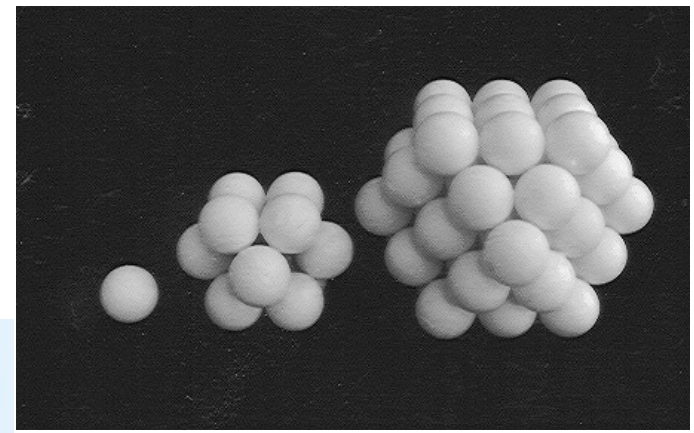
Lessons learned from the Flyspeck Project

Thomas C. Hales

February 23, 2011

- The Kepler conjecture asserts that the densest packing of congruent balls in $\mathbb{R}^3$ is achieved by the familiar "cannonball" arrangement.
- The Kepler Conjecture was formulated in the booklet "The six-cornered snowflake," presented as a gift on New Year's day 1611 to Kepler's patron Lord Wacker von Wackenfels.

Kepler asks why a snowflake has six sides. This leads to honeycombs, pomegranates, and then sphere packings.



- The first proof was presented (by Ferguson and H. in 1998) and published in 2006.
- A project called Flyspeck seeks to give a formal proof of the theorem, which involves a computer verification of every single logical inference in the proof, all the way back to the fundamental axioms of mathematics.
- FLYSPECK comes from F.*P.*K, for the Formal Proof of the Kepler Conjecture.
- The Flyspeck project is about 80% complete.

The Flyspeck project is about 80% complete. The project has five parts:

1. The text part of the proof is contained in an unpublished manuscript "Dense Sphere Packings: a formal blueprint." Formalization is being done by a team of researchers.
2. The first computer program (plane graph generation) was formalized by G. Bauer and T. Nipkow.
3. The second computer program (linear programming) is nearly formalized by S. Obua and A. Solovyev.
4. The third computer program (nonlinear inequality proving) is work in progress (Solovyev). I will describe progress on this part today.
5. The integration of text part of the proof with computer parts.

The HOL Light proof assistant (J. Harrison)

1. A member of the HOL family of proof assistants.
2. It has an unusually small kernel (about 400 lines of code) which has been self-verified.
3. The type system is simple.
4. There is no layer between HOL Light and Ocaml. To write proofs is to program in Ocaml. Anyone can (and does) contribute new tactics at any time.
5. There is no reflection for verified algorithms. Verification occurs with each separate execution of an algorithm.
6. It has a large collection of theorems in elementary mathematics: point-set topology, real and complex analysis.

First lesson: international collaboration works!

Adams, Mark,
Bauer, Gertrud,
Dang Tat Dat,
Hales, Thomas C.,
Harrison, John,
Hoang Le Truong,
Nipkow, Tobias
Nguyen Duc Tam
Nguyen Tat Thang,
Nguyen Quang Truong,
Obua, Steven
Rute, Jason,
Solovyev, Alexey,
Ta Thi Hoi An,
Thi Trieu Diep,
Tran Nam Trung,
Vu Khac Ky,
Vu Thanh,
Vuong Anh Quyen

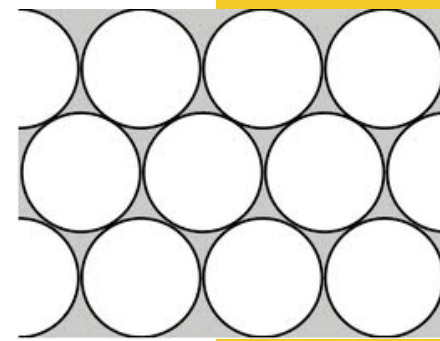
INTERNATIONAL WORKSHOP
on Formal Proofs and Flyspeck Project
Hanoi, June 8- July 31, 2009

Further Lessons learned

1. Follow good software development practices for a medium sized project, including version control, a stable module system, careful control of mutable state, a manager (Mark Adams).
2. Structure the project around a clean mathematical text.
3. Search for theorem names is a major difficulty. There are over 15,000 theorems in HOL Light + FLYSPECK and many dozens of specialized tactics. Better tools are needed.
4. Keep each lemma short and crisp: productivity plummets once lemmas reach a certain level of complexity.

Further lessons. Formal proofs should not merely replicate a body of mathematics. They should transform and improve proofs. Attention to the formal proof of the Kepler conjecture has led to a proof last year of two other long-standing conjectures.

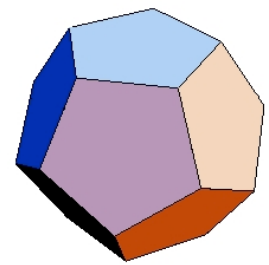
Conjecture 1 (Fejes Tóth's full contact conjecture (1969), Theorem (H., 2011))**.** *In 3-space a packing of equal balls such that each enclosed ball is touched by 12 others consists of hexagonal layers.*

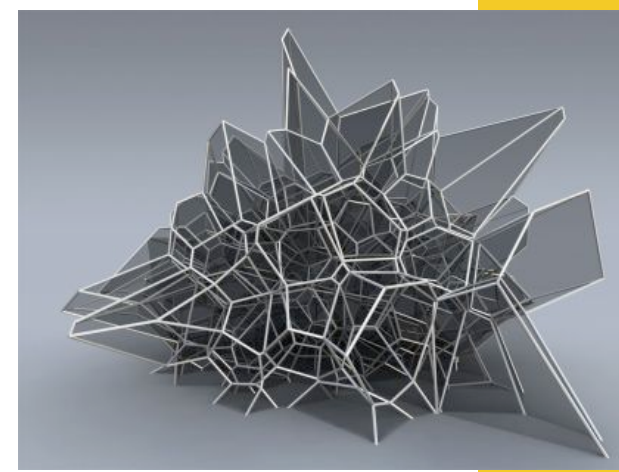


(The corresponding problem in the plane is trivial. If each unit disk in the plane touches 6 others then it must be the regular hexagonal packing of disks.)

Conjecture 2 (K. Bezdek's strong dodecahedral conjecture (2000), Theorem (H., 2011))**.** *In every packing of congruent balls in* $\mathbb{R}^3$*, the surface area of every Voronoi cell is at least that of the (circumscribing) regular dodecahedron.*

(The strong dodecahedral conjecture implies the weak dodecahedral conjecture, which was proved by S. McLaughlin in 1998, and published in 2010.)





Lesson about code verification: The surest way to guarantee that a piece of code will not execute a bug is to delete the code.

1. Nipkow used this principle repeatedly in the verification of the plane graph generator: De-optimize, de-optimize!
2. The total amount of code for the computer part of the Kepler conjecture has been reduced from about 187,000 lines to fewer than 10,000 lines.
3. At the same time, execution times have gone from over 3 months to under 1 day.

A variant of the same lession: the fastest way to formalize a difficult theorem is to simplify the proof.

An example of proof refactoring - Perimeter estimate. The perimeter of a convex spherical polygon on the unit sphere is at most 2π.

Proof. Move one vertex at a time, increasing perimeter, until all vertices line on the equator. The equator has 2π. □

Proof. Take the polar polygon. Perimeter and area are dual concepts under polarity. The upper bound 2π on the perimeter is dual to the the lower bound 0 on area. □

- Trade-off: deformations arguments are very messy in HOL Light (especially without compactness), but the second proof introduces a new concept.
- This one lemma accounts for 10% of what remains in the text formalization.

Final lesson: the Flyspeck project was not nearly ambitious enough.

1. Existing tools have been adequate (but less than optimal) for the formalization of the 300-page text part of the proof.

2. We are still thinking at the wrong level of magnitude when it comes to formalization of mathematics. (We thought a lecture class with 1000 students was large until Thrun and Norvig taught a lecture class at Stanford with 160,000 enrolled students.)

The Flyspeck project is about 80% complete. The project has five parts:

1. The text part of the proof is contained in an unpublished manuscript "Dense Sphere Packings: a formal blueprint." Formalization is being done by a team of researchers.
2. The first computer program (plane graph generation) was formalized by G. Bauer and T. Nipkow.
3. The second computer program (linear programming) is nearly formalized by S. Obua and A. Solovyev.
4. **The third computer program (nonlinear inequality proving) is work in progress (Solovyev). I will describe progress on this part today.**
5. The integration of text part of the proof with computer parts.

- Last year, I did the preliminary work of cleaning up the existing computer code for nonlinear inequality proving.
- I have an informal proofs of a collection of about 500 foramlly specified nonlinear inequalities. They were mostly automatically generated.
- Testing of inequalities is done with a gradient descent program, developed at U. Maryland.
- Interval arithmetic verification is done by code developed for the 1998 proof of the Kepler conjecture.
- This is now a few thousand lines of C++ code.
- Additional C++ code to test and verify a particular inequality is automatically generated from the formal specification. It automatically converts inequalities into an optimized form, splits piecewise analytic functions,. . .

- The main part of the C++ code was translated into about 1K lines of OCaml.
- At this point, late last year, I turned to the project over to Solovyev for formalization.
- The process is similar to that with the linear programming part of the proof. I cleaned up the procedure and implemented it in Ocaml, then Solovyev stepped in and did the formalization of linear programming, optimizing arithmetic in HOL Light for fast execution.
- Solovyev now has automated nonlinear inequality proving in HOL Light. What follows is a report of his recent work.

Problem Domain for Nonlinear inequalities

- $D =$ product of compact intervals in $\mathbb{R}^n$, $(n \leq 6)$.
- f_i analytic function on U_i, for $i = 1, \ldots, k$, such that $D \subset \bigcup U_i$.
- Prove that for all $x \in D$, there exists i such that $x \in U_i$ and $f_i(x) < 0$.
- In most cases, the analytic functions f_i are expressed in terms of field operations on $\mathbb{R}$, $\sqrt{\cdot}$ and $\arctan$.

Definitions

$$\begin{aligned}\Delta(x_1,\ldots,x_6) = {} & x_1x_4(-x_1+x_2+x_3-x_4+x_5+x_6)\\ & + x_2x_5(x_1-x_2+x_3+x_4-x_5+x_6)\\ & + x_3x_6(x_1+x_2-x_3+x_4+x_5-x_6)\\ & - x_2x_3x_4 - x_1x_3x_5 - x_1x_2x_6 - x_4x_5x_6,\end{aligned}$$

$$\Delta_y(y_1,\ldots,y_6) = \Delta(y_1^2,\ldots,y_6^2), \quad \Delta_4 = \frac{\partial\Delta}{\partial x_4},$$

$$\text{dih}\,(y_1,\ldots,y_6) = \frac{\pi}{2} - \arctan_2\left(\sqrt{4y_1^2\Delta_y(y_1,\ldots,y_6)}, -\Delta_4(y_1^2,\ldots,y_6^2)\right).$$

Simple Flyspeck inequalities

Let $D = \{\mathbf{x} \in \mathbb{R}^6 \mid 2 \le x_i \le 2.52\}$.

$$\forall \mathbf{x}.\ \mathbf{x} \in D \Longrightarrow \text{dih}\,(\mathbf{x}) < 1.893,$$

$$\forall \mathbf{x}.\ \mathbf{x} \in D \Longrightarrow \Delta_y(\mathbf{x}) > 0.$$

source: Solovyev thesis overview, March 2012

- Arithmetic is floating point with IEEE-754 directed rounding.
- Real numbers are represented by interval arithmetic.
- Analytic functions f are approximated with Taylor expansions with rigorously computed error terms:

$$|f(x) - f(x^0) - \nabla f(x^0) \cdot (x - x^0)| < \sum_{i,j} m_{ij} \epsilon_i \epsilon_j$$

$$\epsilon_i = |x_i - x_i^0|$$

- The domain D is partitioned into smaller rectangles as needed until the Taylor approximations are accurate enough to yield the desired inequalities.

- The code is optimized in various ways. For example, if the function f is increasing in i, then $f(x) < 0$ on D provided that $f(x) < 0$ on along the right-hand boundary of D.
- The Taylor expansions are generated by symbolic differentiation using the chain rule, product rule, and so forth. A few primitive functions ($\sqrt{\cdot}$, $1/\cdot$, $\arctan$ and some common polynomials) are hand-coded.

Basic constructors

Represent a numeral using an arbitrary base $b \geq 2$. When the base b is fixed, define constants

$$\vdash D_i(n) = bn + i.$$

Example

If $b = 10$, then we can write $123 = D3(D2(D1(0)))$.

source: Solovyev thesis overview, March 2012

Addition of "digits"

If $i + j = k < b$, then

$$\vdash D_i(m) + D_j(n) = D_k(m + n)$$

If $i + j = k \geq b$, then

$$\vdash D_i(m) + D_j(n) = D_{k-b}(\mathrm{SUC}\ (m + n))$$

Addition of numerals

- Store all theorems for addition of "digits" in a hash table.
- The names of the constants are used as key values: the theorem with the left hand side $D_1(m) + D_2(n)$ has the key value "D1D2".
- The addition of numerals is implemented in the usual way.

source: Solovyev thesis overview, March 2012

Table: Performance results for 1000 multiplication operations

Size of operands	HOL Light mult.	Base 16 mult.	Base 256 mult.
5 decimal digits	2.220 s	0.428 s	0.148 s
10 decimal digits	7.216 s	1.292 s	0.376 s
15 decimal digits	16.081 s	3.880 s	1.316 s
20 decimal digits	59.160 s	6.092 s	2.256 s
25 decimal digits	85.081 s	10.645 s	3.592 s

source: Solovyev thesis overview, March 2012

Representation of Formal Floating Point Numbers

Constants

$$\begin{aligned} num\_exp &: num \to num \to num, \\ min\_exp &: num, \\ float &: bool \to num \to num \to real. \end{aligned}$$

Definition

$$\vdash num\_exp\ n\ e = n * b^e$$

b is a numeral: base of the natural number arithmetic.

Definition

$$\vdash float\ s\ n\ e = (-\&1)^{\text{if } s \text{ then } 1 \text{ else } 0} * \&(num\_exp\ n\ e)/\&(b^{min\_exp})$$

In other words, $float\ F\ n\ e = \frac{nb^e}{b^{min\_exp}}$, $float\ T\ n\ e = -float\ F\ n\ e$.

source: Solovyev thesis overview, March 2012

Operations with Floating Point Numbers

- Operations yield inequalities.
- The precision of each operation is controlled by a special (informal) parameter.

Notation: $m = min\_exp$, $n_{exp} = num\_exp$, $f = float$.

Truncation Theorems for *num_exp*: Example

$$\vdash n_{exp}\ 1234\ 10 \leq n_{exp}\ 13\ 12$$

$$\vdash n_{exp}\ 1234\ 10 \geq n_{exp}\ 12\ 12$$

Right hand sides contain at most 2 digits in the first argument (base 10 is assumed).

source: Solovyev thesis overview, March 2012

x/y

To find an upper bound for x/y (assuming all numbers are non-negative), it is enough to find z such that $x \leq z * y$, then $x/y \leq z$.

$\sqrt{x}$

To find an upper bound for $\sqrt{x}$, it is enough to find z such that $x \leq z * z$, then $\sqrt{x} \leq z$.

source: Solovyev thesis overview, March 2012

Elementary Functions: Arctangent

Constants

$$\vdash \mathit{halfatn}\ x = \frac{x}{1+\sqrt{1+x^2}}$$

$$\vdash \mathit{halfatn4} = \mathit{halfatn}\ o\ \mathit{halfatn}\ o\ \mathit{halfatn}\ o\ \mathit{halfatn}$$

$$\vdash \mathit{halfatn4}_{co}(x,j) = \frac{(-1)^j\,\mathit{halfatn4}(x)^{2j+1}}{2j+1}$$

Properties

$$\vdash \forall x.\ \mathit{atn}(x) = 2 * \mathit{atn}(\mathit{halfatn}(x))$$

$$\vdash \forall x.\ |\mathit{halfatn}(x)| < 1$$

$$\vdash \forall n\ x\ v\ \varepsilon_1\ \varepsilon_2\ \varepsilon.\ 2^{-(6n+5)} \le \varepsilon_1 \wedge \left|16\left(\sum_{j=0}^{n} \mathit{halfatn4}_{co}(x,j)\right) - v\right| \le \varepsilon_2$$

$$\wedge\ \varepsilon_1 + \varepsilon_2 \le \varepsilon \Longrightarrow |\mathit{atn}(x) - v| \le \varepsilon.$$

source: Solovyev thesis overview, March 2012

Formal Interval Arithmetic

Definition

$$interval\_arith : real \to (real, real) \to bool$$

$$\vdash interval\_arith\ x\ (lo, hi) \iff lo \leq x \wedge x \leq hi$$

Example

Add intervals $\vdash 1.23 \leq x \leq 1.3$ and $\vdash 0.5 \leq y \leq 1.7$.
The result is (base 10, precision is 2 digits): $\vdash 1.7 \leq x + y \leq 3$.

Intervals with variables

$$a \leq x \leq b \vdash lo \leq f(x) \leq hi$$

This theorem is equivalent to the theorem

$$\vdash \forall x.\ a \leq x \leq b \Longrightarrow lo \leq f(x) \leq hi$$

source: Solovyev thesis overview, March 2012

OCaml Definition of the Solution Certificate

```
Certificate =
| Cell_pass
| Cell_glue of int * Certificate * Certificate
| Cell_mono of bool * int * Certificate
```

source: Solovyev thesis overview, March 2012

Formal Taylor Intervals for Polynomials

Computation Procedure

- Formally compute first and second derivatives of a polynomial.
- Generate a theorem for computing a formal Taylor interval based on expressions for derivatives.
- Generate a procedure which instantiates the generated theorem for different domains and formally evaluates all numerical computations.

source: Solovyev thesis overview, March 2012

Test Polynomial Problems

Prove $m < p(x)$ for all $x \in [a, b]$.

- schwefel: $(x_1 - x_2^2)^2 + (x_2 - 1)^2 + (x_1 - x_3^2)^2 + (x_3 - 1)^2$,
  $m = -5.8806 \times 10^{-10}$, $[a, b] = [(-10, -10, -10), (10, 10, 10)]$
- caprasse:
  $-x_1x_3^3 + 4x_2x_3^2x_4 + 4x_1x_3x_4^2 + 2x_2x_4^3 + 4x_1x_3 + 4x_3^2 - 10x_2x_4 - 10x_4^2 + 2$,
  $m = -3.1801$, $[a, b] = [(-0.5, -0.5, -0.5, -0.5), (0.5, 0.5, 0.5, 0.5)]$
- lv: $x_1x_2^2 + x_1x_3^2 + x_1x_4^2 - 1.1x_1 + 1$, $m = -20.801$,
  $[a, b] = [(-2, -2, -2, -2), (2, 2, 2, 2)]$
- magnetism: $x_1^2 + 2x_2^2 + 2x_3^2 + 2x_4^2 + 2x_5^2 + 2x_6^2 + 2x_7^2 - x_1$,
  $m = -0.25001$,
  $[a, b] = [(-1, -1, -1, -1, -1, -1, -1), (1, 1, 1, 1, 1, 1, 1)]$
- heart: $-x_1x_6^3 + 3x_1x_6x_7^2 - x_3x_7^3 + 3x_3x_7x_6^2 - x_2x_5^3 + 3x_2x_5x_8^2 - x_4x_8^3 +$
  $3x_4x_8x_5^2 - 0.9563453$, $m = -1.7435$,
  $[a, b] = [(-0.1, 0.4, -0.7, -0.7, 0.1, -0.1, -0.3, -1.1)$,
  $(0.4, 1, -0.4, 0.4, 0.2, 0.2, 1.1, -0.3)]$

source: Solovyev thesis overview, March 2012

Performance Tests: Polynomial Inequalities

Table: Test Results for Polynomial Inequalities in PVS and HOL Light

Inequality ID	# variables	PVS Bernstein (s)	HOL Light (s)
schwefel	3	10.23	93.72
caprasse	4	11.44	11.83
lv	4	4.75	2.21
magnetism	7	160.44	11.97
heart	8	79.68	23.21

Note: arithmetic computations in PVS are done by native machine arithmetic. HOL Light verification procedure is completely formal.

source: Solovyev thesis overview, March 2012

- Univariate inequalities in PVS based on Taylor interval arithmetic.
  `http://shemesh.larc.nasa.gov/people/cam/publications/`
- Multivariate polynomial inequalities in PVS based on Bernstein polynomials.
  `http://shemesh.larc.nasa.gov/people/cam/Bernstein/`
  Inspired by Roland Zumkeller's optimization program Sergei.
  `http://code.google.com/p/sergei/`

source: Solovyev thesis overview, March 2012

Verification of Flyspeck Inequalities

$$D = \{\mathbf{x} \in \mathbb{R}^6 \mid 2 \leq x_i \leq 2.52\}$$

Test 1

Verify $\forall y.\ y \in D \Longrightarrow 0 < \Delta_y(y)$.

- Certificate size (the number of Cell_pass elements): 27.
- Formal verification time (float precision = 10): 26.189 seconds.

Test 2

Verify $\forall y.\ y \in D \Longrightarrow \mathrm{dih}\ (y) < 1.893$.

- Certificate size: 4317.
- Formal verification time (float precision = 15): about 13.5 hours.

Test 3

More complicated Flyspeck inequalities.

- Certificate sizes: 8000–67000.

source: Solovyev thesis overview, March 2012

Optimization Strategies

- Cached arithmetic (low level and high level).
- Verification of groups of inequalities (on common subdomains).
- Adaptive arithmetic precision.

source: Solovyev thesis overview, March 2012

Benchmarks from Obua's Thesis.

Finally, the 'Time' column tells us how many minutes the examination of the tame graph lasted. We used the SML mode of the HOL Computing Library. Each tame graph has been examined by its own Isabelle process. Each Isabelle process ran on a dedicated processor of a cluster of 32 four processor 2.4GHz Opteron 850 machines with 8 GB RAM per machine. The quickest process needed 8.4 minutes, the slowest 67. The examination of *all* tame graphs took about 7.5 hours of cluster runtime. This corresponds to about 40 days on a single processor machine.

We were able to prove the inconsistency of 2565 of the graph systems, and failed on 206. This yields a success rate of about 92.5%.

source: Obua's thesis

Benchmarks

Flyspeck 2012

#	Inconsistent	Time
1	Yes	15.4
2	Yes	21.9
3	Yes	17.6
4	Yes	39.8
5	Yes	19.4
6	Yes	23.1
7	Yes	26.9
8	Yes	24.3
9	Yes	41.5
10	Yes	40.7
11	Yes	37.7
12	Yes	30.4
13	Yes	30.9
14	Yes	47.3
15	Yes	53.5
16	Yes	66.8
17	Yes	56.1
18	?	47.3
19	Yes	15.9
20	Yes	12.7
21	Yes	20.0
22	Yes	20.8
23	Yes	22.9
24	Yes	23.6
25	Yes	24.3

source: Obua's thesis

Performance Tests

Each Flyspeck linear program can be completely formally verified in about 5 seconds. There are about 50,000 linear programs in the Flyspeck project.

Linear program ID	# vars	# ineqs	HOL arith	Base 256 arith
18288526809	743	519	4.048 s	2.772 s
168941837467	750	591	5.096 s	3.196 s
25168582633	784	700	8.392 s	4.308 s
72274026085	824	773	7.656 s	5.120 s
28820130324	875	848	9.292 s	5.680 s
202732667936	912	875	9.045 s	5.816 s
156588677070	920	804	8.113 s	5.252 s
123040027899	1074	1002	11.549 s	6.664 s
110999880825	1114	1000	10.085 s	6.780 s

source: Solovyev thesis overview, March 2012

Thank You!

Flyspeck 2012