

Prover9 and Its Application to
Challenging Problems in Mathematics

Symbolic Guidance with
Proof Sketches and Hints

Robert Veroff
University of New Mexico

Czech Technical University
April 2019

Overview

- Context and perspective
- Look and feel (Prover9)
- Searching for a proof
- Advanced methods and features
- Applications

Objectives of Automated Deduction

- Mechanizing mathematics
 - repositories for accumulated knowledge
 - enforce correctness and rigor
- Automatic theorem proving
 - plug and chug
 - consistently and reliably prove “easy” problems easily
- Research tool
 - mathematically challenging problems
 - “interesting” (to someone)
 - new knowledge (e.g., open questions)
- Real-world applications

Background

- First-order logic with equality
- Problem representation
 - language of clauses
 - proof by contradiction
- Inference rules
 - resolution (modus ponens / syllogism)
 - paramodulation (equality substitution)
 - unification
- Demodulation (rewriting)
- Subsumption (deletion)

Clauses

Equivalences

$$p \rightarrow q \iff \neg p \vee q$$

$$(p \wedge q) \rightarrow r \iff \neg p \vee \neg q \vee r$$

Hyperresolution

$$\begin{array}{l} \neg P(x, y) \quad | \quad \neg Q(x, y) \quad | \quad R(x, y) . \quad \{a/x, b/y\} \\ P(a, x) . \quad \{b/x\} \\ Q(x, b) . \quad \{a/x\} \end{array}$$

$$R(a, b) .$$

Paramodulation

$$\begin{array}{l} P(f(a * x, g(x))) \quad \{b/x\} \\ \quad \quad \quad x * b = x \quad \{a/x\} \end{array}$$

$$P(f(a, g(b)))$$

The Task

Given an initial set C of clauses and a set of inference rules, find a derivation of the *empty clause* (for example, by the resolution of two conflicting clauses P and $\neg P$).

Procedure:

```
while (no proof found)
{
  select "given" clause G
  apply inference rules to G together with
    clauses from {initial} union {already given}
  process inferred clauses (demodulation, subsumption)
}
```

Processing Inferred Clauses

Demodulation and subsumption

- Otter Loop (e.g., Prover9)
 - when first generated
 - with *all* kept clauses
- Discount Loop (e.g., E)
 - when chosen as given
 - with already given clauses

The difference is profound, trading off runtime performance against the possibility of making an important simplification.

Look and Feel

Example input files ...

Success vs. Failure

Choice of representation, inference rules (e.g., which variations of resolution to use), rewriting and deletion strategies all matter, but it mostly comes down to *given selection*.

Given selection is the focus of most research activity (e.g., the development of machine learning methods).

Given Selection

Methods

- Symbol count (weighting)
- User-defined weighting patterns
- Attribute-based selection
- Subsumption-based selection (hints)
- Model-based selection (semantic guidance)
- Statistical methods (e.g., machine learning)

The user can specify detailed recipes for combining these mechanisms, including rules based on clause properties.

Demodulation (Rewriting)

Purpose

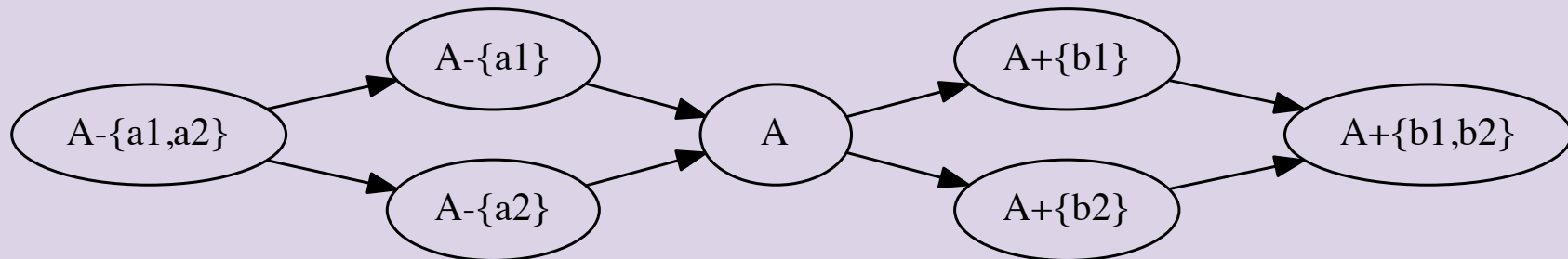
- simplification (identify useful lemmas)
- canonical forms (eliminate redundant information)

Effectiveness can depend on term ordering

- Methods: KBO, RPO ...
- Ordering function and predicate symbols

Advanced Methods for Given Selection

Say we want to prove a theorem t in a target theory A .



We can learn given-selection strategies by looking at

- proofs of t in extensions of A (*proof sketches*)
- countermodels of t in weakenings of A (*semantic guidance*)

Proof Sketches

Consider a derivation of some c_n as a sequence of clauses,

$$c_1, c_2, \dots, c_i, \dots, c_j, \dots, c_n$$

where

- c_i is an extra assumption for the target theory A
- derived clause c_j has c_i in its derivation history

c_j either is derivable from A or it is not.

- if yes, it suffices to find a new derivation of c_j
- if no, it suffices to “bridge the gaps” to the consequences of c_j

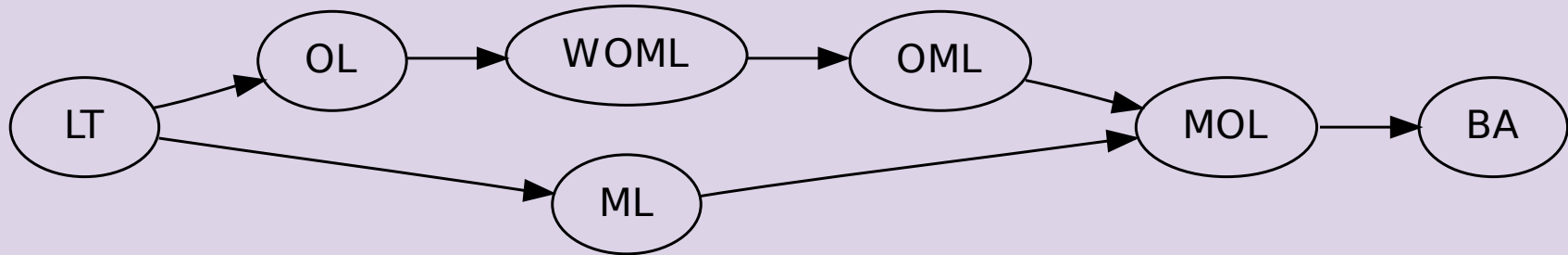
In either case, we have a partial proof that *might* be easier to complete than finding a proof from scratch.

The Proof Sketches Method

- Idea: Collect proofs of the target theorem in extended theories (i.e., with extra assumptions) and have a selection bias for clauses that match clauses in these proofs.
- The emphasis is on the *sufficiency* of the collected “proof sketches”. This does not preclude finding a different proof.
- Move up the hierarchy by systematically generating new proof sketches with fewer extra assumptions, including all previous proof sketches for guidance.
- The challenge is to find effective extensions of the target theory (extra assumptions).

Where Do Extra Assumptions Come From?

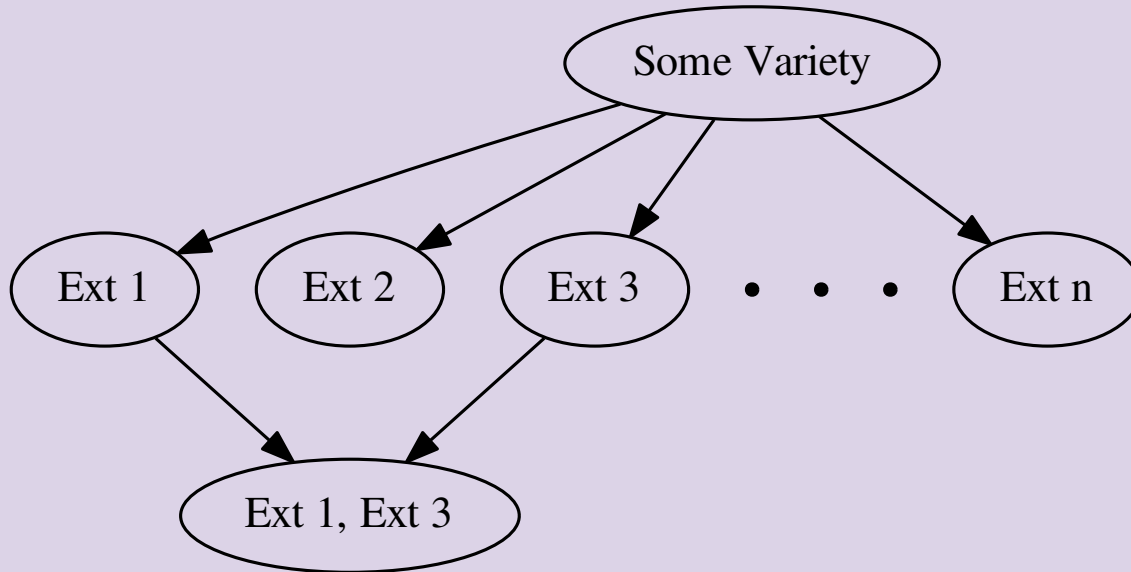
Example: Lattice Theory Hierarchy



\mathcal{LT}

- + Invertibility and Compatibility (\mathcal{OL})
- + Weak Orthomodularity (\mathcal{WOML})
- + Orthomodularity (\mathcal{OML})
- + Modularity (\mathcal{MOL})
- + Distributivity (\mathcal{BA})

Other Extensions



Examples:

- $x * y = y * x$
- $(x * y) * z = x * (y * z)$
- $x * x = x$

Proof Sketches in Prover9

- Proof sketches can be included as *hints*.
- Given selection can be biased toward clauses that *match* (subsume) hints.
- Hints also can come from
 - the mathematician
 - proofs of related theorems in the same theory

Proving target theorems with multiple extra assumptions and then iteratively eliminating them has been an especially effective method for proving difficult theorems.

Semantic Guidance

- Say $A \Rightarrow c$ is a theorem but $A - \{a\} \Rightarrow c$ is *not* a theorem.
- Let I be an interpretation (model) that satisfies $A - \{a\}$ and falsifies c .
- Key observation: In order to infer c at least one parent p of the inference must evaluate to **False** under I . Similarly for the parents of p , and so on ...
- It follows that a proof of c from A will necessarily include steps that evaluate to **False** under I .

... a and a subset of a 's descendants.

- Idea: Have some selection bias for clauses that evaluate to **False** under I .
- The challenge is to find weakenings of A that yield good candidate interpretations I .

... want a to be minimal.

Semantic Guidance in Prover9

- Mace4 can be used to find finite models and counterexamples.
- Prover9 can include the resulting interpretations as input.
- The user can specify how to use the evaluation of clauses for given selection.