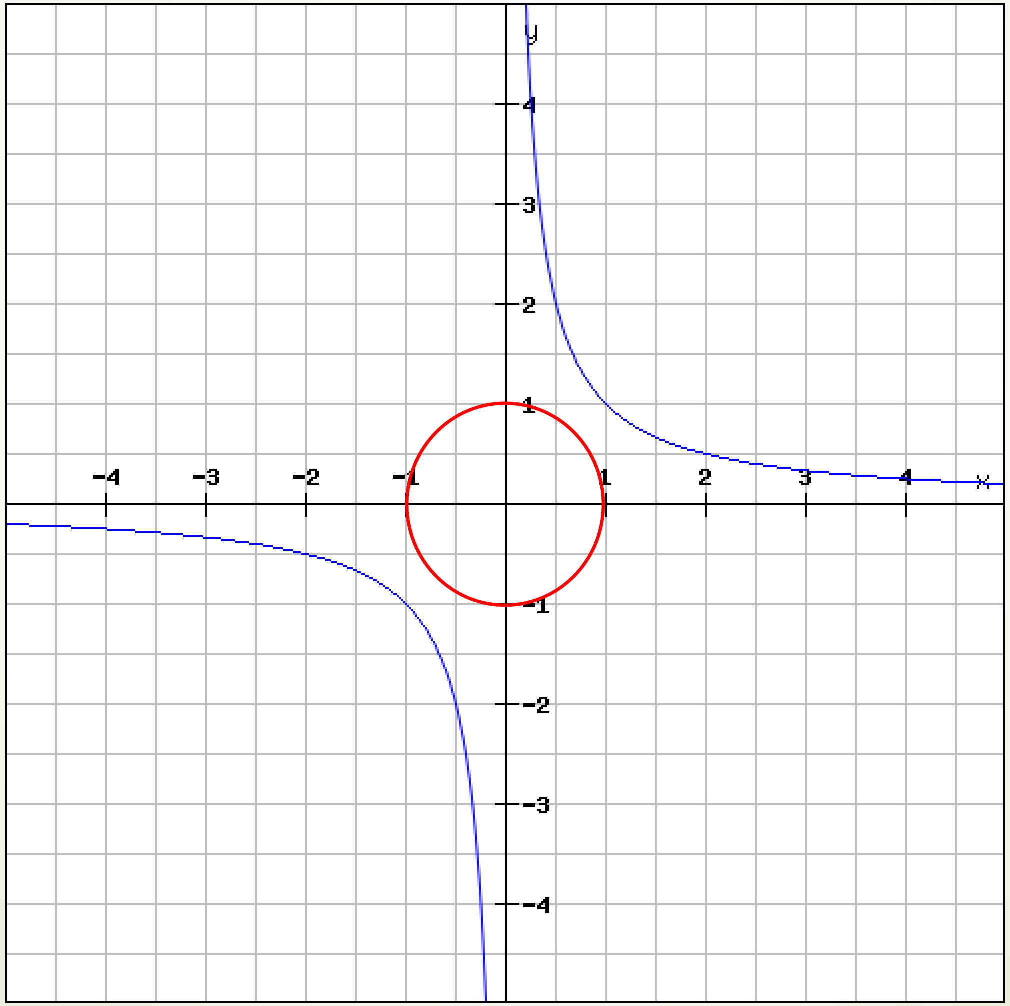
Equality handling and efficiency improvement of SMT for non-linear constraints over reals.

Vu Xuan Tung – Ogawa Lab - JAIST

Non-linear (polynomial) constraints over reals

$$\exists x, y(x^2 + y^2 < 1 \land x * y > 1)$$



Polynomial constraints over reals

Polynomial constraints solving has applications in:

- Automatic termination proving.
- Roundoff error and overflow error analysis.
- Invariant generation.

Polynomial constraints over reals

- In 1930, Tarski: polynomial constraints is decidable
- Methods:
  - QE-CAD: complete but DEXP complexity.
  - Interval constraint propagation: ISAT uses interval arithmetic (IA) only, ability of solving SAT problem is limited. raSAT: IA + testing
  - Bit-blasting: (UCLID, MiniSmt) suffers with high number of variables or high degree of polynomials.
  - Linearization: suffers with high degree of polynomials (Barcelogic, CORD).
  - Virtual substitution: Z3, SMT-RAT. Needs root formulas of polynomial

➔ degree <= 5

raSAT

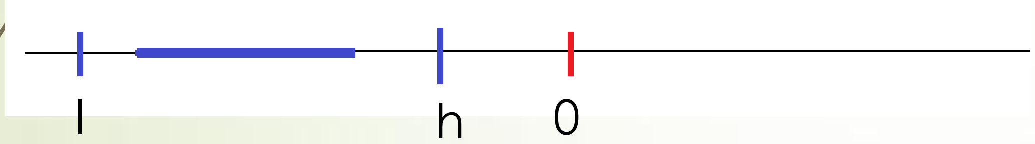
- Developed by Dr. Khanh To who took his PhD in our lab.
- An SMT solver (initially) for solving polynomial strict Inequalities:
  - Approximation can be used.
  - Suppose $f(x) > 0$ has a real solution x_0.
    - Because $f(x)$ is continuous,
    - There is some rational numbers x_1 near x_0 such that $f(x_1) > 0$

Over approximation - Interval arithmetic (IA)

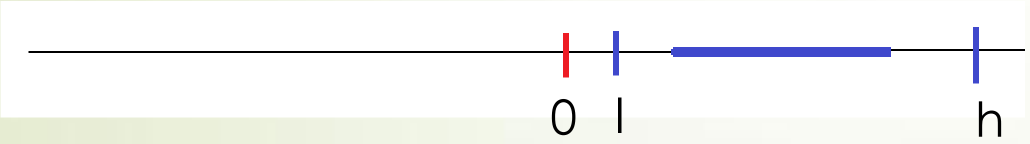
$f(x_1, \ldots, x_n), x_i \in [l_i, h_i]$ → **Interval arithmetic** → $[\mathrm{l}, \mathrm{h}], \{f(x_1, \ldots, x_n) | x_i \in [l_i, h_i]\} \subseteq [l, h]$

Example: $x * y, x \in [-2, 4], y \in [-1, 5]$ → $[-10, 20], \{x * y | x \in [-2, 4], y \in [-1, 5]\} \subseteq [-10, 20]$

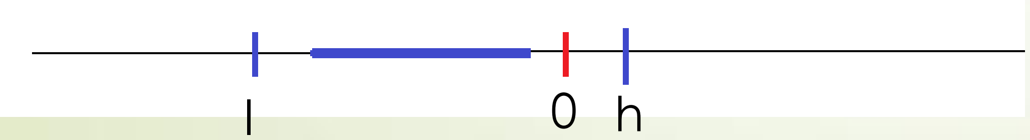
$f(x_1, \ldots, x_n) > 0$



IA-UNSAT → UNSAT



IA-VALID → SAT



IA-UNKNOWN → UNKNOWN

Under approximation - Testing

$x_i \in [l_i, h_i], \wedge f_j(x_1, ..., x_n) > 0$ → **Testing: Randomly generate values for variables** → SAT with variables assignment

→ UNKNOWN

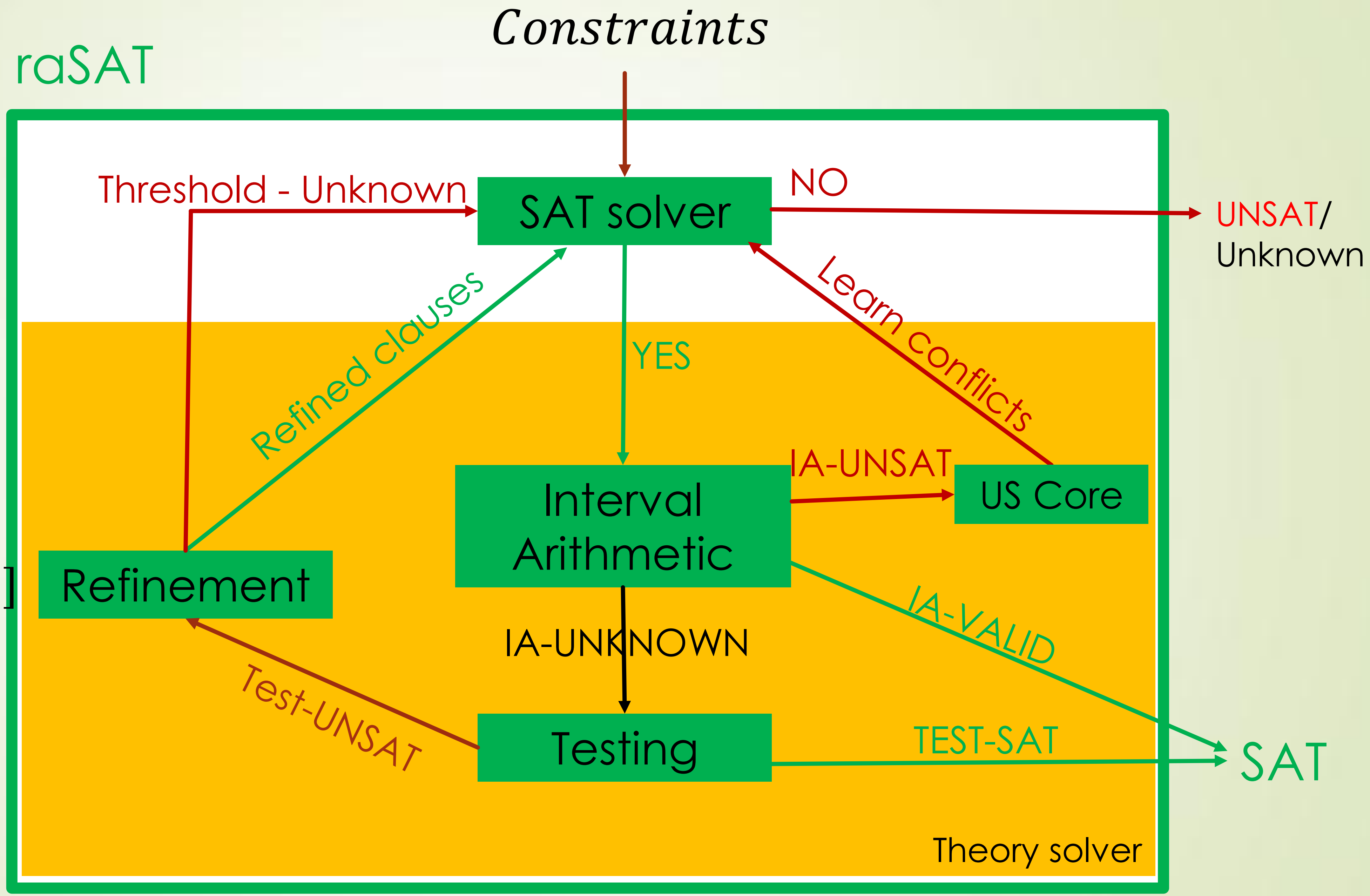
Example:

$x * y > 0, x \in [-2, 4], y \in [-1, 5]$ → **Testing: Randomly generate values for variables** → SAT with $\{x{:}3.7,\ y{:}0.98\}$

$x : -1.9, 3.7$

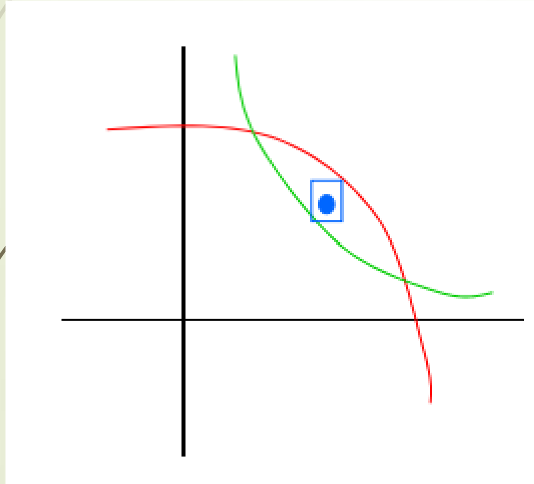
$y : 0.98, 3.65$

raSAT

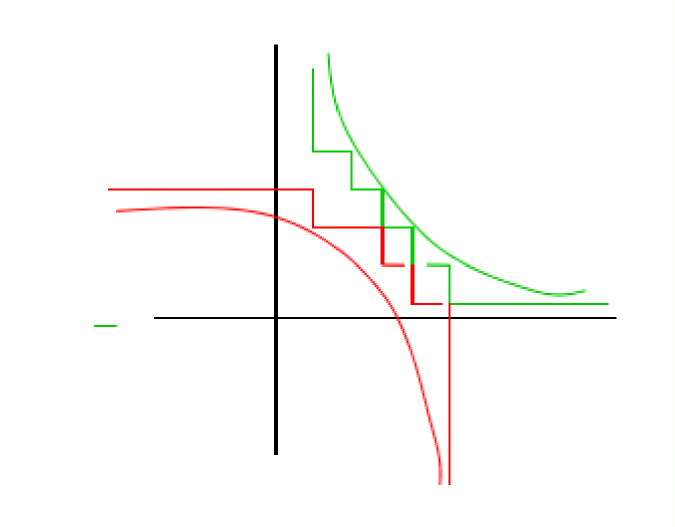


$x \in [0,10] \leftrightarrow$
$x \in [0,4] \vee x \in [4,10]$

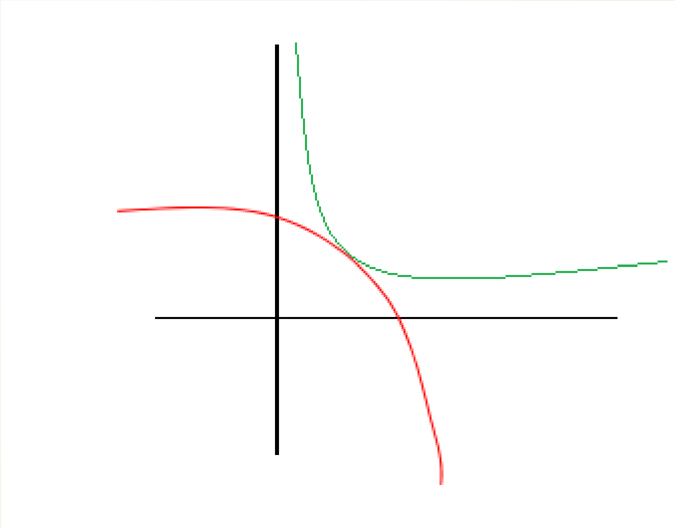
Completeness (strict inequality)



raSAT eventually detects SAT



UNSAT detected



UNSAT failed

raSAT

- In this work:
  - Improve the efficiency of raSAT.
  - Handle equality.
  - Handle polynomial constraints over Integer (QF_NIA).

Problems

1. Exploration of:
   - test cases: n variables, 2 values for 1 variable → 2^n test cases.
     - Example: x: -1.94, 3.7; y: 0.98, 3.65
       - 4 test cases: (x, y) = (-1,9, 0.98), (-1.9, 3.65), (3.7, 0.98), (3.7, 3.65)
   - boxes: n variables are decomposed -> 2^n boxes.
     - Example: $x \in [-2, 4] \rightarrow x \in [-2, 1] \vee x \in [1, 4]$

       $y \in [-1, 5] \rightarrow y \in [-1, 2] \vee y \in [2, 5]$
       - 4 boxes: $x \in [-2, 1] \wedge y \in [-1, 2]$ $\quad x \in [-2, 1] \wedge y \in [2, 5]$

         $x \in [1, 4] \wedge y \in [-1, 2]$ $\quad x \in [1, 4] \wedge y \in [2, 5]$

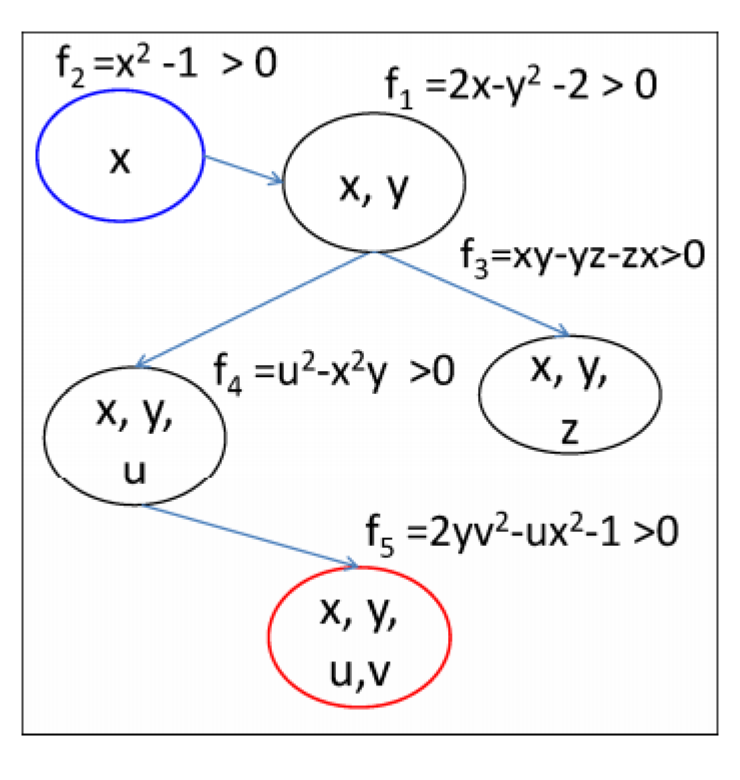
Problems.

2. Soundness.
   - Floating point arithmetic: round-off, overflow errors.
3. Equality handling.
   - Using the intermediate value theorem.

Current status

1. Exploration of test cases, boxes

- n variables → 2^n test cases.
- Priority on variables:
  1. Choice of constraint: Dependency between constraints



  2. Choice of variables in one constraints: Sensitivity

E.g. with $x = 1 + \epsilon_1$, $y = 2 + \epsilon_2$

$xy = 2\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2 + 2$: x is more sensible than y.

2. SAT, UNSAT verification

- Round-off, overflow errors can make the result unsound.
- iRRAM:
  - C++ package
  - Error-bounded real arithmetic
- **Integrated** iRRAM into raSAT for SAT verification.
- **Future work**: Verify UNSAT results
  - Improve UNSAT core.

3. Equality handling.

Intermediate value theorem

- Single equality: **Done in previous work**
- Multiple equalities:
  - Number of variables ≥ number of equations
  - **To be done**.

6. Extend for QF_NIA

- Current approaches:
  - Bit blasting: suffers with high degree of polynomials.
  - Linearization:
    - Bit-blast one operand of a multiplication.
- Can be solved by raSAT:
  - Decomposition: Stop when length of interval is **1**
  - Generate **integer** test cases.
  - **Future work**

raSAT

- Downloadable from **http://www.jaist.ac.jp/~mizuhito/tools/rasat.html**
- Participated in SMT-COMP 2014: 4^{th} over 4 solvers of QF_NRA.
- Preliminary experiments on SMT-LIB.
  - Mostly focus on Zankl family (166 benchmarks).
  - Around 50 problems solved (depending on tuning).

solver	solved	time (s)
nlsat	89	234.57
Mathematica	50	366.10
QEPCAD	21	38.85
Redlog-VTS	42	490.54
Redlog-CAD	21	173.15
z3	21	0.73
iSAT	21	24.52
cvc3	12	3.11
MiniSmt	46	1370.14

Dejan Jovanovic, Leonardo Mendonça de Moura: **Solving Non-linear Arithmetic.** IJCAR 2012: 339-354

Thank you for your attention

Doctor course Proposal

Problems.

1. Equality extension: Grobner basis.
2. UNSAT proof generation

1. Equality extension: Grobner basis.

- Intermediate value theorem:
  - Restriction: Number of variables ≥ number of equations
  - For complete equality handling: Grobner basis.
- Grobner basis computation was implemented in Mathemtica, Reduce
  - as standalone library,
  - might not have been seriously considered in solving polynomial constraints.
- We expect to adapt the computation algorithms to the purpose of proving satisfiabilitiy, unsatisfiability of constraints.
  - During computation process, we expect to integrate decision procedure of constraints so that we might decide SAT (UNSAT) before finishing Grobner basis computation.

1. Grobner basis – Example

Equations:

$$f_1 = x^2 + y^2 + z^2 - 1 = 0$$
$$f_2 = x^2 + z^2 - y = 0$$
$$f_3 = x - z = 0$$

Ordering: $x > y > z$

Grobner basis: $\{-1 + 2z^2 + 4z^4, y - 2z^2, x - z\}$

1. Grobner basis - Algorithms

- Buchberger Algorithm.
  - Reduce **one** s-pair at a time
- F_4, F_5 algorithms.
  - Reduce **many** s-pair at once.
- Need more investigations on algorithms and on how to adapt them to raSAT.

2. UNSAT proof generation

- Proof of UNSAT can be used to extract Craig interpolants.
- Craig interpolants have applications in:
  - Abstraction refinement.
  - Invariant generation.
- Most of the current works focus on Linear Arithmetic.
- Not much research on interpolants of polynomial constraints.
  - Such interpolants arise during verification of complex systems such as hybrid ones.

Primary idea

Two kinds of proofs:

1. Resolution proof: produced by SAT solver.
   - Resolution rule: $(a \vee b) \wedge (\neg a \vee c) \rightarrow b \vee c$
   - Interpolation from resolution proof is straitforward.
2. Proof of conflict clauses: produced by theory solver of raSAT.
   - Theory solver also infers interpolants from this proof.

Primary idea

Example:

- $A = x^2 + y^2 < 1, \; B = xz > 1$
- Intervals: $x \in [0, 10] \wedge y \in [0, 10] \wedge z \in [0, 1]$: $A \wedge B$ is UNSAT
- First, IA cannot conclude UNSAT.

Suppose $x \in [0, 10] \xrightarrow{decomposed} x \in [0, 1] \vee x \in [1,10]$:

- $x \in [0, 1] \wedge y \in [0, 10] \wedge z \in [0, 1]$
- $x \in [1,10] \wedge y \in [0, 10] \wedge z \in [0, 1]$

$$A = x^2 + y^2 < 1, \qquad B = xz > 1$$

$$x \in [0,1] \wedge y \in [0,10] \wedge z \in [0,1] \qquad\qquad x \in [1,10] \wedge y \in [0,10] \wedge z \in [0,1]$$

$$\frac{xz > 1 \qquad \dfrac{x \in [0,1] \qquad z \in [0,1]}{xz \in [0,1]}}{1 < 1}$$

$$\frac{\dfrac{x^2 + y^2 < 1 \qquad \dfrac{y \in [0,10]}{y^2 \in [0,100]}}{x^2 < 1} \qquad \dfrac{x \in [1,10]}{x^2 \in [1,100]}}{1 < 1}$$

Interpolant: $\top$

Interpolant: $x^2 < 1$

From resolution proof, we can infer $x^2 < 1$ as final interpolant of A and B

Thank you for your attention