Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions

Daniel Larraz, Albert Oliveras, Enric Rodríguez-Carbonell, Albert Rubio
Technical University of Catalonia (BarcelonaTech)

SAT'14, Wien, Austria
July 17th, 2014

Solving Formulas involving Non-Linear Polynomials [SMT(NIA)]

Daniel Larraz, Albert Oliveras, Enric Rodríguez-Carbonell, Albert Rubio
Technical University of Catalonia (BarcelonaTech)

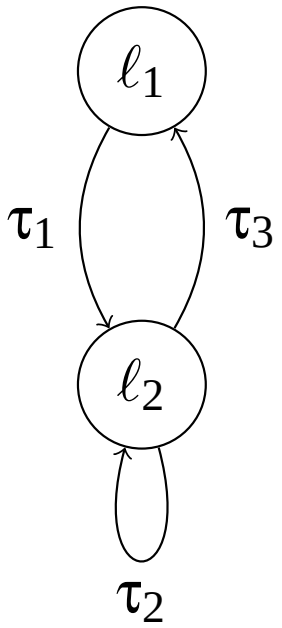
SAT'14, Wien, Austria
July 17th, 2014

Overview of the Talk

- Motivation
- Non-linear Constraint Solving Via Linearitzation
  - Unsatisfiable Cores
  - Max-SMT
  - OMT
- Experimental Evaluation
- Conclusions and Future Work

Does This Program Always Terminate?

```
    int x,y,z;
ℓ1: while (y ≥ 1) {
        x--;
    ℓ2: while (y < z) {
            x++; z--;
        }
        y = x + y;
    }
```



$$ini(l_1) = true$$
$$ini(l_2) = false$$

$$\rho_{\tau_1}: \quad y \geq 1, \quad x' = x - 1, \quad y' = y, \quad z' = z$$
$$\rho_{\tau_2}: \quad y < z, \quad x' = x + 1, \quad y' = y, \quad z' = z - 1$$
$$\rho_{\tau_3}: \quad y \geq z, \quad x' = x, \quad y' = x + y, \quad z' = z$$

Ranking Functions

IDEA:

- Prove that a certain transition τ_i is finitely executable.
- If so, any infinite execution eventually will not use τ_i.
- Hence we can remove τ_i from the SCC for termination analysis and keep removing transitions until no cycle exists.

Ranking Functions

IDEA:

- Prove that a certain transition τ_i is finitely executable.
- If so, any infinite execution eventually will not use τ_i.
- Hence we can remove τ_i from the SCC for termination analysis and keep removing transitions until no cycle exists.

IMPLEMENTATION:

Find a linear ranking function $R(x,y,z) = r_1x + r_2y + r_3z + r_4$ s.t.:

- $\tau_i \vdash R(x,y,z) \geq 0$ **[Boundedness]**
- $\tau_i \vdash R(x,y,z) > R(x',y',z')$ **[Strict Decrease]**
- $\tau_j \vdash R(x,y,z) \geq R(x',y',z')$ **[Non-increase]**
  for all $j \neq i$

EXAMPLE: z is a ranking function showing τ_2 finitely executable

Invariants

IDEA:

- In order to find ranking functions we sometimes need additional information about the program
- Properties that always hold at program locations would help (**invariants**).
- For each location we will try to find an assertion such that
  - It holds the first time the location is reached
  - It is preserved under every cycle back to the location

Invariants

IDEA:

- In order to find ranking functions we sometimes need additional information about the program
- Properties that always hold at program locations would help (**invariants**).
- For each location we will try to find an assertion such that
  - It holds the first time the location is reached
  - It is preserved under every cycle back to the location

IMPLEMENTATION:

For each loc. ℓ, find a lin. inv. $I_\ell(x,y,z) = c_1^\ell x + c_2^\ell y + c_3^\ell z + c_4^\ell \leq 0$ s.t:

- $ini(\ell) \vdash I_\ell$ **[Initiation]**
- $I_\ell \wedge \tau \vdash I'_{\ell'}$ for all transition τ from ℓ to ℓ' **[Consecution]**

EXAMPLE: $y \geq 1$ is invariant at ℓ_2. Boundedness of z is now easy.

Ranking Functions + Invariants

Initiation:	For all locs. ℓ:	$\mathbb{I}_\ell \stackrel{def}{=} ini(\ell) \vdash I_\ell$
Consecution:	For all trans. τ from ℓ to ℓ':	$\mathbb{C}_\tau \stackrel{def}{=} I_\ell \wedge \tau \vdash I'_{\ell'}$
Boundedness:	For all trans. τ from ℓ to ℓ':	$\mathbb{B}_\tau \stackrel{def}{=} I_\ell \wedge \tau \vdash R \geq 0$
Strict Decrease:	For all trans. τ from ℓ to ℓ':	$\mathbb{S}_\tau \stackrel{def}{=} I_\ell \wedge \tau \vdash R > R'$
Non-increase:	For all trans. τ from ℓ to ℓ':	$\mathbb{N}_\tau \stackrel{def}{=} I_\ell \wedge \tau \vdash R \geq R'$
Disability:	For all trans. τ from ℓ to ℓ':	$\mathbb{D}_\tau \stackrel{def}{=} I_\ell \wedge \tau \vdash 1 \leq 0$

If L is the set of locations, T the set of transitions and P the set of transitions pending to be proved finitely executable, we want to solve this formula:

$$\bigwedge_{\ell \in L} \mathbb{I}_\ell \wedge \bigwedge_{\tau \in T} (\mathbb{D}_\tau \vee \mathbb{C}_\tau) \wedge \bigvee_{\tau \in P} (\mathbb{D}_\tau \vee (\mathbb{B}_\tau \wedge \mathbb{S}_\tau)) \wedge ((\bigwedge_{\tau \in P} \mathbb{N}_\tau) \vee \bigvee_{\tau \in P} \mathbb{D}_\tau).$$

Where Do The Non-Linearities Come From?

- Consider $(\mathbb{S}_{\tau_2})$, i.e., strict decrease for τ_2:

  $$I \;\wedge\; y < z \;\wedge\; x' = x+1 \;\wedge\; y' = y \;\wedge\; z' = z-1 \;\vdash\; R' < R$$

  where $I = c_1x + c_2y + c_3z + c_4 \leq 0$ and $R = r_1x + r_2y + r_3z + r_4$.

- This is an $\exists\forall$ problem. Farkas' lemma allows one to convert it into an $\exists$ problem with non-linearities.
- In a nutshell, Farkas states that the above entailment holds iff $R' < R$ can be obtained via a linear combination of the premises.

Where Do The Non-Linearities Come From?

- Consider $(\mathbb{S}_{\tau_2})$, i.e., strict decrease for τ_2:

$$\overbrace{I}^{\lambda_1} \wedge \overbrace{y - z \leq 1}^{\lambda_2} \wedge \overbrace{x' - x - 1 = 0}^{\lambda_3} \wedge \overbrace{y' - y = 0}^{\lambda_4} \wedge \overbrace{z' - z + 1 = 0}^{\lambda_5} \vdash R' - R < 0$$

  where $I = c_1 x + c_2 y + c_3 z + c_4 \leq 0$ and $R = r_1 x + r_2 y + r_3 z + r_4$.

- This is an $\exists\forall$ problem. Farkas' lemma allows one to convert it into an $\exists$ problem with non-linearities.
- In a nutshell, Farkas states that the above entailment holds iff $R' < R$ can be obtained via a linear combination of the premises.
- For this particular example:

$$\begin{aligned}
\lambda_1 c_1 - \lambda_3 &= -r_1 && \text{(variable } x\text{)} \\
\lambda_1 c_2 + \lambda_2 - \lambda_4 &= -r_2 && \text{(variable } y\text{)} \\
\lambda_1 c_3 - \lambda_2 - \lambda_5 &= -r_3 && \text{(variable } z\text{)} \\
\lambda_3 &= r_1 && \text{(variable } x'\text{)} \\
&\ldots
\end{aligned}$$

Key Aspects to Highlight

FORMULAS ARISING FROM OUR APPLICATION:

- Quadratic polynomials with arbitrary Boolean structure
- In quadratic monomials, one var is a coefficient of an invariant
- It is usually the case that:
  - Integer programs have integer invariants
  - Coefficients of invariants are not very large
- UNSAT answers do not give too much information

Key Aspects to Highlight

FORMULAS ARISING FROM OUR APPLICATION:

- Quadratic polynomials with arbitrary Boolean structure
- In quadratic monomials, one var is a coefficient of an invariant
- It is usually the case that:
  - Integer programs have integer invariants
  - Coefficients of invariants are not very large
- UNSAT answers do not give too much information

IN GENERAL, OUR TECHNIQUES:

- Good at finding models, bad at proving unsatisfiability
- Deal with polynomials of any degree over the integers
- Problems with only very large solutions are difficult

Overview of the Talk

- Motivation
- Non-linear Constraint Solving Via Linearitzation
  - Unsatisfiable Cores
  - Max-SMT
  - OMT
- Experimental Evaluation
- Conclusions and Future Work

Non-Linear Constr. Solving via Linearization

- Problem: Given a quantifier-free formula F containing polynomial inequality atoms, is F satisfiable?
- In $\mathbb{Z}$: undecidable (Hilbert's 10th problem)
- In $\mathbb{R}$: decidable, even with quantifiers (Tarski)
  But algorithms have prohibitive complexity
- Goal: procedure that is "good" at finding models in $\mathbb{Z}$
- Basic idea: use bounds on integer variables to linearize the formula

Translating into Linear Arithmetic

- Example

$$u^4v^2 + 2u^2vw + w^2 \leq 4 \wedge 1 \leq u, v, w \leq 2$$

$$\Downarrow$$

$$x_{u^4v^2} + 2x_{u^2vw} + x_{w^2} \leq 4 \wedge 1 \leq u, v, w \leq 2 \wedge$$

$$x_{u^4v^2} = u^4v^2 \wedge x_{u^2vw} = u^2vw \wedge x_{w^2} = w^2$$

- For any formula there is an equisatisfiable one of the form

$$F \wedge (\bigwedge_i y_i = M_i)$$

where F is linear and each M_i is non-linear (introducing fresh variables)

Translating into Linear Arithmetic (2)

- Idea: linearize non-linear monomials with case analysis on some of the variables with finite domain
- $F \wedge x_{u^4v^2} = u^4v^2 \wedge x_{u^2vw} = u^2vw \wedge x_{w^2} = w^2$
  where F is $x_{u^4v^2} + 2x_{u^2vw} + x_{w^2} \leq 4 \wedge 1 \leq u, v, w \leq 2$
- Since $1 \leq w \leq 2$, add $x_{u^2v} = u^2v$ and
  $w = 1 \rightarrow x_{u^2vw} = x_{u^2v}$
  $w = 2 \rightarrow x_{u^2vw} = 2x_{u^2v}$
- Now, since $1 \leq u \leq 2$ replace $x_{u^2v} = u^2v$ by
  $u = 1 \rightarrow x_{u^2v} = v$
  $u = 2 \rightarrow x_{u^2v} = 4v$
- All in all, $x_{u^2vw} = u^2vw$ is replaced by the 4 clauses in green. Same should be done with other non-lineal monomials.

Translating into Linear Arithmetic (3)

- If we can fully linearize then we have a sound and complete procedure.
- If we don't have enough variables with finite domain...
  ... we can add artificial bounds at cost of losing completeness
  We cannot trust UNSAT answers!
- In this case, given a non-linear formula F we have obtained a **linear** formula which can be partioned in:
  - A formula SK that is the result of replacing in F all non-linear monomials by fresh variables
  - A set CS of case splitting clauses
  - A set B of artificial bounds
- An UNSAT answer might mean that artificial bounds are not large enough to include solutions. How can we solve this?

Method 1: Unsatisfiable Cores

- We can analyze the reasons for unsatisfiability:
  an unsatisfiable core (= unsatisfiable subset of clauses) can be obtained from the SMT solver.
- If core contains no clause in B (artificial bound), original non-linear formula is UNSAT.
- If core contains some clause in B (artificial bound):
  - enlarge the domain of at least one integer var of the core
  - add the corresponding case-splitting clauses to CS
- Drawbacks of the approach:
  - Cores are not guaranteed to be minimal
  - No information is obtained about how large the new domain should be

Method 1: Unsatisfiable Cores (2)

```
status solve_NIA(Formula F_0) {
  b = initial_bounds(F_0); // enough artificial bounds to linearize F_0
  F = linearize(F_0, b);
  while (not timed_out()) {
    ⟨st, core⟩ = solve_LIA(F);
    if (st == SAT) return SAT;
    else if (b ∩ core == ∅) return UNSAT;
    else {
      b = relax_domains(b, core); // at least one in the intersection is relaxed
      F = update(F, b); // add new bounds and case splitting clauses
  } }
  return UNKNOWN;
}
```

Method 2: Max-SMT

- Max-SMT: Given a set of weighted clauses, find a model that minimizes cost (= sum of weights) of falsified clauses

- Given a non-linear formula F and its transformation into $SK \wedge CS \wedge B$, we construct the weighted formula:
  - Clauses in SK and CS are given weight ∞ (hard clauses)
  - Clauses in B (artificial bounds) are given finite weights ω_C (soft clauses)

- Hence, we have a Max-SMT(LIA) problem, instead of an SMT(LIA) one

- Depending on the outcome:
  - Model with zero cost found: model for non-linear F
  - Otherwise falsified soft clauses show which bounds to enlarge

Method 2: Max-SMT (2)

- Advantages:
  - The optimal solution falsifies the minimal number of bounds
  - The optimal solution gives values to the variables of the falsified bounds that can be used to enlarge the domain (i.e. if bound is $x \leq 3$ and solution sets $x \mapsto 5$, next iteration should enlarge the bound to at least $x \leq 5$)

- Disadvantages:
  - Values of variables in optimal solution can be unnecessarily large.
    Note that extending a bound from $x \leq U$ to $x \leq U'$ involves adding $O(U' - U)$ case-splitting clauses.

Method 3: Optimization Modulo Theories

- OMT: given a formula F involving numeric vars, find a model that minimizes the value of a particular numeric variable *cost*.

  Note that this allows one to express a variety of optimization problems
- How to use OMT here?

Method 3: Optimization Modulo Theories (2)

- Given a non-linear formula F and its transformation into $SK \wedge CS \wedge B$, we consider the formula $SK \wedge CS$

- If $S = \{x \mid x \text{ has artificial domain } [L_x, U_x]\}$, we want to **minimize** $\sum_{x \in S} \delta(x, [L_x, U_x])$ where $\delta(z, [L,U])$ is the *distance* of z to $[L,U]$:

$$\delta(z,[L,U]) = \begin{cases} L - z & \text{if } z < L \\ 0 & \text{if } L \leq z \leq U \\ z - U & \text{if } z > U \end{cases}$$

- For that purpose, introduce, for each var $x \in S$, two integer vars dl_x and du_x and add the following formula to $SK \wedge CS$:

$$\bigwedge_{x \in S} (dl_x \geq 0 \ \wedge \ dl_x \geq L_x - x \ \wedge \ du_x \geq 0 \ \wedge \ du_x \geq x - U_x) \ \wedge \ cost = \sum_{x \in S} (dl_x + du_x)$$

Method 3: Optimization Modulo Theories (3)

- Depending on the outcome:
  - Cost zero solution: model for non-linear F
  - Otherwise, variables contributing to the cost show which bounds to enlarge and their value indicate how much we have to enlarge the domain.

- Advantages:
  - Domain enlargement is model-guided: which domains to enlarge and how much
  - Domain enlargement is minimal. Indeed, the cost function minimizes the number of necessary clauses to be added in CS once the domain is extended

- Disadvantages:
  - Less mature SAT-based technology

Methods 2 & 3: Max-SMT and OMT

```
status solve_NIA(Formula F0) {
  b = initial_bounds(F0); // enough artificial bounds to linearize F0
  F = linearize(F0, b);
  while (not timed_out()) {
    ⟨st, model⟩ = optimize_LIA(F);
    if (st == UNSAT) return UNSAT;
    else if (cost(model) == 0) return SAT;
    else {
      b = relax_domains(b, model);
      F = update(F, b); // add new bounds and case splitting clauses
  } }
  return UNKNOWN;
}
```

Overview of the Talk

- Motivation
- Non-linear Constraint Solving Via Linearitzation
  - Unsatisfiable Cores
  - Max-SMT
  - OMT
- Experimental Evaluation
- Conclusions and Future Work

Experimental evaluation

Termination benchs	z3		bcl-cores		bcl-maxsmt		bcl-omt	
	#prob	secs	#prob	secs	#prob	secs	#prob	secs
SAT	1136	2578	1838	5464	**1852**	**3198**	1798	7896
UNSAT	0	0	0	0	4	0	**62**	**112**
UNKNOWN	11	2	0	0	0	0	0	0
TIMEOUT	787	47220	96	5760	78	4680	74	4440

Model checking benchs	z3		bcl-cores		bcl-maxsmt		bcl-omt	
	#prob	secs	#prob	secs	#prob	secs	#prob	secs
SAT	30	2	**35**	**55**	35	72	34	263
UNSAT	1	0	1	0	1	0	1	0
UNKNOWN	0	0	0	0	0	0	0	0
TIMEOUT	5	300	0	0	0	0	1	60

Overview of the Talk

- Motivation
- Non-linear Constraint Solving Via Linearitzation
  - Unsatisfiable Cores
  - Max-SMT
  - OMT
- Experimental Evaluation
- Conclusions and Future Work

Conclusions and Future Work

- **Conclusions:**
  - Nice application of SAT-based technology: cores, Max-SMT, OMT.
  - Even though problem is undecidable, efficient techniques for particular classes of formulas exist.
  - Currently Max-SMT seems to be the best option.
    - Skipped for simplicity in the talk: most of our problems are already Max-SMT (over non-linear formulas)
- **Future work:**
  - Core-based algorithms for Max-SMT
  - Combine with simple unsatisfiability procedures
  - Push forward Max-SMT-based program analysis

Thank you!