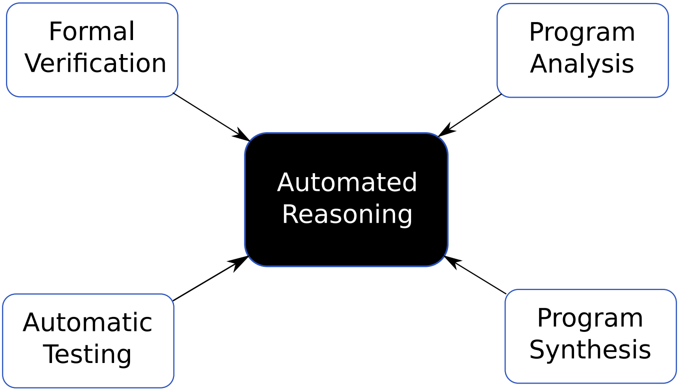
Revisiting Enumerative Instantiation

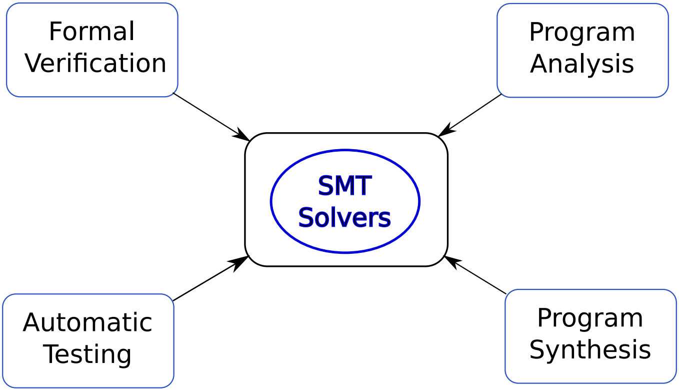
Andrew Reynolds[1], Haniel Barbosa[1,2] and *Pascal Fontaine*[2]

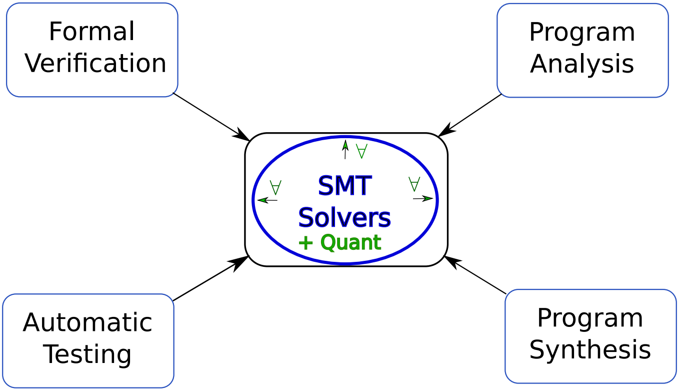
[1]University of Iowa, Iowa City, U.S.A.

[2]University of Lorraine, CNRS, Inria, LORIA, Nancy, France

TACAS 2018/Matryoshka 2018/SMT 2018


Formal Verification
Program Analysis
Automated Reasoning
Automatic Testing
Program Synthesis


Formal Verification
Program Analysis
SMT Solvers
Automatic Testing
Program Synthesis

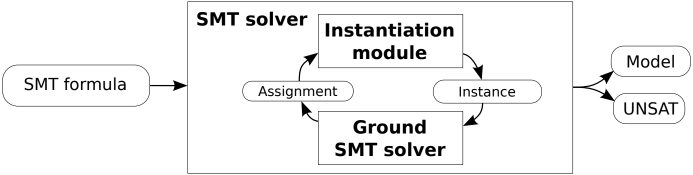

Formal Verification
Program Analysis
SMT Solvers
+ Quant
∀
Automatic Testing
Program Synthesis

Outline

- Quantifier handling in SMT solving
- Strengthening the Herbrand Theorem
- Effective enumerative instantiation
  - Combination with other instantiation strategies
  - Implementation
- Evaluation

Quantifier handling in SMT

Problem statement



Ground solver enumerates assignments $\mathrm{E} \cup \mathrm{Q}$

- E is a set of ground literals $\{a \leq b,\ b \leq a + x,\ x \simeq 0,\ f(a) \not\simeq f(b)\}$
- Q is a set of quantified clauses $\{\forall xyz.\ f(x) \not\simeq f(z) \vee g(y) \simeq h(z)\}$

Instantiation generates instances of Q $f(a) \not\simeq f(b) \vee g(a) \simeq h(b)$

Instantiation strategies: *trigger-based* [Detlefs et al. J. ACM'05]

Trigger-based instantiation (E-matching): search for relevant instantiations according to a set of triggers and E-matching

Instantiation strategies: *trigger-based* [Detlefs et al. J. ACM'05]

Trigger-based instantiation (E-matching): search for relevant instantiations according to a set of triggers and E-matching

- $\mathrm{E} = \{\neg P(a), \neg P(b), P(c), \neg R(b)\}$ and
  $\mathrm{Q} = \{\forall x.\, P(x) \vee R(x)\}$
- Assume trigger $P(x)$
- Find substitution σ for x such $P(x)$ is a know term (in E)
- Suitable substitutions are $x \mapsto a$, $x \mapsto b$, or $x \mapsto c$. E.g.
  $\mathrm{E} \models P(x)[x/a] = P(a)$ and $P(a) \in E$
- Formally

$\mathbf{e}(\mathrm{E}, \forall \bar{x}.\, \varphi)$
1. Select a set of triggers $\{\bar{t}_1, \dots \bar{t}_n\}$ for $\forall \bar{x}.\, \varphi$
2. For each $i = 1, \dots, n$, select a set of substitutions S_i s.t for each $\sigma \in S_i$, $\mathrm{E} \models \bar{t}_i\sigma \simeq \bar{g}_i$ for some tuple $\bar{g}_i \in \mathbf{T}(\mathrm{E})$
3. Return $\bigcup_{i=1}^{n} S_i$

Instantiation strategies: *conflict-based* [Reynolds et al. FMCAD'14]

Conflict-based instantiation: search for instantiations of a quantified formula in Q that makes E unsatisfiable

Instantiation strategies: *conflict-based* [Reynolds et al. FMCAD'14]

Conflict-based instantiation: search for instantiations of a quantified formula in Q that makes E unsatisfiable

- $\mathsf{E} = \{\neg P(a), \neg P(b), P(c), \neg R(b)\}$ and
  $\mathsf{Q} = \{\forall x.\, P(x) \vee R(x)\}$

- Since E, $P(b) \vee R(b) \models \bot$, this strategy returns $x \mapsto b$

- Formally

  $\mathbf{c}(\mathsf{E},\, \forall \bar{x}.\, \varphi)$ Either returns σ where $\mathsf{E}, \varphi\sigma \models \bot$, or return $\emptyset$

Instantiation strategies: *model-based* [Ge and de Moura CAV'09]

Model-based instantiation (MBQI): build a candidate model for $\mathsf{E} \cup \mathsf{Q}$ and instantiate with counter-examples from model checking

Instantiation strategies: *model-based* [Ge and de Moura CAV'09]

Model-based instantiation (MBQI): build a candidate model for $\mathsf{E} \cup \mathsf{Q}$ and instantiate with counter-examples from model checking

- $\mathsf{E} = \{\neg P(a), \neg P(b), P(c), \neg R(b)\}$ and
  $\mathsf{Q} = \{\forall x.\, P(x) \vee R(x)\}$
- Assume that $P^{\mathcal{M}} = \lambda x.\, \mathrm{ite}(x \simeq c,\, \top,\, \bot)$ and $R^{\mathcal{M}} = \lambda x.\, \bot$
- Since $\mathcal{M} \not\models P(a) \vee R(a)$, MBQI may return $x \mapsto a$
- Formally

$\mathbf{m}(\mathsf{E}, \forall \bar{x}.\, \varphi)$
1. Construct a model $\mathcal{M}$ for E
2. Return $\bar{x} \mapsto \bar{t}$ where $\bar{t} \in \mathbf{T}(\mathsf{E})$ and $\mathcal{M} \models \neg\varphi[\bar{x}/\bar{t}]$, or $\emptyset$ if none exists

Shortcomings

- Conflict-based instantiation (**c**)
  - Inherently incomplete
- *E*-matching (**e**)
  - Too many instances
  - Butterfly effect
- MBQI (**m**)
  - Complete for many fragments, but slow convergence for UNSAT
  - Better suited for model finding

Generally SMT solvers implement complete techniques by applying **m** as a last resort after trying **c** and **e**

Strengthening the Herbrand Theorem

Why can we use instantiation?

Theorem (Herbrand)

A set of pure first-order logic formulas is unsatisfiable if and only if there exists a finite unsatisfiable set of its instances

Why can we use instantiation?

Theorem (Herbrand)

A set of pure first-order logic formulas is unsatisfiable if and only if there exists a finite unsatisfiable set of its instances

- The earliest theorem provers relied on *Herbrand instantiation*
  - Instantiate with all possible terms in the language
- Enumerating all instances is unfeasible in practice!
- Enumerative instantiation was then discarded

Why can we use instantiation?

Theorem (Herbrand)

A set of pure first-order logic formulas is unsatisfiable if and only if there exists a finite unsatisfiable set of its instances

- The earliest theorem provers relied on *Herbrand instantiation*
  - Instantiate with all possible terms in the language
- Enumerating all instances is unfeasible in practice!
- Enumerative instantiation was then discarded

We make enumerative instantiation beneficial for state-of-the-art SMT

- strengthening of Herbrand theorem
- efficient implementation techniques

Theorem (Strengthened Herbrand)

If R is a (possibly infinite) set of instances of Q closed under Q-instantiation w.r.t. itself and if $E \cup R$ is satisfiable, then $E \cup Q$ is satisfiable.

Theorem (Strengthened Herbrand)

If there exists an infinite sequence of finite satisfiable sets of ground literals E_i *and of finite sets of ground instances* Q_i *of* Q *such that*

- $\mathsf{Q}_i = \left\{\varphi\sigma \;\middle|\; \forall\bar{x}.\,\varphi \in \mathsf{Q},\, \mathrm{dom}(\sigma) = \{\bar{x}\} \wedge \mathrm{ran}(\sigma) \subseteq \mathbf{T}(\mathsf{E}_i)\right\}$;
- $\mathsf{E}_0 = \mathsf{E}$, $\mathsf{E}_{i+1} \models \mathsf{E}_i \cup \mathsf{Q}_i$;

then $\mathsf{E} \cup \mathsf{Q}$ *is satisfiable in the empty theory with equality*

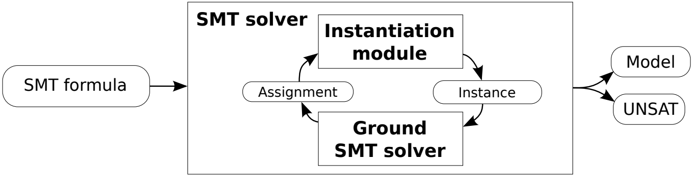
Theorem (Strengthened Herbrand)

If there exists an infinite sequence of finite satisfiable sets of ground literals E_i *and of finite sets of ground instances* Q_i *of* Q *such that*

- $\mathsf{Q}_i = \{\varphi\sigma \mid \forall\bar{x}.\,\varphi \in \mathsf{Q},\ \mathrm{dom}(\sigma) = \{\bar{x}\} \wedge \mathrm{ran}(\sigma) \subseteq \mathbf{T}(\mathsf{E}_i)\}$;
- $\mathsf{E}_0 = \mathsf{E}$, $\mathsf{E}_{i+1} \models \mathsf{E}_i \cup \mathsf{Q}_i$;

then $\mathsf{E} \cup \mathsf{Q}$ *is satisfiable in the empty theory with equality*

Direct application at



- Ground solver enumerates assignments $\mathsf{E} \cup \mathsf{Q}$
- Instantiation module generates instances of Q

Effective enumerative instantiation

Enumerative instantiation

$\mathbf{u}(\mathrm{E}, \forall \bar{x}.\, \varphi)$

1. Choose an ordering $\preceq$ on tuples of ground terms
2. Return $\bar{x} \mapsto \bar{t}$ where $\bar{t}$ is a minimal tuple of terms w.r.t $\preceq$, such that $\bar{t} \in \mathbf{T}(\mathrm{E})$ and $\mathrm{E} \not\models \varphi[\bar{x}/\bar{t}]$, or $\emptyset$ if none exist

- $\mathrm{E} = \{\neg P(a), \neg P(b), P(c), \neg R(b)\}$ and
  $\mathrm{Q} = \{\forall x.\, P(x) \vee R(x)\}$

- $\mathbf{u}$ chooses an ordering on tuples of terms, e.g. $a \prec b \prec c$

- Since $\mathrm{E} \not\models P(a) \vee R(a)$, enumerative instantiation returns $x \mapsto a$

u as an alternative for **m**

- Enumerative instantiation plays a similar role to MBQI
- It can also serve as a “completeness fallback” to **c** and **e**
- However, **u** has advantages over **m** for UNSAT problems
- Moreover it is significantly simpler to implement
  - No model building
  - No model checking

Example

$$
\begin{array}{rcl}
\mathsf{E} & = & \{\neg P(a),\ R(b),\ S(c)\} \\
\mathsf{Q} & = & \{\forall x.\, R(x) \vee S(x),\ \forall x.\, \neg R(x) \vee P(x),\ \forall x.\, \neg S(x) \vee P(x)\}
\end{array}
$$

$$
\mathsf{M} = \left\{ \begin{array}{lcl}
P^{\mathcal{M}} & = & \lambda x.\, \bot, \\
R^{\mathcal{M}} & = & \lambda x.\, \mathsf{ite}(x \simeq b,\ \top,\ \bot), \\
S^{\mathcal{M}} & = & \lambda x.\, \mathsf{ite}(x \simeq c,\ \top,\ \bot)
\end{array} \right\}, \qquad a \prec b \prec c
$$

φ	x s.t. $\mathcal{M} \models \neg\varphi$	x s.t. $\mathsf{E} \not\models \varphi$	$\mathbf{m}(\mathsf{E}, \forall x.\, \varphi)$	$\mathbf{u}(\mathsf{E}, \forall x.\, \varphi)$
$R(x) \vee S(x)$	a	a	$x \mapsto a$	$x \mapsto a$
$\neg R(x) \vee P(x)$	b	a, b, c	$x \mapsto b$	$x \mapsto a$
$\neg S(x) \vee P(x)$	c	a, b, c	$x \mapsto c$	$x \mapsto a$

- **u** instantiates uniformly so that new terms are introduced less often
- **m** instantiates depending on how model was built
- Moreover, **u** leads to $\mathsf{E} \wedge \mathsf{Q}[x/a] \models \bot$
- **m** requires considering E' which satisfies E along the new instances

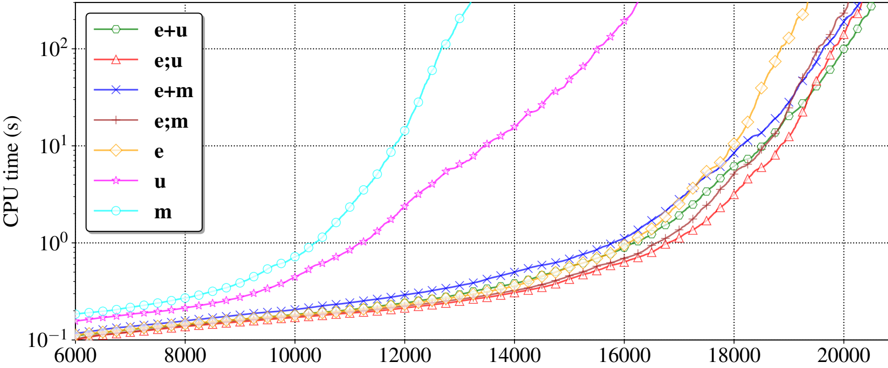
Implementation

Implementing enumerative instantiation efficiently depends on:

- Restricting enumeration space
- Avoiding entailed instantiations
- Term ordering to introduce new terms less often

Evaluation

CVC4 configurations on unsatisfiable benchmarks



- 42 065 benchmarks: 14 731 TPTP + 27 334 SMT-LIB
- **e**+**u** stands for "interleave **e** and **u**", while **e**;**u** for "apply **e** first, then **u** if it fails"
- All CVC4 configurations have "**c**;" as prefix

Impact of **u** on satisfiable benchmarks

Library	#	**u**	**e;u**	**e+u**	**e**	**m**	**e;m**	**e+m**
TPTP	14731	471	492	464	17	930	808	829
UF	7293	39	42	42	0	70	69	65
Theories	20041	3	3	3	3	350	267	267
Total	42065	513	537	509	20	1350	1144	1161

- As expected, **m** greatly outperforms **u**
- **u** answers SAT *half as often* as **m** in empty theory
- **u** solves 13 problems **m** does not

Conclusions

- We have introduced an efficient way of applying enumerative instantiation in SMT solving
- New technique is based on an strengthening of the Herbrand Theorem
- Implementation in SMT solver CVC4
  - Significantly increases success rate
  - Outperforms existing implementations of MBQI for UNSAT
  - Can be used for SAT in the empty theory

Appendix

Restricting Enumeration Space

- Strengthened Herbrand Theorem allows restriction to $\mathbf{T}(\mathsf{E})$
- *Sort inference* reduces instantiation space by computing more precise sort information
  - $\mathsf{E} \cup \mathsf{Q} = \{a \not\simeq b,\ f(a) \simeq c\} \cup \{P(f(x))\}$
    - $a, b, c, x : \tau$
    - $f : \tau \to \tau$ and $P : \tau \to \mathsf{Bool}$
  - This is equivalent to
    $\mathsf{E}^s \cup \mathsf{Q}^s = \{a_1 \not\simeq b_1,\ f_{12}(a_1) \simeq c_2\} \cup \{P_2(f_{12}(x_1))\}$
    - $a_1, b_1, x_1 : \tau_1$
    - $c_2 : \tau_2$
    - $f_{12} : \tau_1 \to \tau_2$ and $P : \tau_2 \to \mathsf{Bool}$
  - $\mathbf{u}$ would derive e.g. $x \mapsto c$ for $\mathsf{E} \cup \mathsf{Q}$, while for $\mathsf{E}^s \cup \mathsf{Q}^s$ the instantiation $x_1 \mapsto c_2$ is not well-sorted

Entailment Checks

Two-layered method for checking whether $\mathsf{E} \models \varphi[\bar{x}/\bar{t}]$ holds

- Cache of instantiations already derived

- Incomplete but fast method for checking $\mathsf{E} \models \ell$

  Repeat until a fix point:

  1. Replace each leaf term t in ℓ with $[t]$
  2. Replace each term $f(t_1, \ldots, t_n)$ in ℓ with s if $(t_1, \ldots, t_n) \rightarrow s \in \mathcal{I}_f$
  3. Replace each term $f(t_1, \ldots, t_n)$ in ℓ where f is an interpreted function with the result of the evaluation $f(t_1, \ldots, t_n){\downarrow}$

  Then, if the resultant ℓ is $\top$, then the entailment holds

  - Extension to incorporate Boolean structure
  - Extension to other theories through theory-specific rewriting

Term Ordering

Instantiations are enumerated according to the order

$$(t_1, \ldots, t_n) \prec (s_1, \ldots, s_n) \quad \text{if} \quad \begin{cases} \max_{i=1}^n t_i \prec \max_{i=1}^n s_i, \text{ or} \\ \max_{i=1}^n t_i = \max_{i=1}^n s_i \text{ and} \\ \qquad\qquad (t_1, \ldots, t_n) \prec_{\text{lex}} (s_1, \ldots, s_n) \end{cases}$$

for a given order $\preceq$ on ground terms.

If $a \prec b \prec c$, then

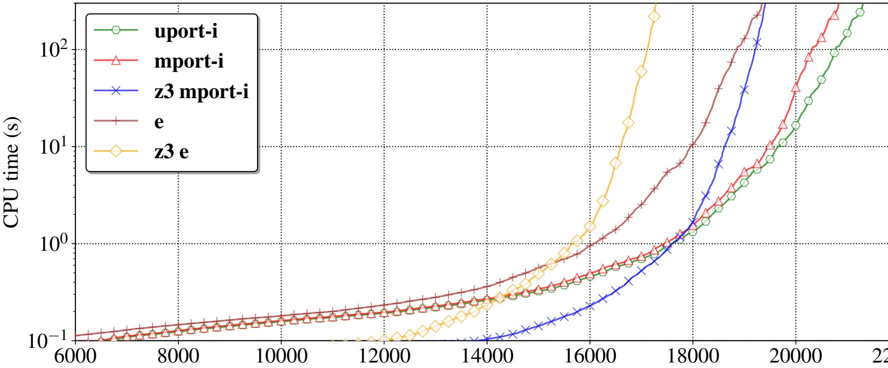
$$(a, a) \prec (a, b) \prec (b, a) \prec (b, b) \prec (a, c) \prec (c, b) \prec (c, c)$$

- We consider instantiations with c only after considering all cases with a and b
- Goal is to introduce new terms less often
- Order on $\mathbf{T}(\mathrm{E})$ fixed for finite set of terms $t_1 \prec \ldots \prec t_n$
  - Instantiate in order with $t_1, \ldots, t_n$
  - Then choose new non-congruent term $t \in \mathbf{T}(\mathrm{E})$ and have $t_n \prec t$

Impact of **u** on unsatisfiable benchmarks

- **u** solves 3 043 more benchmarks than **m**
- **u** solves 1 737 problems not solvable by **e**
- Combinations of **e** with **u** or **m** lead to significant gains
- **e**+**u** is best configuration, solving 253 more problems than **e**+**m** and 1 295 more than **e**
- Some benchmark families only solvable due to enumeration
- Overall the enumerative strategies lead to a virtual portfolio of CVC4 solving 712 more problems

Comparison against other instantiation-based SMT solvers



- Portfolios run without interleaving strategies (not supported by Z3)
- Z3 uses several optimizations for **e** not implemented in CVC4
- Z3 does not implement **c**