Vampire: a Resolution-Based Theorem Prover for First-Order Logic

Lecturer: Will Klieber

April 25, 2012
Vampire is an automated theorem prover for FOL, developed by Andrei Voronkov, Alexandre Riazanov, and Krystof Hoder.

**Uses:**
- Assisting mathematicians.
- Formal verification of software and hardware.
- Automatic synthesis of software and hardware.
- Symbolic computer algebra systems.
- Knowledge representation in AI.

Vampire has consistently won the CADE Automated Theorem Proving competition for many years.
Goal

- Given: A formula $\phi$ in First-Order Logic (FOL).
- Goal: Find out whether $\phi$ is valid (true under all interpretations).
Goal

- Given: A formula $\phi$ in First-Order Logic (FOL).
- Goal: Find out whether $\phi$ is valid (true under all interpretations).
- Equivalently: Is $\neg \phi$ satisfiable?

Undecidable in general.

Vampire is refutationally complete in the following sense:
- If there exists a proof in FOL that $\neg \phi$ is unsat, then Vampire will find such a proof.
Given: A formula $\phi$ in First-Order Logic (FOL).

Goal: Find out whether $\phi$ is valid (true under all interpretations).

Equivalently: Is $\neg\phi$ satisfiable?

Undecidable in general.

Vampire is \textit{refutationally complete} in the following sense:

If a there exists a proof in FOL that $\neg\phi$ is unsat, then Vampire will find such a proof.
At a high-level, Vampire behaves as follows:

1. Input: Formula $\phi$ in FOL.
2. Skolemize and clausify $\phi$. (Preprocessing)
3. Now $\phi$ has the form of a conjunction of a set of clauses.
At a high-level, Vampire behaves as follows:

1. Input: Formula $\phi$ in FOL.
2. Skolemize and clausify $\phi$. (Preprocessing)
3. Now $\phi$ has the form of a conjunction of a set of clauses.
4. Repeat:
   4.1 Infer clauses via resolution and/or paramodulation
   4.2 If added the empty (contradictory) clause, return “unsat”.
   4.3 If we have made all possible inferences, then return “sat”.
   4.4 Try to simplify the set of clauses.
Syntax of First-Order Logic (FOL)

FOL has symbols for:

- Constants, Variables, Functions, Predicates

A term is:

- a constant or variable, or
- a function expression $f(t_1, ..., t_n)$ where $t_1, ..., t_n$ are terms.
Syntax of First-Order Logic (FOL)

FOL has symbols for:

- Constants, Variables, Functions, Predicates

A term is:

- a constant or variable, or
- a function expression $f(t_1, ..., t_n)$ where $t_1, ..., t_n$ are terms.

A well-formed formula (wff) is:

- an atomic formula $P(t_1, ..., t_n)$ where $t_1, ..., t_n$ are terms,
- $\phi_1 \land \phi_2$, $\phi_1 \lor \phi_2$, $\phi_1 \Rightarrow \phi_2$, $\phi_1 \Leftrightarrow \phi_2$, where $\phi_1$ and $\phi_2$ are wffs,
- $\forall x.\phi$, or $\exists x.\phi$, where $\phi$ is a wff and $x$ is a variable.

Functions and predicates (except equality) are uninterpreted.
Example: \( X \neq add(Y, 1) \lor odd(X) \lor odd(Y) \)

- **Terms:** \( X, \; add(Y, 1), \; 1, \; Y \).
- **Variables:** \( X, \; Y \).
- **Constant:** 1.
- **Proper Subterms of** \( add(Y, 1) \): \( Y, \; 1 \).
Clausal form

- Atomic formula: $P(t_1, ..., t_n)$
  - If $n = 0$, $P$ is a nullary predicate and we may omit the parens.
- A literal is an atomic formula or its negation.
Clausal form

- **Atomic formula:** $P(t_1, ..., t_n)$
  - If $n = 0$, $P$ is a nullary predicate and we may omit the parens.
- A *literal* is an atomic formula or its negation.

**Caution:**
- SAT literature uses the word “variables” for what we call ”nullary predicates”.
- What we call “variables” here are not literals.
Clausal form

- **Atomic formula**: \( P(t_1, \ldots, t_n) \)
  - If \( n = 0 \), \( P \) is a nullary predicate and we may omit the parens.
- A **literal** is an atomic formula or its negation.
- A **clause** is a disjunction of literals.
- A formula \( \phi \) is in **Conjunctive Normal Form (CNF)** iff \( \phi \) is the conjunction of clauses.
Clausal form

- **Atomic formula:** $P(t_1, \ldots, t_n)$
  - If $n = 0$, $P$ is a nullary predicate and we may omit the parens.
- A *literal* is an atomic formula or its negation.
- A *clause* is a disjunction of literals.
- A formula $\phi$ is in **Conjunctive Normal Form (CNF)** iff $\phi$ is the conjunction of clauses.

- Predicates and functions are implicitly existentially quantified.
- We require all variables to be universally quantified.
  - Quantifiers may appear either (1) on each clause or (2) at the head of CNF formula (prenex).
  - Universal quantification distributes over conjunction.
Clausification

1. Convert to Negation Normal Form (NNF).
   - Ensures quantifiers occur only positively (not negatively).
2. Skolemize to remove existential quantifiers.
3. Convert to *prenex* form \((Q_1 x_1 \ldots Q_n x_n \cdot \phi_{\text{matrix}})\).
4. Convert to Conjunctive Normal Form (CNF).
Skolemization

- Skolemization is a step of preprocessing.
- We replace each existential variable with a function expression.
- \( \forall x (g(x) \lor \exists y. R(x, y)) \iff \exists f. \forall x. (g(x) \lor R(x, f(x))) \)

Original Formula \( \iff \) Skolemized

\[
\forall x (g(x) \lor \exists y. R(x, y)) \\
= \exists f. \forall x (g(x) \lor \exists y. R(x, y) \land y = f(x)) \\
= \exists f. \forall x (g(x) \lor R(x, f(x)))
\]

- Drop the 2nd-order quantifier. Result is equisatisfiable.
  - Functions are implicitly existentially quantified.
We have now finished discussing preprocessing the FOL formula in clausal form.

We will now discuss Vampire’s kernel, which uses *resolution* to make inferences.
Given-Clause Algorithm

```
var new, passive, active : sets of clauses
var current : clause
active := ∅
passive := set of input clauses
while passive ≠ ∅ do
    current := select(passive)
    passive := passive − {current}
    active := active ∪ {current}
    new := infer(current, active)
    if new contains empty clause
        then return refutable
    passive := passive ∪ new
od
return failure to refute
```

(A. Riazanov, “Implementing an Efficient Theorem Prover” (2003), page 42.)
Resolution in Propositional Logic

From

\[(p_1 \lor \ldots \lor p_n \lor r) \land (\neg r \lor q_1 \lor \ldots \lor q_m)\]

Infer

\[(p_1 \lor \ldots \lor p_n \lor q_1 \lor \ldots \lor q_m)\]

- In a FOL formula with only ground terms (i.e., no variables), resolution works basically the same.
Example of Ground Resolution

Show $\phi$ is unsat where $\phi = G_1 \land G_2 \land G_3 \land G_4$ and:

\[
\begin{align*}
G_1 & = \neg P(a,a) \\
G_2 & = P(a,f(a)) \lor P(a,a) \\
G_3 & = P(f(a),a) \lor P(a,a) \\
G_4 & = \neg P(f(a),a) \lor \neg P(a,f(a))
\end{align*}
\]
Example of Ground Resolution

Show $\phi$ is unsat where $\phi = G_1 \land G_2 \land G_3 \land G_4$ and:

$G_1 = \neg P(a, a)$

$G_2 = P(a, f(a)) \lor P(a, a)$

$G_3 = P(f(a), a) \lor P(a, a)$

$G_4 = \neg P(f(a), a) \lor \neg P(a, f(a))$

Resolution in First-Order Logic

- In non-ground resolution, we can substitute for variables.
- Let $\sigma$ be a substitution (a mapping from variables to terms).
- We write “$\sigma(\phi)$” to denote the result of substituting each assigned variable with the assigned term in $\phi$. 
For example, consider the clauses

\[ C_1 = \neg P(z_1, a) \lor \neg P(z_1, x) \lor \neg P(x, z_1) \]
\[ C_2 = P(z_2, f(z_2)) \lor P(z_2, a) \]
\[ C_3 = \neg P(a, f(a)) \]

- \( x, z_1, \) and \( z_2 \) are variables; \( a \) is a constant.
- Can we derive a contradiction?
For example, consider the clauses

\[ C_1 = \neg P(z_1, a) \lor \neg P(z_1, x) \lor \neg P(x, z_1) \]
\[ C_2 = P(z_2, f(z_2)) \lor P(z_2, a) \]
\[ C_3 = \neg P(a, f(a)) \]

- \( x, z_1, \) and \( z_2 \) are variables; \( a \) is a constant.
- Consider the substitution \( \sigma = \{ z_1 \mapsto a, \ x \mapsto a, \ z_2 \mapsto a \} \):
  \[
  \sigma(C_1) = \neg P(a, a) \\
  \sigma(C_2) = P(a, f(a)) \lor P(a, a)
  \]
- Under \( \sigma \), the three disjuncts in \( C_1 \) unify to \( \neg P(a, a) \).
- \( \sigma(C_1) \) and \( \sigma(C_2) \) resolve to \( \neg C_3 \).
Unification

- A substitution $\sigma$ is a unifier of terms $e_1, \ldots, e_n$ iff
  $\sigma(e_1) = \ldots = \sigma(e_n)$ where “$=$” denotes syntactic identity.

- A unifier $\sigma$ is a most general unifier (mgu) of $e_1, \ldots, e_n$ iff:
  for every unifier $\sigma'$ of $e_1, \ldots, e_n$ there exists a substitution $\sigma''$
  such that $\sigma'(e_i) = \sigma''(\sigma(e_i))$ for all $e_i \in \{e_1, \ldots, e_n\}$. 
A substitution $\sigma$ is a **unifier** of terms $e_1, \ldots, e_n$ iff
\[ \sigma(e_1) = \ldots = \sigma(e_n) \] where “=” denotes syntactic identity.

A unifier $\sigma$ is a **most general unifier (mgu)** of $e_1, \ldots, e_n$ iff:
for every unifier $\sigma'$ of $e_1, \ldots, e_n$ there exists a substitution $\sigma''$ such that $\sigma'(e_i) = \sigma''(\sigma(e_i))$ for all $e_i \in \{e_1, \ldots, e_n\}$.

If a set of terms of FOL can be unified, there exists a mgu.

There is an efficient algorithm for determining whether a set of terms can be unified and, if so, computing the mgu.
Unification

- A substitution $\sigma$ is a unifier of terms $e_1, \ldots, e_n$ iff
  $\sigma(e_1) = \ldots = \sigma(e_n)$ where “$=$” denotes syntactic identity.

- A unifier $\sigma$ is a most general unifier (mgu) of $e_1, \ldots, e_n$ iff:
  for every unifier $\sigma'$ of $e_1, \ldots, e_n$ there exists a substitution $\sigma''$
  such that $\sigma'(e_i) = \sigma''(\sigma(e_i))$ for all $e_i \in \{e_1, \ldots, e_n\}$.

- Examples:
  1. Mgu of $P(x, y)$ and $P(y, f(z))$?
  2. Can $x$ and $f(x)$ be unified?
  3. Mgu of $P(a, y, f(y))$ and $P(z, z, u)$? ($a$ is a constant.)
  4. Mgu of $P(x, g(x), y)$ and $P(z, u, g(u))$. 
Unification

- A substitution \( \sigma \) is a \textit{unifier} of terms \( e_1, \ldots, e_n \) iff
  \[ \sigma(e_1) = \ldots = \sigma(e_n) \]  where “\( = \)” denotes syntactic identity.

- A unifier \( \sigma \) is a \textit{most general unifier (mgu)} of \( e_1, \ldots, e_n \) iff:
  for every unifier \( \sigma' \) of \( e_1, \ldots, e_n \) there exists a substitution \( \sigma'' \) such that
  \[ \sigma'(e_i) = \sigma''(\sigma(e_i)) \]  for all \( e_i \in \{e_1, \ldots, e_n\} \).

- Examples:
  1. Mgu of \( P(x, y) \) and \( P(y, f(z)) \)?
  2. Can \( x \) and \( f(x) \) be unified? \textbf{No, unification is syntactic.}
  3. Mgu of \( P(a, y, f(y)) \) and \( P(z, z, u) \)? (\( a \) is a constant.)
  4. Mgu of \( P(x, g(x), y) \) and \( P(z, u, g(u)) \).
Binary Resolution Rule (Simple version)

From

\[(C \lor A_1) \land (\neg B_1 \lor D)\]

infer

\[\sigma(C \lor D)\]

where \(\sigma\) is the mgu of the atoms \(A_1\) and \(B_1\).

Examples. Find all resolvents of the following:

1. \((P(x, y) \lor P(y, z)) \land \neg P(u, f(u))\)
2. \((P(x, x) \lor \neg R(x, f(x))) \land (R(x, y) \lor Q(y, z))\)
Binary Resolution Rule

From

$$(C \lor A_1 \lor \ldots \lor A_m) \land (\neg B_1 \lor \ldots \lor \neg B_n \lor D)$$

infer

$$\sigma(C \lor D)$$

where $\sigma$ is the mgu of the atoms $A_1, \ldots, A_m, B_1, \ldots, B_n$. 
Example

- Establish the unsatisfiability of the following by resolution:

\[ \forall x \exists y P(x, y) \land \exists x \forall y \neg P(x, y) \]
Example from Riazanov’s thesis

Theory:
\[
\begin{align*}
\forall x \text{ person}(x) & \Rightarrow \exists y \text{ person}(y) \land \text{ female}(y) \land \text{ parent}(y, x) \\
\forall x, y, z \text{ parent}(y, x) \land \text{ parent}(z, y) & \Rightarrow \text{ grandparent}(z, x)
\end{align*}
\]

Goal:
\[
\forall x \text{ person}(x) \Rightarrow \exists y \text{ grandparent}(y, x) \land \text{ female}(y)
\]
Example from Riazanov’s thesis

Theory:
\[
\begin{align*}
\forall x \ person(x) & \Rightarrow \exists y \ person(y) \land \ female(y) \land \ parent(y, x) \\
\forall x, y, z \ parent(y, x) \land \ parent(z, y) & \Rightarrow \ \text{grandparent}(z, x)
\end{align*}
\]

Goal:
\[\forall x \ person(x) \Rightarrow \exists y \ \text{grandparent}(y, x) \land \ female(y)\]

After replacing the goal formula with its negation, this formula set is satisfiable if and only if the following clause set is:

1. \[\neg \ person(x) \lor \ person(\text{mother}(x))\]
2. \[\neg \ person(x) \lor \ female(\text{mother}(x))\]
3. \[\neg \ person(x) \lor \ parent(\text{mother}(x), x)\]
4. \[\neg \ parent(y, x) \lor \ \neg \ parent(z, y) \lor \ \text{grandparent}(z, x)\]
5. \[\ person(\text{somebody})\]
6. \[\neg \ \text{grandparent}(y, \text{somebody}) \lor \neg \ female(y)\]
Example from Riazanov’s thesis (cont.)

\[
\neg \text{person}(x) \lor \text{parent}(\text{mother}(x), x) \quad \neg \text{parent}(x, y) \lor \neg \text{parent}(y, z) \lor \text{grandparent}(x, z)
\]

\[
\neg \text{female}(x) \lor \neg \text{grandparent}(x, \text{somebody})
\]

\[
\neg \text{person}(x) \lor \text{parent}(\text{mother}(x), x) \quad \neg \text{parent}(x, \text{somebody}) \lor \neg \text{female}(\text{mother}(x)) \lor \neg \text{person}(x)
\]

\[
\neg \text{female}(\text{mother}(\text{mother}(\text{somebody}))) \lor \neg \text{person}(\text{mother}(\text{somebody})) \lor \neg \text{person}(\text{somebody})
\]

\[
\neg \text{person}(x) \lor \text{female}(\text{mother}(x))
\]

\[
\neg \text{person}(\text{mother}(\text{somebody})) \lor \neg \text{person}(\text{mother}(\text{somebody})) \lor \neg \text{person}(\text{somebody})
\]

\[
\neg \text{person}(x) \lor \text{person}(\text{mother}(x))
\]

\[
\neg \text{person}(\text{somebody}) \lor \neg \text{person}(\text{somebody})
\]

\[
\text{person}(\text{somebody})
\]

\[
\neg \text{person}(\text{somebody})
\]
So far, we’ve considered FOL without an equality predicate. For equality, we use *paramodulation*:

From

\[(C \lor s \simeq t) \land D[u]_k\]

infer

\[\sigma(C \lor D[t]_k)\]

where \(\sigma\) is the mgu of \(s\) and \(u\), and \(D[u]_k\) denotes substitution of \(u\) into \(k^{th}\) position of \(D\), and “\(\simeq\)” denotes semantic equality in object language.
Reflexivity

From

\[ C' \lor s \not\approx t \]

infer

\[ \sigma(C') \]

where \( \sigma \) is the mgu of \( s \) and \( t \).
Conclusion

- Vampire is the fastest Automated Theorem Prover on the CADE benchmarks.
- To prove a formula valid, we prove that its negation is unsatisfiable.
- We convert the negated formula into clausal form.
- We apply resolution on the clausal form until contradiction, saturation, or timeout.
  - This is refutationally complete method and often fast.
Further Reading


