Proof Search in Conflict Resolution
Lifting CDCL
(Conflict-Driven Clause Learning)
to First-Order Logic

Bruno Woltzenlogel Paleo

joint work with:
Daniyar Itegulov (ITMO, St. Petersburg, Russia)
Ezequiel Postan (National University of Rosario, Argentina)
John Slaney (Australian National University)
**modus ponens**

\[
\begin{align*}
A & \\ A \rightarrow B & \\ B
\end{align*}
\]

**resolution**

\[
\begin{align*}
\Gamma_1 \rightarrow \Delta_1, A & \\ A', \Gamma_2 \rightarrow \Delta_2 & \\ (\Gamma_1, \Delta_1 \rightarrow \Gamma_2, \Delta_2) & \sigma
\end{align*}
\]

**hypothetical reasoning**

\[
\begin{align*}
[A] & \\ \vdots & \\ \vdots & \\ B & \\ A \rightarrow B
\end{align*}
\]
### Results of CASC (2016)

<table>
<thead>
<tr>
<th>Higher-order Theorems</th>
<th>Satallax</th>
<th>Satallax</th>
<th>LEO-II</th>
<th>Leo+III</th>
<th>Leo-III</th>
<th>Isabelle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved: 500</td>
<td>346,500</td>
<td>315,500</td>
<td>238,500</td>
<td>89,500</td>
<td>74,500</td>
<td>356,500</td>
</tr>
<tr>
<td>Av. CPU Time</td>
<td>22.10</td>
<td>19.45</td>
<td>20.93</td>
<td>48.37</td>
<td>42.79</td>
<td>81.08</td>
</tr>
<tr>
<td>Solutions</td>
<td>327,500</td>
<td>313,500</td>
<td>231,500</td>
<td>88,500</td>
<td>74,500</td>
<td>0,500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Typed First-order Theorems</th>
<th>Vampire</th>
<th>VampireZ</th>
<th>CVC4</th>
<th>Beagle</th>
<th>Princess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved: 500</td>
<td>419,500</td>
<td>380,500</td>
<td>343,500</td>
<td>300,500</td>
<td>342,500</td>
</tr>
<tr>
<td>Av. CPU Time</td>
<td>13.39</td>
<td>9.15</td>
<td>5.72</td>
<td>18.76</td>
<td>17.59</td>
</tr>
<tr>
<td>Solutions</td>
<td>419,500</td>
<td>380,500</td>
<td>343,500</td>
<td>300,500</td>
<td>271,500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First-order Non-theorems + &amp;/ -</th>
<th>Vampire</th>
<th>CVC4</th>
<th>Beagle</th>
<th>Princess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved: 50</td>
<td>10.50</td>
<td>9.50</td>
<td>8.50</td>
<td>8.50</td>
</tr>
<tr>
<td>Av. CPU Time</td>
<td>2.11</td>
<td>0.02</td>
<td>1.44</td>
<td>22.90</td>
</tr>
</tbody>
</table>

### Very Sketchy Anatomy of Winning ATPs

#### First/Higher-Order Theorem Prover

![sat-solver](image)
Let’s Open the Black Box!
Implication/Conflict Graphs: Unit Propagation

$c_1: \neg P$
$c_2: R$
$c_3: \neg R \lor P \lor Q$
$c_4: P \lor \neg Q$
Unit-Propagating Resolution

\[
\begin{align*}
\ell_1 & \quad \ldots \quad \ell_n & \quad \bar{\ell}_1 \lor \ldots \lor \bar{\ell}_n \lor \ell \\
\ell & \quad u
\end{align*}
\]

\[
\begin{align*}
\ell & \quad \bar{\ell} \\
\ell & \quad \bar{\ell} \\
\bot & \quad c
\end{align*}
\]
Implication/Conflict Graphs: Unit Propagation

\[
\begin{align*}
\text{c}_1 &: \neg P \\
\text{c}_2 &: R \\
\text{c}_3 &: \neg R \lor P \lor Q \\
\text{c}_4 &: P \lor \neg Q
\end{align*}
\]

\[
\begin{array}{c}
\text{c}_2 \rightarrow \text{R} \\
\text{c}_1 \rightarrow \neg P \\
\text{c}_3 \rightarrow \neg R \lor P \lor Q \\
\text{c}_4 \rightarrow P \lor \neg Q
\end{array}
\]

\[
\begin{array}{c}
\text{Q} \\
\text{u} \\
\text{c}
\end{array}
\]
Implication/Conflict Graphs: Decision Literals

$c_1 : P \lor Q$
$c_2 : P \lor \neg Q$
$c_3 : \neg P \lor Q$
$c_4 : \neg P \lor \neg Q$
$c_5 : \neg P$
Implication/Conflict Graphs

Backtrack and Iterate...

\( c_1 : P \lor Q \)
\( c_2 : P \lor \neg Q \)
\( c_3 : \neg P \lor Q \)
\( c_4 : \neg P \lor \neg Q \)
\( c_5 : \neg P \)
Implication/Conflict Graphs: Decision Literals

Decision literals behave like assumptions

learning a clause is like applying natural deduction’s negation introduction rule

\[
\begin{align*}
[P] \\
\vdots \\
\neg P & \quad \neg I
\end{align*}
\]
Decisions and Conflict-Driven Clause Learning

This can also be a non-tree DAG

\[ \ell_1 \lor \ldots \lor \ell_n \rightarrow \neg P \equiv P \rightarrow \bot \]
First-Order Logic

Propositional Logic
First-Order Unit-Propagation

\[
\frac{\ell_1 \ldots \ell_n \ \bar{\ell}'_1 \lor \ldots \lor \bar{\ell}'_n \lor \ell}{\ell \ \sigma} \quad \mathbf{u}(\sigma)
\]

\[
\frac{\ell \ \bar{\ell}'}{\vert} \quad \mathbf{c}(\sigma)
\]
Which clause should we learn?

\begin{align*}
c_1 & : P(z) \lor Q \\
c_2 & : P(y) \lor \neg Q \\
c_3 & : \neg P(a) \lor Q \\
c_4 & : \neg P(b) \lor \neg Q
\end{align*}
First-Order Conflict-Driven Clause Learning

\[
\begin{align*}
[\ell_1]^i & \quad \vdash (\sigma_1^1, \ldots, \sigma_{m_1}^1) \\
\vdots & \\
[\ell_n]^i & \quad \vdash (\sigma_1^n, \ldots, \sigma_{m_n}^n) \\
\hline
(\ell_1 \sigma_1^1 \lor \ldots \lor \ell_1 \sigma_{m_1}^1) \lor \ldots \lor (\ell_n \sigma_1^n \lor \ldots \lor \ell_n \sigma_{m_n}^n) & \quad \text{cl}^i
\end{align*}
\]
Refutational Completeness
(by simulation of the resolution calculus)

\[
\frac{\ell_1 \lor \ldots \lor \ell_n \lor \ell}{(\ell_1 \lor \ldots \lor \ell_n \lor \ell_1' \lor \ldots \lor \ell_m') \sigma} \quad r(\sigma)
\]
Refutational Completeness
(by simulation of the resolution calculus)

\[
\vdots \varphi' \\
\frac{\ell \lor \ell' \lor \ell_1 \lor \ldots \lor \ell_m}{(\ell \lor \ell_1 \lor \ldots \lor \ell_m) \sigma} f(\sigma)
\]

\[
\psi : \overline{[\ell \sigma]}^1 \psi \overline{[\ell_1]}^2 \ldots \overline{[\ell_{m-1}]}^{m-1} \ell \lor \ell' \lor \ell_1 \lor \ldots \lor \ell_m \\
\frac{\ell_m \sigma}{u(\sigma) \overline{[\ell_m]}^{m+1}} \frac{\perp}{(\ell \lor \ell_1 \lor \ldots \lor \ell_m) \sigma} c(\sigma)
\]

The simulation is linear
Soundness
(via simulation by natural deduction)

Step 1:
ground the conflict resolution proof
(expand DAG to tree when necessary)

Step 2:
simulate each unit propagating resolution or conflict
by a chain of implication eliminations.
simulate each conflict driven clause learning inference
by a chain of negation/implication introductions.

Conflict Resolution = “Chained” Natural Deduction with Unification
A Side-Remark: Linear Simulation of Splitting

Now we could even split when

$$\text{var}(\Gamma) \cap \text{var}(\Delta) \neq \emptyset$$
Conflict Resolution

a First-Order Resolution Calculus with Decision Literals and Conflict-Driven Clause Learning

John Slaney · Bruno Woltzenlogel Paleo
A Theorem Prover is much more than a Logical Calculus

- Implementation Techniques
- Heuristics
- Search Algorithm
- Refinements
- Logical Calculus
Pandora’s Box

4 "evils" that attack first-order logic but not propositional logic
1: Non-Termination of First-Order Unit Propagation

\[ c_1 : P \lor Q \]
\[ c_2 : P \lor \neg Q \]
\[ c_3 : \neg P \lor Q \]
\[ c_4 : \neg P \lor \neg Q \]
\[ c_5 : P(a) \]
\[ c_6 : \neg P(x) \lor P(f(x)) \]

\[ C_5 \rightarrow P(a) \rightarrow P(f(a)) \rightarrow P(f(f(a)))) \rightarrow \ldots \]

Note:
this problem will not occur in some decidable fragments (e.g. Bernays-Schönfinkel)
Solutions

1) Ignore the non-termination.

2) Bound the propagation…
   A) … by the depth of the propagation
   B) … by the depth of terms occurring in propagated literals

   and make decisions when the bound is reached, and then increase the bound.
2: Absence of Uniformly True Literals in Satisfied Clauses

\{p(X) \lor q(X), \neg p(a), p(b), q(a), \neg q(b)\}

is a satisfiable clause set

but there is no model where

\(p(X)\) is uniformly true

or

\(q(X)\) is uniformly true

This makes it harder to detect when all clauses are already satisfied (and, therefore, that we can stop the search)
Solutions

1) Ignore the problem, and accept that some satisfiable problems will not be solved. (not so bad, if we focus on unsatisfiable problems)

2) Keep track of “useless decisions” and consider a clause to be satisfied when all its literals are useless decisions.

\{p(X) \lor q(X), \neg p(a), p(b), q(a), \neg q(b)\}

\{p(X), q(X)\} are useless decisions

they lead to subsumed conflict-driven learned clauses
3: Propagation without Satisfaction

In a model containing \( \neg p(a) \)

The clause \( p(X) \lor q(X) \) becomes propagating

and propagates \( q(a) \) into the model

but having \( q(a) \) in the model does not make the clause satisfied

Even after propagation a clause may be needed for other propagations
Solution

1) Check whether the propagating clause became *uniformly satisfied*.

If so, then it won’t be needed in future propagations.
4: Quasi-Falsification without Propagation

In a model containing $\neg p(a)$ and $\neg q(b)$

the clause

$$p(X) \lor q(X) \lor r(X)$$

is quasi-falsified
(because its first two literals are false)

but $r(X)$ cannot be propagated

Moreover, detection of false literals needs to take unification into account

This prevents direct lifting of two watched literals data structure
Solution

For each literal $L$ occurring in a clause, keep a hashset of literals in the model that are duals of instances of $L$.

If all literals of a clause except one have a non-empty hashset associated with it, the clause is quasi-falsified.

This allows quicker detection of quasi-falsified clauses in a manner that resembles two-watched literals.

The set of quasi-falsified clauses is an over-approximation of the set of clauses that can propagate.
Implementation
The Scavenger 0.1 Theorem Prover

Implemented in Scala

by me and two Google Summer of Code students: Daniyar Itegulov and Ezequiel Postan

Open-Source: http://gitlab.com/aossie/Scavenger

GSoC stipends available this year again!

www.aossie.org

Deadline: 3 April
Basic Data Structures

terms and formulas are simply-typed lambda expressions

future work:
extend Conflict Resolution and Scavenger
to higher-order logic

clauses are immutable sequents
(antecedent and succedent are sets)
Proofs are DAGs of Proof Nodes

```scala
abstract class CRProofNode extends ProofNode[Clause, CRProofNode] {
  def findDecisions(sub: Substitution): Clause = {
    this match {
      case Decision(literal) => !sub(literal)
      case conflict @ Conflict(left, right) =>
        left.findDecisions(conflict.leftMgu) union right.findDecisions(conflict.rightMgu)
      case UnitPropagationResolution(left, right, _, leftMgus, _) =>
        // We don't need to traverse right premise, because it's either initial clause or conflict driven clause
        left .zip(leftMgus) .map {
          case (node, mgu) => node.findDecisions(mgu(sub))
        } .fold(Clause.empty)(_ union _)
      case _ =>
        Clause.empty
    }
  }
}
```
each inference rule is a small class

```scala
class Axiom(override val conclusion: Clause) extends CRProofNode {
  def auxFormulasMap = Map()
  def premises = Seq()
}

case class Decision(literal: Literal) extends CRProofNode {
  override def conclusion: Clause = literal.toClause
  override def premises: Seq[CRProofNode] = Seq.empty
}

case class ConflictDrivenClauseLearning(conflict: Conflict) extends CRProofNode {
  val conflictDrivenClause = conflict.findDecisions(Substitution.empty)
  override def conclusion: Clause = conflictDrivenClause
  override def premises: Seq[CRProofNode] = Seq(conflict)
}
```
each inference rule is a small class

case class UnitPropagationResolution private (left: Seq[CRProofNode], right: CRProofNode, desired: Literal, leftMgu: Seq[Substitution], rightMgu: Substitution) extends CRProofNode {
  require(left.forall(_.conclusion.width == 1), "All left conclusions should be unit")
  require(left.size + 1 == right.conclusion.width, "There should be enough left premises to derive desired")

  override def conclusion: Clause = desired

  override def premises: Seq[CRProofNode] = left ++ right
}

case class Conflict(leftPremise: CRProofNode, rightPremise: CRProofNode) extends CRProofNode {
  require(leftPremise.conclusion.width == 1, "Left premise should be a unit clause")
  require(rightPremise.conclusion.width == 1, "Right premise should be a unit clause")

  private val leftAux = leftPremise.conclusion.literals.head.unit
  private val rightAux = rightPremise.conclusion.literals.head.unit

  val (Seq(leftMgu), rightMgu) = unifyWithRename(Seq(leftAux), Seq(rightAux)) match {
    case None => throw new Exception("Conflict: given premise clauses are not resolvable")
    case Some(u) => u
  }

  override def premises = Seq(leftPremise, rightPremise)
  override def conclusion: Clause = Clause.empty
}
Main Search Loop: 3 variants

1. EP-Scavenger: ignore non-termination of unit-propagation (168 lines)

2. PD-Scavenger: bound propagation by propagation depth (342 lines)

3. TD-Scavenger: bound propagation by term depth (176 lines)
Important Missing Features
(Urgent Future Work)

proper backtracking:
Scavenger currently restarts and throws the model away after every conflict

decision literal selection heuristics:
Scavenger currently selects the first literal from a randomised queue
Preliminary Experiments
TPTP Unsat EPR CNF problems without Equality
TPTP Unsat CNF problems without Equality
What about AI/ML?

CDCL and CR ↔ Reinforcement Learning

current model ↔ state

selection of decision literals ↔ actions

learned clause ↔ punishment for (set of) bad decisions

heuristics selecting decision literals with highest scores ↔ policy selecting actions with highest values
Conclusions
**modus ponens**

\[
\begin{array}{c}
A \\
A \\
\hline
\end{array}
\]

\[
\frac{A \rightarrow B}{B}
\]

**resolution**

\[
\begin{array}{c}
\Gamma_1 \rightarrow \Delta_1, A \quad A', \Gamma_2 \rightarrow \Delta_2 \\
\hline
(\Gamma_1, \Delta_1 \rightarrow \Gamma_2, \Delta_2)\sigma
\end{array}
\]

**unit-resulting resolution**

\[
\begin{array}{c}
l_1 \quad \ldots \quad l_n \quad l_1 \rightarrow \ldots \rightarrow l_n \rightarrow l \\
\hline
l\sigma
\end{array}
\]

\[
u(\sigma)
\]

**hypothetical reasoning**

\[
\begin{array}{c}
[A] \\
\vdots \\
\vdots \\
B \\
\hline
A \rightarrow B
\end{array}
\]

**first-order CDCL**

\[
\begin{array}{c}
[l_1]^i \\
\vdots \quad (\sigma_1^1, \ldots, \sigma_{m_1}^1) \\
\vdots \\
[l_n]^i \\
\vdots \quad (\sigma_1^n, \ldots, \sigma_{m_n}^n) \\
\hline
\ell_1\sigma_1^1, \ldots, \ell_1\sigma_{m_1}^1, \ldots, \ell_n\sigma_1^n, \ldots, \ell_n\sigma_{m_n}^n \rightarrow \bot
\end{array}
\]

\[
\text{cl}^i
\]
Performance

Approaches

Resolution/Superposition Provers after decades of engineering

Scavenger after 6 months of engineering

CR Provers after years of engineering

Hopefully

Immediate Future Work:
More careful backtracking and restarting
Thank you!