Combining solvers with interactive theorem provers

Leopoldo Teixeira
lmt@cin.ufpe.br

based on works by Tjark Weber, Sascha Böhme, Clément Hurlin et al.
Higher-order logic theorem proving

- Supports a variety of specification styles: axiomatic, declarative, functional, executable, etc.
- Analyzing systems/programs thoroughly
- Finding design and specification errors early
- High assurance (mathematical, machine checked proof)
Applications

• Good where cost of failure is highest
• safety-critical systems (NASA)
• security-critical systems (Protocols)
• hardware products (Intel)
• however, very **high cost**
CK: TYPE
[11]: [CK->(AM->[Conf->finite_sets[Asset].finite_set])]

% Axiom over ck evaluation
amRef: AXIOM
  ALL (am1,am2): \( \Rightarrow (am1,am2) \Rightarrow \)
  ALL (K:CK,c:Conf):
    \text{wfProduct}([][K][](am1)(c))
  \Rightarrow \text{wfProduct}([][K][](am2)(c)) \land
    ([][K][](am1)(c)) \Rightarrow ([][K][](am2)(c))

ck,ck1,ck2,ck3: VAR CK

% Definition <CK equivalence>
equivalentCKs(ck1,ck2): bool =
  [[ck1]] = [[ck2]]

% Theorem <CK equivalence properties>
eqCK: THEOREM relations[CK].equivalence?( equivalentCKs )

weakerEqCK(fm,ck1,ck2): bool =
  ALL am:
    ALL c: \( \Rightarrow (fam)(c) \Rightarrow ([][ck1][](am)(c)) = ([][ck2][](am)(c)) \)

SPLrefinement.pvs 30% (109,49) (PVS:ready)

Run time = 0.10 secs.
Real time = 1.48 secs.

U:** "pvs* 97% (488,0) (ILISP :ready)
Manual guidance

THEOREM
empty?(S) IFF card(S) = 0

(""
(skolem!)
(prop)
("1"
(rewrite "emptyset_is_empty?[T]"
(replace -1)
(use "card_emptyset")
("2"
(rewrite "card_def"
(rewrite "Card_bijection"
(skolem!)
(delete -)
(grind)
(typepred "f!1(x!1)"
(propax)))))

THEOREM empty?(S) IFF card(S) = 0

Interactive theorem provers
Expressiveness
Automation
SMT
SAT

Wednesday, March 13, 13
Proof scripts can become big
ITP vs. Solvers

- rich specification logic [undecidable]
- infinite state space
- lots of manual effort
- little to no feedback for ‘unprovable’ theorems

- decidable theories
- state explosion problem
- high degree of automation
- better feedback: counterexamples
Why not combine the best of both worlds: ITP + Solvers (ATP)?
Integrate **zChaff** and **Isabelle/HOL** to prove theorems of propositional logic

*Integrating a SAT Solver with an LCF-style Theorem Prover*

zChaff

- A leading SAT solver (winner of the SAT 2002 and SAT 2004 competitions in several categories)
- Returns a satisfying assignment, or . . .
- . . . a proof of unsatisfiability (since 2003)
Solution overview

Interesting point: zChaff provides feedback to Isabelle, so we can also verify its soundness.

Wednesday, March 13, 13
Running example

\[ \phi \equiv (\neg x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (x_1 \lor x_2) \land (\neg x_2 \lor x_3). \]
Preprocessing

\[ \phi \equiv (\neg x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (x_1 \lor x_2) \land (\neg x_2 \lor x_3). \]

$\vdash \phi = \phi.$

CNF conversion
Normalization
Removal of duplicate literals
Removal of tautological clauses

Output:
theorem in the form $\varphi = \varphi^*$
Proof of unsatisfiability

\[
\begin{align*}
\text{CL: } & 4 \leq 2 0 \\
\text{VAR: } & 2 \ L: \ 0 \ V: \ 1 \ A: \ 4 \ \text{Lits: } \ 4 \\
\text{VAR: } & 3 \ L: \ 1 \ V: \ 0 \ A: \ 1 \ \text{Lits: } \ 5 \ 7 \\
\text{CONF: } & 3 \ == \ 5 \ 6
\end{align*}
\]

zChaff proof trace
Derived clauses

\[ \phi \equiv (\neg x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (x_1 \lor x_2) \land (\neg x_2 \lor x_3). \]

\[ \text{CL: } 4 \leq 2 \leq 0 \rightarrow x_2 \]
Variable Assignments

\( \phi \equiv (\neg x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (x_1 \lor x_2) \land (\neg x_2 \lor x_3). \)

**clause id** \( \leq \) **resolvents**

| CL: | 4 | <= | 2 | 0 | \( x_2 \) |

**variable id**  **level**  **value**  **antecedent**

VAR: 2  L: 0  V: 1  A: 4  Lits: 4  \( \rightarrow x_2 = true \)

VAR: 3  L: 1  V: 0  A: 1  Lits: 5 7  \( \rightarrow x_3 = false \)
Conflict Clause

\[ \phi \equiv (\neg x_1 \lor x_2) \land (\neg x_2 \lor \neg x_3) \land (x_1 \lor x_2) \land (\neg x_2 \lor x_3). \]

<table>
<thead>
<tr>
<th>clause id</th>
<th>resolvents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL: 4</td>
<td>&lt;= 2 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>variable id</th>
<th>level</th>
<th>value</th>
<th>antecedent</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR: 2</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Lits: 4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ x_2 = \text{true} \]

<table>
<thead>
<tr>
<th>variable id</th>
<th>level</th>
<th>value</th>
<th>antecedent</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR: 3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Lits: 5 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ x_3 = \text{false} \]

<table>
<thead>
<tr>
<th>conflict clause id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONF: 3 == 5 6</td>
</tr>
</tbody>
</table>

\[ \text{conflict clause} \]
Proof Reconstruction

resolution: list Thm -> Thm

\[ \phi \Rightarrow X \lor Y \lor Z, \phi \Rightarrow W \lor \neg Y \lor Z \]

\[ \phi \Rightarrow X \lor W \lor Z \]
Proof reconstruction

1. Call `prove_clause` for conflict clause

2. Call `prove_literal` for every literal in the conflict clause, to show that literal must be false

3. Finally resolving the conflict clause with these negated literals yields the theorem $\varphi^* \rightarrow False$.

\[
[\phi \Rightarrow X, \phi \Rightarrow \neg X] \rightarrow \phi \Rightarrow false
\]
A set of clauses is unsatisfiable iff the empty clause is derivable via resolution.
Improving proof reconstruction

- Many clauses may be redundant.
- Clauses and literals may be needed many times.
- Two arrays store:
  each clause’s resolvents or its proof
  each variable’s antecedent or its proof
  ... and are updated during proof reconstruction.

1. Initialize arrays with information from the trace.
2. Prove conflict clause C.
3. Perform resolution with prove_literal for each literal in C.
Evaluation

• Compare integration with built-in procedures \((auto, blast, fast)\)

• 42 propositional problems in TPTP v2.6.0

• Problems were negated, so that unsatisfiable problems became provable

• 19 are solved in less than 1 sec
<table>
<thead>
<tr>
<th>Problem</th>
<th>Status</th>
<th>auto</th>
<th>blast</th>
<th>fast</th>
<th>zChaff</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC007-1.008</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>726.5</td>
</tr>
<tr>
<td>NUM285-1</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.2</td>
</tr>
<tr>
<td>PUZ013-1</td>
<td>unsat.</td>
<td></td>
<td></td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>PUZ014-1</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>6.1</td>
<td>0.1</td>
</tr>
<tr>
<td>PUZ015-2.006</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>10.5</td>
</tr>
<tr>
<td>PUZ016-2.004</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.3</td>
</tr>
<tr>
<td>PUZ016-2.005</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>1.6</td>
</tr>
<tr>
<td>PUZ030-2</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.7</td>
</tr>
<tr>
<td>PUZ033-1</td>
<td>unsat.</td>
<td></td>
<td>6.4</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN001-1.005</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>SYN003-1.006</td>
<td>unsat.</td>
<td></td>
<td>1.6</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN004-1.007</td>
<td>unsat.</td>
<td></td>
<td>822.2</td>
<td>2.8</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN010-1.005.005</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>SYN086-1.003</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN087-1.003</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN090-1.008</td>
<td>unsat.</td>
<td></td>
<td>13.8</td>
<td>x</td>
<td>0.5</td>
</tr>
<tr>
<td>SYN091-1.003</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN092-1.003</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN093-1.002</td>
<td>unsat.</td>
<td></td>
<td>1290.8</td>
<td>16.2</td>
<td>1126.6</td>
</tr>
<tr>
<td>SYN094-1.005</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.8</td>
</tr>
<tr>
<td>SYN097-1.002</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>19.2</td>
<td>0.2</td>
</tr>
<tr>
<td>SYN098-1.002</td>
<td>unsat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>SYN302-1.003</td>
<td>sat.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>Problem</td>
<td>Status</td>
<td>auto</td>
<td>blast</td>
<td>fast</td>
<td>zChaff</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>------</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>MSC007-1.008</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>726.5</td>
</tr>
<tr>
<td>NUM285-1</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.2</td>
</tr>
<tr>
<td>PUZ013-1</td>
<td>unsat.</td>
<td>0.5</td>
<td>x</td>
<td>5.0</td>
<td>0.1</td>
</tr>
<tr>
<td>PUZ014-1</td>
<td>unsat.</td>
<td>1.4</td>
<td>x</td>
<td>6.1</td>
<td>0.1</td>
</tr>
<tr>
<td>PUZ015-2.006</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>10.5</td>
</tr>
<tr>
<td>PUZ016-2.004</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.3</td>
</tr>
<tr>
<td>PUZ016-2.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1.6</td>
</tr>
<tr>
<td>PUZ030-2</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.7</td>
</tr>
<tr>
<td>PUZ033-1</td>
<td>unsat.</td>
<td>0.2</td>
<td>6.4</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN001-1.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>SYN003-1.006</td>
<td>unsat.</td>
<td>0.9</td>
<td>x</td>
<td>1.6</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN004-1.007</td>
<td>unsat.</td>
<td>0.3</td>
<td>822.2</td>
<td>2.8</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN010-1.005.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>SYN086-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN087-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN090-1.008</td>
<td>unsat.</td>
<td>13.8</td>
<td>x</td>
<td>x</td>
<td>0.5</td>
</tr>
<tr>
<td>SYN091-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN092-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN093-1.002</td>
<td>unsat.</td>
<td>1290.8</td>
<td>16.2</td>
<td>1126.6</td>
<td>0.1</td>
</tr>
<tr>
<td>SYN094-1.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.8</td>
</tr>
<tr>
<td>SYN097-1.002</td>
<td>unsat.</td>
<td>x</td>
<td>19.2</td>
<td>x</td>
<td>0.2</td>
</tr>
<tr>
<td>SYN098-1.002</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
<tr>
<td>SYN302-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0.4</td>
</tr>
</tbody>
</table>
We can apply the same concept to do more interesting things
Proving Theorems of Higher-Order Logic with SMT Solvers.
Theorem example (i)

\[
\begin{align*}
x_3 &= |x_2| - x_1 \land x_4 &= |x_3| - x_2 \land x_5 &= |x_4| - x_3 \land x_6 &= |x_5| - x_4 \land \\
x_7 &= |x_6| - x_5 \land x_8 &= |x_7| - x_6 \land x_9 &= |x_8| - x_7 \land x_{10} &= |x_9| - x_8 \land \\
x_{11} &= |x_{10}| - x_9 \rightarrow x_1 = x_{10} \land x_2 = x_{11}
\end{align*}
\]

SMT solver almost instantaneously finds a proof, and proof reconstruction for Z3 takes only a few seconds. In contrast, Isabelle’s arithmetic decision procedure requires several minutes to prove the same result.
Theorem example (ii)

$$\forall f. \text{map } f \text{ Nil} = \text{Nil}$$

$$\forall f, x, xs. \text{map } f (\text{Cons } x \ xs) = \text{Cons } (f \ x) \ (\text{map } f \ xs)$$

$$\text{map } (\lambda x^{\text{int}}. x + 2) (\text{Cons } y \ (\text{Cons } 3 \ \text{Nil})) = \text{Cons } (y + 2) \ (\text{Cons } 5 \ \text{Nil})$$

SMT solver is able to prove this higher-order theorem instantaneously
9 Isabelle theories with altogether 1591 proof goals

SMT solvers prove > 50% of all goals

SMT solvers find 8.8% unique proofs,
i.e., proofs for goals on which all ATPs fail.

Hence, SMT solvers increase proof automation.
Non-trivial goals require additional arguments.

Users tend to query existing proof automation before attempting to find the proof themselves.

SMT solvers add ~8.9% unique proofs again, i.e., proofs for goals on which all ATPs fail.

Z3 alone contributes to 9.7% unique proofs.
Benefits of decision procedures

- SMT solvers perform really well over theories containing lambda abstractions.
- Arithmetic support of SMT solvers is in general not essential, but certain goals can benefit.
- Direct support for datatypes, records and type definition by Z3 gives no clear benefits.
- Extra-logical information that is automatically inferred by our current simple algorithm slightly deteriorates the success rates of SMT solvers.
Proving problems of fixed-size bitvectors works well unless functions that are unsupported by SMT solvers are used.

- « and » shifting by a natural number.
- $\text{ucast } x$ extend $x$ by padding zeros to its front.
- $\text{mask } n$ is defined as $(1 \ll n) - 1$ for natural numbers $n$.
- $x_n$ select bit $n$ from a bitvector $x$ and interpret it as Boolean
  - $\&, |$ and $!$ bitwise conjunction, disjunction and negation.
- $u$ is a bitvector of size 12
- $v$ and $w$ are bitvectors of size 32
- $x$ and $y$ are bitvectors that have arbitrary but equal size.

\[
\begin{align*}
  u \neq 0 & \rightarrow (1 \ll 20) - 1 < (\text{ucast } u \ll 20)_{32 \text{word}} \\
  (\text{if } w = 0 \text{ then } v \leq 0 \text{ else } v \leq w - 1) & \rightarrow v \neq -1 \\
  n < 32 & \rightarrow 1_{32 \text{word}} \leq (1 \ll n) \\
  v \& \text{mask } 14 = 0 & \rightarrow v + (w \gg 20 \ll 2) \& (! (\text{mask } 14)) = v \\
  v \& \text{mask } n = 0 & \land (\forall n'. n \leq n' \land n' < 32 \rightarrow \neg w_{n'}) \rightarrow v + w \& (! (\text{mask } n)) = v \\
  0 < y \land y \leq x & \rightarrow (x - y) \div y = (x \div y) - 1 \\
  (x \& y) + (x | y) & = x + y \\
  x \& y = 0 & \rightarrow x + y = x | y
\end{align*}
\]
Yet another example

\[(X \cap Y = \emptyset \land X \setminus Z = X) \Rightarrow X \cap (Y \cup Z) = \emptyset.\]

I have proved this using PVS+Yices :-)

The PVS strategy **grind** (automatic procedure) fails
Conclusions

• Combine the best of both worlds
  • rich specification logics
  • counterexamples for unprovable formulae
  • performance improvements
  • automation
  • plus: solver verification
  • still room for improvements, there is work to be done!