Proving UNSAT in SMT: Quantifier Free Non-Linear Real Arithmetic

Erika Ábrahám[1] James H. Davenport[2]
Matthew England[3] Gereon Kremer[4]

RWTH Aachen University, 52062 Aachen, Germany
abraham@cs.rwth-aachen.de

University of Bath, Bath BA2 7AY, United Kingdom
J.H.Davenport@bath.ac.uk (EPSRC EP/T015748/1)

Coventry University, Coventry, CV1 5FB, United Kingdom
Matthew.England@coventry.ac.uk (EPSRC EP/T015713/1)

Stanford University, Stanford, California 94305, United States
gkremer@cs.stanford.edu

16 July 2021 at ARCADE 2021

Plan of Talk

1. UNSAT in SAT-Solving Contests
2. UNSAT in SMT: Prior work
3. The `QF_NRA` (Quantifier-Free Nonlinear Real Arithmetic) challenges
4. `QF_NRA` methodologies
5. CDCAC — Conflict-Driven Cylindrical Algebraic Coverings
6. Way forward?

UNSAT in SAT-Solving Contests

SAT is easy to demonstrate — give the assignment

2013 Contestants in UNSAT track must also return proofs of UNSAT

2020 All (sequential?) tracks require proofs of UNSAT

Proofs (sometimes $>$100GB) are verified offline [HJS18]

DRAT is the standard format (although there are some flavours [RB19])

UNSAT in SMT: prior work

Good idea! SMT-LIB Language (v2.6) [BFT16] specifies API commands for requesting and inspecting proofs from solvers

but sets no requirements on the form those proofs take

[BdF15] summarises some of the requirements, challenges and various approaches taken to proofs in SMT

LFSC [SRT$^+$12]: Logical Framework with Side Conditions

veriT [BBFF20]: linear arithmetic; proofs verifiable in Isabelle/HOL and Coq

The QF_NRA Challenges

[BdF15] "since in SMT the propositional and theory reasoning are not strongly mixed, an SMT proof can be an interleaving of SAT proofs and theory reasoning proofs"

[BdF15] "the main challenge of proof production is keeping enough information to produce proofs"

QF_NRA Actually providing the theory proofs can be a challenge

This is the main topic of this talk.

The QF_NRA Methodology

Any SMT solver which claims to tackle this logic completely relies in some way on the theory of Cylindrical Algebraic Decomposition (CAD) [Col75].

1. Decompose R^n into a finite number of *disjoint* regions, on each of which the truth of the constraints is constant.
2. Take a sample point in each region.
   * In practice the sample points are built at the same time as the regions.
3. Now we have a finite set of theory values and the SMT methodology applies.
   * In practice, we will try to merge the phases, and do the decomposition incrementally [KÁ20].

How do we formally prove this decomposition? Attempts to prove the correctness [Mah06, Mah07, CM10, CM12] have failed, essentially on the topology.

More QF_NRA: NLSAT etc.

- [JdM12] `nlsat`: Allow the Boolean model and the theory model to develop simultaneously.
  - $\pm$ very powerful, but contradicts "not strongly mixed": not obvious how to construct the proof.
- [dMJ13] Generalises this to "the model constructing satisfiability calculus (mcSAT) framework".
  - $+$ The search for a Boolean model and a theory model are mutually guided by each other away from unsatisfiable regions.
  1) Boolean conflicts are generalised using propositional resolution
  2) Theory conflicts: generalise the sample point to a region containing the point on which the same constraints fail for the same reason.
  - – Also contradicts "not strongly mixed":

CDCAC: Cylindrical Algebraic Coverings [ÁDMK21]

- Essentially, a depth first search is performed according to the theory variables.
- Conflicts over particular assignments are generalised to cells until a covering of a dimension is obtained,
- and then this covering is generalised to a cell in the dimension below.
- And repeat until R^1 is covered.

Like CAD
: Decompose R^n into a finite number of ~~*disjoint*~~ regions, on each of which the truth of the constraints is constant.

Unlike NLSAT
: Build the cells cylindrically, so the proof that they're a covering is easy.

Like both
: Correctness of the algorithm relies on CAD theory, so beyond current proof theory to prove

Why Cylindrical Algebraic Coverings? [ÁDE+20]

Unike CAD (more like NLSAT) each cell is built to generalise a specific conflict, so has a *local* rationale.

[ÁDE+20] The trace of a CDCAC computation appears far closer to a human derived proof than any of the other algorithms.

Hence There's another option: verifying a *specific* CDCAC computation, rather than the algorithm.

1. Verify that the set of cells is a covering (recursively in dimension).
2. For each cell, verify that the sample point is a conflict which extends over the whole cell.

Hope that these proofs are easier to do in a formal system (no topology).

Fits the SMT-with-proof methodology.

QF_NRA: CAD is not the only option

Above "Any complete solver relies on CAD".

True but many incomplete methods work very effectively, notably Virtual Term Substitution (VTS) [Ton20].

VTS transforms a CAD problem in $x_1, \ldots, x_n$, where x_n is linear or quadratic, into a problem in $x_1, \ldots, x_{n-1}$.

VTS And if x_{n-1} is linear or quadratic, repeat ....

CAD When this runs out of steam

Unclear (to say the least) how this would fit into the SMT-with-proof methodology.

Also other transformations: $\Phi(x_1, \ldots, x_{n-1}, x_n^3)$ is SAT iff $\Phi(x_1, \ldots, x_{n-1}, x_n')$ is SAT, but this can reduce the number of cells required doubly-exponentially in n.

Way forward

1a. Work with theorem provers to clarify the "hope" that these proofs are easy in a formal system.
1b. Work inside CDCAC to actually extract the proof "clues".
2. Put these together to prove "theory leaves" of an SMT proof.
3. Integrate with the Boolean part to produce a true SMT proof.
4. Worry about systematising this — build on existing SMT-LIB APIs.
5. Worry about VTS and other "non-fitting" heuristics.

Volunteers/ expressions of interest welcome, especially 1a and descendants.

A Ph.D. Opportunty

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

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)*, CEUR-WS 2752, 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, 2021.

Bibliography II

URL: https://doi.org/10.1016/j.jlamp.2020.100633.

Haniel Barbosa, Jasmin Christian Blanchette, Mathias Fleury, and Pascal Fontaine.
Scalable fine-grained proofs for formula processing.
Journal of Automated Reasoning, 64(3):485–510, 2020.
URL: https://doi.org/10.1007/s10817-018-09502-y.

C. Barrett, L. de Moura, and P. Fontaine.
Proofs in satisfiability modulo theories.
In D. Delahaye and B. Woltzenlogel Paleo, editors, *All about Proofs, Proofs for All*, volume 55 of *Mathematical Logic and Foundations*, pages 23–44. College Publications, London, UK, 2015.
URL: http://www.cs.stanford.edu/~barrett/pubs/BdMF15.pdf.

C. Barrett, P. Fontaine, and C. Tinelli.
The Satisfiability Modulo Theories Library (SMT-LIB).
Online Resource, 2016.
URL: `http://www.SMT-LIB.org`.

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`.

C. Cohen and A. Mahboubi.
A formal quantifier elimination for algebraically closed fields.
In S. Autexier, J. Calmet, D. Delahaye, P. Ion, L. Rideau, R. Rioboo, and A.P. Sexton, editors, *Intelligent Computer Mathematics*, volume 6167 of *Lecture Notes in Computer Science*, pages 189–203. Springer Berlin Heidelberg, 2010.
URL: https://doi.org/10.1007/978-3-642-14128-7_17.

C. Cohen and A. Mahboubi.
Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination.
Logical Methods in Computer Science, 8(1):1–40, 2012.
URL: https://doi.org/10.2168/LMCS-8(1:2)2012.

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 [CJ98]), 1975.
URL: `https://doi.org/10.1007/3-540-07407-4_17`.

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://doi.org/10.1007/978-3-642-35873-9_1`.

Bibliography VI

M.J. Heule, M. Järvisalo, and M. Suda.
Proceedings of SAT competition 2018: Solver and benchmark descriptions.
In *Technical Report B-2018-1, University of Helsinki Department of Computer Science*, 2018.
URL: https://helda.helsinki.fi/handle/10138/237063.

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.

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`.

A. Mahboubi.
Programming and certifying a CAD algorithm in the Coq system.
In T. Coquand, H. Lombardi, and M.F. Roy, editors, *Mathematics, Algorithms, Proofs*, number 05021 in Dagstuhl Seminar Proceedings. Internationales Begegnungs und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

Bibliography VIII

A. Mahboubi.
Implementing the cylindrical algebraic decomposition within the Coq system.
Mathematical Structures in Computer Science, 17(1):99–127, 2007.
URL: `https://doi.org/10.1017/S096012950600586X`.

A. Rebola-Pardo and A. Biere.
Two flavours of DRAT.
In *Proceedings of Pragmatics of SAT 2015 and 2018*, volume 59 of *EPiC Series in Computing*, pages 91–110, 2019.
URL: `https://doi.org/10.29007/lt8r`.

A. Stump, A. Reynolds, C. Tinelli, A. Laugesen, H. Eades, C. Oliver, and R. Zhang.
LFSC for SMT proofs: Work in progress.
In *Proof Exchange for Theorem Proving—Second International Workshop, PxTP 2012*, CEUR-WS 878, pages 21–27, 2012.
URL: `http://ceur-ws.org/Vol-878/paper1.pdf`.

Z. Tonks.
A poly-algorithmic quantifier elimination package in Maple.
In J. Gerhard and I. Kotsireas, editors, *Maple in Mathematics Education and Research*, volume 1125 of *Communications in Computer and Information Science*, pages 330–333. Springer International Publishing, 2020.
URL: `https://doi.org/10.1007/978-3-030-41258-6_13`.