Mechanising Mathematical Proof: A Progress Report

Rob Arthan

Lemma 1 Ltd. / DCS Queen Mary, London

3rd September, 2008

An A – Z of Mathematical Opinions

"I will say nothing, for example, about the great events in the area between logic and computing"

Sir Michael Atiyah. *Mathematics in the 20th Century.*

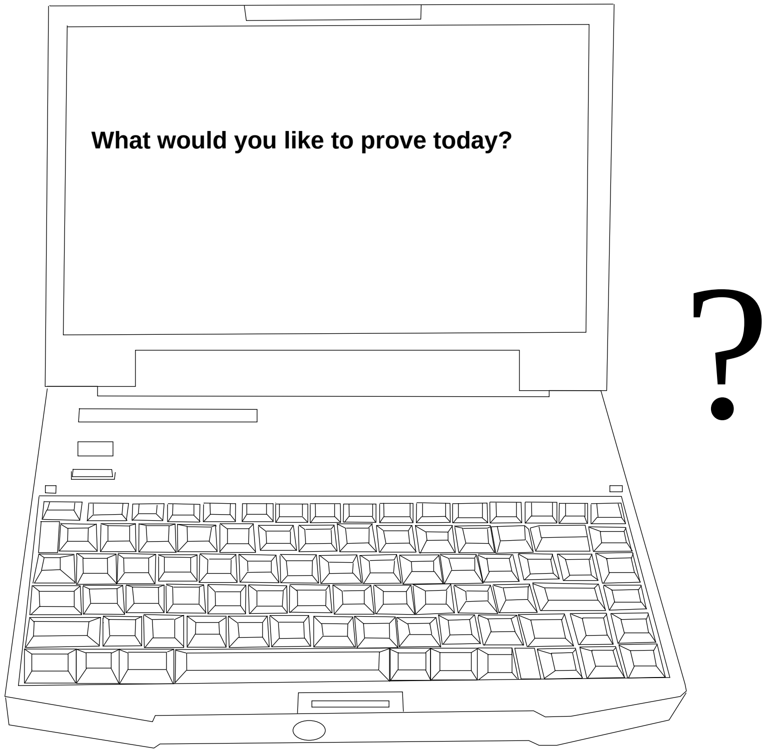
...

"But the MOST significant contribution by 20th-Century human mathematicians and computer scientists was the creation of COMPUTER ALGEBRA."

Doron Zeilberger. *Opinion 47.*

A Hypothetical Question

What would Newton or the Bernoullis or Gauss or Delaunay or Hilbert or ...your list goes here ... have done with one of these:



$1 + 1 =$ hmmm?

"The properties of $\dot{2}$ are largely analogous to those of 1, while the properties of 2_r are more analogous to those of 2."

A.N. Whitehead and B. Russell *Principia Mathematica *56* **p. 375** 1910

What about properties of e or π or $\zeta(1)$ or $\pi_{13}(S^5)$ or $\ldots$?

- The immense symbolic processing tasks defy human capabilities.
- Machines don't know what symbols to process.
- Synergy between human and computer is what's called for.

Why bother?

- Mathematical arguments do contain errors!
- How many in the classification of finite simple groups?
- Even Aigner & Ziegler's *"Proofs from THE BOOK"* (ed. 1).

  See Bill Casselman. *The Difficulties of Kissing in Three Dimensions.*
- Controversies about use of computers:
  - 4 colour theorem
  - Kepler sphere packing conjecture

and, of course, because it's there!

The Kepler Conjecture

> The density of a packing of congruent spheres in three dimensions is never greater than $\pi/\sqrt{18}$.

- Asserted by Kepler in 1611.
- 1998 proof by Hales reduces a nonlinear optimisation problem in an infinite number of variables to a problem in a finite number of variables (≈ 150). This in turn reduces to:
  - Enumerating graphs that give possible local geometry.
  - A large set of linear programming problems.

  The enumeration and the linear programming problems are then solved by computer.

Referees' Verdict

Published subject to a disclaimer.

"The news from the referees is bad .... They have not been able to certify ... the proof, and will not be able to certify it ..., because they have run out of energy to devote to the problem."

"[The chief of the 12 referees] thinks that this situation will occur more and more often in mathematics. He says it is similar to the situation in experimental science — other scientists acting as referees can't certify the correctness of an experiment, they can only subject the paper to consistency checks. He thinks that the mathematical community will have to get used to this state of affairs."

Robert MacPherson. Editor of *Annals of Mathematics*

At least one member of the community disagrees ...

Hales's Flyspeck Project

- Proposed approach: a complete machine-checked formal proof — not a verification of the programs.
- Estimated 20 person/year collaborative work in progress.
- Enumeration of tame planar graphs has been verified

  Tobias Nipkow, Gertrud Bauer, TUM.
- Tools to handle the linear programming problems developed

  Steve Obua, TUM.

But where does the assurance come from?

A Deductive System

Judgments: $m \mathsf{D} n$, $m, n \in \mathbb{Z}$ (intended meaning: m divides n)

Rules:

$$\frac{}{m \mathsf{D} m} : \text{axiom}, m \neq 0$$

$$\frac{m \mathsf{D} n}{m \mathsf{D} -n} : - \qquad \frac{m \mathsf{D} n_1 \quad m \mathsf{D} n_2}{m \mathsf{D} n_1 + n_2} : +$$

A deduction:

$$\frac{\dfrac{}{1 \mathsf{D} 1} : \text{axiom} \qquad \dfrac{\dfrac{}{1 \mathsf{D} 1} : \text{axiom}}{1 \mathsf{D} -1} : -}{1 \mathsf{D} 0} : +$$

- Robin Milner's LCF method implements a deductive system as a data type.
- Strongly typed programming language enforces the rules.

The Deductive System in ML

Demo 1

```
local
  datatype THEOREM = D of (int * int);
in
  type THEOREM = THEOREM;
  infix D; infix ++;
  exception NOT_ALLOWED;

  fun axiom m = if m <> 0 then m D m else raise NOT_ALLOWED;
  fun -- (m D n) = m D ~n;
  fun (m1 D n1) ++ (m2 D n2) =
    if m1 = m2 then m1 D (n1 + n2) else raise NOT_ALLOWED;
end;
```

A Decision Procedure

Demo 1 concluded.

Given `m` and `n`, the function **`decide`** tries to prove that `m` divides `n`:

```
fun decide m n =
  if n < 0 then --(decide m (~n))
  else if n <= m then axiom m
  else decide m (n-m) ++ axiom m;
```

Logical kernel will not allow invalid deductions:

```
> decide 2 6;
val it = 2 D 6 : THEOREM
> decide 2 7;
val it = 2 D 8 : THEOREM
> decide 0 0;
Exception- NOT_ALLOWED raised
```

A Real System: ProofPower-HOL

Demo 2.

- Member of the HOL family implemented for industrial use.

  Cf. Classic HOL (Gordon), HOL IV (Slind, Norrish), HOL Light (Harrison).

- Expressions and predicates represented by the type $TERM$, entered using "Quine corners": $\ulcorner 1 + 2 \urcorner$, $\ulcorner 1 = 2 \urcorner$

- Abstract data type of theorems is the type THM. Printed with a turnstile: $\vdash \neg 1 = 2$.

- Extensive facilities for programming with syntax.

- Maintains a database of theories containing specifications (i.e., defining properties of types and constants) and theorems.

- Powerful higher-level tools for automated and interactive proof.

Characteristics of the LCF Approach

- Logical kernel is small and simple to check
  — possibly even formally verifiable.
- Derived rules may fail to produce desired output but can't produce unsound results.
- Can safely code very complex algorithms.
- Potential for cross-checking using alternative compilers and/or implementations.
- There are other complementary approaches.

Other Real Systems

- Many systems around, LCF style and other approaches.
- Deductive system usually mathematical foundation system, e.g.,
  - Set theory — Mizar, Isabelle-ZF
  - Polymorphic type theory — HOL family, Coq, PVS
  - Theory of recursive functions — NQTHM, ACL2
- Many systems designed for computer science applications, e.g., program verification, but general purpose.
- Libraries available, but often application-oriented.
- Mizar is one system with a very extensive mathematical library.

But can you prove real theorems?

Some Achievements

A personal selection — no warranty offered or implied!

- Mizar Mathematical Library. Trybulec *et al.* 1970s – present
- Calculus. Harrison, Gottliebsen, Arthan, *et al.* 1990s – present
- Nonstandard analysis. Fleuriot, 1996 – present
- Gödel's incompleteness theorem. Shankar, 1994
- Sylow's theorem. Kammüller. 1997.
- Prime Number Theorem. Avigad *et al.*, 2004, Harrison, 2008
- Jordan Curve Theorem. Hales, 2005
- Brouwer fixed point theorem. Harrison, 2005
- 4 Colour Theorem. Gonthier, 2005

Calculus In ProofPower-HOL 1

- Write $(f\ \ Deriv\ \ c)\ x$ to mean function f has derivative c at x (i.e., $df/dx = c$ or $f'(x) = c$ in the usual vernacular).

$$\$\boldsymbol{Deriv} : (\mathbb{R} \to \mathbb{R}) \to \mathbb{R} \to \mathbb{R} \to BOOL$$

$$\begin{array}{l} \forall f\ \ c\ \ x\bullet\ (f\ \ Deriv\ \ c)\ x \\ \Leftrightarrow\quad \forall e\bullet\ 0. < e \Rightarrow \exists d\bullet\ 0. < d\ \wedge \\ \qquad\quad \forall y\bullet\ Abs(y{-}x) < d\ \wedge\ \neg y{=}x \Rightarrow Abs((f\ \ y{-}f\ \ x)/(y{-}x)\ -\ c) < e \end{array}$$

- This definition is trivially consistent.
- All the usual theorems: product rule, chain rule, Rolle, IVT, MVT, ...

Calculus In ProofPower-HOL 2

- Define the exponential function by the differential equation:

$$\mathbf{\mathit{Exp}} : \mathbb{R} \rightarrow \mathbb{R}$$

$$\mathit{Exp}\ \mathit{0.} = \mathit{1.} \wedge (\forall x \bullet\ (\mathit{Exp}\ \mathit{Deriv}\ \mathit{Exp}\ x)\ x)$$

- Prove the consistency via theory of power series and differentiation of limits.
- Logarithm defined as left inverse of exponential.
- All the usual basic theorems, e.g.,

$$\vdash \forall\ x \bullet\ \mathit{0.} < x \Rightarrow (\mathit{Log}\ \mathit{Deriv}\ x^{\ -\mathit{1}})\ x : \mathit{THM}$$

Calculus In ProofPower-HOL 3

- Integration via the Kurzweil-Henstock gauge integral. FTC, areas, ...
- A theorem of Minkowski:

$$
\begin{array}{l}
\vdash \forall A\ a\bullet \\
\quad A \in \mathit{Convex} \wedge A \in \mathit{Bounded} \wedge \neg A = \{\} \\
\wedge\ (\forall x\ y\bullet\ (x,\ y) \in A \Rightarrow (\sim x,\ \sim y) \in A) \\
\wedge\ A\ \mathit{Area}\ a \wedge a > \mathit{4.} \\
\Rightarrow \exists i\ j : \mathbb{Z}\bullet\ \ (\mathbb{ZR}\ i,\ \mathbb{ZR}\ j) \in A \wedge \neg(\mathbb{ZR}\ i,\ \mathbb{ZR}\ j) = (\mathit{0.},\ \mathit{0.})
\end{array}
$$

- de Bruijn factor. Formal / Informal. Typically 0.5 – 5?
- See Freek Wiedijk's web site `http://www.cs.ru.nl/~freek/`

Example: A Combinatorics Problem

Demo 2 concluded.

- $(m+n)^m$ should give the number of ways of drawing m samples with replacement out of a set of $m+n$ elements.
- *DistinctSamples n m* should give the number of ways of drawing m samples without replacement out of a set of $m+n$ elements.

$$\boldsymbol{DistinctSamples} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$

$$\begin{array}{l} \quad (\forall n \bullet\ \mathit{DistinctSamples}\ n\ \mathit{0} = \mathit{1}) \\ \land\ (\forall m\ n \bullet \\ \quad \mathit{DistinctSamples}\ n\ (m{+}\mathit{1}) = (n{+}m{+}\mathit{1}) * \mathit{DistinctSamples}\ n\ m) \end{array}$$

- Plan: give high assurance solution to a combinatorics problem by:
  - 1) Proving that these functions do give the desired results.
  - 2) Symbolically executing them inside the theorem prover.

Final Remarks

- Machine-checked proof has a part to play.
- Should the need to compute make maths an empirical science?
- Technology can help: critical bugs are not inevitable!
- Formalisation need not lead to combinatorial explosion.
- Lots of fascinating work to do.

Thank you!

Links

- For ProofPower
  `http://www.lemma-one.com/ProofPower/index/index.html`
- Other implementations of HOL:
  `http://hol.sourceforge.net/`
  `http://www.cl.cam.ac.uk/research/hvg/Isabelle/`
  `http://www.cl.cam.ac.uk/~jrh13/hol-light/`
- The Flyspeck project:
  `http://code.google.com/p/flyspeck/`
- Freek Wiedijk's progress report on 100 theorems:
  `http://www.cs.ru.nl/~freek/100/index.html`
- The Mizar project:
  `http://mizar.uwb.edu.pl/`

The following is the Standard ML source of the demos:

Text dumped to file 64dedsys.ML

```
local
  datatype THEOREM = D of (int * int);
in
  type THEOREM = THEOREM;
  infix D; infix ++;
  exception NOT_ALLOWED;

  fun axiom m = if m <> 0 then m D m else raise NOT_ALLOWED;

  fun -- (m D n) = m D ~n;

  fun (m1 D n1) ++ (m2 D n2) =
    if m1 = m2 then m1 D (n1 + n2) else raise NOT_ALLOWED;
end;
```

Text dumped to file 64s1.ML

```
1+2;
fun fact n = if n <= 0 then 1 else n * fact(n-1);
fact 40;
map fact [1, 2, 3, 4, 5, 6, 7];
use "64dedsys.ML";
axiom 1;
1 D 1;
3 D 5;
val thm1 = axiom 1;
val thm2 = -- (axiom 1);
val thm3 = thm1 ++ thm2;
fun decide m n =
  if n < 0 then --(decide m (~n))
  else if n <= m then axiom m
  else decide m (n-m) ++ axiom m;
```

```
decide 1 1;
decide 1 ~1;
decide 2 6;
decide 13 1001;
decide 2 7;
decide 0 0;
decide ~1 1;
```

Text dumped to file 64s2.ML

```
val tm1 = ⌜0 * 1⌝;
val ty1 = type_of tm1;
val (f1, args1) = strip_app tm1;
val ty_f1 = type_of f1;
val thm1 = get_spec f1;
val thm2 = list_∀_elim [ ⌜1⌝, ⌜1⌝ ] thm1;
val thm3 = ∧_left_elim thm2;
val thm4 = rewrite_conv[] ⌜(0+1+2+3)*(3+2+1+0)⌝;
set_goal([], ⌜∀m i j : ℕ• m^(i + j) = m^i * m^j⌝);
a( REPEAT strip_tac );
a(induction_tac ⌜j⌝);
a(rewrite_tac[ ℕ_exp_def ]);
a(asm_rewrite_tac[ plus_assoc_thm1 , ℕ_exp_def ]);
a( PC_T1 "lin_arith" prove_tac [] );
val thm5 = pop_thm();
val tm2 = ⌜(λx•x + 1)⌝;
val ty2 = type_of tm2;
val thm6 = rewrite_conv[]⌜(λx•x + 1) 2⌝;
val tm3 = ⌜[x; y; 1]⌝;
val ty3 = type_of tm3;
val (f3, args3) = strip_app ⌜[x; y; z]⌝;
val ty_f3 = type_of f3;
val thm7 = rewrite_conv[nth_def] ⌜Nth [x; y; z] 2⌝;
val thm8 = get_spec⌜Map⌝;
val ty4 = type_of⌜Map⌝;
```

```
val thm9 = rewrite_conv[map_def] ⌜Map (λx● x + 1) [1; 2; 3]⌝;
val tm4 = ⌜2 ∈ {1; 2; 3}⌝;
val (f4, args4) = strip_app tm4;
val ty_f4 = type_of f4;
val thm4 = get_spec f4;
val thm10 = prove_rule[] tm4;
val thm11 = pc_rule1 "sets_ext1"prove_rule[elems_def]
    ⌜Elems [1; 2; 3] = {1; 2; 3}⌝;
distinct_samples_def;
val thm12 = (print "\n\n"; conv_rule(MAP_C plus_conv)(pure_rewrite_conv
    [distinct_samples_def,
    pc_rule1 "lin_arith" prove_rule[]⌜∀m●m*1 = m⌝,
    plus_assoc_thm1]
    ⌜DistinctSamples 10 (0+1+1+1)⌝));
val thm13 = rewrite_rule[] thm12;
distinct_samples_finite_size_thm;
samples_finite_size_thm;
(* and then .... *)
use_file"64demo2proof.ML";
val thm14 = pop_thm();
```

Text dumped to file 64demo2proof.ML

```
set_goal([], ⌜ (* Try not to show from HERE ... *)
    let S = {L | Elems L ⊆ {i | 1 ≤ i ∧ i ≤ 365} ∧ # L = 23}
    in let  X = {L | L ∈ S ∧ ¬L ∈ Distinct}
    in  S ∈ Finite
    ∧  ¬#S = 0
    ∧  X ⊆ S
    ∧  #X / #S > 1/2
⌝);
a(rewrite_tac[let_def]);
a(strip_asm_tac(∀_elim⌜365⌝ range_finite_size_thm1));
a(lemma_tac⌜23 ≤ #{i | 1 ≤ i ∧ i ≤ 365}⌝ THEN1 asm_rewrite_tac[]);
a(all_fc_tac[distinct_samples_finite_size_thm]);
```

```
a(all_fc_tac[samples_finite_size_thm]);
a(REPEAT_N 2 (POP_ASM_T(ante_tac o ∀_elim⌜23⌝)) );
a(POP_ASM_T discard_tac THEN strip_tac THEN strip_tac);
a(pure_asm_rewrite_tac[conv_rule(ONCE_MAP_C eq_sym_conv)ℕℝ_one_one_thm,
    ℕℝ_ℕ_exp_thm,
    ℝ_frac_def]);
a(asm_tac(rewrite_conv[]⌜ℕℝ 365 ^ 23⌝));
a(PC_T1"predicates" rewrite_tac[] THEN strip_tac
    THEN1 (pure_asm_rewrite_tac[ℕℝ_one_one_thm]
        THEN PC_T1 "lin_arith" prove_tac[]));
a(REPEAT strip_tac THEN1 PC_T1 "sets_ext1" prove_tac[]);
a(pure_rewrite_tac[ℕℝ_one_one_thm]);
a(LEMMA_T⌜{L | (Elems L ⊆ {i | 1 ≤ i ∧ i ≤ 365} ∧ # L = 23)
        ∧ ¬L ∈ Distinct}
 = {L | Elems L ⊆ {i | 1 ≤ i ∧ i ≤ 365} ∧ # L = 23} \
    {L | Elems L ⊆ {i | 1 ≤ i ∧ i ≤ 365} ∧ # L = 23
        ∧ L ∈ Distinct}⌝
    pure_rewrite_thm_tac
    THEN1 PC_T1 "sets_ext1" prove_tac[]);
a(lemma_tac⌜{L | Elems L ⊆ {i | 1 ≤ i ∧ i ≤ 365} ∧ # L = 23 ∧
        L ∈ Distinct} ⊆
    {L | Elems L ⊆ {i | 1 ≤ i ∧ i ≤ 365} ∧ # L = 23}⌝
    THEN1 PC_T1 "sets_ext1" prove_tac[]);
a(ALL_FC_T (MAP_EVERY (ante_tac o
    once_rewrite_rule[conv_rule(ONCE_MAP_C eq_sym_conv)ℕℝ_one_one_thm]))
    [size_⊆_diff_thm]);
a(pure_rewrite_tac[ℕℝ_plus_homomorphism_thm,
    pc_rule1 "ℝ_lin_arith" prove_rule[]
    ⌜∀a b c:ℝ•a = b + c ⇔ b = a − c⌝]
    THEN STRIP_T pure_rewrite_thm_tac);
a(LIST_DROP_NTH_ASM_T [3, 6, 9] pure_rewrite_tac);
a(LEMMA_T ⌜∀a b c d:ℝ•ℕℝ 0 < b ∧ ℕℝ 0 < d ∧ a*d < b*c ⇒ a/b < c/d⌝
    bc_thm_tac
    THEN1 (REPEAT strip_tac
        THEN1 ALL_FC_T1 fc_⇔_canon asm_rewrite_tac[ℝ_cross_mult_less_thm]));
a(pure_asm_rewrite_tac[ℕℝ_ℕ_exp_thm,
```

```
    pc_rule1 "ℝ_lin_arith" prove_rule[]
    ⌜∀a b c:ℝ•a < ℕℝ 2 * (b − c) ⇔ a + ℕℝ 2 * c < ℕℝ 2 * b⌝,
    REPEAT_C (once_rewrite_conv[distinct_samples_rw_thm]
        THEN_C rewrite_conv[])
    ⌜DistinctSamples (365−23) 23⌝]);
a(pure_rewrite_tac[ℕℝ_plus_homomorphism_thm1,
    ℕℝ_times_homomorphism_thm1,
    ℕℝ_less_thm]);
a(rewrite_tac[]) (* ... down to HERE! *);
```