding Machine Learning within Quantifier Elimination Procedures* further investigated the use of ML for CAD variable ordering choice.

- Experimented with several ML classifiers in `sklearn`: they all do better than the existing human-made heuristics [England and Florescu, 2019].
- New approach to generate features of the input polynomials [Florescu and England, 2019].

ML classifiers are trained not on input polynomials themselves but vectors of floating point numbers derived from them (the **features**). Prior work used features inspired by Brown's heuristic. The new approach enumerated all (appropriate) combinations of basic (i.e. cheap) functions used by Brown and removes redundancies to leaves 78 features in 3-variable and 105 in 4.

# Our Recent Work I



EPSRC Project EP/R019622/1 (2018-2020) *Embedding Machine Learning within Quantifier Elimination Procedures* further investigated the use of ML for CAD variable ordering choice.

- Experimented with several ML classifiers in `sklearn`: they all do better than the existing human-made heuristics [England and Florescu, 2019].
- New approach to generate features of the input polynomials [Florescu and England, 2019].

ML classifiers are trained not on input polynomials themselves but vectors of floating point numbers derived from them (the **features**). Prior work used features inspired by Brown's heuristic. The new approach enumerated all (appropriate) combinations of basic (i.e. cheap) functions used by Brown and removes redundancies to leaves 78 features in 3-variable and 105 in 4.

# Our Recent Work II



- Proposed a more suitable measure of ML classifier accuracy and used this to write an improved method for cross-validation hyper-parameter selection in `sklearn` [Florescu and England, 2020a].

Classifiers usually seek the optimal (ordering which used least resources) but this gives no difference between *almost optimal* and *terrible*. Instead define accurate if close to the optimal and minimise directly on computation time during cross-validation.

- Combined into a software pipeline [Florescu and England, 2020b] and released for free as a Zenodo repository: https://doi.org/10.5281/zenodo.3731703

# Our Recent Work II



- Proposed a more suitable measure of ML classifier accuracy and used this to write an improved method for cross-validation hyper-parameter selection in `sklearn` [Florescu and England, 2020a].

Classifiers usually seek the optimal (ordering which used least resources) but this gives no difference between *almost optimal* and *terrible*. Instead define accurate if close to the optimal and minimise directly on computation time during cross-validation.

- Combined into a software pipeline [Florescu and England, 2020b] and released for free as a Zenodo repository: https://doi.org/10.5281/zenodo.3731703

# Successes and Challenges



Experiments showed that the new classifiers tried did better than SVM; that all classifiers benefited from the additional features; that all classifiers benefited from the new cross validation procedure (with and without the additional features).

On our 3-variable dataset: human made heuristics achieved times 27% above the minimum; ML achieved times 6% above.

On our 4-variable dataset: human made heuristics achieved times 98% above the minimum; ML achieved times 67% above.

**Challenges:**

- Dealing with exponential growth in the number of choices.
- Finding datasets that are sufficiently varied, and representative of all use-cases.
- Finding datasets large enough for deep learning.

# Successes and Challenges



Experiments showed that the new classifiers tried did better than SVM; that all classifiers benefited from the additional features; that all classifiers benefited from the new cross validation procedure (with and without the additional features).

On our 3-variable dataset: human made heuristics achieved times 27% above the minimum; ML achieved times 6% above.

On our 4-variable dataset: human made heuristics achieved times 98% above the minimum; ML achieved times 67% above.

**Challenges:**

- Dealing with exponential growth in the number of choices.
- Finding datasets that are sufficiently varied, and representative of all use-cases.
- Finding datasets large enough for deep learning.

# Contact Details and Advert



## Contact Details

Matthew.England@coventry.ac.uk

https://matthewengland.coventry.domains/

## Advert

**Fully Funded PhD Position** available at Coventry to work on Machine Learning to Improve Symbolic Integration and Symbolic Simplification. Sponsored by Maplesoft.
https://tinyurl.com/3exmk9vk
**Deadline to Apply:** 13th September 2021
**Interviews and Decision:** End September
**PhD Start:** Jan 2022

# Bibliography I

J. Abbott and A.M. Bigatti. What is new in CoCoA? In H. Hong and C. Yap, editors, *Mathematical Software – ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 352–358. Springer Heidelberg, 2014. URL https://doi.org/10.1007/978-3-662-44199-2_55.

E. Ábrahám. Building bridges between symbolic computation and satisfiability checking. In *Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 1–6. ACM, 2015. URL https://doi.org/10.1145/2755996.2756636.

# Bibliography II

E. Ábrahám, J. Abbott, B. Becker, A.M. Bigatti, M. Brain, B. Buchberger, A. Cimatti, J.H. Davenport, M. England, P. Fontaine, S. Forrest, A. Griggio, D. Kroening, W.M. Seiler, and T. Sturm. SC$^2$: Satisfiability checking meets symbolic computation. In M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, editors, *Intelligent Computer Mathematics: Proceedings CICM 2016*, volume 9791 of *Lecture Notes in Computer Science*, pages 28–43. Springer International Publishing, 2016. URL https://doi.org/10.1007/978-3-319-42547-4_3.

# Bibliography III

E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z. Tonks. New opportunities for the formal proof of computational real geometry? In P. Fontaine, K. Korovin, I.S. Kotsireas, P. Rümmer, and S. Tourret, editors, *Proceedings of the 5th Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2020)*, number 2752 in CEUR Workshop Proceedings, pages 178–188, 2020. URL http://ceur-ws.org/Vol-2752/.

E. Ábrahám, J.H. Davenport, M.England, and G. Kremer. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. *Journal of Logical and Algebraic Methods in Programming*, 119: 100633, 2021. URL https://doi.org/10.1016/j.jlamp.2020.100633.

# Bibliography IV

R. Bradford, J.H. Davenport, M. England, and D. Wilson. Optimising problem formulations for cylindrical algebraic decomposition. In J. Carette, D. Aspinall, C. Lange, P. Sojka, and W. Windsteiger, editors, *Intelligent Computer Mathematics*, volume 7961 of *Lecture Notes in Computer Science*, pages 19–34. Springer Berlin Heidelberg, 2013. URL http://dx.doi.org/10.1007/978-3-642-39320-4_2.

C. Bright, I. Kotsireas, A. Heinle, and V. Ganesh. Enumeration of complex Golay pairs via programmatic sat. In *Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '18, pages 111–118. ACM, 2018. URL https://doi.org/10.1145/3208976.3209006.

# Bibliography V

C. Bright, D.Z. Doković, I. Kotsireas, and V. Ganesh. A SAT+CAS approach to finding good matrices: New examples and counterexamples. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. URL https://doi.org/10.1609/aaai.v33i01.33011435.

C. Bright, I. Kotsireas, and V. Ganesh. Applying computer algebra systems with SAT solvers to the Williamson conjecture. *Journal of Symbolic Computation*, 100:187–209, 2020. URL https://doi.org/10.1016/j.jsc.2019.07.024.

C. Brown. Projection and quantifier elimination using non-uniform cylindrical algebraic decomposition. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 53–60. ACM, 2017. URL https://doi.org/10.1145/3087604.3087651.

# Bibliography VI

C.W. Brown. Companion to the tutorial: Cylindrical algebraic decomposition, presented at ISSAC '04. URL http://www.usna.edu/Users/cs/wcbrown/research/ISSAC04/handout.pdf, 2004.

C.W. Brown. Open non-uniform cylindrical algebraic decompositions. In *Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 85–92. ACM, 2015. URL https://doi.org/10.1145/2755996.2756654.

C.W. Brown and G.C. Daves. Applying machine learning to heuristics for real polynomial constraint solving. In A. Bigatti, J. Carette, J.H. Davenport, M. Joswig, and T. de Wolff, editors, *Mathematical Software – ICMS 2020*, volume 12097 of *Lecture Notes in Computer Science*, pages 292–301. Springer International Publishing, 2020. URL https://doi.org/10.1007/978-3-030-52200-1_29.

# Bibliography VII

B. Caviness and J. Johnson. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Texts & Monographs in Symbolic Computation. Springer-Verlag, 1998. URL https://doi.org/10.1007/978-3-7091-9459-1.

A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Incremental linearization: A practical approach to satisfiability modulo nonlinear arithmetic and transcendental functions. In *20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 19–26, 2018. URL http://doi.org/10.1109/SYNASC.2018.00016.

# Bibliography VIII

G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 134–183. Springer-Verlag (reprinted in the collection Caviness and Johnson [1998]), 1975. URL https://doi.org/10.1007/3-540-07407-4_17.

J.H. Davenport, M. England, A. Griggio, T. Sturm, and C. Tinelli. Symbolic computation and satisfiability checking: Editorial. *Journal of Symbolic Computation*, 100:1–10, 2020. URL https://doi.org/10.1016/j.jsc.2019.07.017.

L. de Moura and D. Jovanović. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Verification, Model Checking, and Abstract Interpretation (Proc. VMCAI 2013)*, pages 1–12. Springer Berlin Heidelberg, 2013. URL https://link.springer.com/bookseries/558.

# Bibliography IX

A. Dolzmann, A. Seidl, and T. Sturm. Efficient projection orders for CAD. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, ISSAC '04, pages 111–118. ACM, 2004. URL https://doi.org/10.1145/1005285.1005303.

M. England and D. Florescu. Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition. In C. Kaliszyk, E. Brady, A. Kohlhase, and C.C. Sacerdoti, editors, *Intelligent Computer Mathematics*, volume 11617 of *Lecture Notes in Computer Science*, pages 93–108. Springer International Publishing, 2019. URL https://doi.org/10.1007/978-3-030-23250-4_7.

# Bibliography X

D. Florescu and M. England. Algorithmically generating new algebraic features of polynomial systems for machine learning. In J. Abbott and A. Griggio, editors, *Proceedings of the 4th Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2019)*, number 2460 in CEUR Workshop Proceedings, 2019. URL http://ceur-ws.org/Vol-2460/.

D. Florescu and M. England. Improved cross-validation for classifiers that make algorithmic choices to minimise runtime without compromising output correctness. In D. Slamanig, E. Tsigaridas, and Z. Zafeirakopoulos, editors, *Mathematical Aspects of Computer and Information Sciences (Proc. MACIS '19)*, volume 11989 of *Lecture Notes in Computer Science*, pages 341–356. Springer International Publishing, 2020a. URL https://doi.org/10.1007/978-3-030-43120-4_27.

# Bibliography XI

D. Florescu and M. England. A machine learning based software pipeline to pick the variable ordering for algorithms with polynomial inputs. In A. Bigatti, J. Carette, J.H. Davenport, M. Joswig, and T. de Wolff, editors, *Mathematical Software – ICMS 2020*, volume 12097 of *Lecture Notes in Computer Science*, pages 302–322. Springer International Publishing, 2020b. URL https://doi.org/10.1007/978-3-030-52200-1_30.

P. Fontaine, M. Ogawa, T. Sturm, V. Khanh To, and X. Tung Vu. Wrapping computer algebra is surprisingly successful for non-linear SMT. In A.M. Bigatti and M. Brain, editors, *Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2018)*, number 2189 in CEUR Workshop Proceedings, pages 110–117, 2018. URL http://ceur-ws.org/Vol-2189/.

# Bibliography XII

S.A. Forrest. Integration of smt-lib support into maple. In M. England and V. Ganesh, editors, *Proceedings of the 2nd International Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2017)*, number 1974 in CEUR Workshop Proceedings, 2017. URL http://ceur-ws.org/Vol-1974/.

J. Horáček and M. Kreuzer. On conversions from CNF to ANF. *Journal of Symbolic Computation*, 100:164–186, 2020. URL https://doi.org/10.1016/j.jsc.2019.07.023.

Z. Huang, M. England, D. Wilson, J.H. Davenport, L. Paulson, and J. Bridge. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In S.M. Watt, J.H. Davenport, A.P. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics*, volume 8543 of *Lecture Notes in Artificial*

# Bibliography XIII

*Intelligence*, pages 92–107. Springer International, 2014. URL http://dx.doi.org/10.1007/978-3-319-08434-3_8.

Z. Huang, M. England, J.H. Davenport, and L. Paulson. Using machine learning to decide when to precondition cylindrical algebraic decomposition with Groebner bases. In *18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '16)*, pages 45–52. IEEE, 2016. URL https://doi.org/10.1109/SYNASC.2016.020.

T. Jebelean. Techniques for natural-style proofs in elementary analysis (work in progress). In A.M. Bigatti and M. Brain, editors, *Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2018)*, number 2189 in CEUR Workshop Proceedings, pages 122–131, 2018. URL http://ceur-ws.org/Vol-2189/.

# Bibliography XIV

D. Jovanovic and L. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning: 6th International Joint Conference (IJCAR)*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012. URL https://doi.org/10.1007/978-3-642-31365-3_27.

S. Junges, U. Loup, F. Corzilius, and E. Ábrahám. On Gröbner bases in the context of Satisfiability-Modulo-Theories solving over the real numbers. In T. Muntean, D. Poulakis, and R. Rolland, editors, *Algebraic Informatics*, volume 8080 of *Lecture Notes in Computer Science*, pages 186–198. Springer Berlin Heidelberg, 2013. URL https://doi.org/10.1007/978-3-642-40663-8_18.

# Bibliography XV

D. Kaufmann. Formal verification of integer multiplier circuits using algebraic reasoning: A survey. In R. Drechsler and D. Große, editors, *Recent Findings in Boolean Techniques: Selected Papers from the 14th International Workshop on Boolean Problems*, pages 1–27. Springer International Publishing, 2021. URL https://doi.org/10.1007/978-3-030-68071-8_1.

D. Kaufmann, A. Biere, and M. Kauers. Verifying large multipliers by combining SAT and computer algebra. In *Formal Methods in Computer Aided Design (FMCAD 2019)*, pages 28–36. IEEE, 2019. URL https://doi.org/10.23919/FMCAD.2019.8894250.

G. Kremer and E. Ábrahám. Modular strategic SMT solving with SMT-RAT. *Acta Universitatis Sapientiae, Informatica*, 10(1): 5–25, 2018. URL http://dx.doi.org/10.2478/ausi-2018-0001.

# Bibliography XVI

G. Kremer and E. Ábrahám. Fully incremental CAD. *Journal of Symbolic Computation*, 100:11–37, 2020. URL https://doi.org/10.1016/j.jsc.2019.07.018.

J. Kuipers, T. Ueda, and J.A.M. Vermaseren. Code optimization in FORM. *Computer Physics Communications*, 189:1–19, 2015. URL https://doi.org/10.1016/j.cpc.2014.08.008.

H. Li, B. Xia, H. Zhang, and T. Zheng. Choosing the variable ordering for cylindrical algebraic decomposition via exploiting chordal structure. In *Proceedings of the 2021 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '21, pages 281–288. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3452143.3465520.

# Bibliography XVII

J.H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki. Learning rate based branching heuristic for SAT solvers. In N. Creignou and D. Le Berre, editors, *Theory and Applications of Satisfiability Testing – SAT 2016*, volume 9710 of *Lecture Notes in Computer Science*, pages 123–140. Springer International Publishing, 2016. URL https://doi.org/10.1007/978-3-319-40970-2_9.

A. Lisitsa and A. Vernitski. Automated reasoning for knot semigroups and $\pi$-orbifold groups of knots. In J. Blömer, I.S. Kotsireas, T. Kutsia, and D.E. Simos, editors, *Mathematical Aspects of Computer and Information Sciences (Proc. MACIS '17)*, volume 10693 of *Lecture Notes in Computer Science*, pages 3–18. Springer International Publishing, 2017. URL https://doi.org/10.1007/978-3-319-72453-9_1.

# Bibliography XVIII

S.M. Meesum and T.V.H. Prathamesh. Unknot recognition through quantifier elimination. In A.M. Bigatti and M. Brain, editors, *Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2018)*, number 2189 in CEUR Workshop Proceedings, pages 77–87, 2018. URL http://ceur-ws.org/Vol-2189/.

C. Mulligan, R. Bradford, J.H. Davenport, M. England, and Z. Tonks. Non-linear real arithmetic benchmarks derived from automated reasoning in economics. In A.M. Bigatti and M. Brain, editors, *Proceedings of the 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2018)*, number 2189 in CEUR Workshop Proceedings, pages 48–60, 2018. URL http://ceur-ws.org/Vol-2189/.

# Bibliography XIX

R. Vajda and Z. Kovács. GeoGebra and the realgeom reasoning tool. In P. Fontaine, K. Korovin, I.S. Kotsireas, P. Rümmer, and S. Tourret, editors, *Proceedings of the 5th Workshop on Satisfiability Checking and Symbolic Computation (SC$^2$ 2020)*, number 2752 in CEUR Workshop Proceedings, pages 204–219, 2020. URL http://ceur-ws.org/Vol-2752/.

D. Wilson, M. England, J.H. Davenport, and R. Bradford. Using the distribution of cells by dimension in a cylindrical algebraic decomposition. In *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC '14, pages 53–60. IEEE, 2014. URL http://dx.doi.org/10.1109/SYNASC.2014.15.

# Bibliography XX

Haoze Wu. Improving SAT-solving with machine learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 787–788. ACM, 2017. URL https://doi.org/10.1145/3017680.3022464.

L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal Of Artificial Intelligence Research*, 32:565–606, 2008. URL https://doi.org/10.1613/jair.2490.