Automated Assistance for Proof Assistants

Lawrence C Paulson

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Tech Support Question

Dear Aunt Verity,

I am trying to prove this obvious fact:

$b < a \;\Rightarrow\; c < 0 \;\Rightarrow\; c \times a < c \times b$

It has been 3 days and I'm getting nowhere. What can I do?

Yours, Confused.

Aunt Verity’s Reply

Dear Confused,

That theorem is already in the system. It is called mult_strict_left_mono_neg. You must look harder next time.

Yours, Aunt Verity

Question #2

Dear Aunt Verity,

Now I am trying to prove

$$b < a \;\Rightarrow\; 0 < c \;\Rightarrow\; -a \times c < -(c \times b)$$

It's practically the same as the other one but I still can't do it.

Yours, Desperate.

Reply #2

Dear Desperate,

Moving symbols in a theorem can be tricky. After a few years' experience, such tasks should not take more than one hour. Work hard and one day you shall succeed. Meanwhile, try this [horrible code]

Yours, Aunt Verity

Question #3??

Dear Aunt Verity,

Instead of struggling to prove theorems, I have decided to sell buggy C software and charge extra for technical support.

Yours, Joyful At Last.

Automation Ideas

- Rewriters and auto-tactics can be weak.
- Decision procedures are powerful, but only for narrow domains.
- SMT solvers are best for ground problems.
- Can general-purpose automatic theorem provers (ATPs) make a difference?

Advantages of ATPs

- They are fully automatic, even with quantifiers.
- They handle *large* problems.
- They are clever with *equality*: not just directed rewriting!
- They find long, obscure proofs.

Drawbacks of ATPs

- They use untyped first-order logic (FOL); we don't!
- They need to run for a long time.
- They often fail.

Users risk wasting time and effort.

Ideas for a *Useful* Tool

- *One-click invocation*
  - automatic translation to FOL
  - automatic selection of lemmas
- *Background execution*: we don’t have to wait! (let’s exploit our multi-core machines!)
- *Source-level proof reconstruction*: we don’t have to call ATPs next time!

Isabelle Overview

- *Generic* proof assistant: extensible to support ZF set theory and other logics.
- (using Huet's higher-order unification!)
- Isabelle/HOL: classical higher-order logic (simple type theory)
- *Some automation*: rewriting engine, arithmetic solvers, backtracking search, automatically referring to 2000 lemmas.

Encoding Types in FOL

- Isabelle's type system is *order-sorted polymorphism* (as in Haskell).
- Type classes, such as *partial ordering*, are defined by axioms.
- Types can be modelled as first-order terms, type classes as predicates.
- Modelling the types prevents the incorrect use of properties such as *transitivity*.

Translation to FOL

- Detect whether the problem is already first-order (no function variables...)
- Convert to clause form, eliminating higher-order features if necessary
- Include *some* type information

Effectiveness Issues

- We don’t ask users to select relevant lemmas: that’s too much work.
- The full Isabelle lemma library converts to 8500 clauses!
- ATPs gag if you give them such huge problems.
- We need *automatic relevance filtering*.

Soundness Issues

- Attaching types to all terms and subterms is safe, but quadratic in space.
- Omitting types admits many absurd proofs.
- We include enough types to disambiguate polymorphic constants.
- *This still admits absurd proofs!*

Reconstruction Issues

- Proof reconstruction is essential, since we use unsound translations.
- ATPs use many different inference rules; they are complicated.
- Their output is incomplete and ambiguous.

Related Work

- KIV, integrated with the prover 3TAP
- Coq, integrated with the prover Bliksem
- Omega, integrated with numerous tools
- HOL, integrated with Metis: a prover designed to allow proof reconstruction

The Metis Prover

- Designed by Joe Hurd for use with HOL4
- A complete implementation of the superposition calculus
- ...with an ML interface to support proof reconstruction.
- It's good enough to prove modest-sized problems.

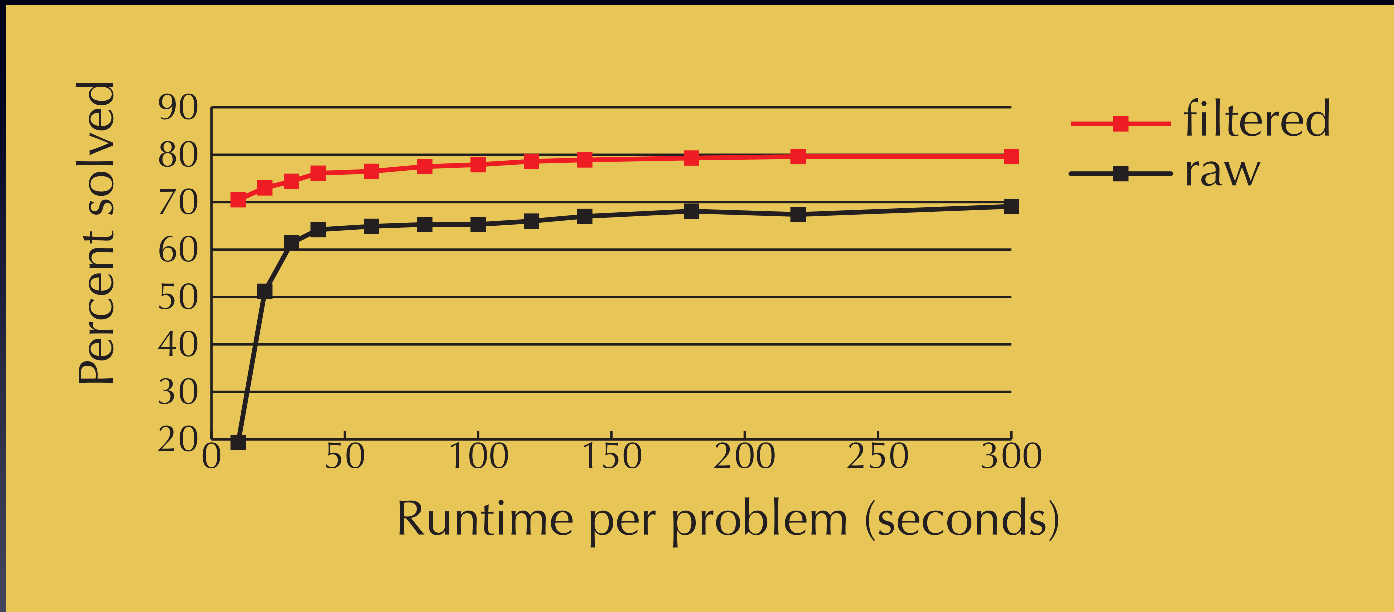
Fixing Our Issues

- Like KIV, use *relevance filtering* to reduce problem size.
- First, a simple signature-based filter reduces a problem from 8500 clauses to say 300.
- Second, use *the ATP itself* as a giant relevance filter, leaving perhaps 7 clauses.
- For proof reconstruction, let Metis prove it again!

Relevance Filtering

- A clause is relevant if it shares “enough” symbols with the goal being proved.
- The symbols of relevant clauses are used to measure the relevance of other clauses.
- The iteration must be limited, or too many clauses become relevant.
- The algorithm is ad-hoc but effective.

Effect of Relevance Filtering



Filtering gives a higher success rate, esp. for short runtimes. (Figures for E prover.)

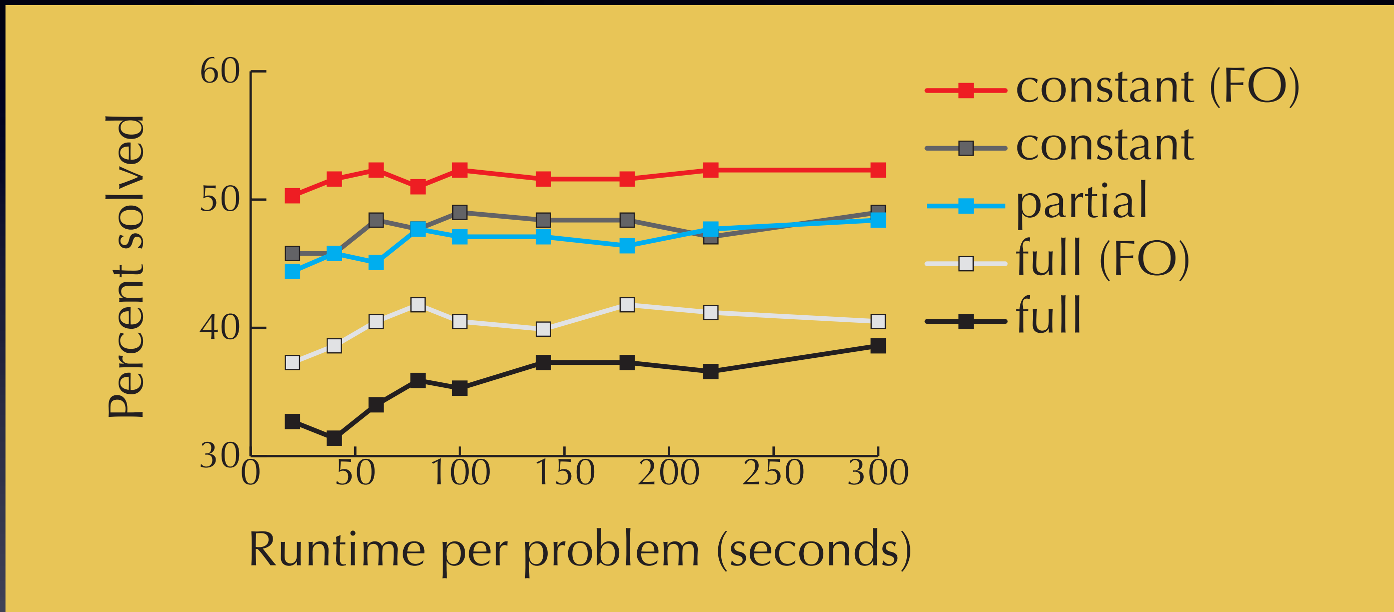
Higher-Order Problems

- We cannot hope for full higher-order reasoning from first-order provers.
- We merely remove higher-order features to make the problems look first-order.
  - explicit "apply" function and "is true" predicate for booleans
  - removal of λ by combinators or λ-lifting

HO Translations

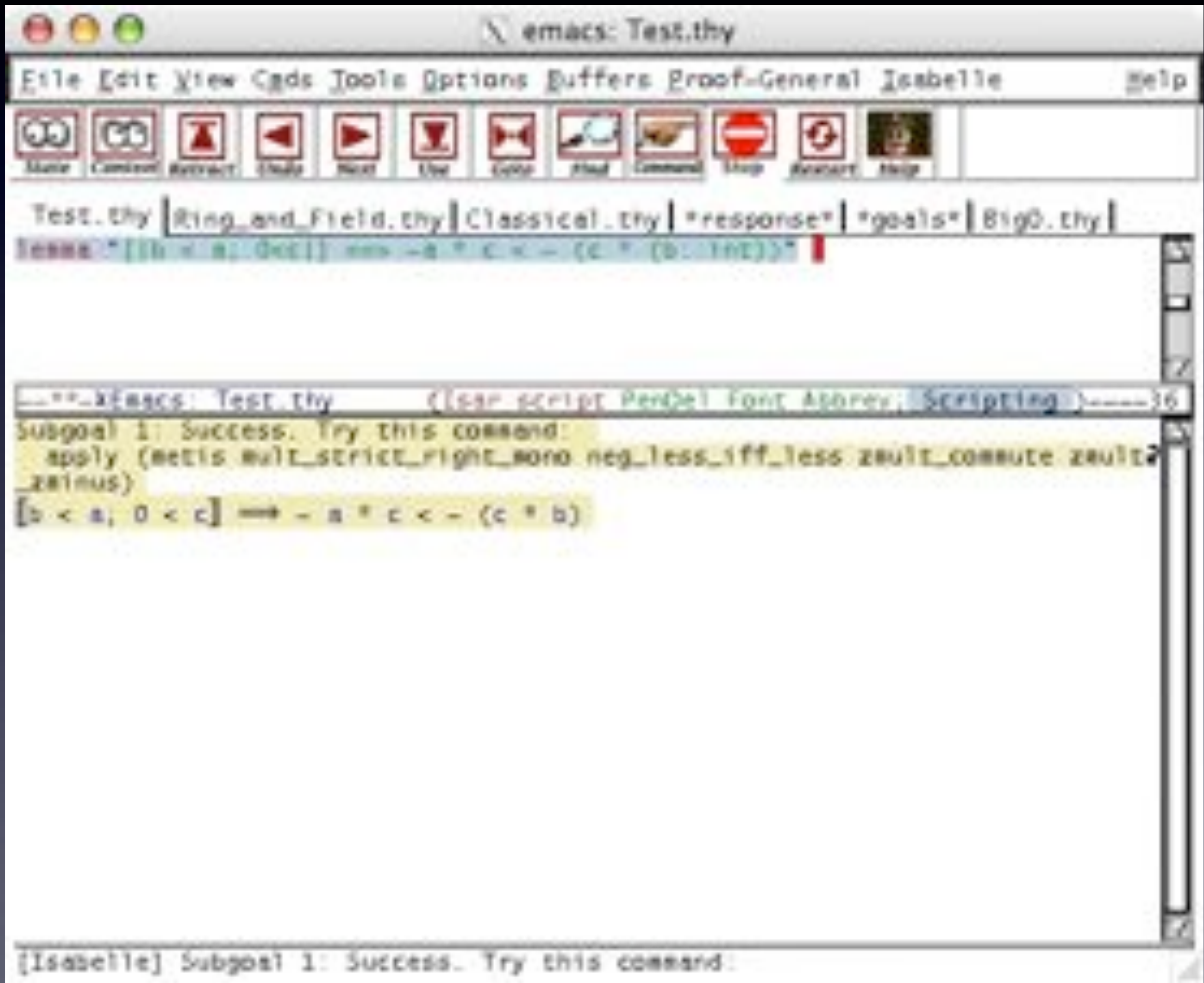
- We tried many treatments of types:
  - *full types*: sound but too big (quadratic!)
  - *reduced types*: compact but unsound
- For terms, do we preserve the full apply-structure, or use *built-in function application*?
- We ran many, many tests!

Effects of Translations



The difference between best and worst is immense. (Figures for E prover.)

Source-Level Proofs



Single-Step Proofs

- The resolution proof can be emulated in Isabelle, line by line or in small chunks.
- Each step is a separate Metis call.
- Such proofs are useful if Metis cannot prove the theorem in a single call.
- This requires an ATP that outputs TSTP format. (Currently, only the E prover)

A Single-Step Proof

Some Findings

- Naive relevance filtering is surprisingly effective (and fast).
- Unsound methods coupled with checking can be better than strictly sound methods.
- There is no substitute for extensive experimentation with real data.

Final Remarks

- The ATP linkup offers one-click assistance.
- It is available at any point in a proof.
- It helps novices by finding easy proofs and many of moderate difficulty.
- It gives multi-core machines a purpose.
- It is not a magic bullet for hard problems.

Dear Aunt Verity,

I have completed a deep and difficult proof, but I just can't decide which journal to publish it in. Help!!

Yours, Helpless.

Acknowlegements

- Claire Quigley: process management

- Kong Woei Susanto: Metis

- Jia Meng: relevance, HO translations, etc.

- EPSRC project GR/S57198/01 *Automation for Interactive Proof*

EPSRC Engineering and Physical Sciences Research Council