Pretending to be an SMT Solver with Vampire (and How We Do Instantiation)

Giles Reger$^1$, Martin Suda$^2$, and Andrei Voronkov$^{1,2}$

$^1$School of Computer Science, University of Manchester, UK
$^2$TU Wien, Vienna, Austria

SMT 2017 – Heidelberg, July 22, 2017
Introducing Vampire

- Automatic Theorem Prover (ATP) for first-order logic
- Main paradigm: superposition calculus + saturation
- a.k.f.: indexing, incomplete strategies, strategy scheduling
Introducing Vampire

- Automatic Theorem Prover (ATP) for first-order logic
- Main paradigm: superposition calculus + saturation
- a.k.f.: indexing, incomplete strategies, strategy scheduling
Introducing Vampire

- Automatic Theorem Prover (ATP) for first-order logic
- Main paradigm: superposition calculus + saturation
- a.k.f.: indexing, incomplete strategies, strategy scheduling

Reasoning with Theories

- since 2010: progressively adding support for theories
- since 2016: participating in SMT-COMP
Two Dimensions of Complexity

∀∃

\(\mathbb{Z}/\mathbb{R}: + - \ast/\)

select/store

gnd
Two Dimensions of Complexity

∀∃
Z/R: + − ∗ /
select/store
gnd
ATP
Reasoning with quantifiers and theories

Two Dimensions of Complexity

∀ ∃

Z / R: + − ∗ /

select / store

gnd

ATP

SMT
Two Dimensions of Complexity

∀∃
Z/R: +− ∗ /
select/store
gnd
ATP
SMT
E
SPASS
VAMPIRE...
CVC4
veriT
Z3...

∀∃

ATP

E
SPASS
VAMPIRE...

CVC4
veriT
Z3...

SMT

select/store
Reasoning with quantifiers and theories

Two Dimensions of Complexity

∀∃

Z/R: +− ∗ /

select/store

gnd

ATP

SMT

select/store
Reasoning with quantifiers and theories

Two Dimensions of Complexity

∀∃

Z/R: + − ∗ /

Instantiation...
select/store
gnd
ATP

SMT

\( Z/R: + - */ \) select/store
Reasoning with quantifiers and theories

Two Dimensions of Complexity

\( \forall \exists \)

Z/R: + − ∗ /

ATP

theory axioms

...

SMT

Instantiation

...

gnd

select/store

Z/R: + − ∗ /

select/store
Reasoning with quantifiers and theories

Two Dimensions of Complexity

∀∃
Z/R: + − ∗ /
ATP
Instantiation...
theory axioms...
select/store
gnd
?

Z/R: + − */
select/store

SMT
Instantiation...
Outline

1. A Brief Introduction to Saturation-Based Proving
2. Theory Reasoning in Vampire
3. Theory Instantiation and Unification with Abstraction
4. Where We Currently Stand
Standard form of the input:

\[ F := (Axiom_1 \land \ldots \land Axiom_n) \rightarrow Conjecture \]
Standard form of the input:

\[ F := (Axiom_1 \land \ldots \land Axiom_n) \rightarrow Conjecture \]

1. Negate F to seek a refutation:

\[ \neg F := Axiom_1 \land \ldots \land Axiom_n \land \neg Conjecture \]
Theorem Proving Pipeline in One Slide

Standard form of the input:

\[ F := (\text{Axiom}_1 \land \ldots \land \text{Axiom}_n) \rightarrow \text{Conjecture} \]

1. Negate F to seek a refutation:

\[ \neg F := \text{Axiom}_1 \land \ldots \land \text{Axiom}_n \land \neg \text{Conjecture} \]

2. Preprocess and transform \( \neg F \) to clause normal form (CNF)

\[ S := \{ C_1, \ldots, C_n \} \]
Standard form of the input:

\[ F := (Axiom_1 \land \ldots \land Axiom_n) \rightarrow Conjecture \]

1. Negate F to seek a refutation:

\[ \neg F := Axiom_1 \land \ldots \land Axiom_n \land \neg Conjecture \]

2. Preprocess and transform \( \neg F \) to clause normal form (CNF)

\[ S := \{C_1, \ldots, C_n\} \]

3. Saturate \( S \) with respect to the superposition calculus

aiming to derive the obvious contradiction \( \bot \)
Saturation = fixed-point computation

Given Clause Algorithm:

- set of active clauses is stored in indexing structures
- passive works like a priority queue
- the process is “explosive” in nature
Superposition rule

\[
\begin{align*}
L \simeq r \lor C_1 & \quad L[s]_p \lor C_2 \quad \text{or} \quad L \simeq r \lor C_1 \quad t[s]_p \otimes t' \lor C_2 \\
(L[r]_p \lor C_1 \lor C_2) & \theta \\
(t[r]_p \otimes t' \lor C_1 \lor C_2) & \theta
\end{align*}
\]

where \( \theta = \text{mgu}(l, s) \) and \( r \theta \not\preceq l \theta \) and, for the left rule \( L[s] \) is not an equality literal, and for the right rule \( \otimes \) stands either for \( \simeq \) or \( \not\simeq \) and \( t' \theta \not\preceq t[s] \theta \)
Controlling the Growth of the Search Space

Superposition rule

\[
\begin{align*}
I & \simeq r \lor C_1 & L[s]_p \lor C_2 \\
\frac{L[r]_p \lor C_1 \lor C_2}{\theta} & \lor \\
& \text{or} \\
I & \simeq r \lor C_1 & t[s]_p \otimes t' \lor C_2 \\
\frac{t[r]_p \otimes t' \lor C_1 \lor C_2}{\theta}
\end{align*}
\]

where \( \theta = \text{mgu}(l, s) \) and \( r\theta \not\preceq l\theta \) and, for the left rule \( L[s] \) is not an equality literal, and for the right rule \( \otimes \) stands either for \( \simeq \) or \( \not\simeq \) and \( t'\theta \not\preceq t[s]\theta \)

Saturation up to Redundancy

- redundant clauses can be safely removed
- subsumption - an example reduction:

remove \( C \) in the presence of \( D \) such that \( D\sigma \subseteq C \)
Controlling the Growth of the Search Space

Superposition rule

\[
\frac{l \simeq r \lor C_1 \quad L[s]_p \lor C_2}{(L[r]_p \lor C_1 \lor C_2) \theta} \quad \text{or} \quad \frac{l \simeq r \lor C_1 \quad t[s]_p \otimes t' \lor C_2}{(t[r]_p \otimes t' \lor C_1 \lor C_2) \theta},
\]

where \( \theta = \text{mgu}(l, s) \) and \( r\theta \not\preceq l\theta \) and, for the left rule \( L[s] \) is not an equality literal, and for the right rule \( \otimes \) stands either for \( \simeq \) or \( \not\simeq \) and \( t'\theta \not\preceq t[s]\theta \)

Saturation up to Redundancy

- redundant clauses can be safely removed
- subsumption - an example reduction:

\[
\text{remove } C \text{ in the presence of } D \text{ such that } D\sigma \subset C
\]

Completeness considerations
Outline

1. A Brief Introduction to Saturation-Based Proving
2. Theory Reasoning in Vampire
3. Theory Instantiation and Unification with Abstraction
4. Where We Currently Stand
Basic Support for Theories

- Normalization of interpreted operations, e.g.
  \[ t_1 \geq t_2 \equiv \neg (t_1 < t_2) \quad a - b \equiv a + (-b) \]

- Evaluation of ground interpreted terms, e.g.
  \[ f(1 + 2) \equiv f(3) \quad f(x + 0) \equiv f(x) \quad 1 + 2 < 4 \equiv true \]

- Balancing interpreted literals, e.g.
  \[ 4 = 2 \times (x + 1) \equiv (4 \div 2) - 1 = x \equiv x = 1 \]

- Interpreted operations treated specially by ordering
Adding Theory Axioms

\begin{align*}
& x + (y + z) = (x + y) + z & x + 0 = x \\
& x + y = y + x & -(x + y) = (-x + -y) \\
& -x = x & x + (-x) = 0 \\
& x \cdot 0 = 0 & x \cdot (y \cdot z) = (x \cdot y) \cdot z \\
& x \cdot 1 = x & x \cdot y = y \cdot x \\
& (x \cdot y) + (x \cdot z) = x \cdot (y + z) & \neg(x < y) \lor \neg(y < z) \lor \neg(x < z) \\
& x < y \lor y < x \lor x = y & \neg(x < y) \lor \neg(y < x + 1) \\
& \neg(x < y) \lor x + z < y + z & \neg(x < x) \\
& x < y \lor y < x + 1 \text{ (for ints)} & x = 0 \lor (y \cdot x)/x = y \text{ (for reals)}
\end{align*}

- a handcrafted set
- subsets added based on the signature
- ongoing research on how to tame them [IWIL17]
The AVATAR architecture [Voronkov14]

- modern architecture of first-order theorem provers
- combines saturation with SAT-solving
- efficient realization of the \textit{clause splitting rule}

\[
\forall x, z, w. \ s(x) \lor \neg r(x, z) \lor \neg q(w) \\
\text{share } x \text{ and } z \quad \text{is disjoint}
\]

- “propositional essence” of the problem delegated to SAT solver
The AVATAR architecture [Voronkov14]

- modern architecture of first-order theorem provers
- combines saturation with SAT-solving
- efficient realization of the *clause splitting rule*

\[ \forall x, z, w. \ (s(x) \lor \neg r(x, z) \lor \neg q(w)) \]

*share* \(x\) and \(z\) *is disjoint*

- “propositional essence” of the problem delegated to SAT solver

AVATAR modulo Theories

- use an SMT solver instead of the SAT solver
- sub-problems considered are **ground-theory-consistent**
- implemented in Vampire using Z3
One Slightly Imprecise View of AVATAR

**Vampire**
- *Incremental* Theory Solver for Quantified Formulas

**SMT Solver**
- Theory Solver for Arithmetic
- Theory Solver for BitVectors
- Theory Solver for Uninterpreted Functions

**Core**
- Quantifier Instantiation

**CDCL SAT Solver**
...and please remember: Vampire is the boss here!
Outline

1. A Brief Introduction to Saturation-Based Proving
2. Theory Reasoning in Vampire
3. Theory Instantiation and Unification with Abstraction
4. Where We Currently Stand
Example

Consider the conjecture $(\exists x)(x + x \simeq 2)$ negated and clausified to

$$x + x \not\simeq 2.$$ 

It takes Vampire 15 s to solve using theory axioms deriving lemmas such as

$$x + 1 \simeq y + 1 \lor y + 1 \leq x \lor x + 1 \leq y.$$
Example

Consider the conjecture $(\exists x)(x + x \approx 2)$ negated and clausified to

$$x + x \not\approx 2.$$ 

It takes Vampire 15 s to solve using theory axioms deriving lemmas such as

$$x + 1 \approx y + 1 \lor y + 1 \leq x \lor x + 1 \leq y.$$ 

Heuristic instantiation would help, but normally any instance of a clause is immediately subsumed by the original!
Does Vampire Need Instantiation?

Example

Consider the conjecture $(\exists x)(x + x \approx 2)$ negated and clausified to

\[ x + x \not\approx 2. \]

It takes Vampire 15s to solve using theory axioms deriving lemmas such as

\[ x + 1 \approx y + 1 \lor y + 1 \leq x \lor x + 1 \leq y. \]

Heuristic instantiation would help, but normally any instance of a clause is immediately subsumed by the original!

Recall the abstraction rule

\[ L[t] \lor C \implies x \not\approx t \lor L[x] \lor C, \]

where $L$ is a theory literal, $t$ a non-theory term, and $x$ fresh.
The Theory Instantiation

Instantiation which makes some theory literals immediately false
The Theory Instantiation

Instantiation which makes some theory literals immediately false

As an inference rule

\[ \frac{C}{(D[x])\theta} \text{ TheoryInst} \]

where \( T[x] \rightarrow D[x] \) is a (partial) abstraction of \( C \) and \( \theta \) a substitution such that \( T[x]\theta \) is valid in the underlying theory
The Theory Instantiation

Instantiation which makes some theory literals immediately false

As an inference rule

\[
\frac{C}{(D[x])\theta} \quad \text{TheoryInst}
\]

where \( T[x] \rightarrow D[x] \) is a (partial) abstraction of \( C \) and \( \theta \) a substitution such that \( T[x]\theta \) is valid in the underlying theory

Implementation:

- Abstract relevant literals
- Collect relevant pure theory literals \( L_1, \ldots, L_n \)
- Run an SMT solver on \( T[x] = \neg L_1 \land \ldots \land \neg L_n \)
- If the SMT solver returns a model, transform it into a substitution \( \theta \) and produce an instance
The Theory Instantiation

Instantiation which makes some theory literals immediately false

As an inference rule

\[
\frac{C}{(D[x])\theta} \text{ TheoryInst}
\]

where \( T[x] \rightarrow D[x] \) is a (partial) abstraction of \( C \) and \( \theta \) a substitution such that \( T[x]\theta \) is valid in the underlying theory

Implementation:

- Abstract relevant literals
- Collect relevant pure theory literals \( L_1, \ldots, L_n \)
- Run an SMT solver on \( T[x] = \neg L_1 \land \ldots \land \neg L_n \)
- If the SMT solver returns a model, transform it into a substitution \( \theta \) and produce an instance
Example

Consider two clauses

\[ r(14y) \quad \neg r(x^2 + 49) \lor p(x) \]
Consider two clauses
\[
\begin{align*}
    r(14y) & \quad \neg r(x^2 + 49) \lor p(x) \\
\end{align*}
\]
We could fully abstract them to obtain:
\[
\begin{align*}
    r(u) \lor u \not\equiv 14y & \quad \neg r(v) \lor v \not\equiv x^2 + 49 \lor p(x),
\end{align*}
\]
Unification with Abstraction

Example

Consider two clauses

\[ r(14y) \quad \neg r(x^2 + 49) \lor p(x) \]

We could fully abstract them to obtain:

\[ r(u) \lor u \not\equiv 14y \quad \neg r(v) \lor v \not\equiv x^2 + 49 \lor p(x), \]

then resolve to get

\[ u \not\equiv 14y \lor u \not\equiv x^2 + 49 \lor p(x) \]
Consider two clauses

\[ r(14y) \quad \neg r(x^2 + 49) \lor p(x) \]

We could fully abstract them to obtain:

\[ r(u) \lor u \not\equiv 14y \quad \neg r(v) \lor v \not\equiv x^2 + 49 \lor p(x), \]

then resolve to get

\[ u \not\equiv 14y \lor u \not\equiv x^2 + 49 \lor p(x) \]

Finally, Theory Instantiation could produce

\[ p(7) \]
Explicit abstraction may be harmful:
- fully abstracted clauses are typically much longer
- abstraction destroys ground literals
- theory part requires special treatment
Explicit abstraction may be harmful:

- fully abstracted clauses are typically much longer
- abstraction destroys ground literals
- theory part requires special treatment

Instead of full abstraction …

- incorporate the abstraction process into unification
- thus abstractions are “on demand” and lazy
- implemented by extending the substitution tree indexing
Outline

1. A Brief Introduction to Saturation-Based Proving
2. Theory Reasoning in Vampire
3. Theory Instantiation and Unification with Abstraction
4. Where We Currently Stand
### SMT-COMP 2017 results – ∀∃ problems

<table>
<thead>
<tr>
<th>Logic</th>
<th>Vampire</th>
<th>VeriT</th>
<th>CVC4</th>
<th>Z3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIA</td>
<td>36</td>
<td>27</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>AUFDTLIA</td>
<td>624</td>
<td>-</td>
<td>728</td>
<td>-</td>
</tr>
<tr>
<td>AUFLIA</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>AUFLIRA</td>
<td>19778</td>
<td>19316</td>
<td>19766</td>
<td>19849</td>
</tr>
<tr>
<td>AUFNIRA</td>
<td>1072</td>
<td>-</td>
<td>1052</td>
<td>1031</td>
</tr>
<tr>
<td>LIA</td>
<td>229</td>
<td>170</td>
<td>388</td>
<td>388</td>
</tr>
<tr>
<td>LRA</td>
<td>1092</td>
<td>-</td>
<td>2048</td>
<td>2208</td>
</tr>
<tr>
<td>NIA</td>
<td>5</td>
<td>-</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>NRA</td>
<td>3803</td>
<td>-</td>
<td>3776</td>
<td>3805</td>
</tr>
<tr>
<td>UF</td>
<td>4317</td>
<td>3242</td>
<td>4125</td>
<td>2846</td>
</tr>
<tr>
<td>UFDT</td>
<td>2283</td>
<td>-</td>
<td>2503</td>
<td>-</td>
</tr>
<tr>
<td>UFDTLIA</td>
<td>75</td>
<td>-</td>
<td>73</td>
<td>-</td>
</tr>
<tr>
<td>UFIDL</td>
<td>55</td>
<td>55</td>
<td>60</td>
<td>59</td>
</tr>
<tr>
<td>UFLIA</td>
<td>7559</td>
<td>7518</td>
<td>7687</td>
<td>7221</td>
</tr>
<tr>
<td>UFLRA</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>UFNIA</td>
<td>2561</td>
<td>-</td>
<td>2189</td>
<td>2197</td>
</tr>
</tbody>
</table>
Thank you for your attention!