Proof Assistants: From Symbolic Logic To Real Mathematics?

Lawrence C Paulson FRS

Inria Sophia-Antipolis, 18/5/2017

I. Formalised Mathematics

Computers and mathematical proof

- Appel and Haken’s 1976 proof of the Four Colour Theorem: a computer checked nearly 2000 cases
- Hales's 1998 proof of the Kepler Conjecture, on the optimal packing of spheres: also a huge case analysis
- McCune’s 1996 proof of the Robbins Conjecture using special software

Mathematicians hate such proofs!

Proof assistants in mathematics

- The Four Colour Theorem *checked in Coq*
- The Kepler Proof *checked using HOL Light and Isabelle*

Case analysis still required, but runs in a verified environment.

The mathematical reasoning also formally verified.

But are proof assistants ready for mathematical research?

What *are* they really?

Early proof assistants

Boyer/Moore (1971): functional programs
w

LCF (1978): functional programs in domain

LCF_LSM (1983): hardware verification
i

HOL (1988): functions and hardware in

LEGO (1991): calculus of constructions and other type theories

AUTOMATH (1968): mathematics in type theory, using "propositions as types"

Mizar (1973): mathematics in classical set theory

Most intended for *verification*, not mathematics. And using weird formalisms!

From verification to mathematics

- John Harrison (2000): formalised real analysis to verify *floating point algorithms* for sqrt, ln, exp [in HOL]

- Joe Hurd (2003): formalised measure and probability to verify *probabilistic algorithms* [in HOL]

- Sylvie Boldo (2013): verified a *numerical analysis program* for solving a wave equation, "covering all aspects from partial differential equations to actual numerical results" [in Coq]

Verifying maths for its own sake

- A formalisation of geometry and nonstandard analysis to check infinitesimal proofs in Newton's Principia (Fleuriot, 1998) [in Isabelle]
- Prime number theorem (Avigad; Harrison) [separate proofs in Isabelle and HOL Light]
- **Odd order theorem** (Gonthier et al.) [in Coq]
- Gödel's constructible universe and (both) incompleteness theorems [in Isabelle]

But why do maths by machine?

To *validate* questionable proofs

To *reveal* hidden assumptions

To *codify* mathematical knowledge

But the main reason is…

Mathematicians are fallible

Look at the footnotes on a **single page**
(118) of Jech's *The Axiom of Choice*

[1] The result of Problem 11 contradicts the results announced by Levy [1963b]. Unfortunately, the construction presented there cannot be completed.
[2] The transfer to ZF was also claimed by Marek [1966] but the outlined method appears to be unsatisfactory and has not been published.
[3] A contradicting result was announced and later withdrawn by Truss [1970].
[4] The example in Problem 22 is a counterexample to another condition of Mostowski, who conjectured its sufficiency and singled out this example as a test case.
[5] The independence result contradicts the claim of Felgner [1969] that the Cofinality Principle implies the Axiom of Choice. An error has been found by Morris (see Felgner's corrections to [1969]).

Mathematicians are fallible, II

"When the Germans were planning to publish Hilbert's collected papers …, they realized that they could not publish the papers in their original versions because they were full of errors, some of them quite serious. Thereupon they hired a young unemployed mathematician, Olga Taussky-Todd, to go over Hilbert's papers and correct all mistakes."

[Gian-Carlo Rota, *Indiscrete Thoughts*, p. 201]

"Olga laboured for three years."

2. Formalised Mathematics: Our Choices

The dimensions of formalised mathematics

Types? — and what sort of types?

What is 1/0?

Search and automation

Notation for terms and proofs

Type theory or set theory?

polymorphism — type classes

Simple types

HOL

Isabelle/HOL

predicate subtypes

PVS

Dependent types

Coq

AUTOMATH

Agda

Classical sets

Mizar

Isabelle/ZF

Type class polymorphism!

axiomatically define groups, rings, topological spaces, metric spaces and other type classes

prove that a type is in some class, inheriting its properties

… supporting uniform mathematical *notation*

But less flexible than dependent types — or classical sets!

…exchanging some flexibility for clarity

Definedness, or what is 1/0?

- *Don't care*: all terms denote *something*, and $1/0 = 1/0$. [HOL, Isabelle]
- *Dependent types*: to use x/y, must prove $y \neq 0$ (but does the value of x/y depend on this proof?) [Coq, PVS]
- *Free logic*: a formalism where defined[x/y] can be expressed. So $x/0 = x/0$ is false. But is $x/0 \neq x/0$ true? [IMPS]

Search and automation

decision procedures: linear arithmetic, elementary set theory, Gröbner basis methods

heuristic methods: obvious rewriting and chaining steps, e.g. $x+0 = x$

fast, predictable, powerful, but of limited scope

natural, flexible but ad-hoc; changes can break proofs

Syntax, or the legibility problem

Mathematical notation is elegant but ambiguous!

$$f(x) \quad f(X) \quad f^{-1}[X]$$

$$x^{-1}y \quad f^{-1}(x) \quad \sin^{-1}(x) \quad \sin^2(x)$$

$$xy \quad x \cdot y \quad \frac{d^2 f}{dx}$$

`Machine notations are merely hideous`

Example: a HOL Light lemma

```
let SIMPLE_PATH_SHIFTPATH = prove
 (`!g a. simple_path g /\ pathfinish g = pathstart g /\
         a IN interval[vec 0,vec 1]
         ==> simple_path(shiftpath a g)`,
  REPEAT GEN_TAC THEN REWRITE_TAC[simple_path] THEN
  MATCH_MP_TAC(TAUT
   `(a /\ c /\ d ==> e) /\ (b /\ c /\ d ==> f)
    ==>  (a /\ b) /\ c /\ d ==> e /\ f`) THEN
  CONJ_TAC THENL [MESON_TAC[PATH_SHIFTPATH]; ALL_TAC] THEN
  REWRITE_TAC[simple_path; shiftpath; IN_INTERVAL_1; DROP_VEC;
              DROP_ADD; DROP_SUB] THEN
  REPEAT GEN_TAC THEN DISCH_THEN(CONJUNCTS_THEN2 MP_TAC ASSUME_TAC) THEN
  ONCE_REWRITE_TAC[TAUT `a /\ b /\ c ==> d <=> c ==> a /\ b ==> d`] THEN
  STRIP_TAC THEN REPEAT GEN_TAC THEN
  REPEAT(COND_CASES_TAC THEN ASM_REWRITE_TAC[]) THEN
  DISCH_THEN(fun th -> FIRST_X_ASSUM(MP_TAC o C MATCH_MP th)) THEN
  REPEAT(POP_ASSUM MP_TAC) THEN
  REWRITE_TAC[DROP_ADD; DROP_SUB; DROP_VEC; GSYM DROP_EQ] THEN
  REAL_ARITH_TAC);;
```

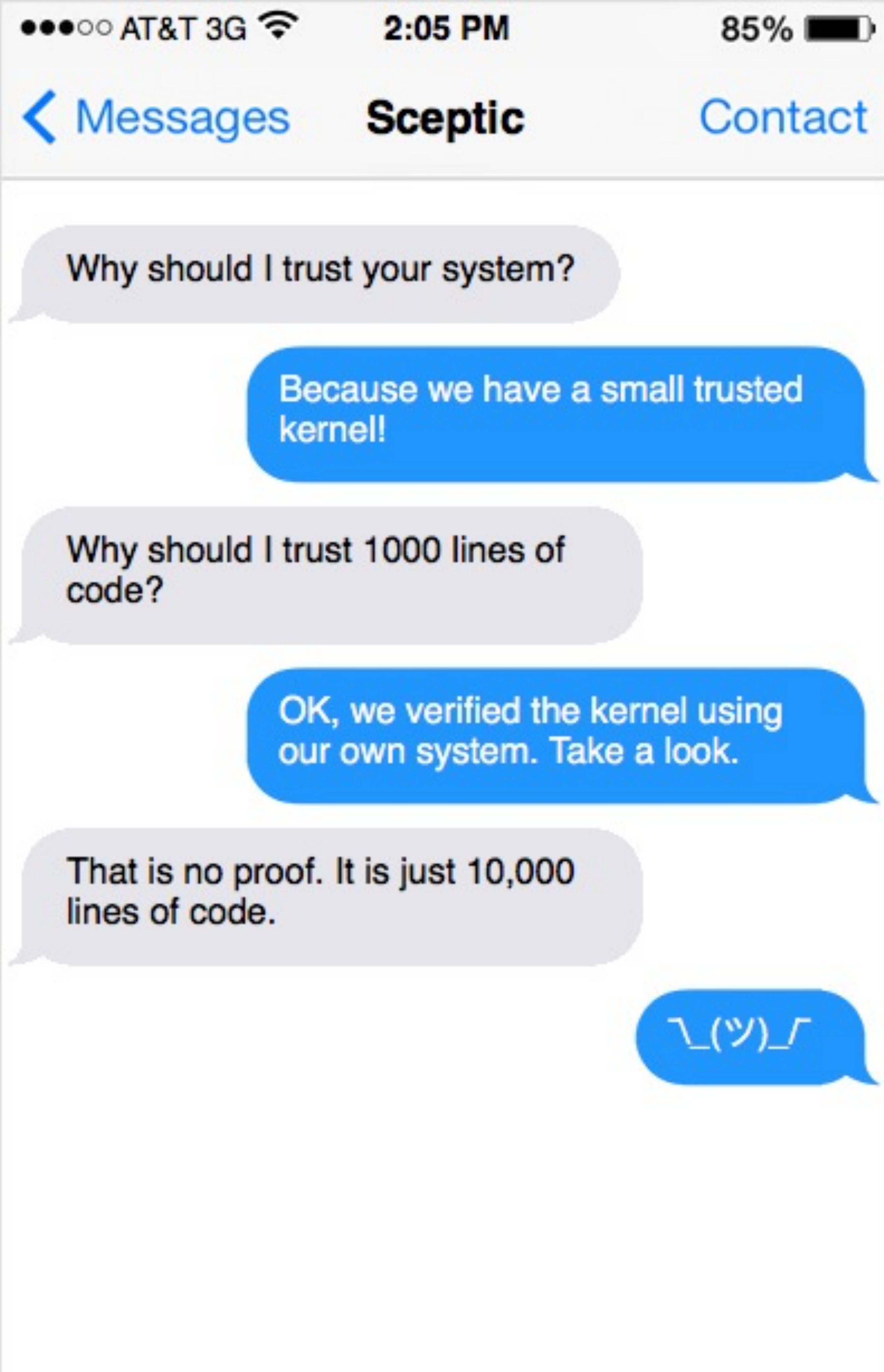
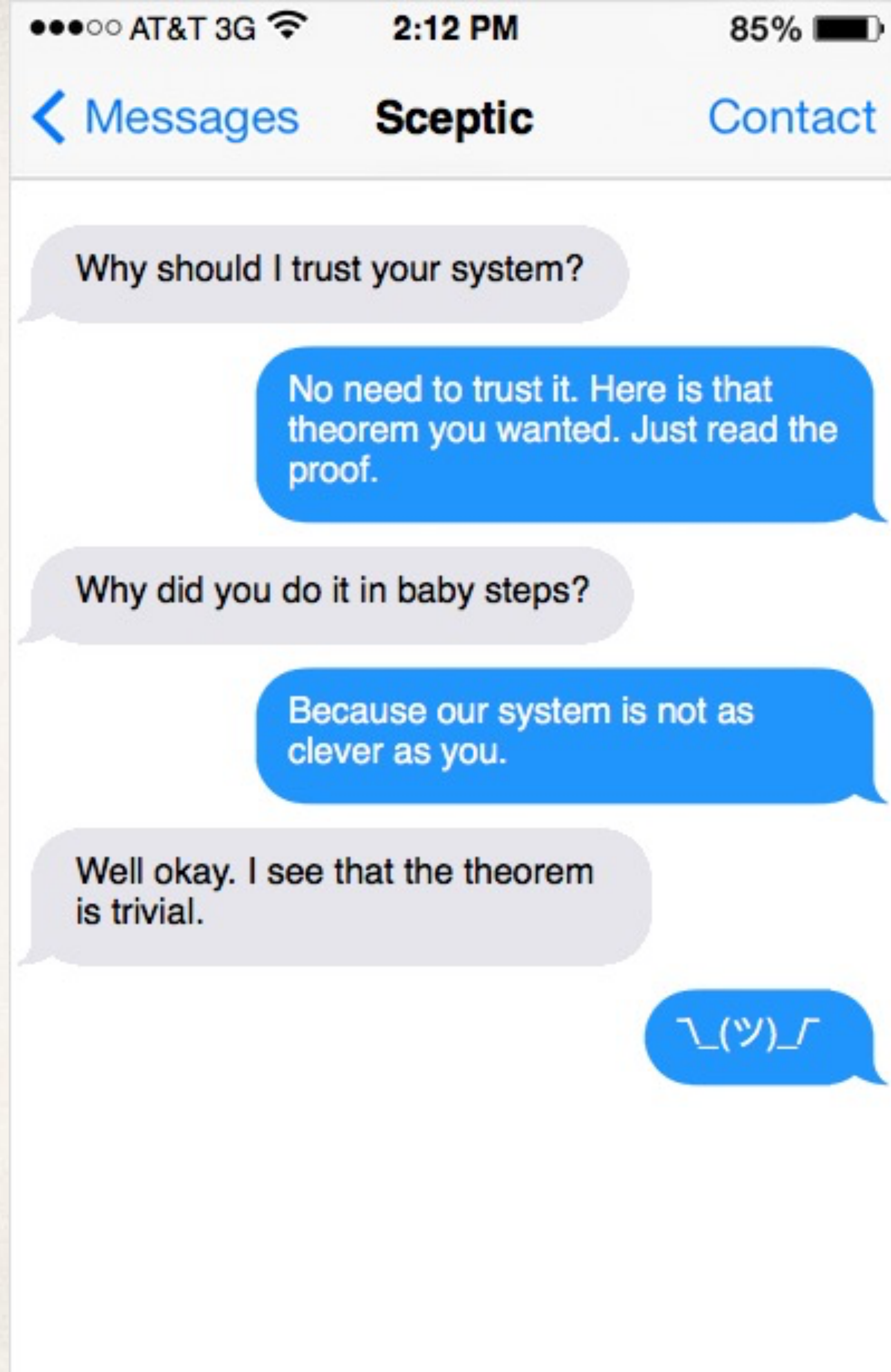
Some proofs are 50× longer than this one!

The same, as a *structured* proof

```
lemma simple_path_shiftpath:
  assumes "simple_path g" "pathfinish g = pathstart g" and a: "0 ≤ a" "a ≤ 1"
    shows "simple_path (shiftpath a g)"
  unfolding simple_path_def
proof (intro conjI impI ballI)
  show "path (shiftpath a g)"
    by (simp add: assms path_shiftpath simple_path_imp_path)
  have *: "⋀x y. ⟦g x = g y; x ∈ {0..1}; y ∈ {0..1}⟧ ⟹ x = y ∨ x = 0 ∧ y = 1 ∨ x = 1 ∧ y = 0"
    using assms by (simp add:  simple_path_def)
  show "x = y ∨ x = 0 ∧ y = 1 ∨ x = 1 ∧ y = 0"
    if "x ∈ {0..1}" "y ∈ {0..1}" "shiftpath a g x = shiftpath a g y" for x y
    using that a unfolding shiftpath_def
    apply (simp add: split: if_split_asm)
      apply (drule *; auto)+
    done
qed
```

Structured proofs are necessary!

- For *maintenance* (fixing proofs when they break)
- For *reuse* and (one day) *translation* to other systems
- Legibility builds *confidence* in our verification tools, **especially for sceptical mathematicians**.


●●●○○ AT&T 3G
2:05 PM
85%
Messages
Sceptic
Contact
Why should I trust your system?
Because we have a small trusted kernel!
Why should I trust 1000 lines of code?
OK, we verified the kernel using our own system. Take a look.
That is no proof. It is just 10,000 lines of code.
¯_(ツ)_/¯

●●●○○ AT&T 3G
2:12 PM
85%
Messages
Sceptic
Contact
Why should I trust your system?
No need to trust it. Here is that theorem you wanted. Just read the proof.
Why did you do it in baby steps?
Because our system is not as clever as you.
Well okay. I see that the theorem is trivial.
¯_(ツ)_/¯

3. Are Proof Assistants Ready for Mathematics?

Robust and mature architectures

- *soundness*: all proof steps checked by a small kernel (the "LCF approach")
- *automation*: rewriting, logical reasoning, computer algebra techniques, decision procedures
- *scalability*: large specification hierarchies handled
- *expressive formalisms* covering at least applied maths

Comprehensive libraries

Mathematical Components (Coq):
everything from lists to advanced algebra

Coquelicot: real analysis including limits,
derivatives, integrals, power series

Multivariate Analysis (HOL Light): 300K lines on
homotopic paths, complex analysis, polytopes

Archive of Formal Proofs (Isabelle): 1.6M lines on
numerous topics, not only mathematics

But …

Is formalised maths even possible?

Whitehead and Russell needed 362 pages to prove 1+1=2!

We have better formal systems than theirs.

Gödel proved that all reasonable formal systems must be incomplete!

We don't need a *universal* formal system.

Church proved that first-order logic is undecidable!

We use automation to **assist** people, not to **replace** them.

The real problem areas

- No library covers undergraduate mathematics.
- Formal proofs are unreadable and don't link to any real mathematical text.
- Libraries are difficult to search, especially for *concepts*.
- Automation falls *far short* of mathematical intuition.

What could we aim for?

Natural language search

Codified textbooks

Similarity-based search (*proof idioms*)

Verified computer algebra tools

Where do we go now?

Grow our libraries

Mine libraries for re-use

Keep building tools

Work with mathematicians!