Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

A conflict-driven adaption of CAD

Based on [Ábrahám et al. 2020]

Erika Ábrahám, James H. Davenport, Matthew England, Gereon Kremer

July 14th, 2020 – ICMS 2020 – TU Braunschweig

… what?

Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

… what?

Motivation: SMT solving
Find SAT or UNSAT

Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

RWTH AACHEN UNIVERSITY

… what?

Motivation: SMT solving
Find SAT or UNSAT

Sets of constraints
Polynomial inequalities

Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

RWTH AACHEN UNIVERSITY

… what?

Motivation: SMT solving
Find SAT or UNSAT

Sets of constraints
Polynomial inequalities

Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

Make guesses
Learn from errors

RWTH AACHEN UNIVERSITY

... what?

Motivation: SMT solving
Find SAT or UNSAT

Sets of constraints
Polynomial inequalities

Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings

Borrow theory from CAD
Apply it differently

Make guesses
Learn from errors

Goals of this talk

Convince you that

- this approach is solidly based on CAD theory and
- all the underlying machinery can be re-used from CAD.

However, also convince you that

- this approach has real benefits compared to CAD,
- is reasonably easy to implement, and
- performs well in practice.

RWTH AACHEN UNIVERSITY

In a nutshell

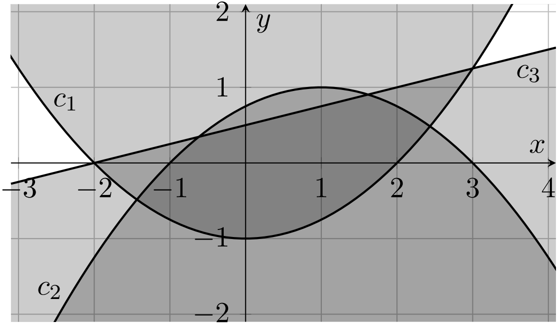
- Fix a variable ordering
- For the kth variable
  - Use constraints to exclude unsatisfiable intervals
  - Guess a value for the kth variable
  - Recurse to $k+1$st variable and obtain
    - a full variable assignment ($\rightarrow$ return SAT)
    - or a covering for the $k+1$st variable
  - Use CAD machinery to infer an interval from this covering
- Until the collected intervals form a covering for the kth variable

Called for the first variable, we get either

- a model, or
- a conflict (formulated as a covering).

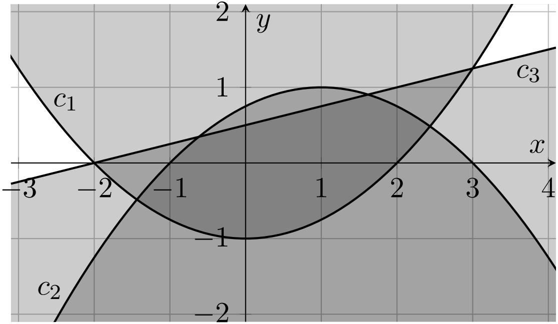
An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



An example

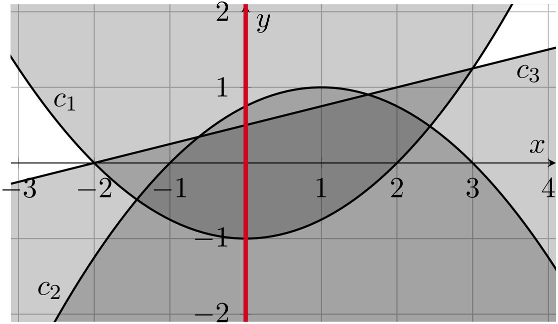
$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

An example

$$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$$

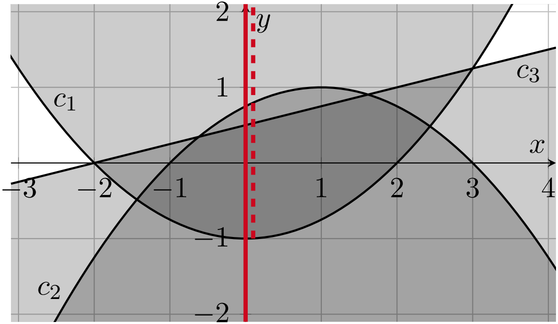


No constraint for x

Guess $x \mapsto 0$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



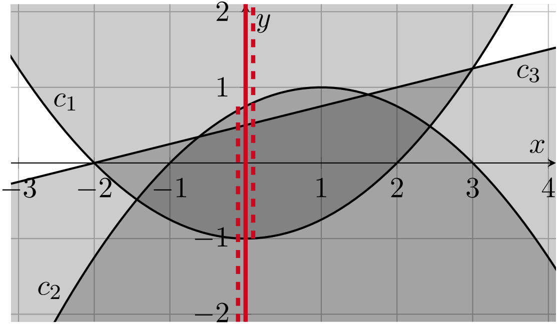
No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

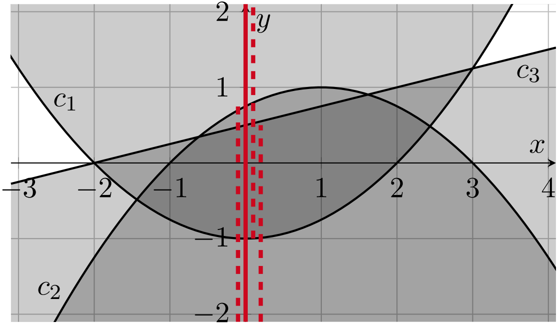
$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

RWTH AACHEN UNIVERSITY

An example

$c_1 : 4 \cdot y < x^2 - 4$ $c_2 : 4 \cdot y > 4 - (x-1)^2$ $c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

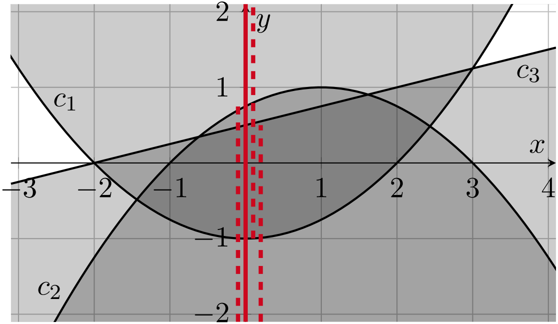
$c_2 \to y \notin (-\infty, 0.75)$

$c_3 \to y \notin (-\infty, 0.5)$

RWTH AACHEN UNIVERSITY

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

$c_3 \to y \notin (-\infty, 0.5)$

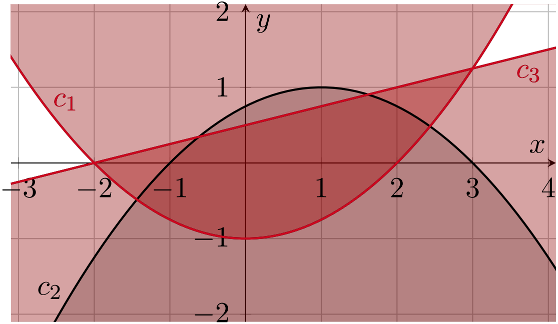
Construct covering

$(-\infty, 0.5), (-1, \infty)$

RWTH AACHEN UNIVERSITY

An example

$c_1 : 4 \cdot y < x^2 - 4$ $\quad c_2 : 4 \cdot y > 4 - (x-1)^2$ $\quad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

$c_3 \to y \notin (-\infty, 0.5)$

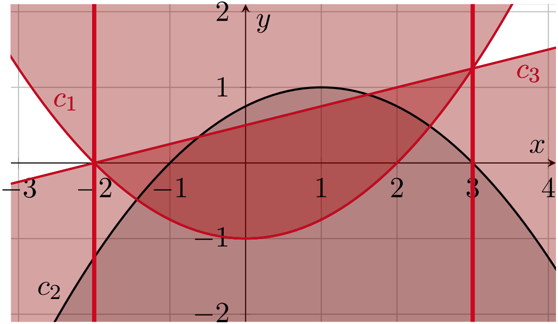
Construct covering

$(-\infty, 0.5), (-1, \infty)$

RWTH AACHEN UNIVERSITY

An example

$c_1 : 4 \cdot y < x^2 - 4$ $\qquad$ $c_2 : 4 \cdot y > 4 - (x-1)^2$ $\qquad$ $c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \to y \notin (-1, \infty)$

$c_2 \to y \notin (-\infty, 0.75)$

$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

$(-\infty, 0.5), (-1, \infty)$

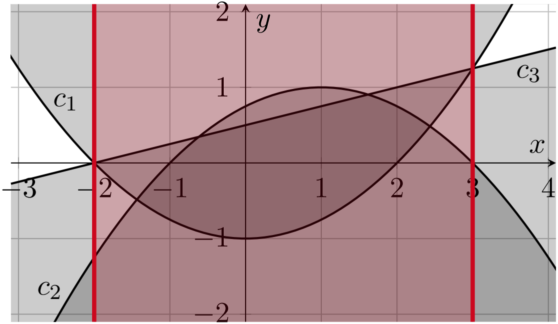
Construct interval for x

$x \notin (-2, 3)$

RWTH AACHEN UNIVERSITY

An example

$c_1 : 4 \cdot y < x^2 - 4 \qquad c_2 : 4 \cdot y > 4 - (x-1)^2 \qquad c_3 : 4 \cdot y > x + 2$



No constraint for x

Guess $x \mapsto 0$

$c_1 \rightarrow y \notin (-1, \infty)$

$c_2 \rightarrow y \notin (-\infty, 0.75)$

$c_3 \rightarrow y \notin (-\infty, 0.5)$

Construct covering

$(-\infty, 0.5), (-1, \infty)$

Construct interval for x

$x \notin (-2, 3)$

New guess for x

RWTH AACHEN UNIVERSITY

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))

𝕀 := get_unsat_intervals(s)
while ⋃_{I∈𝕀} I ≠ ℝ do
  | s_i := sample_outside(𝕀)
  | if i = n then return (SAT, (s_1, ..., s_{i-1}, s_i))
  | (f, O) := get_unsat_cover((s_1, ..., s_{i-1}, s_i))
  | if f = SAT then return (SAT, O)
  | else if f = UNSAT then
  |   | R := construct_characterization((s_1, ..., s_{i-1}, s_i), O)
  |   | J := interval_from_characterization((s_1, ..., s_{i-1}), s_i, R)
  |   | 𝕀 := 𝕀 ∪ {J}
return (UNSAT, 𝕀)
```

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))

𝕀 := get_unsat_intervals(s)
while ⋃_{I∈𝕀} I ≠ ℝ do
  s_i := sample_outside(𝕀)
  if i = n then  return (SAT, (s_1, ..., s_{i-1}, s_i))
  (f, O) := get_unsat_cover((s_1, ..., s_{i-1}, s_i))
  if f = SAT then  return (SAT, O)
  else if f = UNSAT then
    R := construct_characterization((s_1, ..., s_{i-1}, s_i), O)
    J := interval_from_characterization((s_1, ..., s_{i-1}), s_i, R)
    𝕀 := 𝕀 ∪ {J}
return (UNSAT, 𝕀)
```

Real root isolation over a partial sample point

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))

𝕀 := get_unsat_intervals(s)
while ⋃_{I∈𝕀} I ≠ ℝ do
    s_i := sample_outside(𝕀)
    if i = n then return (SAT, (s_1, ..., s_{i-1}, s_i))
    (f, O) := get_unsat_cover((s_1, ..., s_{i-1}, s_i))
    if f = SAT then return (SAT, O)
    else if f = UNSAT then
        R := construct_characterization((s_1, ..., s_{i-1}, s_i), O)
        J := interval_from_characterization((s_1, ..., s_{i-1}), s_i, R)
        𝕀 := 𝕀 ∪ {J}
return (UNSAT, 𝕀)
```

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

The main algorithm

function `get_unsat_cover`$((s_1, \ldots, s_{i-1}))$

$\mathbb{I}$:= `get_unsat_intervals(s)`

while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**

- s_i := `sample_outside`$(\mathbb{I})$
- **if** $i = n$ **then return** $(\mathrm{SAT}, (s_1, \ldots, s_i))$
- (f, O) := `get_unsat_cover`$((s_1, \ldots, s_{i-1}, s_i))$
- **if** $f = \mathrm{SAT}$ **then return** (SAT, O)
- **else if** $f = \mathrm{UNSAT}$ **then**
  - R := `construct_characterization`$((s_1, \ldots, s_{i-1}, s_i), O)$
  - J := `interval_from_characterization`$((s_1, \ldots, s_{i-1}), s_i, R)$
  - $\mathbb{I} := \mathbb{I} \cup \{J\}$

return $(\mathrm{UNSAT}, \mathbb{I})$

> Real root isolation over a partial sample point

> Select sample from $\mathbb{R} \setminus I$

> Recurse to next variable

RWTH AACHEN UNIVERSITY

The main algorithm

```
function get_unsat_cover((s_1, ..., s_{i-1}))

𝕀 := get_unsat_intervals(s)
while ⋃_{I∈𝕀} I ≠ ℝ do
  | s_i := sample_outside(𝕀)
  | if i = n then  return (SAT, (s_1, ..., s_i
  | (f, O) := get_unsat_cover((s_1, ..., s_{i-1}, s_i
  | if f = SAT then  return (SAT, O)
  | else if f = UNSAT then
  |  | R := construct_characterization((s_1, 
  |  | J := interval_from_characterization(
  |  | 𝕀 := 𝕀 ∪ {J}
return (UNSAT, 𝕀)
```

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

CAD-style projection: Roots of polynomials restrict where covering is still applicable

RWTH AACHEN UNIVERSITY

The main algorithm

function get_unsat_cover($(s_1, \ldots, s_{i-1})$)

$\mathbb{I}$:= get_unsat_intervals(s)
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  s_i := sample_outside($\mathbb{I}$)
  if $i = n$ **then return** $(\mathrm{SAT}, (s_1, \ldots, s_i$
  (f, O) := get_unsat_cover($((s_1, \ldots, s_{i-1}, s_i$
  if $f = \mathrm{SAT}$ **then return** (SAT, O)
  else if $f = \mathrm{UNSAT}$ **then**
    R := construct_characterization($((s_1,$
    J := interval_from_characterization(
    $\mathbb{I} := \mathbb{I} \cup \{J\}$
return $(\mathrm{UNSAT}, \mathbb{I})$

Real root isolation over a partial sample point

Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

CAD-style projection: Roots of polynomials restrict where covering is still applicable

Extract interval from polynomials

RWTH AACHEN UNIVERSITY

The main algorithm

function get_unsat_cover($(s_1, \ldots, s_{i-1})$)

$\mathbb{I}$:= get_unsat_intervals(s)
while $\bigcup_{I \in \mathbb{I}} I \neq \mathbb{R}$ **do**
  s_i := sample_outside($\mathbb{I}$)
  if $i = n$ **then** **return** $(\mathrm{SAT}, (s_1, \ldots, s_i$
  (f, O) := get_unsat_cover($(s_1, \ldots, s_{i-1}, s_i)$)
  if $f = \mathrm{SAT}$ **then** **return** (SAT, O)
  else if $f = \mathrm{UNSAT}$ **then**
    R := construct_characterization($(s_1,$
    J := interval_from_characterization(
    $\mathbb{I} := \mathbb{I} \cup \{J\}$
return $(\mathrm{UNSAT}, \mathbb{I})$

Real root isolation over a partial sample point

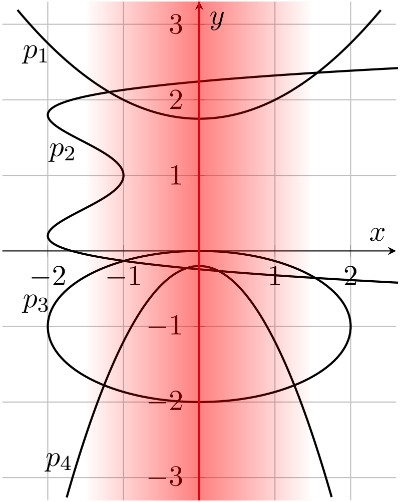
Select sample from $\mathbb{R} \setminus I$

Recurse to next variable

CAD-style projection: Roots of polynomials restrict where covering is still applicable

Extract interval from polynomials

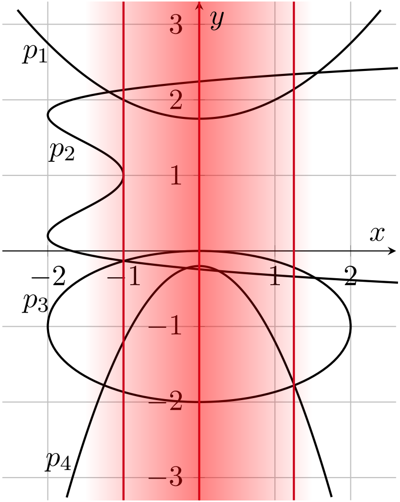
construct_characterization



Identify region around sample

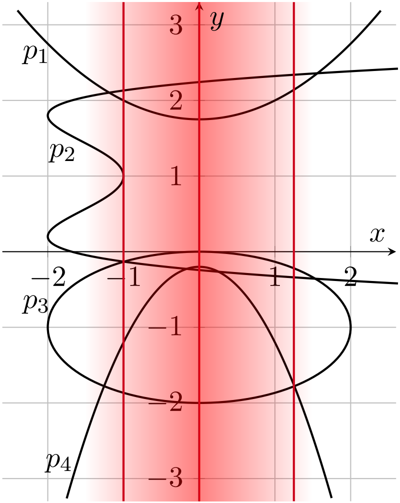
RWTH AACHEN UNIVERSITY

construct_characterization



Identify region around sample

construct_characterization



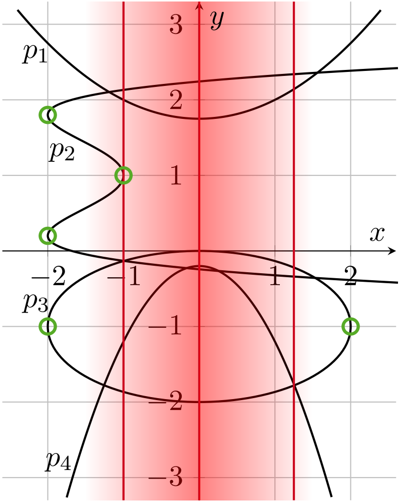
Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

RWTH AACHEN UNIVERSITY

construct_characterization

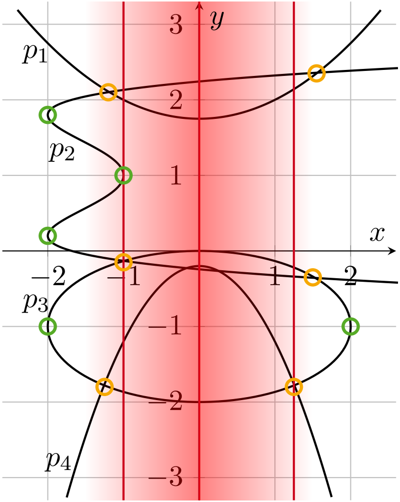


Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

construct_characterization



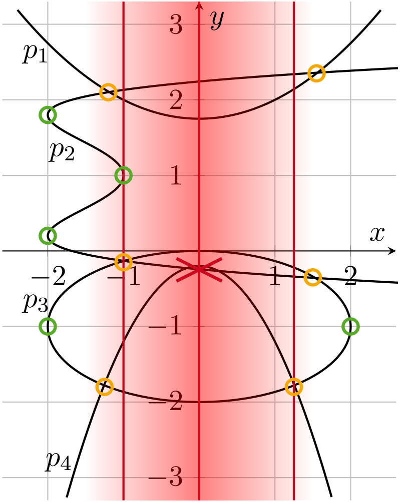
Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

RWTH AACHEN UNIVERSITY

construct_characterization



Identify region around sample

CAD projection:

- Discriminants (and coefficients)
- Resultants

Improvement over CAD:

- Resultants between neighbouring intervals only!

RWTH AACHEN UNIVERSITY

Experiments

First implemented in SMT-RAT

- Preliminary implementation (no incrementality, no optimizations)
- Easily outperforms regular CAD (from [Kremer et al. 2020])

Second implementation in CVC4

- Work in progress
- Used when linear solving insufficient
- Outperforms currently best solver (yices)

Solver	SAT	UNSAT	overall	
CVC4 (without CAC)	2148	3268	5416	47.1%
CVC4 (CAC)	4990	5369	10359	90.1%
yices (NLSAT)	4904	5437	10341	90.0%

Some interesting points

Comparison to regular CAD (like [Kremer et al. 2020]):

- Projection is "more local" (similar to [Moura et al. 2013] or [Brown 2015])
- Cells are larger (due to avoided resultants)
- Less bookkeeping to manage projection and lifting
- Allows to construct reasons for infeasibility

Future work

- Variable ordering (right now from [England et al. 2014])
- Incrementality and backtracking
- Use other projection operators (like [Lazard 1994])
- Exploit equational constraints
- Adaption to universally quantified problems or full first-order logic
- Asymptotic complexity

RWTH AACHEN UNIVERSITY

References

- Erika Ábrahám, James H. Davenport, Matthew England, and Gereon Kremer. Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings. Under review. 2020. arXiv: 2003.05633.
- Christopher W. Brown. "Open Non-uniform Cylindrical Algebraic Decompositions". In: ISSAC. 2015, pp. 85–92. DOI: 10.1145/2755996.2756654.
- Matthew England, Russell Bradford, James H. Davenport, and David Wilson. "Choosing a Variable Ordering for Truth-Table Invariant Cylindrical Algebraic Decomposition by Incremental Triangular Decomposition". In: ICMS. Vol. 8592. LNCS. 2014. DOI: 10.1007/978-3-662-44199-2_68.
- Gereon Kremer and Erika Ábrahám. "Fully Incremental Cylindrical Algebraic Decomposition". In: Journal of Symbolic Computation 100 (2020), pp. 11–37. DOI: 10.1016/j.jsc.2019.07.018.
- Daniel Lazard. "An Improved Projection for Cylindrical Algebraic Decomposition". In: Algebraic Geometry and its Applications. Springer, 1994, pp. 467–476. DOI: 10.1007/978-1-4612-2628-4_29.
- Leonardo de Moura and Dejan Jovanović. "A Model-Constructing Satisfiability Calculus". In: VMCAI. Vol. 7737. LNCS. 2013, pp. 1–12. DOI: 10.1007/978-3-642-35873-9_1.

RWTH AACHEN UNIVERSITY

Bath is recruiting!

Post-doctoral researcher for three years, ideally starting 1 October 2020 (but can be flexible).

Typical starting salary £39,000.

To work on a joint project with Matthew England on "Pushing Back the Doubly-Exponential Wall of Cylindrical Algebraic Decomposition".

Covid-19 has got in the way of formal advertising, but express interest to `J.H.Davenport@bath.ac.uk`