

The Kepler Conjecture (Eight Years Later)

Thomas C. Hales
September 1, 2006



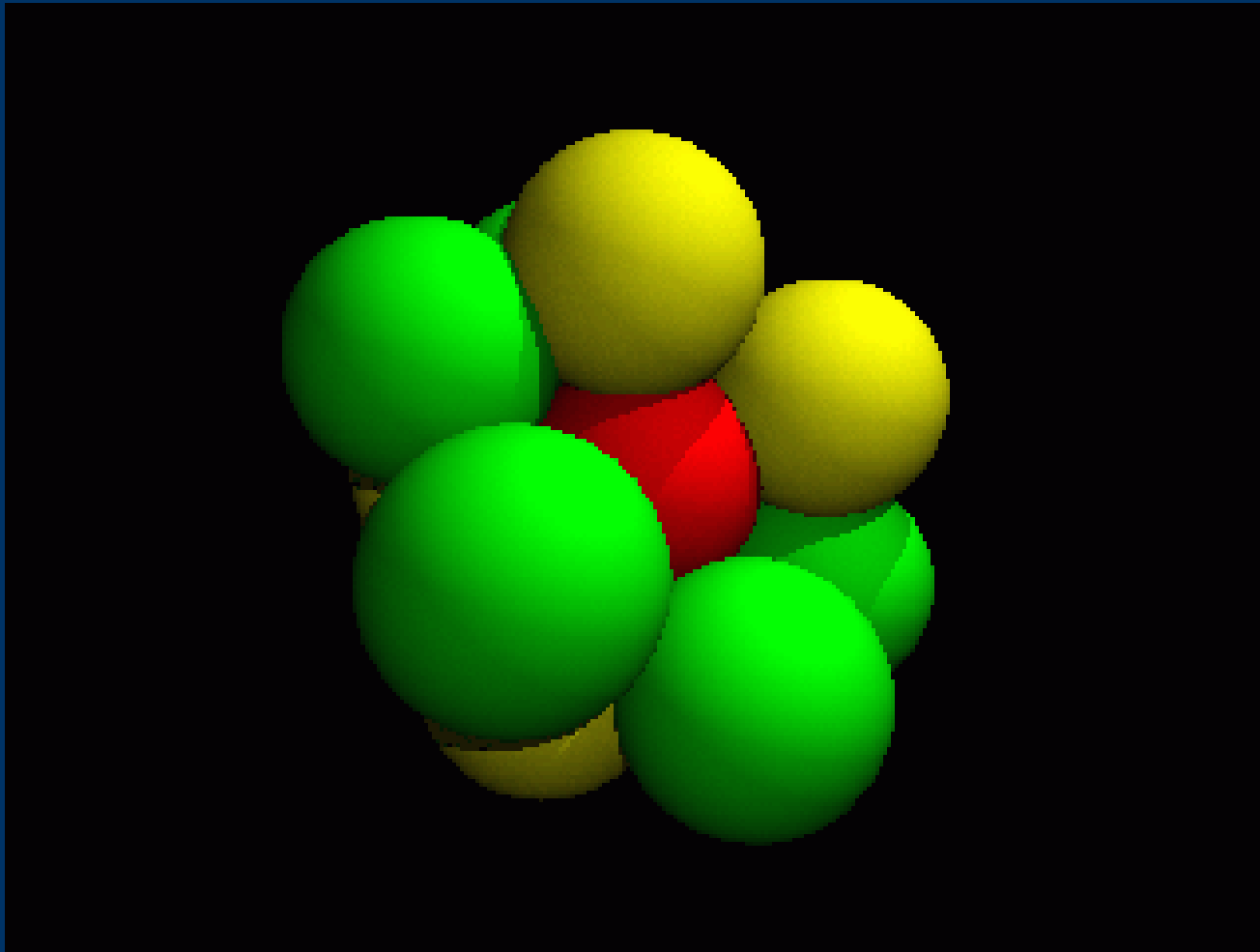
The Kepler Conjecture

Theorem (Ferguson, H, 1998)

No packing of congruent balls can have density greater than that of the face-centered cubic packing.



The face-centered cubic packing



Other packings with the same density



History

- 1610 Kepler
 - 1900 Hilbert
 - 1953 L. Fejes Toth
 - 1994 Proposed strategy
 - 1996 Detailed strategy (Mt. Holyoke)
 - 1997 Ferguson's thesis
 - 1998 solution announced
 - 2005 abridged publication
 - 2006 full publication
 - 20?? formal proof
-
-

FLYSPECK

- Flyspeck = FPK = Formal Proof of Kepler
 - A formal proof is one in which every single logical step of the proof has been checked by computer.
 - The axioms are put into the computer. The basic rules of inference are put into the computer. Nothing is allowed out, except for statements whose proofs have been comprehensively checked.
 - The goal of the flyspeck project is to give a formal proof of the Kepler conjecture (including all of the computer code).
-
-

Why flyspeck?

- The referees refused to examine the computer code.
- Twelve referees spent several years on the process became exhausted, before the proof was fully certified.
- The limits of traditional refereeing of computer-based proofs became evident in the process.



Traditional vs. Formal Proof

Traditional:

- Checked by referees
- Elementary inferences are skipped
- Knowledge base is that of an expert in the field
- Foundational Axioms of math are remote, almost irrelevant.
- Proofs are intuitive.



Formal Proofs

- Checked by computer
- Every inference is checked.
- Only previously formally checked facts may be used.
- The details of the axiomatic system are of great importance.
- It is extremely thorough.



Shortcomings of Traditional Review of Computer Code

- Referees generally refuse to review computer code.
 - Referees want to be convinced that computer code performs according to specification.
 - The mathematical community has no software standards.
 - Many mathematicians lack computer skills.
 - Software houses (those guys who produce all those buggy programs) have much tighter standards of quality control than we do.
-
-

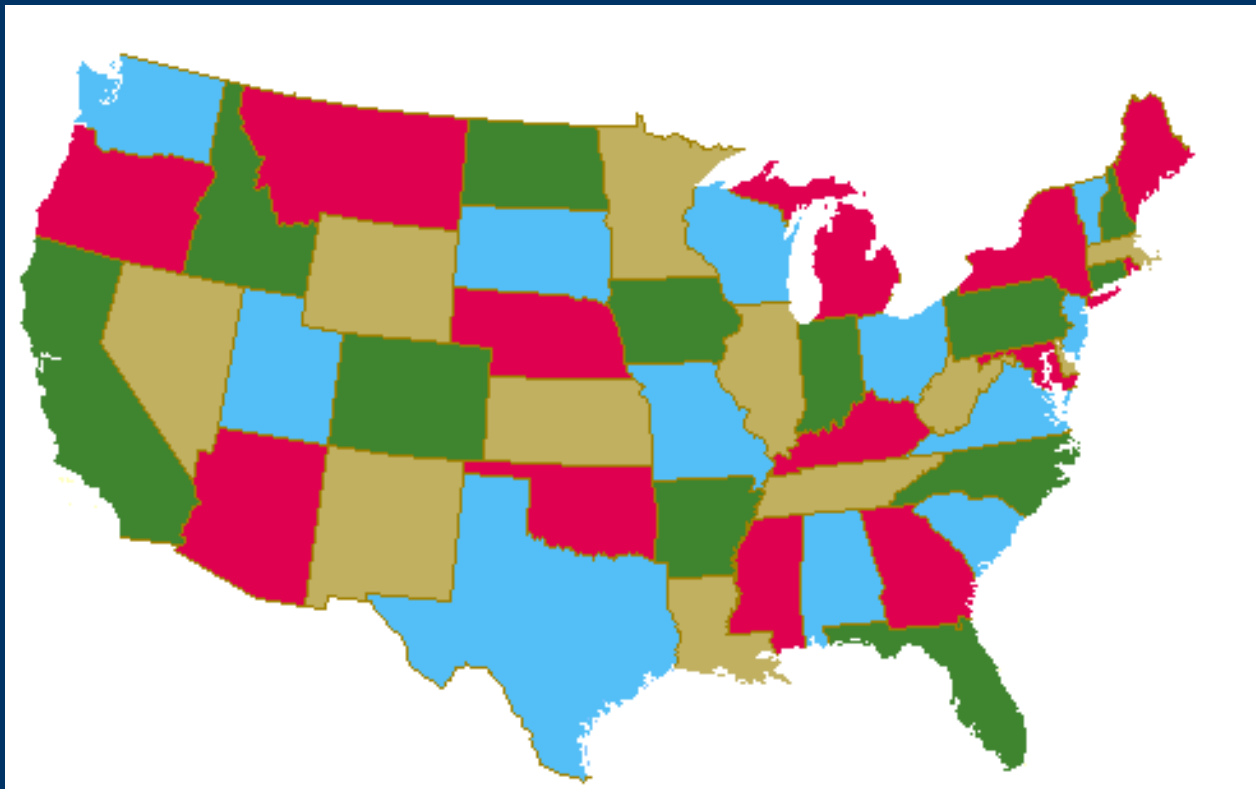
Answer: formal proofs

- A formal proof gives the editors and referees some assurance that computer code performs according to specification, even if the referees do not examine the code in any detail.



Examples of Formal Proofs

- Four color Theorem
- (2004 Gonthier)



Examples of Formal Proofs

- Prime Number Theorem
- (2004 Avigad)



Examples of Formal Proofs

- Jordan Curve Theorem
- (2005 H-)
- (2005 Mizar group)



Examples of Formal Proofs

- Brouwer Fixed Point Theorem
- (2005 Harrison)
- (2005 Mizar Group)



Some formal proof checkers

- Isabelle/HOL
- COQ
- HOL Light



The Perfect Realm of Mathematics

Proof that $1 + 1 = 2$

$$\begin{aligned}1 + 1 &= 1 + \text{SUC } 0 \\ &= \text{SUC } (1 + 0) \\ &= \text{SUC } 1 \\ &= 2\end{aligned}$$

Are Formal Proofs Infallible?

- No, but neither is the current system in which referees refuse as a matter of principle to examine any code.
 - The system I use (HOL Light) becomes self-checking after the first 1000 lines of code.
 - Integrity of the system is enforced by the type system of CAML.
 - These first 1000 lines of code are treated with the same care that we treat our most valuable mathematical treasures.
-
-

What is involved in the the Kepler conjecture formalization

- Every formal proof begins with a traditional proof. (The traditional proof may involve computer code.)
- The 1998 proof of the Kepler conjecture has about 300 pages of traditional mathematical text.
- The text is supplemented by three major computer programs.
 - 40 K lines of code in Java, C, C++, CPLEX, Mathematica.
 - 3 GB of data.
- The three programs are for (1) proving nonlinear inequalities, (2) tame graphs, and (3) linear programs.

Outline of the Proof of Kepler

- The problem of finding a packing of greatest density can be viewed as an optimization problem in an infinite number of variables, where the variables represent the coordinates of the spheres.
 - An initial reduction brings this optimization problem down to a finite number of variables. In other words, it is enough to examine finite clusters of spheres.
 - To each cluster of spheres, a planar graph is attached, that describes the combinatorial structure of nearby spheres.
-
-

Primary Computer Programs in Kepler

- **Classification of Tame Graphs**
- Nonlinear Inequality Prover
- Linear Programming



Outline of the proof: tame graphs

- A graph attached to a small cluster of spheres that is a potential counterexample to the Kepler conjecture is called a “tame graph.”
 - The graphs typically have about 12-15 vertices, and about 30 edges.
 - Vertices of the graph = center of spheres
 - Edges of the graph = nearby centers of spheres.
 - The classification of all tame graphs is a major component of the proof of the Kepler conjecture.
 - This classification is carried out by computer.
 - There are about 2000 such graphs.
-
-

Tame Graphs

- The definition of a tame graph can be given directly, without appeal to sphere packings and the Kepler conjecture
 - The graph is planar and connected.
 - It has no loops or multiple joins.
 - Each face has between 3 and 8 sides.
 - Each vertex has degree between 2 and 6.
 - It has an admissible weight assignment.
 - Etc.

Tame Graphs (continued)

- The proof eliminates each tame graph case-by-case and concludes no counterexamples exist.
 - In 2000, I found a bug in the computer code that caused it to miss a few tame graphs. (The problem was easily fixed, onced detected.)
 - In December 2005, Nipkow, Bauer, and Schutz completed a formal verification of the enumeration of tame graphs. (So there are no further bugs in the code.)
-
-

Formal Enumeration of Tame Graphs

- Bauer translated my Java code into ML.
 - The code was de-optimized.
 - The proof assistant Isabelle/HOL checked the correctness of every line of code.
 - McLaughlin has recently adapted this work to the enumeration of graphs in the dodecahedral conjecture.
 - References:
 - G. Bauer “Formalizing Plane Graph Theory”, 2005.
 - Nipkow, Bauer, Schulz, “Flyspeck I: Tame Graphs” 2006.
-
-

Primary Computer Programs in Kepler

- Classification of Tame Graphs
- **Nonlinear Inequality Prover**
- Linear Programming



Nonlinear Inequalities

- Part of the proof of the Kepler conjecture involves nonlinear inequalities
 - $f(x_1, \dots, x_6) < 0$
 - The function f is an expression in sqrt, atan, and rational functions of x .
 - The inequalities express relations among volume, edge lengths, dihedral angles, solid angles, etc.
-
-

Nonlinear inequalities

- The proof of Kepler involves about 1000 such inequalities (each involving about 6 variables).
- Traditional calculus estimates would be far too slow and cumbersome for this many inequalities.
- A program that automates the proving of such inequalities was written.



Nonlinear inequalities: Method of Proof

- Take a subdivision of the domain into small rectangles (adapting the size of the rectangle to the required accuracy).
 - Take a Taylor series expansion on each rectangle, with explicit error bound.
 - Evaluate the Taylor polynomial by computer to see that the inequality holds on each rectangle.
 - (Use interval arithmetic to control computer floating point errors.)
-
-

Nonlinear inequalities: Drawbacks of the Method

- The calculations are slow. (About 3 months on a 1998 workstation.)
- The computer code is very hard to check for correctness. (Example: string to floating point conversion.)
- Bugs if they exist can be extremely subtle. (Example: $x*(y*z) = (x*y)*z?$)

Nonlinear inequalities: Reasons to trust the code

- Sam Ferguson and I had independent versions of the code and many results were checked in both systems.
 - Most bugs would alter the results by at most a machine epsilon or so. The inequalities had much larger margins than this.
 - The inequalities were independently checked with a nonlinear optimization package. (This is not mathematically rigorous, but it is a useful check.)
 - I wrote two versions of the code and checked many results in both systems.
-
-

Nonlinear inequalities: Formal Verification

- This is still work in progress. It will probably be one of the last parts of the Flyspeck project to be completed.
 - Zumkeller has made significant progress on the formal verification.
 - Reference: “Formal Global Optimisation with Taylor Models” Zumkeller, 2006.
-
-

Nonlinear inequalities: Zumkeller's work

- Zumkeller gives totally automated proofs in COQ of inequalities
 - $f(x_1, \dots, x_n) < 0$
 - His code and algorithms are certifiably correct.
 - His calculations use real numbers rather than floating point.
 - This gives a sample implementation of what is needed, but the algorithms are still too slow to complete FLYSPECK. (Typically, formal computations run 10000 or even 100000 times slower than nonformal computations.)
-
-

Nonlinear inequalities: Calculating with real numbers

- Zumkeller uses actual real numbers, not floating point or symbolic representations.
 - One common representation of real numbers is as a Cauchy sequence. However, this delta-epsilon definition requires an existential quantifier.
 - If, instead of an existential quantifier, we give a certificate of existence, then we have what is called a constructive real numbers.
 - Computers can directly manipulate and do computations with constructive real numbers.
-
-

Nonlinear inequalities: Constructive real numbers

- A constructive real number is a pair (x, m) , where
 - $x(n)$ is a sequence of rational numbers
 - $m(r)$ is a function on rational taking values in natural numbers
 - The function m is called the modulus function. It tells how fast the sequence x converges.
 - For every positive epsilon, if k and k' are larger than the modulus (at epsilon), then $x(k)$ and $x(k')$ are within epsilon.
 - References:
 - E. Bishop, *Foundations of Constructive Analysis*, 1967.
 - H. Schwichtenberg, *Constructive Analysis with Witnesses*, 2004.
-
-

Nonlinear Inequalities: Zumkeller

- Zumkeller's code is completely certified by COQ.
 - By the design of COQ, Zumkeller cannot do anything to compromise the system.
 - If there is a bug, it will never be traced to Zumkeller, it will always point to a severe bug in a chip, OS, compiler, or COQ itself.
 - COQ is a general purpose mathematical assistant that has nothing in particular to do with Kepler, Flyspeck, nonlinear inequalities.
 - COQ was used by Gonthier to give a formal proof of the four color theorem (2004).
-
-

Primary Computer Programs in Kepler

- Classification of Tame Graphs
- Nonlinear Inequality Prover
- **Linear Programming**



Vanishing Box Trick

- Let D be a counterexample to the Kepler conjecture.
- Put D in a box.
- Measure the width of the box.



Vanishing Box Trick

- If the width of the box is negative, then the box is empty, and the counterexample D does not exist.
- More generally, take all counterexamples and put them in polyhedra defined by hyperplanes.
- If each system of inequalities is infeasible, then no counterexamples exist.



Vanishing Box Trick



Linear Programming

- Weaknesses of the 1998 proof
 - 3 GB of data
 - The branch and bound arguments were based on extensive human-computer interaction.
 - The branch and bound arguments relied on detailed geometrical information about the space of counterexamples.
 - To check the correctness of the proof, it is necessary to study the logs of these interactive sessions.
 - It is hard to articulate a precise theorem proved by the linear programming/branch and bound methods.
-
-

Linear Programming infeasibility

A certificate of infeasibility for the linear system

$$Ax \leq c; \quad a \leq x \leq b$$

is a pair (u,v) such that

$u, v, uA + v$ are non-negative, and

$u(c - Aa) + v(b - a)$ are negative.

(Think of x as the counterexample, and (u,v) as the negative width of the box.)

[vanishing-box-trick theorem] if a certificate of infeasibility exists, then x does not exist.

Proof: $(uA + v)(x - a)$ is both negative and non-negative.

Linear Programming

- Steven Obua is implementing a formal proof of the linear programming part of the Kepler conjecture.
 - Reference: S. Obua, Proving Bounds for Real Linear Programming in Isabelle/HOL.
 - He has developed the tools to do the linear programming. He is just waiting for the final specs from me.
-
-

Linear Programming Specs

- I have just produced specs for the linear programming part of the proof
 - The geometry is eliminated (only combinatorics and linear algebra remain).
 - The branch and bound makes no reference to the space of counterexamples. Branching follows its own internal logic.
 - In fact, the linear programming is now entirely self-contained.
 - It clarifies the various views of inequalities.
-
-

Linear Programming: Two Views of Inequalities



Linear Programming Theorem

Theorem (H., 2006) [Based on 1998 proof]
Every primary tame hypermap system is weakly
infeasible. [Combinatorics and Linear Programs]

Corollary: Such a system admits no strong solutions.

Theorem. A counterexample to the Kepler conjecture
is a strong solution to a primary tame hypermap
system. [Geometry]

What is a hypermap system??

- In 2004, Gonthier gave a formal proof of the four-colored theorem in the COQ system (Dec 2004).
 - It contains about 50K SLOC.
 - My proof of the Jordan curve theorem (Jan 2005) contains about 40K SLOC.
 - Curiosities:
 - Gonthier finished first, even though the proof of the four-colored theorem relied heavily on the Jordan curve theorem.
 - The Jordan curve theorem is much a much easier theorem than 4CT, but they use about the same amount of code.
-
-

Hypermap systems

- Gonthier achieves these efficiencies by encoding the four-color theorem combinatorially as a planar hypermap (a finite set with 2 partitions).
 - Seeing these efficiencies, I have reworked the proof of Kepler in terms of hypermaps along the lines of Gonthier's proof of the 4CT.
 - In this way the geometry and topology can be eliminated from the linear programming component of the proof.
-
-

Planar Hypermaps

