Incremental CAD

Making CAD work for SMT solving

based on [KA18]

Gereon Kremer

July 26th, 2018

ICMS'18 – University of Notre Dame

RWTH AACHEN UNIVERSITY

Context: SC^2

EU H2020-FETOPEN-2015-CSA 712689

Satisfiability Checking and Symbolic Computation

EU project to stimulate cooperations
More than 50 partners and associates

Industry: Altran, BTC, ClearSy, Imandra, L4B, Maplesoft, Microsoft, MJC2, NAG, SRI, Systerel, Wolfram

Also at ICMS: Erika Ábrahám, James Davenport, Matthew England, Stephen Forrest, Xiao-Shan Gao, Jürgen Gerhard, Jan Horacek, Martin Kreuzer, Alexei Lisitsa, Thomas Sturm

RWTH AACHEN UNIVERSITY

SMT solving

Satisfiability Modulo Theories (SMT)

Is an existentially quantified first-order formula φ satisfiable?

$$\exists x.\varphi(x) \equiv true$$

RWTH AACHEN UNIVERSITY

SMT solving

Satisfiability Modulo Theories (SMT)

Is an existentially quantified first-order formula φ satisfiable?

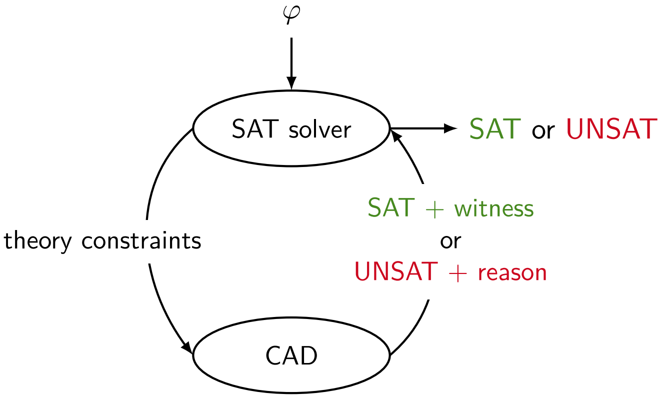
$$\exists x.\varphi(x) \equiv true$$

Applications:

- Software verification, test-case generation
- Termination proving
- Controller synthesis
- Scheduling and planning
- Product design automation
- And growing ...

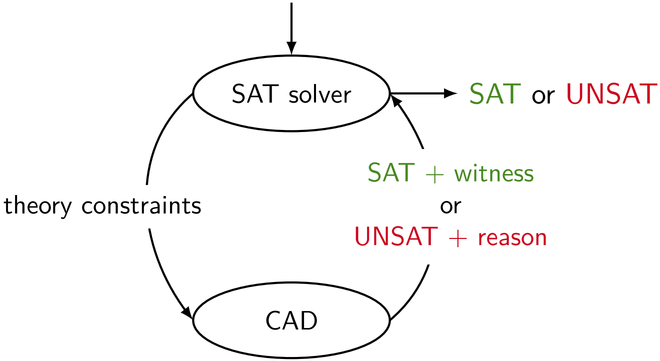
RWTH AACHEN UNIVERSITY

SMT solving



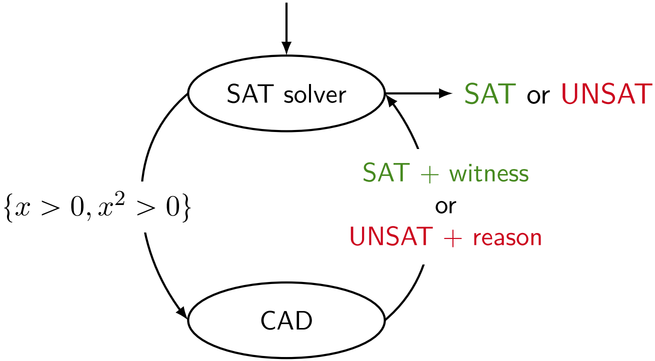
SMT solving

$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3)$$



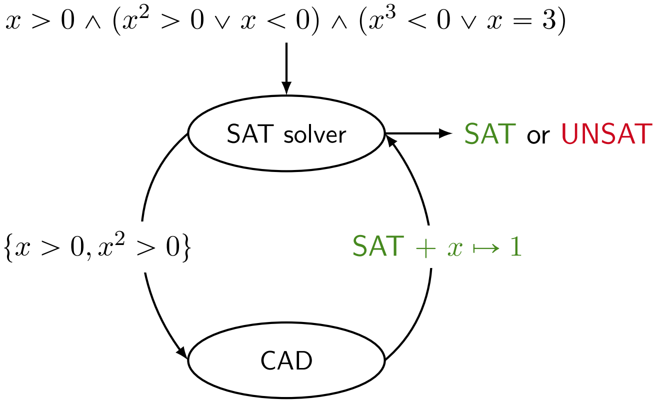
SMT solving

$$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3)$$



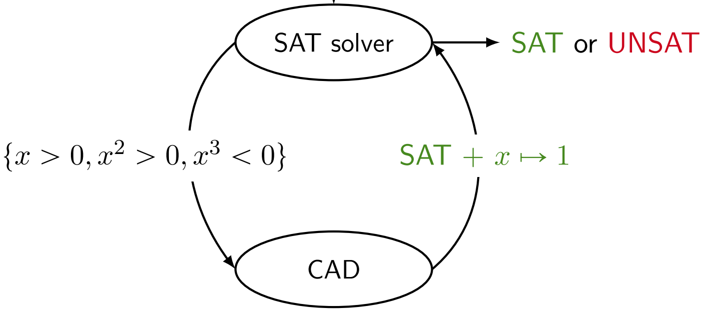
RWTH AACHEN UNIVERSITY

SMT solving



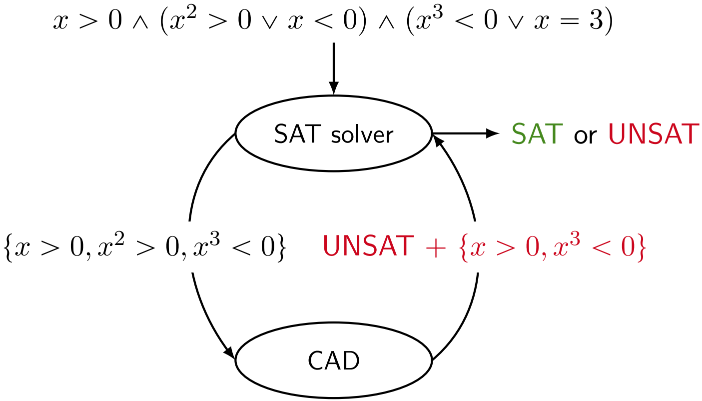
SMT solving

$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3)$$



RWTH AACHEN UNIVERSITY

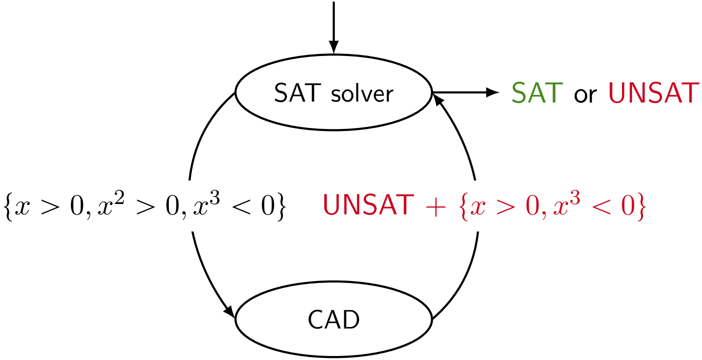
SMT solving



RWTH AACHEN UNIVERSITY

SMT solving

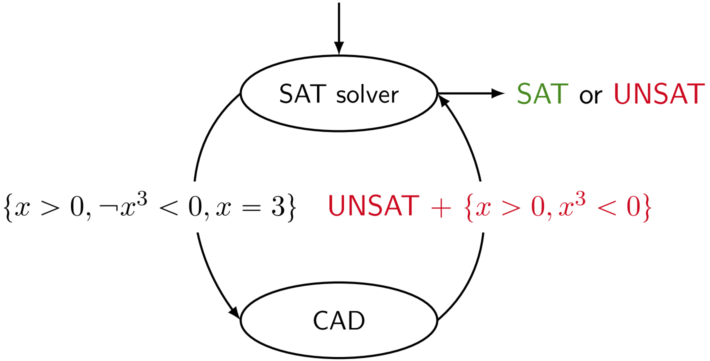
$$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3) \wedge (\neg x > 0 \vee \neg x^3 < 0)$$



RWTH AACHEN UNIVERSITY

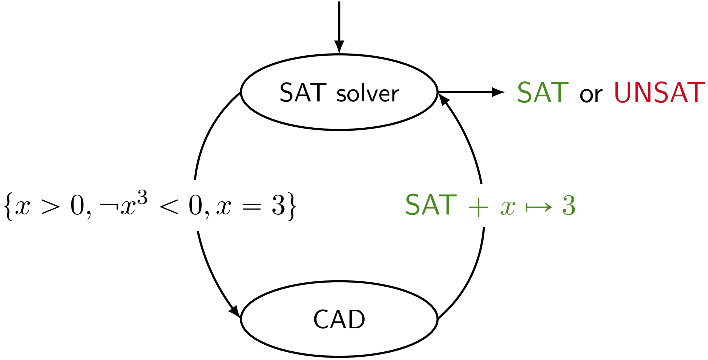
SMT solving

$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0)$$



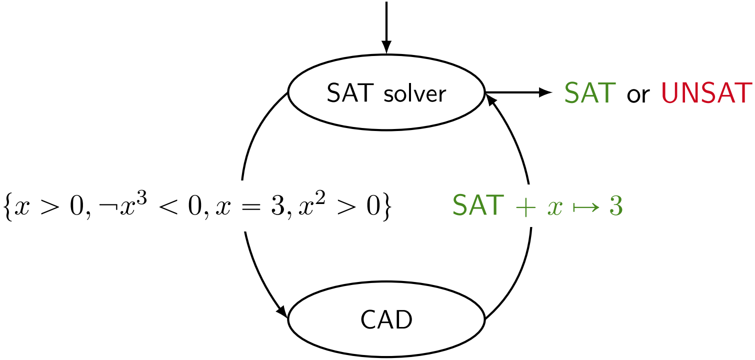
SMT solving

$$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3) \wedge (\neg x > 0 \vee \neg x^3 < 0)$$



SMT solving

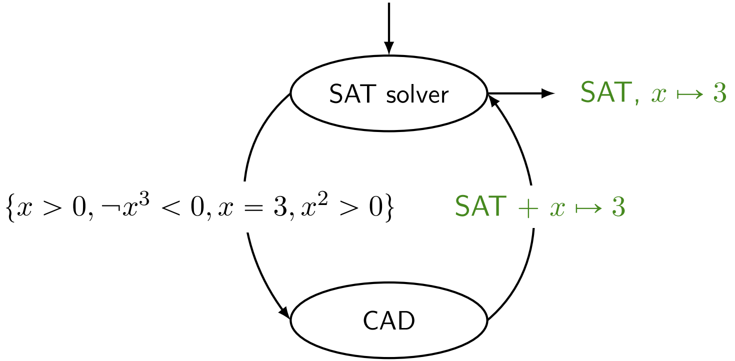
$$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3) \wedge (\neg x > 0 \vee \neg x^3 < 0)$$



RWTH AACHEN UNIVERSITY

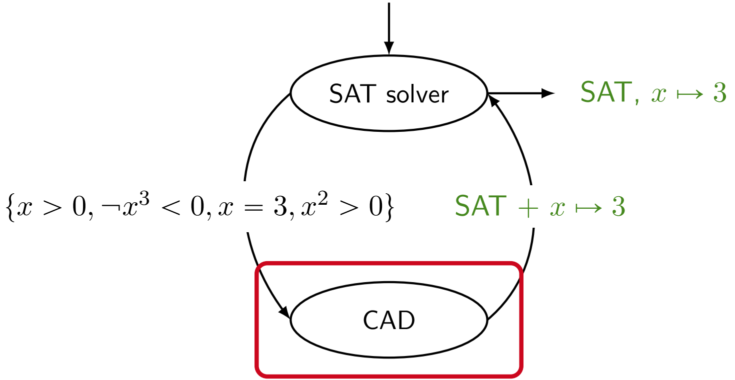
SMT solving

$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0)$$



SMT solving

$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3) \land (\neg x > 0 \lor \neg x^3 < 0)$$



RWTH AACHEN UNIVERSITY

Cylindrical Algebraic Decomposition

Constraints

$\downarrow$

$P_n \subset \mathbb{Z}[x_1..x_n]$

$\downarrow$

$P_{n-1} \subset \mathbb{Z}[x_1..x_{n-1}]$

$\downarrow$

$\vdots$

$\downarrow$

$P_1 \subset \mathbb{Z}[x_1] \longrightarrow Z_1 \subset \mathbb{R}$

$\uparrow$

$Z_2 \subset Z_1 \times \mathbb{R}$

$\uparrow$

$\vdots$

$\uparrow$

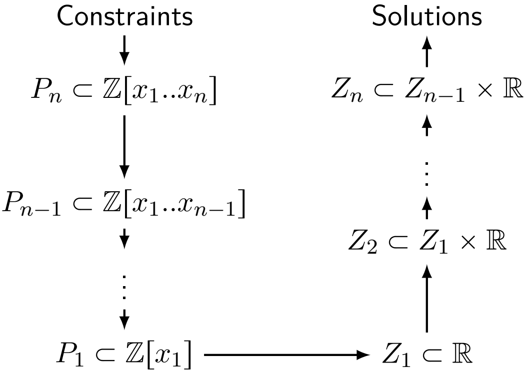
$Z_n \subset Z_{n-1} \times \mathbb{R}$

$\uparrow$

Knowledge

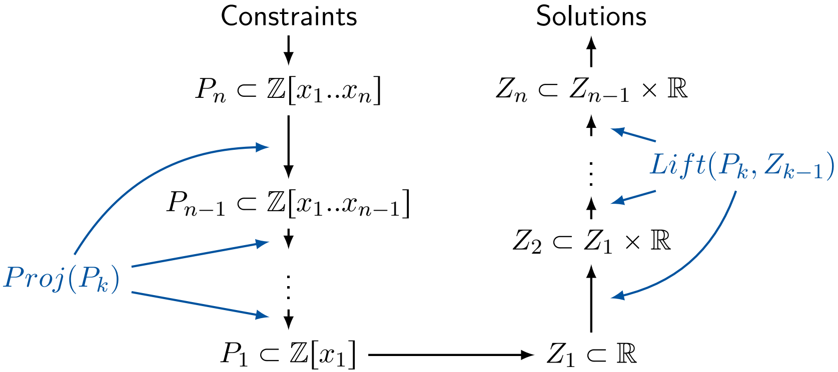
RWTH AACHEN UNIVERSITY

Cylindrical Algebraic Decomposition



Note: We only deal with the purely existential case!

Cylindrical Algebraic Decomposition



Note: We only deal with the purely existential case!

RWTH AACHEN UNIVERSITY

SMT compliancy

- Stop early when a satisfying witness is found
- Add constraints and check again
- Remove constraints
- Provide reason for unsatisfiability

RWTH AACHEN UNIVERSITY

SMT compliancy

- Stop early when a satisfying witness is found
  How to get there fast?
  How to keep a consistent state?
- Add constraints and check again
- Remove constraints
- Provide reason for unsatisfiability

SMT compliancy

- Stop early when a satisfying witness is found
  - How to get there fast?
  - How to keep a consistent state?
- Add constraints and check again
  - How to extend projection and lifting dynamically?
  - How to retain as much information as possible?
- Remove constraints
- Provide reason for unsatisfiability

RWTH AACHEN UNIVERSITY

SMT compliancy

- Stop early when a satisfying witness is found
  - How to get there fast?
  - How to keep a consistent state?
- Add constraints and check again
  - How to extend projection and lifting dynamically?
  - How to retain as much information as possible?
- Remove constraints
  - How to remove some of the polynomials and samples?
  - How to throw away as little information as possible?
- Provide reason for unsatisfiability

RWTH AACHEN UNIVERSITY

SMT compliancy

- Stop early when a satisfying witness is found
  - How to get there fast?
  - How to keep a consistent state?
- Add constraints and check again
  - How to extend projection and lifting dynamically?
  - How to retain as much information as possible?
- Remove constraints
  - How to remove some of the polynomials and samples?
  - How to throw away as little information as possible?
- Provide reason for unsatisfiability
  - Which constraints reject all the samples?
  - Solved in [JDF15], though implementation differs.

RWTH AACHEN UNIVERSITY

Taking a step back

What is the purpose of CAD for us?

RWTH AACHEN UNIVERSITY

CAD – Traditional approach [Col75]

- Extract P_n from constraints
- For $k = n \dots 2$: $P_{n-1} = Proj(P_n)$
- Sample Z_1 from P_1
- For $k = 2 \dots n$: $Z_k = Lift(P_n, Z_{n-1})$
- Extract solutions from Z_n

RWTH AACHEN UNIVERSITY

CAD – Traditional approach [Col75]

- Extract P_n from constraints
- For $k = n \dots 2$: $P_{n-1} = Proj(P_n)$
- Sample Z_1 from P_1
- For $k = 2 \dots n$: $Z_k = Lift(P_n, Z_{n-1})$
- Extract solutions from Z_n

- We compute all polynomials
- We compute all sample points
- We have no idea how to add or remove constraints

RWTH AACHEN UNIVERSITY

Partial CAD [CH91]

Observations:

- Every $s \in Z_{k-1}$ can be lifted separately.
- Every $s \in Z_{k-1}$ induces a separate set $Z_k^s \subseteq Z_k$.

RWTH AACHEN UNIVERSITY

Partial CAD [CH91]

Observations:

- Every $s \in Z_{k-1}$ can be lifted separately.
- Every $s \in Z_{k-1}$ induces a separate set $Z_k^s \subseteq Z_k$.

Essential idea

Consider the lifting to be a tree of sample points.
Explore the tree recursively.
Evaluate partial sample points during exploration.
Eagerly propagate evaluation results and skip redundant sample points.

RWTH AACHEN UNIVERSITY

Partial CAD [CH91]

Observations:

- Every $s \in Z_{k-1}$ can be lifted separately.
- Every $s \in Z_{k-1}$ induces a separate set $Z_k^s \subseteq Z_k$.

Essential idea

Consider the lifting to be a tree of sample points.
Explore the tree recursively.
Evaluate partial sample points during exploration.
Eagerly propagate evaluation results and skip redundant sample points.

Additionally:

- Lift $s \in Z_{k-1}$ with every $p \in P_k$ separately.
- Keep a queue of remaining lifting steps (s, p) for continuation.

RWTH AACHEN UNIVERSITY

Partial projection

Rough template for recent projection operators [McC98, Bro01]

$$Proj(P) = \{\mathrm{disc}(p), \mathrm{coeffs}^*(p) \mid p \in P\} \cup \\ \{\mathrm{res}(p,q) \mid p,q \in P\}$$

RWTH AACHEN UNIVERSITY

Partial projection

Rough template for recent projection operators [McC98, Bro01]

$$
\begin{aligned}
Proj(P) = &\{\mathrm{disc}(p), \mathrm{coeffs}^*(p) \mid p \in P\} \cup \\
&\{\mathrm{res}(p, q) \mid p, q \in P\}
\end{aligned}
$$

Observations:

- Every step is local to only one or two polynomials.
- We can interrupt the computation frequently.

RWTH AACHEN UNIVERSITY

Partial projection

Rough template for recent projection operators [McC98, Bro01]

$$Proj(P) = \{\mathrm{disc}(p), \mathrm{coeffs}^*(p) \mid p \in P\} \cup \\ \{\mathrm{res}(p,q) \mid p,q \in P\}$$

Observations:

- Every step is local to only one or two polynomials.
- We can interrupt the computation frequently.

Key ideas:

- Split $Proj(P)$ into a sequence of projection steps.
- Keep a queue of projection steps for continuation.

Lazy projection

Observations:

- We lift every (s, p) individually.
- We can also lift $(s, \cdot)$ and guess.
- We can stop as soon as a satisfying sample point is found.

RWTH AACHEN UNIVERSITY

Lazy projection

Observations:

- We lift every (s, p) individually.
- We can also lift $(s, \cdot)$ and guess.
- We can stop as soon as a satisfying sample point is found.

Why should we even start with the projection?

Lazy projection

Observations:

- We lift every (s, p) individually.
- We can also lift $(s, \cdot)$ and guess.
- We can stop as soon as a satisfying sample point is found.

Why should we even start with the projection?

Key ideas:

- Start with lifting.
- Perform lifting with respect to an incomplete projection.
- Only when lifting is complete*, spend time on the projection.

RWTH AACHEN UNIVERSITY

Lazy partial CAD

1. Perform lifting.
2. Return SAT if satisfying sample point was found.
3. Return UNSAT if the projection is complete.
4. Perform a projection step, go back to 1.

RWTH AACHEN UNIVERSITY

Lazy partial CAD

1. Perform lifting.
2. Return SAT if satisfying sample point was found.
3. Return UNSAT if the projection is complete.
4. Perform a projection step, go back to 1.

Observations:

- Completely driven by the search for a solution.
- Eventually converges to a complete CAD (if UNSAT).
- New choices to be made:
  - Order of lifting steps? (DFS? BFS? Something else?)
  - Order of projection steps? (Level? Degree?)
- Can be continued easily.

RWTH AACHEN UNIVERSITY

Back to our topic

What about SMT compliancy now?

Reminder:

- ✓ early abort
- add constraints
- remove constraints
- ✓ reasons for unsatisfiability

RWTH AACHEN UNIVERSITY

Adding constraints

Observations:

- Partial projection maintains a queue of remaining projection steps.
- Partial lifting maintains a queue of remaining lifting steps.
- We can extend these queues for a new polynomial.
- We can continue from there.

RWTH AACHEN UNIVERSITY

Adding constraints

Observations:

- Partial projection maintains a queue of remaining projection steps.
- Partial lifting maintains a queue of remaining lifting steps.
- We can extend these queues for a new polynomial.
- We can continue from there.

If partial projection and partial lifting is in place...

- ... adding new polynomials is easy.
- ... extending the partial CAD is natural.
- ... all previous computations can be reused.

RWTH AACHEN UNIVERSITY

Removing constraints

Our lazy partial CAD is monotically growing.
$\Rightarrow$ adding constraints is easy, but removing constraints is hard.

Removing constraints

Our lazy partial CAD is monotically growing.
$\Rightarrow$ adding constraints is easy, but removing constraints is hard.

Different options:

- Keep everything and ignore removals.
- Reset whenever we remove something.
- Save a snapshot once in a while and restore it.
- Figure out which polynomials to remove and update properly.

Removing constraints

Our lazy partial CAD is monotically growing.
$\Rightarrow$ adding constraints is easy, but removing constraints is hard.

Different options:

- Keep everything and ignore removals.
  Constraints accumulate and CAD eventually blows up.
- Reset whenever we remove something.
- Save a snapshot once in a while and restore it.
- Figure out which polynomials to remove and update properly.

RWTH AACHEN UNIVERSITY

Removing constraints

Our lazy partial CAD is monotically growing.
$\Rightarrow$ adding constraints is easy, but removing constraints is hard.

Different options:

- Keep everything and ignore removals.
  Constraints accumulate and CAD eventually blows up.
- Reset whenever we remove something.
  Mostly destroys the idea of retaining state.
- Save a snapshot once in a while and restore it.
- Figure out which polynomials to remove and update properly.

Removing constraints

Our lazy partial CAD is monotically growing.
$\Rightarrow$ adding constraints is easy, but removing constraints is hard.

Different options:

- Keep everything and ignore removals.
  Constraints accumulate and CAD eventually blows up.
- Reset whenever we remove something.
  Mostly destroys the idea of retaining state.
- Save a snapshot once in a while and restore it.
  When to snapshot? How many to keep? Memory usage?
- Figure out which polynomials to remove and update properly.

Removing constraints

Our lazy partial CAD is monotically growing.
$\Rightarrow$ adding constraints is easy, but removing constraints is hard.

Different options:

- Keep everything and ignore removals.
  Constraints accumulate and CAD eventually blows up.
- Reset whenever we remove something.
  Mostly destroys the idea of retaining state.
- Save a snapshot once in a while and restore it.
  When to snapshot? How many to keep? Memory usage?
- Figure out which polynomials to remove and update properly.
  How to do that?

RWTH AACHEN UNIVERSITY

Backtracking in Projection

Add $z^2 + y^2 + x^2 < 4$

$$z^2 + y^2 + x^2 < 4$$

z

y

x

Backtracking in Projection

Add $z^2 + y^2 + x^2 - 4$

$$z^2 + y^2 + x^2 < 4$$

z	$z^2 + y^2 + x^2 - 4$
y	
x	

Backtracking in Projection

Project $z^2 + y^2 + x^2 - 4$

$$z^2 + y^2 + x^2 < 4$$

z: $z^2 + y^2 + x^2 - 4$

y: $y^2 + x^2 - 4$

x:

RWTH AACHEN UNIVERSITY

Backtracking in Projection

Project $y^2 + x^2 - 4$

$$z^2 + y^2 + x^2 < 4$$

z: $z^2 + y^2 + x^2 - 4$

↓

y: $y^2 + x^2 - 4$

↓

x: $x^2 - 4$

RWTH AACHEN UNIVERSITY

Backtracking in Projection

Add $z^2 < 1$

$z^2 < 1$ $\qquad$ $z^2 + y^2 + x^2 < 4$

z: $z^2 + y^2 + x^2 - 4$

↓

y: $y^2 + x^2 - 4$

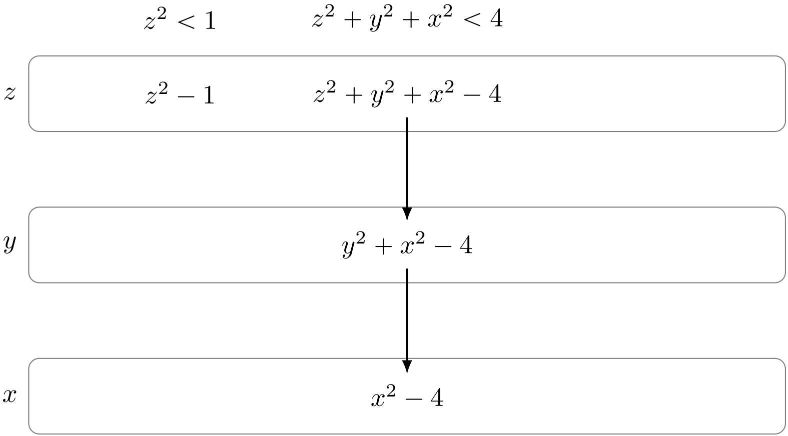
↓

x: $x^2 - 4$

RWTH AACHEN UNIVERSITY

Backtracking in Projection

Add $z^2 - 1$

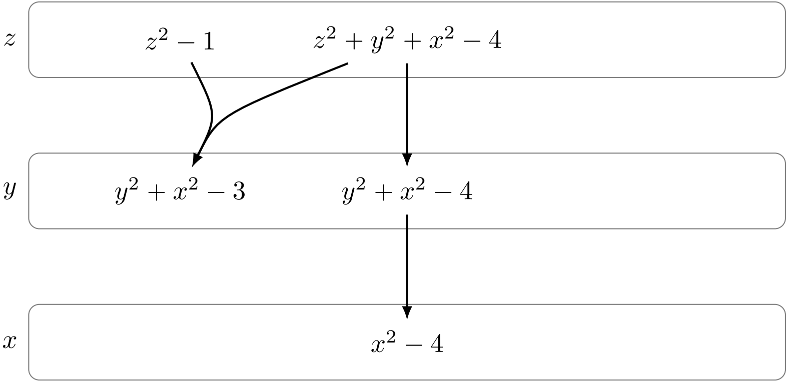


RWTH AACHEN UNIVERSITY

Backtracking in Projection

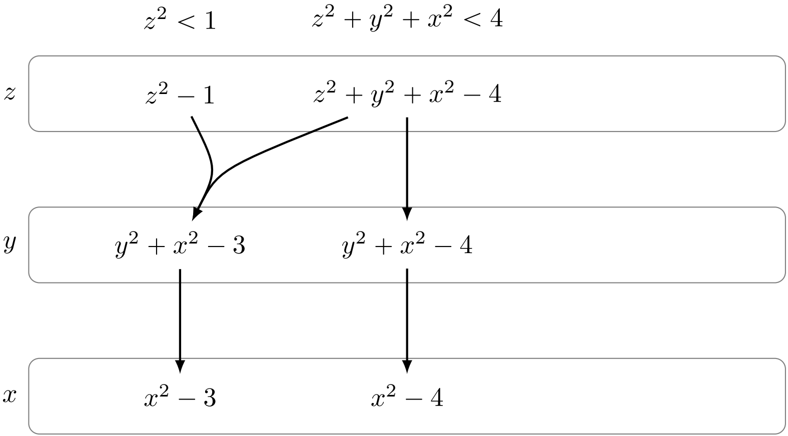
Project $z^2 - 1$ and $z^2 + y^2 + x^2 - 4$

$z^2 < 1$ $\quad$ $z^2 + y^2 + x^2 < 4$



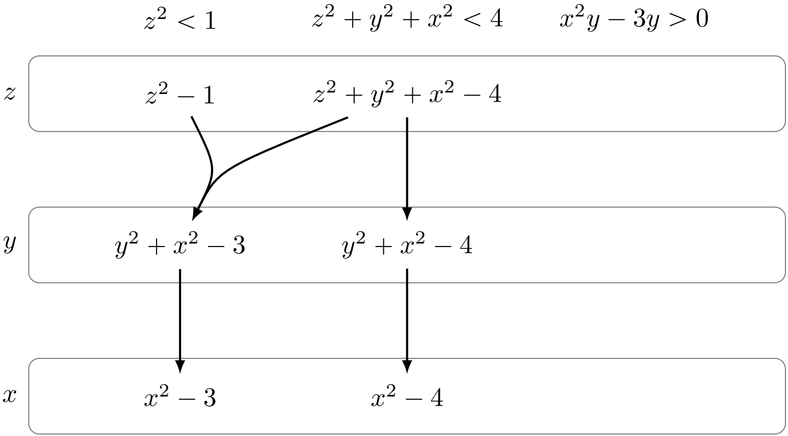
Backtracking in Projection

Project $y^2 + x^2 - 3$



Backtracking in Projection

Add $x^2y - 3y > 0$

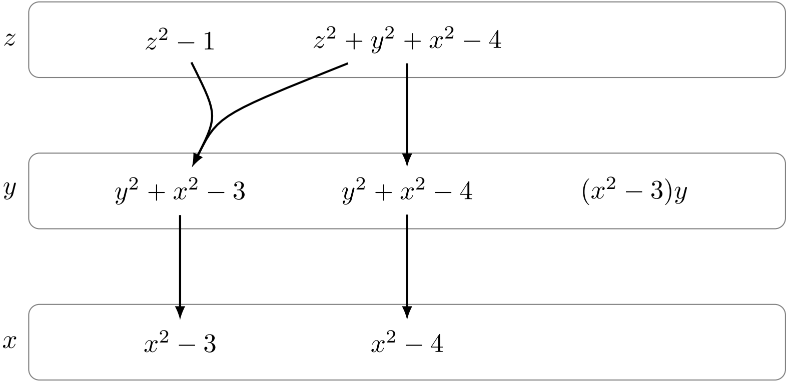


RWTH AACHEN UNIVERSITY

Backtracking in Projection

Add $(x^2 - 3)y$

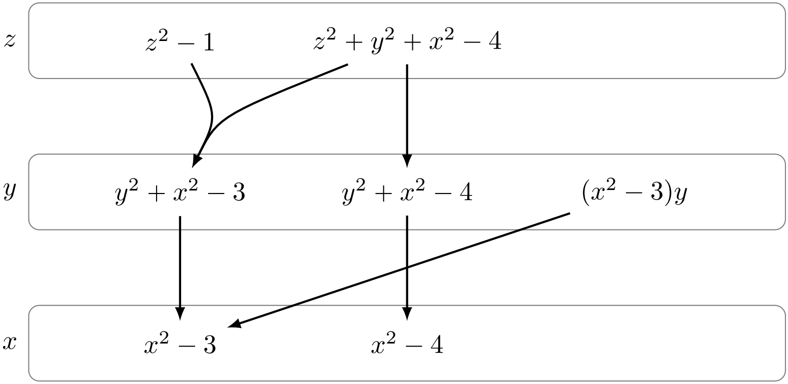
$z^2 < 1$ $\quad$ $z^2 + y^2 + x^2 < 4$ $\quad$ $x^2y - 3y > 0$



Backtracking in Projection

Project $(x^2 - 3)y$

$z^2 < 1$ $\quad z^2 + y^2 + x^2 < 4$ $\quad x^2y - 3y > 0$

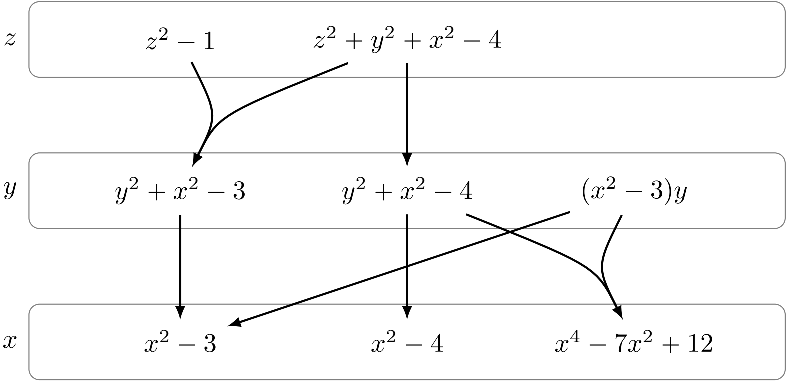


RWTH AACHEN UNIVERSITY

Backtracking in Projection

Project $y^2 + x^2 - 4$ and $(x^2 - 3)y$

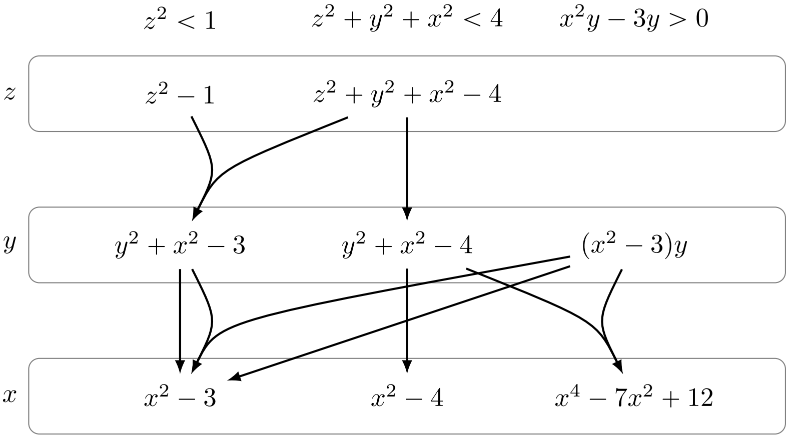
$z^2 < 1$ $\quad$ $z^2 + y^2 + x^2 < 4$ $\quad$ $x^2y - 3y > 0$



RWTH AACHEN UNIVERSITY

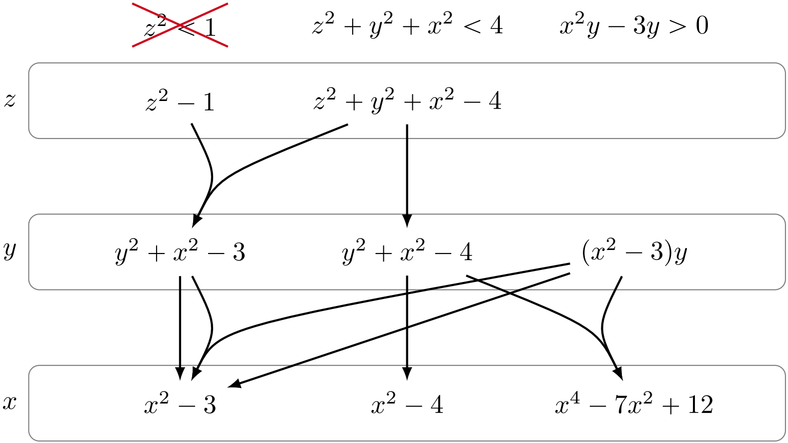
Backtracking in Projection

Project $y^2 + x^2 - 3$ and $(x^2 - 3)y$



Backtracking in Projection

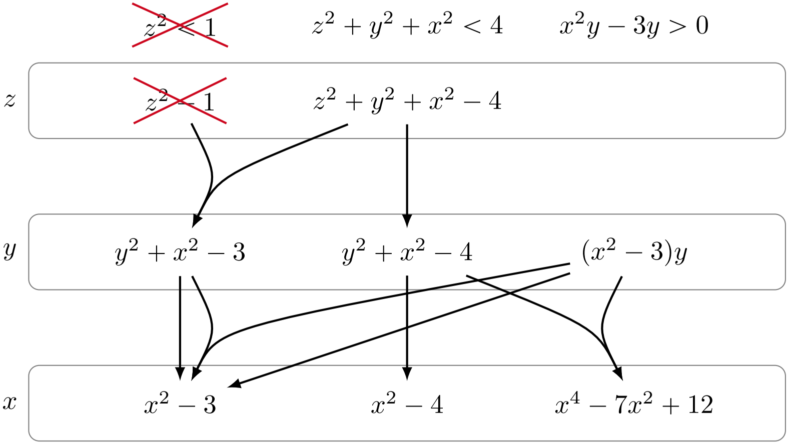
Remove $z^2 < 1$



RWTH AACHEN UNIVERSITY

Backtracking in Projection

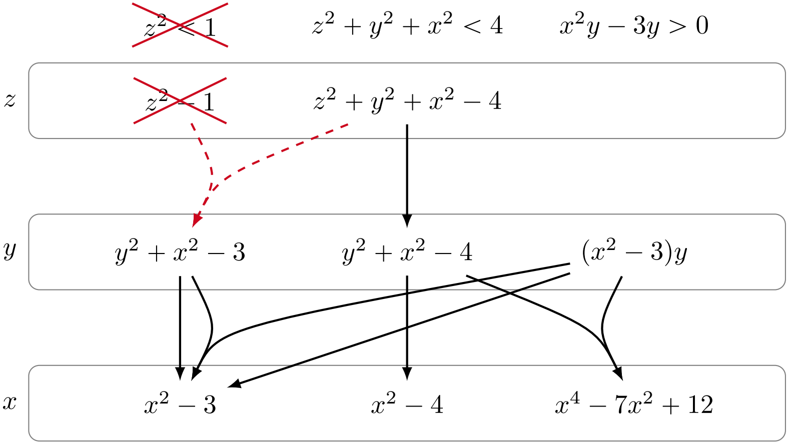
Remove $z^2 - 1$



RWTH AACHEN UNIVERSITY

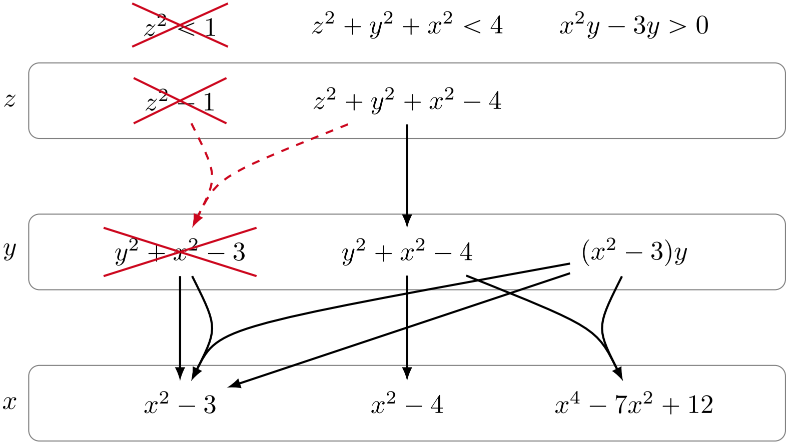
Backtracking in Projection

Remove $z^2 - 1$



Backtracking in Projection

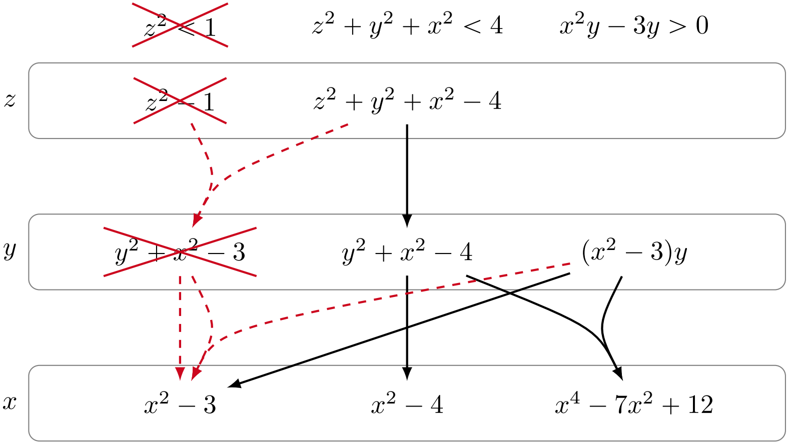
Remove $y^2 + x^2 - 3$



RWTH AACHEN UNIVERSITY

Backtracking in Projection

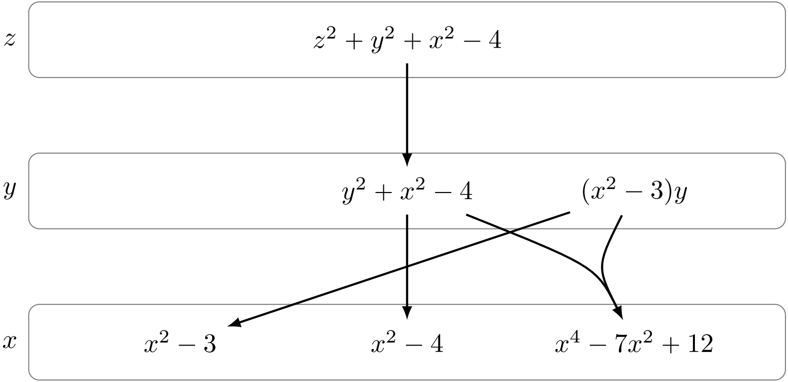
Remove $y^2 + x^2 - 3$



RWTH AACHEN UNIVERSITY

Backtracking in Projection

$$z^2 + y^2 + x^2 < 4 \qquad x^2y - 3y > 0$$



Pruning the lifting tree

Prune lifting after polynomials are removed.

RWTH AACHEN UNIVERSITY

Pruning the lifting tree

Prune lifting after polynomials are removed.

Observations:

- A sample is either
  - a root of one or more polynomial(s) or
  - a value in-between two roots.
- We store the reasons for a root as a set of polynomials.

Pruning the lifting tree

Prune lifting after polynomials are removed.

Observations:

- A sample is either
  - a root of one or more polynomial(s) or
  - a value in-between two roots.
- We store the reasons for a root as a set of polynomials.

To prune:

- Remove a root if the reasons are gone.
- Remove one of the neighboring samples with every root.

RWTH AACHEN UNIVERSITY

Evaluation

Is it worth it?

Experiments

- Benchmarks from SMT-LIB `QF_NRA` (11354 from 10 sources)
- Only `SAT` + `CAD` with different options:

Experiments

- Benchmarks from SMT-LIB `QF_NRA` (11354 from 10 sources)
- Only `SAT` $+$ `CAD` with different options:
  - CAD_{Naive}: Fresh CAD on every theory call
  - CAD_{Eager}: Eagerly compute full projection
  - CAD_{Simple}: Eagerly add one constraint at a time
  - CAD_{Full}: Incremental projection

RWTH AACHEN UNIVERSITY

Experiments

- Benchmarks from SMT-LIB `QF_NRA` (11354 from 10 sources)
- Only `SAT` + `CAD` with different options:
  - CAD_{Naive}: Fresh CAD on every theory call
  - CAD_{Eager}: Eagerly compute full projection
  - CAD_{Simple}: Eagerly add one constraint at a time
  - CAD_{Full}: Incremental projection

Solver	**solved**		**runtime**
CAD_{Naive}	5571	49.1 %	0.69
CAD_{Eager}	7559	66.6 %	0.60
CAD_{Simple}	7924	69.8 %	1.11
CAD_{Full}	8158	71.9 %	1.22

RWTH AACHEN UNIVERSITY

Conclusion

- Consider CAD as search method for a satisfying solution.
- Perform projection and lifting incrementally.

RWTH AACHEN UNIVERSITY

Conclusion

- Consider CAD as search method for a satisfying solution.
- Perform projection and lifting incrementally.

- Queues allow for easy continuation.
- Track reasons for polynomials and samples for removal.

Conclusion

- Consider CAD as search method for a satisfying solution.
- Perform projection and lifting incrementally.

- Queues allow for easy continuation.
- Track reasons for polynomials and samples for removal.

- Very beneficial for practical solving.
- More details in [KA18]

Further topics

- Factorization of polynomials?
  Integrates easily, only slight improvement
- Equational constraints?
  Somewhat tricky, only slight improvements [Hae17, HKÁ18]
- Impact of different heuristics?
  Surprisingly small, as long as we exploit incrementality
- Delineating polynomials?
  Integrates easily, somewhat obsolete (?)
- Generic quantifier elimination?
  Disable early abort and obtain a full CAD.
- Implementation?
  Bookkeeping is somewhat involved, but not to bad. See SMT-RAT!

RWTH AACHEN UNIVERSITY

References

[Bro01] Christopher Brown. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447 – 465, 2001.

[CH91] George Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[Col75] George Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern*, pages 134–183, 1975.

[Hae17] Rebecca Haehn. Using equational constraints in an incremental CAD projection. Master's thesis, RWTH Aachen University, 2017.

[HKÁ18] Rebecca Haehn, Gereon Kremer, and Erika Ábrahám. Evaluation of equational constraints for cad in smt solving. In *SC^2@ FLoC*, 2018.

[JDF15] Maximilian Jaroschek, Pablo Federico Dobal, and Pascal Fontaine. Adapting real quantifier elimination methods for conflict set computation. In *Frontiers of Combining Systems*, pages 151–166, 2015.

[KA18] Gereon Kremer and Erika Ábrahám. Fully incremental cad. *Journal of Symbolic Computation (submitted)*, 2018.

[McC98] Scott McCallum. An improved projection operation for cylindrical algebraic decomposition. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. 1998.