Machine learning for instance selection in SMT solving

Jasmin Christian Blanchette [1, 2] Daniel El Ouraoui[2] Pascal Fontaine[2] Cezary Kaliszyk[3]

Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

University of Lorraine, CNRS, Inria, and LORIA, Nancy, France

University of Innsbruck, Innsbruck, Austria

13th June 2019

Contents

Contents

Motivations

Satisfiability modulo theories (SMT)

- Automation
  - Proof assistant
  - Verification conditions
  - Model checking
- Solvers
  - Z3, CVC4, VERIT, ...

Instantiation

- Hard for SMT solvers
- Heuristically solved

Challenge

- Improve instantiation techniques
- Solve more problems
- Be more efficient

Our tool

veriT

Université de Lorraine/UFRN (`http://www.verit-solver.org`)

Contents

Context

- Ground
- Instantiation

$$b \neq a \wedge f(a) = f(b) \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

Ground problem

How efficiently check the satisfiability of a ground formula

$$(f(a, b) = g(a) \vee d = b) \wedge d = g(b) \wedge d \neq f(a, b) \wedge b = a \wedge d \neq g(a)$$

Ground problem

How efficiently check the satisfiability of a ground formula

$$(\underbrace{f(a,b) = g(a)}_{l_1} \vee \underbrace{d = b}_{l_2}) \wedge \underbrace{d = g(b)}_{l_3} \wedge \underbrace{d \neq f(a,b)}_{l_4} \wedge \underbrace{b = a}_{l_5} \wedge \underbrace{d \neq g(a)}_{l_6}$$

Ground problem

How efficiently check the satisfiability of a ground formula

$$(\underbrace{f(a,b) = g(a)}_{l_1} \vee \underbrace{d = b}_{l_2}) \wedge \underbrace{d = g(b)}_{l_3} \wedge \underbrace{d \neq f(a,b)}_{l_4} \wedge \underbrace{b = a}_{l_5} \wedge \underbrace{d \neq g(a)}_{l_6}$$

$$\downarrow$$

Ground problem

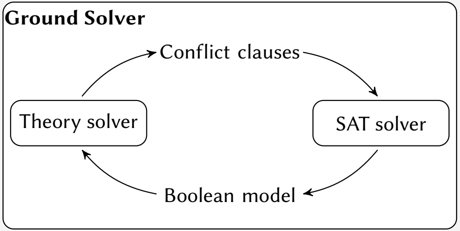
How efficiently check the satisfiability of a ground formula

$$(\underbrace{f(a,b) = g(a)}_{l_1} \vee \underbrace{d = b}_{l_2}) \wedge \underbrace{d = g(b)}_{l_3} \wedge \underbrace{d \neq f(a,b)}_{l_4} \wedge \underbrace{b = a}_{l_5} \wedge \underbrace{d \neq g(a)}_{l_6}$$

$$\downarrow$$

$$(l_1 \vee \neg l_2) \wedge l_3 \wedge l_4 \wedge l_5 \wedge l_6$$

CDCL(T)



- Formulas are embedded in SAT
- SAT solver produces a boolean model
- Theory solvers produce conflict clauses
- Conflict clauses guide the SAT solver

First-Order problem

$$b \neq a \wedge f(a) = f(b) \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

- Instantiation

First-Order problem

How to find an instance such that the problem is UNSAT

$$b \neq a \wedge f(a) = f(b) \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

First-Order problem

How to find an instance such that the problem is UNSAT

$$\underbrace{b \neq a \wedge f(a) = f(b)}_{\textbf{SAT}} \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

First-Order problem

How to find an instance such that the problem is UNSAT

$$\underbrace{b \neq a \land f(a) = f(b)}_{\textbf{SAT}} \land \forall xy\, f(x) = f(y) \Rightarrow x = y$$

$$\downarrow$$

First-Order problem

How to find an instance such that the problem is UNSAT

$$\underbrace{b \neq a \wedge f(a) = f(b)}_{\textbf{SAT}} \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

$$\downarrow$$

$$f(a) \neq f(b) \vee a = b$$

First-Order problem

How to find an instance such that the problem is UNSAT

$$\underbrace{b \neq a \wedge f(a) = f(b)}_{\text{SAT}} \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

$$\downarrow$$

$$f(a) \neq f(b) \vee a = b$$

$$\downarrow$$

First-Order problem

How to find an instance such that the problem is UNSAT

$$\underbrace{b \neq a \wedge f(a) = f(b)}_{\textbf{SAT}} \wedge \forall xy\, f(x) = f(y) \Rightarrow x = y$$

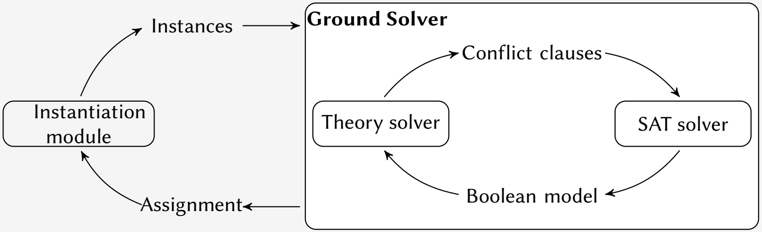
$$\downarrow$$

$$f(a) \neq f(b) \vee a = b$$

$$\downarrow$$

$$\underbrace{b \neq a \wedge f(a) = f(b) \wedge f(a) \neq f(b)}_{\textbf{UNSAT}}$$

First-Order CDCL(T)



Contents

- **Conflict based instantiation** Introduced by Reynolds, this technique produces relevant sets of instances. The idea is that, given a ground model $\mathcal{M}$ and a quantified formula $\forall(\overline{x}_n : \overline{\tau}_n).\varphi$, we find a substitution σ such that $\mathcal{M} \models \neg\varphi\sigma$.
- **Congruence Closure with Free Variable (CCFV)** Introduced by Barbosa et al., generalizes the idea of Conflict based instantiation by reasoning over equivalence classes.

State of the art

Enumerative instantiation $\forall(x:\tau).\psi[x] \equiv \bigwedge_{t\in\mathcal{D}_\tau} \psi[t]$

Enumerate all ground terms over the domain of x (aka. Herbrand universe)

Trigger based instantiation

Triggers

A trigger T for a quantified formula $\forall\overline{x}_n.\psi$ is a set of non-ground terms $u_1,\ldots,u_n \in \mathbf{T}(\psi)$ such that: $\{\overline{x}\} \subseteq \mathsf{FV}(u_1) \cup \ldots \cup \mathsf{FV}(u_n)$.

$E = f(a) \simeq g(b),\ a \simeq g(b)$
$Q = \forall x\, f(g(x)) \not\simeq g(x)$
$T = f(g(x))$

$f(a)$ E-matches $f(g(x))$ under $x \mapsto b$

Strategie

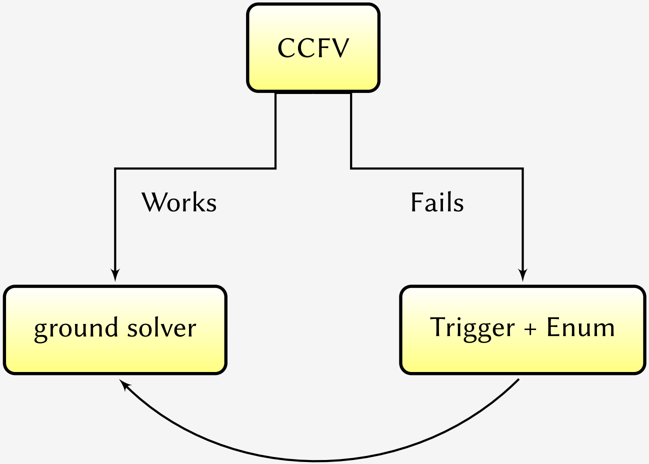


Figure: Instantiation strategie

Summarize

Conflict based instantiation and CCFV :

- Pro Efficient, if find substitution kill the model
- Pro All generated instances are useful
- Cons Finds contradiction involving only one instance

Enumerative and Trigger based instanciation :

- Pro Useful when CCFV fail
- Cons Many heuristics
- Cons Generates a lot of junk, and many instances

Summarize

Conflict based instantiation and CCFV :

- Pro: Efficient, if find substitution kill the model
- Pro: All generated instances are useful
- Cons: Finds contradiction involving only one instance

Enumerative and Trigger based instanciation :

- Pro: Useful when CCFV fail
- Cons: Many heuristics
- Cons: Generates a lot of junk, and many instances

Indeed

This is what we want improve!

Contents

Problem

- **How many lemmas are <u>generated</u> to solve a problem?**
  - around 300 for the UF category of the SMT-LIB
  - some generate more than 100 000 instances
- **How many lemmas are <u>needed</u> to solve a problem?**
  - Only 10% of this number, and sometimes much less

Problem

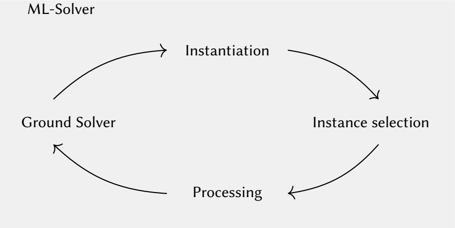
- **How many lemmas are generated to solve a problem?**
  - around 300 for the UF category of the SMT-LIB
  - some generate more than 100 000 instances
- **How many lemmas are needed to solve a problem?**
  - Only 10% of this number, and sometimes much less

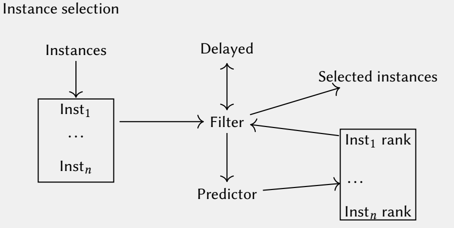
Question

Could we select the good one?

Our approach

- Instances in a priority queue
- Encode instances
- Call predictor
- Several strategies for selection



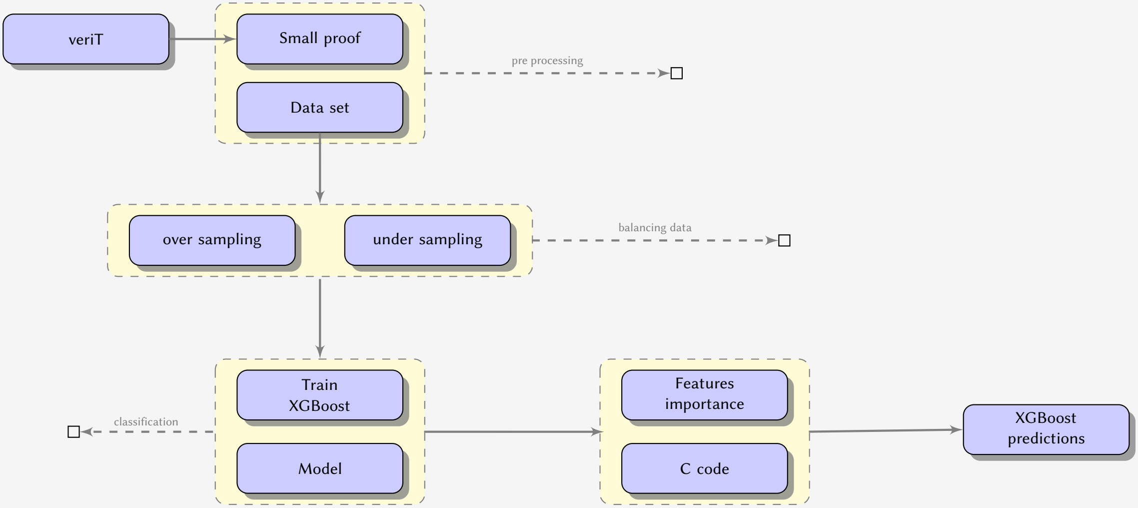


State description

$$\Big(\underbrace{l_1, \ldots, l_n}_{\text{Model}}, \underbrace{\forall \overline{x_n} \,.\, \psi[\overline{x_n}]}_{\text{Formula}}, \underbrace{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n}_{\text{Instances}}\Big)$$

$$\text{rounds}\left\{ \begin{bmatrix} (model_1 & Qformula_1^1 & Inst_{1,1}^1 & \ldots & Inst_{1,m}^1) \\ (model_1 & Qformula_1^2 & Inst_{1,1}^2 & \ldots & Inst_{1,m}^2) \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & \\ (model_k & Qformula_k^i & Inst_{k,1}^i & \ldots & Inst_{k,m}^i) \end{bmatrix} \right. \hookrightarrow \begin{bmatrix} 0 & x_{12} & x_{13} & \ldots & x_{1n} \\ 1 & x_{22} & x_{23} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & x_{d2} & x_{d3} & \ldots & x_{dn} \end{bmatrix}$$

Experiments



Contents

Time evaluation

	30s	60s	120s	180s
veriT	2892	2906	2922	2927
veriT($\mathcal{M}$)	2904	2915	2925	2936
veriT($\mathcal{M}^2$)	2914	2927	2936	2942
veriT($\mathcal{M} + \mathcal{M}^2$)	2934	2957	2968	2971
veriT + portfolio	3176	3211	3226	3232
veriT($\mathcal{M} + \mathcal{M}^2$) + portfolio	3184	3240	3307	3317
Vampire smtcomp mode	3274	3286	3297	3319
CVC4 modifed portfolio	**3311**	**3348**	**3392**	**3402**

Table: Results on the benchmarks in the UF category of the SMT-LIB.

Evaluation on test + training set

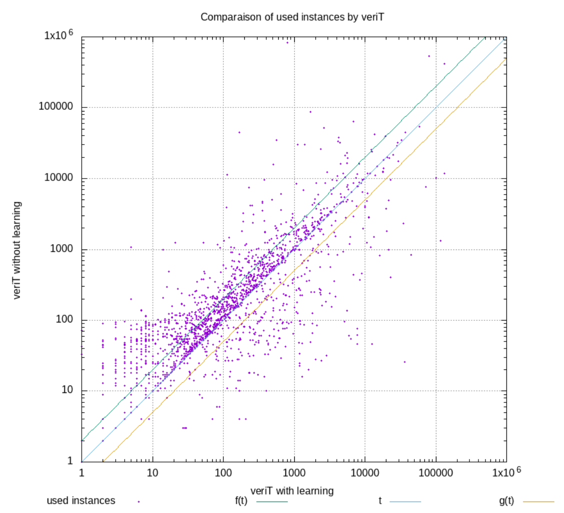


Figure: comparison of veriT configurations on UF SMT-LIB benchmarks.

Evaluation on test set only

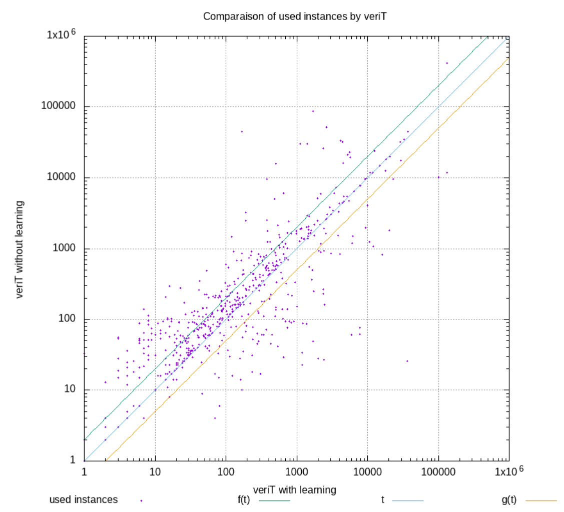


Figure: comparison of veriT configurations on UF SMT-LIB benchmarks.

Contents

Conclusions and future directions

- Could be a significant improvement
- Reduces the number of instances by two in average
- Reinforcement learning
- Features embedding can be improved

Thank you for you attention

Questions or suggestions?

Evaluation

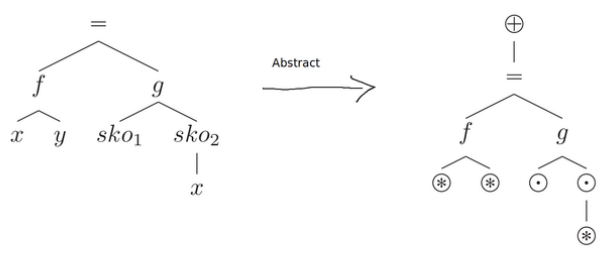
	All			*Test only*		
	unsat	avg	less	unsat	avg	less
with learning	1443	113	1317	423	130	363
without learning	1443	318	128	423	264	62

Table: veriT configurations on UF SMTLIB benchmarks with 30s timeout.

Features encoding

Terms abstraction

- Variables
- Skolem constants
- Polarity



Features

- FEATURE: Literal $\rightarrow \Sigma^3$
- FEATURES: $\Sigma^3 \rightarrow \mathbb{N}$
- Occurrences of term walks

Example

FEATURES $(f(x, y) = g(sk1, sk2(x)))$ = $(\oplus, =, f) \mapsto 1, (\oplus, =, g) \mapsto 1, (=, f, \circledast) \mapsto 2, (=, g, \odot) \mapsto 2, (g, \odot, \circledast) \mapsto 1$

State description version 2

$$\big(\underbrace{E_{t_1}, D_{t_1}, \dots, E_{t_n}, D_{t_n}}_{\text{Model}}, \underbrace{T_1, \dots T_n}_{\text{Triggers}}, \underbrace{x_1 \mapsto t_1, \dots, x_n \mapsto t_n}_{\text{Instances}}\big)$$

- E_{t_i} is the congruence class of t_i
- D_{t_i} is the set of all terms explicitly disequals with t_i
- T_i is the set of triggers of x_i

This description reduce drastically the size of the problem