Computer-Assisted Proofs in Geometry

Thomas C. Hales

October 4, 2006

Computer-Assisted Proofs in Geometry

Thomas C. Hales

October 4, 2006

1 Interval Arithmetic

Interval Arithmetic is a way to do reliable computing on a computer. It keeps track of all the round-off errors. Rigorous mathematical results can be obtained in this way.

Example. Suppose we have a toy model computing device that only represents the numbers

$$0, \pm 1, \pm 2, \pm 4, \pm 8, \ldots$$

On this device, we can give an upper bound to $(1+3)*5$ as follows

$$\begin{aligned}(1+3)*5 \quad &\mapsto (1+4)*8 \\ &\mapsto (8)*8 = 64.\end{aligned}$$

Similarly, we obtain a lower bound

$$\begin{aligned}(1+3)*5 \quad &\mapsto (1+2)*4 \\ &\mapsto (2)*4 = 8.\end{aligned}$$

So $(1+3)*5 \in [8, 64]$.

We extend addition, subtraction, multiplication, and division to interval operations.

Example. We approximate $[1, 3] + [5, 6]$ as follows on this device:

$$[1, 3] + [5, 6] \quad \subset [4, 16].$$

We approximate $[8, 16]/[2, 64]$ as follows.

$$[8, 16]/[2, 64] \quad \mapsto [8/64, 16/2] \mapsto [0, 8].$$

We can evaluate polynomials on intervals. (The answer depends on the syntactical form of the polynomial.)

$$\begin{aligned} f(x) &= x - x & \qquad f([-1,1]) &= [-1,1] - [-1,1] = [-2,2]. \\ g(x) &= 0 & \qquad g([-1,1]) &= 0. \end{aligned}$$

We can approximate analytic functions with Taylor approximations.

$$\begin{aligned} |\arctan x - (x - x^3/3)| &< x^5/5, \text{ if } x \in [0,1] \\ \arctan x &\in [x - x^3/3 - x^5/5, x - x^3/3 + x^5/5]. \end{aligned}$$

On a real computer, we can represent numbers more accurately than in this toy model computing device. However, the same ideas apply. We still get bad behavior from $x - x$ on the interval $[-1, 1]$.

2 Applications of Interval Arithmetic

Strange Attractors in the Lorenz Equations

In 1963, Lorenz introduced the ODE

$$\begin{aligned}\dot{x}_1 &= -10x_1 + 10x_2\\ \dot{x}_2 &= 28x_1 - x_2 - x_1x_3\\ \dot{x}_3 &= -(8/3)x_3 + x_1x_2\end{aligned}$$

and found that the solutions are extremely sensitive to initial conditions.

The Lorenz equations have been extensively studied over the years, but it was only in 2002 that W. Tucker proved that the solutions have the structure of a strange attractor.

Tucker

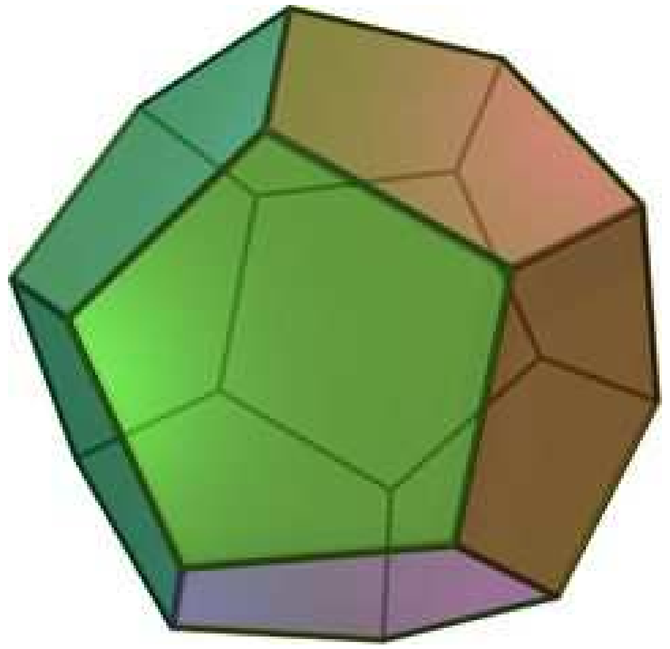
Tucker does not offer a traditional proof of this result. Rather, he gives an algorithm, which he implements in C++. The algorithm uses interval arithmetic with directed rounding to control the round-off errors. By running the algorithm, he is able to prove the existence of a strange attractor.

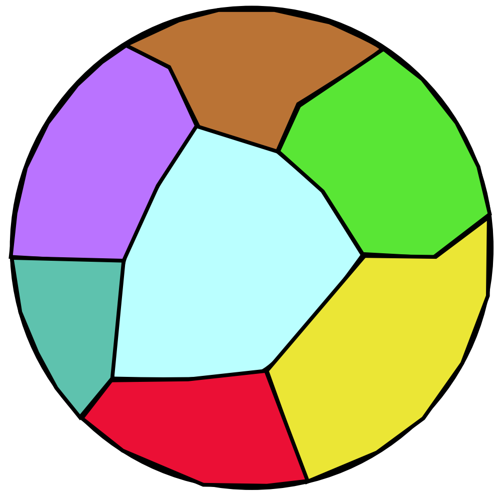
Example: The Double Bubble Problem

In 2004, Hutchings, Morgan, Ritoré and Ros solved the general double bubble problem. In 1995, the special case of equal volume bubbles was solved with the help of computer by Hass, Hutchings, and Schlafly.

The computer was used to eliminate various exotic double bubbles such as the torus bubble.

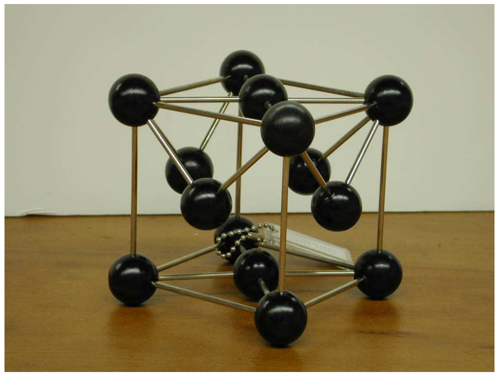
Example: The dodecahedral conjecture



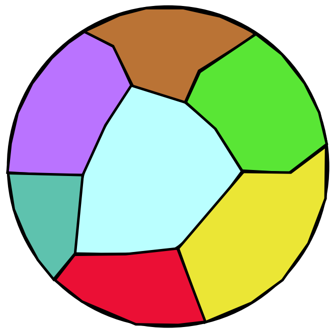


In the last lecture, I described the basic strategy of the proof. McLaughlin uses Interval arithmetic to prove lower bounds on chunks of volume of Voronoi cells for each polygonal region on the unit sphere.

Example: The Kepler Conjecture



The proof of the Kepler conjecture follows the same outline as the proofs of Leech and McLaughlin: draw a graph on the surface of the sphere, make estimates for each face of the graph, and collect the estimates together into a global bound.



- In 13 spheres, we are estimating the areas of the spherical polygons.
- In McLaughlin's case, we are estimating the volumes of truncated Voronoi cells.
- In the proof of Kepler, we are estimating the a modified volume function on truncated Voronoi cells: called the *scoring function* on decomposition stars.

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

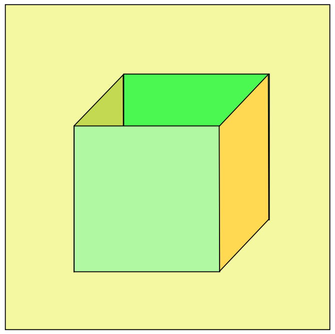
- The proof of Kepler involves about 1000 such inequalities (each involving about 6 variables).
- Traditional calculus estimates would be far too slow and cumbersome for this many inequalities.
- A program that automates the proving of such inequalities was written.
- It is based on interval arithmetic

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.)

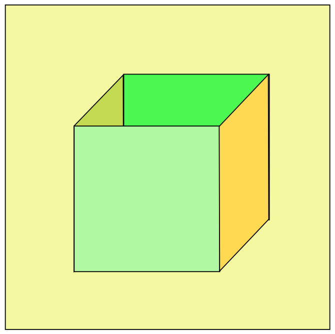
3 Linear Programming

Another computational method of great use in optimal geometry is linear programming and linear relaxation.

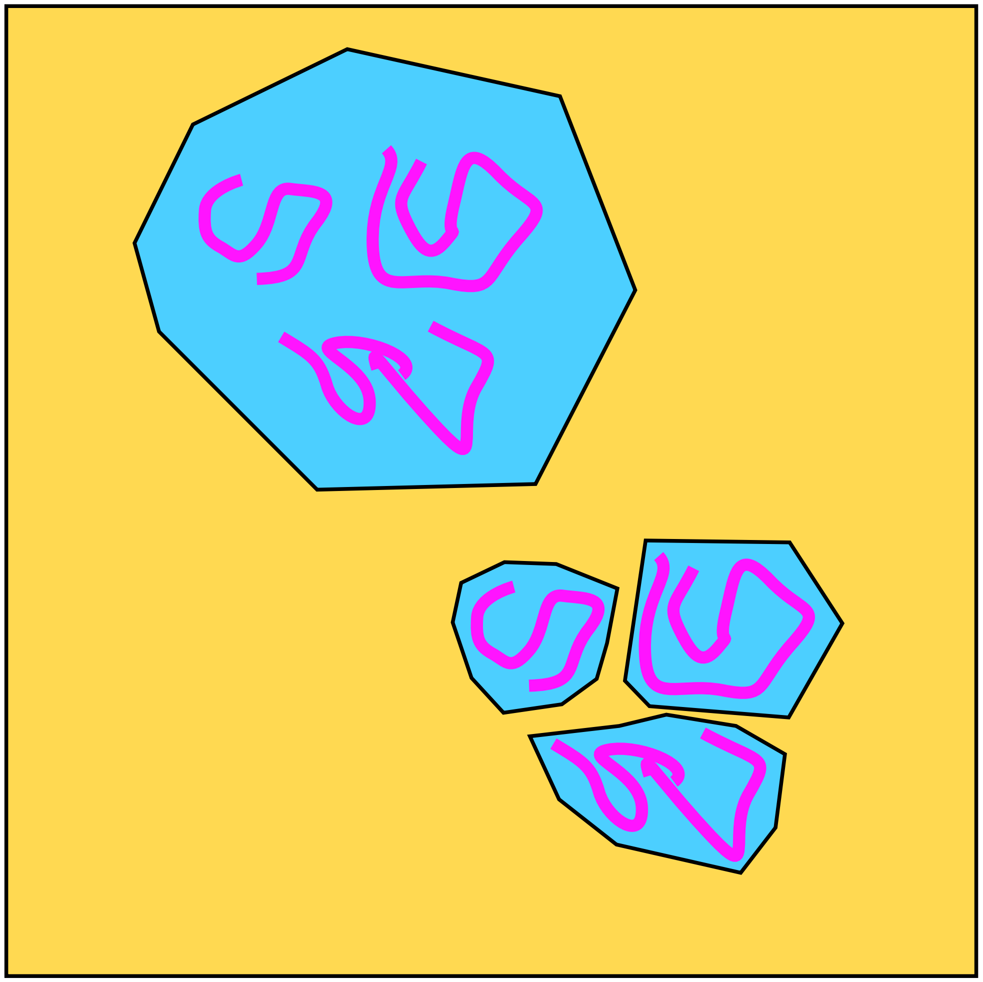


Vanishing Box Trick

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



- 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.
- This is linear relaxation: the space of counterexamples is nonlinear, but the set of counterexamples is relaxed into the larger set of polyhedra.



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 Specs

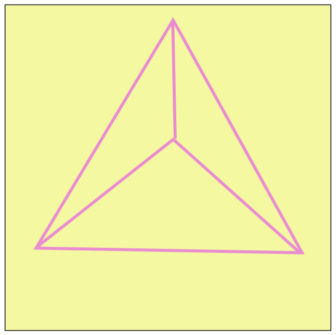
I have just produced specs for the linear programming part of the proof.

- 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.

This is all part of a formal verification project for the Kepler Conjecture (called FLYSPECK).

Linear Programming Specs: Nonlinear-linear convertibility

- The angles at a vertex satisfy a linear relation, the angle sum is 2π.
- The angles at a vertex are nonlinear functions (of the edge lengths).
- The linear programming relations are linear.
- The archive of interval arithmetic inequalities is nonlinear.



Linear programming specs: Nonlinear-linear convertibility

- To achieve automation of nonlinear-linear convertibility, we introduce a new structure, called a formal inequality.
- A formal inequality is a unification of the nonlinear and linear aspects of the inequalities in the Kepler conjecture.
- A formal inequality has a deformalization that can be used in linear programs.
- It also admits nonlinear interpretations, such as those in the database on nonlinear inequalities.
- This gives us a procedure to look up formal inequalities in a nonlinear inequality database.
- Formal inequalities can be combined with standard Boolean operations.

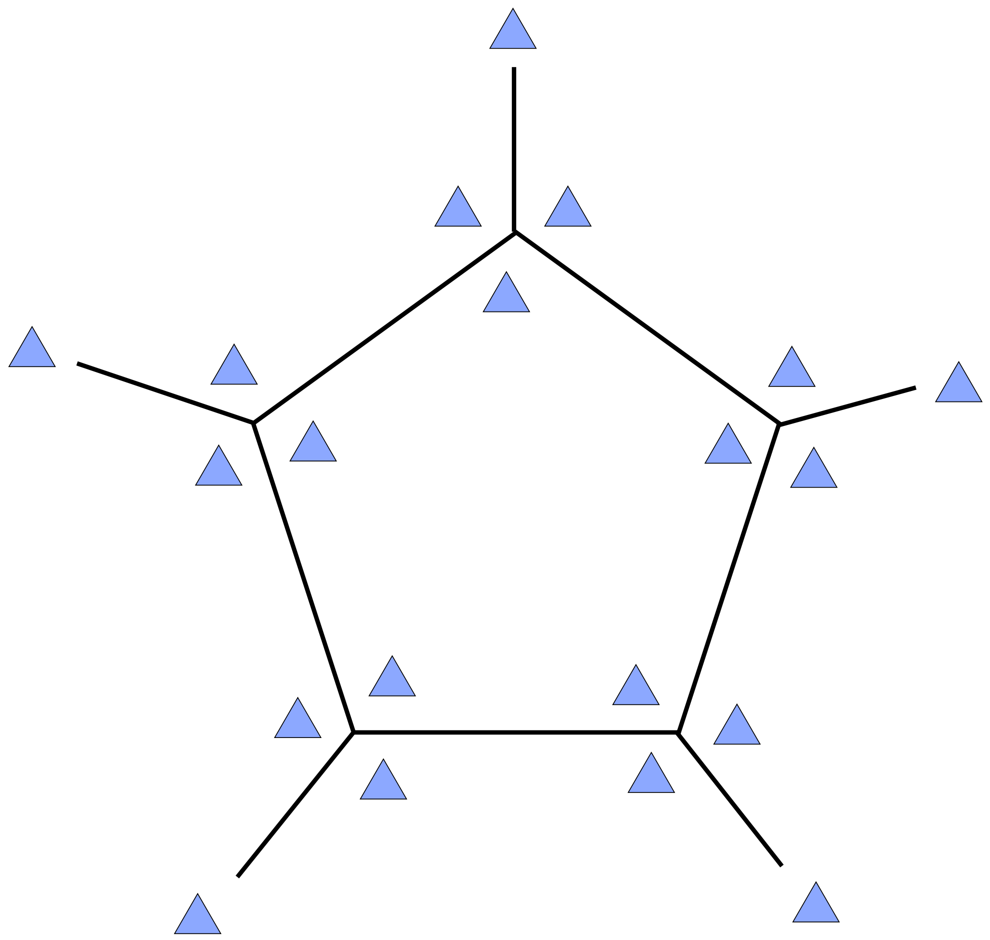
Linear Programming Specs: Removal of Geometry

- The 1998 proof uses extremely detailed information about the geometry of finite clusters of spheres to guide the branch and bound process in linear programming. In the new linear programming spec, the geometry has been completely removed from this part of the proof.
- It is now entirely combinatorial and linear programming on one side, and geometry on the other.

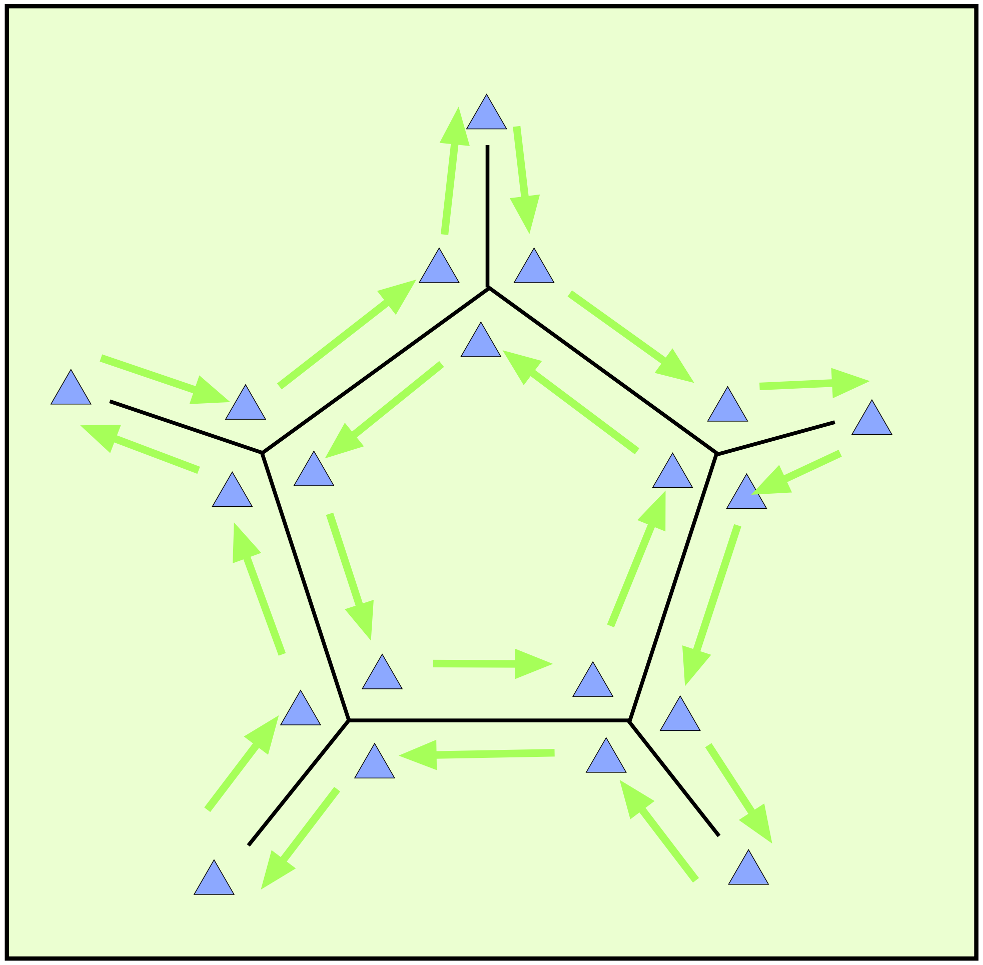
Linear programming specs: Removal of Geometry

- The combinatorial structure of finite clusters are encoded combinatorially as hypermaps (following Gonthier). Everywhere geometrical information is used in the 1998, a flag has been introduced (taking finitely many values) that serves as a combinatorial proxy for the geometry.
- It is not necessary to understand anything of the geometry to use these flags.

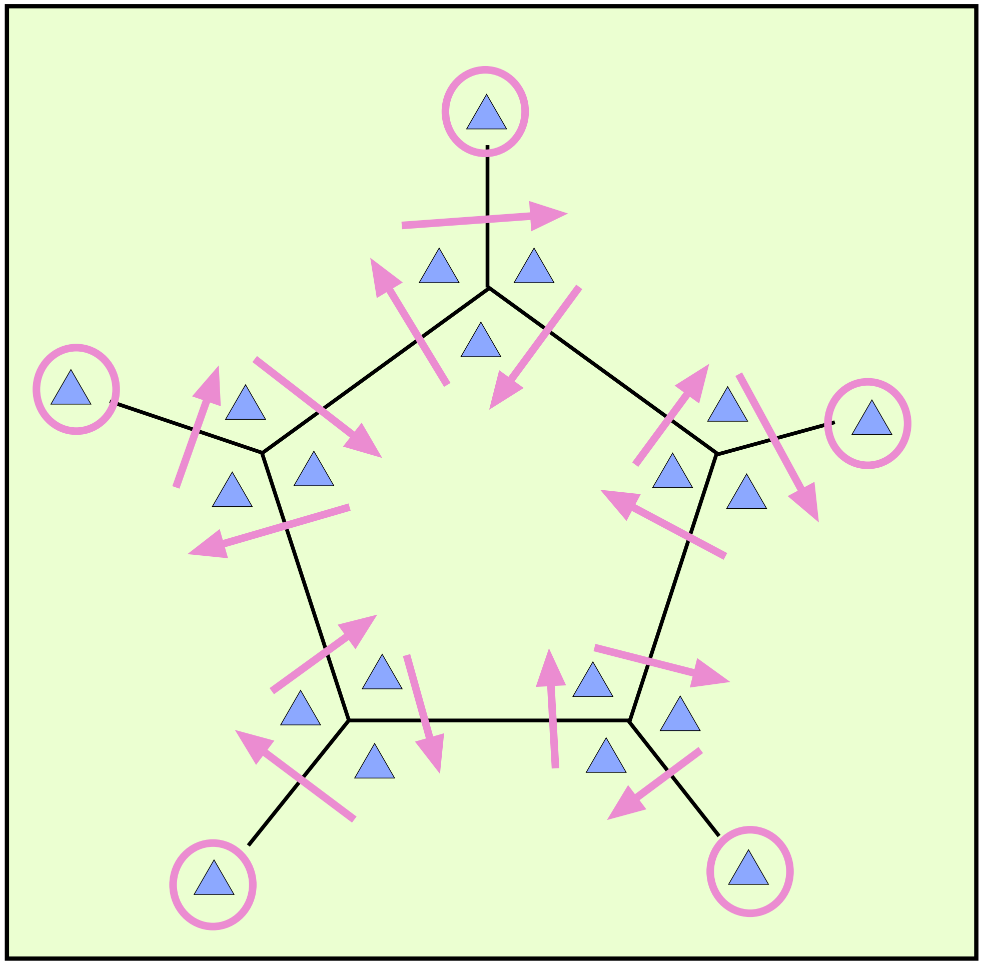
Planar Hypermaps



Planar Hypermaps: face map



Planar Hypermaps: node map

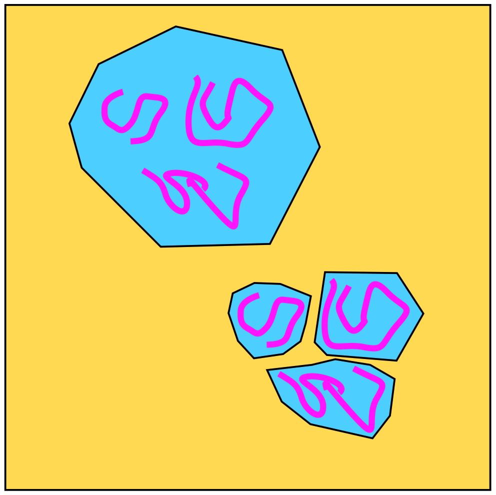


Linear Programming Specs: Weak infeasibility

- In the 1998 proof, there is no stated theorem about the results of the linear programming part of the proof, except that the linear programming branch-and-bound methods eliminate all the counterexamples.
- The reason is that it is hard to say exactly what the linear programming does, except by reference to set of counterexamples to the Kepler conjecture.

Linear Programming Spec: Weak infeasibility

- In the 1998 proof, the set of counterexamples to the KC guides the branch-and-bound operations.
- In the current spec, we define the notion of weak infeasibility, that is, admissible branch-and-bound operations lead to infeasibility.



Linear Programming Specs: Weak infeasibility

- **(Linear part of KC) Theorem**: Every (suitable) linear system is weakly infeasible.
- **(Linear-nonlinear link) Theorem**: A weakly infeasible linear program does not admit a strongly feasible solution. [The vanishing box trick.]
- **(The non-linear part of the KC.) Theorem**: If there is a counterexample to the Kepler conjecture, then it gives a strongly feasible solution to a [suitable] linear system.

Summary

- There are three computer programs in the proof of the KC.
- It is easy to state what two of the computer programs should do. (Give all tame plane graphs and prove the given list of nonlinear inequalities.)
- It is rather difficult to state what the third computer program should do (linear programming).
- This is the most poorly explained part of the published proof of the KC.
- We finally have an (approximate) specification of what the linear programming problem should do.
- It now seems possible to fully automate the linear programming part of the proof, rather than rely on long interactive sessions, as was done in the 1998 proof.