Research Work Presentation

Nonlinear Arithmetic Solving

Zhonghan Wang

Institute of Software, Chinese Academy of Sciences

March 26, 2024

Overview

Zhonghan Wang

SMT Solving

SMT-NRA helps in many areas

- Nonlinear hybrid automata
- Generating ranking function for termination analysis (LassoRanker Benchmark)
- Constraint Programming Solving
- Automatic or interactive theorem prover (Isabelle or Coq)
- Biological networks
- ......

Syntax of SMT(NRA)

- polynomial: $p ::= x \mid c \mid p + p \mid p - p \mid p \times p$
- atoms: $a ::= b \mid p = 0 \mid p > 0 \mid p < 0$
- formula: $f ::= a \mid \neg f \mid f \wedge f \mid f \vee f$

SMT: Determine whether the formula is satisfied by some assignment (local search focuses), or prove unsat

Example

$x^2 + y^2 \leq 1 \wedge x + y < 1 \wedge x + z > 0$
assignment with $\{x \to 0, y \to 0, z \to 1\}$ satisfies all clauses.

Satisfiability Modulo Nonlinear Arithmetic

- Decision Procedure for Nonlinear Arithmetic
  - Complete Method
    - CDCL(T)
    - MCSAT/NLSAT
    - Math Method for CAD
      - New Projection Order
      - Lift Structure
  - InComplete Method
    - Interval Constraint Propagation (ICP)
    - Incremental Linearization
    - Subtropical
    - Local Search
    - DPLL(T) + Symbolic Verification

My Contribution in Satisfiability Modulo Theory

- Tool for Competition
  - Portfolio in Z3-Plus-Plus (smt-comp 2022 & 2023)
- Incomplete Method
  - Local Search on Nonlinear Arithmetic
- Z3 Nlsat Solver
  - Dynamic Variable Order in NLSAT
  - Clause-level semantics decision in NLSAT

Local Search for Nonlinear Arithmetic

Zhonghan Wang

Fragment of Local Search

```
Input   : A set of clauses F
Output: An assignment of variables that satisfy F, or failure
Initialize assignment to variables;
while ⊤ do
    if all clauses satisfied then
        return success with assignment;
    end
    if time or step limit reached then
        return failure;
    end
    Critical move procedure.
end
```

Algorithm 1: Basic Fragment of Local Search

Fragment of Local Search

$var, new\_value, score \leftarrow$ best move according to make-break score;
if *score* $>$ *0* **then**
    Perform move, assigning var to new_value;
end
else
    Update clause weight according to PAWS scheme;
    $var, new\_value, score \leftarrow$ critical move making random unsat clause satisfied;
    if *score* $\neq -\infty$ **then**
        Perform move, assigning var to new_value;
    end
    if *no move performed in previous loop* **then**
        Change assignment of some variable in some unsatisfied clause;
    end
end

Local Search for SAT and SMT[1]

LS \ Problem	SAT	SMT
Operation (Move)	Flip	Critical Move
Score Definition	Weighted unsat clauses	
Score Computation	Cached score	No Caching, time costly

- What LS for SAT brings us:
  - Maintain scoring information after each iteration.
- Difficulty:
  - Predetermine critical move shift value.
- Our Solution
  - Introduce Scoring Boundaries

Infeasible Set

Definition

infeasible set of a clause *c* with respect to an assignment $asgn$ is the set of values that the variables in *c* can take under $asgn$ such that *c* is unsatisfied.

Example

Current assignment: $\{x \mapsto 1\}$
Calculate infeasible set for *y*:

- $x^2 + y^2 \leq 1 : (-\infty, 0) \cup (0, \infty)$.
- $x + y < 1 : [0, \infty)$.

If we choose values from infeasible set, the satisfied clause will be unsatisfied, which changes the whole score.

Make-break Intervals

Definition

make-break interval is a combination of (in)feasible intervals of arithmetic varaible x with respect to **all clauses**.

Example

Current assignment: $\{x \mapsto 1, y \mapsto 1, z \mapsto 1\}$
Calculate infeasible set for each clause.

- $x^2 + y^2 \leq 1$ (unsat): $(-\infty, 0) \cup (0, \infty)$.
- $x + y < 1$ (unsat): $[0, \infty)$.
- $x + z > 0$ (sat): $(-\infty, -1]$.

Combined information: x: $(-\infty, -1] \mapsto 0$, $(-1, 0) \mapsto 1$, $[0, 0] \mapsto 1$, $(0, \infty) \mapsto 0$.

Traditional Computation

```
Input   : unsat clauses F
Output: Best critical move (variable, value)
foreach variable v in unsat clauses do
    foreach unsat clause c with v do
        Compute interval-score info of v in c.
    end
    Combine interval-score information.
    Update best var-value move.
end
return best critical move
```

Repeated computation:

- variable's (in)feasible set
- clause's sat staus

Boundary

Definition. A quadruple $\langle val, is\_open, is\_make, cid\rangle$, where val is a real number, is_open and is_make are boolean values, and cid is a clause identifier.

Meaning

- *val* : make-break value.
- *is_open* : active or not at *val* point.
- *is_make* : make or break, increase or decrease score.
- *cid* : causing clause.

Sorting: First ordered by *val*, then by *is_open* ($\perp < \top$).

Boundary

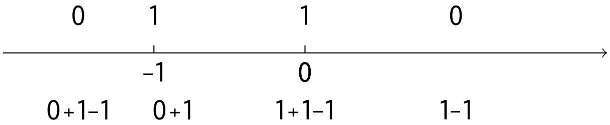
Current assignment: $\{x \mapsto 1, y \mapsto 1, z \mapsto 1\}$

- $x^2 + y^2 \leq 1$: starting score 0, boundary set $\{(0, \bot, \top, 1), (0, \top, \bot, 1)\}$, indicating no change for large negative values, *make* at boundary $[0, \cdots$, followed by *break* at boundary $(0, \cdots$.
- $x + y < 1$: starting score 1, boundary set $\{(0, \bot, \bot, 1)\}$, indicating *make* at large negative values, and *break* at boundary $[0, \ldots$.
- $x + z > 0$: starting score -1, boundary set $\{(-1, \top, \top, 1)\}$, indicating *break* at large negative values, and *make* at boundary $(-1, \ldots$.

sorted boundary set: $\{(-1, \top, \top, 1), (0, \bot, \top, 1), (0, \bot, \bot, 1), (0, \top, \bot, 1)\}$

Boundary Example

boundary set: $\{(-1, \top, \top, 1), (0, \bot, \top, 1), (0, \bot, \bot, 1), (0, \top, \bot, 1)\}$



Starting score: Score when *x* moves to $-\infty$.
Maintain and Change: We maintain the boundary info for all arithmetic varaibles, unless the neighbour does a critical move.

Algorithm for computing boundary

Input : Variable *v* that is modified
Output: Make-break score for all variables

$S \leftarrow \{\}$; `// set of updated variables`

for *clause* cls *that contains v* **do**
    for *variable v′ appearing in* cls **do**
        add *v′* to *S*;
        recompute starting score and boundary of *v′* with respect to cls;
    end
end
for *variable v′ in S* **do**
    recompute best critical move and score in terms of boundary information;
end

Complexity of Values[3]

Definition

We define a preorder $\prec_c$ on algebraic numbers as follows. $x \prec_c y$ if x is rational and y is irrational, or if both x and y are rational numbers, and the denominator of x is less than that of y. We write $x \sim_c y$ if neither $x \prec_c y$ nor $y \prec_c x$.

Previous work ignores equalities constraints, or only consider multi-linear (one-degree)examples[2].
Our Solution: Introducing relaxation, temporarily enlarge the point irrational interval

Relaxation

Example

Given assignment $\{x \mapsto 1, y \mapsto 1\}$

$$z^3 \geq 5x^2 + y \vee z^3 \leq 3x + 3y$$

$$z^2 = x^2 + y^3$$

Both situations force z to an irrational number.

Relaxation

- If the constraint is of the form $p = 0$, it is relaxed into the pair of inequalities $p < \epsilon_p$ and $p > -\epsilon_p$.
- If the constraint is of the form $p \geq 0$, it is relaxed into $p > -\epsilon_p$. Likewise, if the constraint is of the form $p \leq 0$, it is relaxed into $p < \epsilon_p$.
- **Slacked var:** the var that is being assigned.

Restore

Input : slacked clauses
Output: succeed or not
for *each slacked clause* cls **do**
  v $\leftarrow$ slacked variable in cls;
  accu_val $\leftarrow$ *inf_set*(*cls*);
  move *v* to *accu_val*;
end
for *variable v′ in slacked clauses* **do**
  recompute best critical move and score in terms of boundary information;
end
return *all clauses satisfied by accurate value*

Algorithm 3: Lazy restore mechanism

Relaxation and Restore Demo

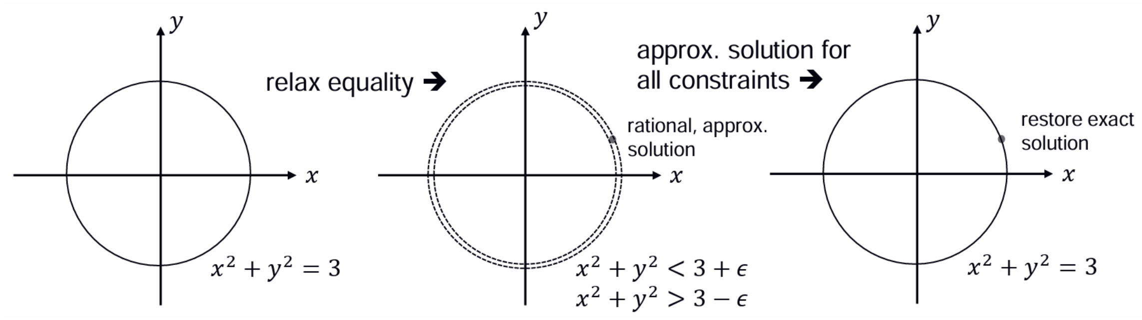


Figure: Relaxation and Restore

Local Search with Relaxation

Input : A set of clauses *F*
Output: Success or failure
Initialize assignment to variables;
while ⊤ **do**
  if *all clauses satisfied* **then**
    check relaxation assignment
  end
  if *time or step limit reached* **then**
    return *failure*;
  end
  Proceed traditional local search (slack).
end

Algorithm 4: Local Search with Relaxation

Input : formula *F*, assignment *ass*
Output: Success or failure
if *find exact solution* **then**
  return *success with assignment*;
end
else
  Restore relaxed constraints to original form;
  if *find exact solution by limited local search* **then**
    return *success with assignment*;
  end
end

Algorithm 5: Check relaxation assignment

Implementation Detail

code available at: `https://github.com/yogurt-shadow/LS_NRA`

Preprocessing

- Combine constraints $p \geq 0$ and $p \leq 0$ into equality $p = 0$.
- Eliminate variable x in an equation of the form $c \cdot x + q = 0$, where c is a constant and q is a polynomial with degree at most 1 and containing at most 2 variables.

Restart mechanism Two-level restart mechanism with two parameters $T_1 = 100$ and $T_2 = 100$.

- **Minor restart:** randomly change one of the variables in one of the unsatisfied clauses.
- **Major restart:** reset the value of all variables.

Overall Result

Category	#inst	Z3	cvc5	Yices	Ours	Unique
20161105-Sturm-MBO	120	0	0	0	**88**	88
20161105-Sturm-MGC	2	**2**	0	0	0	0
20170501-Heizmann	60	3	1	0	**8**	6
20180501-Economics-Mulligan	93	**93**	89	91	90	0
2019-ezsmt	61	**54**	51	52	19	0
20200911-Pine	237	**235**	201	**235**	224	0
20211101-Geogebra	112	**109**	91	99	101	0
20220314-Uncu	74	73	66	**74**	70	0
LassoRanker	351	155	**304**	122	272	13
UltimateAtomizer	48	**41**	34	39	27	2
hycomp	492	**311**	216	227	304	11
kissing	42	**33**	17	10	**33**	1
meti-tarski	4391	**4391**	4345	4369	4351	0
zankl	133	70	61	58	**100**	27
Total	6216	5570	5476	5376	**5687**	148

Scatter Plot

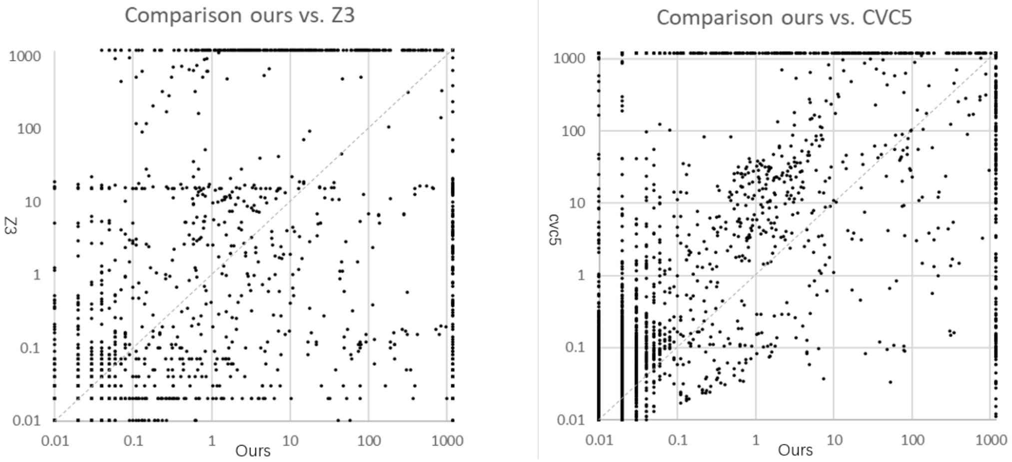


Figure: Scatter plots of running time vs. Z3[4] and cvc5[5].

Category	#inst	Incremental	Naive	Limit-45
20161105-Sturm-MBO	120	88	85	85
20161105-Sturm-MGC	2	0	0	0
20170501-Heizmann	60	8	5	5
20180501-Economics-Mulligan	93	90	89	89
2019-ezsmt	61	19	19	15
20200911-Pine	237	224	222	222
20211101-Geogebra	112	101	101	101
20220314-Uncu	74	70	70	70
LassoRanker	351	272	264	269
UltimateAtomizer	48	27	26	26
hycomp	492	304	298	298
kissing	42	33	32	33
meti-tarski	4391	4351	4352	4352
zankl	133	100	100	100
Total	6216	5687	5663	5665

Table: Comparison of incremental computation

Category	#inst	Relaxation	Threshold	NoOrder
20161105-Sturm-MBO	120	88	100	99
20161105-Sturm-MGC	2	0	0	0
20170501-Heizmann	60	8	9	3
20180501-Economics-Mulligan	93	90	89	86
2019-ezsmt	61	19	19	19
20200911-Pine	237	224	223	222
20211101-Geogebra	112	101	98	92
20220314-Uncu	74	70	70	70
LassoRanker	351	272	277	278
UltimateAtomizer	48	27	26	20
hycomp	492	304	211	164
kissing	42	33	31	27
meti-tarski	4391	4351	4353	4360
zankl	133	100	100	100
Total	6216	5687	5606	5540

Table: Comparison of temporary relaxation of constraints

Improved NLSAT

Zhonghan Wang

Introduction of NLSAT[7, 8]

Difference with DPLL(T)[6]

- handle both boolean level and semantics decision
- Integrate theory algorithms into DPLL and CDCL
- $decide \rightarrow semanticsdecide$
- $unit - propagate \rightarrow R - propagate$

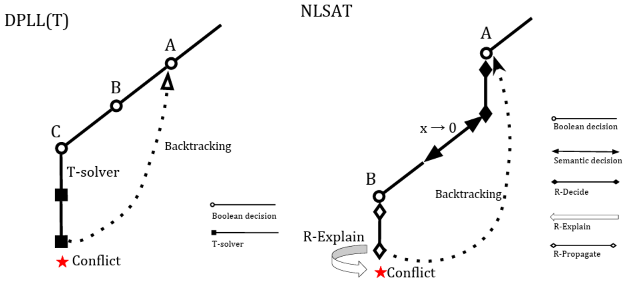


Figure: Difference between DPLL(T) and NLSAT

Example of NLSAT (1)

Example

$$\begin{cases} x^2 - y^2 > 0 \\ x^2 + xz > 10 \end{cases}$$

set for x	select witness	update set for y	select witness	update set for z	select witness
{ }	x -> 0.125	{(-oo, -0.125], [0.125, p1, oo)}	y -> 0	{(-oo, 79.875]}	z -> 81

Resolve: CDCL(T) + Explain[9]

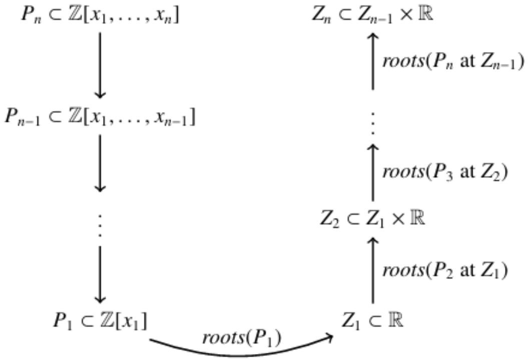


Figure: Structure of CAD

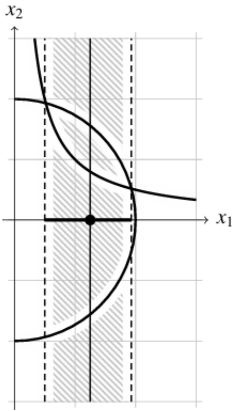


Figure: Cylinder of CAD

Example of NLSAT(2)

Example

$$\begin{cases} x_1^2 + x_1x_3 \geq 10 \\ x_2^2 + x_2x_3 \leq 12 \\ x_4^2 + x_3x_4 \leq 8 \end{cases}$$

Conflict Status:

$$\left.\begin{array}{c} x_1 \to 0.125 \\ x_3 \to 0.5 \\ x_4 \to -0.5 \end{array}\right\} \Rightarrow \bigvee \begin{cases} x_1^4 + 28x_1^2 + 100 \leq 0 \\ x_1 \leq 0 \\ 2x_1x_3 - x_1^2 + 10 \leq 0 \\ x_1x_3^2 - x_1^2x_3 + 10x_3 - 12x_1 \leq 0 \end{cases}$$

Dynamic Ordering of NLSAT

- Using VSIDS and LRB branching heuristic in mcsat framework, instead of static ordering
- what means dynamic: decide branching variable using state information
- Using reverse order of assigned variables for cylindrical algebraic decomposition
- dynamic clause learning: remove useless clauses after each restart

Code available: `https://github.com/yogurt-shadow/z3_dnlsat`

solver	**solved**	**unsat**	**sat**	**unsolved**
z3_nlsat	10730	5546	5184	1404
dnlsat_v1	10883	5611	5272	1251
dnlsat_v2	10967	5612	5355	1167

Table: **Comparison of dynamic and static variable order**

Semantic Decision in NLSAT

Input : current processing variable *v*
for $cls \in watches$ **do**
  for $lit \in cls$ **do**
    try propagate literal lit;
  end
  if *all false* **then**
    label conflict;
  else if *one unassigned* **then**
    unit propagate;
  else
    semantics decide first unassigned literal;
end

Algorithm 6: Semantics Decision in NLSAT

Semantic Decision Conflict

Example

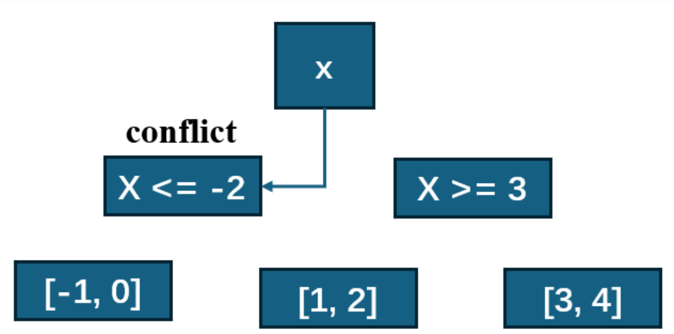


Figure: Traditional decision strategy causes conflict

Clause-Level Infeasible Set

Definition

Clause Level Infeasible set an arithmetic variable v under assignment $asgn$ is the set of values that can satisfy all clauses $clauses$ where v is the last varaible to assign.

$$clause\_set(v) = \bigcup_{c \in watches} \left(\bigcap_{l \in c} inf\_set(l, v) \right) \tag{1}$$

Relationship between clause set and decision path

Consistent Decision Path for variable v exists $\Leftrightarrow$ clause_set(v) not full

Improved Semantic Decision

```
Input    : current processing variable v
clause_set ← compute_clause_set(v);
if clause_set is full then
│  resolve_conflict();
end
else
│  val ← choose_value(clause_set, v);
│  for each clause cls ∈ watches do
│  │  decide first satisfied literal with val;
│  end
│  select_witness(v, val);
end
```

Algorithm 7: Semantics Decision with Look-ahead

Example of Improved Semantic Decision

Example

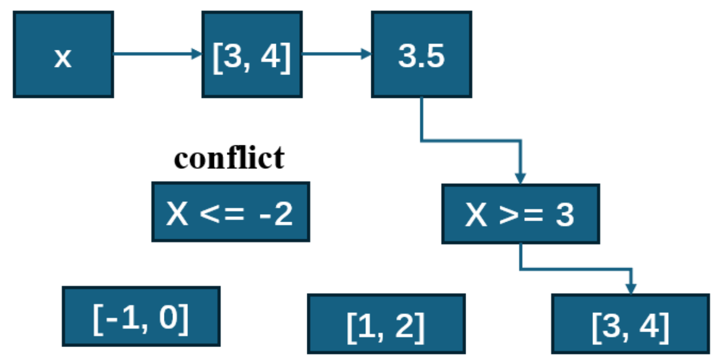


Figure: Improved Semantic Decision: Consistent Path

Portfolio of Nonlinear Arithmetic in Z3-Plus-Plus

Zhonghan Wang

Z3-Plus-Plus webpage

z3-plus-plus.github.io

View My GitHub Profile

Hosted on GitHub Pages — Theme by orderedlist

Z3++

Overview

Z3++ is a derived SMT solver based on Z3. It participates in the SMT-COMP 2022, and significantly improves Z3 on the following logics:

QF_IDL, QF_LIA, QF_BV, QF_NIA and QF_NRA

It is a project mainly developed in State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China.

Detailed description and source code are available at the github repository.

Contact

z3_plus_plus@outlook.com

Awards

At the FLoC Olympic Games, Z3++ won 2 gold medals (6 in total) for Biggest Lead Model Validation and Largest Contribution Model Validation.

People

Leader:

Shaowei Cai.

Figure: https://z3-plus-plus.github.io/

z3pp Overview

Portfolio in Z3-Plus-Plus QF_NRA

- Heuristic for Static Variable Order
- Simple Interval Constraint Checker
- Sample-Cell Projection
- Symmetry

Code Structure

- nlsat_variable_ordering_strategy.cpp
- nlsat_simple_checker.cpp
- nlsat_symmetry_checker.cpp
- nlsat_explain.cpp

Portfolio of Z3pp: static variable ordering

- Heuristic variable ordering of nlsat (nlsat_variable_ordering_strategy.cpp)
  - number of univariate polynomials
  - max degree of variable
  - `BROWN`: max degree, max degree of total terms, number of terms containing the variable
  - `TRIANGULAR`: max degree, max leading coefficient degree, sum of degree

Portfolio of Z3pp: Interval Constraint Propagation

- Target Instances: MBO - Methylene Blue Oscillator System
  formula:

$$(\bigwedge x_i > 0) \wedge p(x_1, x_2, ..., x_i) = 0$$

- Whether certain polynomial has a zero where all variables are positive.
- Example:

$$f := h1 > 0 \wedge h2 > 0 \wedge h3 > 0 \wedge h1^3 + 2h1h2 + h3^4 = 0$$

- Implementation:

$$\left.\begin{array}{c} h1 > 0 \rightarrow h1^3 > 0 \\ 2\ h1 > 0 \wedge h2 > 0 \rightarrow h1h2 > 0 \\ h3 > 0 \rightarrow h3^4 > 0 \end{array}\right\} \Rightarrow h1^3 + 2h1h2 + h3^4 > 0$$

Portfolio of Z3pp: symmetry

Instance: Hong (fully symmetry)

Hong_n

$$\exists x_1, ..., \exists x_n \sum_{i=1}^{n} x_i^2 < 1 \wedge \Pi_{i=1}^{n} x_i > 1$$

Hong_2n

$$\exists x_1, ..., \exists x_n \sum_{i=1}^{n} x_i^2 < 2n \wedge \Pi_{i=1}^{n} x_i > 1$$

Insert ordering clauses for variables: If x, y, z are symmetry, insert new clause

$$x \leq y \leq z$$

Portfolio of Z3pp: sample cell projection

Definition

sample-cell projection Suppose a is a sample of x in R^n and $F = \{f_1, ..., f_r\}$ is a polynomial set in $Z[x]$. The sample cell projection of F on x_n at a is

$$proj_{sc}(F, x_n, a) = \bigcup_{f \in F} s\_coeff(f, x_n, a) \cup \bigcup_{f \in F} disc(F, x) \cup \bigcup_{f \in F, g \in s\_poly(F,x,a), f \neq g} res(f, g, x)$$

- difference from McCallum's projection: calculate resultant only between sample polynomials
- sample polynomials: one or two polynomials whose root is the closest to the assignment point

Portfolio of Z3pp: sample polynomials

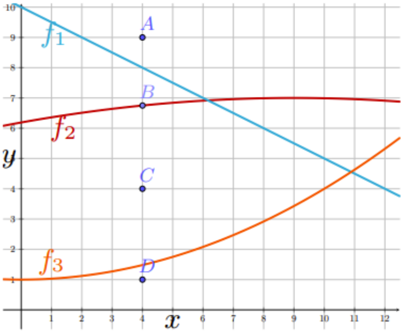


Figure: Demo for sample polynomial

Z3pp: competition result on SMT_LIB (QF_NRA)

Sequential Performance

Solver	Error Score	Correct Score	CPU Time Score	Wall Time Score	Solved	Solved SAT	Solved UNSAT	Unsolved	Abstained	Timeout	Memout
Z3++-fixed[n]	0	2641	379531.82	379433.129	2641	1340	1301	267	0	265	0
2019-Par4[n]	0	2629	394912.029	356695.171	2629	1292	1337	279	0	221	58
cvc5	0	2545	525901.735	526314.738	2545	1244	1301	363	0	363	0
NRA-LS	0	2488	550489.833	551413.565	2488	1198	1290	420	0	5	0
Yices2	0	2341	702255.323	702324.97	2341	1150	1191	567	0	567	0
z3-4.8.17[n]	0	2275	666874.65	666955.286	2275	1229	1046	633	0	499	0
SMT-RAT-MCSAT 22.06	0	2189	895361.649	895423.466	2189	1123	1066	719	0	674	21
veriT+raSAT+Redlog	0	1879	1206512.928	1206107.221	1879	905	974	1029	0	989	0
MathSAT[n]	0	1544	1671561.013	1671677.835	1544	417	1127	1364	0	1364	0
Z3++	6	2634	379866.348	379759.488	2634	1333	1301	274	0	264	1

Figure: https://tools-comp.github.io/2022/results/qf-nonlinearrealarith-single-query

Z3pp: competition result on SMT_LIB (QF_NRA)

Parallel Performance

Solver	Error Score	Correct Score	CPU Time Score	Wall Time Score	Solved	Solved SAT	Solved UNSAT	Unsolved	Abstained	Timeout	Memout
2019-Par4^n	0	2650	412116.989	346590.821	2650	1310	1340	258	0	200	58
Z3++-fixed^n	0	2641	379553.38	379423.299	2641	1340	1301	267	0	265	0
cvc5	0	2545	526363.395	526298.488	2545	1244	1301	363	0	363	0
NRA-LS	0	2488	550607.043	551413.405	2488	1198	1290	420	0	5	0
Yices2	0	2341	702330.553	702302.83	2341	1150	1191	567	0	567	0
z3-4.8.17^n	0	2275	666962.06	666934.046	2275	1229	1046	633	0	499	0
SMT-RAT-MCSAT 22.06	0	2189	895429.739	895399.226	2189	1123	1066	719	0	674	21
veriT+raSAT+Redlog	0	1879	1206582.328	1206082.811	1879	905	974	1029	0	989	0
MathSAT^n	0	1544	1671701.033	1671625.855	1544	417	1127	1364	0	1364	0
Z3++	6	2634	379887.798	379749.928	2634	1333	1301	274	0	264	1

Figure: https://tools-comp.github.io/2022/results/qf-nonlinearrealarith-single-query

References I

[1] Shaowei Cai, Bohan Li, and Xindi Zhang. "Local Search for SMT on Linear Integer Arithmetic". In: *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*. Ed. by Sharon Shoham and Yakir Vizel. Vol. 13372. Lecture Notes in Computer Science. Springer, 2022, pp. 227–248. DOI: `10.1007/978-3-031-13188-2\_12`. URL: `https://doi.org/10.1007/978-3-031-13188-2%5C_12`.

[2] Haokun Li, Bican Xia, and Tianqi Zhao. "Local Search for Solving Satisfiability of Polynomial Formulas". In: *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II*. Ed. by Constantin Enea and Akash Lal. Vol. 13965. Lecture Notes in Computer Science. Springer, 2023, pp. 87–109. DOI: `10.1007/978-3-031-37703-7\_5`. URL: `https://doi.org/10.1007/978-3-031-37703-7%5C_5`.

References II

[3] Bohan Li and Shaowei Cai. “Local Search For SMT On Linear and Multilinear Real Arithmetic”. In: *CoRR* abs/2303.06676 (2023). accepted for FMCAD. DOI: 10.48550/arXiv.2303.06676. arXiv: 2303.06676. URL: https://doi.org/10.48550/arXiv.2303.06676.

[4] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. “Z3: An Efficient SMT Solver”. In: *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Vol. 4963. Lecture Notes in Computer Science. Springer, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24. URL: https://doi.org/10.1007/978-3-540-78800-3%5C_24.

References III

[5] Haniel Barbosa et al. “cvc5: A Versatile and Industrial-Strength SMT Solver”. In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*. Ed. by Dana Fisman and Grigore Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. DOI: 10.1007/978-3-030-99524-9_24. URL: https://doi.org/10.1007/978-3-030-99524-9%5C_24.

[6] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. “Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T)”. In: *J. ACM* 53.6 (2006), pp. 937–977. DOI: 10.1145/1217856.1217859. URL: https://doi.org/10.1145/1217856.1217859.

References IV

[7] Dejan Jovanovic and Leonardo Mendonça de Moura. "Solving Non-linear Arithmetic". In: *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*. Ed. by Bernhard Gramlich, Dale Miller, and Uli Sattler. Vol. 7364. Lecture Notes in Computer Science. Springer, 2012, pp. 339–354. DOI: 10.1007/978-3-642-31365-3_27. URL: https://doi.org/10.1007/978-3-642-31365-3%5C_27.

References V

[8] Leonardo Mendonça de Moura and Dejan Jovanovic. "A Model-Constructing Satisfiability Calculus". In: *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*. Ed. by Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni. Vol. 7737. Lecture Notes in Computer Science. Springer, 2013, pp. 1–12. DOI: 10.1007/978-3-642-35873-9_1. URL: https://doi.org/10.1007/978-3-642-35873-9%5C_1.

[9] Gereon Kremer. "Cylindrical algebraic decomposition for nonlinear arithmetic problems". PhD thesis. RWTH Aachen University, Germany, 2020. URL: https://publications.rwth-aachen.de/record/792185.

Thank you for your attention

Zhonghan Wang

Institute of Software, Chinese Academy of Sciences

March 26, 2024