Challenges for the Formal Approach to Proof

John Harrison

Intel Corporation

ASL Meeting

March 21, 2005

The cruelty of really doing proofs

Some pioneers like Frege and Peano actually wanted to use their formal systems in practice.

With the rise of Hilbert’s program, this came to be seen as something only to be done ‘in principle’.

Russell claims that his intellect ‘never quite recovered from the strain’ of writing *Principia*

Are long formal proofs really more reliable than informal ones? Probably the opposite.

Let the computers do it

Doing formal proofs is much more practical if we can delegate some of the tedium to computers.

Moreover, this should achieve a *real* improvement in reliability.

Some theorem provers are a mass of powerful, but complex and perhaps unreliable, inference rules.

We claim it's often desirable to perform all proofs using something close to a traditional system of deduction rules.

LCF

A methodology for making a prover extensible by ordinary users, yet reliable.

Idea due to Milner in Edinburgh LCF project, now used in many other sytems like Coq, HOL, Isabelle and Nuprl.

- Implement in a strongly-typed functional programming language (usually a variant of ML)
- Make `thm` (‘theorem’) an abstract data type with only simple primitive inference rules
- Make the implementation language available for arbitrary extensions.

The problem of producing proofs

Many efficient decision procedures do not naturally produce a proof using traditional inference rules.

Their justification is often based on metatheoretic reasoning.

Simple example: a propositional formula with $\Leftrightarrow$ as the only connective can be seen to be valid by seeing that each atom occurs an even number of times.

This doesn't obviously help us to produce a proof using one of the standard sets of inference rules for propositional logic.

Not as bad as it seems

Over the years a distinctive methodology has been developed to implement standard decision procedures so they produce proofs.

Quite often, traditional algorithms can easily be reformulated to produce proofs without an infeasible increase in runtime or complexity.

Classic example: Kalmár’s completeness proof for propositional logic is essentially a proof-producing formulation of the truth-table method.

Reflective reasoning

Often, metatheoretic reasoning can be performed by appealing to general object-level theorems via a partial semantic reflection. (No new ‘reflection principles’ needed for modest subset of logic.)

Example: HOL implementation of Cooper’s algorithm for Presburger arithmetic. Define a type of trees representing NNF Presburger formulas:

```
let cform_INDUCT,cform_RECURSION = define_type
  "cform = Lt int
         | Gt int
         | Eq int
         | Ne int
         | Divides int int
         | Ndivides int int
         | And cform cform
         | Or cform cform
         | Nox bool";;
```

The semantics of formulas

The meaning of these formulas is now defined recursively:

```
let interp = new_recursive_definition cform_RECURSION
  ‘(interp x (Lt e) = x + e < &0) /\
   (interp x (Gt e) = x + e > &0) /\
   (interp x (Eq e) = (x + e = &0)) /\
   (interp x (Ne e) = ~(x + e = &0)) /\
   (interp x (Divides c e) = c divides (x + e)) /\
   (interp x (Ndivides c e) = ~(c divides (x + e))) /\
   (interp x (And p q) = interp x p /\ interp x q) /\
   (interp x (Or p q) = interp x p \/ interp x q) /\
   (interp x (Nox P) = P)‘;;
```

We can map into and out of the ‘reflected’ representation in linear time, then at the reflected level appeal to metatheorems.

Certifying decision procedures

The ideal situation is where a decision procedure can produce a ‘certificate’ which can be efficiently checked by proof.

- Generally much more efficient since the entire decision procedure doesn’t need to be internalized
- The implementation is usually much simpler (for the same reason).

Simple examples: satisfying valuation for propositional formula, resolution proof discovered by extensive search, factorization for number to prove compositeness.

Main theoretical question

Which decision procedures do lend themselves to a separate certificate? Generally, we don't know the answer, but we want to find out more.

If efficient = polynomial, certificates for *unsatisfiability* of propositional formulas exist iff NP = co-NP (which is an open problem).

For primality, Pratt certificates and later refinements exist. These show primality testing to be in NP $\cap$ co-NP (we now know it's in P).

What do we know about other decision procedures?

Nullstellensatz refutations

Hilbert’s weak Nullstellensatz tells us that a conjunction of polynomial equations

$$p_1(\overline{x}) = 0 \wedge \cdots \wedge p_n(\overline{x}) = 0$$

is inconsistent over $\mathbb{C}$ iff there exist ‘cofactor’ polynomials $q_i(\overline{x})$ with

$$p_1(\overline{x})q_1(\overline{x}) + \cdots + p_n(\overline{x})q_n(\overline{x}) = 1$$

We can arrange for standard decision methods like Gröbner bases to produce the cofactors.

But how large are the cofactors? Perhaps just checking the polynomial identity is non-trivial!

Real Nullstellensatz refutations

Similar certificates (based on sums of squares) exist for inconsistency over $\mathbb{R}$.

One way of finding them (Parrilo . . .) is using semidefinite programming; checking the certificate can be much easier than finding it. Suppose we want to verify the following:

$$\forall a\ b\ c\ x.\ ax^2 + bx + c = 0 \Rightarrow b^2 - 4ac \geq 0$$

It's easy once we find the certificate:

$$b^2 - 4ac = (2ax + b)^2 - 4a(ax^2 + bx + c)$$

But again, how big may the certificates get?

Applications

There are many applications where we want to certify decision methods that rely on extensive case analysis.

- Configurations of cache coherence protocols verified by exhaustive state enumeration in model-checking tools
- Polynomial inequalities in Flyspeck project currently verified using interval enclosure over many small subintervals.

Is there a natural approach to these problems where we can easily obtain a proof?

Summary

- Producing proofs/certificates from decision procedures has obvious advantages
- In many cases, we can quite easily adapt standard algorithms to produce proofs.
- Need more information about where efficient certificates exist.
- Perhaps some interesting theoretical questions relating to NP-completeness.