Automated Theorem Proving

Jeremy Avigad

Department of Philosophy and
Department of Mathematical Sciences
Carnegie Mellon University

May 2015

Sequence of lectures

1. Formal methods in mathematics
2. Automated theorem proving
3. Interactive theorem proving
4. Formal methods in analysis

Automated reasoning

Ideal: given an assertion, φ, either

- provide a proof that φ is true (or valid), or
- give a counterexample

Dually: given some constraints, φ, either

- provide a solution, or
- prove that there aren't any.

Partial solutions:

- search for proofs
- search for solutions

Automated reasoning

One can distinguish between:

- Domain-general methods and domain-specific methods
- Decision procedures and search procedures
- “Principled” methods and heuristics

Automated reasoning

Domain-general methods:

- Propositional theorem proving
- First-order theorem proving
- Equational reasoning
- Higher-order theorem proving
- Nelson-Oppen "combination" methods

Domain-specific methods:

- Linear arithmetic (integer, real, or mixed)
- Nonlinear real arithmetic (real closed fields, transcendental functions)
- Algebraic methods (such as Gröbner bases)

References

An excellent starting point:

- Harrison, John, *Handbook of Practical Logic and Automated Reasoning*, Cambridge University Press, 2009.

An authoritative reference:

- J. Allen Robinson and Andrei Voronkov, *Handbook of Automated Reasoning*, MIT Press and North Holland, 2001.

General message: automated methods do especially well on large, homogeneous problems, but often fail to capture even the most straightforward mathematical inferences.

Propositional logic

In this talk, I will focus on *classical* logic.

Start with variables $p, q, r, \ldots$. (Semantics: each can be either "true" or "false".)

Build compound formulas with $\wedge$, $\vee$, $\neg$, $\to$, for example

$$p \wedge q \wedge \neg r \to \neg(\neg p \vee s) \vee (\neg s \wedge q).$$

A formula φ is

- *satisfiable* if there is some assignment of truth values that makes is true,
- *valid* if every truth assignment makes it true.

Propositional logic

Note: φ is valid iff $\neg\varphi$ is unsatisfiable.

Challenge: given φ, prove φ, for find a falsifying assignment.

Dually: given φ, find a satisfying assignment, or establish that there is none.

Propositional logic

A variable p is called an *atomic* formula, p and $\neg p$ are *literals*

Normal forms:

- Negation normal form (NNF): built up from literals using only $\wedge$ and $\vee$. Example:

$$\neg p \vee \neg q \vee r \vee (p \wedge \neg s) \vee (\neg s \wedge q).$$

- Disjunctive normal form (DNF): $\bigvee \varphi_i$, where each φ_i is a conjunction of literals.
- Conjunctive normal form (CNF): $\bigwedge \varphi_i$, where each φ_i is a disjunction of literals.

Putting a formula in NNF is cheap, but putting a formula in DNF or CNF can yield exponential increase in length.

Propositional logic

Tseitin's trick (1968): given φ, one can find an equisatisfiable DNF φ' efficiently (length $O(n)$).

Dually, there is a short CNF φ'' that is valid iff φ is.

Idea: introduce new variables to define subformulas and avoid blowup.

Rephrased challenge:

- Decide whether a formula in DNF is satisfiable.
- Decide whether a set of clauses (disjunctions of literals) is satisfiable.
- Decide whether a formula in CNF is valid.

Propositional logic

I will describe three approaches

- tableau (cut-free) proofs
- resolution
- DPLL (Davis-Putnam-Logemann-Loveland)

Tableau proofs

Consider a 1-sided sequent calculus:

- Use formulas in negation normal form ($\wedge$, $\vee$, p, $\bar{p}$).
- Define $\neg\varphi$ by switching $\wedge$ and $\vee$, p and $\bar{p}$, e.g.

$$\neg(p \wedge (\bar{q} \vee r)) \quad \mapsto \quad \bar{p} \vee (q \wedge \bar{r}).$$

- A *sequent* is a finite set $\{\varphi_1, \ldots, \varphi_n\}$, read disjunctively.

Rules:

$$\Gamma, p, \bar{p} \qquad \frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi} \qquad \frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}$$

Tableau proofs

Notice that each rule is “bidirectional”: the conclusion is valid iff the hypothesis is.

Reading upwards, the $\wedge$ and $\vee$ rules remove one connective.

A sequent with only literals (variables and negated variables) is valid if and only if it is an axiom.

Applying the rules backwards then either yields a proof, or a counterexample.

Resolution theorem proving

Using Tseitin's trick, we can reduce the goal of proving φ to refuting a set of clauses.

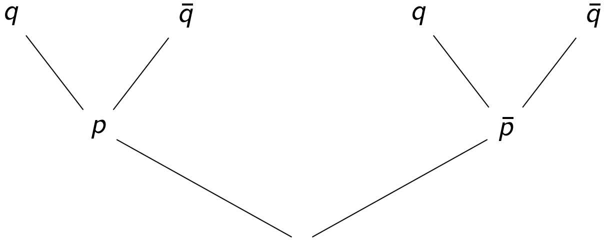
Let Γ, Δ stand for clauses, e.g. $\{p, q, \bar{r}, \bar{s}, t\}$.

Use the resolution rule:

$$\frac{\Gamma \vee \varphi \qquad \Delta \vee \neg\varphi}{\Gamma \vee \Delta}$$

Keep deriving new clauses in this way, until we obtain the empty clause $\{\}$, or cannot make any more progress.

DPLL



- Try to decide satisfiability of a set of clauses.
- Keep track of which clauses are not yet satisfied.
- Use unit propagation.
- Backtrack when a clause is unsatisfiable.

SAT solvers

Most modern SAT solvers use variants of DPLL.

Innovations:

- non-chronological backtracking
- conflict-driven clause learning
- efficient data structures and implementation tricks

Modern SAT solvers can handle tens of thousands of variables and millions of clauses.

First-order theorem proving

First-order logic adds relations $r(x, y, z)$, functions $f(x, y)$, $g(x)$, and quantifiers $\exists x\ \varphi(x), \forall x\ \varphi(x)$.

Formulas can still be put in negation normal form.

For tableau search, add the following rule for the universal quantifier:

Some rules for working backwards:

$$\frac{\Gamma, \varphi(a)}{\Gamma, \forall x\ \varphi(x)}$$

First-order theorem proving

What about the existential quantifier?

$$\frac{\Gamma, \exists x\ \varphi(x), \varphi(?t(a, b, c, \ldots))}{\Gamma, \exists x\ \varphi(x)}$$

Notes:

- $?t$ can be instantiated to any term involving the other parameters.
- It's best to delay the choice.
- More than one term may be needed.
- All the background knowledge is lumped into Γ.

Unification

Suppose you know that for every x and y, $A(x, f(x, y)) \to B(x, y)$.

Suppose you also know that for every w and z, $A(g(w), z)$.

Then you can conclude $B(g(w), y)$ by solving $x = g(w)$ and $z = f(g(w), y)$.

Theorem (Robinson)

There is an algorithm that determines whether a set of pairs $\{(s_1, t_1), \ldots, (s_n, t_n)\}$ *of first-order terms has a unifier, and, if it does, finds a most general unifier.*

Skolemization

If φ is the formula $\forall x\, \exists y\, \forall z\, \exists w\, \theta(x, y, z, w, u)$, the *Skolem normal form* φ^S is the formula

$$\forall x, z\, \theta(x, g(x, u), z, f(x, g(x, u), z, u), u)$$

Dually, the Herbrand normal form ψ^H of ψ replaces the universal quantifiers.

- $\vdash \varphi^S \to \varphi$
- If $\varphi^S \vdash \alpha$ then $\varphi \vdash \alpha$.
- $\vdash \psi \to \psi^H$
- If $\Delta \vdash \psi^H$ then $\Delta \vdash \psi$.

Putting it all together: $T \vdash \varphi$ if and only if $T^H \vdash \varphi^S$.

Resolution

Herbrand's theorem (1930): $T^H \vdash \varphi^S$ if and only if there is a *propositional proof* of a disjunction of instances of φ^S from instances of T^H.

Resolution tries to prove $\perp$ from $T^H \cup \{\neg\varphi^S\}$:

- Leave the universal quantifiers implicit.
- Put all formulas in conjunctive normal form, and split up conjuncts.
- So, the goal is to prove $\perp$ from *clauses*, i.e. disjunctions of atomic formulas and literals.
- Use the resolution rule:
$$\frac{\Gamma \vee \varphi \qquad \Delta \vee \neg\varphi}{\Gamma \vee \Delta}$$
More generally, use unification to instantiate clauses to the form above.

Resolution

Main loop:

1. Use resolution to generate new clauses.
2. Check for redundancies (subsumption) and delete clauses.

Issues:

- How much effort to put into each phase?
- How to choose new clause (biggest, widest, heaviest, ...)?
- How to handle equality? (paramodulation, superposition)
- How to handle other equivalence relations, transitive relations?
- How to distinguish different kinds of information (like sort information)?
- How to incorporate domain specific information, like arithmetic, or AC operations?

Comparison

General approaches to theorem proving:

- global / top-down (e.g. tableaux): goal directed, works backwards to construct a proof (or countermodel)
- local / bottom-up (e.g. resolution): start with a set of facts, reason forwards to derive additional facts

Equality reasoning

It is reasonable to simplify terms:

- $x + 0 = x$
- $x > 0 \to |x| = x$
- $y \neq 0 \to (x/y) * y = x$
- $x + (z + (y + 0) + x) = x + x + y + z$

There are stand-alone equational systems, but equational reasoning is also built-in to first-order systems.

Higher-order unification

Sometimes mathematics requires higher-order unification:

$$P(0) \wedge \forall x\, (P(x) \to P(x+1)) \to \forall x\, P(x)$$

or

$$\sum_{x \in A} (f(x) + g(x)) = \sum_{x \in A} f(x) + \sum_{x \in A} g(x)$$

Notes:

- Second-order unification is undecidable (Goldfarb).
- Huet's algorithm is complete.
- Miller patterns are a decidable fragment.

Decision procedures

Full first-order theory:

- Quantifier elimination (integer / linear arithmetic, *RCF*, *ACF*)
- "Global" methods (Cooper, CAD)
- Reductions to Rabin's *S2S*
- Feferman-Vaught (product structures)

Sometimes it is enough to focus on the universal fragment:

- Some theories are only decidable at this level (e.g. uninterpreted functions)
- Can be more efficient (integer / linear arithmetic).
- Can use certificates.
- A lot of mathematical reasoning is close to quantifier-free.
- These can be *combined*.

Combining decision procedures

Theorem (Nelson-Oppen, 1979)

Suppose T_1 and T_2 are "stably infinite" and there is a decision procedure for their universal consequences. Suppose that the languages are disjoint, except for the equality symbol. Then the set of universal consequences of $T_1 \cup T_2$ is decidable.

In particular, if T_1 and T_2 have only infinite models, they are stably infinite.

This allows you to design decision procedures for individual theories and then put them together.

Combining decision procedures

First idea: one can "separate variables" in universal formulas.

That is, $\forall \vec{x}\, \varphi(\vec{x})$ is equivalent to $\forall \vec{y}\, (\varphi_1(\vec{y}) \vee \varphi_2(\vec{y}))$, where φ_1 is in the language of T_1, and φ_2 is in the language of T_2.

To do this, just introduce new variables to name subterms.

Second idea: the Craig interpolation theorem.

Theorem (Craig, 1957)

Suppose ψ_1 is a sentence in L_1 and ψ_2 is a sentence in L_2, such that $\vdash \psi_1 \to \psi_2$. Then there is a sentence θ in $L_1 \cap L_2$ such that

- $\vdash \psi_1 \to \theta$
- $\vdash \theta \to \psi_2$

Combining decision procedures

Let φ be any universal sentence, equivalent to $\forall \vec{x}\ (\varphi_1(\vec{x}) \vee \varphi_2(\vec{x}))$.

Then $T_1 \cup T_2 \vdash \varphi$ if and only if there is θ in the common language, such that

- $T_1 \cup \{\neg\varphi_1(\vec{x})\} \vdash \theta(\vec{x})$
- $T_2 \cup \{\neg\varphi_2(\vec{x})\} \vdash \neg\theta(\vec{x})$

We can assume θ is in disjunctive normal form. All that each disjunct can do is declare certain variables equal to one another, and others unequal!

Use the decision procedures for T_1 and T_2 to test each possibility.

Combining decision procedures

Nelson-Oppen methods are based on this idea.

- A fast propositional SAT solver "core" tries to build a satisfying assignment.
- Individual decision procedures examine proposals, and report conflicts.
- The SAT solver incorporates this information into the search.
- Some systems go beyond the universal fragment, for example, instantiating universal axioms in sensible ways.

For example, SMT solvers user methods for integer/linear arithmetic that support backtracking search.

Combining decision procedures

SMT solvers are both constraint solvers and theorem provers.

They incorporate domain-specific methods in a domain-general framework.

They are modular and extensible.

There are used to verify hardware and software, but also to synthesize objects.

Summary

Formal methods provide languages for

- expressing mathematical background,
- making mathematical assertions, and
- describing mathematical objects.

They provide general ways of

- searching for proofs, and
- searching for objects.

Combination methods provide ways of incorporating domain-specific methods.

Thesis: the matematical potential has not yet been realized.