Efficiently Checking Propositional Resolution Proofs in Isabelle/HOL

Tjark Weber
webertj@in.tum.de

6th International Workshop on the Implementation of Logics
November 12, 2006
1 Introduction

2 System Description

3 Evaluation

4 Conclusions, Future Work
Isabelle is a **generic** theorem prover.
Isabelle/HOL

- Isabelle is a generic theorem prover.
- Isabelle/HOL provides a rich specification language.
Isabelle/HOL

- Isabelle is a **generic** theorem prover.
- Isabelle/HOL provides a **rich specification language**.
- Isabelle/HOL offers a reasonable degree of **automation**.
Isabelle is a generic theorem prover.
Isabelle/HOL provides a rich specification language.
Isabelle/HOL offers a reasonable degree of automation.
Isabelle/HOL is used for hardware and software verification.
Verification problems can often be reduced to Boolean satisfiability.
Motivation

- Verification problems can often be reduced to Boolean satisfiability.
- Recent SAT solver advances have made this approach feasible in practice.
Motivation

- Verification problems can often be reduced to Boolean satisfiability.
- Recent SAT solver advances have made this approach feasible in practice.

Can an **LCF-style** theorem prover benefit from these advances?
Motivation

- Verification problems can often be reduced to Boolean satisfiability.
- Recent SAT solver advances have made this approach feasible in practice.

Can an **LCF-style** theorem prover benefit from these advances?

Can we increase the degree of **automation** in Isabelle/HOL while keeping the **trusted code base** small?
The Oracle Approach

A formula is *accepted* as a theorem if the external tool claims it to be provable.
The Oracle Approach

A formula is *accepted* as a theorem if the external tool claims it to be provable.

The LCF-style Approach

The external tool provides a *certificate* of its answer that is translated into an Isabelle proof.
System Overview

Isabelle

Input formula → Preprocessing

Counterexample

Theorem → Proof reconstruction

DIMACS CNF

SAT Solver

satisfiable?

yes no

Model

Proof Trace
Preprocessing

- The input formula is negated.
Preprocessing

- The input formula is negated.
- The negated input formula is transformed into CNF.
  - Naive CNF transformation
  - Definitional CNF
The input formula is negated.

The negated input formula is transformed into CNF.

- Naive CNF transformation
- Definitional CNF

The CNF transformation must be proof-producing. The result is not just a CNF formula $\phi^*$, but a theorem $\vdash \phi = \phi^*$. 
SAT Solvers

zChaff, MiniSat
SAT Solvers

zChaff, MiniSat

- leading SAT solvers (winner of recent SAT competitions in several categories)
SAT Solvers

zChaff, MiniSat

- leading SAT solvers (winner of recent SAT competitions in several categories)
- zChaff: developed by Sharad Malik and Zhaohui Fu, Princeton University
SAT Solvers

**zChaff, MiniSat**

- leading SAT solvers (winner of recent SAT competitions in several categories)
- zChaff: developed by Sharad Malik and Zhaohui Fu, Princeton University
- MiniSat: developed by Niklas Eén and Niklas Sörensson, Chalmers University
SAT Solvers

zChaff, MiniSat

- leading SAT solvers (winner of recent SAT competitions in several categories)
- zChaff: developed by Sharad Malik and Zhaohui Fu, Princeton University
- MiniSat: developed by Niklas Eén and Niklas Sörensson, Chalmers University
- return a satisfying assignment, or . . .
SAT Solvers

zChaff, MiniSat

- leading SAT solvers (winner of recent SAT competitions in several categories)
- zChaff: developed by Sharad Malik and Zhaohui Fu, Princeton University
- MiniSat: developed by Niklas Eén and Niklas Sörensson, Chalmers University
- return a satisfying assignment, or . . .
- . . . a proof of unsatisfiability (since 2003 (zChaff)/2006 (MiniSat))
Proof Formats

The proofs generated by zChaff and MiniSat differ in detail, but both are based on the **propositional resolution** rule.
The proofs generated by zChaff and MiniSat differ in detail, but both are based on the **propositional resolution** rule.

### Propositional Resolution

\[ P \lor x \quad Q \lor \lnot x \]

\[ \frac{\quad}{P \lor Q} \]
The proofs generated by zChaff and MiniSat differ in detail, but both are based on the **propositional resolution** rule.

**Propositional Resolution**

\[
P \lor x \quad Q \lor \neg x \\
\hline
P \lor Q
\]

**Proofs: Internal Representation**

```plaintext
type proof = int list Inttab.table * int

- “Clause \( n \) is the result of resolving clauses \( n_1, \ldots, n_k \).”
- “Clause \( m \) is the empty clause.”
```
# Isabelle's Previous Automation (on TPTP)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Status</th>
<th>auto</th>
<th>blast</th>
<th>fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC007-1.008</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NUM285-1</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PUZ013-1</td>
<td>unsat.</td>
<td>0.5</td>
<td>x</td>
<td>5.0</td>
</tr>
<tr>
<td>PUZ014-1</td>
<td>unsat.</td>
<td>1.4</td>
<td>x</td>
<td>6.1</td>
</tr>
<tr>
<td>PUZ015-2.006</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PUZ016-2.004</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PUZ016-2.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PUZ030-2</td>
<td>unsat.</td>
<td>0.2</td>
<td>6.4</td>
<td>0.1</td>
</tr>
<tr>
<td>PUZ033-1</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN001-1.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN003-1.006</td>
<td>unsat.</td>
<td>0.9</td>
<td>x</td>
<td>1.6</td>
</tr>
<tr>
<td>SYN004-1.007</td>
<td>unsat.</td>
<td>0.3</td>
<td>822.2</td>
<td>2.8</td>
</tr>
<tr>
<td>SYN010-1.005.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN086-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN087-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN090-1.008</td>
<td>unsat.</td>
<td>13.8</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN091-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN092-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN093-1.002</td>
<td>unsat.</td>
<td>1290.8</td>
<td>16.2</td>
<td>1126.6</td>
</tr>
<tr>
<td>SYN094-1.005</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN097-1.002</td>
<td>unsat.</td>
<td>x</td>
<td>19.2</td>
<td>x</td>
</tr>
<tr>
<td>SYN098-1.002</td>
<td>unsat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SYN302-1.003</td>
<td>sat.</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
A Naive Approach (Weber, 2005)

Start from $\vdash (\phi \Rightarrow False) \Rightarrow (\phi \Rightarrow False)$. 
A Naive Approach (Weber, 2005)

- Start from \( \vdash (\phi \Rightarrow \text{False}) \Rightarrow (\phi \Rightarrow \text{False}) \).
- Reduce the premise \( \phi \Rightarrow \text{False} \) to \( \text{True} \).
<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>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>
A Naive Approach (Weber, 2005)

- Start from $\vdash (\phi \Rightarrow False) \Rightarrow (\phi \Rightarrow False)$.
- Reduce the premise $\phi \Rightarrow False$ to $True$.
- A huge improvement over Isabelle’s previous automation . . .
A Naive Approach (Weber, 2005)

- Start from $\vdash (\phi \Rightarrow False) \Rightarrow (\phi \Rightarrow False)$.
- Reduce the premise $\phi \Rightarrow False$ to True.
- A huge improvement over Isabelle’s previous automation . . .
- . . . but still not satisfactory:

<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>
</tbody>
</table>
A Naive Approach (Weber, 2005)

- Start from $\vdash (\phi \Rightarrow False) \Rightarrow (\phi \Rightarrow False)$.
- Reduce the premise $\phi \Rightarrow False$ to True.

- A huge improvement over Isabelle’s previous automation . . .
- . . . but still not satisfactory:

<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>
</tbody>
</table>

- Explicit treatment of associativity and commutativity for $\lor$, $\land$ required.
How to check propositional resolution proofs in Isabelle/HOL efficiently?
The Main Question

How to check propositional resolution proofs in Isabelle/HOL efficiently?

Theorems in Isabelle/HOL

A theorem is a sequent $\Gamma \vdash \phi$, where $\Gamma$ is a finite set of hypotheses.
The Main Question

How to check propositional resolution proofs in Isabelle/HOL efficiently?

Theorems in Isabelle/HOL

A theorem is a sequent $\Gamma \vdash \phi$, where $\Gamma$ is a finite set of hypotheses.

Assume $\{\phi\} \vdash \phi$

$\Gamma \vdash \psi$

$\Gamma \setminus \phi \vdash \phi \implies \psi$ (impl)

$\Gamma \vdash \phi \iff \psi$

$\Gamma \cup \Gamma' \vdash \psi$ (impE)

$\Gamma' \vdash \phi$
Each clause $p_1 \lor \ldots \lor p_n$ is encoded as a single theorem

$$\{ p_1 \lor \ldots \lor p_n \} \vdash \neg p_1 \Rightarrow \ldots \Rightarrow \neg p_n \Rightarrow \text{False}$$
Separate Clauses (Alwen Tiu et al., 2006)

Each clause $p_1 \lor \ldots \lor p_n$ is encoded as a single theorem

$$\{p_1 \lor \ldots \lor p_n\} \vdash \overline{p_1} \Rightarrow \ldots \Rightarrow \overline{p_n} \Rightarrow \text{False}$$

Resolution is based on a derived Isabelle tactic which performs cuts.
Each clause $p_1 \lor \ldots \lor p_n$ is encoded as a single theorem

$$\{ p_1 \lor \ldots \lor p_n \} \vdash \overline{p_1} \Rightarrow \ldots \Rightarrow \overline{p_n} \Rightarrow \text{False}$$

Resolution is based on a derived Isabelle tactic which performs cuts.

- Proof reconstruction for MSC007-1.008: 7.8 s
- The problem is a set of clauses.
- Clauses are not viewed as sets of literals.
Sequent Representation

Each clause $p_1 \lor \ldots \lor p_n$ is encoded as a single theorem

$$\{p_1 \lor \ldots \lor p_n, \overline{p_1}, \ldots, \overline{p_n}\} \vdash \text{False}$$
Sequent Representation

Each clause $p_1 \lor \ldots \lor p_n$ is encoded as a single theorem

$$\{p_1 \lor \ldots \lor p_n, \overline{p_1}, \ldots, \overline{p_n}\} \vdash \text{False}$$

Resolution:

1. $\text{impl}: \Gamma_1 := \{p_1 \lor \ldots \lor p_n, \overline{p_1}, \ldots, \overline{p_n}\} \setminus \{p\} \vdash p \Rightarrow \text{False}$
2. $\text{impl}: \Gamma_2 := \{q_1 \lor \ldots \lor q_m, \overline{q_1}, \ldots, \overline{q_m}\} \setminus \{\neg p\} \vdash \neg p \Rightarrow \text{False}$
3. $\text{instantiate}: \vdash (p \Rightarrow \text{False}) \Rightarrow (\neg p \Rightarrow \text{False}) \Rightarrow \text{False}$
4. $\text{impE}: \Gamma_1 \vdash (\neg p \Rightarrow \text{False}) \Rightarrow \text{False}$
5. $\text{impE}: \Gamma_1 \cup \Gamma_2 \vdash \text{False}$
Sequent Representation

Each clause $p_1 \lor \ldots \lor p_n$ is encoded as a single theorem

$$\{p_1 \lor \ldots \lor p_n, \overline{p_1}, \ldots, \overline{p_n}\} \vdash \text{False}$$

- Proof reconstruction for MSC007-1.008: 1.2 s
- The problem is a set of clauses.
- Clauses are sets of literals.
- Clause hypotheses accumulate during resolution, until the set of hypotheses eventually contains every clause used in the proof.
The whole CNF problem is assumed: \( \{\bigwedge_{i=1}^{k} C_i\} \vdash \bigwedge_{i=1}^{k} C_i \).

Each clause is derived: \( \{\bigwedge_{i=1}^{k} C_i\} \vdash C_1, \ldots, \{\bigwedge_{i=1}^{k} C_i\} \vdash C_k \).

Then the (modified) sequent representation is used:
\[
\{\bigwedge_{i=1}^{k} C_i, \overline{p_1}, \ldots, \overline{p_n}\} \vdash False.
\]
The whole CNF problem is assumed: $\{\bigwedge_{i=1}^{k} C_i\} \vdash \bigwedge_{i=1}^{k} C_i$.

Each clause is derived: $\{\bigwedge_{i=1}^{k} C_i\} \vdash C_1, \ldots, \{\bigwedge_{i=1}^{k} C_i\} \vdash C_k$.

Then the (modified) sequent representation is used:
$\{\bigwedge_{i=1}^{k} C_i, \overline{p_1}, \ldots, \overline{p_n}\} \vdash False$.

Proof reconstruction for MSC007-1.008: 0.5 s

The right way to do things.
Further Optimizations

- **Pivot search**: Instead of searching the hypotheses of clauses to determine the pivot literal for resolutions, we associate our own data structure (an ordered tree of integers) with each clause.
Further Optimizations

- **Pivot search**: Instead of searching the hypotheses of clauses to determine the pivot literal for resolutions, we associate our own data structure (an ordered tree of integers) with each clause.

- **Backwards proof**: Instead of chronologically replaying the proof trace, we perform “backwards” proof reconstruction, starting from the empty clause’s identifier.
Further Optimizations

- **Pivot search**: Instead of searching the hypotheses of clauses to determine the pivot literal for resolutions, we associate our own data structure (an ordered tree of integers) with each clause.

- **Backwards proof**: Instead of chronologically replaying the proof trace, we perform “backwards” proof reconstruction, starting from the empty clause’s identifier.

- **Lemmas**: Instead of proving the same intermediate clause multiple times, we store proven clauses in an array and simply retrieve them from there if they are needed again.
Individual SATLIB problems typically contain several ten thousand variables and several hundred thousand clauses.
SATLIB: Pushing Isabelle to its Limits

- **Parser**: very general, but unable to parse very large terms in reasonable time.
SATLIB: Pushing Isabelle to its Limits

- **Parser**: very general, but unable to parse very large terms in reasonable time.
- **Solution**: we implemented our own little parser for DIMACS files, which turns them directly into ML term values.
SATLIB: Pushing Isabelle to its Limits

- **Parser:** very general, but unable to parse very large terms in reasonable time.
- **Solution:** we implemented our own little parser for DIMACS files, which turns them directly into ML term values.
- **User interface:** nice features (e.g. syntax highlighting), but unable to display very large terms in reasonable time.
SATLIB: Pushing Isabelle to its Limits

- **Parser**: very general, but unable to parse very large terms in reasonable time.

- **Solution**: we implemented our own little parser for DIMACS files, which turns them directly into ML term values.

- **User interface**: nice features (e.g. syntax highlighting), but unable to display very large terms in reasonable time.

- **Solution**: we worked at the system’s ML level, thereby avoiding the additional user interface layer.
SATLIB: Pushing Isabelle to its Limits

- **Parser**: very general, but unable to parse very large terms in reasonable time.

- **Solution**: we implemented our own little parser for DIMACS files, which turns them directly into ML term values.

- **User interface**: nice features (e.g. syntax highlighting), but unable to display very large terms in reasonable time.

- **Solution**: we worked at the system’s ML level, thereby avoiding the additional user interface layer.

- **Inference kernel**: minor inefficiencies, which became significant when the kernel had to deal with very large terms.
SATLIB: Pushing Isabelle to its Limits

- **Parser**: very general, but unable to parse very large terms in reasonable time.
  - **Solution**: we implemented our own little parser for DIMACS files, which turns them directly into ML term values.
- **User interface**: nice features (e.g. syntax highlighting), but unable to display very large terms in reasonable time.
  - **Solution**: we worked at the system’s ML level, thereby avoiding the additional user interface layer.
- **Inference kernel**: minor inefficiencies, which became significant when the kernel had to deal with very large terms.
  - **Solution**: small fixes to the kernel.
## Evaluation on SATLIB Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Variables</th>
<th>Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>c7552mul.miter</td>
<td>11282</td>
<td>69529</td>
</tr>
<tr>
<td>6pipe</td>
<td>15800</td>
<td>394739</td>
</tr>
<tr>
<td>6pipe_6_ooo</td>
<td>17064</td>
<td>545612</td>
</tr>
<tr>
<td>7pipe</td>
<td>23910</td>
<td>751118</td>
</tr>
</tbody>
</table>
### Evaluation on SATLIB Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Variables</th>
<th>Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>c7552mul. miter</td>
<td>11282</td>
<td>69529</td>
</tr>
<tr>
<td>6pipe</td>
<td>15800</td>
<td>394739</td>
</tr>
<tr>
<td>6pipe_6_ooo</td>
<td>17064</td>
<td>545612</td>
</tr>
<tr>
<td>7pipe</td>
<td>23910</td>
<td>751118</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>zChaff (s)</th>
<th>Proof (s)</th>
<th>Resolutions</th>
<th>Total (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c7552mul. miter</td>
<td>73</td>
<td>70</td>
<td>252200</td>
<td>145</td>
</tr>
<tr>
<td>6pipe</td>
<td>167</td>
<td>321</td>
<td>268808</td>
<td>512</td>
</tr>
<tr>
<td>6pipe_6_ooo</td>
<td>308</td>
<td>2575</td>
<td>870345</td>
<td>3179</td>
</tr>
<tr>
<td>7pipe</td>
<td>495</td>
<td>1132</td>
<td>357136</td>
<td>1768</td>
</tr>
</tbody>
</table>
Isabelle’s automation for propositional logic has been greatly enhanced.

Efficient proof checking for propositional logic is possible in a general LCF-style system.

Our implementation scales well to proofs with hundreds of thousands of resolution steps.

Our techniques are applicable to other interactive provers, e.g. to HOL 4 and HOL-Light.
Future Work

- Analysis and optimization of resolution proofs
- SAT-based decision procedures beyond propositional logic (e.g. SMT)
- Standard proof formats (for propositional logic and beyond)