Credo quia absurdum (?)

Proof Generation for Saturating First-Order Theorem Provers

Stephan Schulz
Geo Sutcliffe
Agenda

Structure and Representation of Proofs

Proof Generation

Proof Applications

Conclusion
Structure and Representation of Proofs
Refutational Theorem Proving

\[
\{A_1, A_2, \ldots, A_n\} \models C
\]
Refutational Theorem Proving

\[ \{A_1, A_2, \ldots, A_n\} \models C \]

iff

\[ \{A_1, A_2, \ldots, A_n, \neg C\} \text{ is unsatisfiable} \]
Refutational Theorem Proving

\( \{A_1, A_2, \ldots, A_n\} \models C \)

iff

\( \{A_1, A_2, \ldots, A_n, \neg C\} \) is unsatisfiable

iff

\( \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \) is unsatisfiable
Refutational Theorem Proving

\( \{A_1, A_2, \ldots, A_n\} \models C \)

iff

\( \{A_1, A_2, \ldots, A_n, \neg C\} \) is unsatisfiable

iff

\( \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \) is unsatisfiable

iff

\( \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \vdash \Box \)
Refutational Theorem Proving

\[ \{A_1, A_2, \ldots, A_n\} \models C \]

iff

\[ \{A_1, A_2, \ldots, A_n, \neg C\} \text{ is unsatisfiable} \]

iff

\[ \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \text{ is unsatisfiable} \]

iff

\[ \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \vdash \square \]
Ideal: Proofs as Sequences of Proof Steps

A derivation is a list of steps
Each step carries a clause/formula
Each step is either . . .
  ▶ Assumed (e.g. axioms, conjecture)
  ▶ Logically derived from earlier steps
A proof is a derivation that either . . .
  ▶ derives the conjecture
  ▶ derives a contradiction from the negated conjecture

Good mental model!
Reality: Proofs as Sequences of Proof Steps

Initial clauses/formulas
- Axioms/Conjectures/Hypotheses
- Justified by assumption

Derived clauses/formulas
- Justified by reference to (topologically) preceding steps
- Defined logical relationship to predecessors
  - Most frequent case: theorem of predecessors
  - Exceptions: Skolemization, negation of conjecture, ...

(Introduced definitions)
- Don’t affect satisfiability/provability
- Justified by definition
Logical Languages for FOF

Historical

Problems
- Otter “lists”
- LOP (CNF only)
- DFG
- TPTP (v1, v2)
- ...

Proofs/Derivations
- Otter “proof object”
- PCL (UEQ only)
- DFG (but nobody uses DFG)
- ...

First-order proofs

Stephan Schulz
Logical Languages for FOF

Historical

Problems
- Otter “lists”
- LOP (CNF only)
- DFG
- TPTP (v1, v2)
- ...

Proofs/Derivations
- Otter “proof object”
- PCL (UEQ only)
- DFG (but nobody uses DFG)
- ...

Modern Convergence: TPTP v3
TPTP v3 language

Consistent syntax for different classes
  ► CNF is sub-case of FOF
  ► FOF is sub-case of TFF
Applicable for a wide range of applications
  ► Problem specifications
  ► Proofs/derivations
  ► Models
Easily parsable
  ► Prolog-parsable
  ► Lex/Yacc grammar
  ► Recursive-descent with 1-token look-ahead
Widely used and supported
  ► CASC
  ► Major provers (E, SPASS, Vampire, iProver, ...)
  ► Used by integrators
Example

fof(c_0_0, conjecture, (?[X3]:(human(X3)&X3!=john)), file('humen.p', someone_not_john)).
fof(c_0_1, axiom, (?[X3]:(human(X3)&grade(X3)=a)), file('humen.p', someone_got_an_a)).
fof(c_0_2, axiom, (grade(john)=f), file('humen.p', john_failed)).
fof(c_0_3, axiom, (a!=f), file('humen.p', distinct_grades)).
fof(c_0_4, negated_conjecture, (¬(?[X3]:(human(X3)&X3!=john))),
inference(assume_negation, [status(cth)], [c_0_0])).
fof(c_0_5, negated_conjecture, (¬[X4]:(¬human(X4)|X4=john)),
inference(variable_rename, [status(thm)], [inference(fof_nnf, [status(thm)], [c_0_4]))]).
fof(c_0_6, plain, ((human(esk1_0)&grade(esk1_0)=a)),
inference(skolemize, [status(esa)], [inference(variable_rename, [status(thm)], [c_0_1]))]).
cnf(c_0_7, negated_conjecture, (X1=john|¬human(X1)),
inference(split_conjunct, [status(thm)], [c_0_5])).
cnf(c_0_8, plain, (human(esk1_0)),
inference(split_conjunct, [status(thm)], [c_0_6])).
cnf(c_0_9, plain, (grade(esk1_0)=a),
inference(split_conjunct, [status(thm)], [c_0_6])).
cnf(c_0_10, negated_conjecture, (esk1_0=john),
inference(spm, [status(thm)], [c_0_7, c_0_8])).
cnf(c_0_11, plain, (grade(john)=f),
inference(split_conjunct, [status(thm)], [c_0_2])).
cnf(c_0_12, plain, (a!=f),
inference(split_conjunct, [status(thm)], [c_0_3])).
cnf(c_0_13, plain, ($false),
inference(sr, [status(thm)], [inference(rw, [status(thm)],
[inference(rw, [status(thm)], [c_0_9, c_0_10]), c_0_11]), c_0_12]), ['proof']).
cnf(c_0_13,
    plain,
    ($false),
inference(sr,[status(thm)],
    [inference(rw,[status(thm)],
        [inference(rw,[status(thm)],
            [c_0_9, c_0_10]),
            c_0_11]),
    c_0_12]),
    ['proof']).
cnf(c_0_13, 
    plain,
    ($false),
    inference(sr,[status(thm)],
              ...
    ...
    c_0_11),
    c_0_12),
    ['proof']).
Language (cnf, fof, tff, …)
cnf(c_0_13, 
  plain, 
  ($false),  
  inference(sr,[status(thm)],  
        [inference(rw,[status(thm)],  
          [inference(rw,[status(thm)],  
            [c_0_9, c_0_10]),  
              c_0_11]),  
                c_0_12]),  
      ['proof']).
cnf(c_0_13, 
    plain, 
    ($false),  
    ...             
                          
    inference(sr,[status(thm)],  
                   inference(rw,[status(thm)],  
                       inference(rw,[status(thm)],  
                           [c_0_9, c_0_10]),  
                           c_0_11)),  
                           c_0_12)),  
    ['proof'])).
Language (cnf, fof, tff, ...)
Name (arbitrary, but unique)
Type (axiom, lemma, conjecture, ...)

\text{cnf}(c_{0\_13}, \text{plain}, ($false), \text{inference}(sr, [\text{status(thm)}],
\text{inference}(rw, [\text{status(thm)}],
\text{inference}(rw, [\text{status(thm)}],
\text{inference}(rw, [\text{status(thm)}],
[c_{0\_9}, c_{0\_10}]),
c_{0\_11}),
c_{0\_12}]),
['proof']).
Language (cnf, fof, tff, ...) Name (arbitrary, but unique) Type (axiom, lemma, conjecture, ...) Logical formula (the empty clause in this case)

cnf(c_0_13,
plain,$false),
inference(sr,[status(thm)],
    [inference(rw,[status(thm)],
        [inference(rw,[status(thm)],
            [c_0_9, c_0_10]),
            c_0_11]),
        c_0_12]),
    ['proof'])).
Language (cnf, fof, tff, …)
Name (arbitrary, but unique)
Type (axiom, lemma, conjecture, …)
Logical formula (the empty clause in this case)
Source (derivation from premises)

cnf(c_0_13, plain, ($false),
inference(sr,[status(thm)],
  [inference(rw,[status(thm)],
    [inference(rw,[status(thm)],
      [c_0_9, c_0_10]),
      c_0_11]),
     c_0_12]),
['proof'])
null
cnf(c_0_13,
plain,
($false),
inference(sr,[status(thm)],
    [inference(rw,[status(thm)],
        [inference(rw,[status(thm)],
            [c_0_9, c_0_10]),
            c_0_11]),
        c_0_12]),
[('proof')].
cnf(c_0_13,
   plain,
   ($false),
   inference(sr,[status(thm)],
               [inference(rw,[status(thm)],
                            [inference(rw,[status(thm)],
                                         [c_0_9, c_0_10]),
                                          c_0_11]),
                             c_0_12]),
    ['proof']).
cnf(c_0_13,
   plain,
   ($false),
   inference(sr,[status(thm)],
               [inference(rw,[status(thm)],
                 [inference(rw,[status(thm)],
                   [c_0_9, c_0_10]),
                     c_0_11]),
                     c_0_12]),
   ['proof']).
Inference rule (sr: Simplify-reflect, rw: Rewriting, pm: Paramodulation, ...)

cnf(c_0_13, plain, ($false),
    inference(sr,[status(thm)],
    [inference(rw,[status(thm)],
    [inference(rw,[status(thm)],
    [c_0_9, c_0_10]),
    c_0_11]),
    c_0_12]),
    ['proof']).
First-order proofs

Inference rule (sr: Simplify-reflect, rw: Rewriting, pm: Paramodulation, ...)

“Useful information”: logical status (formula is theorem of premises)

cnf(c_0_13, plain, ($false),
inference(sr,[status(thm)],
        [inference(rw,[status(thm)],
          [inference(rw,[status(thm)],
            [c_0_9, c_0_10]),
            c_0_11]),
            c_0_12]),
        ['proof'])).
Inference rule (sr: Simplify-reflect, rw: Rewriting, pm: Paramodulation, …)

“Useful information”: logical status (formula is theorem of premises)

\[ \text{cnf(c\_0\_13, plain, ($false), } \]
\[ \text{inference(sr,[status(thm)], } \]
\[ \text{[inference(rw,[status(thm)], } \]
\[ \text{[inference(rw,[status(thm)], } \]
\[ \text{[c\_0\_9, c\_0\_10]), } \]
\[ \text{c\_0\_11]), } \]
\[ \text{c\_0\_12]), } \]
\[ \text{['proof']].} \]

Names of the premises
cnf(c_0_13,  
    plain,  
    ($false),  
    inference(sr,[status(thm)],  
              ...             
    [inference(rw,[status(thm)],  
      [inference(rw,[status(thm)],  
        [c_0_9, c_0_10]),  
        c_0_11]),  
      c_0_12]),  
    ['proof'])).
cnf(c_0_13,
    plain,
    ($false),
    inference(sr,[status(thm)],
        inference(rw,[status(thm)],
            inference(rw,[status(thm)],
                c_0_9, c_0_10),
                c_0_11),
                c_0_12),
    ['proof']).

\[
\begin{align*}
    c_0_9: & \quad \text{grade(esk1}_0)=a \\
    c_0_10: & \quad \text{esk1}_0=john \\
    c_0_11: & \quad \text{grade(john)}=f \\
    c_0_12: & \quad a\neq f
\end{align*}
\]
cnf(c_0_13,  
    plain,  
    ($false),  
    inference(sr,[status(thm)],  
        [inference(rw,[status(thm)],  
            [inference(rw,[status(thm)],  
                [inference(rw,[status(thm)],  
                    [c_0_9, c_0_10],  
                    c_0_11),  
                    c_0_12)],  
                ['proof'])).  

Innermost inference: Rewrite c_0_9 with c_0_10

\(c_0_9: \text{grade}(esk1_0)=a\)
\(c_0_10: esk1_0=\text{john}\)
\(c_0_11: \text{grade}(\text{john})=f\)
\(c_0_12: a!=f\)
cnf(c_0_13,
plain,
($false),
inference(sr,[status(thm)],
  [inference(rw,[status(thm)],
    [inference(rw,[status(thm)],
      [c_0_9, c_0_10]),
      c_0_11]),
    c_0_12]),
['proof'])).

\begin{align*}
c_0_9 &: \text{grade}(esk1_0)=a \\
c_0_10 &: \text{esk1}_0=\text{john} \\
c_0_11 &: \text{grade}(\text{john})=f \\
c_0_12 &: a\neq f
\end{align*}
Innermost inference: Rewrite \( c_0_9 \) with \( c_0_10 \)

Intermediate inference: Rewrite the result of the innermost inference with \( c_0_11 \)

Outermost (final) inference: Cut off a literal from the result of the intermediate inference with \( c_0_12 \)

\[
\text{cnf}(c_{0,13}, \text{plain}, ($false), \text{inference}(sr,[\text{status(thm)}], \text{inference}(rw,[\text{status(thm)}], \text{inference}(rw,[\text{status(thm)}], \text{inference}(rw,[\text{status(thm)}], [c_{0,9}, c_{0,10}], c_{0,11}], c_{0,12})), ['proof'])).
\]

- \( c_{0,9}: \) grade(esk1_0)=a
- \( c_{0,10}: \) esk1_0=john
- \( c_{0,11}: \) grade(john)=f
- \( c_{0,12}: \) a!=f
Compl[i]mentary Example

```prolog
fof(c_0_1, 
    axiom, 
    (?[X3]: (human(X3) & grade(X3)=a)),
    file('humen.p', someone_got_an_a)).
```
TPTP v3 idiosyncrasies

No inference semantics
  ▶ Rules are just names
  ▶ Rules are system-dependent

Incomplete inference description
  ▶ “Rules are just names”
  ▶ Syntactic support not widely supported
TPTP v3 idiosyncrasies

No inference semantics
- Rules are just names
- Rules are system-dependent

Incomplete inference description
- “Rules are just names”
- Syntactic support not widely supported

Workarounds:
- Inference status
- Proof reconstruction
Proof Generation
Refutational Theorem Proving

\[ \{A_1, A_2, \ldots, A_n\} \models C \]

iff

\[ \{A_1, A_2, \ldots, A_n, \neg C\} \text{ is unsatisfiable} \]

iff

\[ \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \text{ is unsatisfiable} \]

iff

\[ \text{cnf}(\{A_1, A_2, A_n, \neg C\}) \vdash \square \]
Clasusification and Saturation

Clasusification

- Terminating
- (Usually) deterministic
- (Usually) non-destructive
- Sometimes done by external tool

Saturation

- Many degrees of freedom
- Arbitrary search time
- Generating inferences
  - Create new clauses
  - Necessary for completeness
- Simplifying inferences
  - Modify/remove existing clauses
  - Necessary for performance
Clausification and Saturation

Clausification
- Terminating
- (Usually) deterministic
- (Usually) non-destructive
- Sometimes done by external tool

Saturation
- Many degrees of freedom
- Arbitrary search time
- Generating inferences
  - Create new clauses
  - Necessary for completeness
- Simplifying inferences
  - Modify/remove existing clauses
  - Necessary for performance

Recording clausification is straightforward
- ...but not always done

Efficiently recording saturation is difficult
- ...some settle for inefficient
Deduction vs. Simplification

Superposition

\[
\frac{s \simeq t \lor S \quad u \not\equiv v \lor R}{\sigma(u[p \leftarrow t] \not\equiv v \lor S \lor R)}
\]

if \( \sigma = \text{mgu}(u|_p, s) \), [...]

Rewriting

\[
\frac{s \simeq t \quad u \not\equiv v \lor R}{s \simeq t \quad u[p \leftarrow \sigma(t)] \not\equiv v \lor R}
\]

if \( u|_p = \sigma(s) \) and \( \sigma(s) > \sigma(t) \)
The Given-Clause Algorithm

- **Aim:** Move everything from $U$ to $P$

$P$
(processed clauses)

$U$
(unprocessed clauses)

$g$

**Invariant:** All generating inferences with premises from $P$ have been performed

**Invariant:** $P$ is interreduced

Clauses added to $U$ are simplified with respect to $P$
The Given-Clause Algorithm

- **Aim:** Move everything from \( U \) to \( P \)
- **Invariant:** All generating inferences with premises from \( P \) have been performed
The Given-Clause Algorithm

- **Aim**: Move everything from $U$ to $P$
- **Invariant**: All generating inferences with premises from $P$ have been performed
- **Invariant**: $P$ is interreduced
The Given-Clause Algorithm

- **Aim:** Move everything from $U$ to $P$
- **Invariant:** All generating inferences with premises from $P$ have been performed
- **Invariant:** $P$ is interreduced
- **Clauses added to $U$ are simplified with respect to $P$**
Naive Proof Generation

Basic approach:
- Store (or dump) all intermediate proof steps
- Extract proof steps in post-processing

Problem: Necessary steps only known after the proof concludes
- Intermediate results are expensive to store
- Example: A ring with $X^4 = X$ is Abelian
  - Proof search (E): 5.4s
  - Proof search with inference dump: 11.4s
  - Post-processing: 17.6s
  - Temporary file size: 480 000 steps, 117MB
  - Proof size: 154 steps, 31 kB

Only suitable for small problems/short run-times
Optimized Proof Object Construction

Observation: Only clauses in $P$ are premises!

$P$
(processed clauses)

$U$
(unprocessed clauses)

Generate

Simplifiable?

Cheap
Simplify

$g$

Simplify

Proof recording:
- Simplified $P$-clauses archived
- Clauses record their history
- Inference rules
- $P$-clauses involved

Proof extraction:
- Track parent relation
- Topological sort
- Print proof
Optimized Proof Object Construction

Observation: Only clauses in $P$ are premises!

Proof recording:
- Simplified $P$-clauses are archived
- Clauses record their history
  - Inference rules
  - $P$-clauses involved
Optimized Proof Object Construction

Observation: Only clauses in $P$ are premises!

Proof recording:
- Simplified $P$-clauses are archived
- Clauses record their history
  - Inference rules
  - $P$-clauses involved

Proof extraction
- Track parent relation
- Topological sort
- Print proof
Optimized Proof Generation

Example: A ring with $X^4 = X$ is Abelian

▶ Naive approach
  ▶ Proof search (E): 5.4s
  ▶ Proof search with inference dump: 11.4s
  ▶ Post-processing: 17.6s
  ▶ Temporary file size: 480 000 steps, 117MB
  ▶ Proof size: 154 steps, 31 kB

▶ Optimized approach
  ▶ Proof search (E): 5.5s
  ▶ Proof search with inference dump: -
  ▶ Post-processing: -
  ▶ Temporary file size: -
  ▶ Proof size: 154 steps, 31 kB

▶ Example is typical
  ▶ Optimized overhead: 0.24% over TPTP 5.4.0
Proof Applications
Why Proofs?

Trust
- in the proof
- in the ATP system
- in the specification

Understanding
- of the proof
- of the domain
- of the search process

Learning
- of important domain statements
- of search control information
- of the domain structure
Proof Checking

Semantic proof checking
▶ Step-by-step check
▶ Verify semantic status (conclusion can be derived “somehow” from premises)
▶ Use alternative theorem prover (or configuration)

Syntactic proof checking
▶ Show correctness of individual inference rule applications
▶ With TPTP syntax: Requires proof reconstruction
▶ E.g. Metis in Isabelle/Sledgehammer
Proof Visualization
Proof Visualization

\[
\text{fof(c}_0\text{.0, conjecture, } (?[X3]: (human(X3) \& X3 \neq john))).}
\]

\[
\text{fof(c}_0\text{.4, negated_conjecture, } (\neg (?[X3]: (human(X3) \& X3 \neq john))).}
\]

\[
\text{fof(c}_0\text{.1, axiom, } (?[X3]: (human(X3) \& grade(X3) = a))).}
\]

\[
\text{fof(c}_0\text{.6, plain, } ((human(esk1_0) \& grade(esk1_0) = a))).}
\]

\[
\text{fof(c}_0\text{.2, axiom, } (grade(john) = f)).}
\]

\[
\text{cnf(c}_0\text{.12, plain, } (a = f)).}
\]

\[
\text{fof(c}_0\text{.3, axiom, } (a \neq f)).}
\]

\[
\text{cnf(c}_0\text{.10, negated_conjecture, } (esk1_0 = john)).}
\]

\[
\text{cnf(c}_0\text{.9, plain, } (grade(esk1_0) = a)).}
\]

\[
\text{cnf(c}_0\text{.7, negated_conjecture, } (X1 = john \& \neg human(X1))).}
\]

\[
\text{cnf(c}_0\text{.8, plain, } (human(esk1_0))).}
\]

\[
\text{cnf(c}_0\text{.13, plain, } ($false)).}
\]

\[
\text{cnf(c}_0\text{.11, plain, } (grade(john) = f)).}
\]
Another Example
Another Example

(A ring with $X^4 = X$ is Abelian)
Interactive Visualization
Learning

Heuristics learning
  ▶ Find formulas that frequently appear in proofs
  ▶ Generalize and reuse

Axiom selection
  ▶ Learn relationship between conjecture and useful axioms

...
Conclusion

Efficient proof generation is non-trivial, but possible

TPTP v3 is a useful and used standard for proof representation

Proof objects are useful for trust building and learning

Use of proof objects is still in its infancy - we need more tools
Conclusion

Efficient proof generation is non-trivial, but possible

TPTP v3 is a useful and used standard for proof representation

Proof objects are useful for trust building and learning

Use of proof objects is still in its infancy - we need more tools

Proof presentation is a big open area
Ceterum Censeo...

Bug reports for E should include:
- The exact command line leading to the bug
- All input files needed to reproduce the bug
- A description of what seems wrong
- The output of `eprover --version`