Sledgehammer: a Saga

Lawrence C Paulson

Vampire Workshop, 1 July 2024, Nancy, France

Origins

- A suggestion by Andrei Voronkov (at IJCAR 2001 in Siena?): let's combine Isabelle with a **real** theorem prover
- Meetings with Weidenbach and Siekmann in Saarbrücken
- A grant of £249,905 starting in January 2004
- … and a report of early results that July, at IJCAR 2004
- First release early in 2007, integrating Isabelle with E, SPASS, Vampire

Some precursors and influences

Ωmega (Siekmann, Benzmüller et al.)

KIV + $_{3}T^{A}P$ (Ahrendt, Beckert et al.)

Integrating Gandalf and HOL (Hurd)

Coq + Bliksem (Bezem et al.)

Unfortunately, they demanded **too much work** from the user

“Now that 2GHz processors are commonplace, we should abandon the traditional mode of interaction, where the proof tool does nothing until the user types a command. Background processes … should try to prove the outstanding subgoals.”

[from the original proposal, 2003]

Original design criteria

- **Easy invocation** (1-click, or even 0-click)
- **Automatic translation** from higher-order logic to first-order logic
- Instant access to the **entire lemma library**, with relevance checking
- Result as a **proof certificate**
  - To avoid having to **rerun** the search
  - To avoid **trusting external tools**

First Working Prototype

The tasks

Relevance filtering

Translating to FOL: types

Translating to FOL: λ-bindings

Proof reconstruction

Relevance filtering

[AKA *premise selection*]

- An Isabelle session may have 10,000+ accessible facts
- Theorem provers (at that time) could cope with a couple of hundred
- Relevance may be more obvious to the interactive prover (cf KIV)
- We adopted a crude approach based on **symbol occurrences**

Translating to FOL: types

A fully typed translation is **heavy** (quadratic), burying the formulas themselves

$$E = mc^2$$

$$((=)E)(\times m(\uparrow c2))$$

So I adopted a **partially typed translation** (unsound!)

… handling **polymorphism** and **type classes**

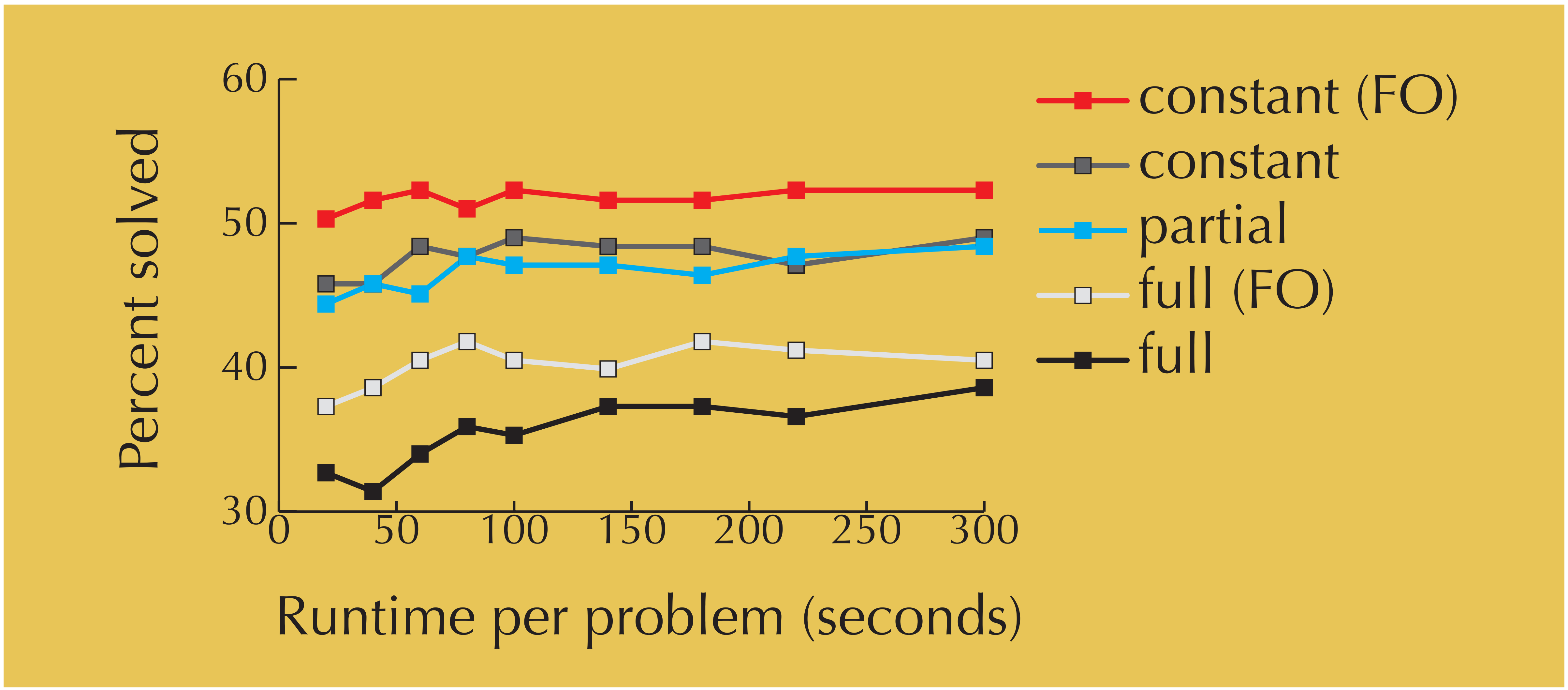
[Joe Hurd had success with a **completely typeless** translation]

Translating to FOL: λ-bindings

- Translation approaches (*neither works well*!) include:
  1. **Combinator** form S, K, I, B, C, ...
  2. **λ-lifting** (generating new function definitions)
- Have an explicit "apply" function and "is true" predicate for booleans, but **full higher-order reasoning** is not possible
- All of this omitted if the problem is **fully first-order**; in fact a "smooth" translation is possible

Thousands of hours of testing

Here we compare various translations by % problems solved



Proof reconstruction

Proofs given by ATPs are too ambiguous to use

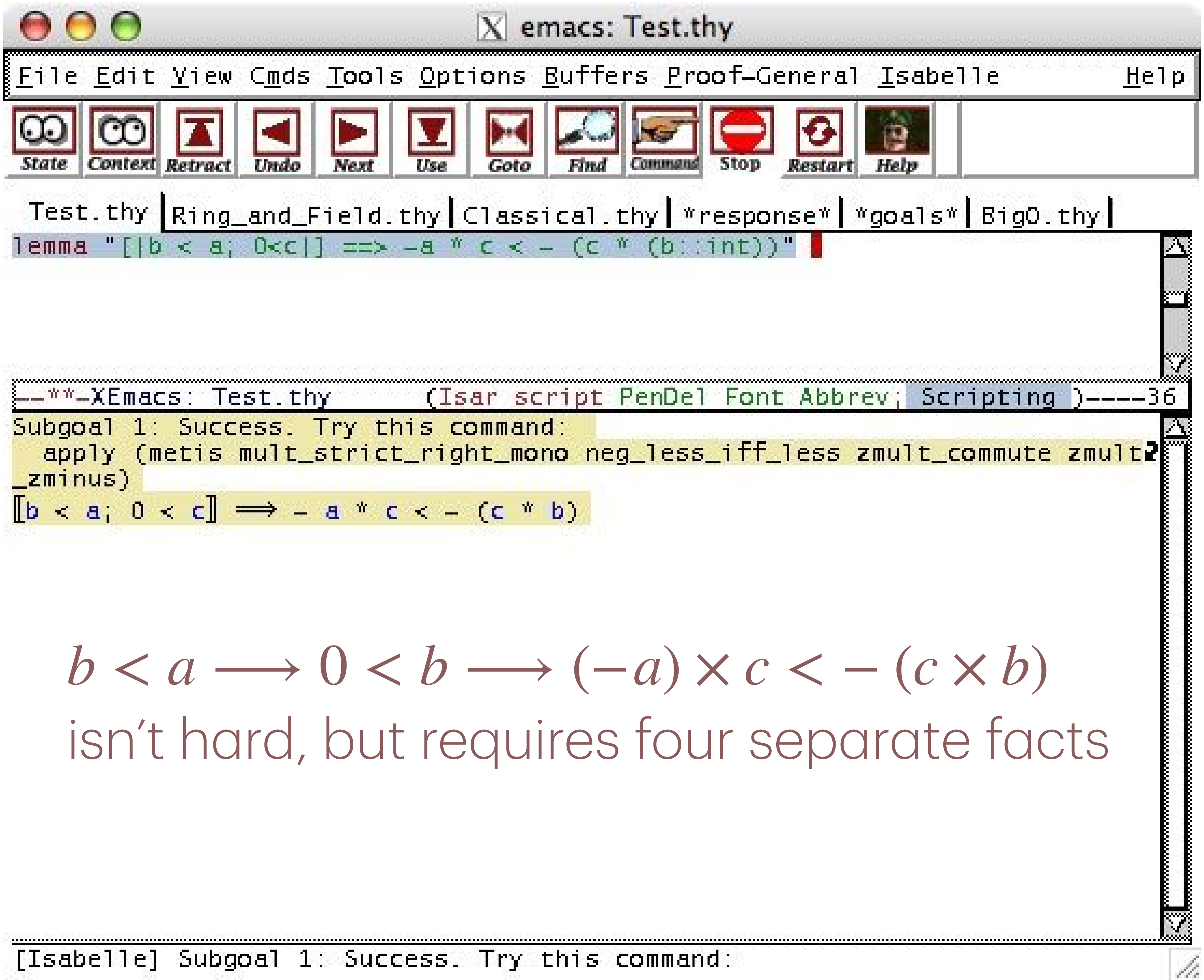
So we decided to use ATPs as powerful **relevance filters**

From the proof we extract nothing but the fact names

... giving them to one of Isabelle's own proof tools

Hurd's **metis**, a superposition prover integrated with the kernel

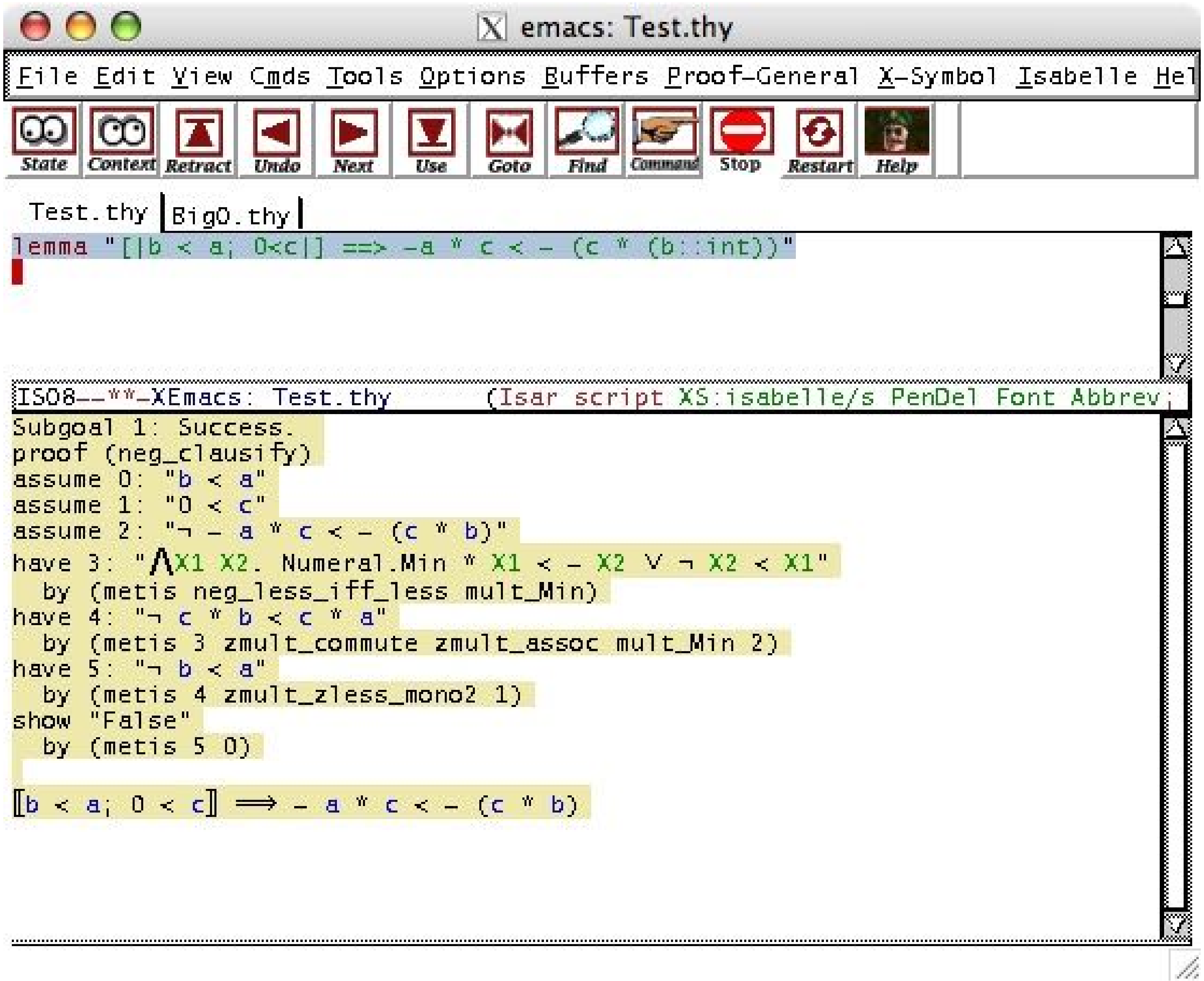
Working by February 2007



$$b < a \longrightarrow 0 < b \longrightarrow (-a) \times c < -(c \times b)$$

isn’t hard, but requires four separate facts

Also with single-step proofs



Others Take Over

Issues with the prototype

Unsound translations
(resulting in worthless “proofs")

Simplistic methods
(esp. relevance filtering)

Truly horrible code

The all-new sledgehammer

- A family of efficient, sophisticated and **sound** translations for monomorphic and polymorphic types
- An **ML based** relevance filter for premise selection
- Additional external provers, notably **SMT solvers** such as Z3
- ... justified by additional internal provers, including Isabelle's Z3

The work of Jasmin Blanchette, Sascha Böhme and Tobias Nipkow

Running three different theorem provers (E, SPASS and Vampire) each for **five seconds** solves as many problems as running the best theorem prover (Vampire) for **two full minutes**.

Higher-order superposition

- An effective alternative to translating λ-calculus into first-order logic
- a **sound and complete calculus** for higher-order logic with polymorphism, extensionality, Hilbert choice, and Henkin semantics
- And a **term ordering** to limit the search space
- And an implementation! **Zipperposition** outperforms all other higher-order theorem provers

The work of Bentkamp, Blanchette, Tourret, Vukmirovic

Giving back to the ATP community

(By verifying their theoretical canon)

A verified SAT solver framework with learn, forget, restart, and incrementality

A verified prover based on ordered resolution

Formalizing Bachmair and Ganzinger's ordered resolution prover

Formalized superposition

Impact

Synergy with structured proofs

Every line justified by sledgehammer!

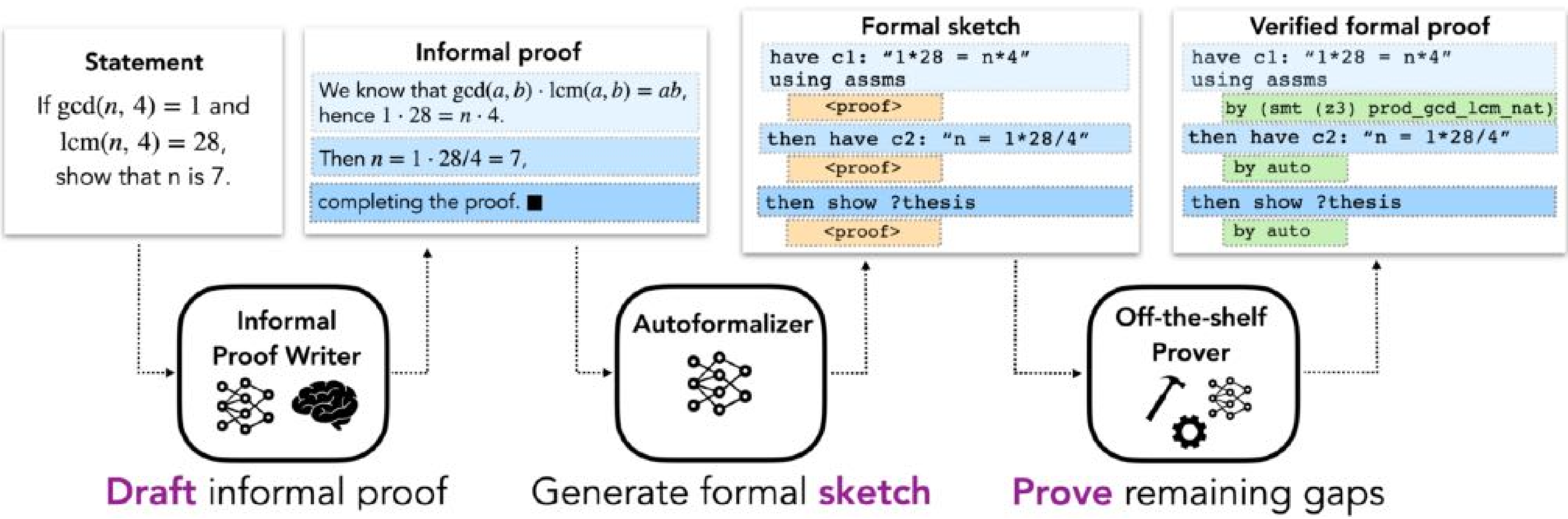
```
lemma "sqrt 2 ∉ ℚ"
proof
  assume "sqrt 2 ∈ ℚ"
  then obtain q::rat where "sqrt 2 = of_rat q"
    using Rats_cases by blast
  then have "q² = 2"
    by (metis abs_numeral of_rat_eq_iff of_rat_numeral_eq of_rat_power real_sqrt_abs
        real_sqrt_power)
  then obtain m n where "coprime m n" "q = of_int m / of_int n"
    by (metis Fract_of_int_quotient Rat_cases)
  then have "(rat_of_int m)² / (rat_of_int n)² = 2"
    by (metis ‹q² = 2› power_divide)
  then have 2: "(rat_of_int m)² = 2 * (rat_of_int n)²"
    by (metis div_by_0 nonzero_mult_div_cancel_right times_divide_eq_left zero_neq_numeral)
  then have "2 dvd m"
    by (metis (mono_tags, lifting) even_mult_iff even_numeral of_int_eq_iff of_int_mult
              of_int_numeral power2_eq_square)
  then have "2² dvd m²"
    using dvd_power_same by blast
  then have "2 dvd n"
    by (metis "2" even_mult_iff of_int_eq_iff of_int_mult of_int_numeral power2_eq_square
        zdvd_mono zero_neq_numeral)
  then show False
    using ‹coprime m n› ‹even m› by auto
qed
```

… hence, easier for beginners

- No more memorising lists of built-in facts
- No more learning obscure tactics for pushing symbols around
- The key skill: **thinking up intermediate goals**
- Given the proof structure, *Sledgehammer does the rest!*

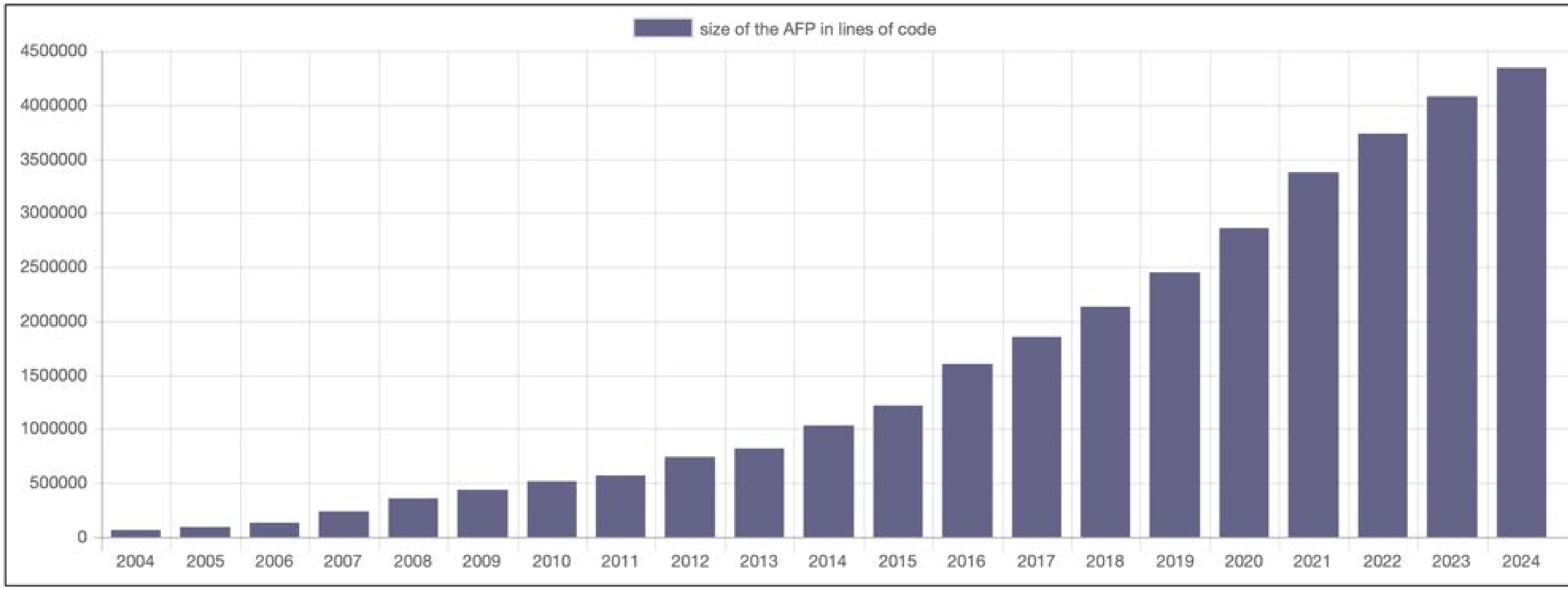
Turning English into proofs using AI

Draft, sketch and prove: Jiang et al.



Strong growth in lines of code

Isabelle's Archive of Formal Proofs



SH

SH mk II

New applications for ATPs themselves

A limitless supply of users with tough problems

Motivation for extensions such as types and polymorphism

Strong justification for automating higher-order logic, e.g. in CVC and E

And other hammers, notably for HOL, Coq and Lean (forthcoming)

Hopes for the future

- Strong support for problems involving **λ-binding**
- Genuine, powerful **higher-order reasoning**
- **Hints to users**, say about possibly missing assumptions or lemmas
- A truly effective and sound **integration with AI**

Acknowledgements

Claire Quigley: process management

Jia Meng: relevance, HO translations

Kong Woei Susanto: Metis

EPSRC Engineering and Physical Sciences Research Council

Project GR/S57198/01 *Automation for Interactive Proof*

Plus the group at TU Munich that created the second generation