Implementing arithmetic over algebraic numbers

A tutorial for Lazard's lifting scheme

Gereon Kremer, Jens Brandt

Can people use computer algebra methods?

Can people use computer algebra methods?

Of course!

Can people use computer algebra methods?

Of course!

- Powerful software packages
  Random selection: GAP, Magma, Maple, Mathematica, Sage, SINGULAR, ...
- Most new results can be implemented concisely
  with one of the above software packages
- Oftentimes new results are published as software
  within one of the above software packages

Can people use computer algebra methods?

Of course!

- Powerful software packages
  Random selection: GAP, Magma, Maple, Mathematica, Sage, SINGULAR, ...
- Most new results can be implemented concisely
  with one of the above software packages
- Oftentimes new results are published as software
  within one of the above software packages

We are done here, thanks for the attention!

Can people use computer algebra methods?

Of course!

- Powerful software packages
  Random selection: GAP, Magma, Maple, Mathematica, Sage, SINGULAR, ...
- Most new results can be implemented concisely
  with one of the above software packages
- Oftentimes new results are published as software
  within one of the above software packages

We are done here, thanks for the attention!

Heretical question:
Is your implementation in ${your favourite CAS} actually useful?

A different point of view

Assume ${your favourite computer algebra method} shall be

A different point of view

Assume ${your favourite computer algebra method} shall be

- used in a project that can not use closed-source software[1].
  ~~Macsyma~~, ~~Magma~~, ~~Maple~~, ~~Mathcad~~, ~~Mathematica~~, ~~MATLAB~~, ~~SMath~~, ~~Wolfram Alpha~~

[1] Licensing and stuff. Ask you least distrusted lawer.

A different point of view

Assume ${your favourite computer algebra method} shall be

- used in a project that can not use closed-source software[1].

  ~~Macsyma~~, ~~Magma~~, ~~Maple~~, ~~Mathcad~~, ~~Mathematica~~, ~~MATLAB~~, ~~SMath~~, ~~Wolfram Alpha~~
- maintained reliably[2].

  ~~Axiom~~, ~~Derive~~, ~~KANT/KASH~~, ~~Magnus~~, ~~Mathomatic~~, ~~MuMATH~~, ~~MuPAD~~, ~~OpenAxiom~~

[1] Licensing and stuff. Ask you least distrusted lawer.

[2] Because, you know, people care about having issues fixed.

A different point of view

Assume ${your favourite computer algebra method} shall be

- used in a project that can not use closed-source software[1].
  ~~Macsyma~~, ~~Magma~~, ~~Maple~~, ~~Mathcad~~, ~~Mathematica~~, ~~MATLAB~~, ~~SMath~~, ~~Wolfram Alpha~~
- maintained reliably[2].
  ~~Axiom~~, ~~Derive~~, ~~KANT/KASH~~, ~~Magnus~~, ~~Mathomatic~~, ~~MuMATH~~, ~~MuPAD~~, ~~OpenAxiom~~
- integrated reasonably easily and efficiently[3].
  ~~SageMath~~, ~~SymPy~~

[1]Licensing and stuff. Ask you least distrusted lawer.
[2]Because, you know, people care about having issues fixed.
[3]Conversion overhead sometimes is an issue.

A different point of view

Assume ${your favourite computer algebra method} shall be

- used in a project that can not use closed-source software[1].
  ~~Macsyma~~, ~~Magma~~, ~~Maple~~, ~~Mathcad~~, ~~Mathematica~~, ~~MATLAB~~, ~~SMath~~, ~~Wolfram Alpha~~
- maintained reliably[2].
  ~~Axiom~~, ~~Derive~~, ~~KANT/KASH~~, ~~Magnus~~, ~~Mathomatic~~, ~~MuMATH~~, ~~MuPAD~~, ~~OpenAxiom~~
- integrated reasonably easily and efficiently[3].
  ~~SageMath~~, ~~SymPy~~
- extended and modified[4].
  model construction? incrementality? infeasible subsets?

[1] Licensing and stuff. Ask you least distrusted lawer.
[2] Because, you know, people care about having issues fixed.
[3] Conversion overhead sometimes is an issue.
[4] Using a black box is not always appropriate.

Computer algebra techniques in SMT solvers

SMT $\hat{=}$ Satisfiability modulo theories $\hat{=}$ first-order formulae over theories

Computer algebra techniques in SMT solvers

SMT $\hat{=}$ Satisfiability modulo theories $\hat{=}$ first-order formulae over theories

SMT solvers have found widespread use: all of "Big Tech" heavily use them.

Licensing issues, performance and reliability are really important

Computer algebra techniques in SMT solvers

SMT $\hat{=}$ Satisfiability modulo theories $\hat{=}$ first-order formulae over theories

SMT solvers have found widespread use: all of "Big Tech" heavily use them.
Licensing issues, performance and reliability are really important

For NRA solvers implement (variants of) cylindrical algebraic decomposition.

Computer algebra techniques in SMT solvers

SMT $\hat{=}$ Satisfiability modulo theories $\hat{=}$ first-order formulae over theories

SMT solvers have found widespread use: all of "Big Tech" heavily use them.
Licensing issues, performance and reliability are really important

For NRA solvers implement (variants of) cylindrical algebraic decomposition.

How do these SMT solvers implement CAD?
They do somehow – so everything is fine?

Computer algebra techniques in SMT solvers

SMT $\hat{=}$ Satisfiability modulo theories $\hat{=}$ first-order formulae over theories

SMT solvers have found widespread use: all of "Big Tech" heavily use them.
Licensing issues, performance and reliability are really important

For NRA solvers implement (variants of) cylindrical algebraic decomposition.

How do these SMT solvers implement CAD?
They do somehow – so everything is fine?

Few libraries exist within or alongside SMT solvers that implement the bare minimum. Developers that are able and willing to do this job are rare. The problems fill another talk.

This paper: Lazard's lifting and projection scheme

Assume you already have a working CAD with McCallum's projection.

[McCallum 1985] [Lazard 1994] [McCallum et al. 2019]

This paper: Lazard's lifting and projection scheme

Assume you already have a working CAD with McCallum's projection.

You want Lazard's projection & lifting:

- (Almost) the smallest projection set for CAD
  resultant + discriminant + leading coefficient + trailing coefficient
- Changing the projection is trivial
- Proven to be sound (unlike McCallum)

[McCallum 1985] [Lazard 1994] [McCallum et al. 2019]

This paper: Lazard's lifting and projection scheme

Assume you already have a working CAD with McCallum's projection.

You want Lazard's projection & lifting:

- (Almost) the smallest projection set for CAD
  resultant + discriminant + leading coefficient + trailing coefficient
- Changing the projection is trivial
- Proven to be sound (unlike McCallum) if you implement the lifting scheme.

[McCallum 1985] [Lazard 1994] [McCallum et al. 2019]

This paper: Lazard's lifting and projection scheme

Assume you already have a working CAD with McCallum's projection.

You want Lazard's projection & lifting:

- (Almost) the smallest projection set for CAD
  resultant + discriminant + leading coefficient + trailing coefficient
- Changing the projection is trivial
- Proven to be sound (unlike McCallum) if you implement the lifting scheme.

$$\begin{aligned}
&\textbf{for } i := 0 \textbf{ to } n-1 \textbf{ do}\\
&\quad v_i := \arg\max_{v\in\mathbb{Z}}(x_i - a_i)^v \text{ divides } q\\
&\quad q := q/(x_i - a_i)^{v_i}\\
&\quad q := subst(a_i, x_i, q)\\
&\text{isolate real roots of } q \text{ (now univariate in } x_n)
\end{aligned}$$

[McCallum 1985] [Lazard 1994] [McCallum et al. 2019]

This paper: Lazard's lifting and projection scheme

Assume you already have a working CAD with McCallum's projection.

You want Lazard's projection & lifting:

- (Almost) the smallest projection set for CAD
  resultant + discriminant + leading coefficient + trailing coefficient
- Changing the projection is trivial
- Proven to be sound (unlike McCallum) if you implement the lifting scheme.

$$\begin{aligned}
&\textbf{for } i := 0 \textbf{ to } n-1 \textbf{ do}\\
&\quad v_i := \arg\max_{v\in\mathbb{Z}} (x_i - a_i)^v \text{ divides } q\\
&\quad q := q/(x_i - a_i)^{v_i}\\
&\quad q := subst(a_i, x_i, q)\\
&\text{isolate real roots of } q \text{ (now univariate in } x_n)
\end{aligned}$$

Seems easy enough?

[McCallum 1985] [Lazard 1994] [McCallum et al. 2019]

This paper: Lazard's lifting and projection scheme

Assume you already have a working CAD with McCallum's projection.

You want Lazard's projection & lifting:

- (Almost) the smallest projection set for CAD
  resultant + discriminant + leading coefficient + trailing coefficient
- Changing the projection is trivial
- Proven to be sound (unlike McCallum) if you implement the lifting scheme.

```
for i := 0 to n − 1 do
    v_i := arg max_{v∈ℤ} (x_i − a_i)^v divides q
    q := q/(x_i − a_i)^{v_i}
    q := subst(a_i, x_i, q)
isolate real roots of q (now univariate in x_n)
```

$$\begin{array}{l}\textbf{for}\ i := 0\ \textbf{to}\ n-1\ \textbf{do}\\ \quad v_i := \arg\max_{v\in\mathbb{Z}} (x_i - a_i)^v \text{ divides } q\\ \quad q := q/(x_i - a_i)^{v_i}\\ \quad q := subst(a_i, x_i, q)\\ \text{isolate real roots of } q \text{ (now univariate in } x_n)\end{array}$$

Seems easy enough?
What if $a_i \in \overline{\mathbb{Q}} \setminus \mathbb{Q}$?

[McCallum 1985] [Lazard 1994] [McCallum et al. 2019]

So what?

for $i := 0$ **to** $n-1$ **do**
    $v_i := \arg\max_{v \in \mathbb{Z}} (x_i - a_i)^v$ divides q
    $q := q / (x_i - a_i)^{v_i}$
    $q := subst(a_i, x_i, q)$
isolate real roots of q (now univariate in x_n)

If $a_i \in \overline{\mathbb{Q}} \setminus \mathbb{Q}$, this requires proper polynomial arithmetic over algebraic numbers!

So what?

for $i := 0$ **to** $n-1$ **do**
$v_i := \arg\max_{v \in \mathbb{Z}} (x_i - a_i)^v$ divides q
$q := q / (x_i - a_i)^{v_i}$
$q := subst(a_i, x_i, q)$
isolate real roots of q (now univariate in x_n)

If $a_i \in \overline{\mathbb{Q}} \setminus \mathbb{Q}$, this requires proper polynomial arithmetic over algebraic numbers!

Core issue: multivariate factorization over a field extension.

Also: take care which operation is performed in which structure!

So what?

$$\textbf{for}\ \ i := 0\ \ \textbf{to}\ \ n-1\ \ \textbf{do}$$
$$v_i := \arg\max_{v\in\mathbb{Z}} (x_i - a_i)^v \text{ divides } q$$
$$q := q/(x_i - a_i)^{v_i}$$
$$q := subst(a_i, x_i, q)$$
$$\text{isolate real roots of } q \text{ (now univariate in } x_n)$$

If $a_i \in \overline{\mathbb{Q}} \setminus \mathbb{Q}$, this requires proper polynomial arithmetic over algebraic numbers!

Core issue: multivariate factorization over a field extension.

Also: take care which operation is performed in which structure!

Observation: Implementing multivariate factorization is prohibitive

for the SMT people

</lament>

What's in the paper?

[Abbott et al. 2018] [Jovanovic et al. 2017]

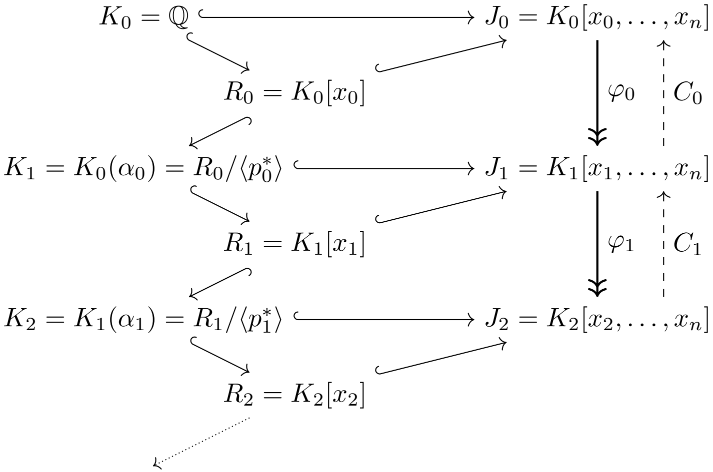
</lament>

What's in the paper?

How to implement Lazard's lifting given CoCoALib and LibPoly.

[Abbott et al. 2018] [Jovanovic et al. 2017]

Algebraic framework: tower of field extensions



Implementation

In the paper: actual working code!

https://github.com/cvc5/cvc5/blob/master/src/theory/arith/nl/cad/lazard_evaluation.cpp

```
vector<RingElem> p; // p_0 ... p_{n-1}
vector<ring> K;     // K_0 ... K_n
vector<ring> R;     // R_0 ... R_n
K[0] = RingQQ();
// assigned variables x_0, ... x_{n-1}
for (size_t i = 0; i < n; ++i)
{
    R[i] = NewPolyRing(K[i],
    {NewSymbol()});
    RingElem mipo = /* from R_i */;
    auto facs = factor(mipo);
    p[i] = /* fac that vanishes */;
    K[i+1] = NewQuotientRing(R[i],
    ideal(p[i]));
}
// free variable x_n
R[n] = NewPolyRing(K[n], {NewSymbol()});
```

Experiments (SMT-LIB, QF_NRA, 10min)

lifting	projection	sat	unsat	total
libpoly	McCallum	5064	5378	10442
libpoly	Lazard	5062	5377	10439
Lazard	McCallum	5088	5370	10458
Lazard	Lazard	5090	5371	10461

- 4752 of 11552 benchmarks entered the nonlinear solver
- 925 of 11552 benchmarks require lifting non-rational assignments
- 664 of 11552 benchmarks see vanishing factors (750k total)
- McCallum and Lazard are identical on $> 99.5\%$ of the benchmarks

Experiments (SMT-LIB, `QF_NRA`, 10min)

lifting	projection	sat	unsat	total
libpoly	McCallum	5064	5378	10442
libpoly	Lazard	5062	5377	10439
Lazard	McCallum	5088	5370	10458
Lazard	Lazard	5090	5371	10461

- 4752 of 11552 benchmarks entered the nonlinear solver
- 925 of 11552 benchmarks require lifting non-rational assignments
- 664 of 11552 benchmarks see vanishing factors (750k total)
- McCallum and Lazard are identical on $> 99.5\%$ of the benchmarks

Correctness is for free!

Conclusion

- an implementation is not useful per se
- people will need to modify your implementation
- many applications need free libraries (e.g. CoCoALib)

Conclusion

- an implementation is not useful per se
- people will need to modify your implementation
- many applications need free libraries (e.g. CoCoALib)

- Lazard's lifting requires additional algorithms
- we provide an implementation based on CoCoALib
- oftentimes not necessary ($\approx 6\%$ of SMT-LIB benchmarks)
- no significant impact on performance
- implementation is sound now

References I

- John Abbott, Anna M. Bigatti, and Elisa Palezzato. "New in CoCoA-5.2.4 and CoCoALib-0.99600 for SC-Square". In: SC^2. FLoC. Vol. 2189. July 2018, pp. 88–94. URL: http://ceur-ws.org/Vol-2189/paper4.pdf.
- Dejan Jovanovic and Bruno Dutertre. "LibPoly: A Library for Reasoning about Polynomials". In: SMT. CAV. Vol. 1889. 2017. URL: http://ceur-ws.org/Vol-1889/paper3.pdf.
- Daniel Lazard. "An Improved Projection for Cylindrical Algebraic Decomposition". In: Algebraic Geometry and its Applications. 1994. Chap. 29, pp. 467–476. DOI: 10.1007/978-1-4612-2628-4_29.
- Scott McCallum. "An Improved Projection Operation for Cylindrical Algebraic Decomposition". In: EUROCAL. Vol. 204. 1985, pp. 277–278. DOI: 10.1007/3-540-15984-3_277.
- Scott McCallum, Adam Parusiński, and Laurentiu Paunescu. "Validity proof of Lazard's method for CAD construction". In: Journal of Symbolic Computation 92 (2019), pp. 52–69. DOI: 10.1016/j.jsc.2017.12.002.