Vampire Usage and Demo

Krystof Hoder
Laura Kovacs
Andrei Voronkov

http://vprover.org/
Vampire modes

- ‘Vampire’ mode
  - uses a single specified strategy
- **CASC mode** (--mode casc)
  - selects best strategy based on problem characteristics
- **LTB mode** (--mode casc_ltb)
  - like CASC, allows solving multiple problems sharing large amounts of axioms
- **Clausify** (--mode clausify)
  - converts problem to CNF and outputs
- **Axiom selection** (--mode axiom_selection)
  - outputs axioms selected by Sine selection
- **Grounding** (--mode grounding)
  - performs grounding of EPR problems
- **Consequence elimination** (--mode consequence_elimination)
  - given set of claims, searches for relations between them
CASC Mode

• Usually the best for proving theorems
• First scan problem to determine characteristics
  – Unit, EPR, Horn, equality, large
• Then assign problem into one class
  – currently 43 classes
• Each class has a sequence of strategies that should solve problems in it
• Obtaining the strategies
  – run random strategies on a cluster of computers
  – take the best performing ones and try to further improve by doing slight changes
  – optimization techniques find the best sequence
LTB Mode

- **Strategy selection** like in CASC mode

- **Input** is a batch file according to CASC LTB specification

- **First** parse shared axioms

- **Then** add them into each of the problems
  - save on expensive parsing

- **Supports** multiprocessing
  - running multiple strategies in parallel

```%
% SZS start BatchConfiguration
division.category LTB.SMO
output.required Assurance
output.desired Proof Answer
limit.time.problem.wc 60
% SZS end BatchConfiguration
% SZS start BatchIncludes
include('Axioms/CSR003+2.ax').
include('Axioms/CSR003+5.ax').
% SZS end BatchIncludes
% SZS start BatchProblems
/TPTP/Problems/CSR/CSR083+3.p /outputs/CSR083+3
/TPTP/Problems/CSR/CSR075+3.p /outputs/CSR075+3
/TPTP/Problems/CSR/CSR082+3.p /outputs/CSR082+3
/TPTP/Problems/CSR/CSR086+3.p /outputs/CSR086+3
/TPTP/Problems/CSR/CSR091+3.p /outputs/CSR091+3
/TPTP/Problems/CSR/CSR092+3.p /outputs/CSR092+3
% SZS end BatchProblems%
```
Axiom Selection Mode

• Takes and outputs TPTP formulas/CNF
• Can be used as filter
  
  cat big_problem.tptp | vampire --mode axiom_selection | other_tool

• Performs Sine axiom selection
• Supports the Sine options (see CADE paper)
  
  --sine_tolerance (float >=1)
  --sine_depth (0,1,...)
Clausify Mode

• Converts TPTP formulas problem to CNF
  – supports typed formulas, arithmetic, answer literals

• Allows application of various Vampire preprocessing rules
  – axiom selection, transforming predicate definitions (inlining, merging, removing unused), naming, splitting,...
Grounding mode

- Converts EPR problem into propositional
- Input TPTP, output DIMACS
- Use splitting to reduce amount of variables in clauses (and therefore number of generated propositional clauses)

```prolog
fof(a1,axiom, p(X,X)).
fof(a2,axiom, p(X,Y) => p(Y,X)).
fof(a3,axiom, p(a,b)).
fof(a3,axiom, ~p(b,c)).
```
Consequence Elimination Mode

• Given a set of claims (possibly with underlying theory), attempts to **discover which claims follow from others**

  \[
  \text{fof}(c_1, \text{claim, } a \Rightarrow b).
  \]
  \[
  \text{fof}(c_2, \text{claim, } b \Rightarrow c).
  \]
  \[
  \text{fof}(c_3, \text{claim, } a \Rightarrow c).
  \]

  # vampire --mode consequence_elimination
  Pure cf clause:  \( c_2 \mid c_1 \)
  Pure cf clause:  \( \neg c_1 \mid c_3 \mid \neg c_2 \)
  Consequence found:  \( c_3 \)

  \( c_3 \) is a consequence of other claims

  clauses stating relations between claims:
  \( c_2 \mid c_1 \)
  - both \( c_1 \) and \( c_2 \) cannot be false
  \( \neg c_1 \mid c_3 \mid \neg c_2 \)
  - can be written as
  \( c_3 :\neg c_1, c_2 \)
Vampire has an API for building, manipulating, preprocessing and clausifying formulas

```
FormulaBuilder api;
Var xv = api.var("Var");
Term x = api.varTerm(xv);
Predicate p=api.predicate("p",1);
Predicate q=api.predicate("q",0);
Formula fpX=api.formula(p,x);
Formula fq=api.formula(q);
Formula fQpx=api.formula(FormulaBuilder::FORALL, xv, fpX);
Formula fQpx0q=api.formula(FormulaBuilder::OR, fQpx, fq);

AnnotatedFormula af=api.annotatedFormula(fQpx0q,FormulaBuilder::CONJECTURE, "conj1");
Problem prb;
prb.addFormula(af);
prb.output(cout);

Problem cprb=prb.clausify(0,false,Problem::INL_OFF,false);
cprb.output(cout);
```
Solution Output

- **Proof**
  - may use TPTP format
- **Interpolant** (see Session 3)
- **Answer**
  - for existentially quantified conjectures
- **Model**
  - currently only for certain strategies on EPR problems
Proofs

2_01_proof_ex.tptp:
cnf(commutativity, axiom, \( f(X,Y) = f(Y,X) \)).
cnf(identity, axiom, \( f(i,X) = X \)).
fof(c, conjecture, \( \neg [X]: f(j,X) = X \Rightarrow j = i \)).

22. $\texttt{false}$ (2:0) [subsumption resolution 16,7]
7. \( i \neq j \) (0:3) [cnf transformation 5]
5. \( \neg [X0] : f(j,X0) = X0 \) & \( i \neq j \)[ennf transformation 4]
4. \( \neg(\neg [X0] : f(j,X0) = X0 \Rightarrow i = j) \)[negated conjecture 3]
3. \( \neg [X0] : f(j,X0) = X0 \Rightarrow i = j \)[input]
16. \( i = j \) (2:3) [superposition 8,2]
2. \( f(i,X0) = X0 \) (0:5) [input]
8. \( f(X0,j) = X0 \) (1:5) [superposition 1,6]
6. \( f(j,X0) = X0 \) (0:5) [cnf transformation 5]
1. \( f(X0,X1) = f(X1,X0) \) (0:7) [input]
Proofs

**Vampire native** format:
11. \( \neg \text{female}(\text{X0}) \lor \neg \text{from_venus}(\text{X0}) \lor \text{truthteller}(\text{X0}) \) (0:6) [input]

48_2. \( \text{false} \lor (\neg \text{bdd4} \land (\text{bdd3} \land \text{bddnode1})) \) (2:0) [merge 48_3,107_1]

BDD definition: \( \text{bddnode1} = (\text{bdd2} \lor \text{bdd1} : \neg \text{bdd1}) \)

**TPTP proof format:**

```prolog
fof(f11,axiom,(  
  ( ! [X0] : (\neg \text{female}(\text{X0}) \lor \neg \text{from_venus}(\text{X0}) \lor \text{truthteller}(\text{X0}))))),
file('Problems/PUZ/PUZ007-1.p',unknown)).
fof(f48_2,plain,(  
  \( \text{false} \lor ((\neg \text{bdd4} \Rightarrow \text{false}) \land (\neg \text{bdd4} \Rightarrow ( (\text{bdd2} \Rightarrow \text{bdd1}) \land (\neg \text{bdd2} \Rightarrow \neg \text{bdd1})))) \land (\neg \text{bdd3} \Rightarrow \text{false}))))),
inference(merge,[],[f48_3,f107_1])).
```

**LaTeX output:**

\begin{align*}
[11, \text{input}] & \\
\neg \text{female()} \lor \neg \text{from_venus()} \lor \text{truthteller()} & \\
[48_3,107_1 \rightarrow 48_2, \text{merge}] & \\
\Box \lor n_1 & \\
\Box \lor (\neg b_4 \lor b_1) & \\
\Box \lor (\neg b_4 \land (b_3 \land n_0)) & \\
\quad n_0 \leftrightarrow (b_2 \land b_1 : \neg b_1) & \\
\quad n_1 \leftrightarrow (b_4 \land (\neg b_3 \land (\neg b_2 \land \neg b_1)) : (b_3 \land n_0)) & \\
\end{align*}
Question Answering

2_02_answer_ex.tptp:
fof(a1,axiom,son("jimmy","jane")).
fof(a2,axiom,son("johny","jane")).
fof(a3,axiom, (son(X,Z) & son(Y,Z) & X!=Y) => brother(X,Y)).
fof(q,question, ?[X] : brother("jimmy", X)).

# vampire PROBLEM.p -question_answering answer_literal
% SZS answers Tuple [["johny"]|_] for PROBLEM2
23. $false (0:0) [unit resulting resolution 22,21]
21. ~sP0_ans("johny") (1:2) [resolution 20,15]
15. ~brother("jimmy",X0) | ~sP0_ans(X0) (0:5) [cnf transformation 10]
10. ! [X0] : (~sP0_ans(X0) | ~brother("jimmy",X0))[ennf transformation 6]
 6. ~? [X0] : (sP0_ans(X0) & brother("jimmy",X0))[answer literal 5]
 5. ~? [X0] : brother("jimmy",X0)[negated conjecture 4]
 4. ? [X0] : brother("jimmy",X0)[input]
20. brother("jimmy","johny") (0:3) [distinct equality removal 19]
19. "jimmy" = "johny" | brother("jimmy","johny") (2:6) [resolution 16,13]
...

# vampire PROBLEM.p -sa inst_gen -updr off
Refutation not found!

fof(model1,interpretation_domain,
   ! [X] : ( X = "d" | X = "c" | X = "b" | X = "a" ) ).

fof(model2,interpretation_terms,
   ( b = "b" & d = "d" & a = "a" & c = "c" ) ).

fof(model3,interpretation_atoms,
   ( p("c","d") &
     p("d","c") &
     ~p("b","d") &
     ~p("d","b") &
     ~p("b","c") &
     ~p("c","b") &
     p("d","d") &
     p("c","c") &
     p("a","b") &
     ~p("a","c") &
     ~p("c","a") &
     p("b","b") &
     p("a","a") &
     ~p("a","d") &
     ~p("d","a") &
     p("b","a") ) ).
Input Language

- **Sorts**

  \[
  \text{tff(list\_type, type, (}
  \text{list: } \text{tType }))
  \]
  \[
  \text{tff(nil\_type, type, (}
  \text{nil: } \text{list }))
  \]
  \[
  \text{tff(cons\_type, type, (}
  \text{cons: ( }\text{int} \times \text{list } > \text{list } ))}
  \]

- **If-then-else (both for terms and formulas)**

  \[
  \text{tff(c1, axiom, } \text{itef(p \& q, } \neg \text{p} | \neg \text{q, p \& q) )}
  \]
  \[
  \text{sP0 } \leftrightarrow \text{ (p \& q)}
  \]
  \[
  \text{((p \& q) } \& \text{~sP0) } | \text{ ((} \neg \text{p} | \neg \text{q) } \& \text{sP0)}
  \]

  \[
  \text{tff(c2, axiom, } \text{itet(p,a,b) } \neq \text{ a } \& \text{ p) ).}
  \]
  \[
  \text{\text{itef(p, sG0(X0,X1) = X0, sG0(X0,X1) = X1)}}
  \]
  \[
  \text{sG0(a,b) } \neq \text{ a } \& \text{ p}
  \]
Input Language

- Let...in
  - inside terms or formulas
  - assigning to functions or predicates

```
tff(c1, axiom, $lettt(f(X), g(X), f(a)) != g(a) ).
g(a) != g(a)

tff(c2, axiom, $letff(p(X), q(X) | r(X), p(c)) & ¬q(c) & ¬r(c) ).
  (q(c) | r(c)) & ¬q(c) & ¬r(c)

tff(c3, axiom, $lettf(f(X), g(X), p(f(X))) & ¬p(g(X)) )
  ! [X1 : (p(g(X1)) & ¬p(g(X1)))]

tff(c4, axiom, $letf(p(X), q, $itet(p(a), a, b)) != $itet(q, a, b) ).
  $itet(q, sG0(X0, X1) = X0, sG0(X0, X1) = X1)
  $itet(q, sG1(X0, X1) = X0, sG1(X0, X1) = X1)
  sG0(a, b) != sG1(a, b)
```
Arithmetic

- TFA arithmetic syntax specified in the TPTP standard
  - integers, rationals, reals
- Currently we
  - add axioms for the interpreted symbols present in the problem
  - evaluate interpreted expressions with numeric arguments
    - e.g. $10 < 5 + 3 \rightarrow 10 < 8 \rightarrow \bot$

2_04_arith_ex.tptp:
```
tff(f_type, type, (f: $\text{int} > $\text{int})).
```
```
tff(integers, axiom, ![Y: $\text{int}]: ![X: $\text{int}]: (f(X) = $\text{sum}(X, Y)) ).
```
```
tff(integers, conjecture, ![X: $\text{int}, Y: $\text{int}]: ($\text{less}(f(X), f(Y)) <=> $\text{less}(X, Y)) ).
```

2_05_arith_answer.tptp:
```
tff(integers, question, ![X: $\text{int}]: ($\text{product}(X, X) = $\text{sum}(X, X) \& X != 0)).
```
```
% SZS status Theorem for alt_2_05_arith_answer_ex
% SZS answers Tuple [(2) [ ]] for alt_2_05_arith_answer_ex
% SZS output start Proof for alt_2_05_arith_answer_ex
450. $\text{false}$ (0:0) [unit resulting resolution 449,448]
448. $\sim$\text{P0\_ans}(2) (0:2) [distinct equality removal 447]
447. 0 = -1 | $\sim$\text{P0\_ans}(2) (8:5) [trivial inequality removal 446]
446. 4 != 4 | 0 = -1 | $\sim$\text{P0\_ans}(2) (8:8) [evaluation 445]
445. $\text{product}(2, 2) != $\text{uminus}(-4) | 0 = -1 | $\sim$\text{P0\_ans}(2) (8:11)
   [evaluation 444]
444. $\text{product}($\text{uminus}(-2),$\text{uminus}(-2)) != $\text{uminus}($\text{sum}(-2,-2))
   | $\text{sum}(1,-2) = 0 | $\sim$\text{P0\_ans}($\text{uminus}(-2)) (8:18) [evaluation 443]
...
Preprocessing

• Eliminate if-then-else and let...in terms and formulas
• Sine selection
• Predicate definitions and EPR
  – Skolemization of definitions such as “p(X) <=> F[X]” introduces non-constant functions
  – if all occurrences of p(X) are ground, this is not necessary
  – blind inlining may be infeasible (exponential blow-up)
  – Vampire has several rules to deal with this situation
• Removal of trivial predicates
  – E.g. “p(X) | ~p(b)” “p(a)”
• Equivalent predicate discovery, naming, splitting, detecting Horn structure,...
Strategies

• **Saturation** (*Discount, Otter, LRS*)
  – splitting (backtracking, without backtracking)
  – BDDs (to represent propositional predicates)
  – global subsumption resolution
  – unit-resulting resolution
• **Tabulation**
• **Instantiation**
  – InstGen calculus

• **Instantiation and saturation** can run in parallel
  – saturation clauses are used in the InstGen literal selection
  – global subsumption resolution indexes are shared
New symbol introduction

- Some Vampire rules may introduce **new symbols**
  - in certain applications (**interpolation**) this is not desirable
  - some such rules cannot be disabled (**skolemization**), other can
- **BDDs** (**introducing prop. predicates for BDD variables**)
  - `--forced_options propositional_to_bdd=off`
- **Splitting** (**introducing prop. predicates for decision points**)
  - `--forced_options splitting=off`
- **Other rules**
  - `equality_proxy, general_splitting, inequality_splitting`
- **Naming** introduces new predicates to avoid exponential blow-up during clausification
  - setting naming to larger values will lead to less introduced names, 0 disables it
    - `naming=32000`
    - `naming=0`
    - `naming=8` (default)
- **To disable all of the above**
  - `--forced_options
    propositional_to_bdd=off:splitting=off:equality_proxy=off:
    general_splitting=off:inequality_splitting=0:naming=0`
## Overview

<table>
<thead>
<tr>
<th>Usage</th>
<th>Supported input</th>
<th>Solving strategies</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single strategy</td>
<td>Sorts</td>
<td>Saturation</td>
<td>Sine selection</td>
</tr>
<tr>
<td>CASC mode</td>
<td>Arithmetic</td>
<td>Tabulation</td>
<td>EPR restoring</td>
</tr>
<tr>
<td>LTB</td>
<td>If-then-else</td>
<td>InstGen</td>
<td>Trivial predicate removal</td>
</tr>
<tr>
<td>Clausifier</td>
<td>Let...in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Axiom selection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consequence elimination</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grounding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Solution output</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proof</td>
<td></td>
<td>Inlining definitions</td>
</tr>
<tr>
<td></td>
<td>Interpolant</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Answer</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://vprover.org/