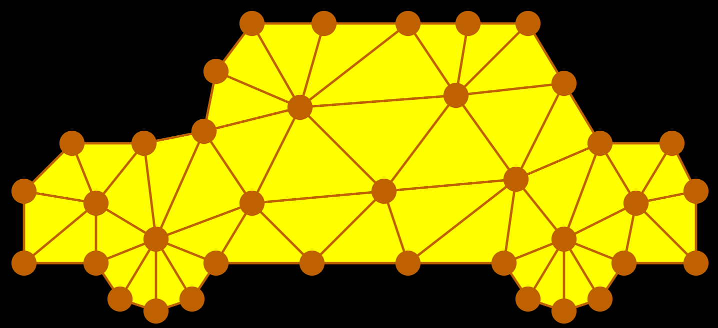
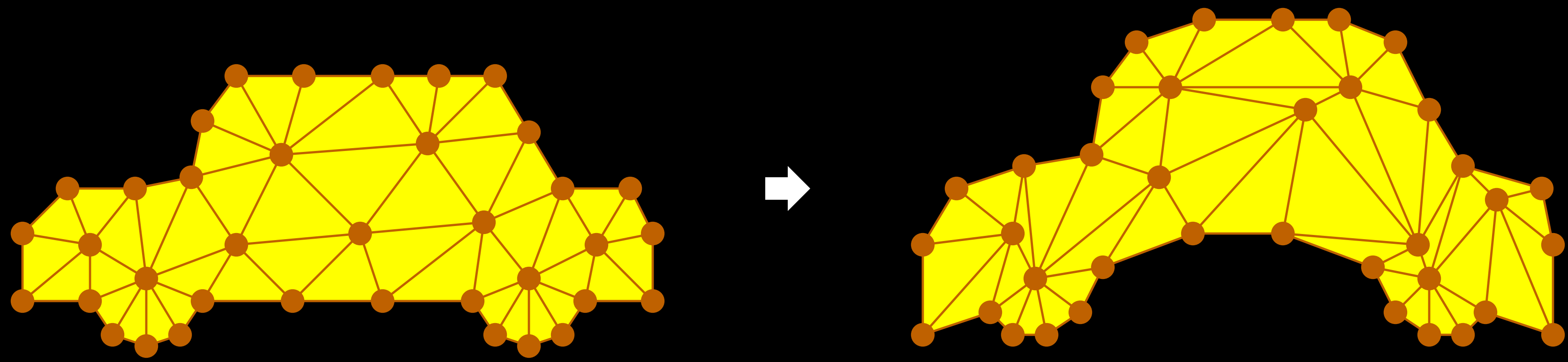
Star Splaying

Jonathan Shewchuk
University of California at Berkeley

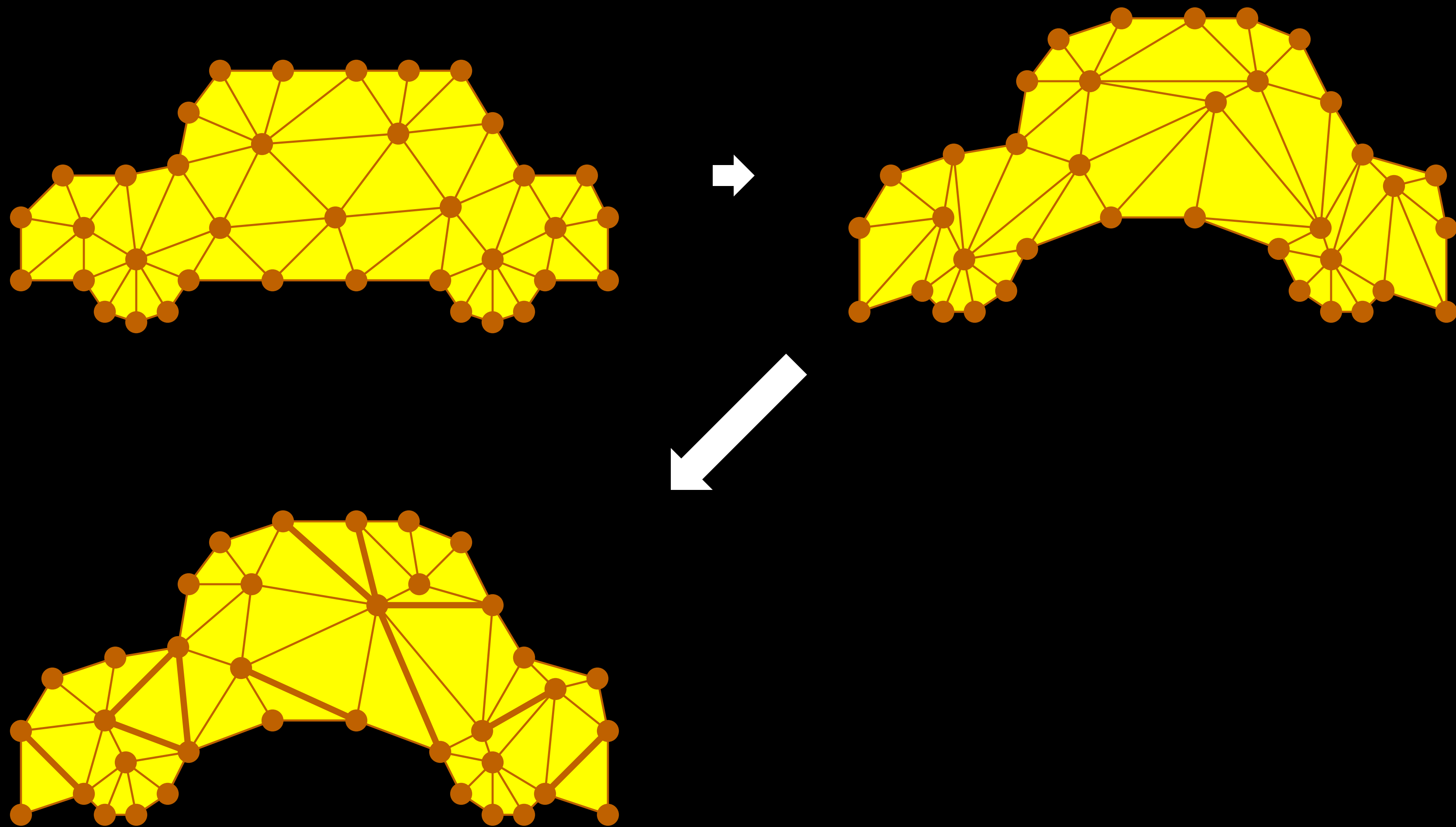
Application: 3D Dynamic Meshing



Application: 3D Dynamic Meshing

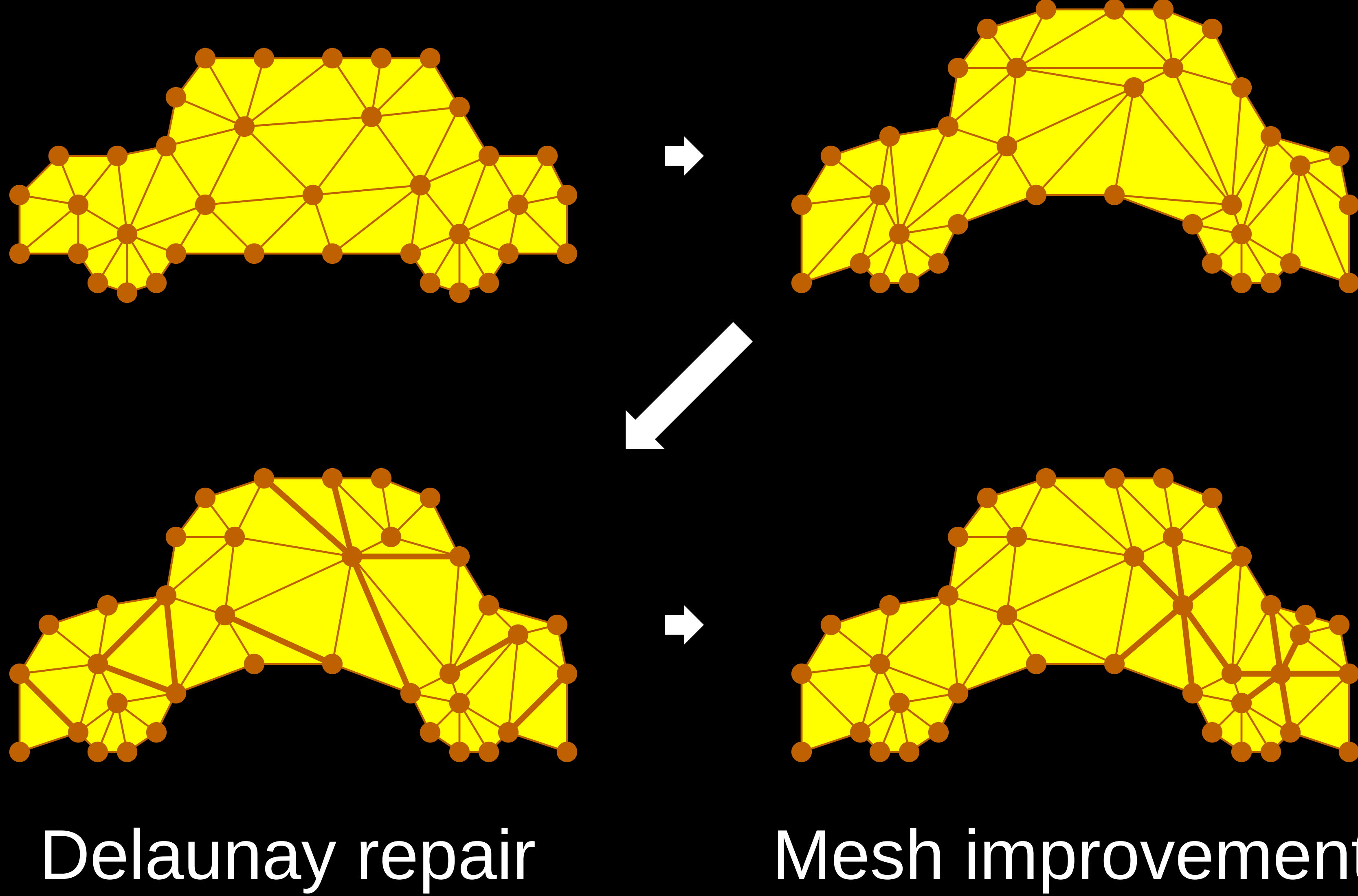


Application: 3D Dynamic Meshing



Delaunay repair

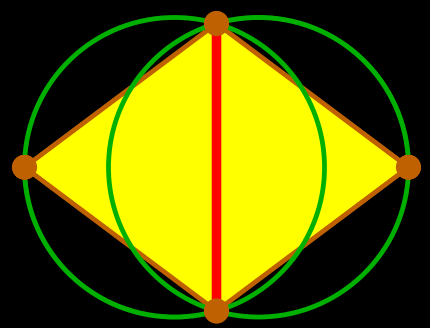
Application: 3D Dynamic Meshing



The Flip Algorithm

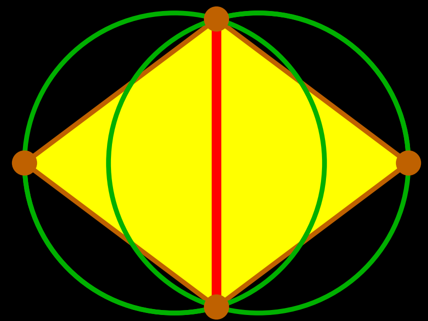
The Delaunay Triangulation

An edge is *locally Delaunay* if the two triangles sharing it have no vertex in each others' circumcircles.

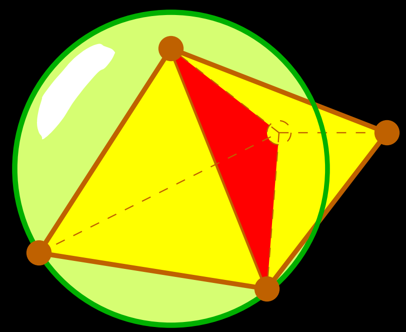


The Delaunay Triangulation

An edge is *locally Delaunay* if the two triangles sharing it have no vertex in each others' circumcircles.

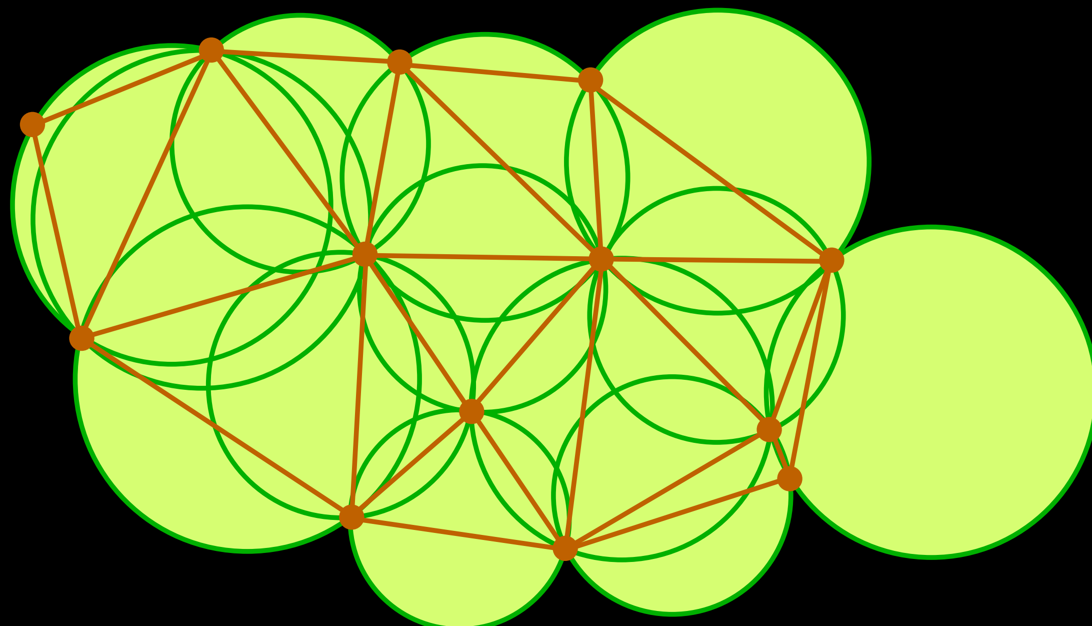


A triangular face is *locally Delaunay* if the two tetrahedra sharing it have no vertex in each others' circumspheres.

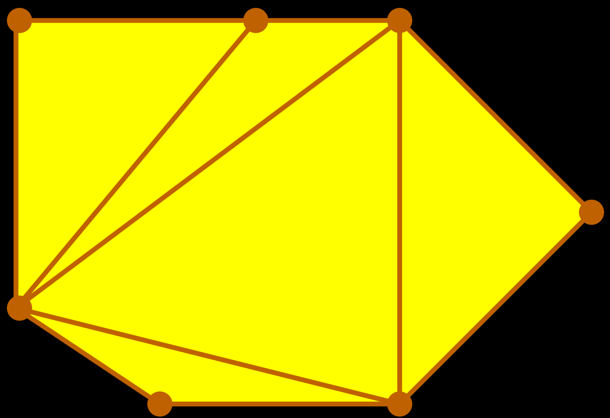


The Delaunay Triangulation

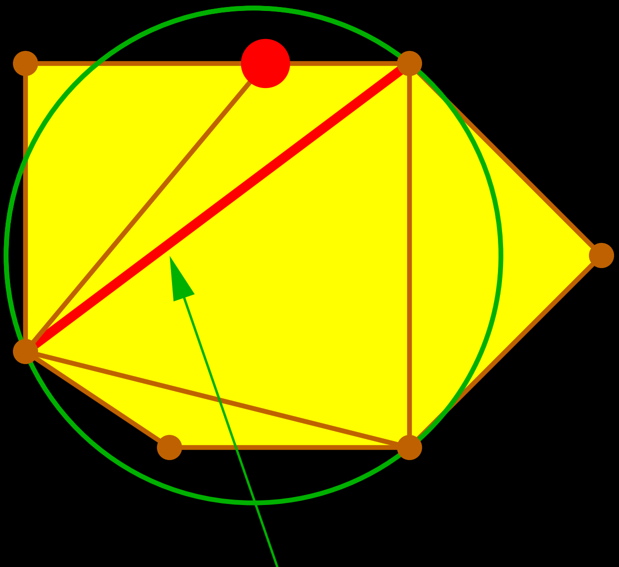
A *Delaunay triangulation* is a triangulation of a point set in which every edge (face in 3D) is locally Delaunay.



Lawson’s Flip Algorithm

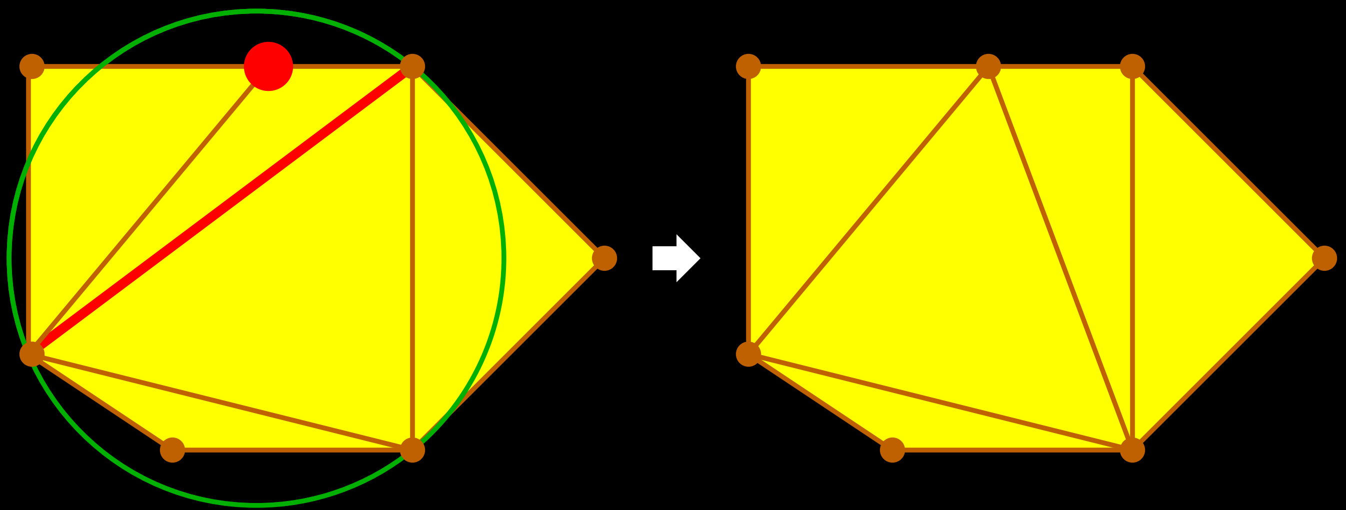


Lawson's Flip Algorithm

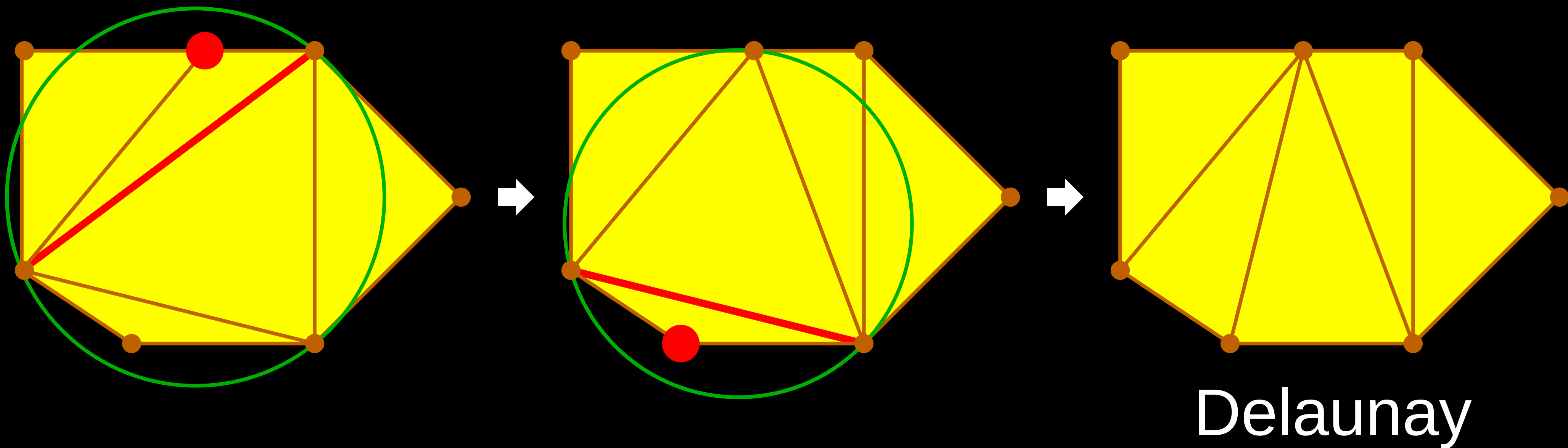


Not locally Delaunay

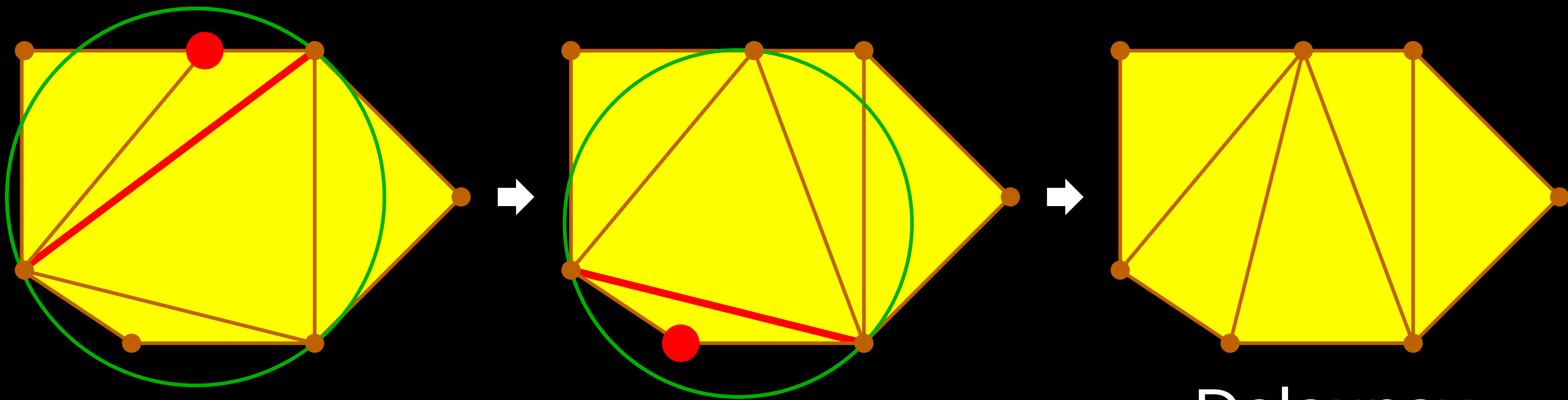
Lawson’s Flip Algorithm



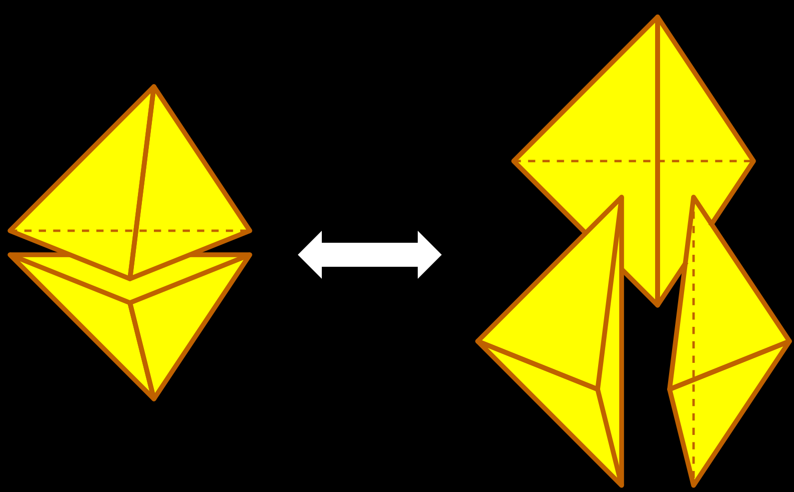
Lawson's Flip Algorithm



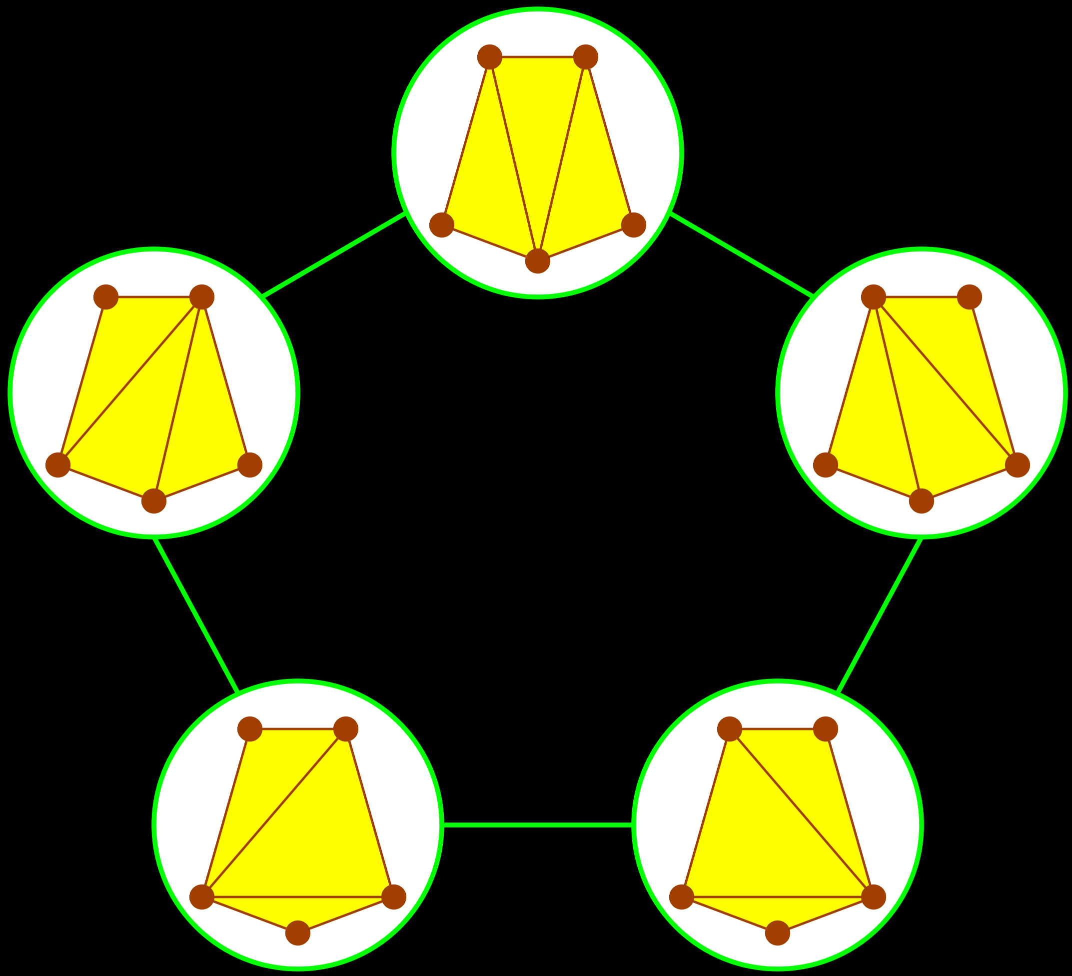
Lawson's Flip Algorithm



Delaunay



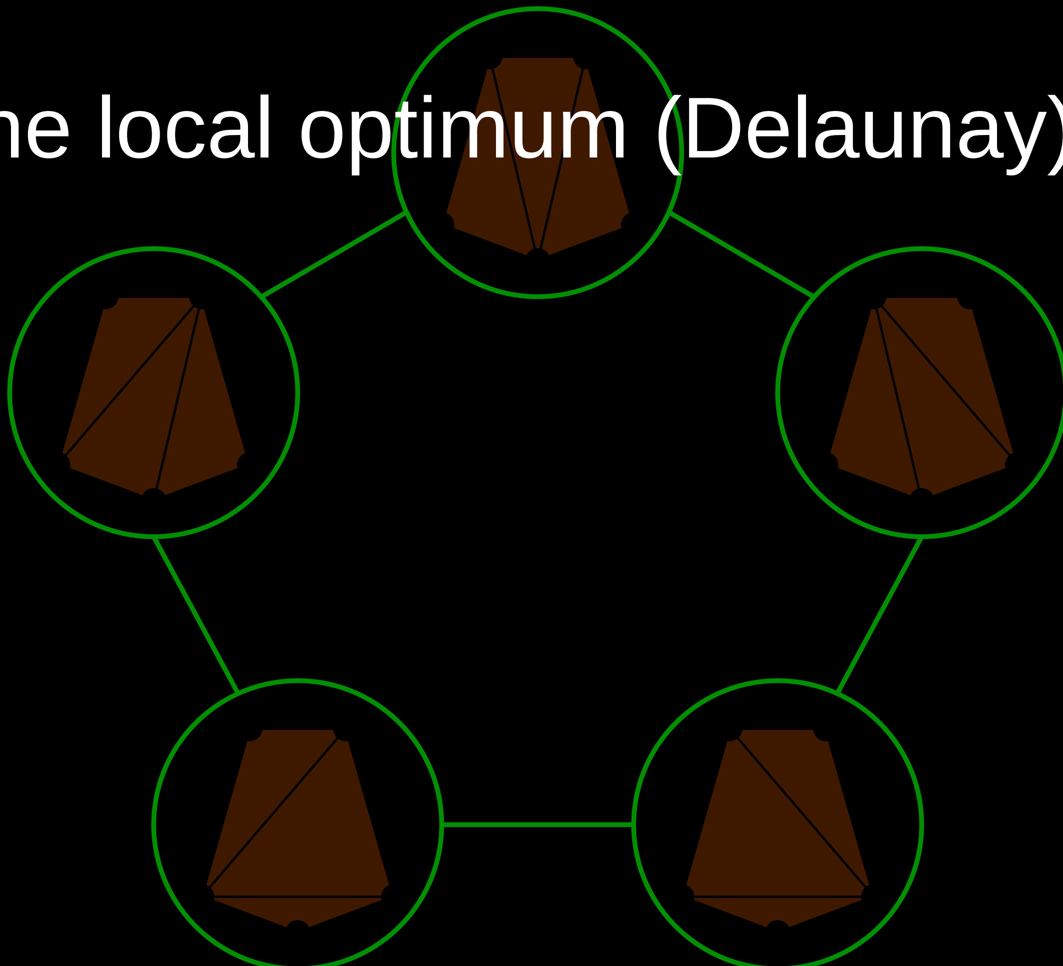
Flip Graph



Lawson’s flip algorithm performs hill−climbing optimization on the flip graph.

Flip Graph

2D: Only one local optimum (Delaunay).



Flip Graph

2D: Only one local optimum (Delaunay).

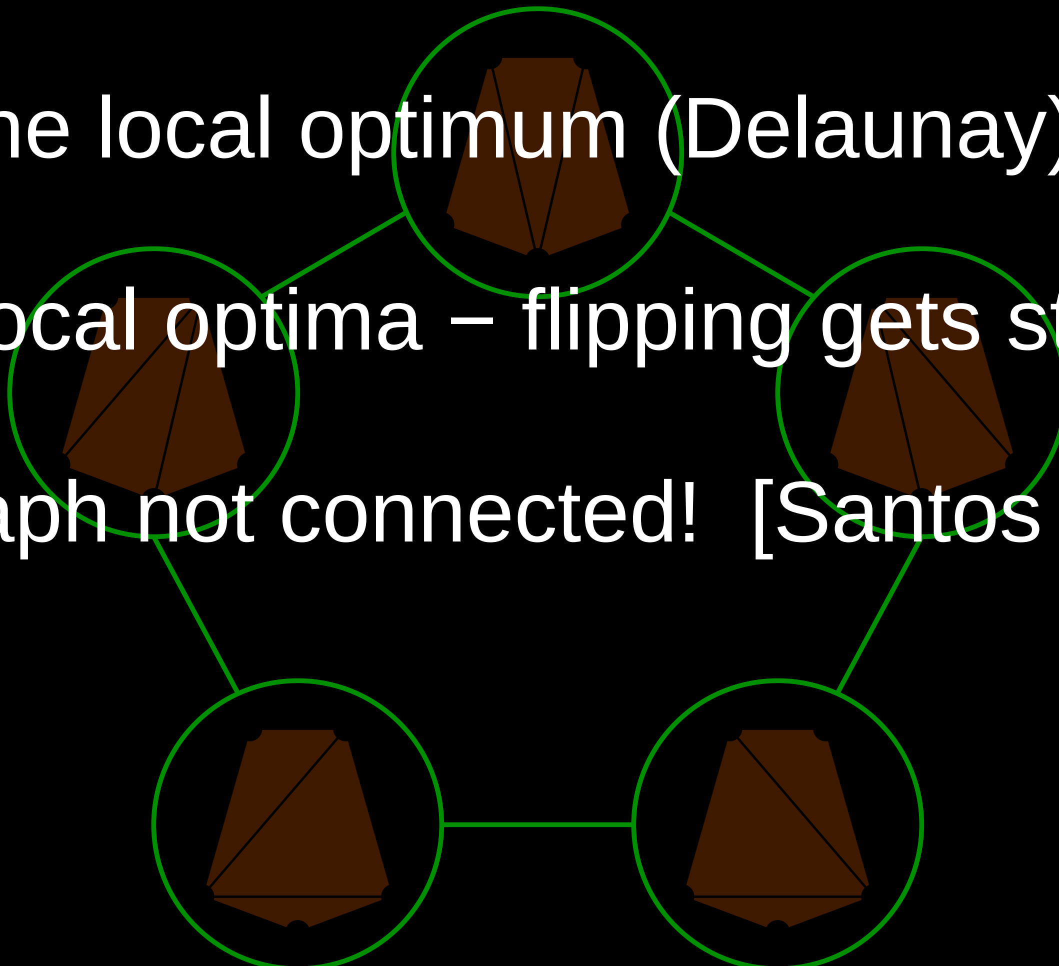
3D: Many local optima – flipping gets stuck!

Flip Graph

2D: Only one local optimum (Delaunay).

3D: Many local optima − flipping gets stuck!

5D: Flip graph not connected! [Santos 2004]



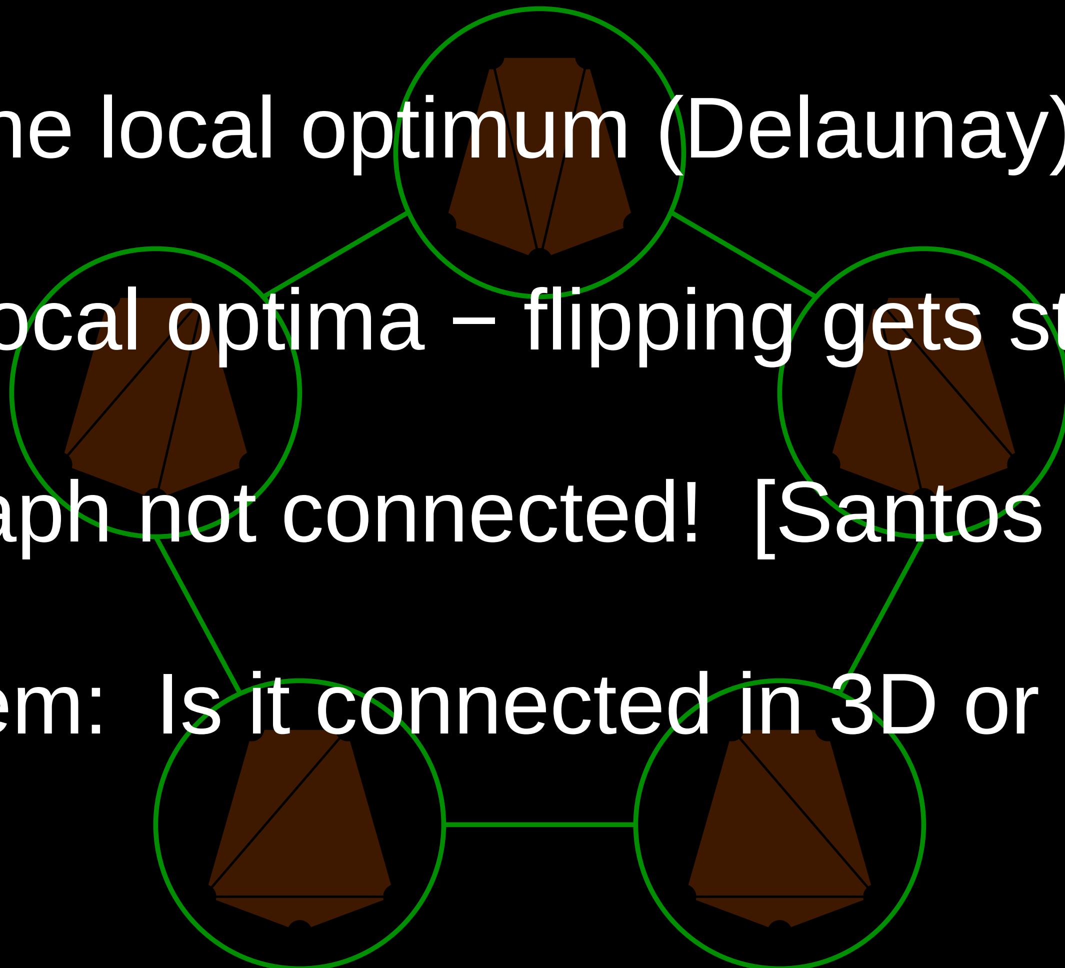
Flip Graph

2D: Only one local optimum (Delaunay).

3D: Many local optima – flipping gets stuck!

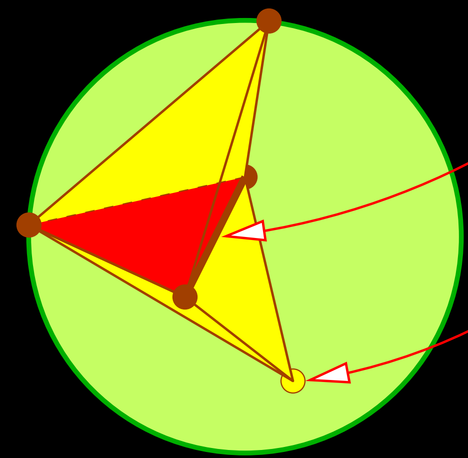
5D: Flip graph not connected! [Santos 2004]

Open problem: Is it connected in 3D or 4D?



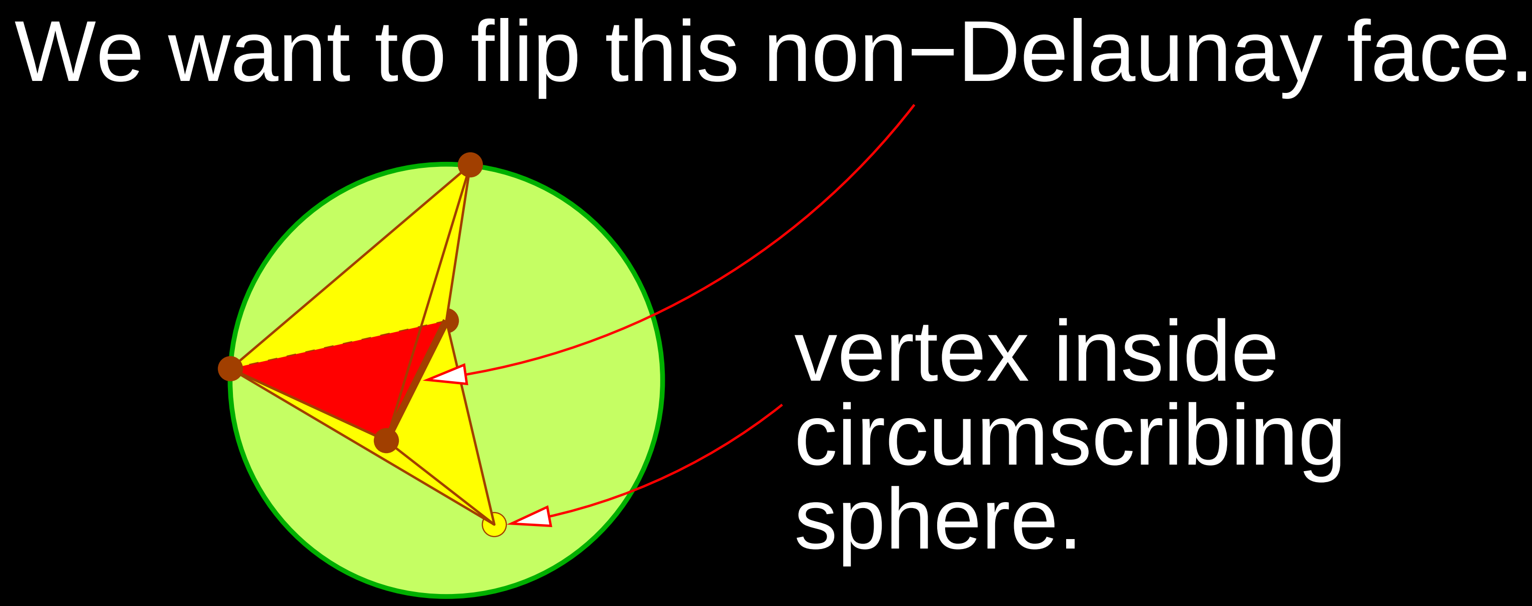
3D Delaunay Flips Can Get “Stuck”

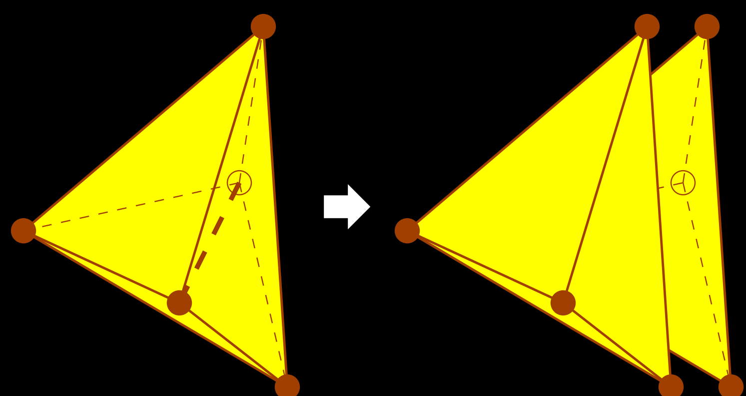
We want to flip this non−Delaunay face.



vertex inside
circumscribing
sphere.

3D Delaunay Flips Can Get “Stuck”

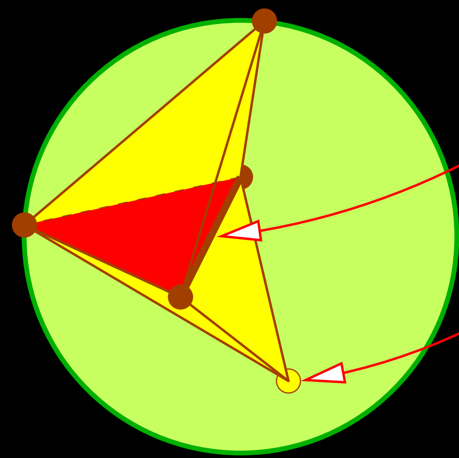




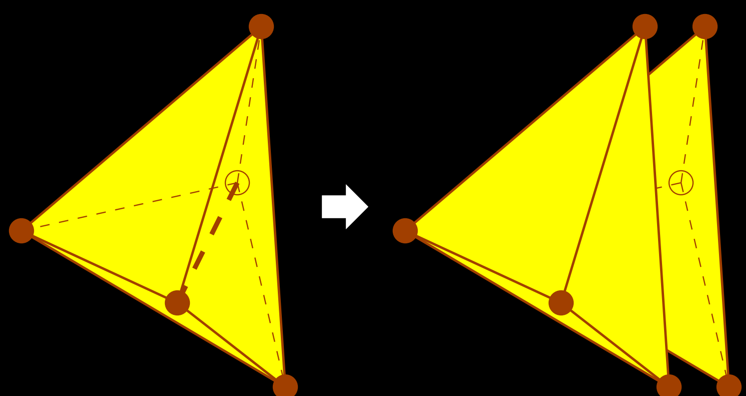
Sometimes we can.

3D Delaunay Flips Can Get “Stuck”

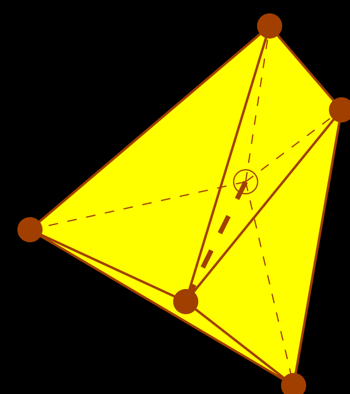
We want to flip this non−Delaunay face.



vertex inside circumscribing sphere.

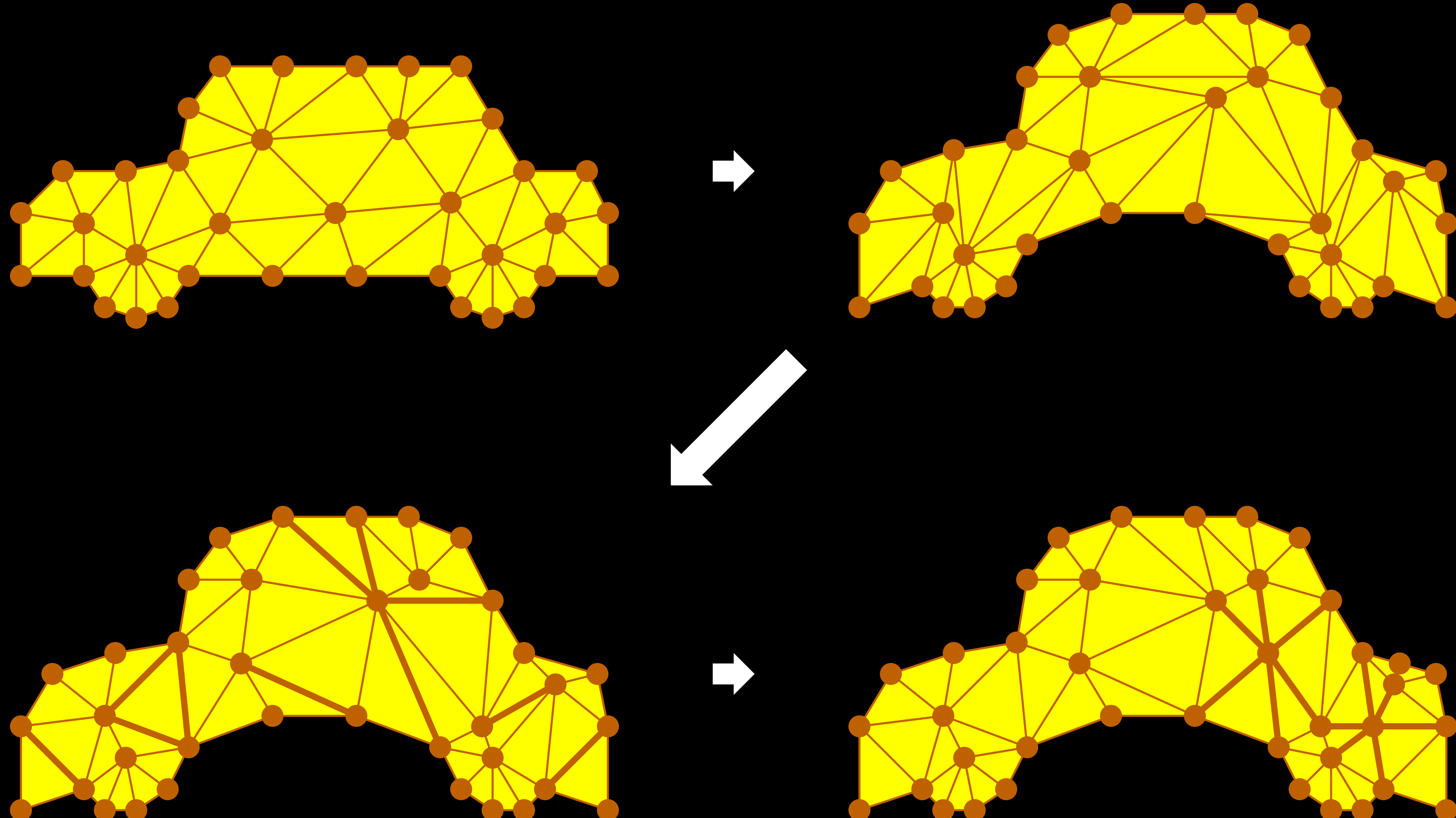


Sometimes we can.



Sometimes we can’t.

Application: 3D Dynamic Meshing



Delaunay repair   Mesh improvement

3D Delaunay Repair

Guibas and Russel [2004]: It’s faster to flip than to recompute the 3D Delaunay triangulation from scratch...

3D Delaunay Repair

Guibas and Russel [2004]: It’s faster to flip than to recompute the 3D Delaunay triangulation from scratch...

...when flipping doesn’t get stuck.

My Question

How much more is possible if we bend the usual rules of flipping?

Star Splaying

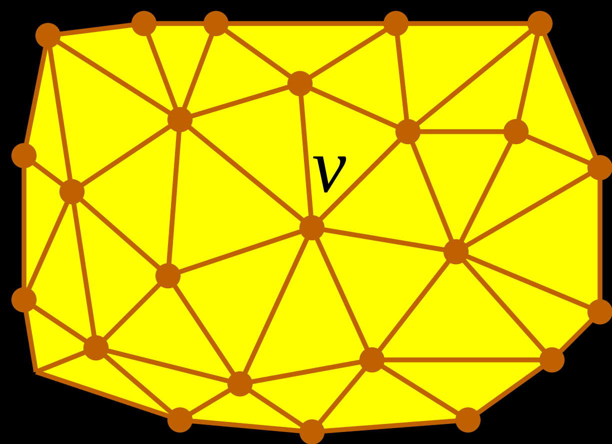
- A hill−climbing optimization algorithm for Delaunay repair (in any dimension) that never gets stuck.

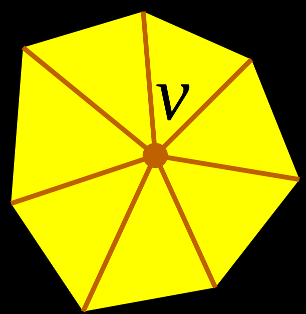
Star Splaying

- ✫ A hill−climbing optimization algorithm for Delaunay repair (in any dimension) that never gets stuck.
- ✫ Recommendation: try flipping first; let star splaying take over if flipping gets stuck.

Stars, Rays, & Cones

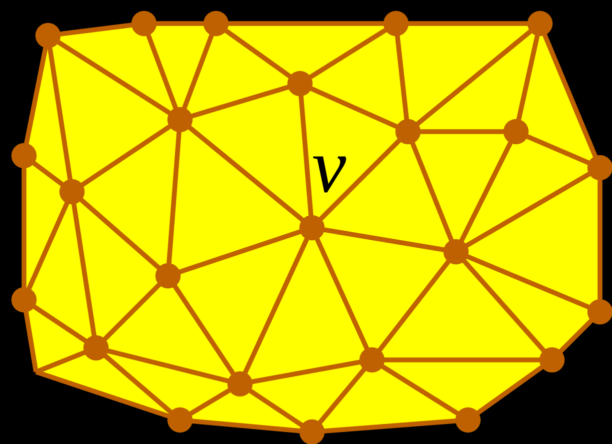
Definition: The Star of a Vertex

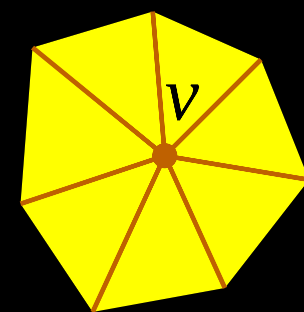




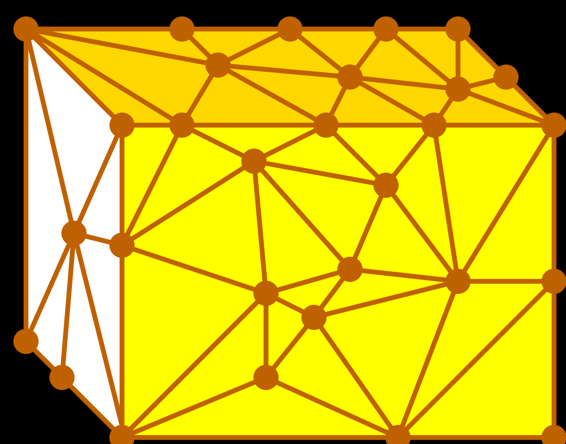
star of v

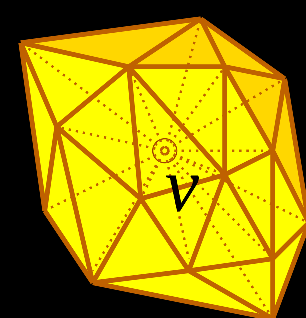
Definition: The Star of a Vertex





star of *v*

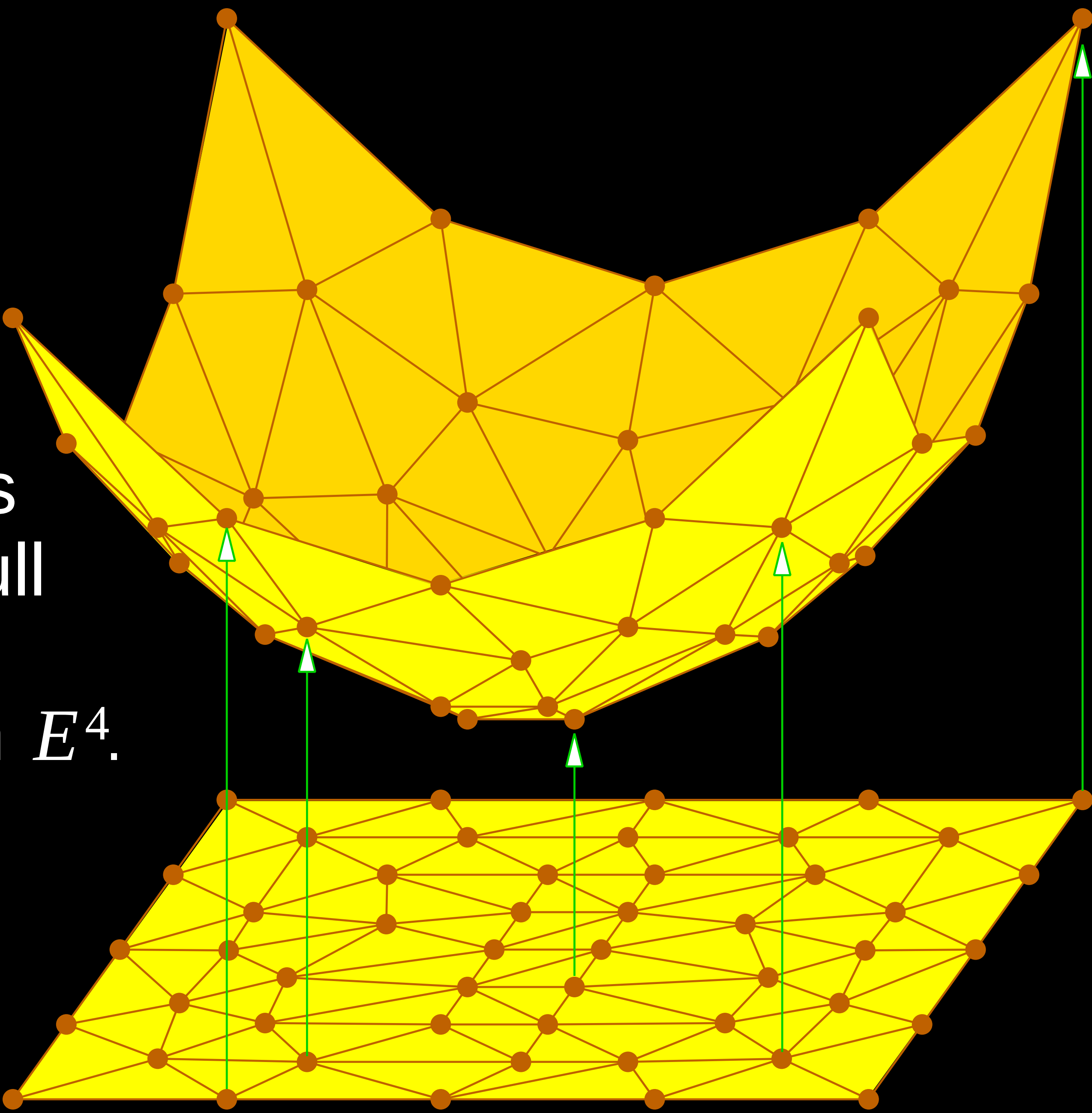




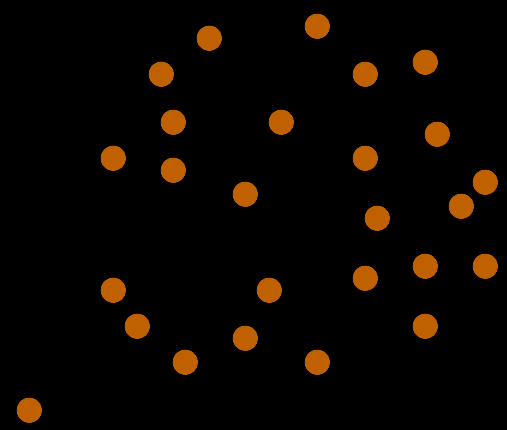
star of *v*

Seidel’s Parabolic Lifting Map

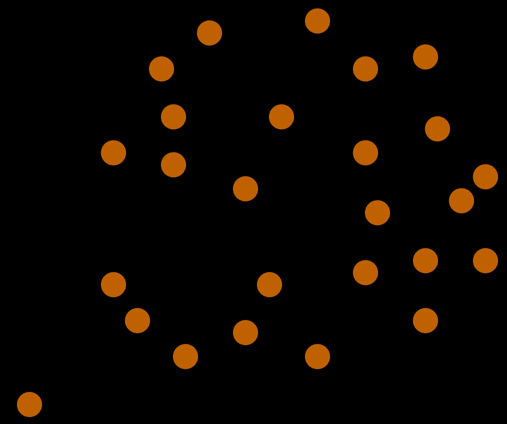
The 3D DT matches the lower convex hull of the vertices lifted onto a paraboloid in E^4.

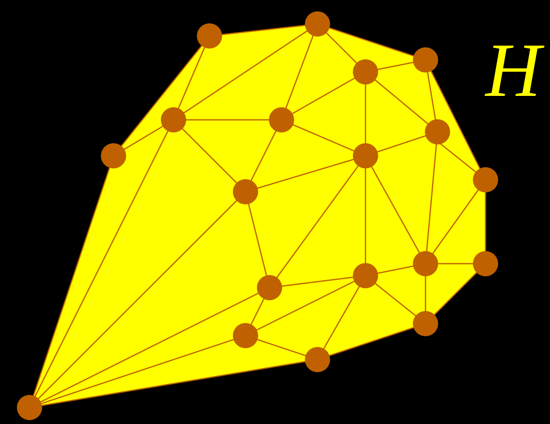


Stars, Rays, and Cones

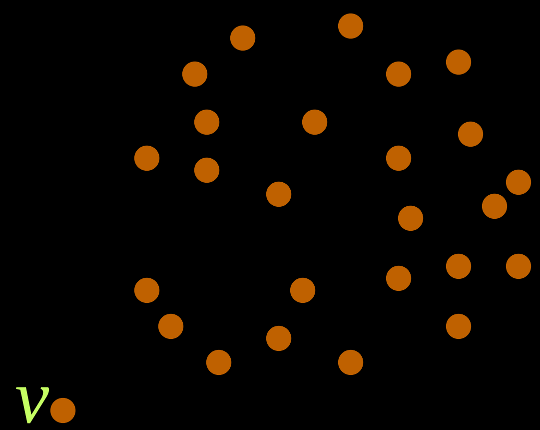


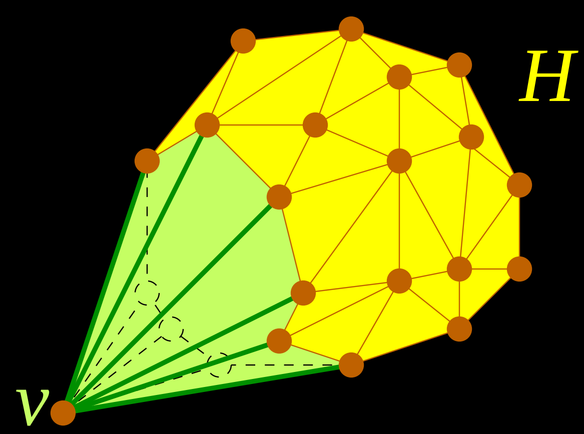
Stars, Rays, and Cones



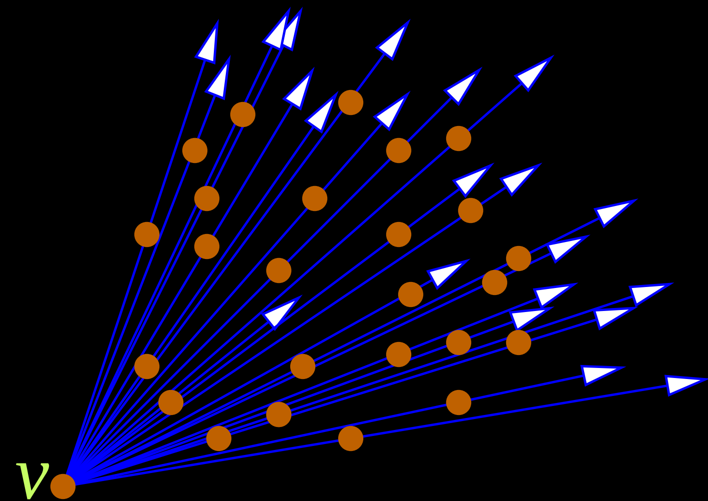


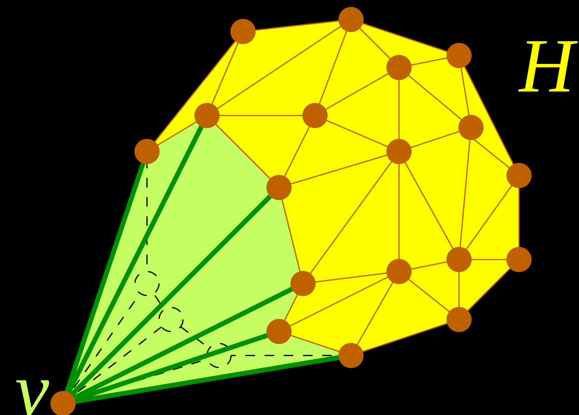
Stars, Rays, and Cones



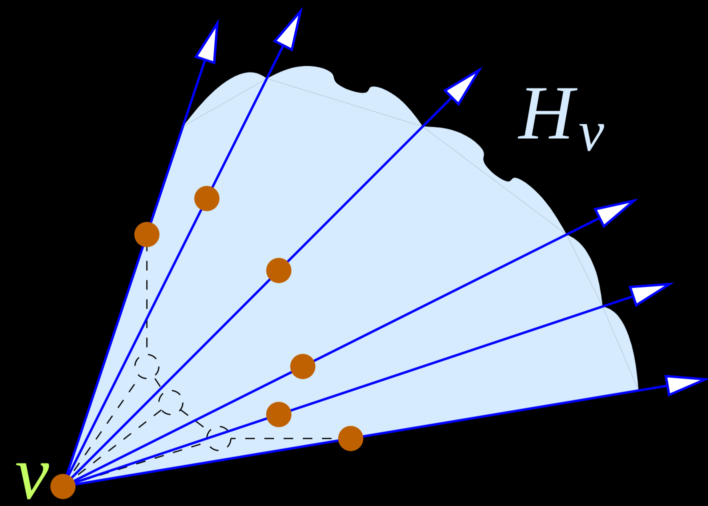


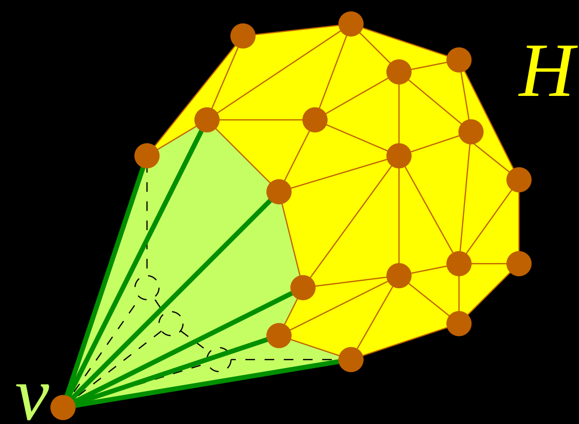
Stars, Rays, and Cones



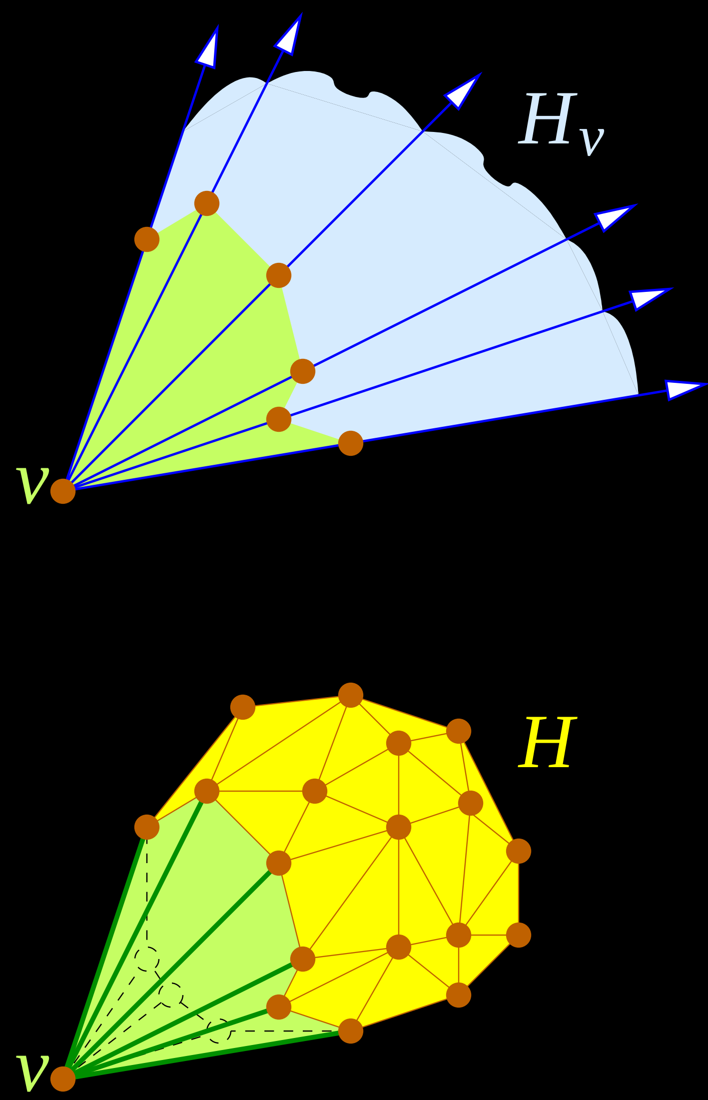


Stars, Rays, and Cones

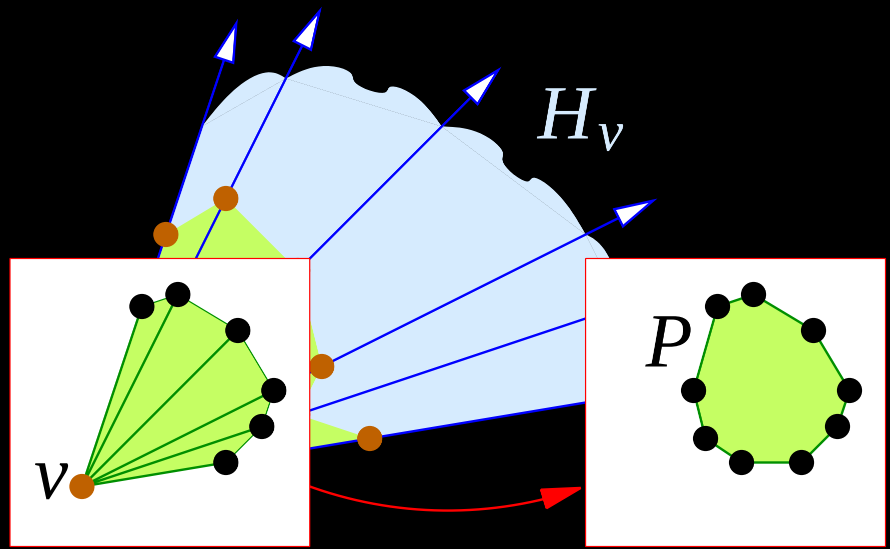


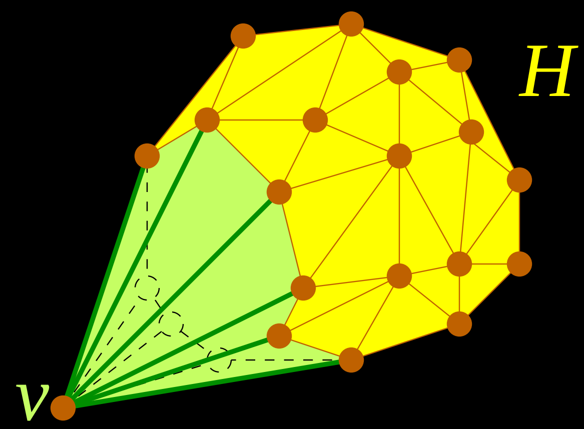


Stars, Rays, and Cones



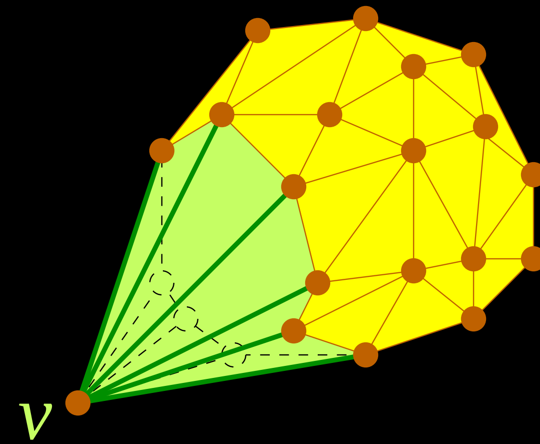
Stars, Rays, and Cones



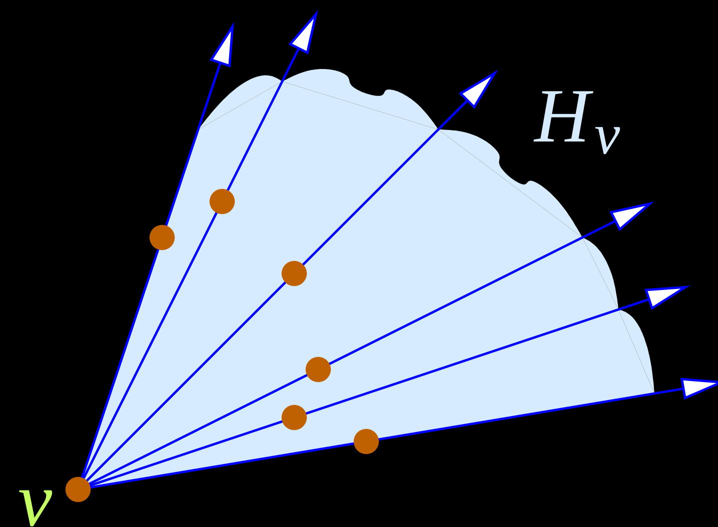


3 Equivalent Computations

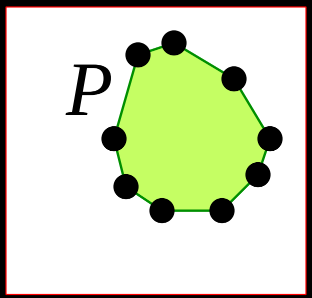
☆ The star of v on a 4D convex hull of points.



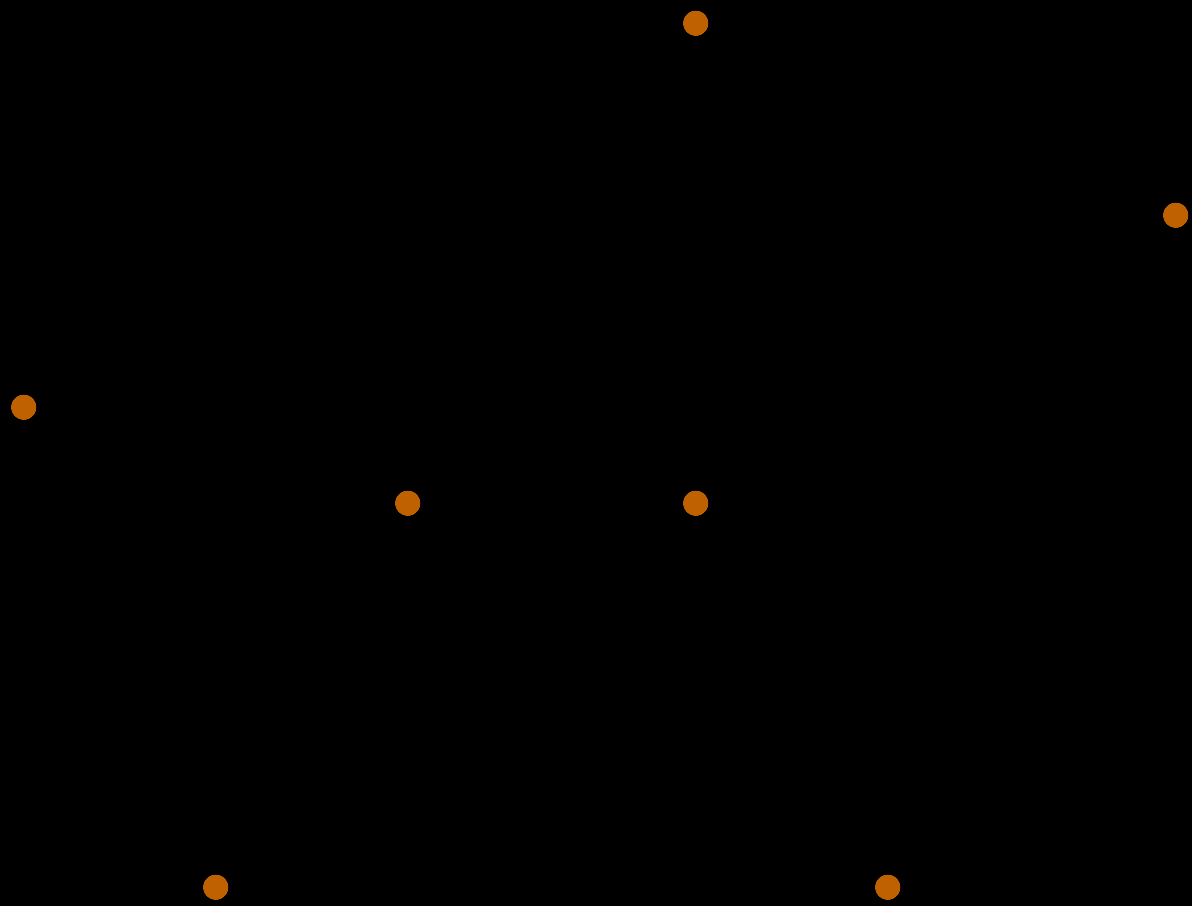
☆ The 4D cone H_v (convex hull of rays).



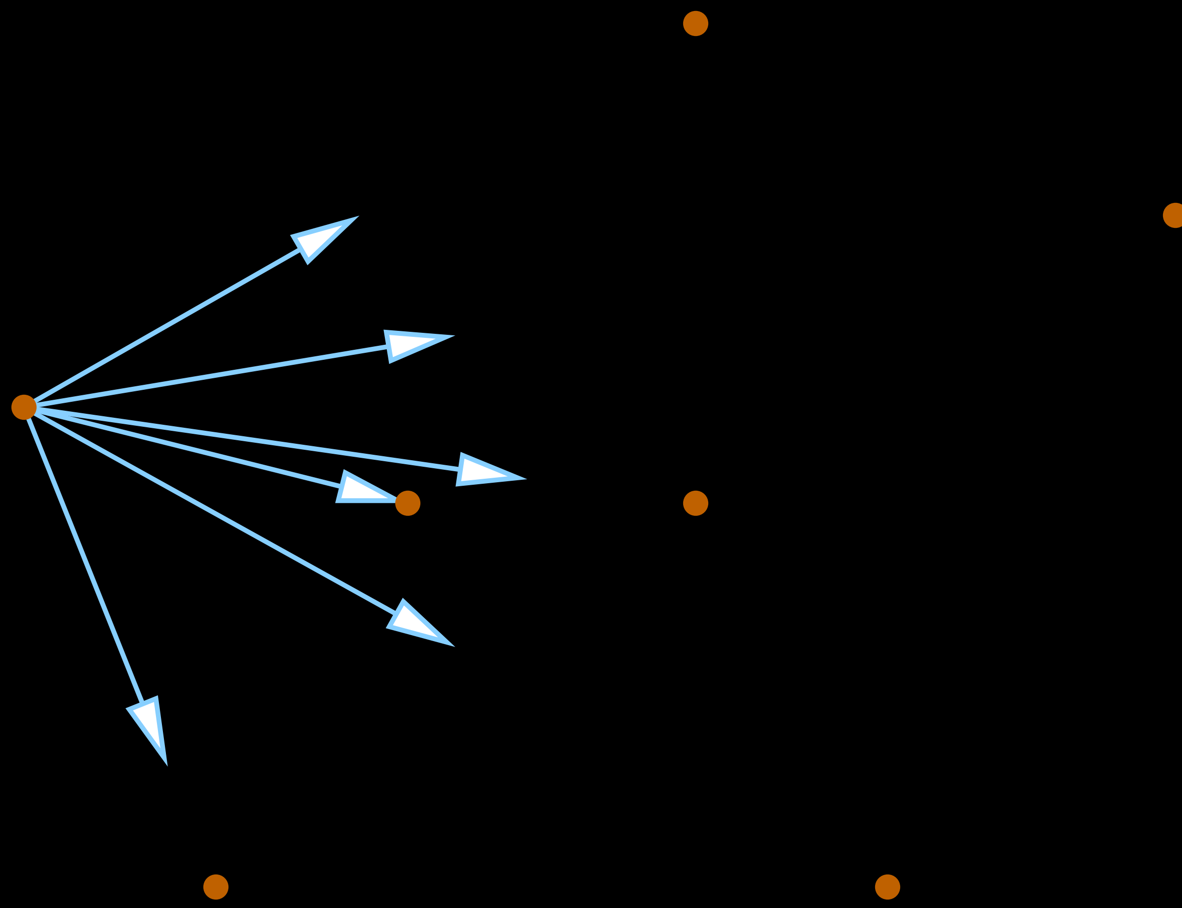
☆ The 3D polytope P (convex hull of points).



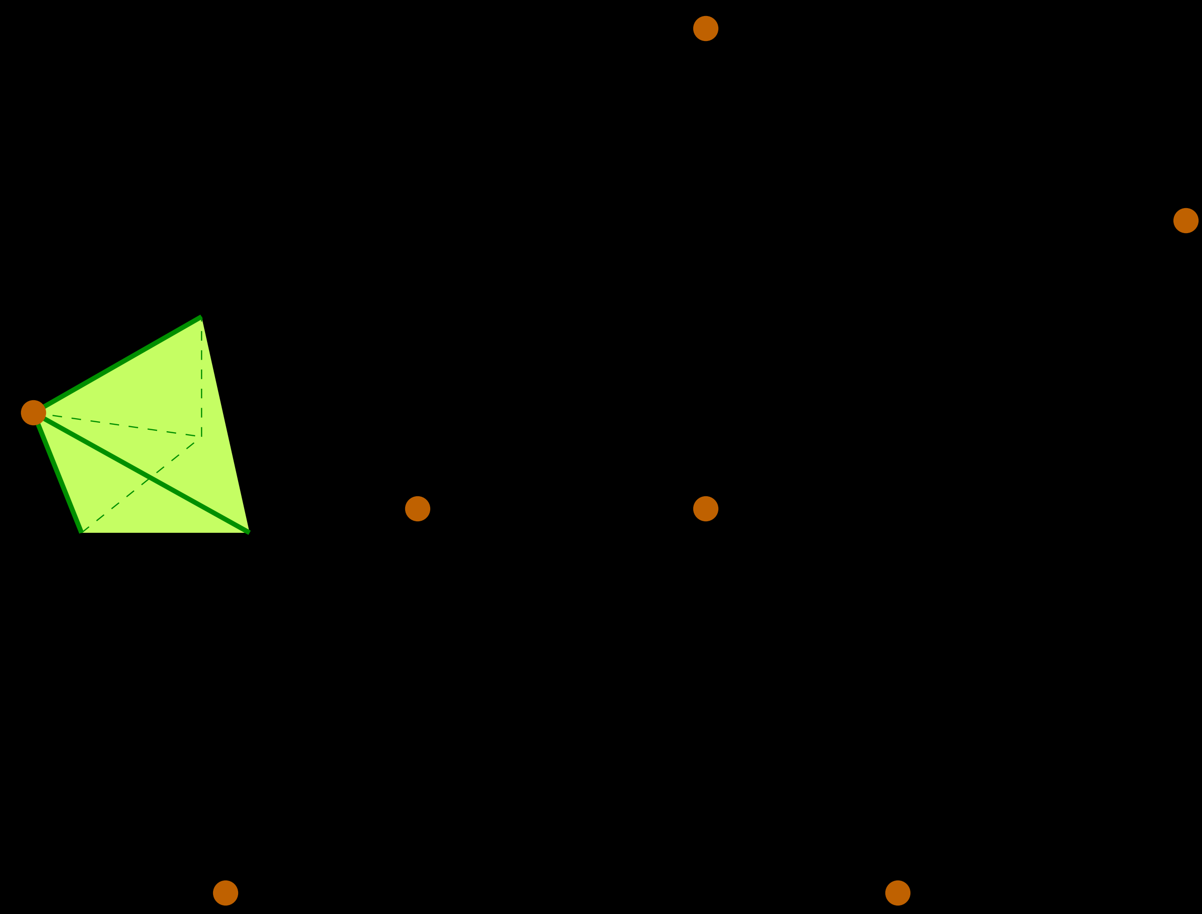
Computing the Convex Hull



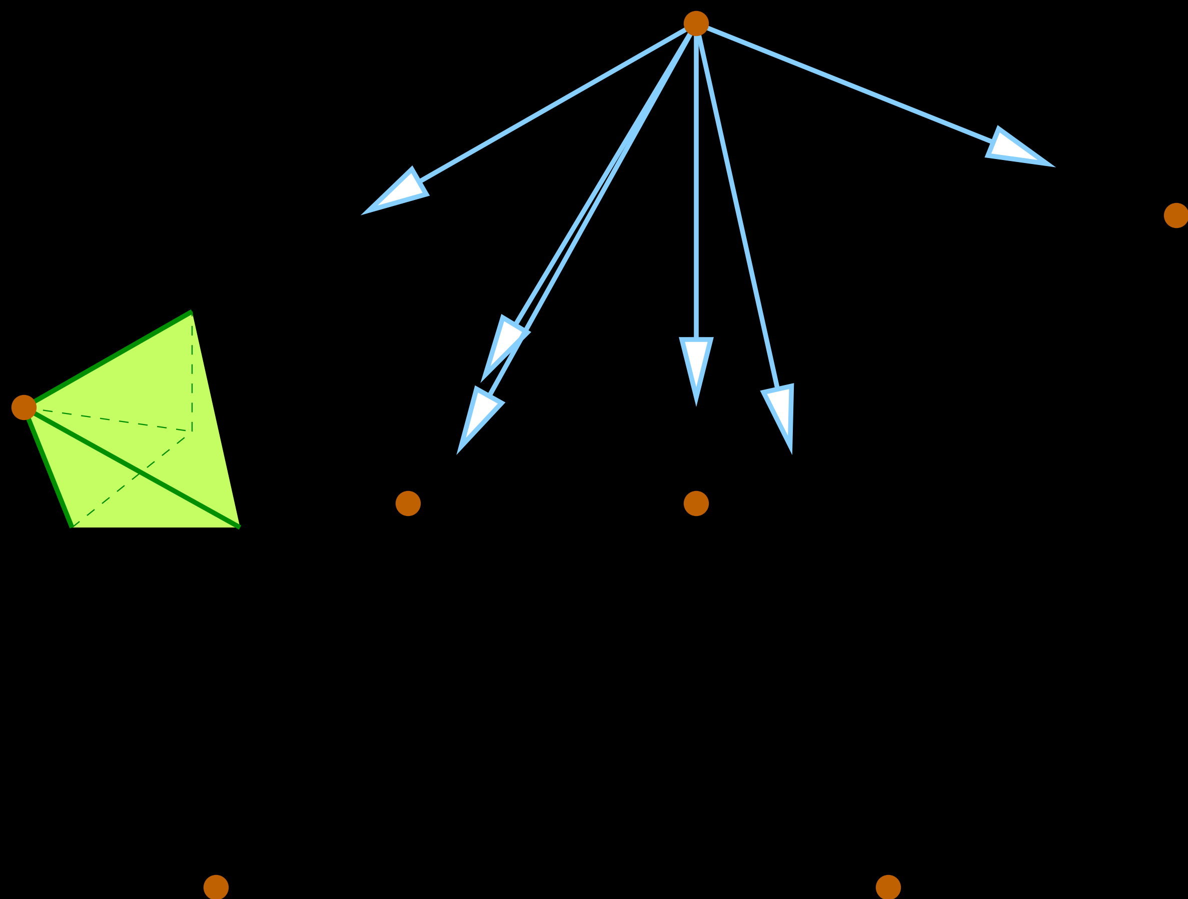
Computing the Convex Hull



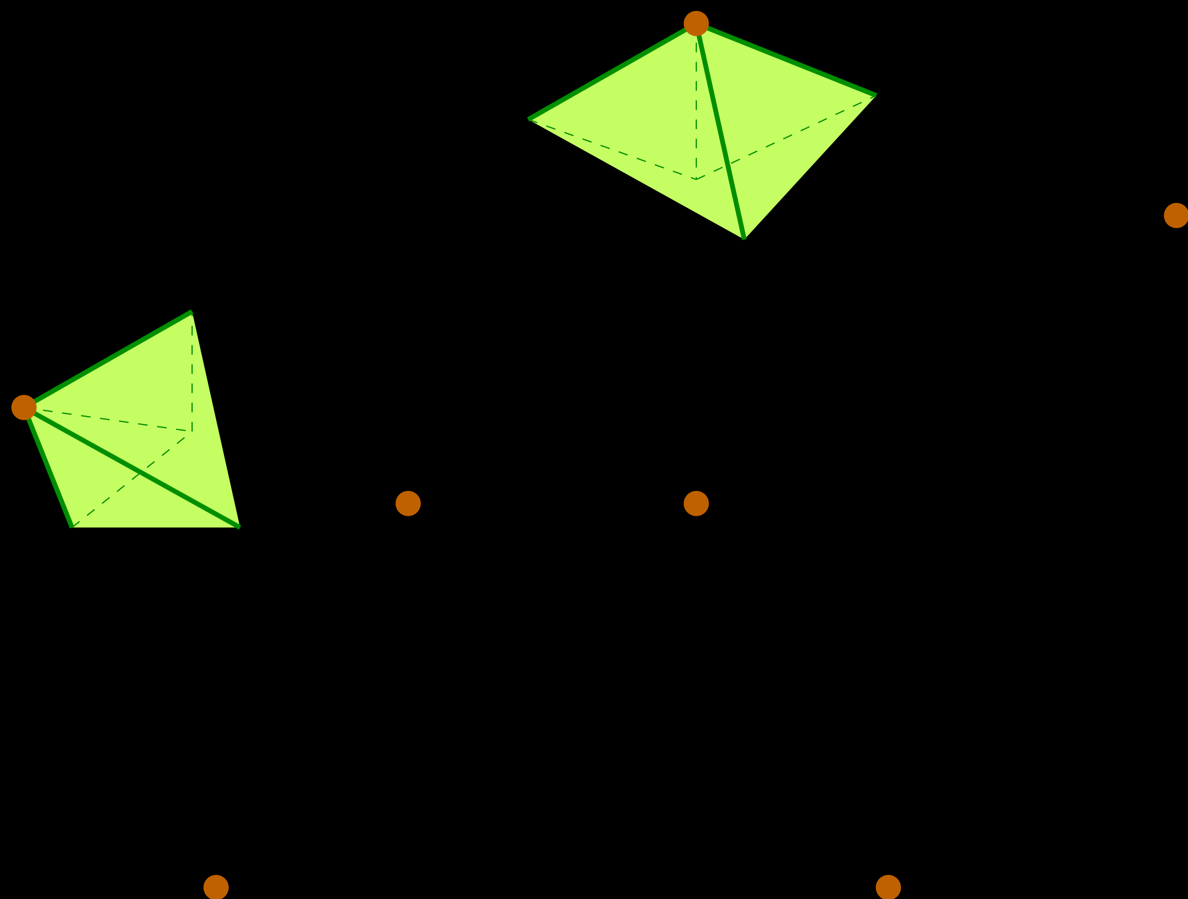
Computing the Convex Hull



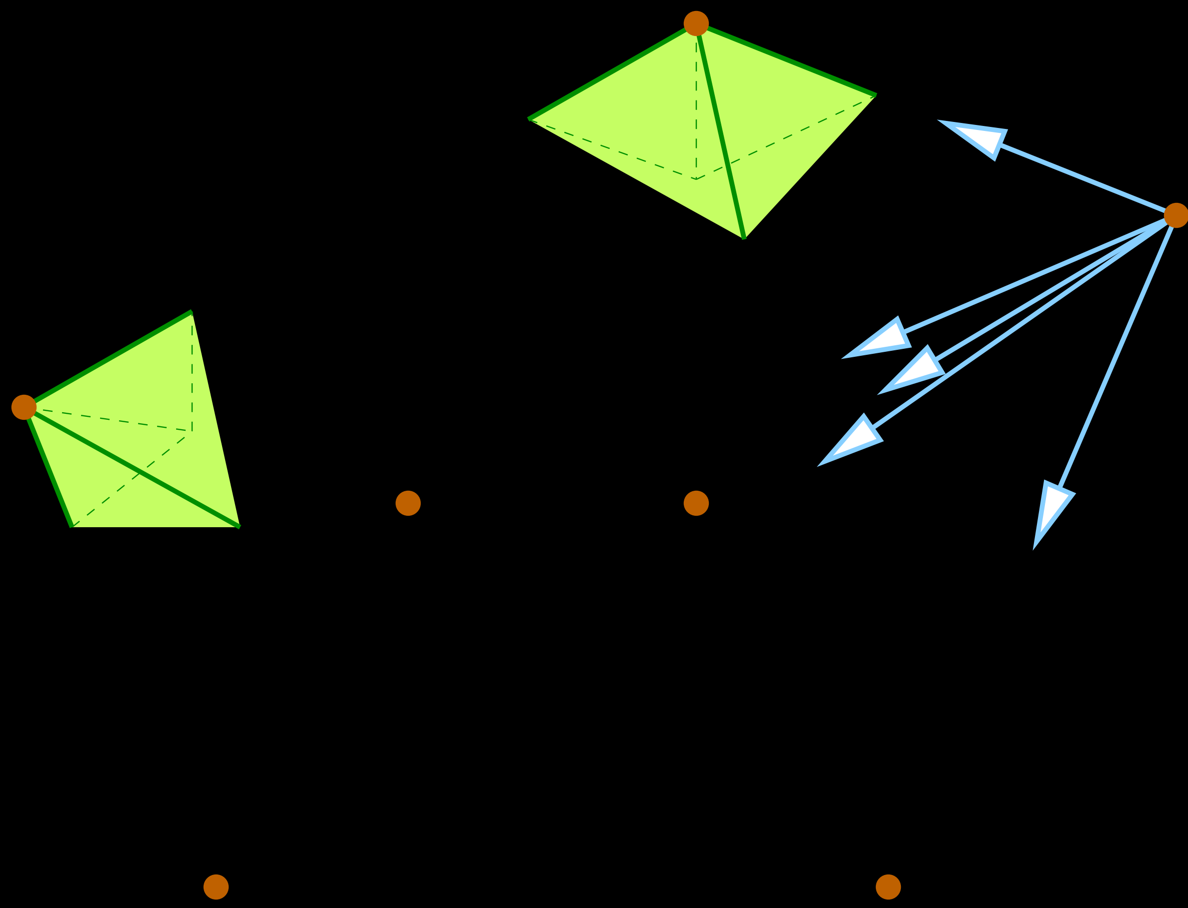
Computing the Convex Hull



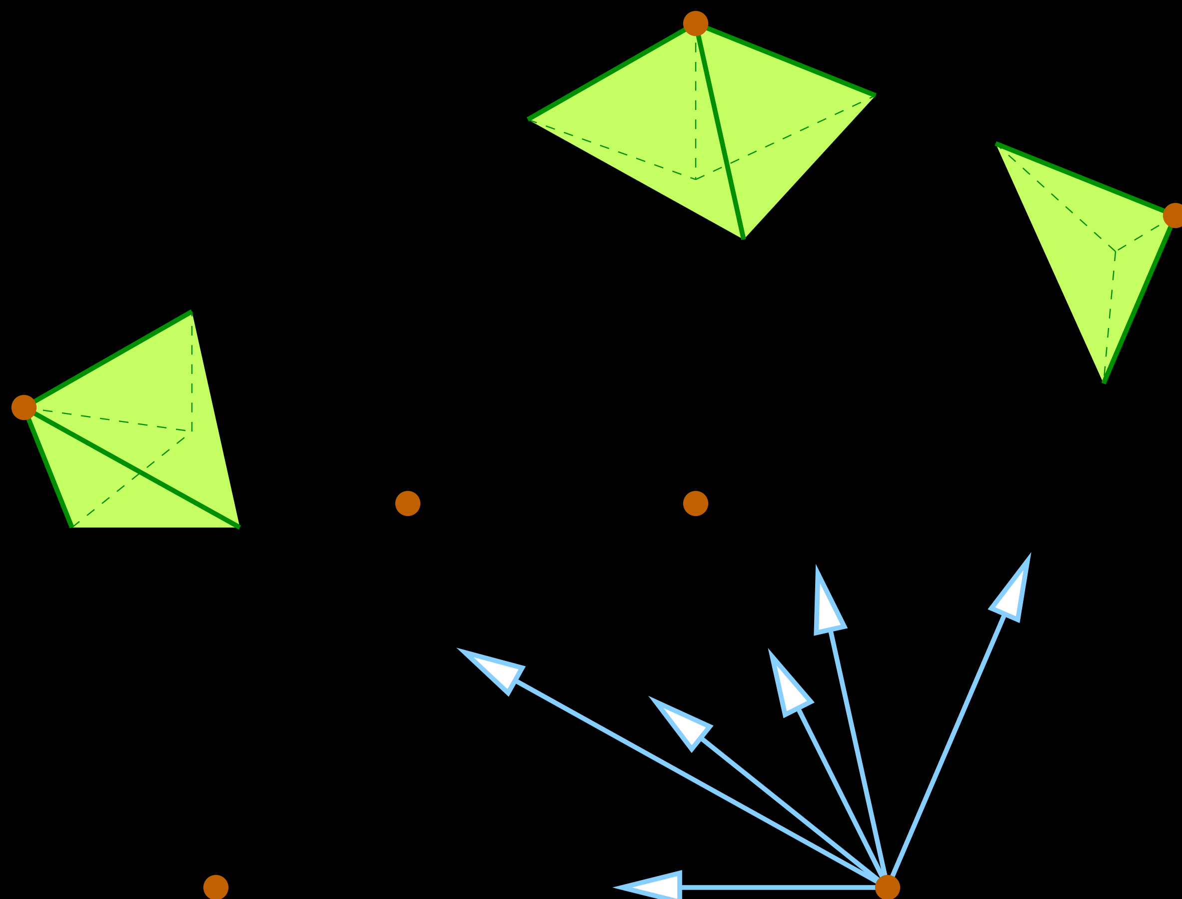
Computing the Convex Hull



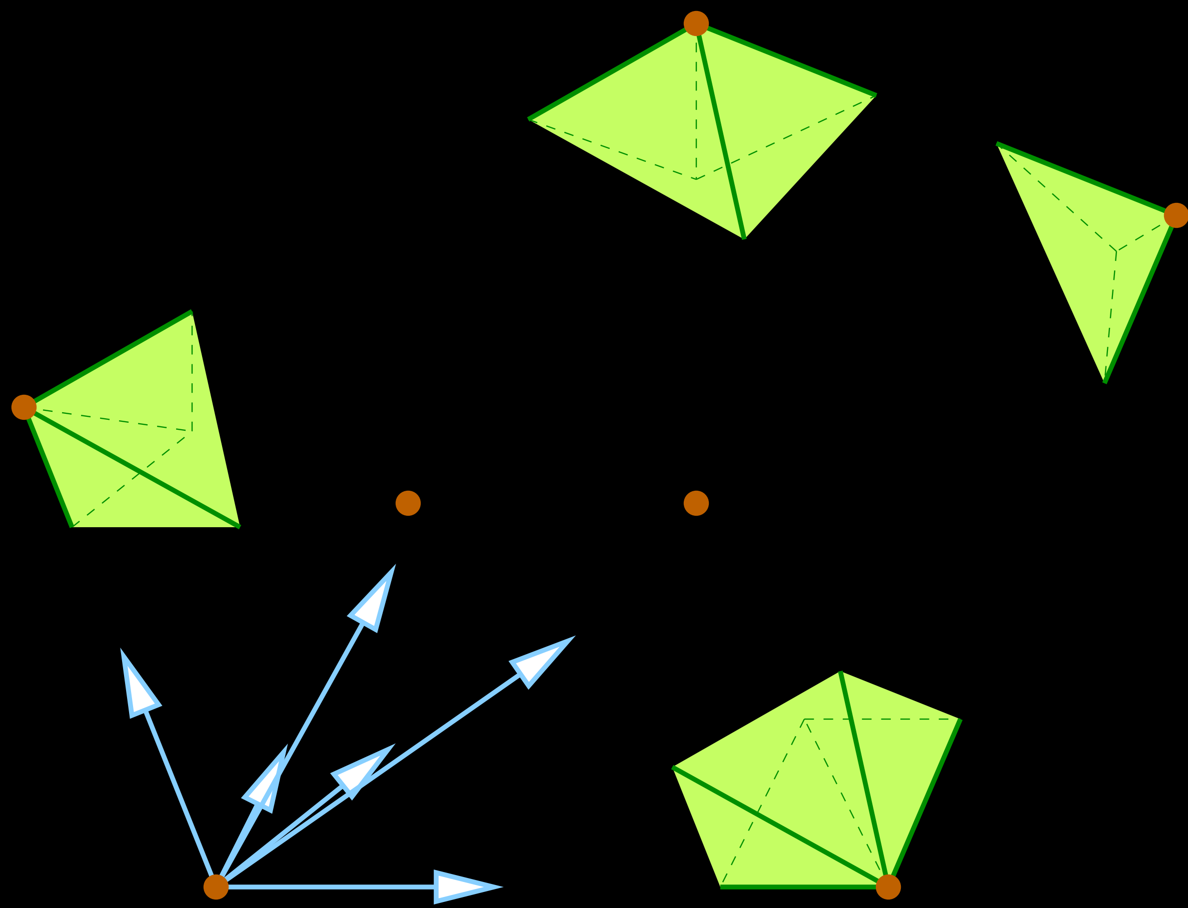
Computing the Convex Hull



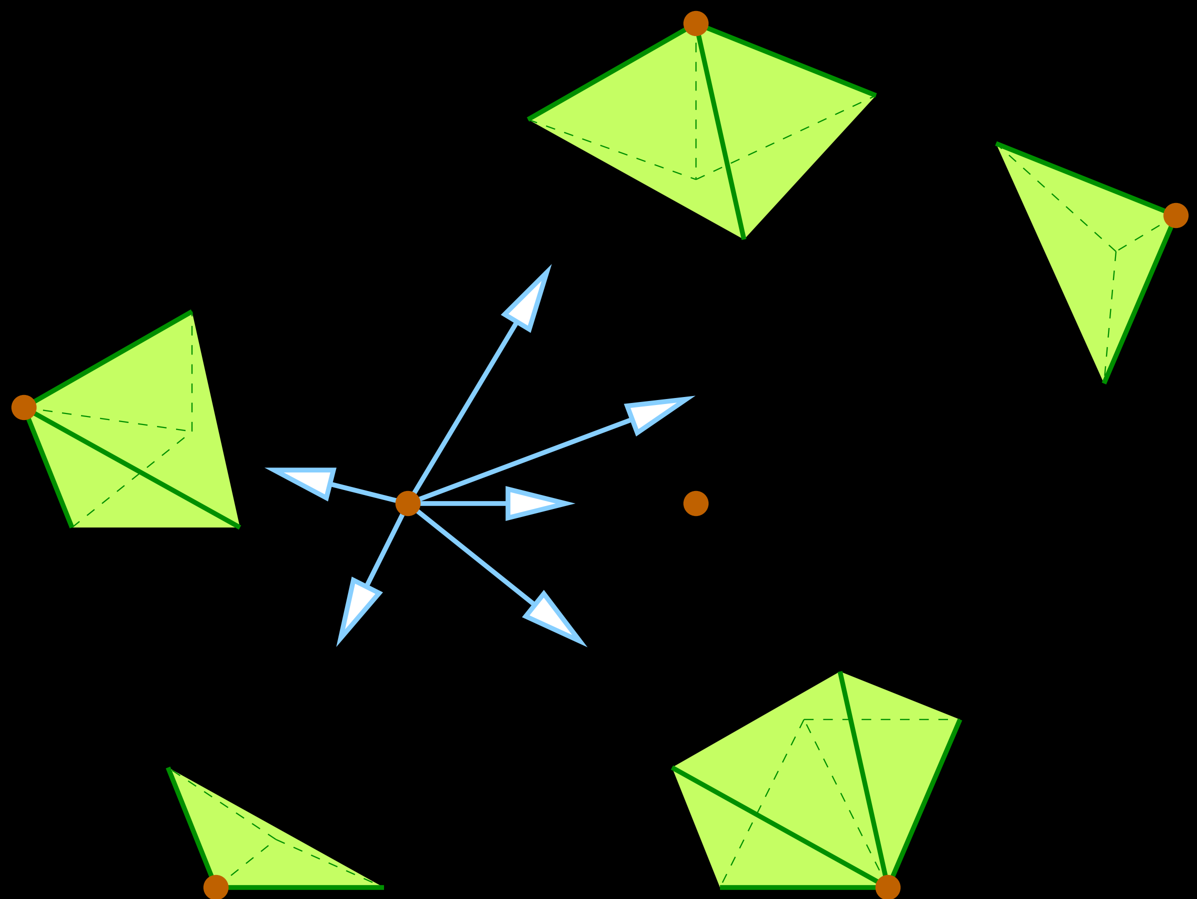
Computing the Convex Hull



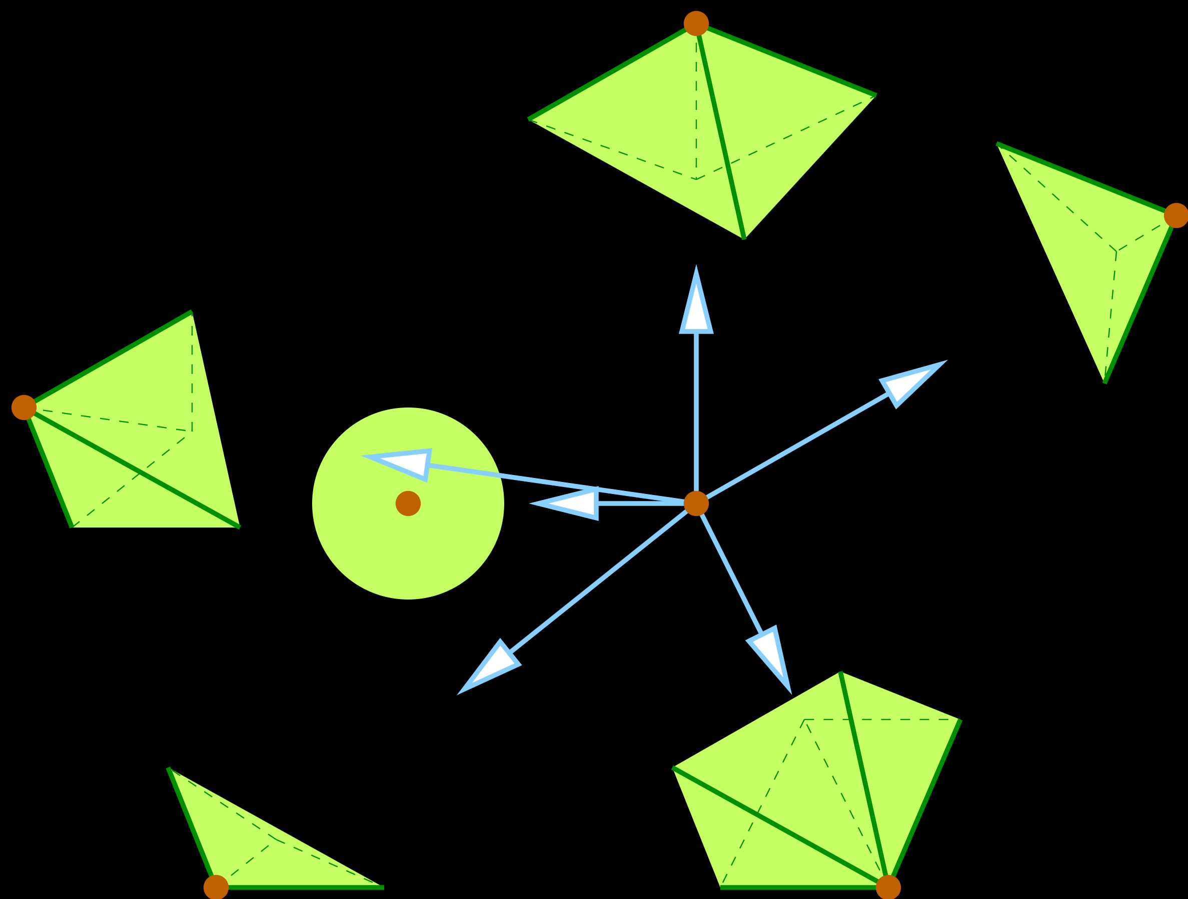
Computing the Convex Hull



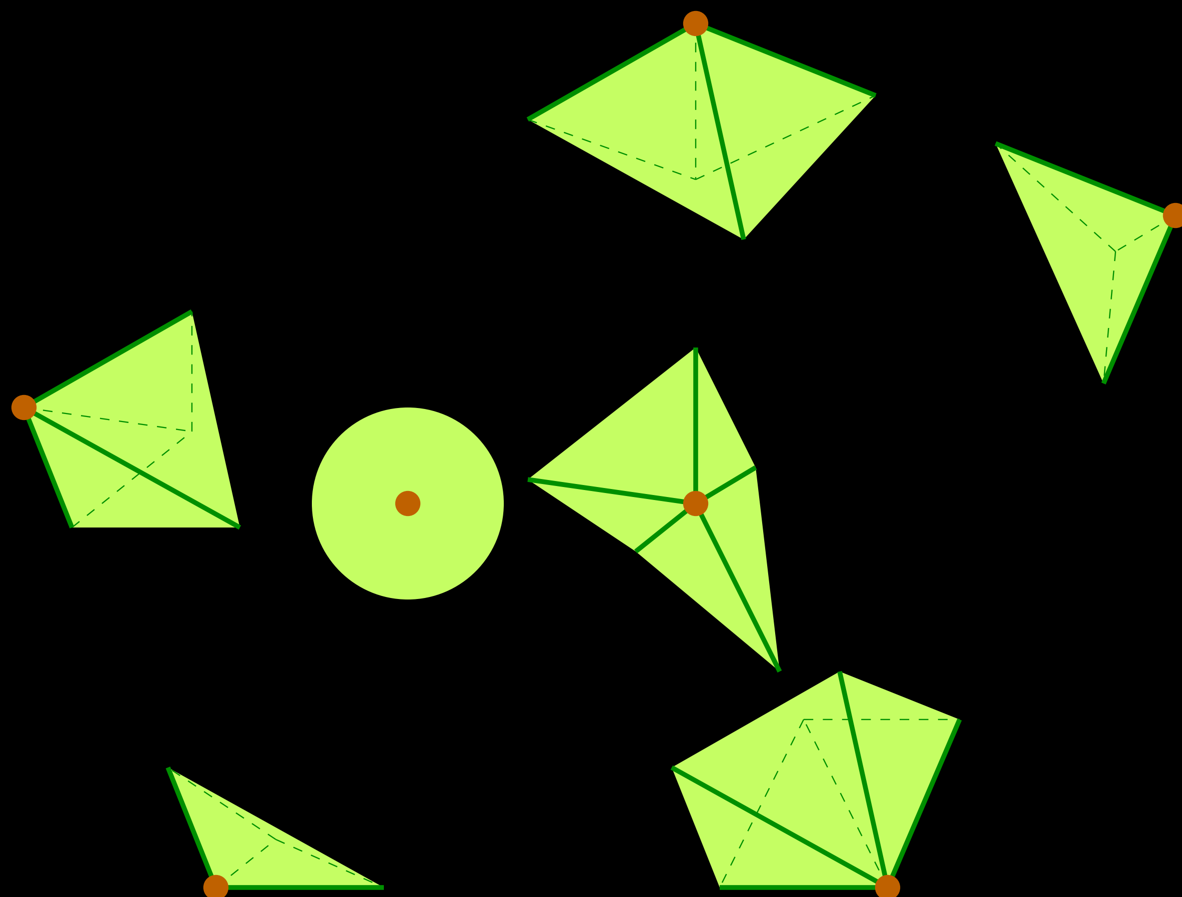
Computing the Convex Hull



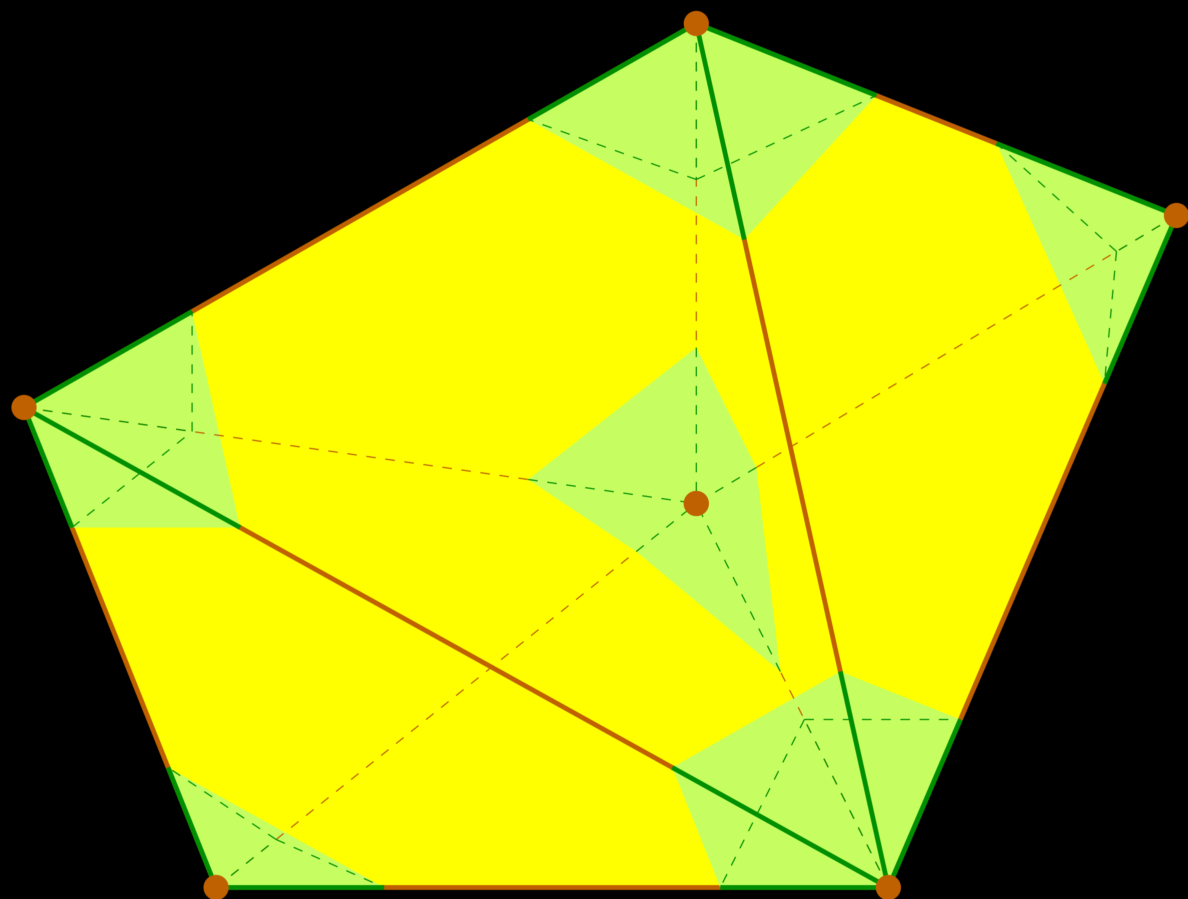
Computing the Convex Hull



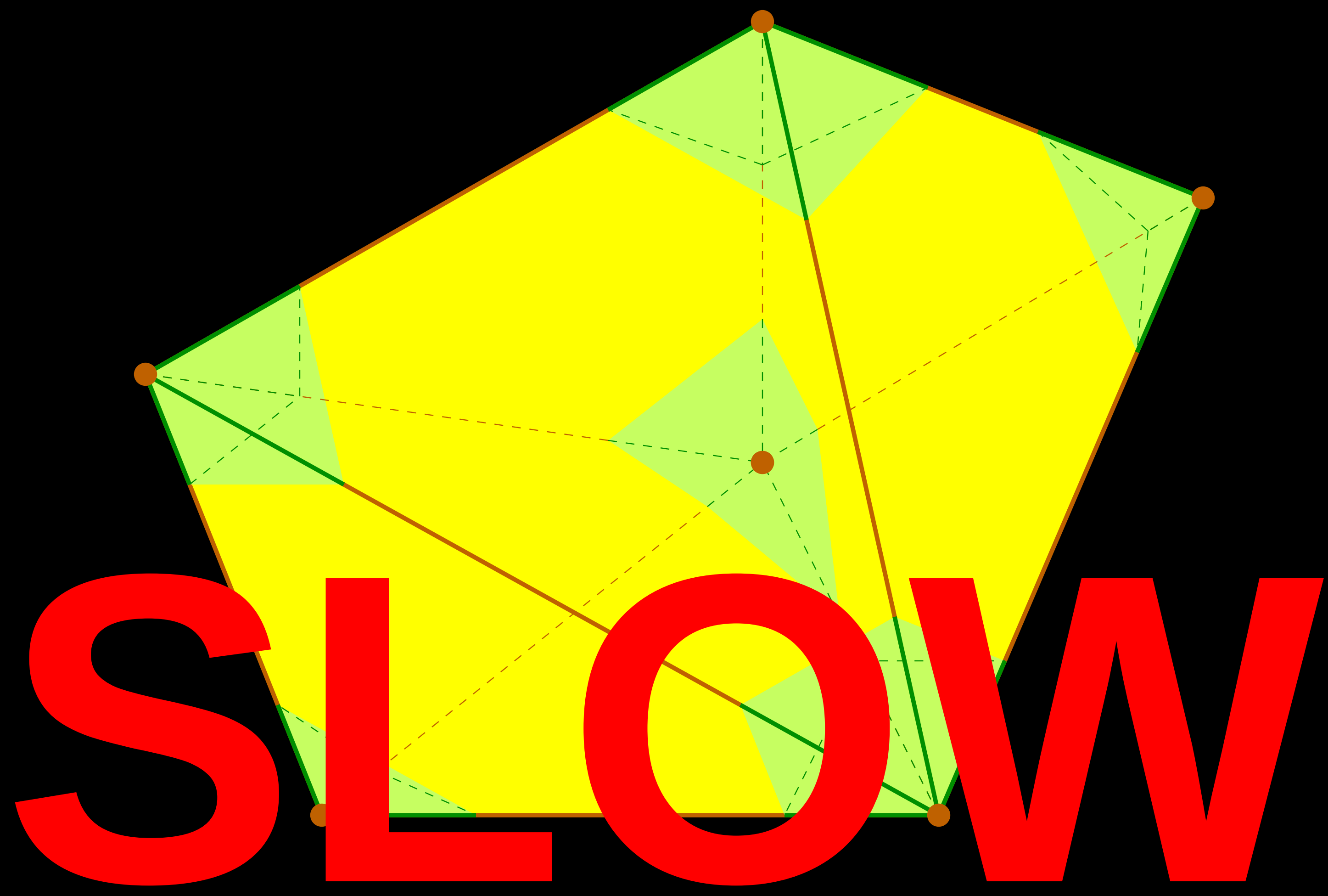
Computing the Convex Hull



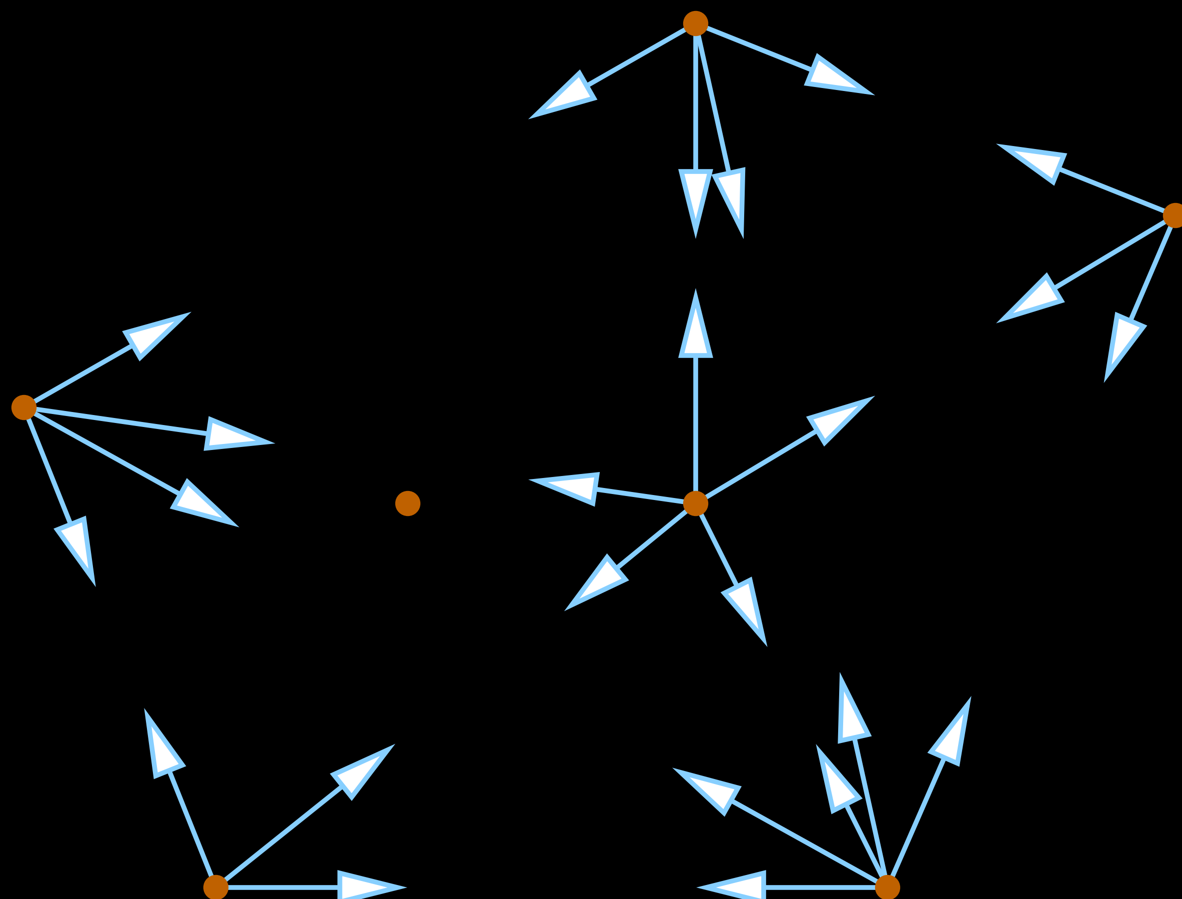
Computing the Convex Hull



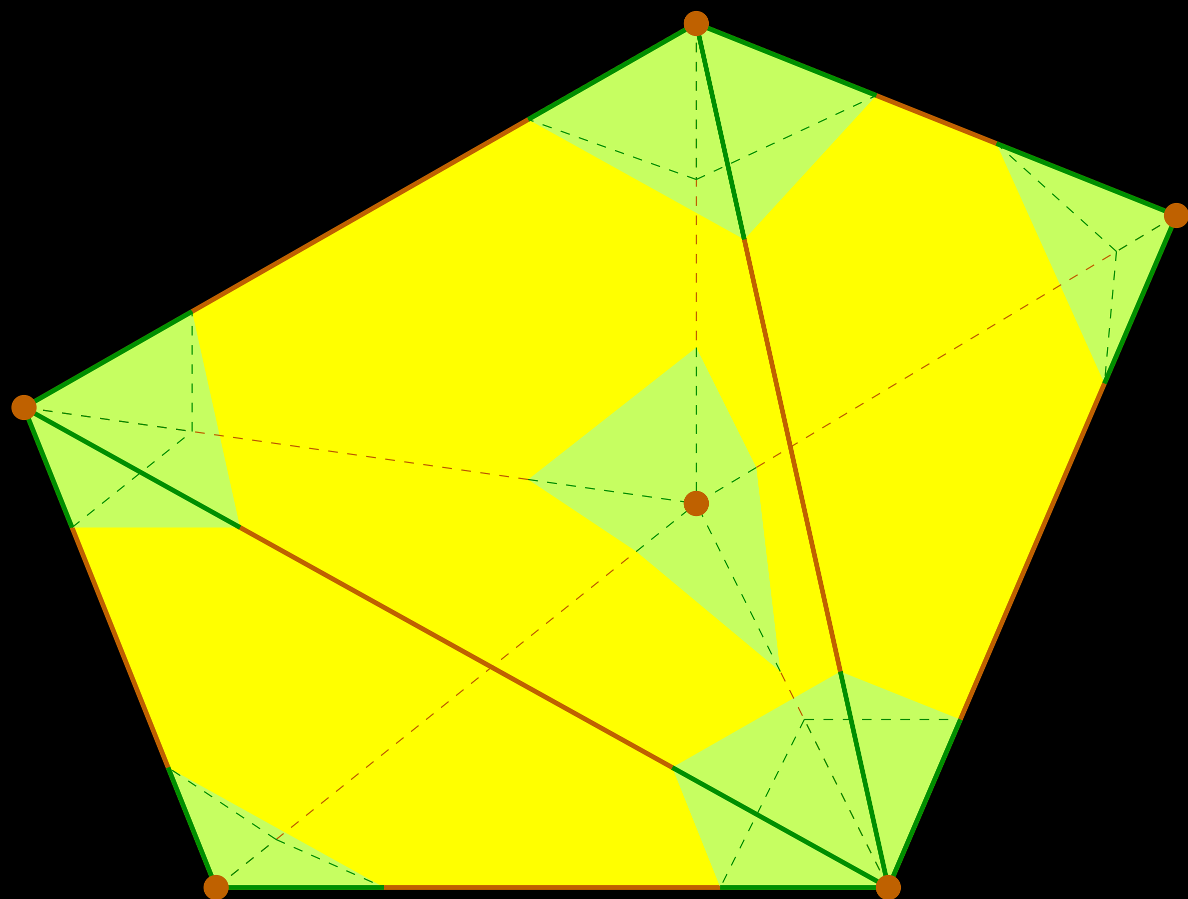
Computing the Convex Hull



Give Each Vertex a Good *Starting Set*



Give Each Vertex a Good *Starting Set*

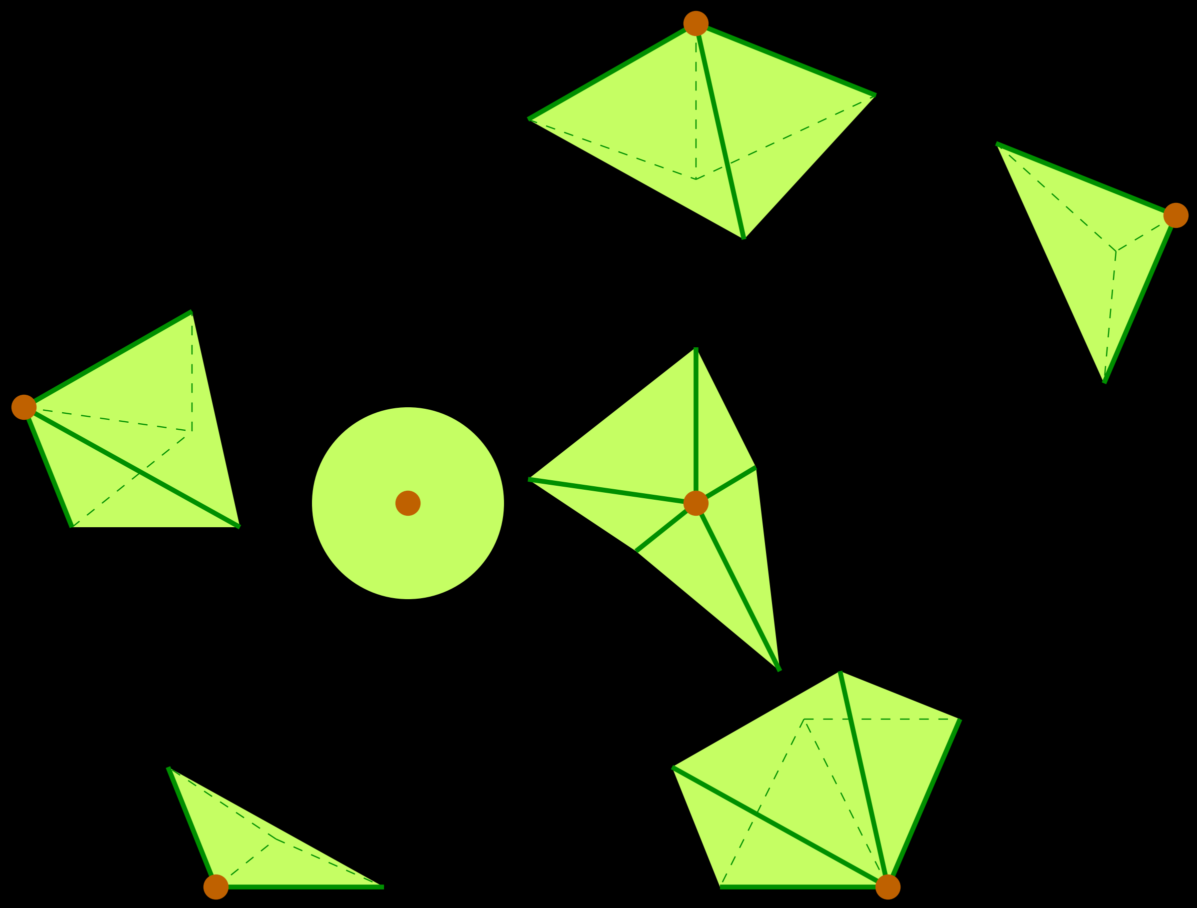


Linear Time!

Star Splaying

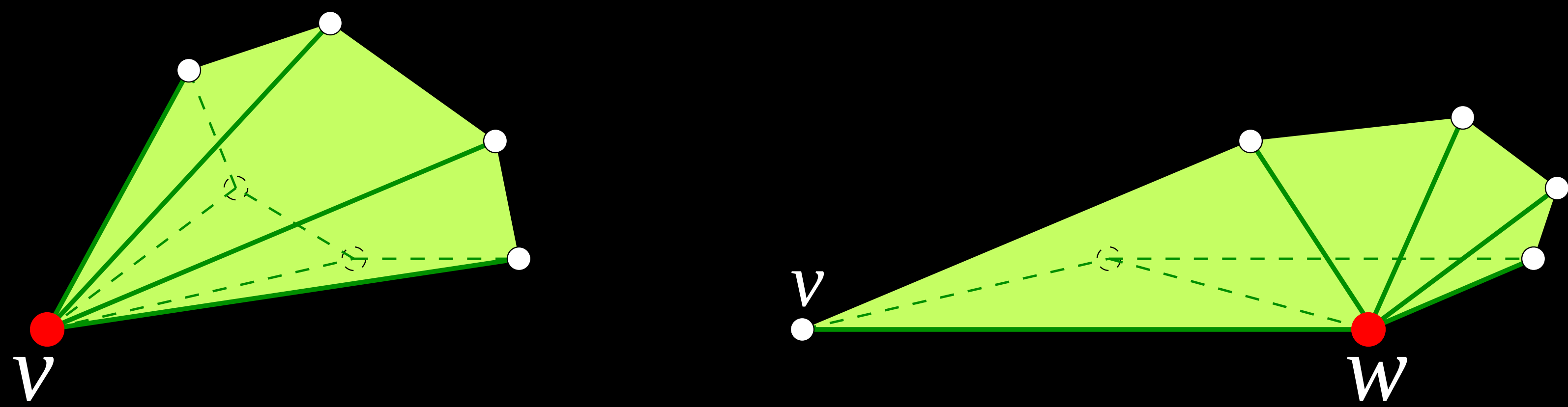
Idea #1

The representation is a collection of stars.



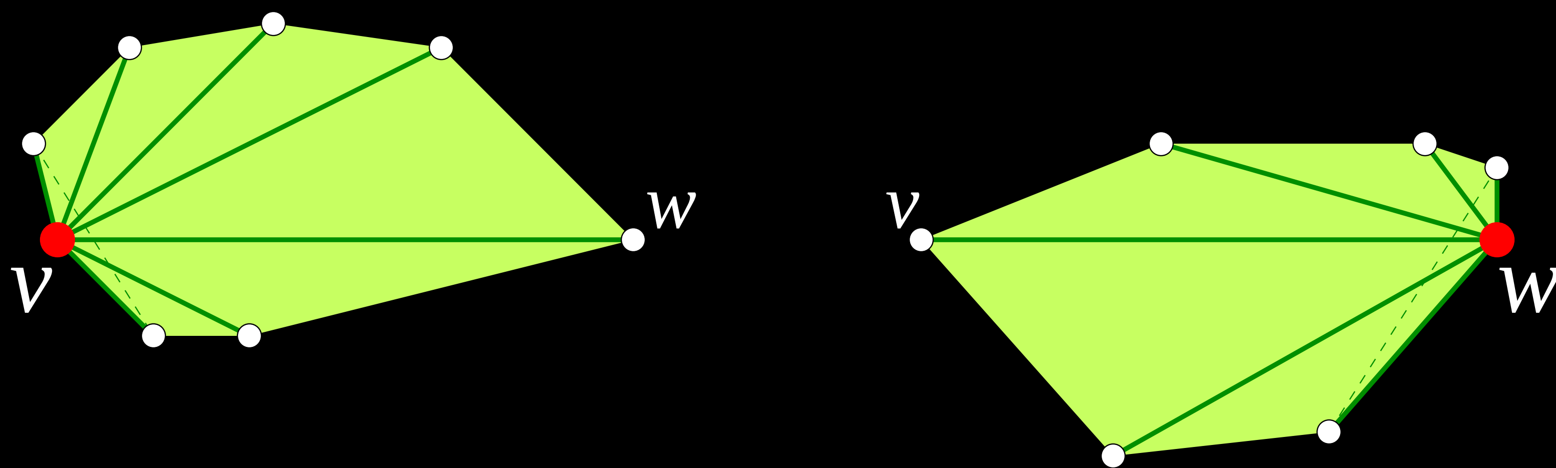
Idea #2

Stars may be *inconsistent* with each other.



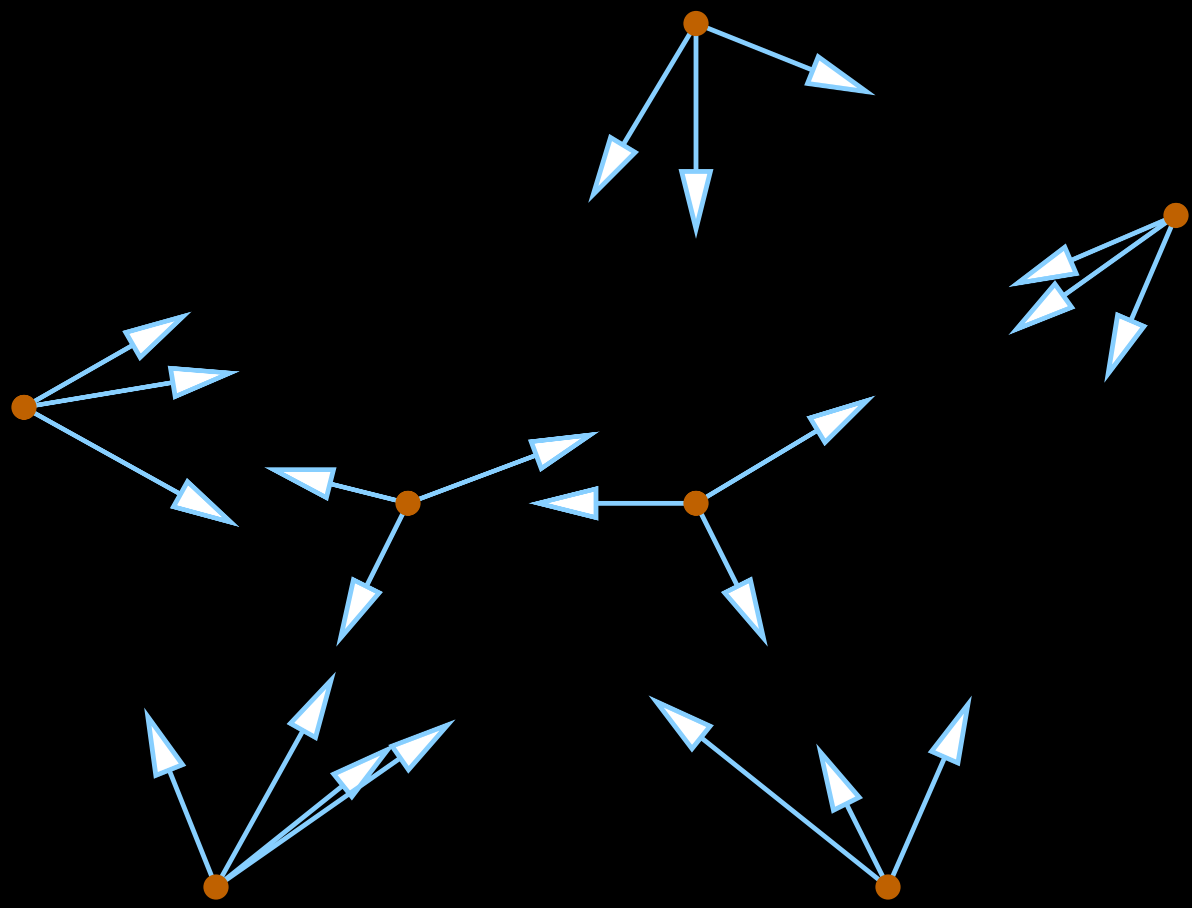
Idea #2

Stars may be *inconsistent* with each other.

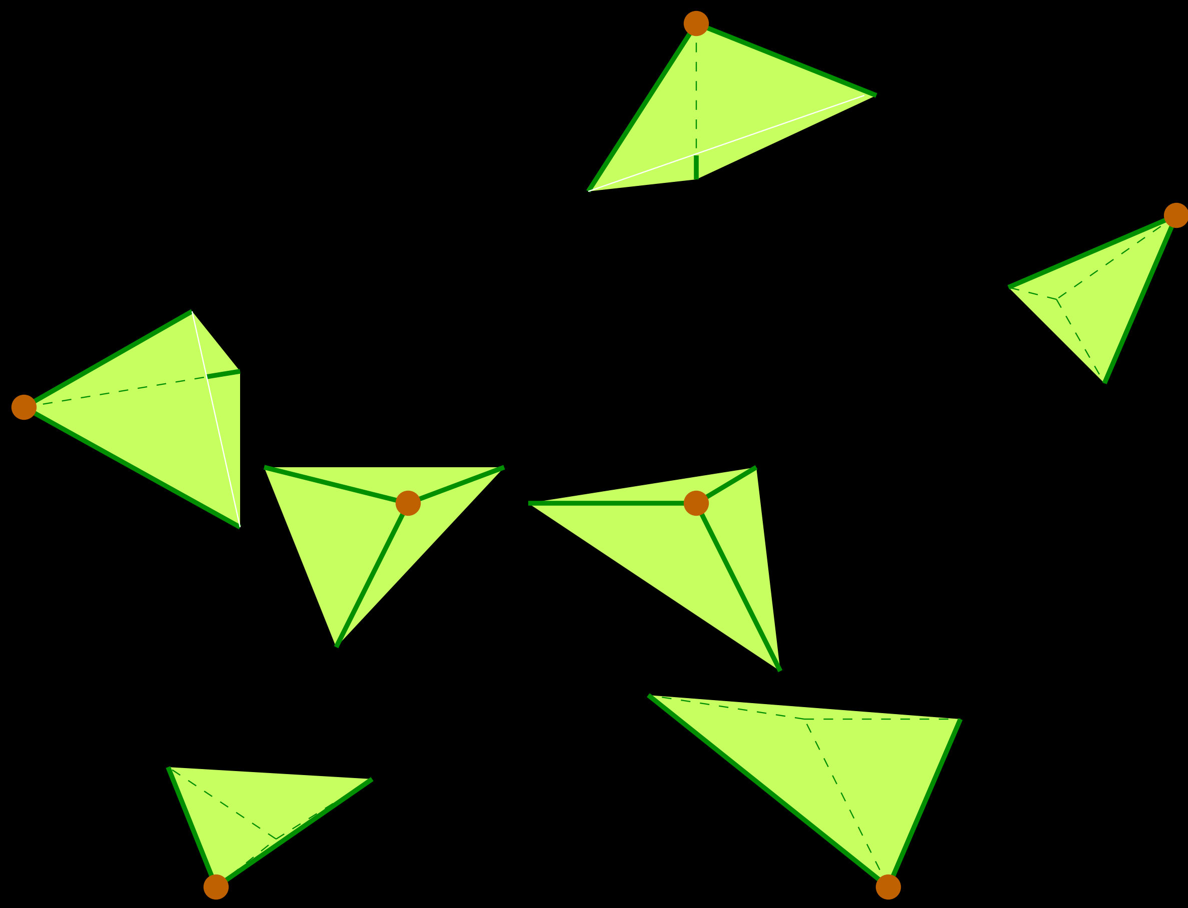


(A simplex appears in the star of one of its vertices, but not in the stars of all its vertices.)

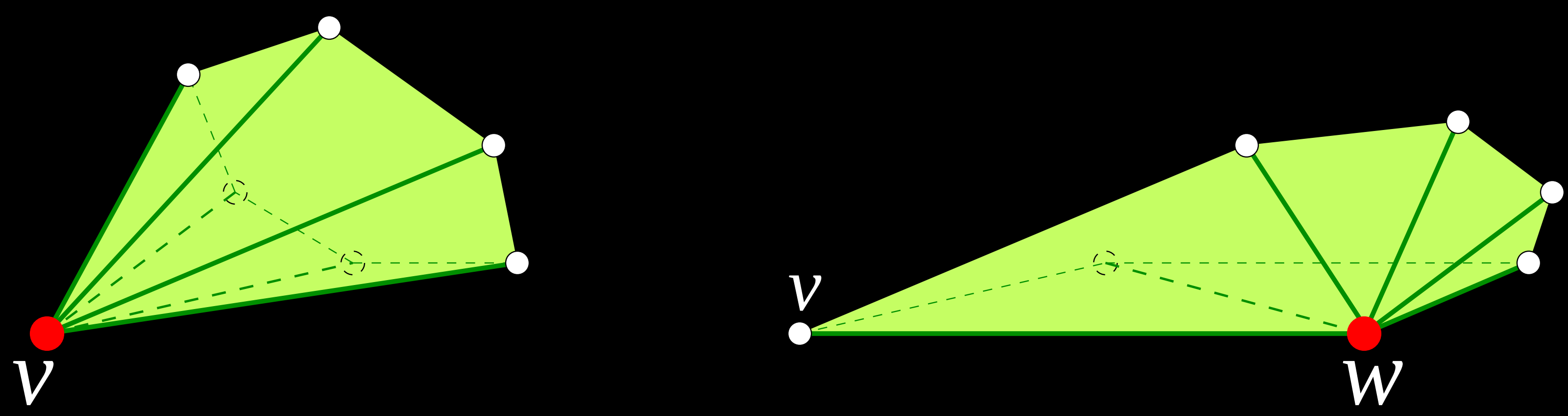
Computing the Convex Hull with Incomplete Starting Sets



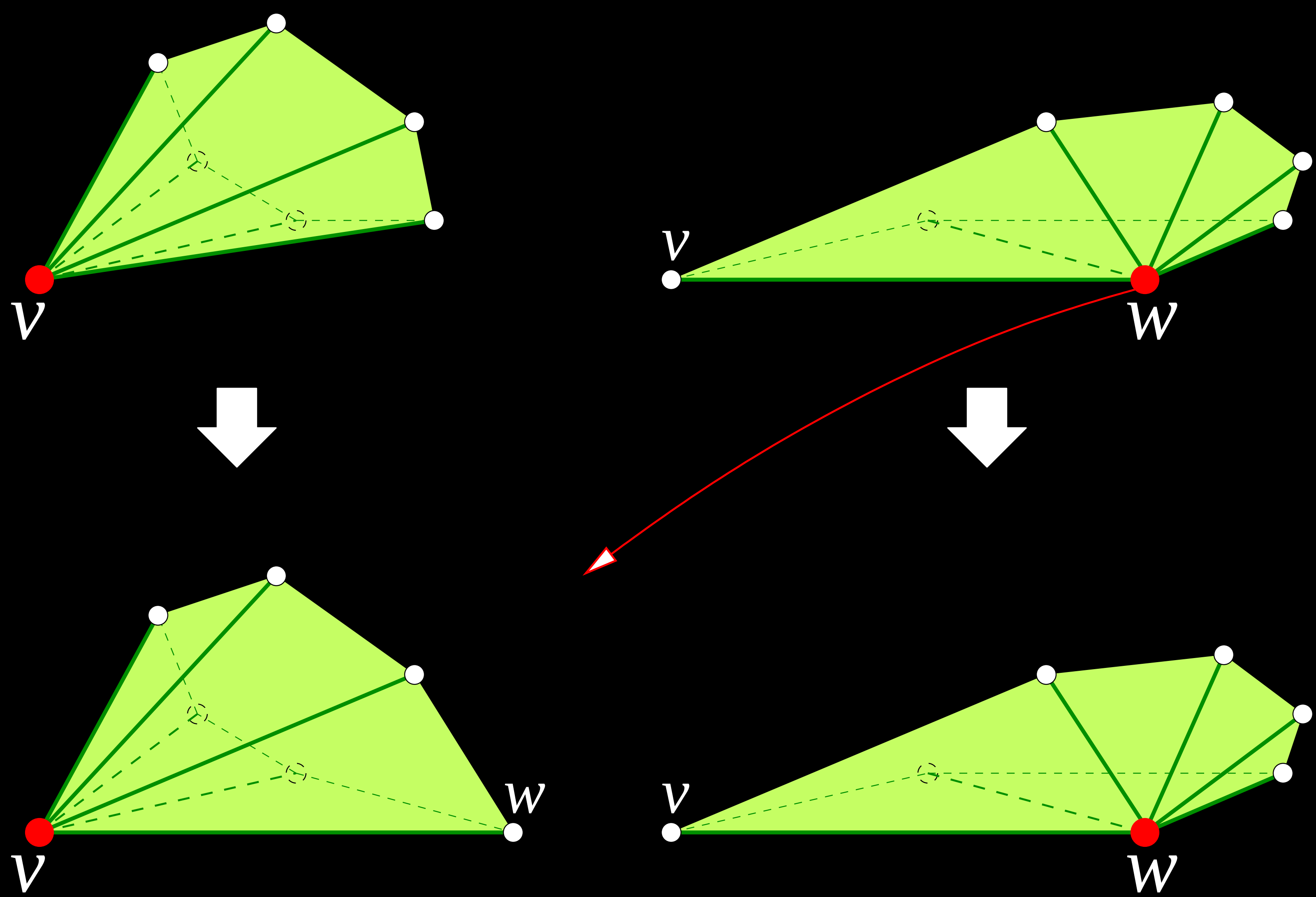
Computing the Convex Hull with Incomplete Starting Sets



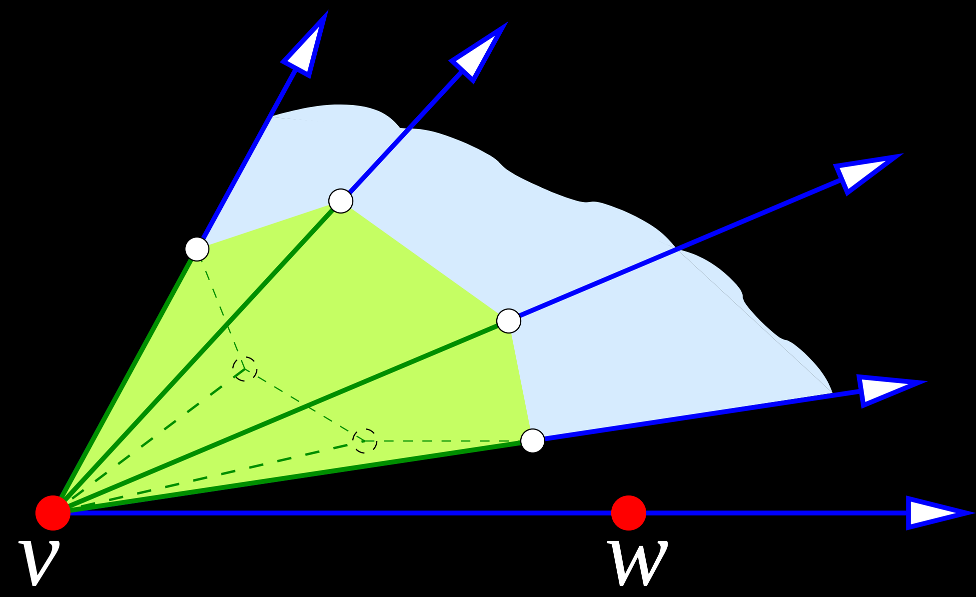
Consistency Resolution



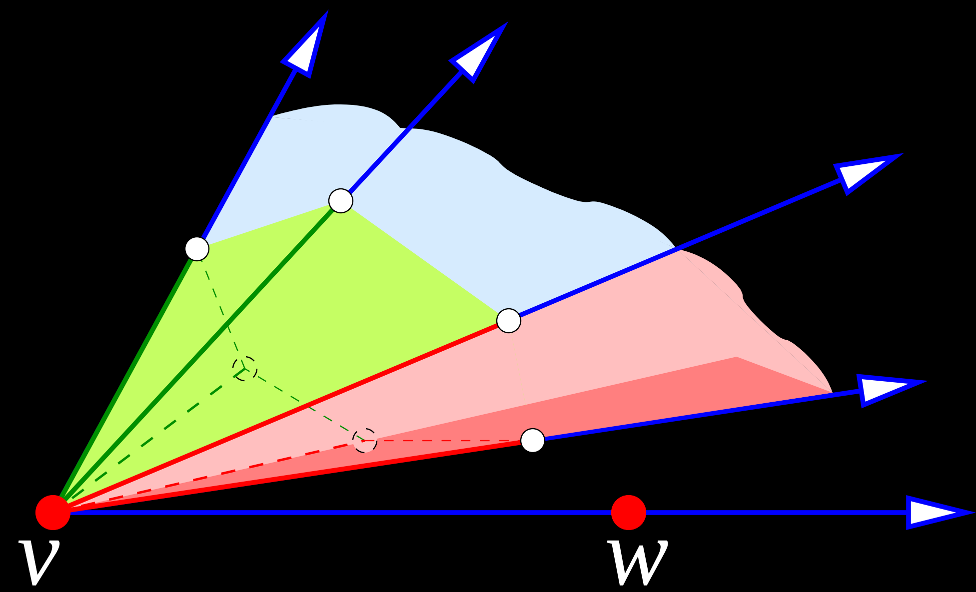
Consistency Resolution



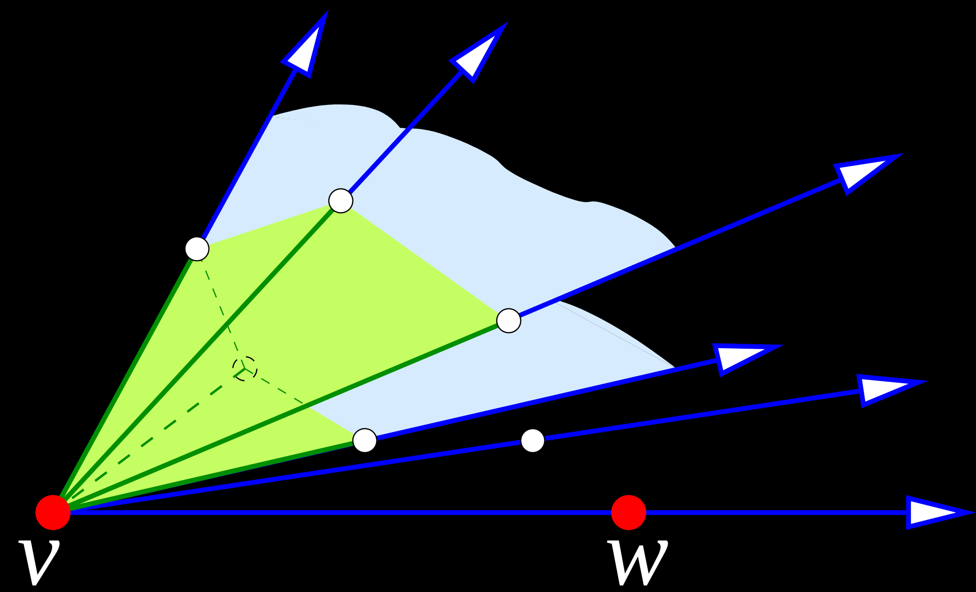
Kallay's Beneath–Beyond Algorithm for Incremental Update of Convex Hulls



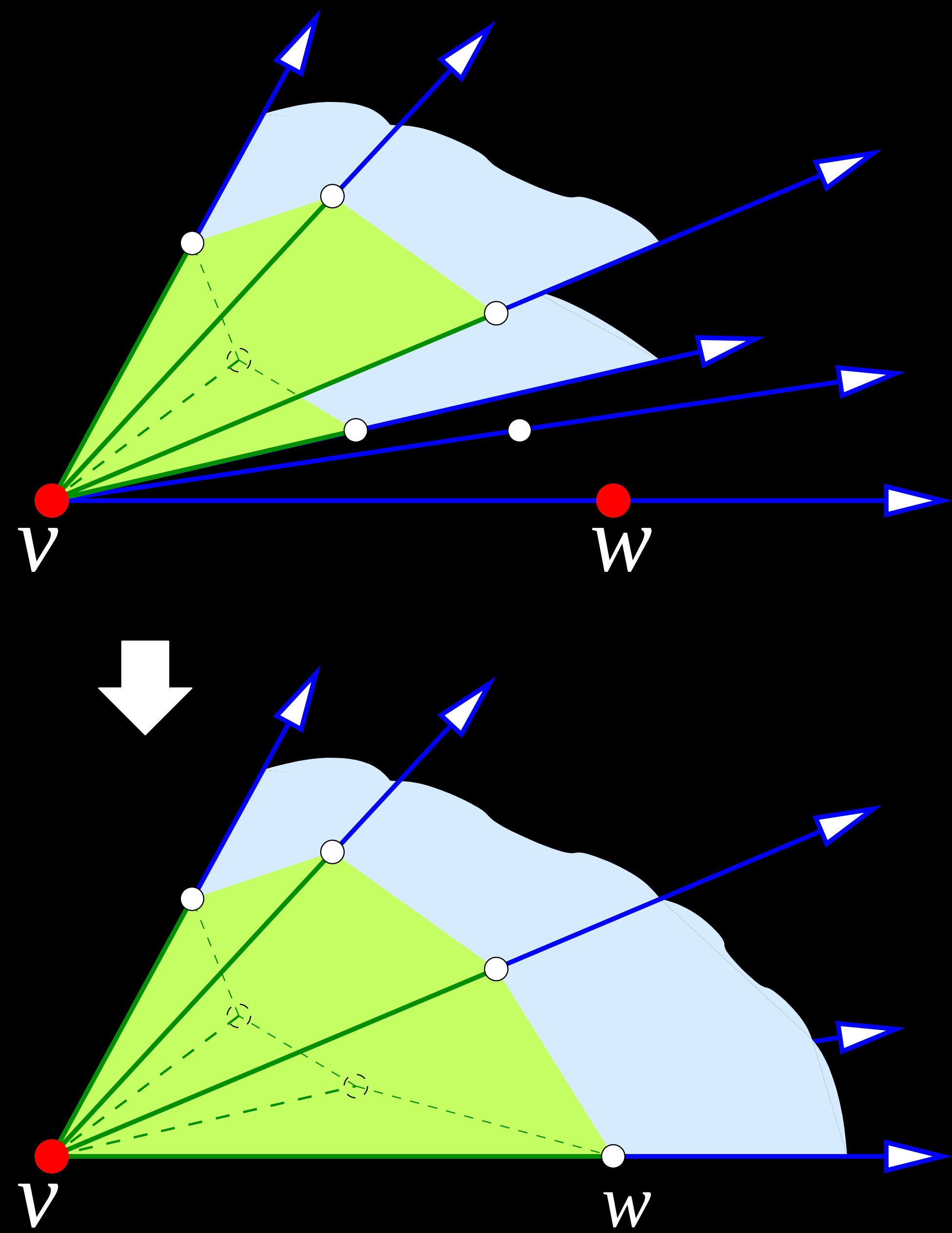
Kallay's Beneath–Beyond Algorithm for Incremental Update of Convex Hulls



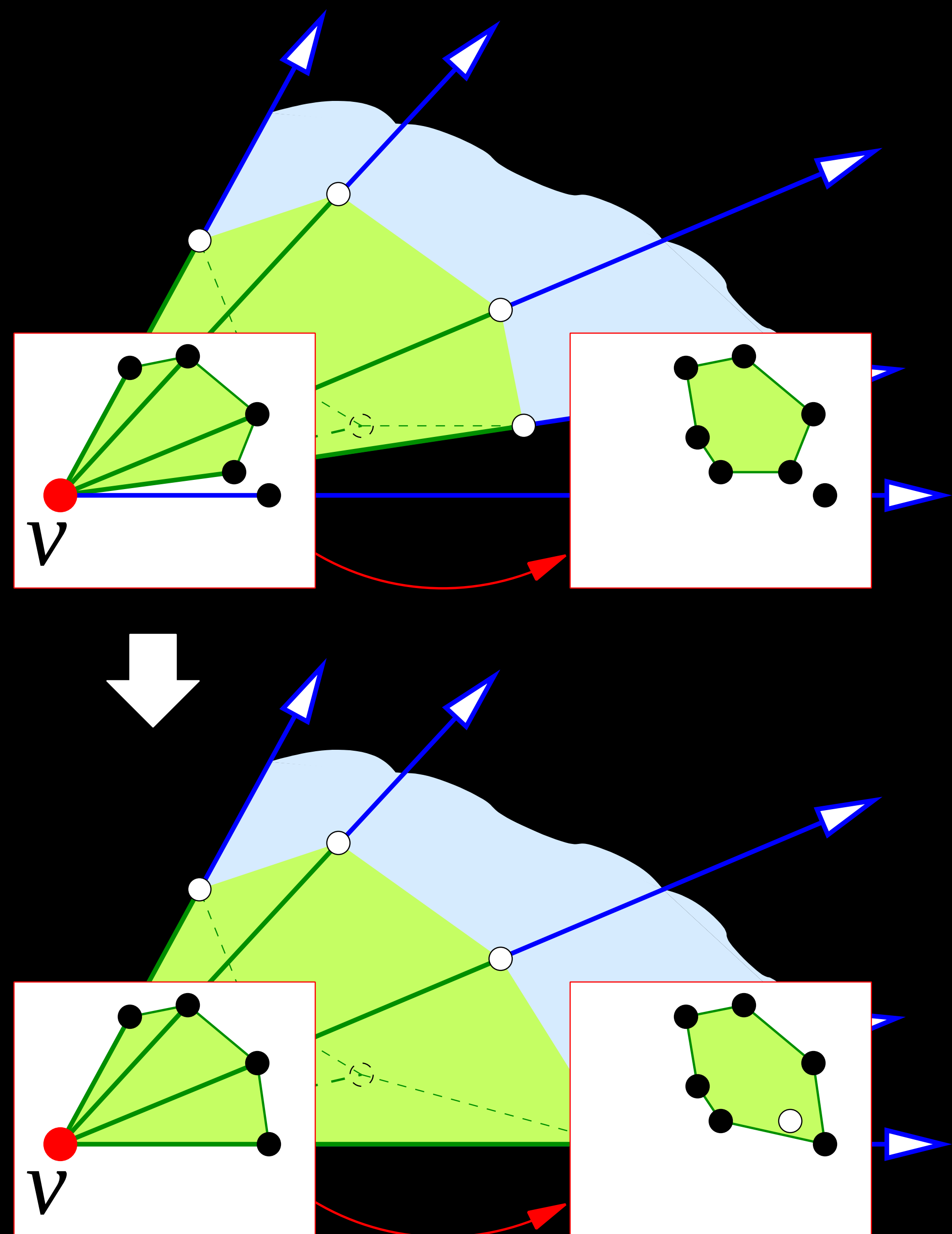
Kallay’s Beneath–Beyond Algorithm for Incremental Update of Convex Hulls



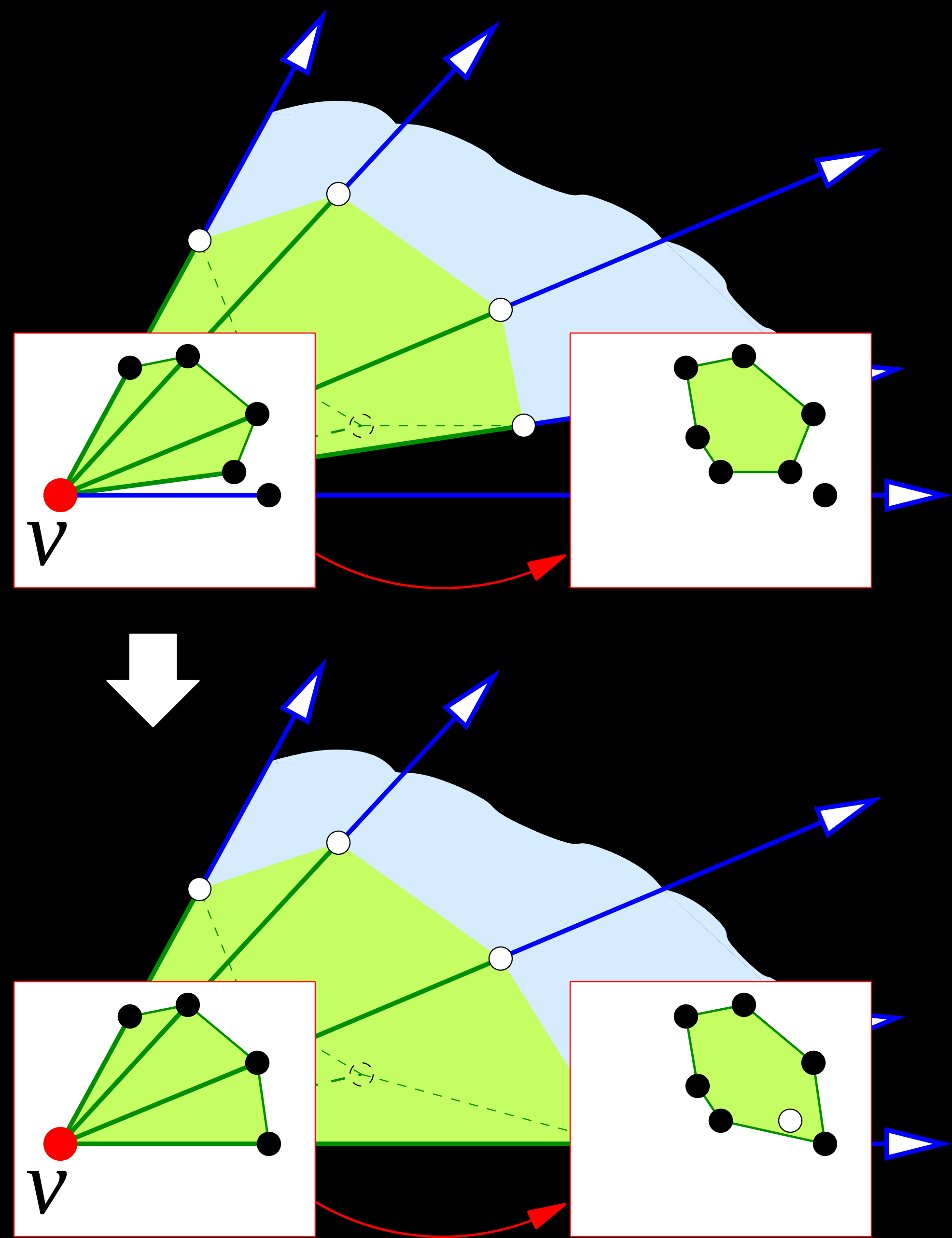
Kallay's Beneath–Beyond Algorithm for Incremental Update of Convex Hulls



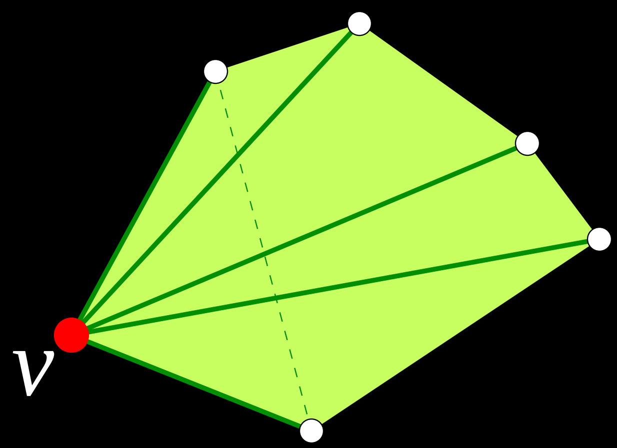
Kallay's Beneath–Beyond Algorithm for Incremental Update of Convex Hulls

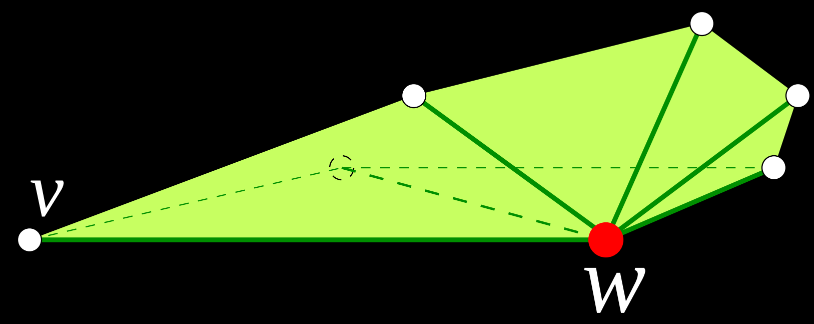


Kallay's Beneath–Beyond Algorithm for Incremental Update of Convex Hulls

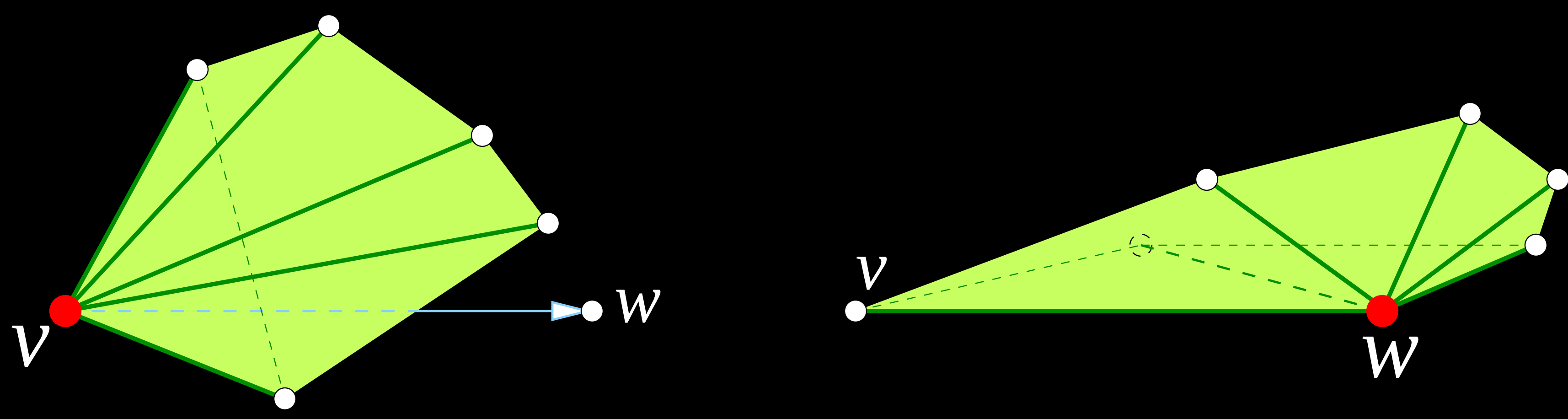


Consistency Resolution

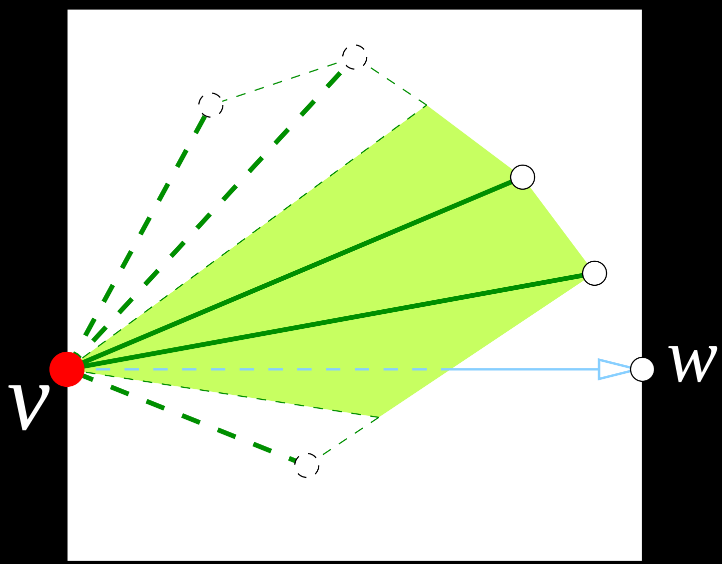


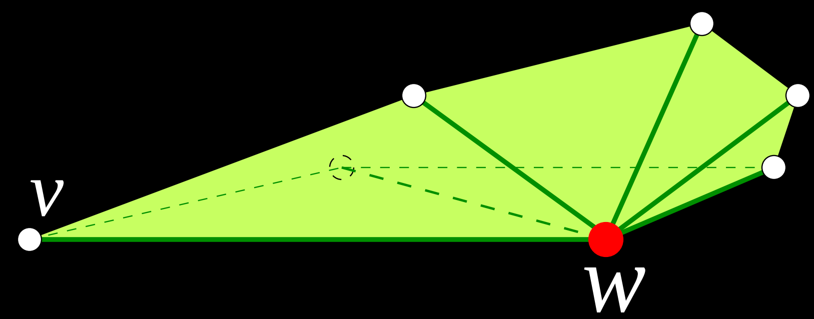


Consistency Resolution

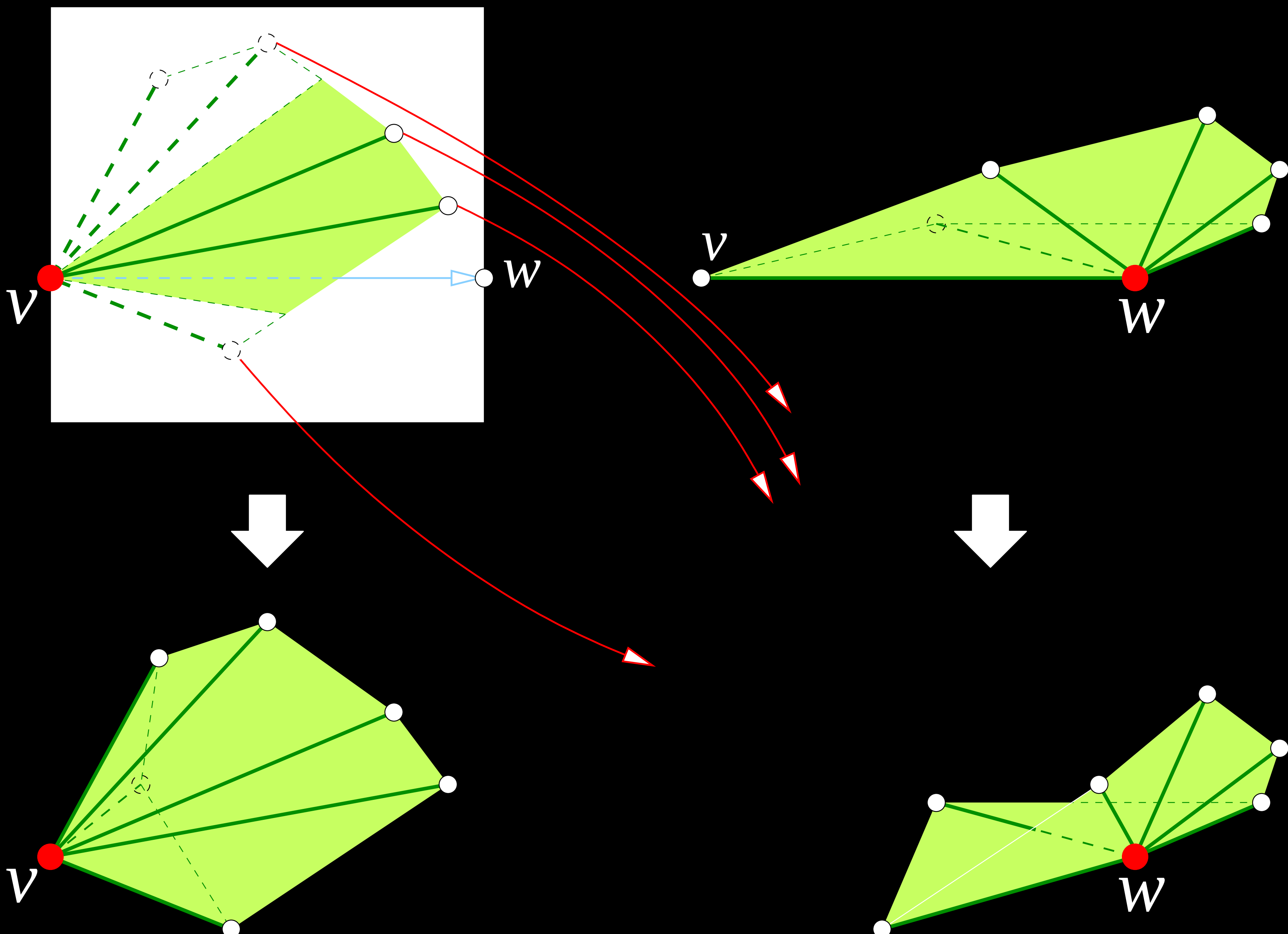


Consistency Resolution

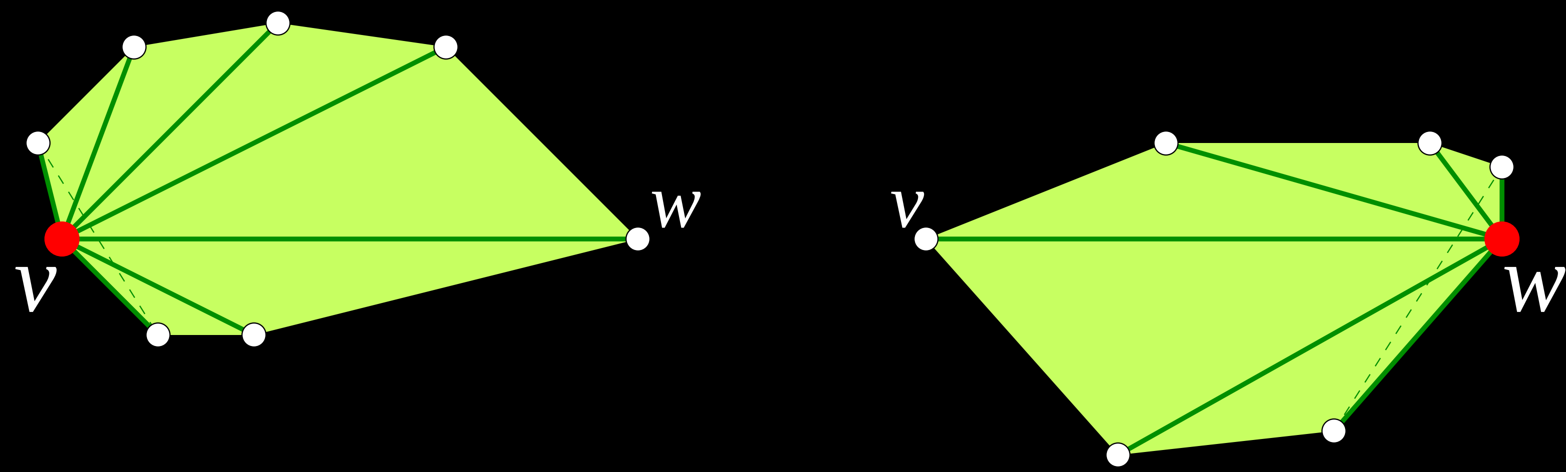




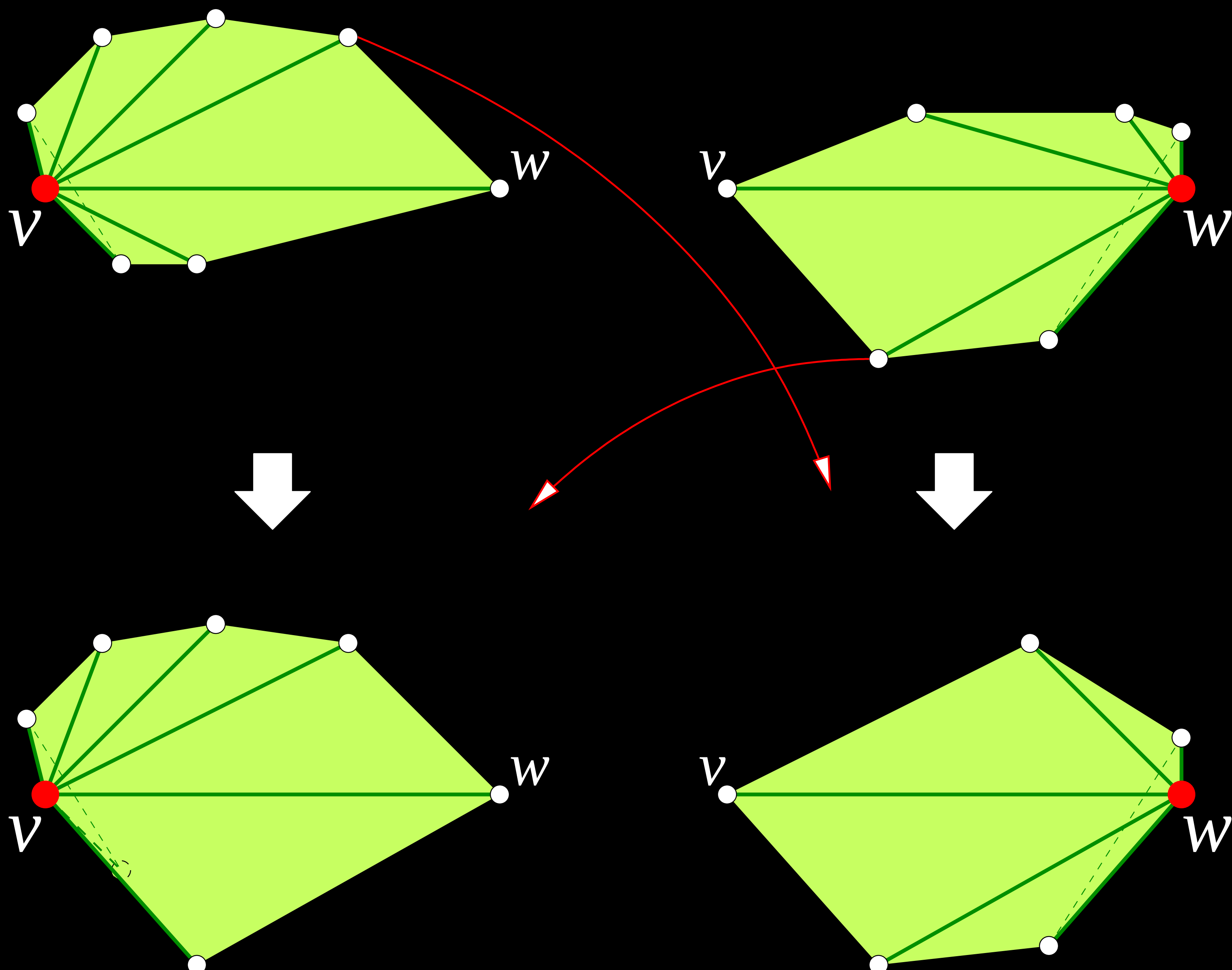
Consistency Resolution



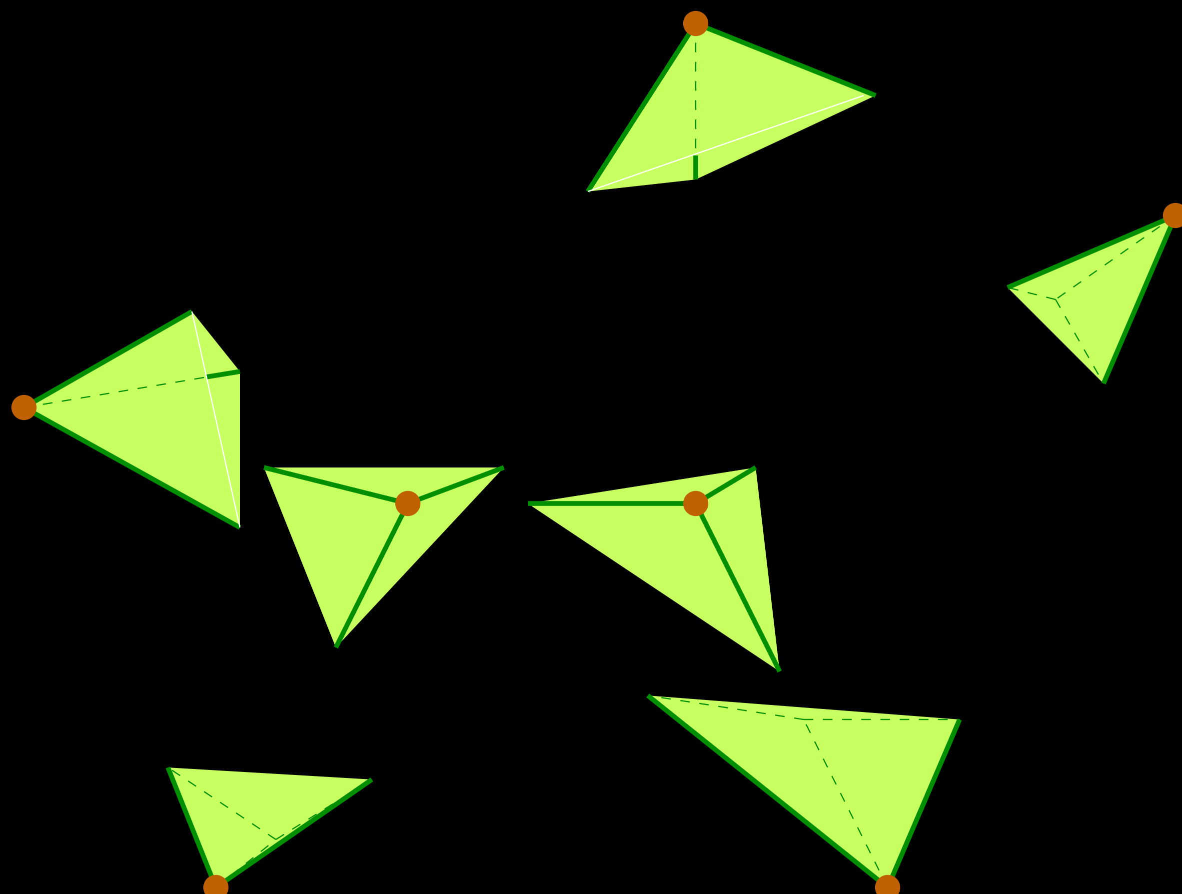
Consistency Resolution



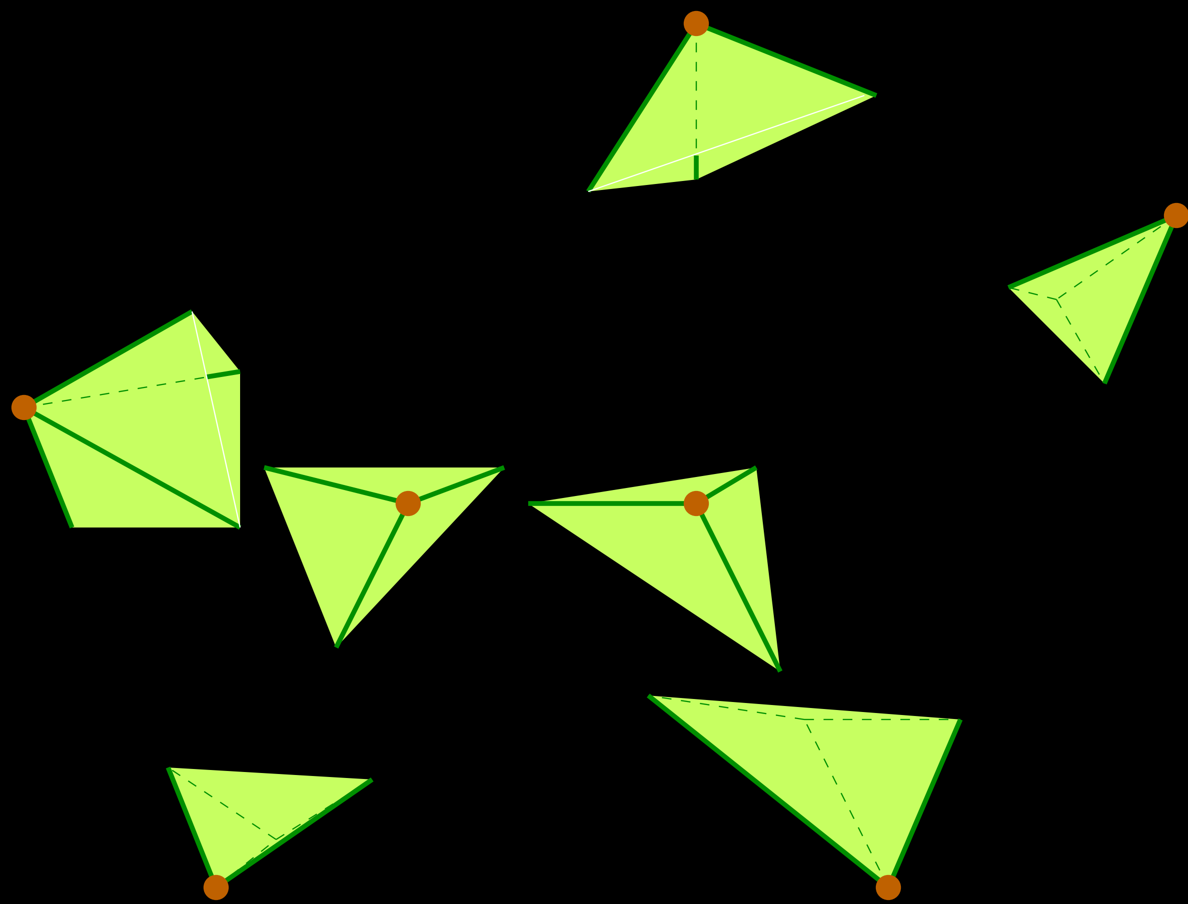
Consistency Resolution



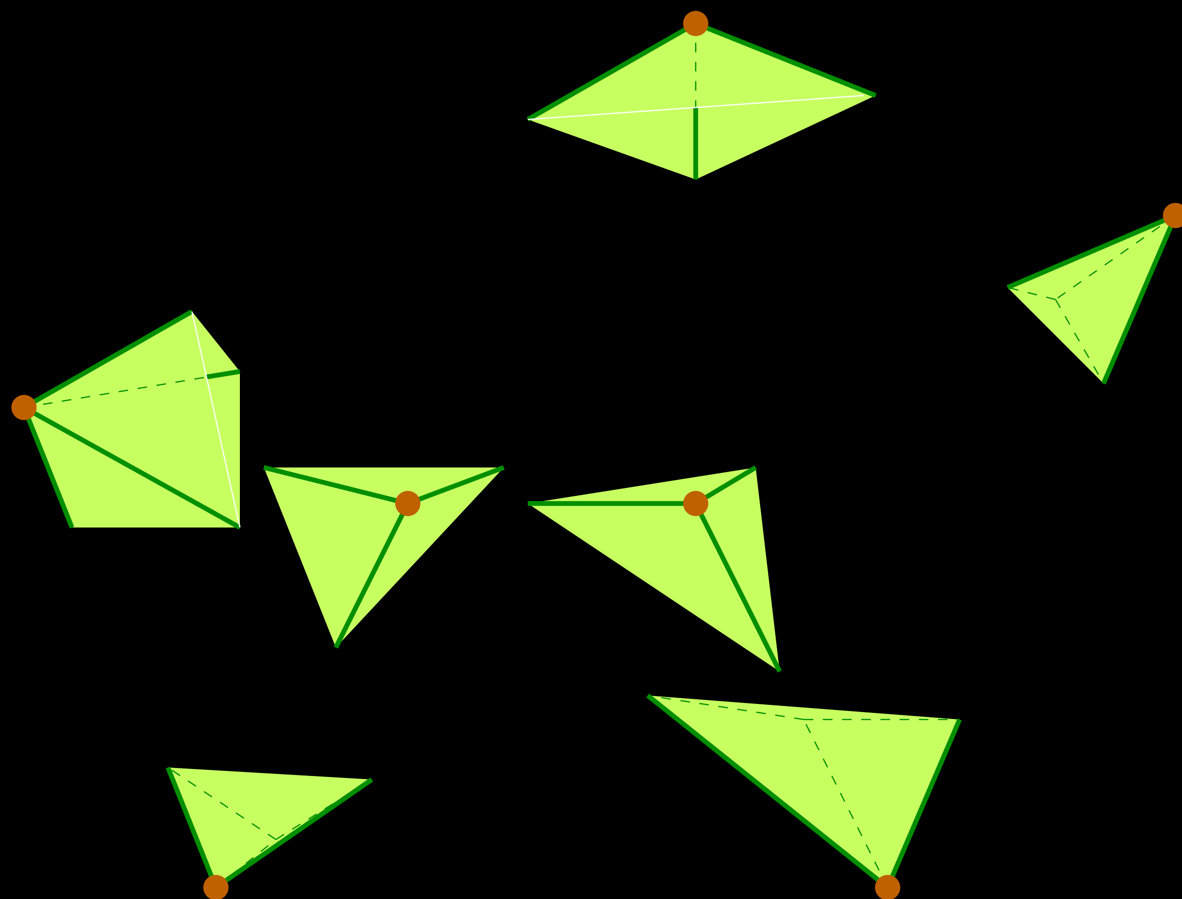
Computing the Convex Hull with Star Splaying



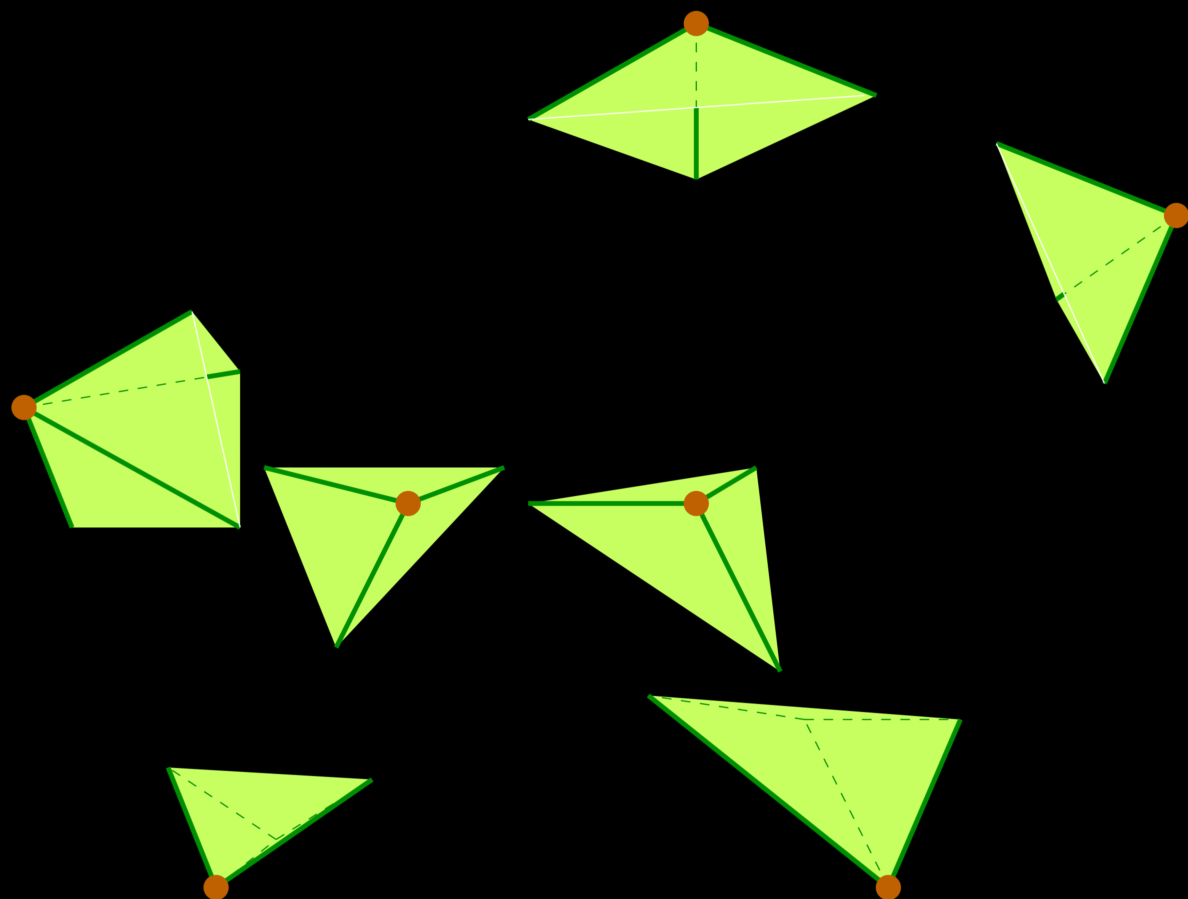
Computing the Convex Hull with Star Splaying



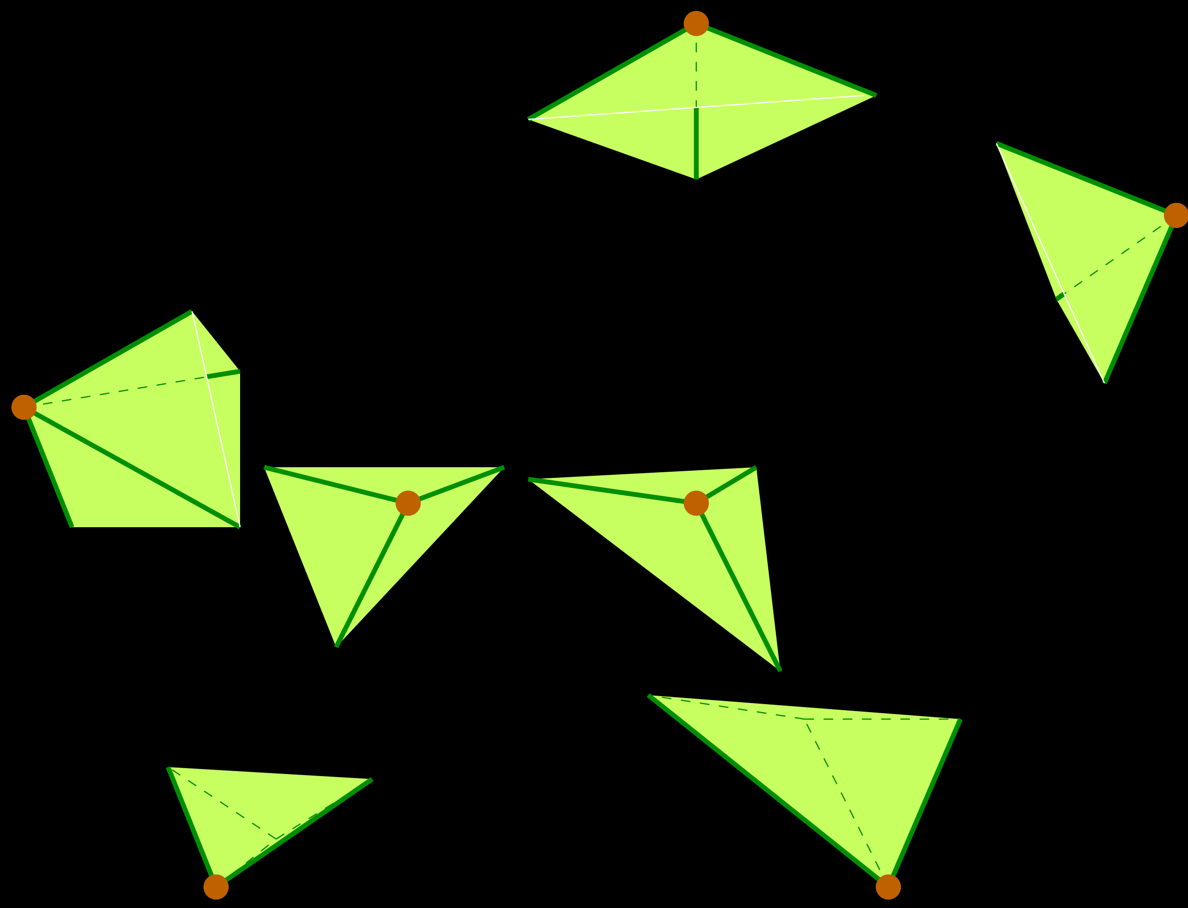
Computing the Convex Hull with Star Splaying



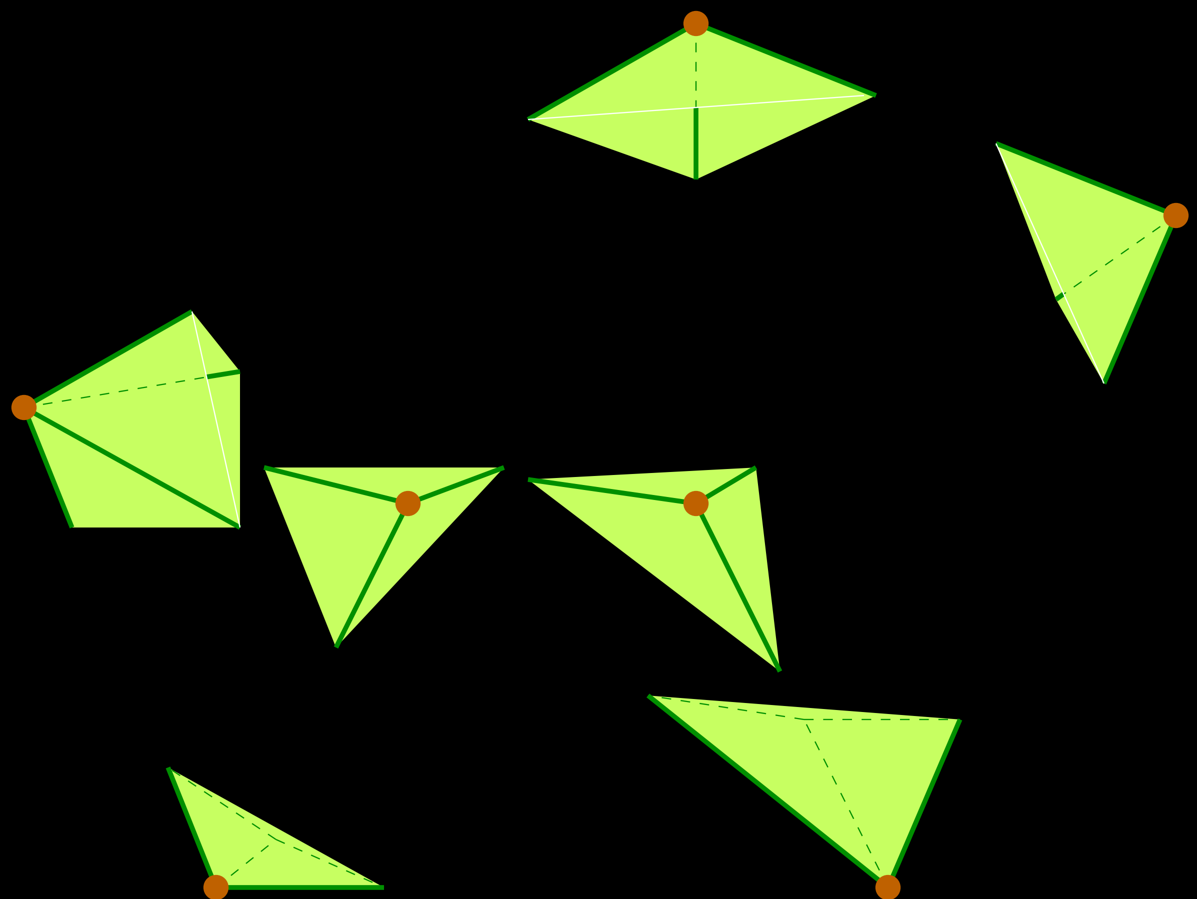
Computing the Convex Hull with Star Splaying



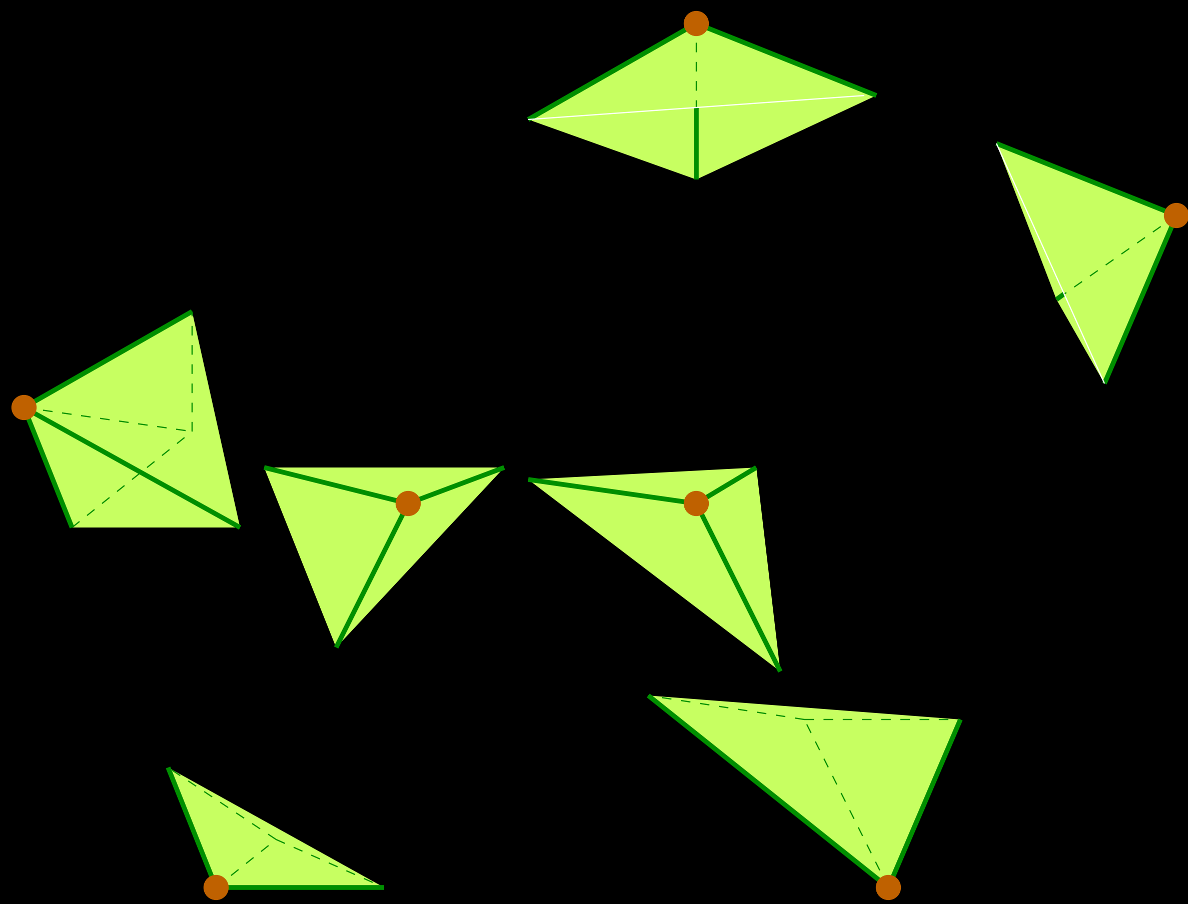
Computing the Convex Hull with Star Splaying



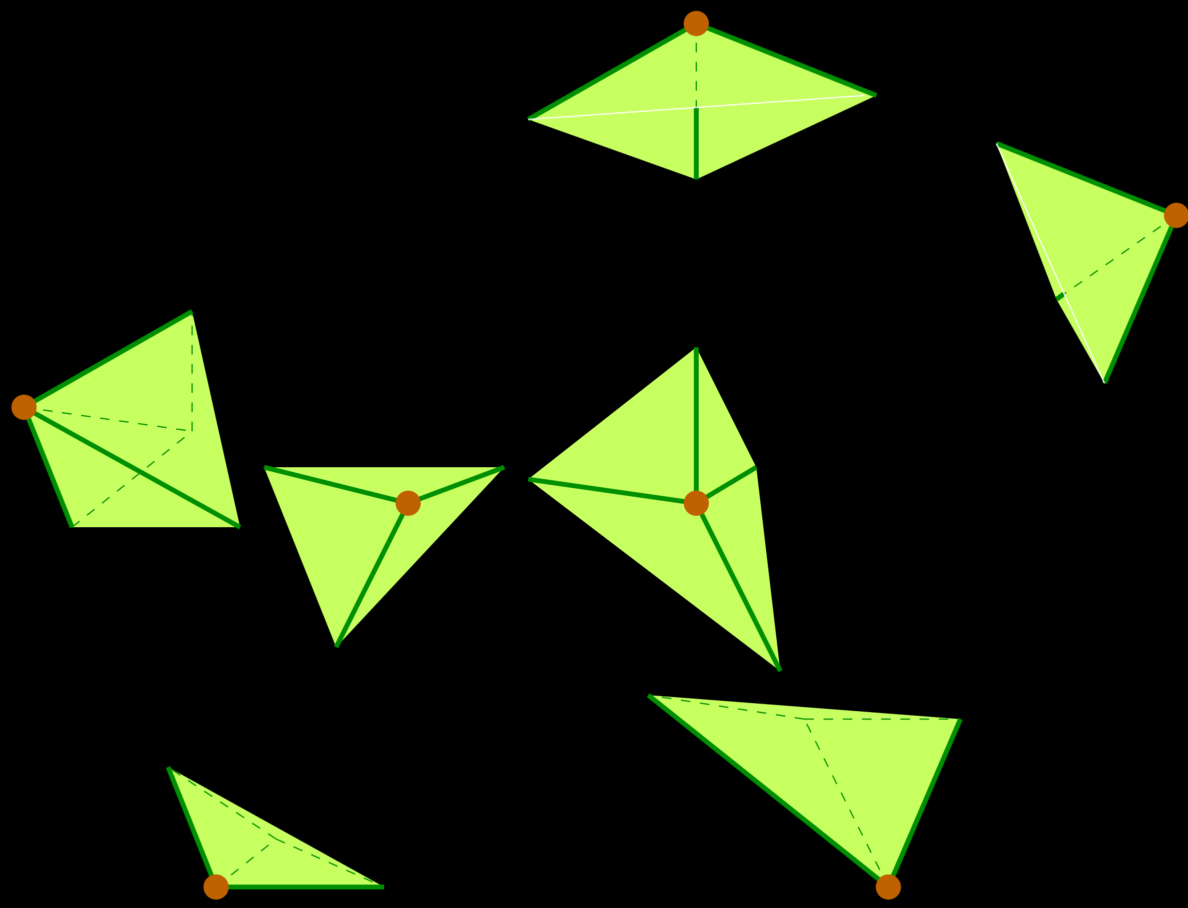
Computing the Convex Hull with Star Splaying



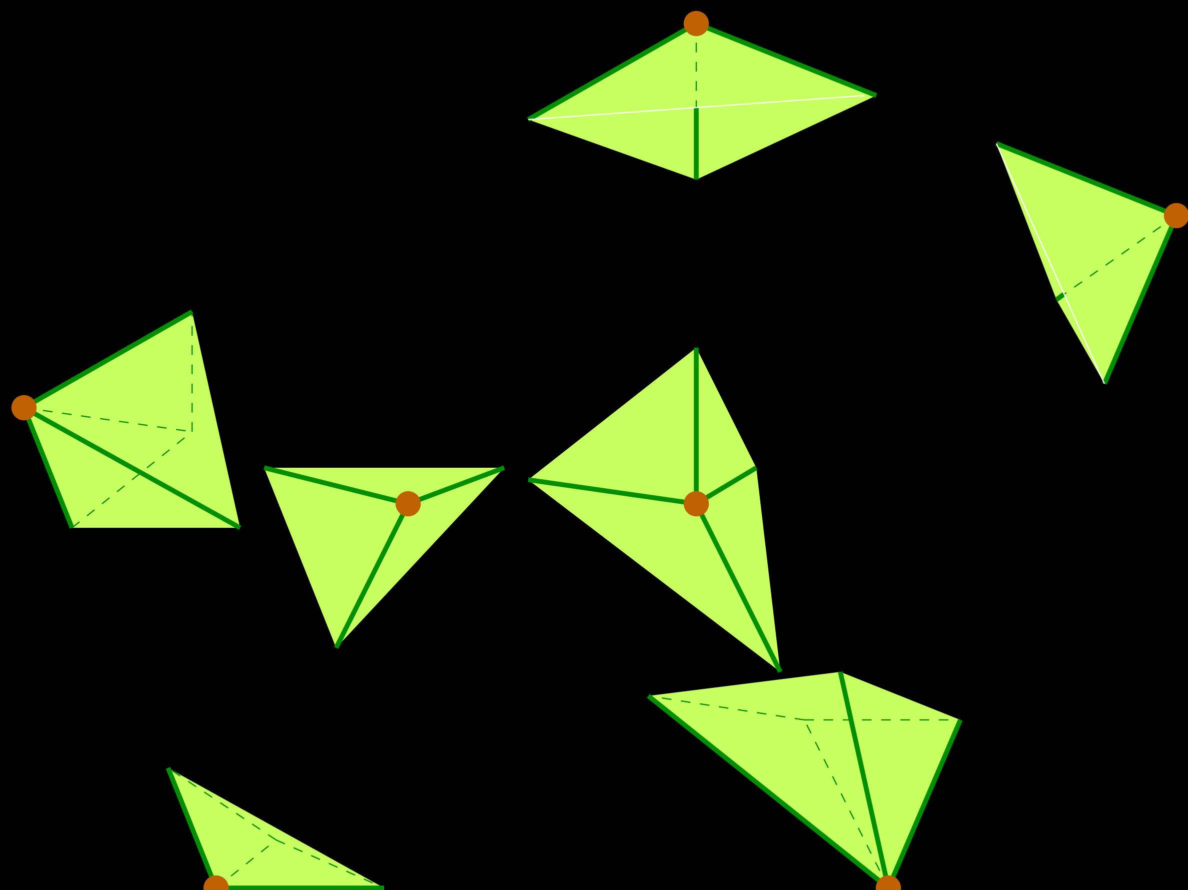
Computing the Convex Hull with Star Splaying



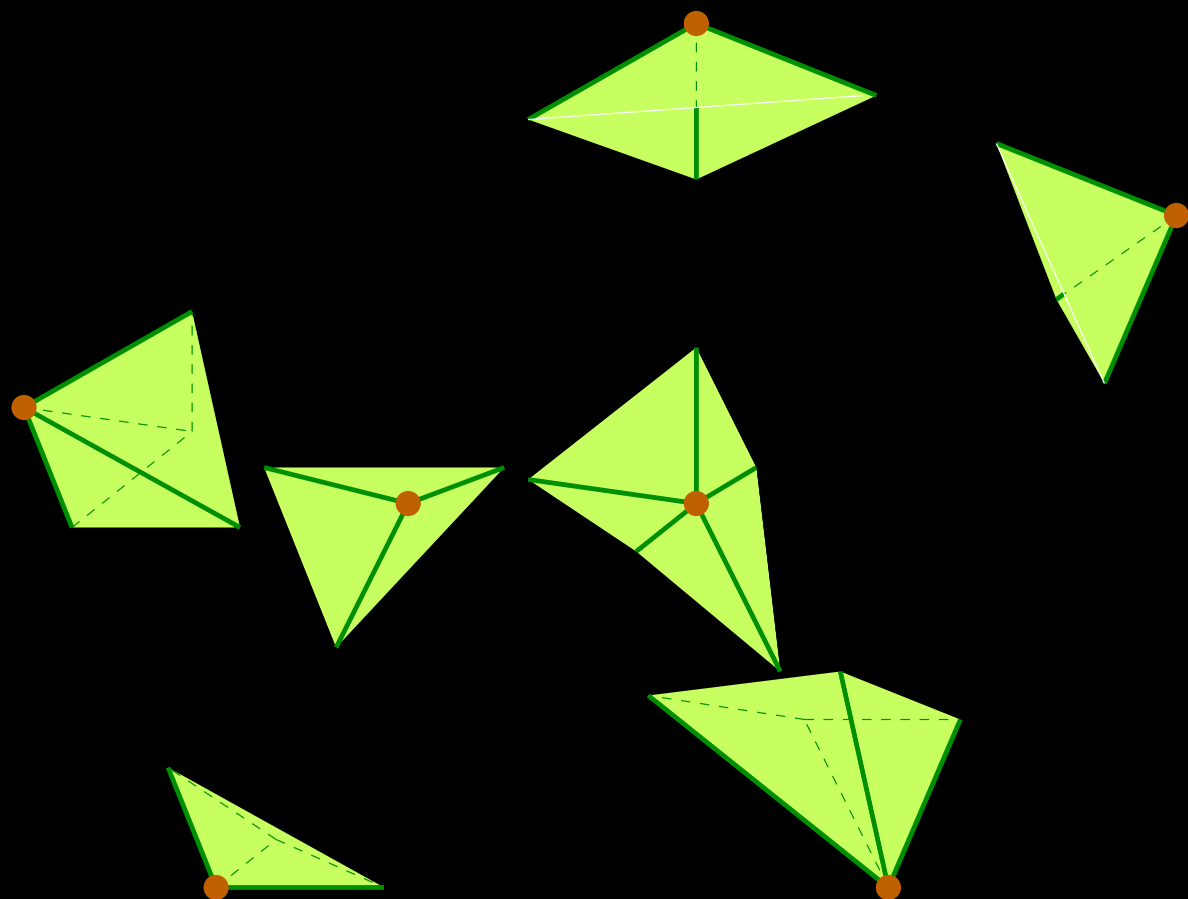
Computing the Convex Hull with Star Splaying



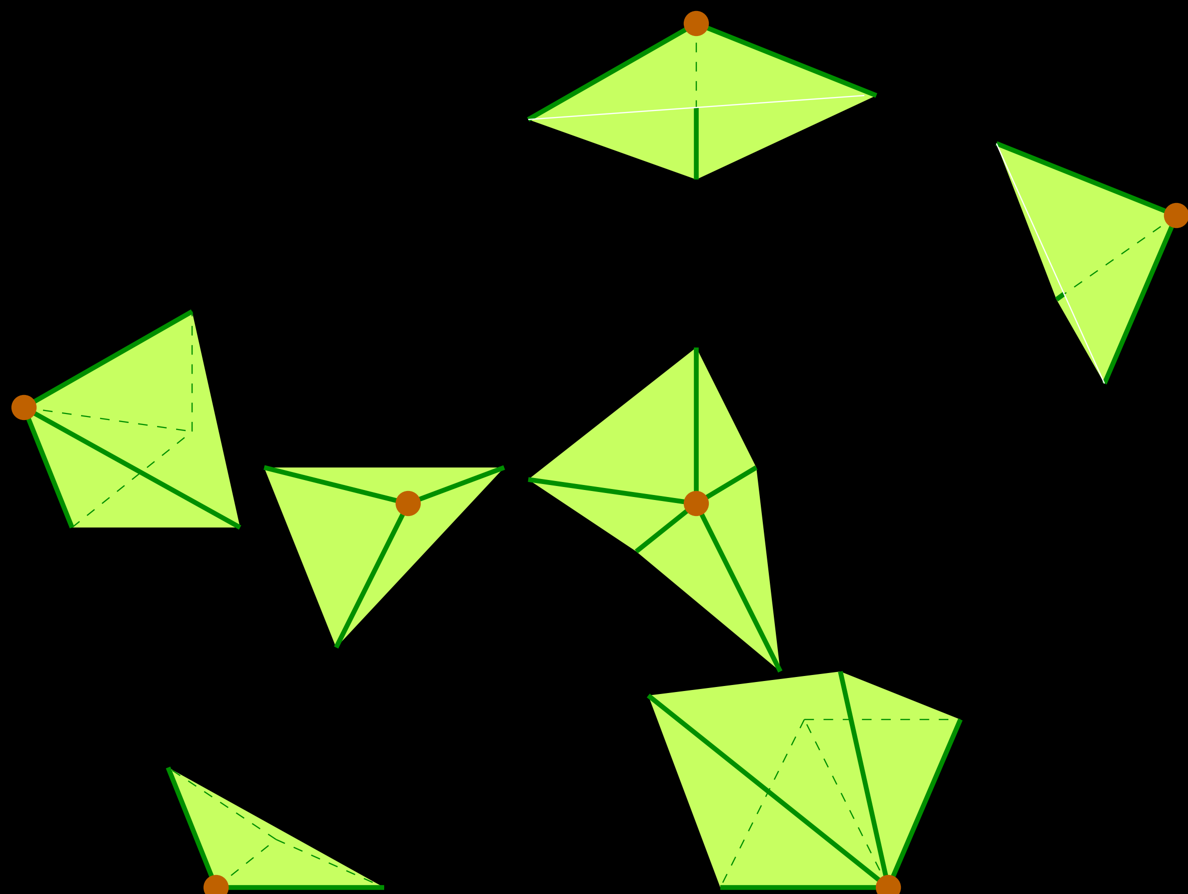
Computing the Convex Hull with Star Splaying



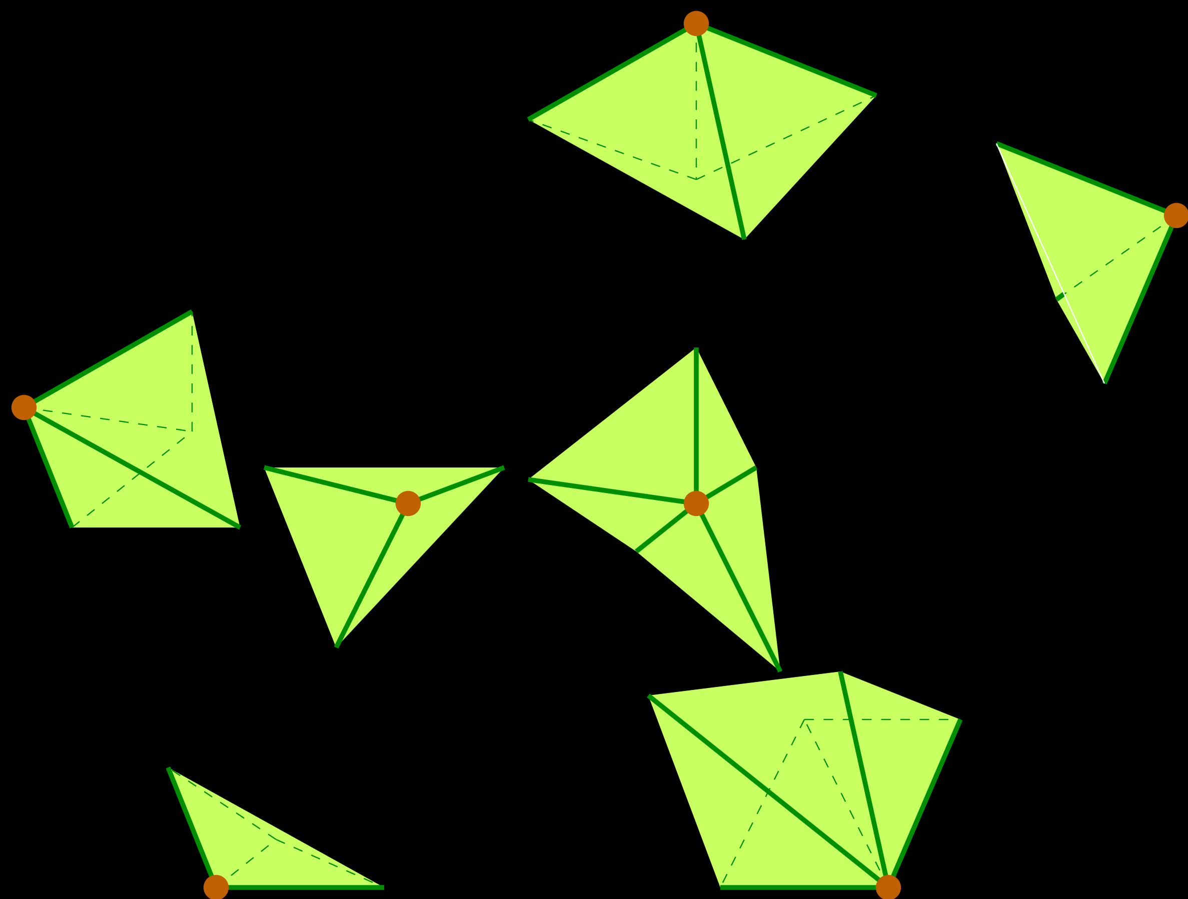
Computing the Convex Hull with Star Splaying



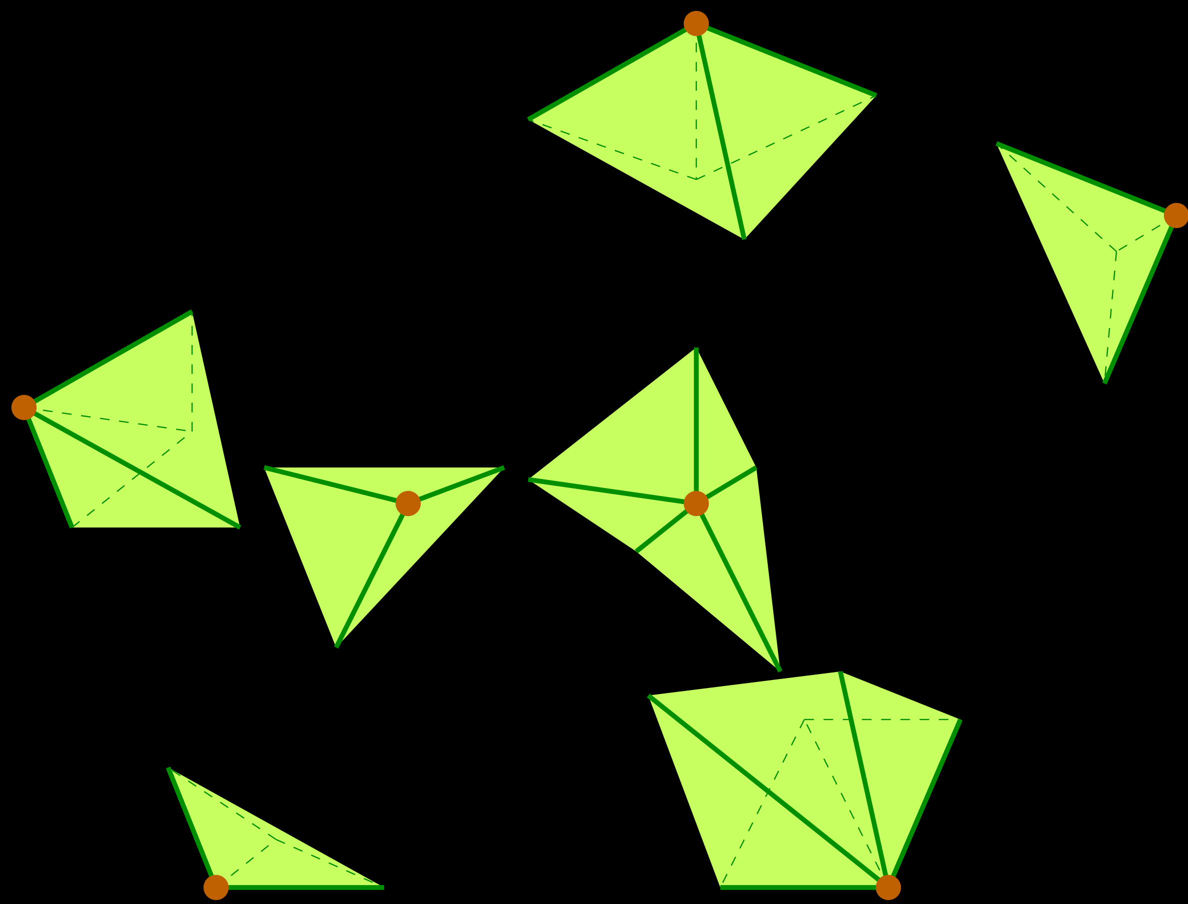
Computing the Convex Hull with Star Splaying



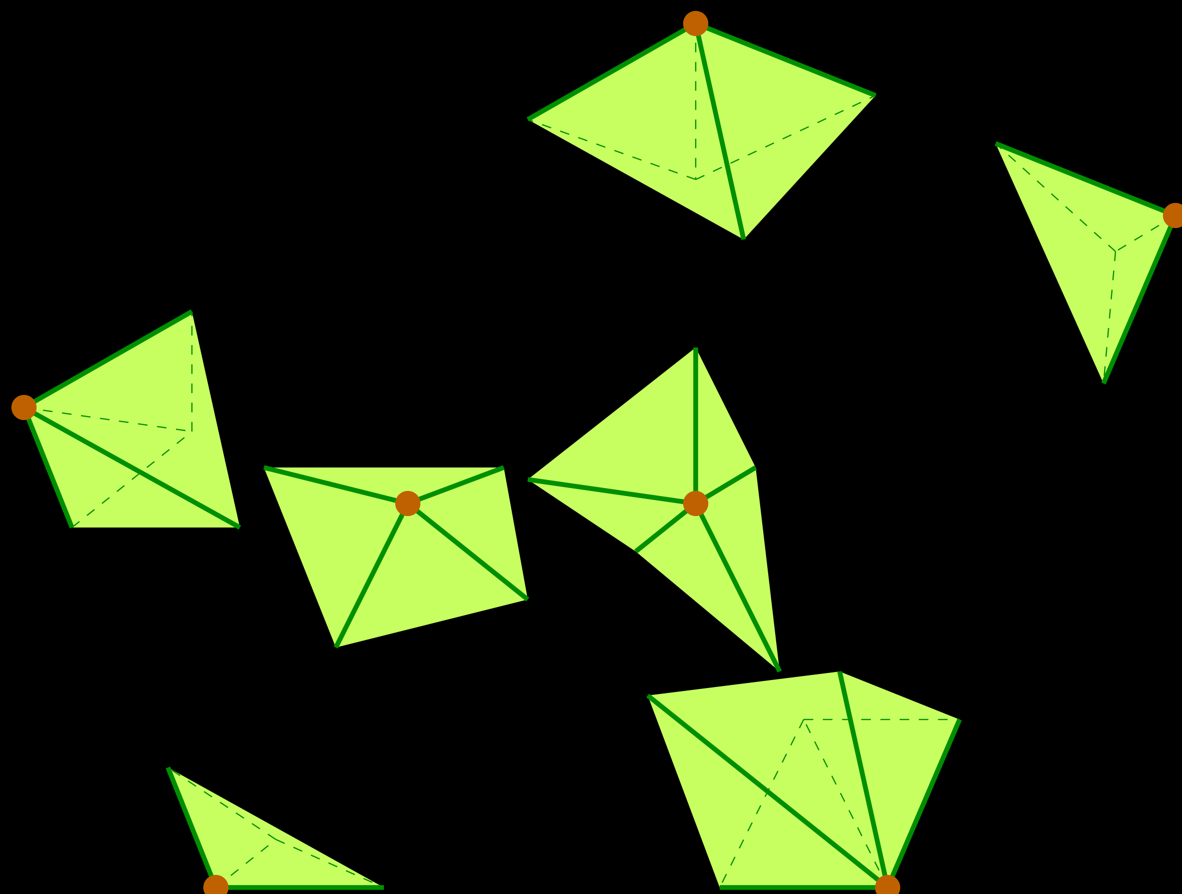
Computing the Convex Hull with Star Splaying



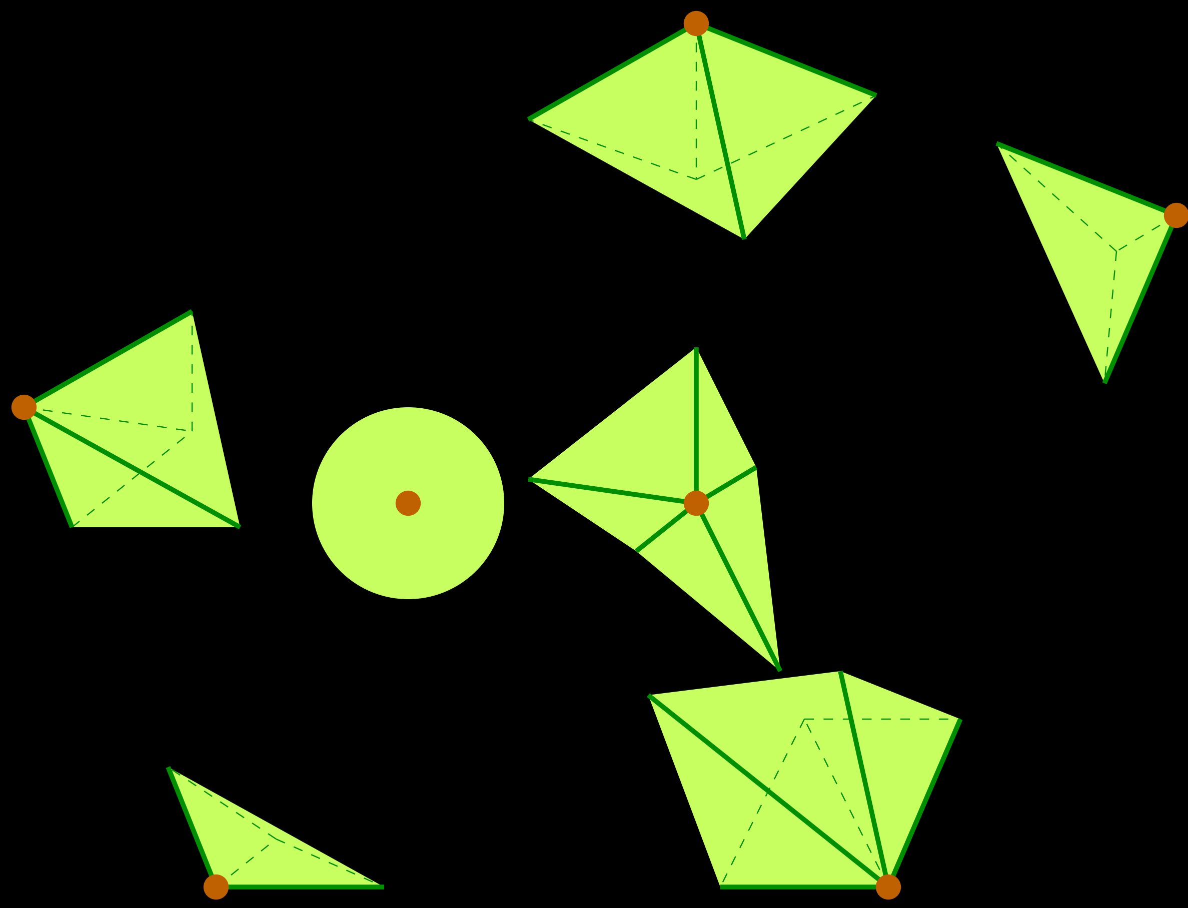
Computing the Convex Hull with Star Splaying



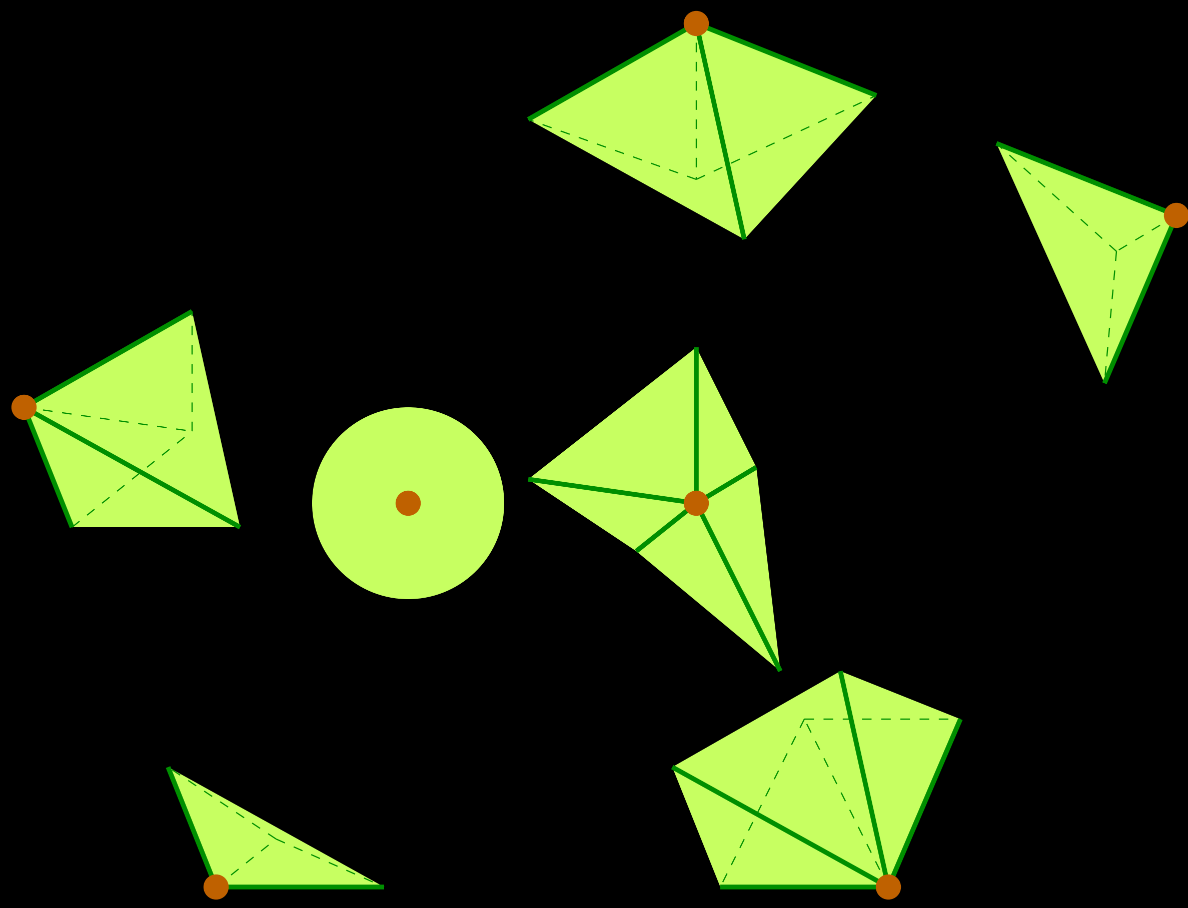
Computing the Convex Hull with Star Splaying



Computing the Convex Hull with Star Splaying

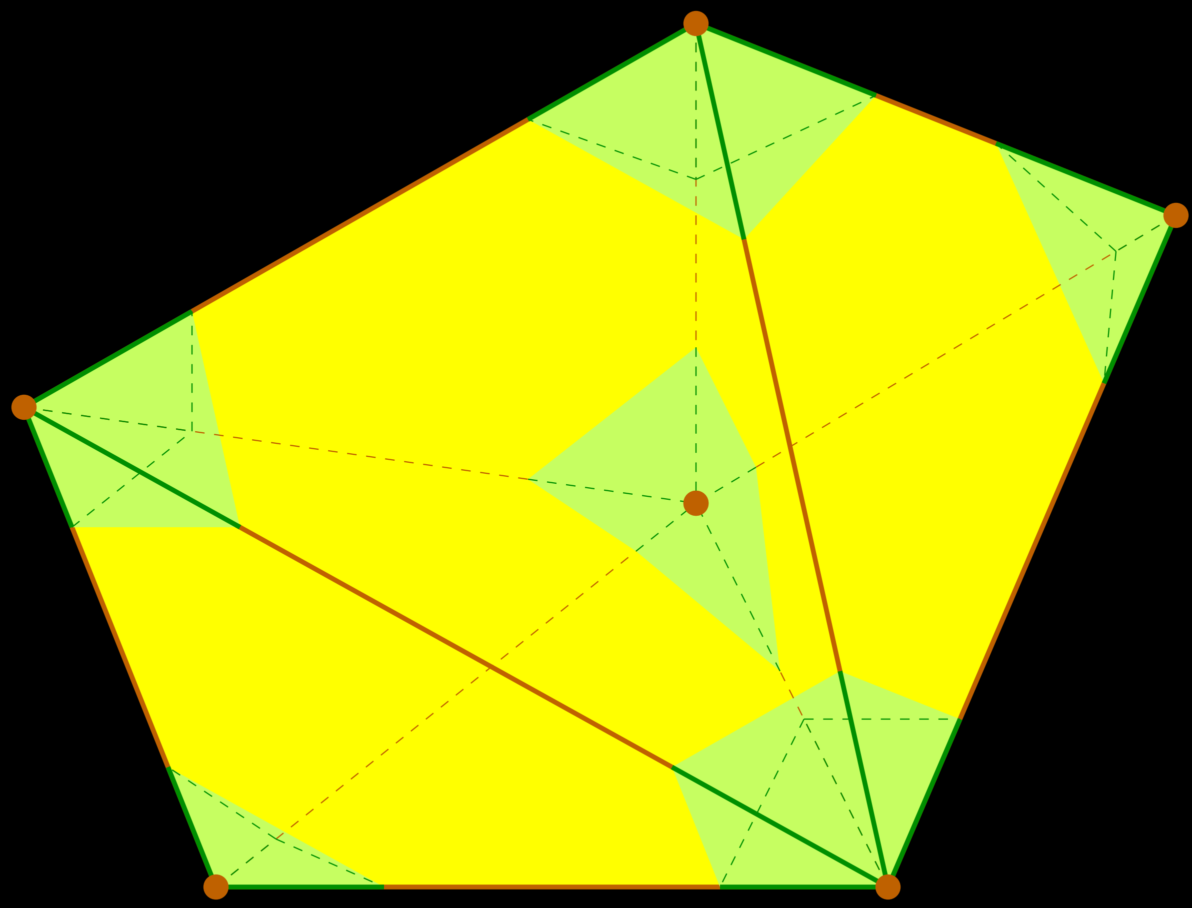


Computing the Convex Hull with Star Splaying



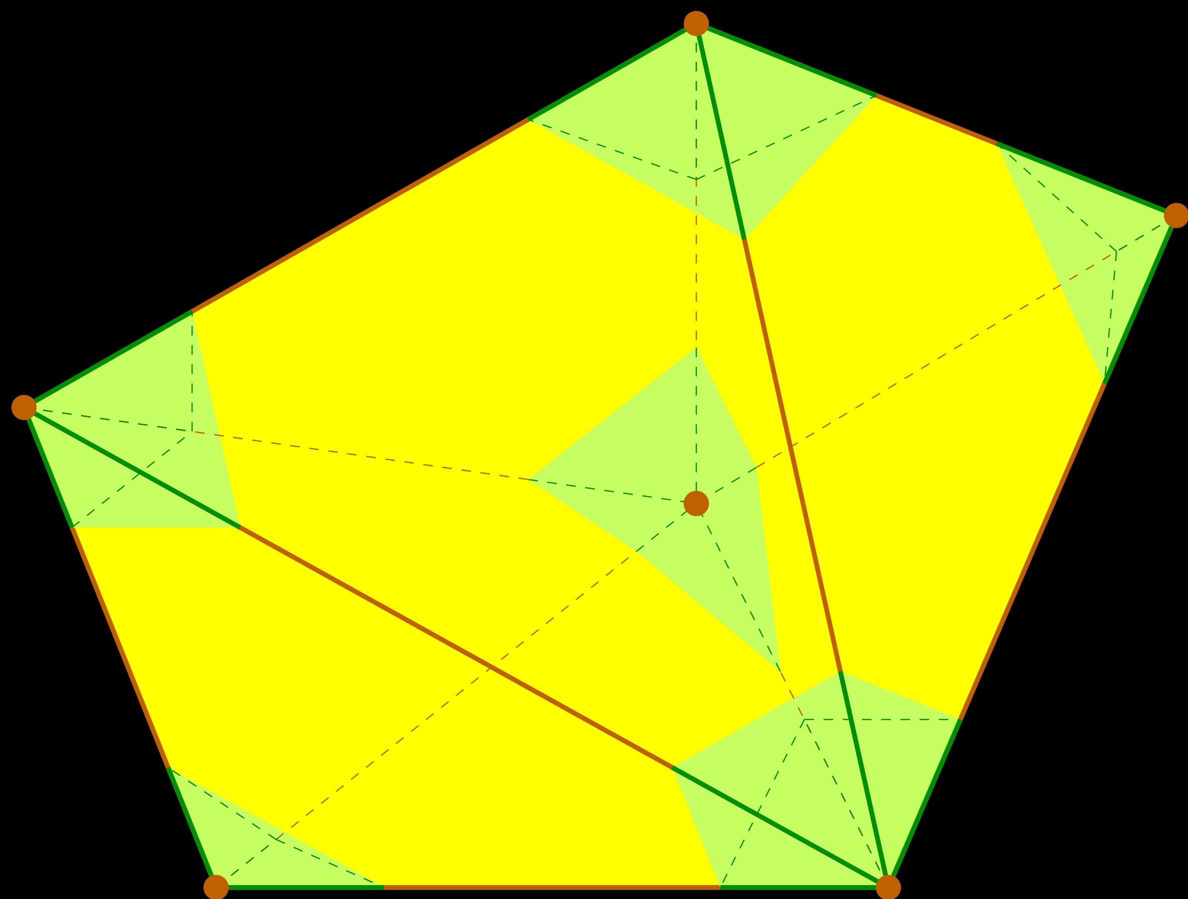
Termination Guarantee

- ☆ Stars splay open – they never get smaller.
- ☆ Consistency resolution always splays a star.



Star Splaying as Hill−Climbing

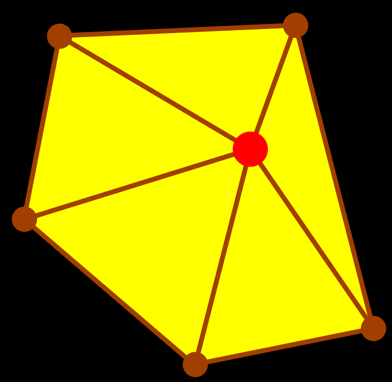
- ✫ Objective: maximize the sum of solid angles of the cones.



- ✫ Worst−case time: O(n^3).

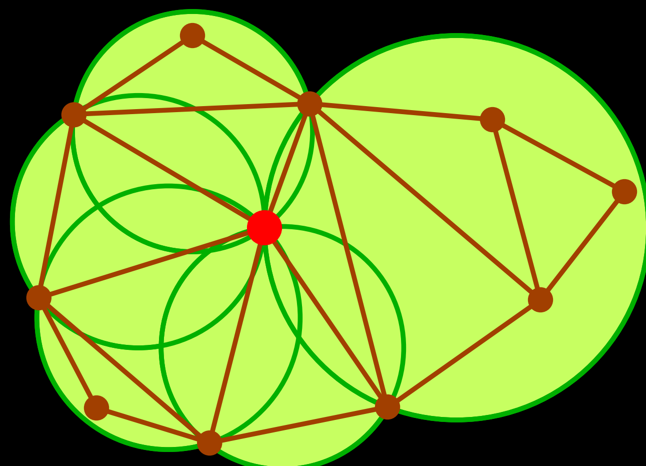
Repairing "Nearly Delaunay" Triangulations

- ✫ Suppose vertex degree is bounded by a constant.



Repairing "Nearly Delaunay" Triangulations

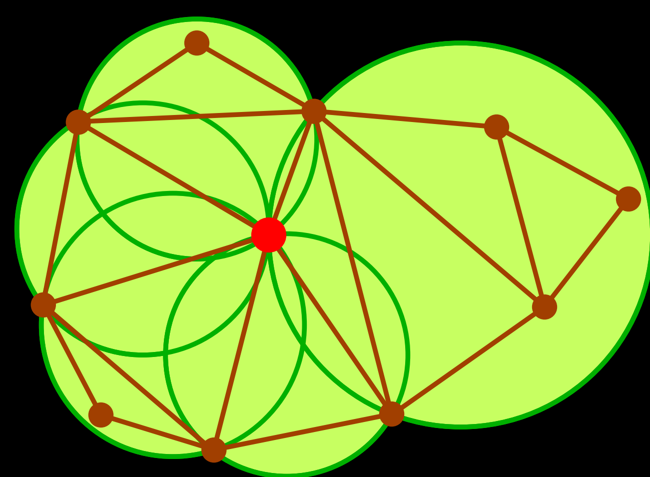
- ☆ Suppose vertex degree is bounded by a constant.
- ☆ Suppose every circumsphere encloses at most a constant number of vertices.



Repairing "Nearly Delaunay" Triangulations

☆ Suppose vertex degree is bounded by a constant.

☆ Suppose every circumsphere encloses at most a constant number of vertices.



☆ Worst−case time: O(n).

Repairing "Nearly Delaunay" Triangulations

- ☆ Suppose vertex degree is bounded by a constant.
- ☆ Suppose every circumsphere encloses at most a constant number of vertices.

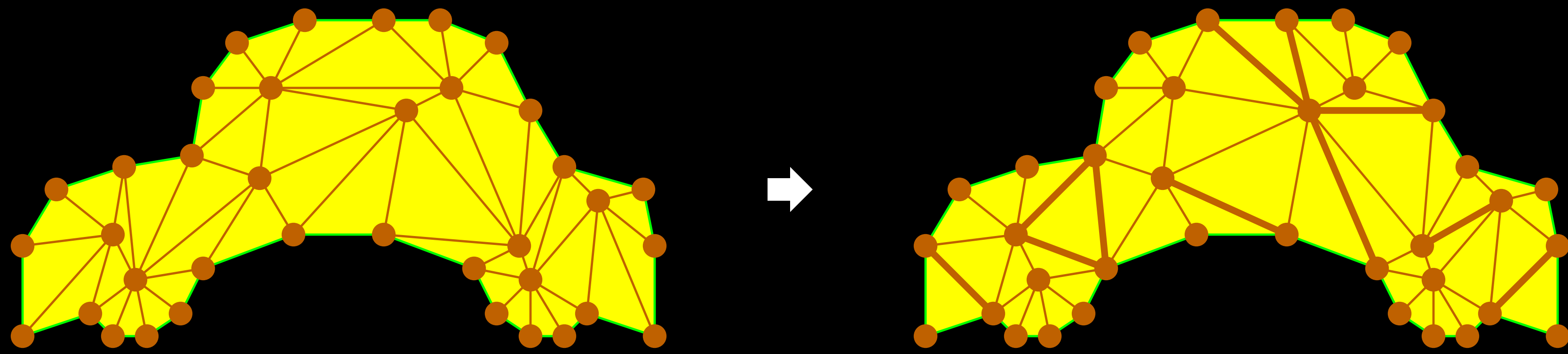
MAIN RESULT

- ☆ Worst−case time: O(n).

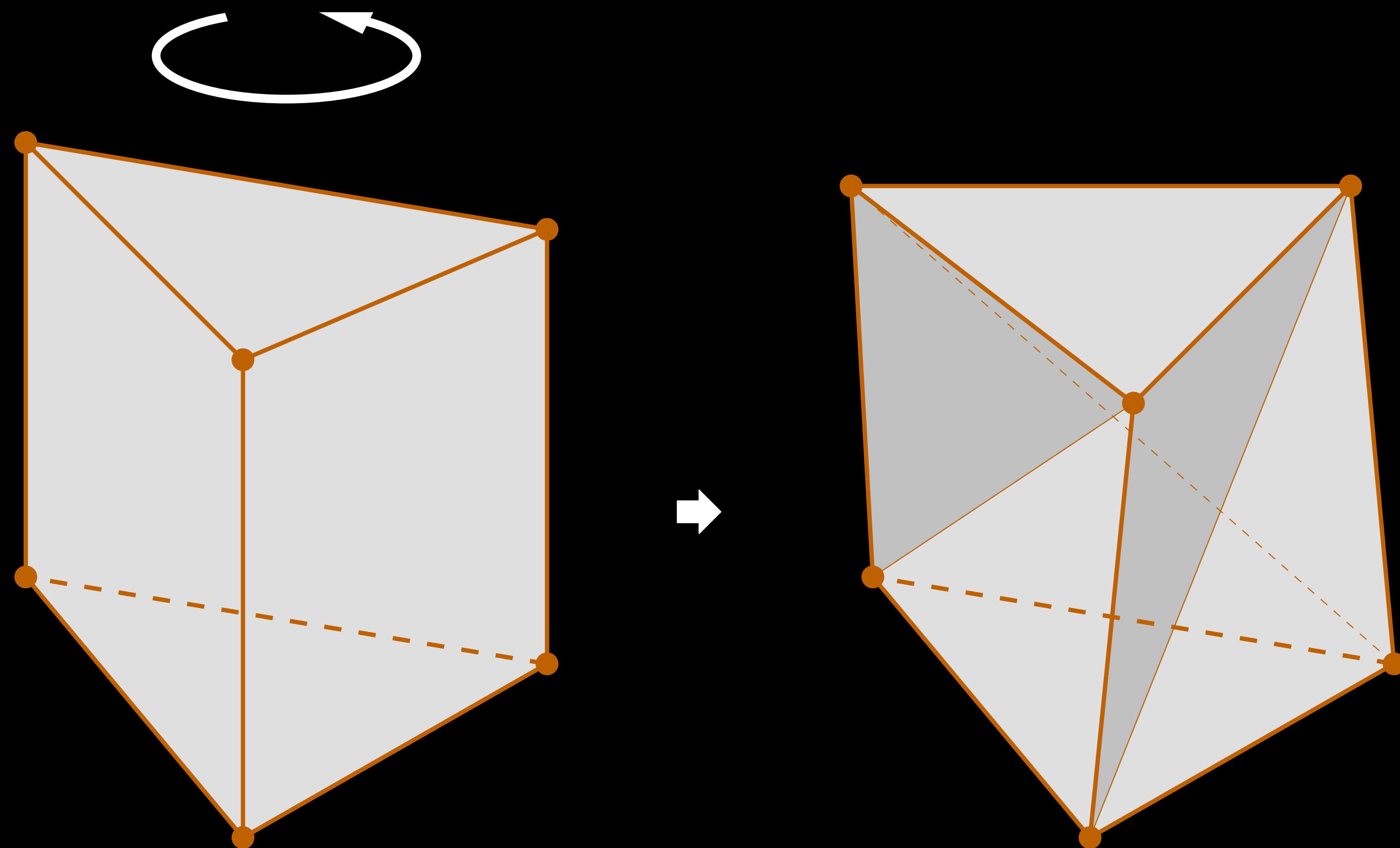
Constrained Triangulations

Constrained Delaunay Repair

Delaunay triangulations are convex. We really want to repair a *constrained* Delaunay triangulation.

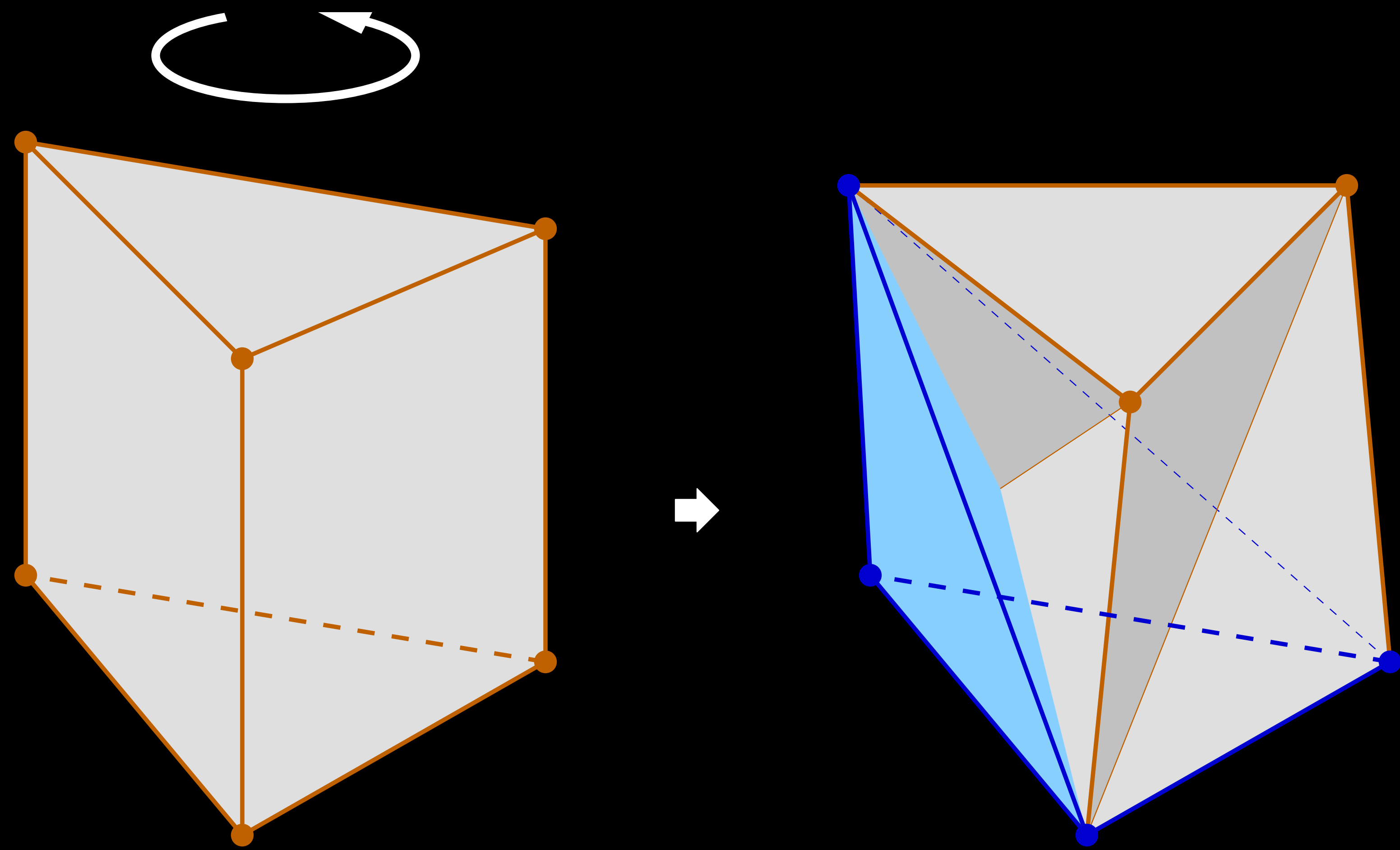


Some Polyhedra Have No Tetrahedralization



Schönhardt’s polyhedron

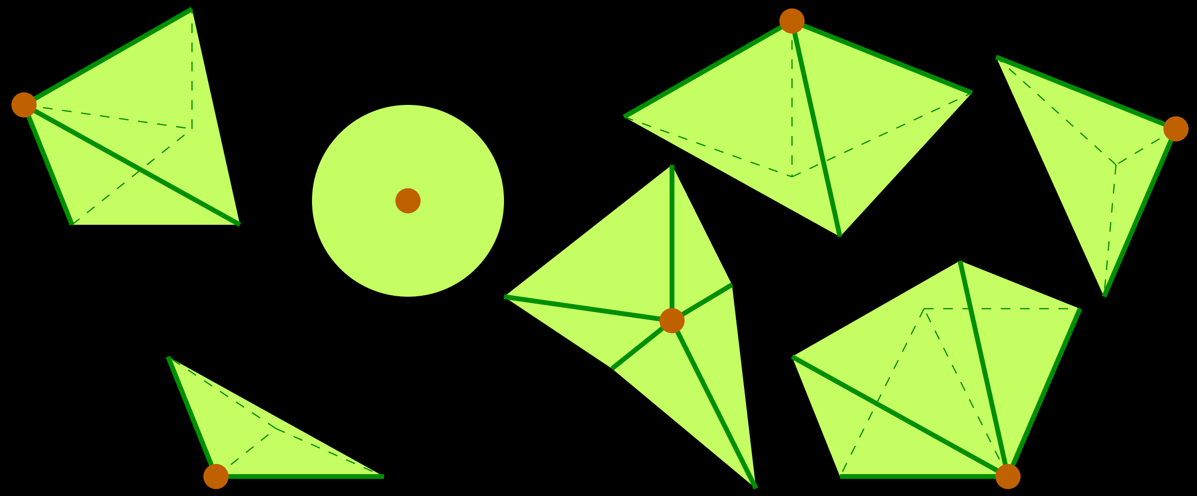
Some Polyhedra Have No Tetrahedralization



Any four vertices of Schönhardt’s polyhedron yield a tetrahedron that sticks out a bit.

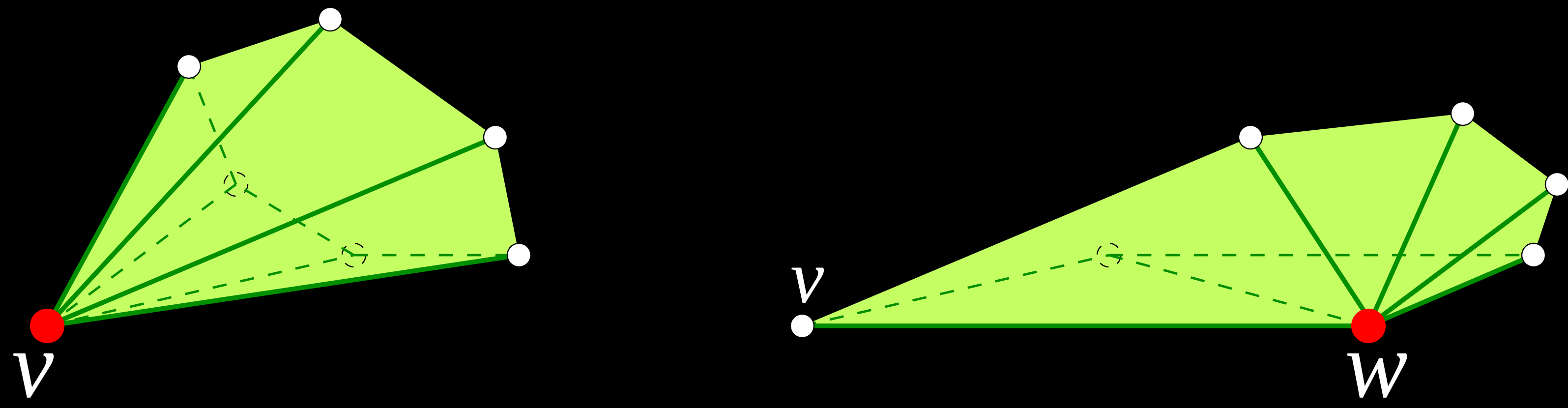
Idea #1

The representation is a collection of stars.

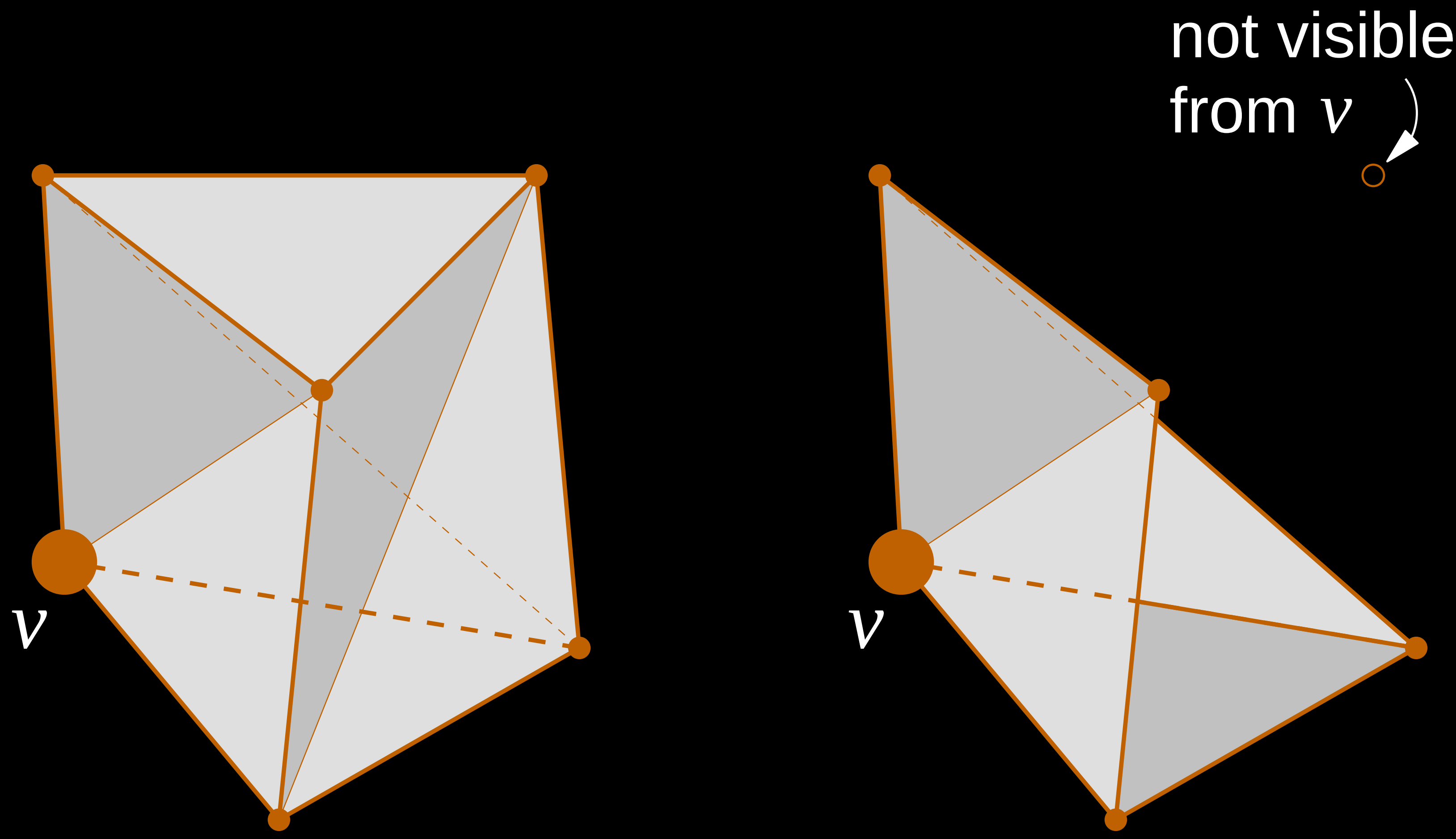


Idea #2

Stars may be *inconsistent* with each other.

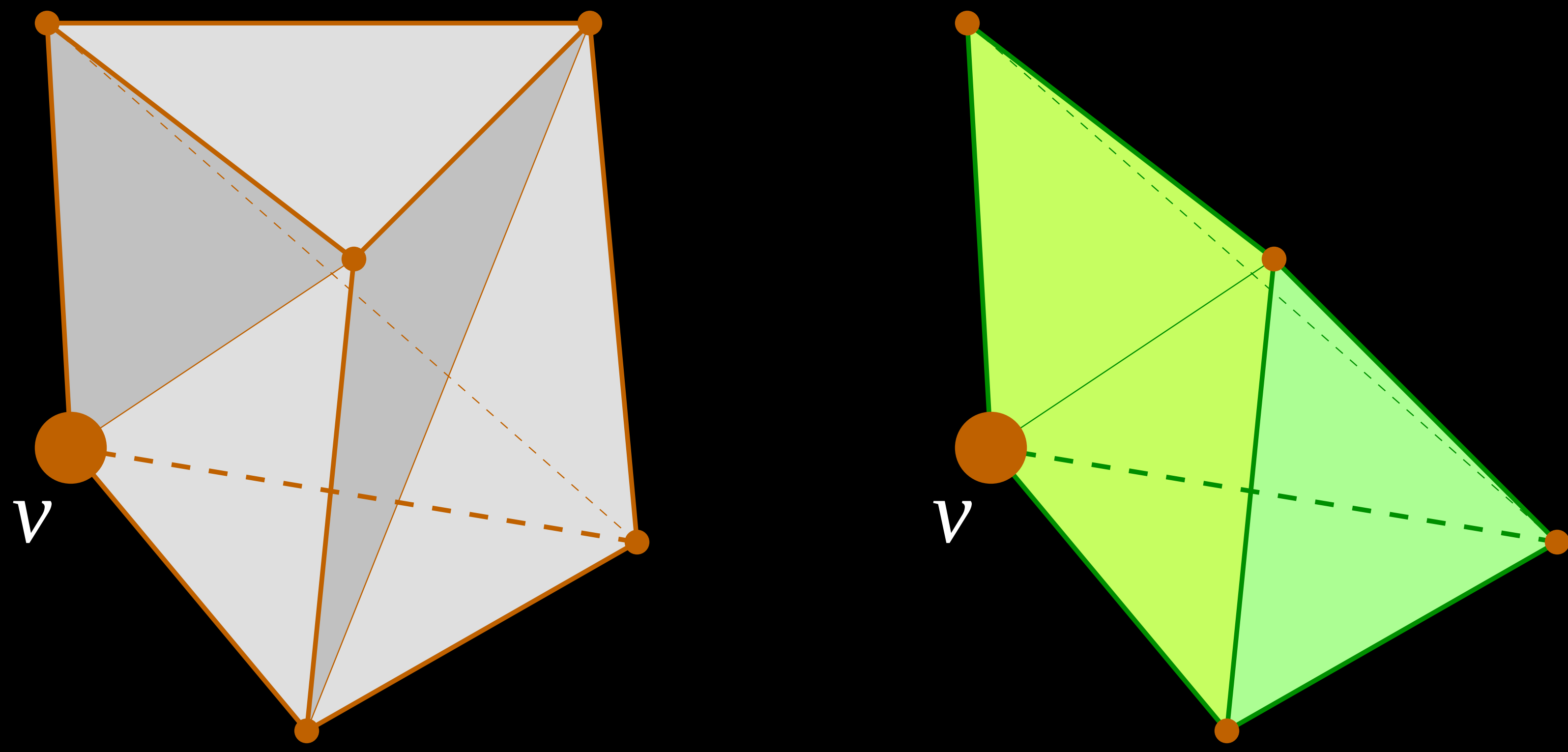


The View from One Vertex



Ignore vertices not visible from v,
faces not connected to v.

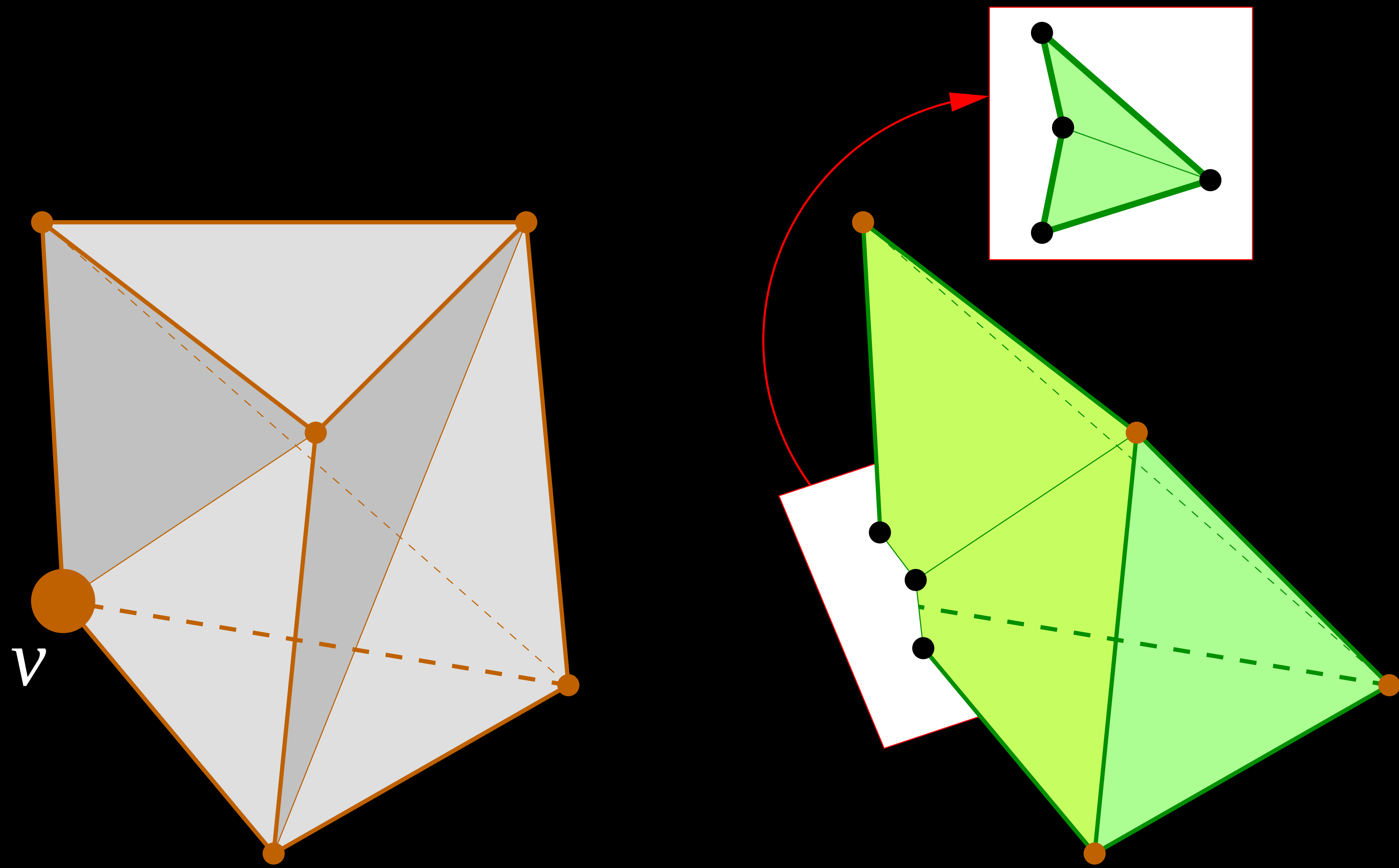
The View from One Vertex



Find a star tetrahedralization for *v*.

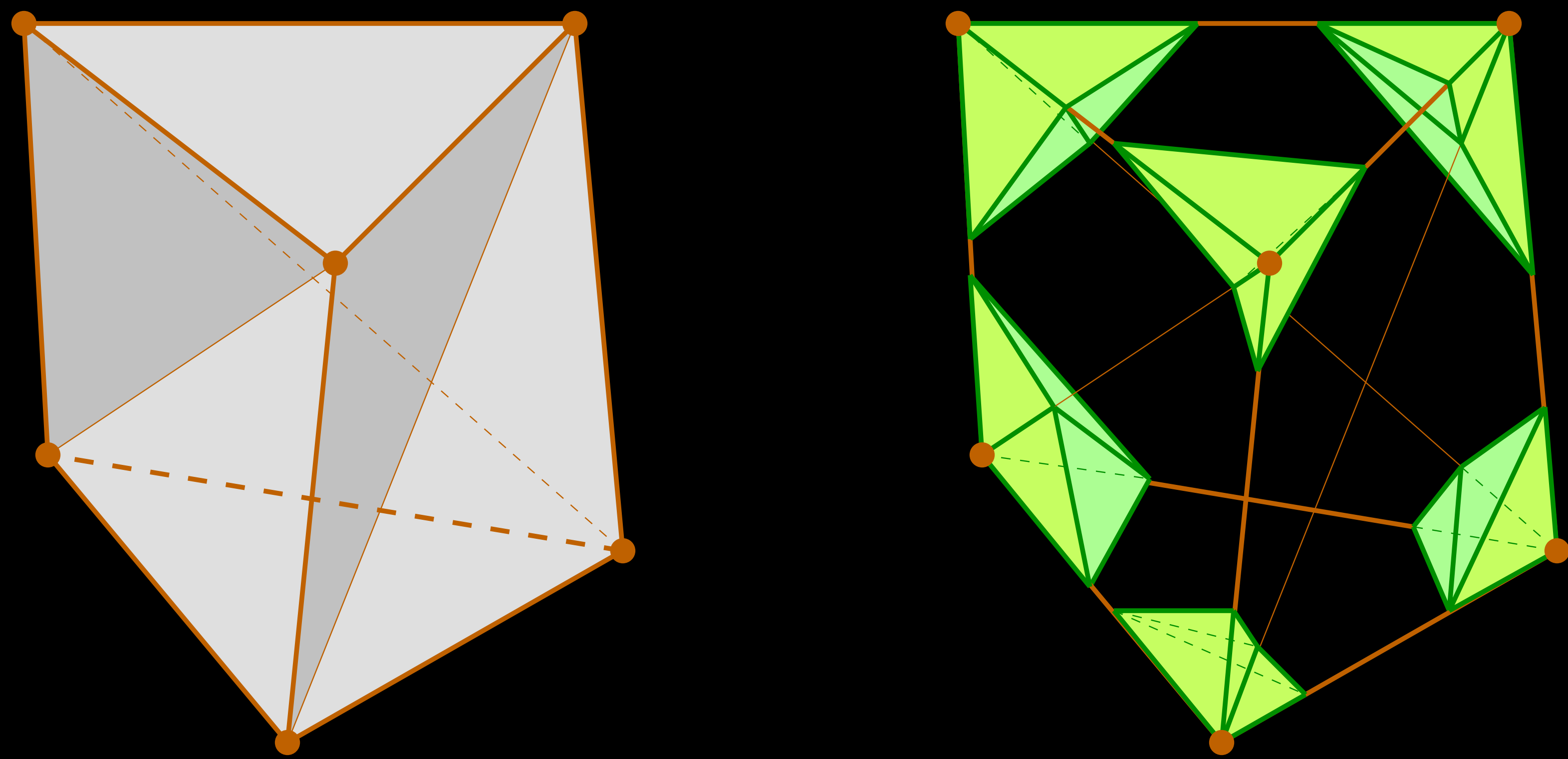
May disrespect faces not adjoining *v*.

The View from One Vertex



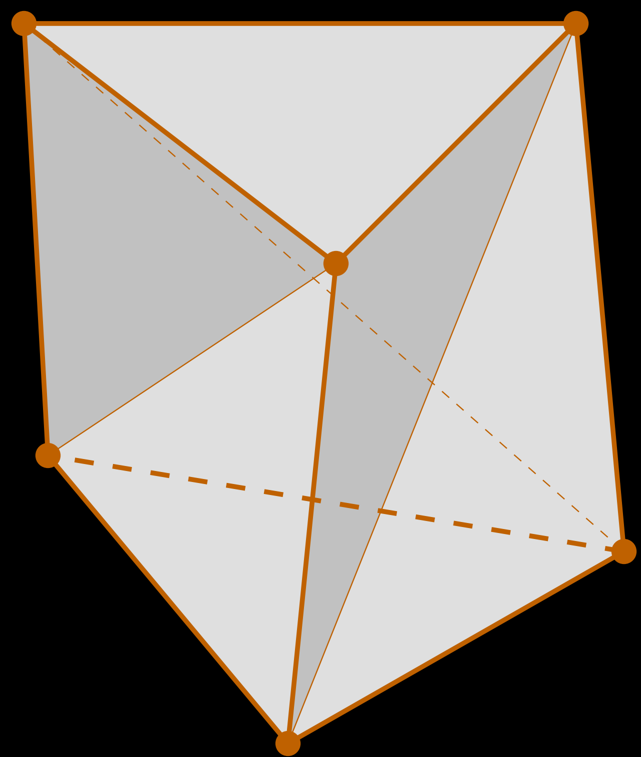
v's star tetrahedralization is really a 2D constrained triangulation.

Star Representation of Schönhardt

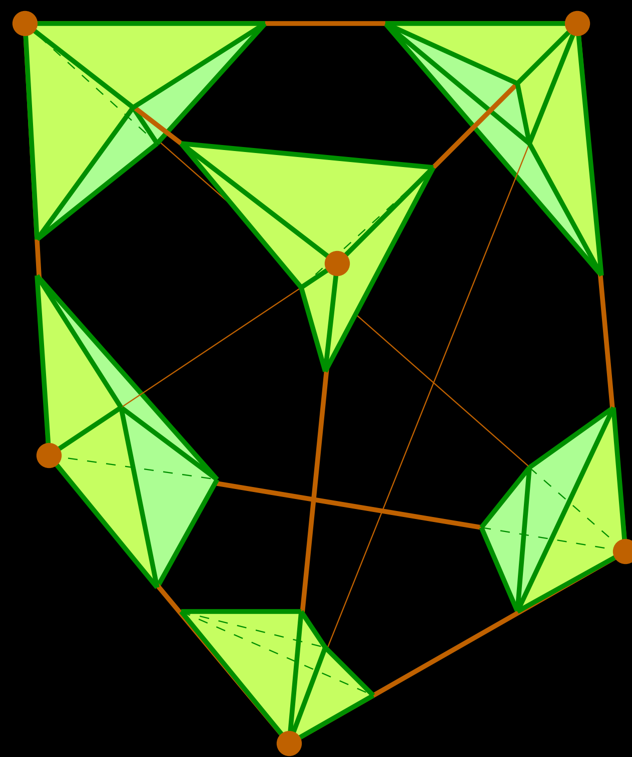


The constrained vertex stars are inconsistent with each other.

Star Representation of Schönhardt



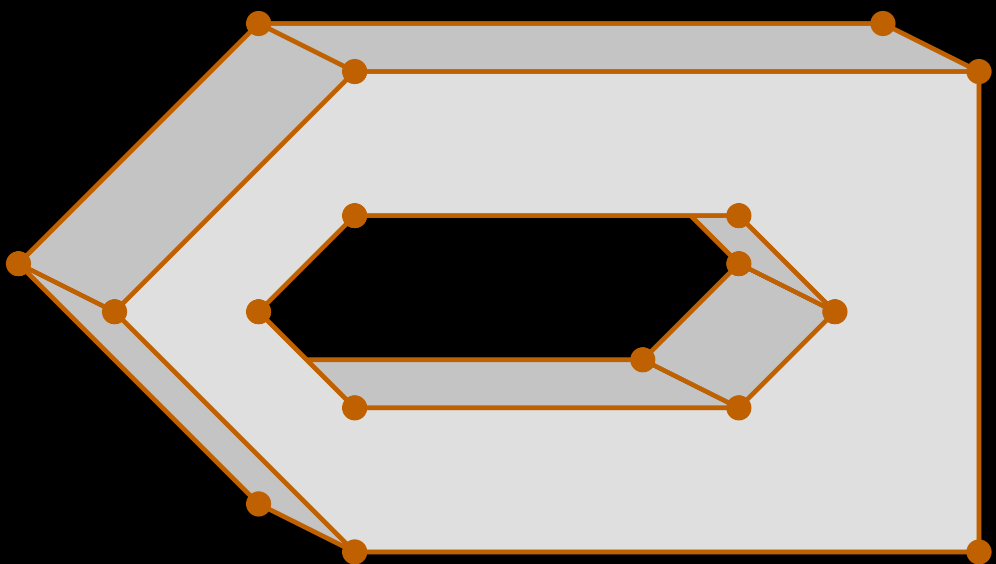
“Tangulation”

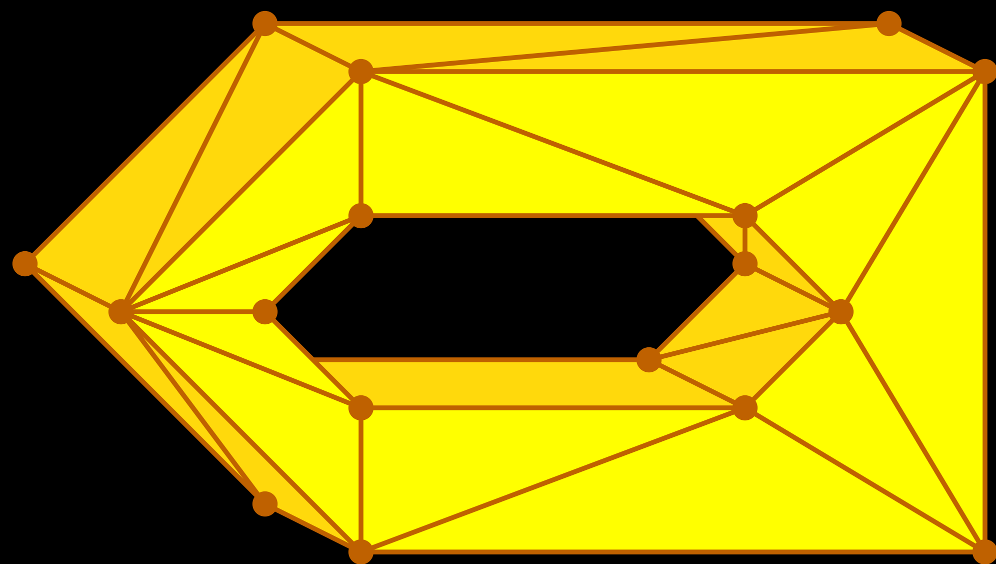


The constrained vertex stars are inconsistent with each other.

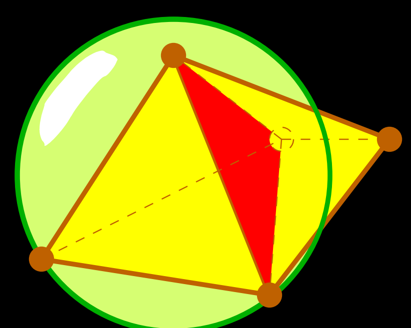
Constrained Delaunay Triangulations

Constrained Delaunay Triangulations (CDTs)





A tetrahedralization of a polyhedron is *constrained Delaunay* if every triangular face is locally Delaunay.



CDT Repair

A “tangulation” represents a CDT if

✫1✫ Each vertex’s star respects the polyhedron faces adjoining that vertex.

CDT Repair

A “tangulation” represents a CDT if

✫1✫ Each vertex’s star respects the polyhedron faces adjoining that vertex.

✫2✫ Every triangular face in every star is locally Delaunay.

CDT Repair

A ‘‘tangulation’’ represents a CDT if

✫1✫ Each vertex’s star respects the polyhedron faces adjoining that vertex.

✫2✫ Every triangular face in every star is locally Delaunay.

✫3✫ The stars are consistent with each other.

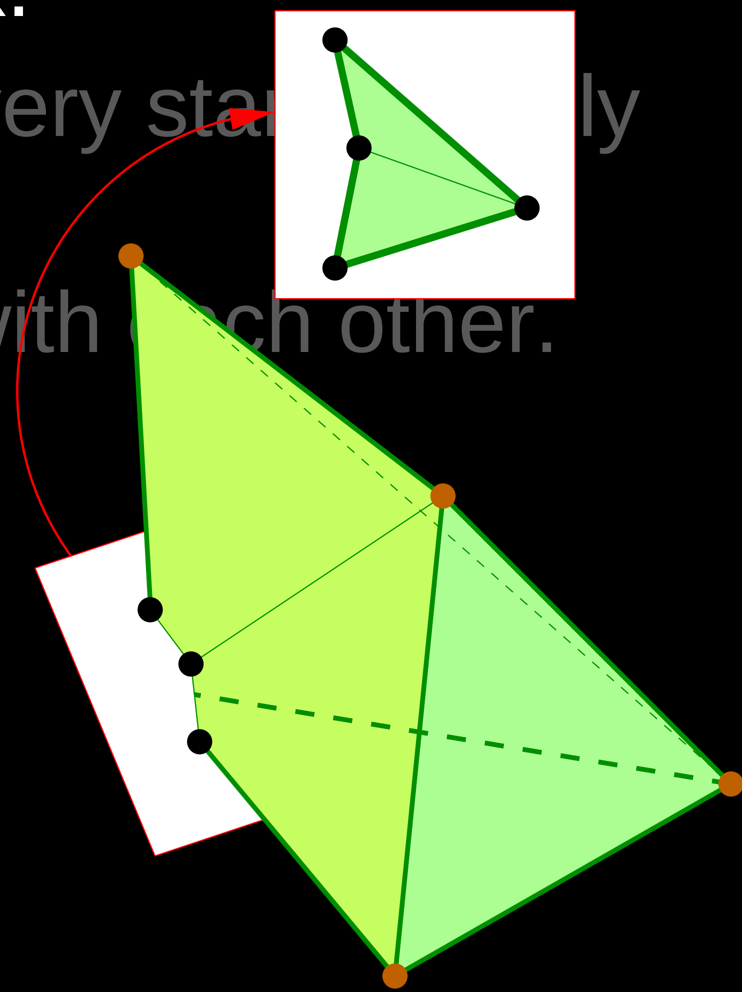
CDT Repair

A "tangulation" represents a CDT if

☆1☆ Each vertex's star respects the polyhedron faces adjoining that vertex.

☆2☆ Every triangular face in every star is locally Delaunay.

☆3☆ The stars are consistent with each other.



A 3D star is really a 2D polygon triangulation. Every polygon has a triangulation.

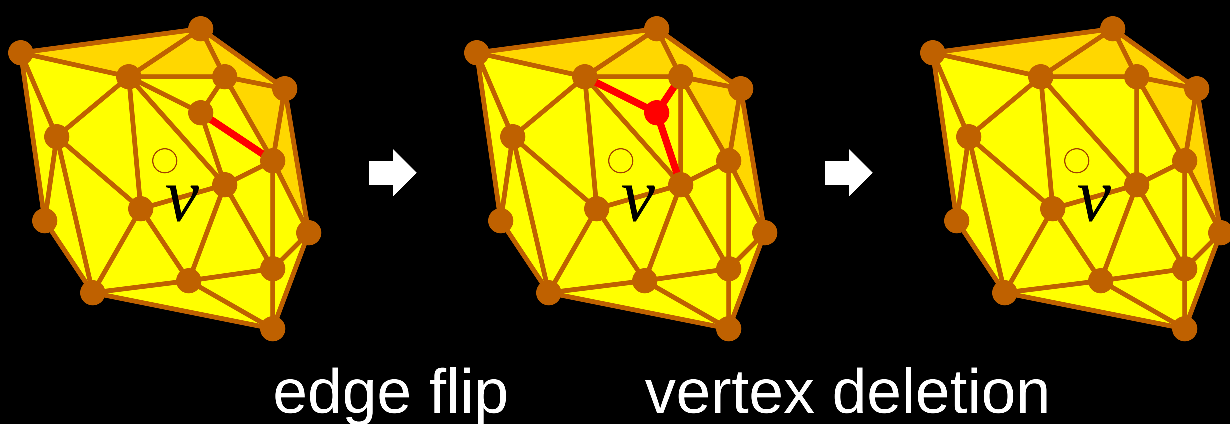
CDT Repair

A “tangulation” represents a CDT if

✩1✩ Each vertex’s star respects the polyhedron faces adjoining that vertex.

✩2✩ Every triangular face in every star is locally Delaunay.

✩3✩ The stars are consistent with each other.



Use 2D flip algorithm in stars; makes most stars Delaunay.

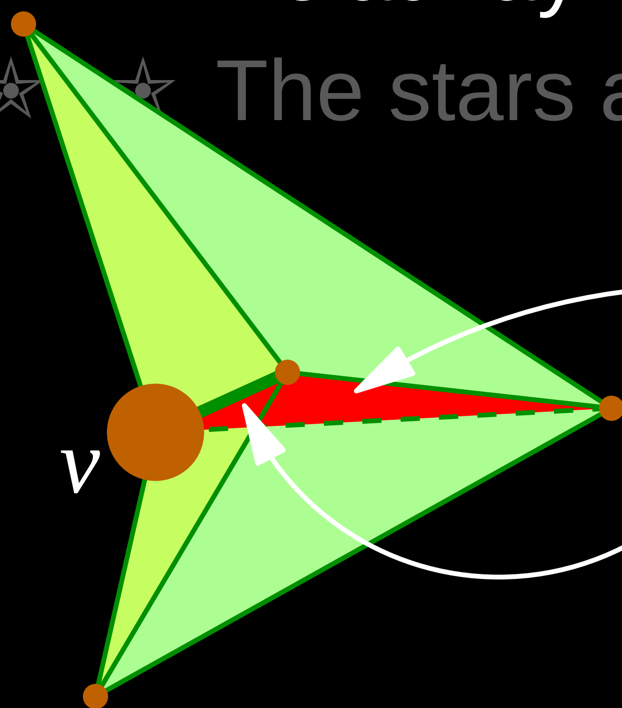
CDT Repair

A "tangulation" represents a CDT if

☆1☆ Each vertex's star respects the polyhedron faces adjoining that vertex.

☆2☆ Every triangular face in every star is locally Delaunay.

☆ ☆ The stars are consistent with each other.



This face is not locally Delaunay, but it can't be flipped because of this reflex edge of the polygon.

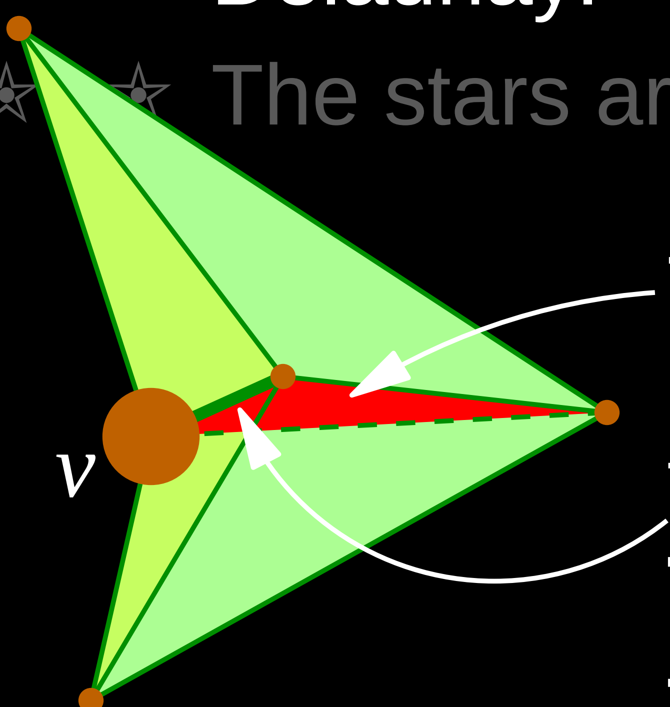
CDT Repair

A “tangulation” represents a CDT if

✫1✫ Each vertex’s star respects the polyhedron faces adjoining that vertex.

✫2✫ Every triangular face in every star is locally Delaunay.

✫ ✫ The stars are consistent with each other.



This face is not locally Delaunay, but it can’t be flipped because of this reflex edge of the polygon.

The tangulation diagnoses why the polygon has no CDT.

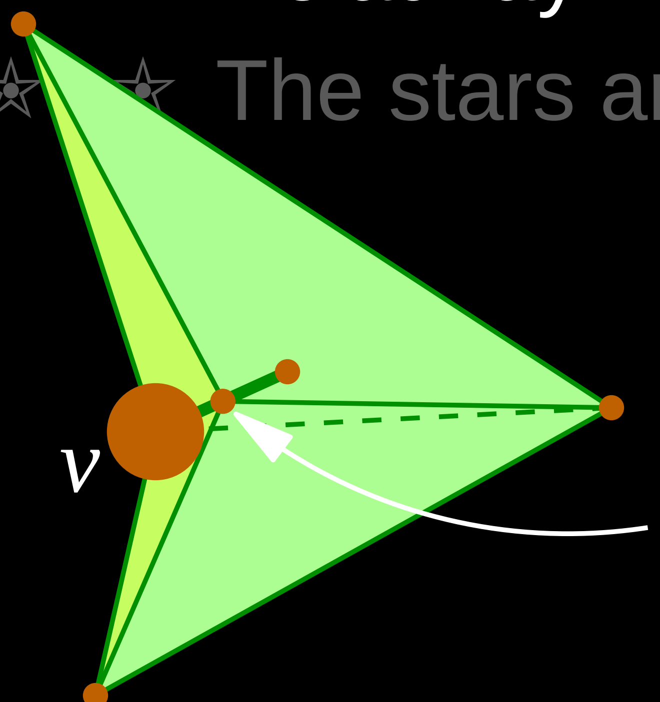
CDT Repair

A "tangulation" represents a CDT if

☆1☆ Each vertex's star respects the polyhedron faces adjoining that vertex.

☆2☆ Every triangular face in every star is locally Delaunay.

☆3☆ The stars are consistent with each other.



Possible solution: insert new vertex.

CDT Repair

A “tangulation” represents a CDT if

☆1☆ Each vertex’s star respects the polyhedron faces adjoining that vertex.

☆2☆ Every triangular face in every star is locally Delaunay.

☆3☆ The stars are consistent with each other.

The hard part! Use consistency resolution, but...

CDT Repair

A "tangulation" represents a CDT if

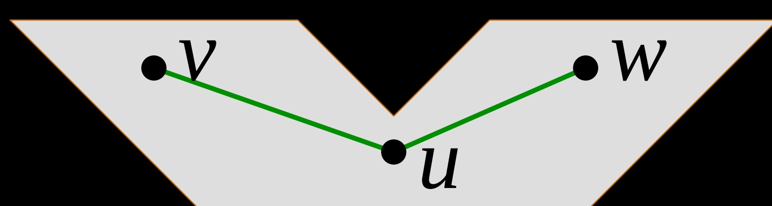
☆1☆ Each vertex's star respects the polyhedron faces adjoining that vertex.

☆2☆ Every triangular face in every star is locally Delaunay.

☆3☆ The stars are consistent with each other.

The hard part! Use consistency resolution, but...

☆ If two vertices can't see each other in the polyhedron, they can't be in each others' stars!



CDT Repair

A “tangulation” represents a CDT if

☆1☆ Each vertex’s star respects the polyhedron faces adjoining that vertex.

☆2☆ Every triangular face in every star is locally Delaunay.

☆3☆ The stars are consistent with each other.

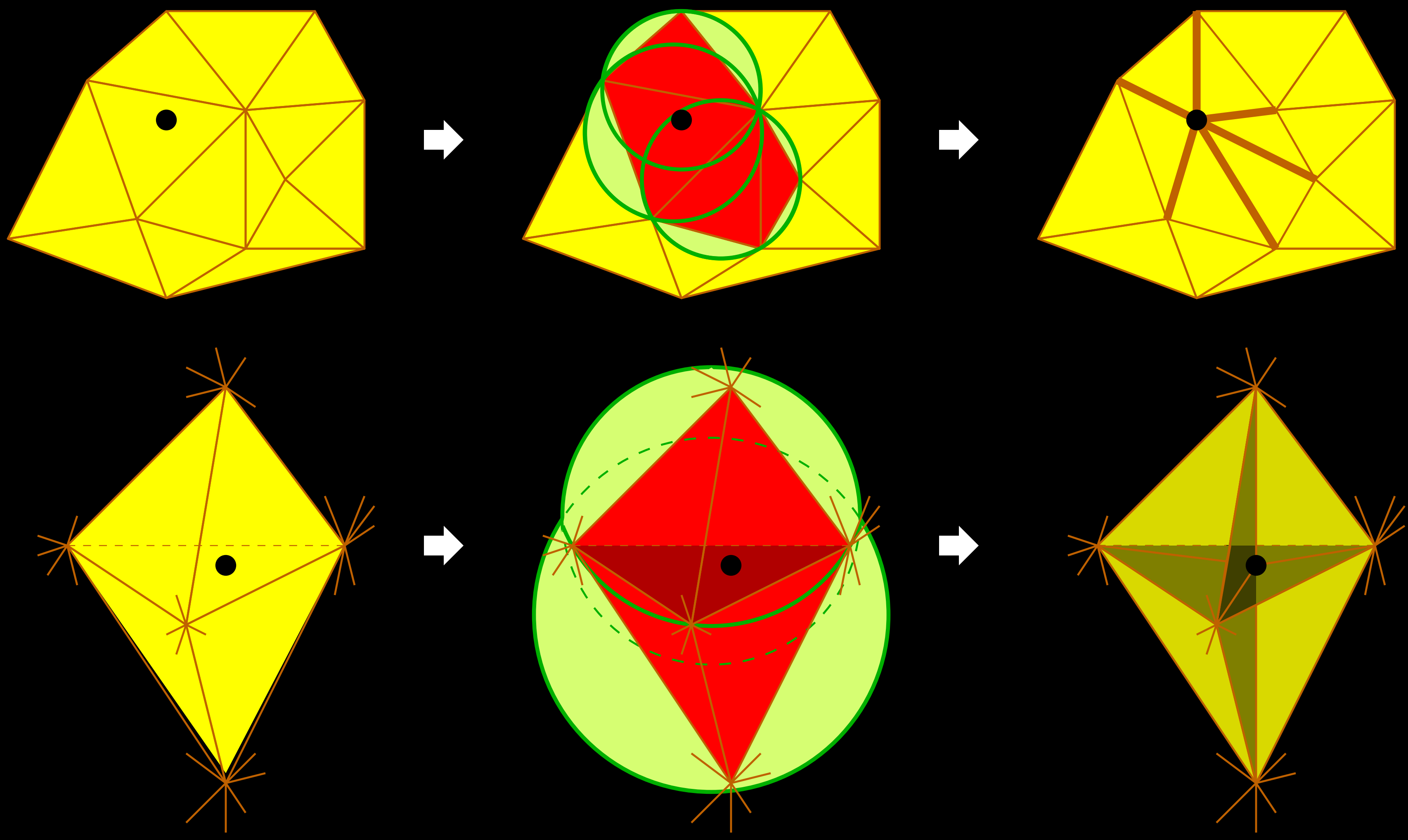
The hard part! Use consistency resolution, but...

☆ If two vertices can’t see each other in the polyhedron, they can’t be in each others’ stars!

☆ Two vertices that should be connected by an edge might fail to find each other. Global search...

Incremental CDT Update

Bowyer–Watson Algorithm

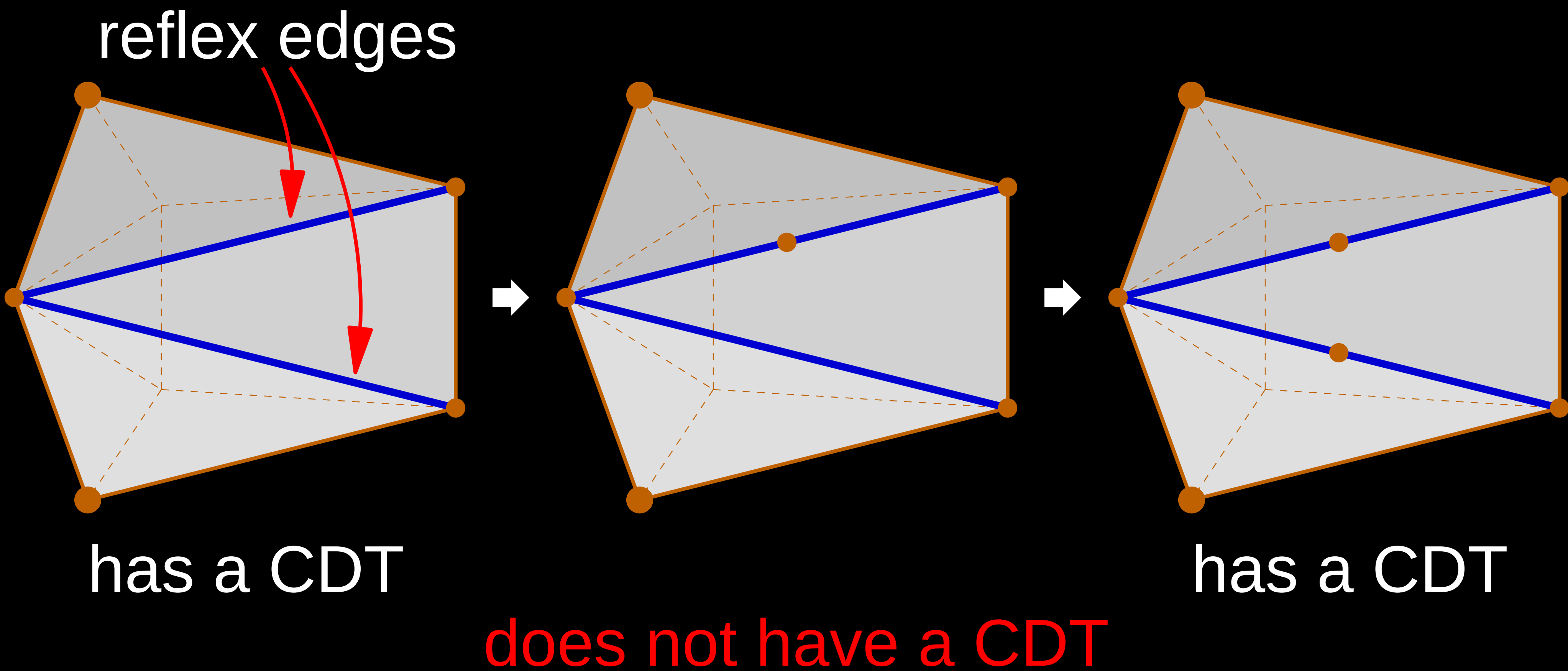


Insert one vertex at a time.

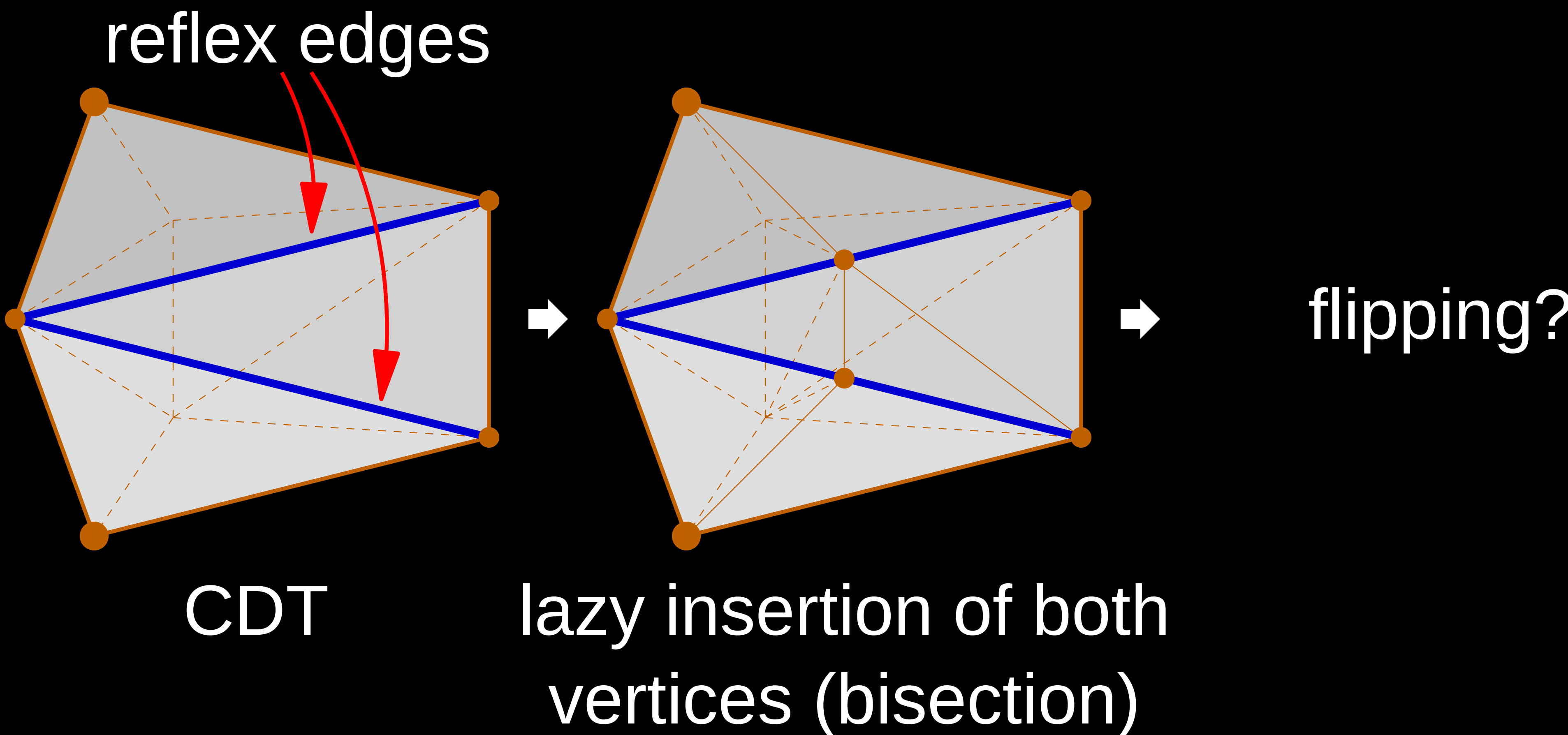
Remove all triangles/tetrahedra that are no longer Delaunay.

Retriangulate the cavity with a fan around the new vertex.

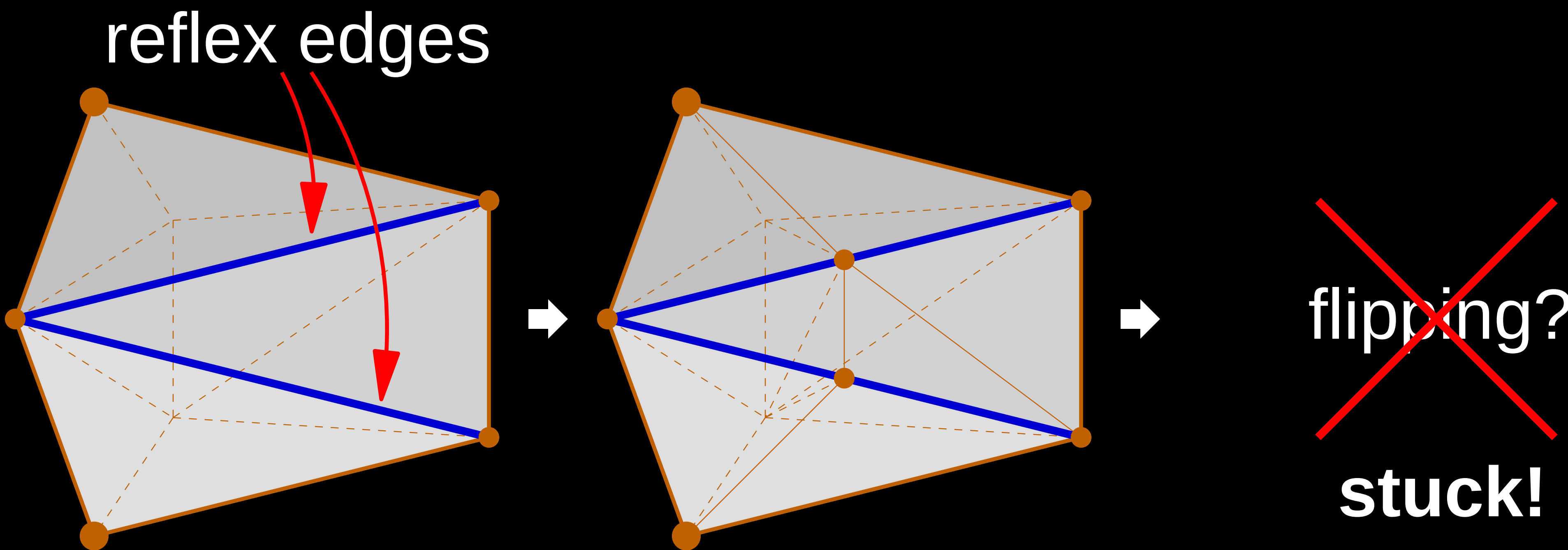
Inserting Vertices into 3D CDTs



Inserting Vertices into 3D CDTs



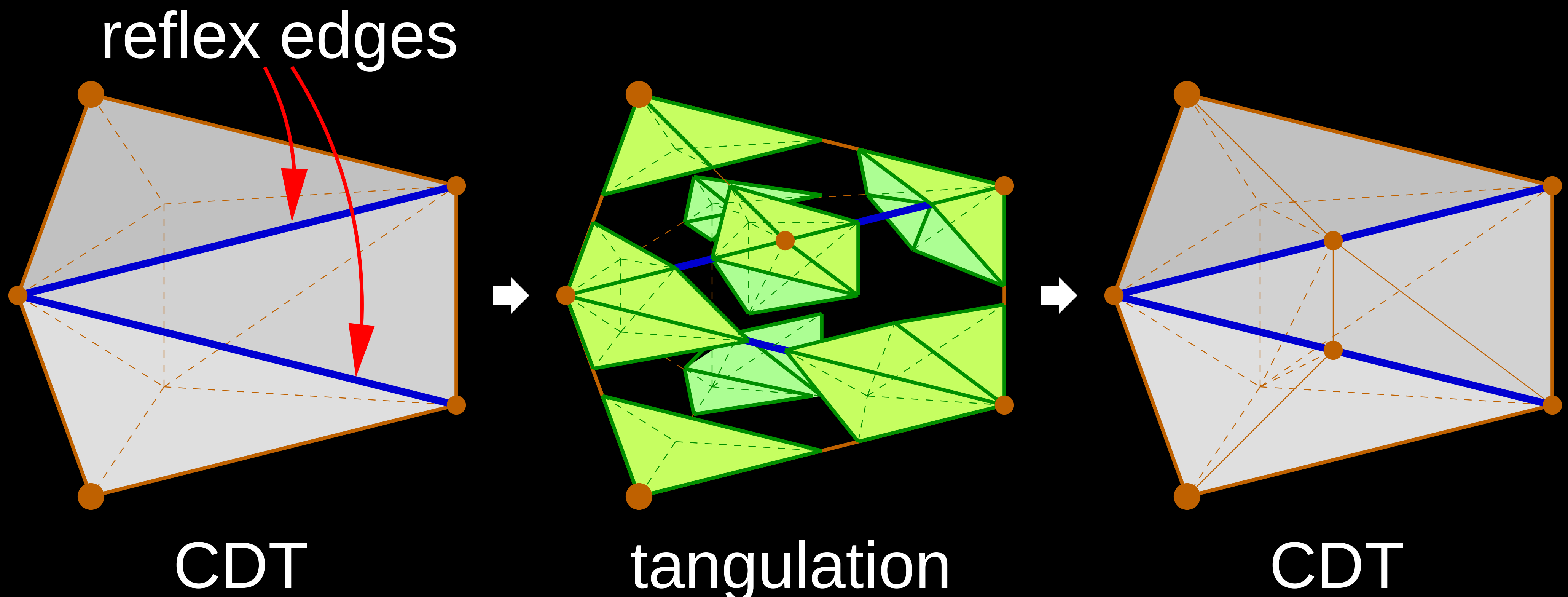
Inserting Vertices into 3D CDTs



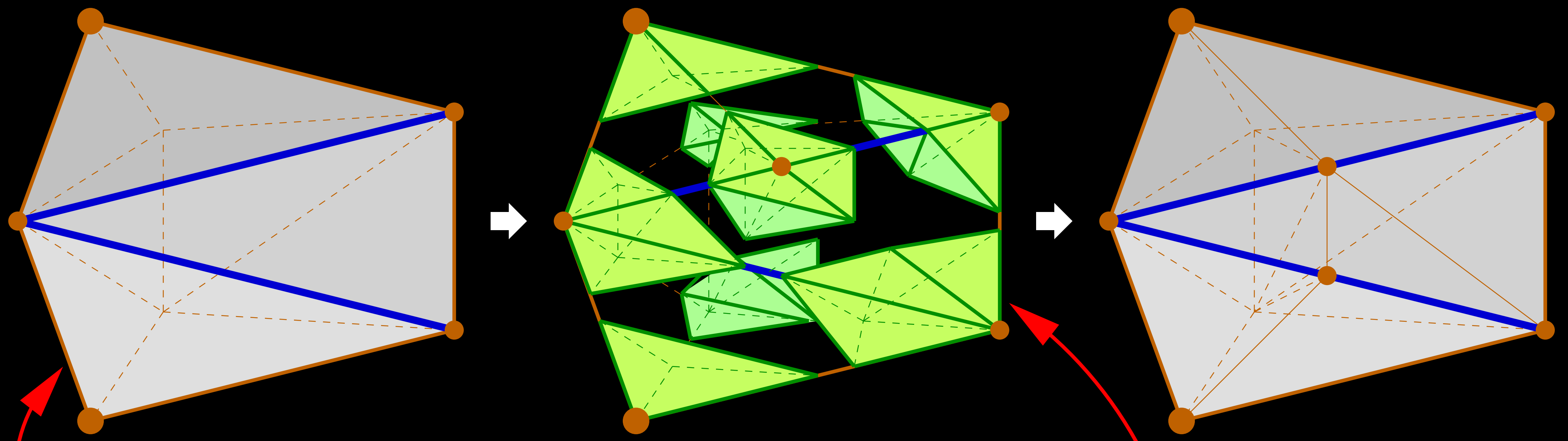
CDT

lazy insertion of both vertices (bisection)

Inserting Vertices into 3D CDTs



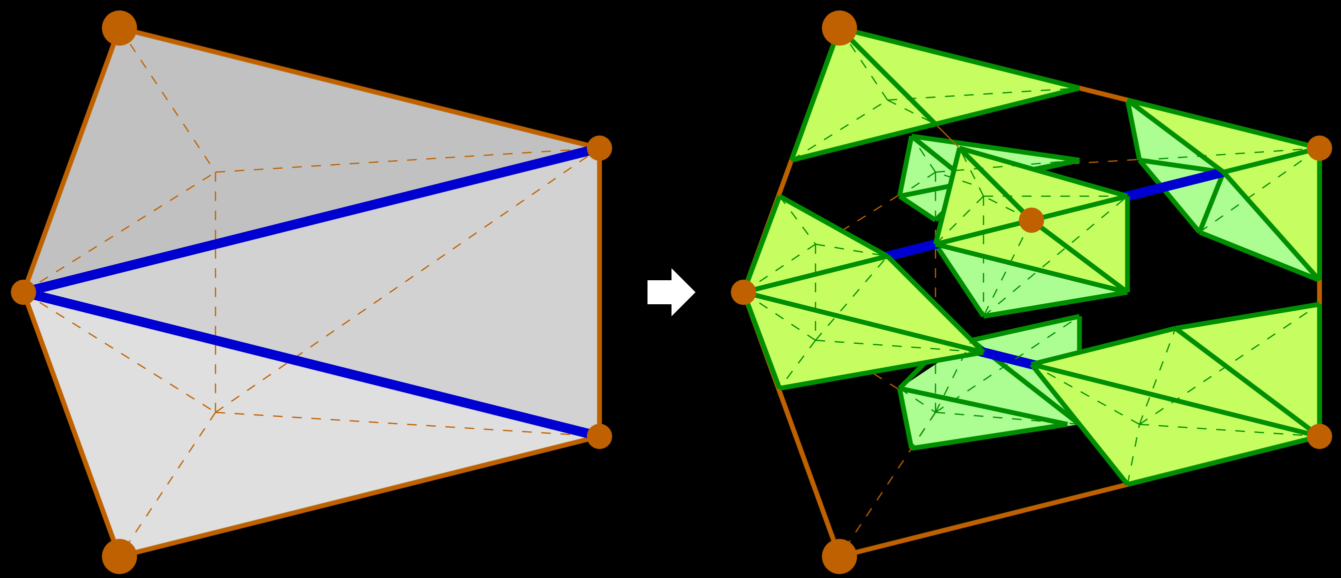
Inserting Vertices into 3D CDTs



Idea: maintain two copies of the triangulation.

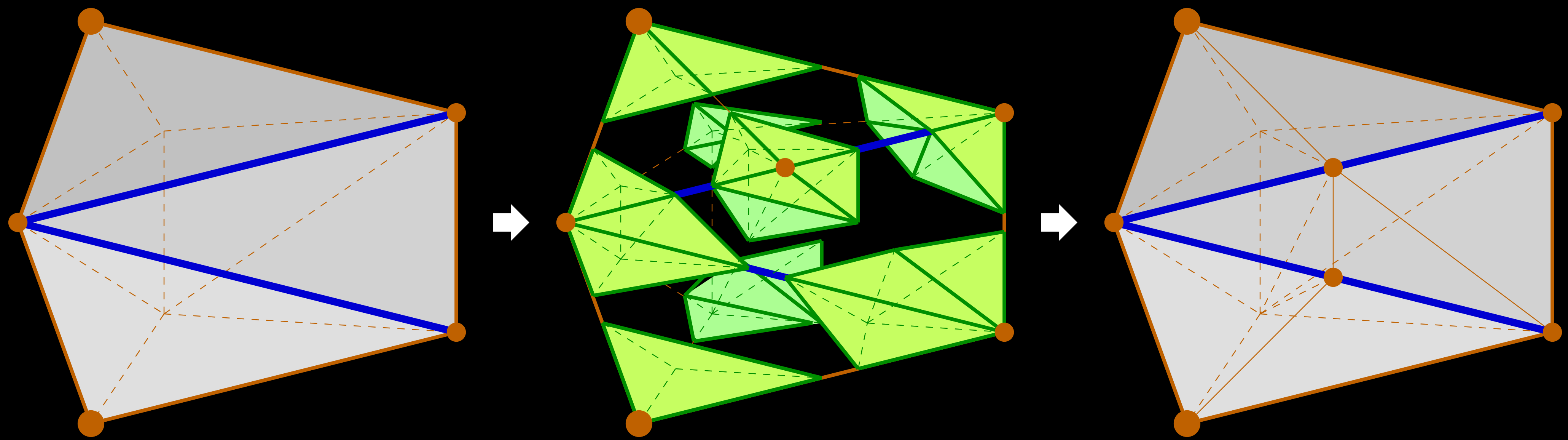
- ✫ Use the last internally consistent triangulation for visibility & Bowyer−Watson tests.
- ✫ Perform updates on the tangulation; use it to decide where to add new vertices.

Inserting Vertices into 3D CDTs



Actually, just maintain the last consistent triangulation, plus copies of the stars that have changed.

Inserting Vertices into 3D CDTs



Incrementally insert vertices. Loop:

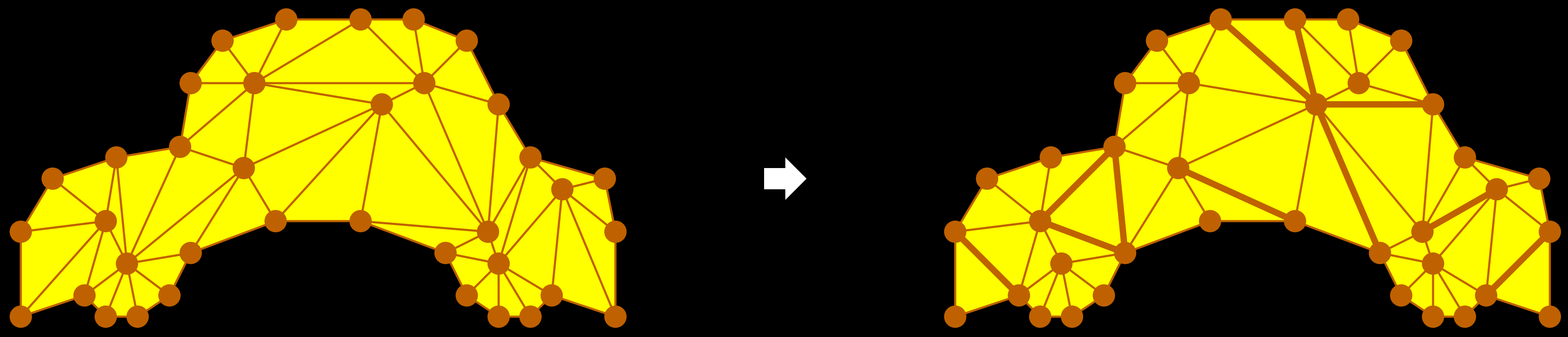
- ✫ New stars decide where to insert a vertex.
- ✫ Old stars tests visibility & which stars change.
- ✫ Insert new vertex into affected new stars.

Conclusions

☆ Star splaying performs Delaunay repair without getting stuck – sometimes in linear time.

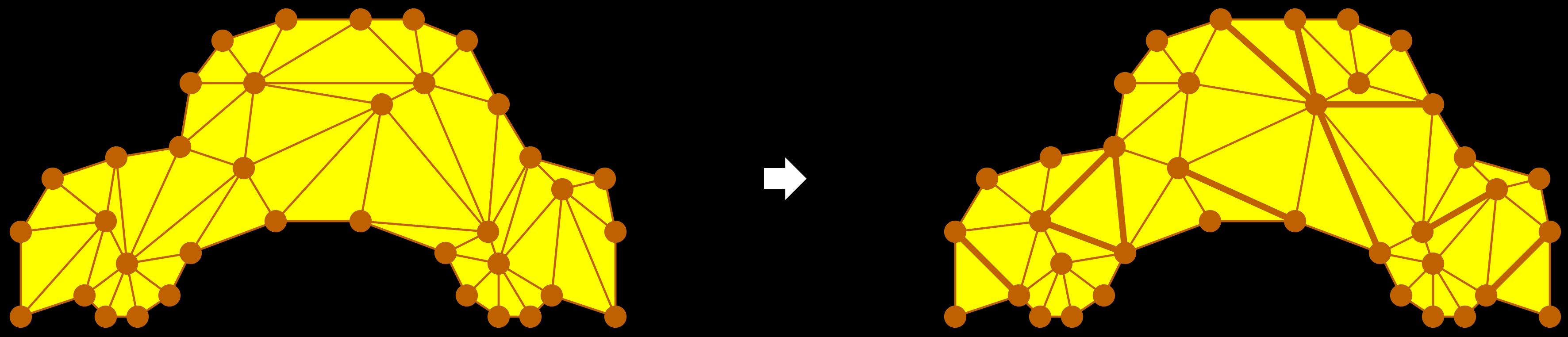
☆ Star splaying enables incremental vertex insertion in 3D CDTs, even through intermediate states where no CDT exists.

☆ Unfortunately, CDT repair seems much harder to do "locally" than Delaunay repair.



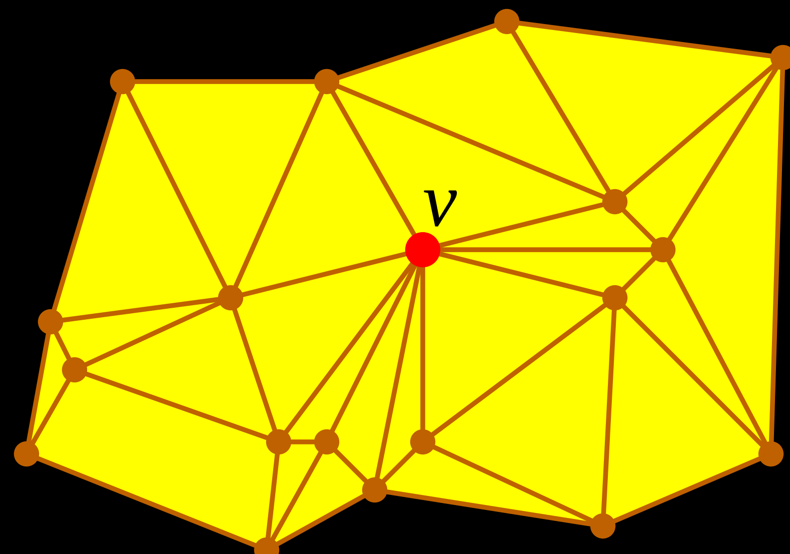
Open Problem

☆ Repair only the parts of the mesh where the quality is bad.

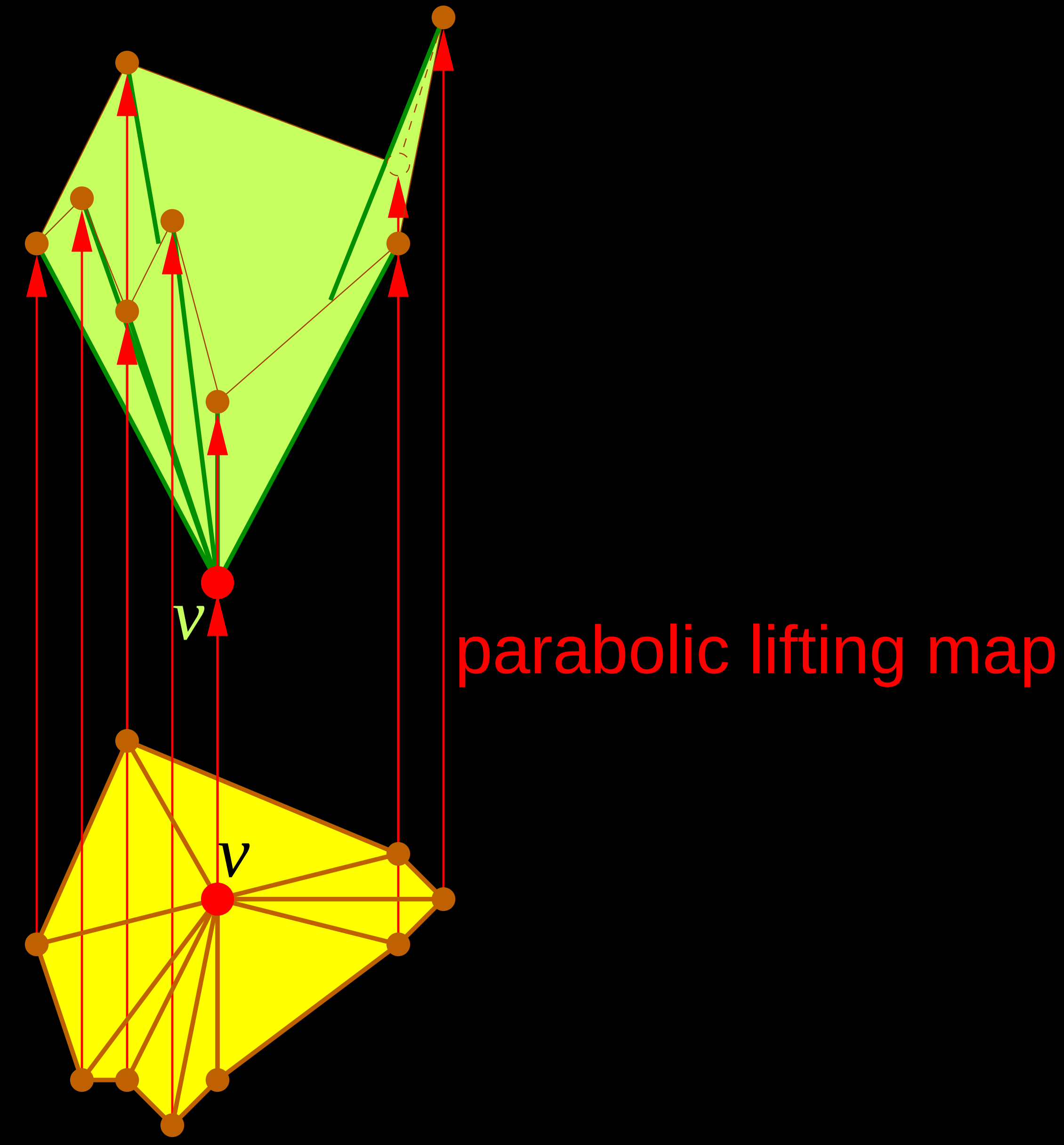


Star Flipping

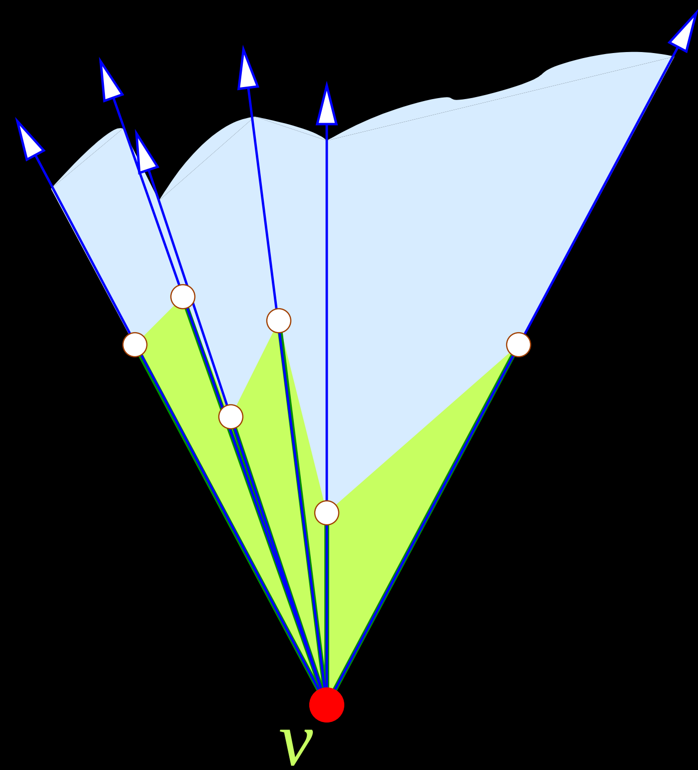
Each Vertex Has a Starting Cone

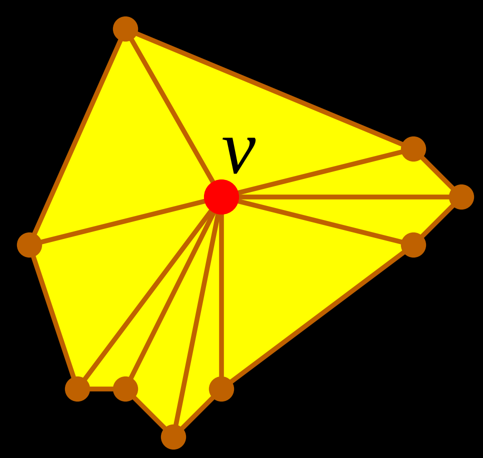


Each Vertex Has a Starting Cone

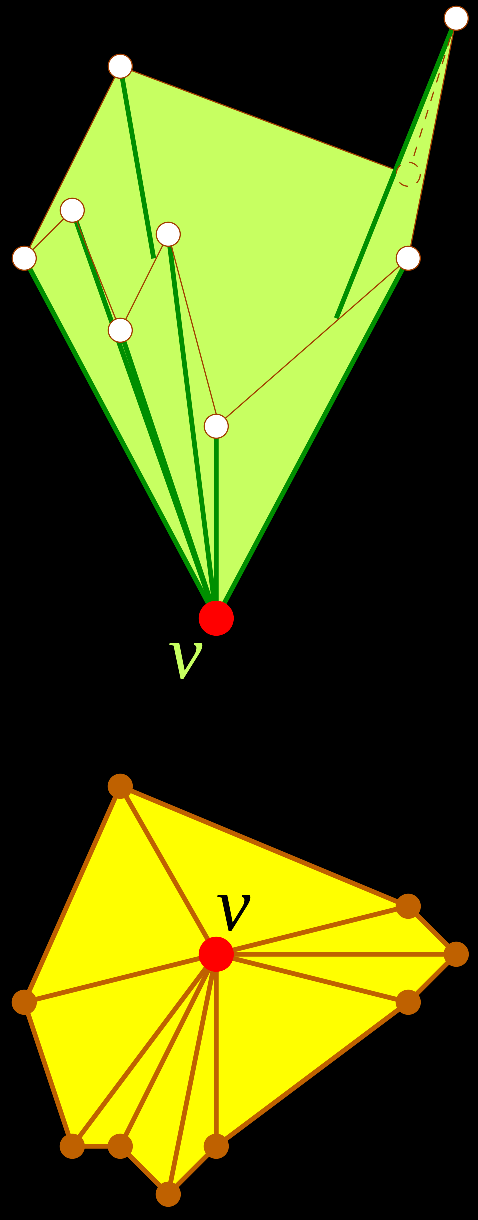


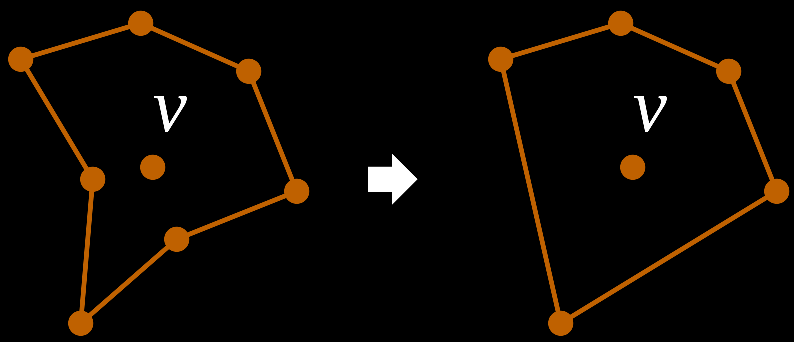
Each Vertex Has a Starting Cone



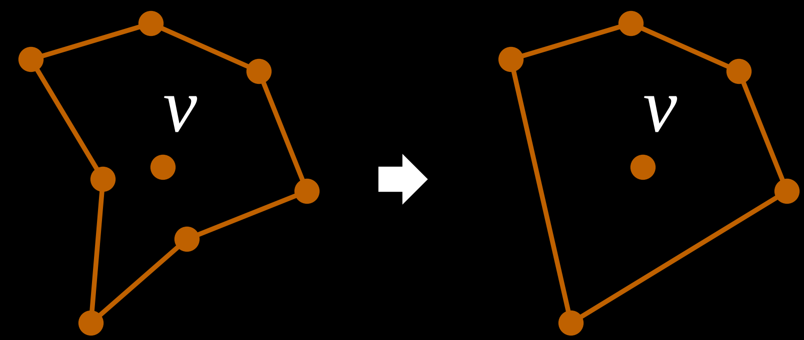


Each Vertex Has a Starting Cone

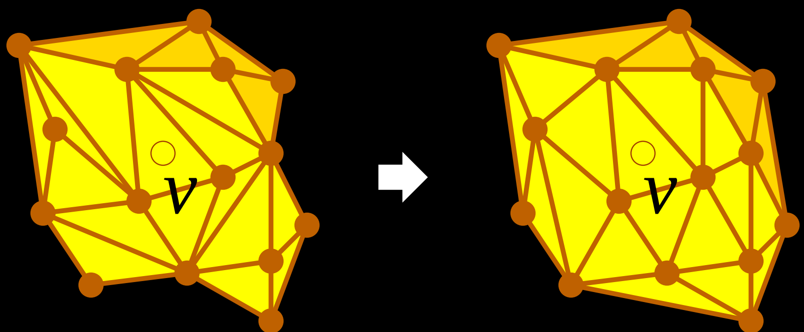




☆ In a 2D triangulation, v's link is a 1D triangulation.



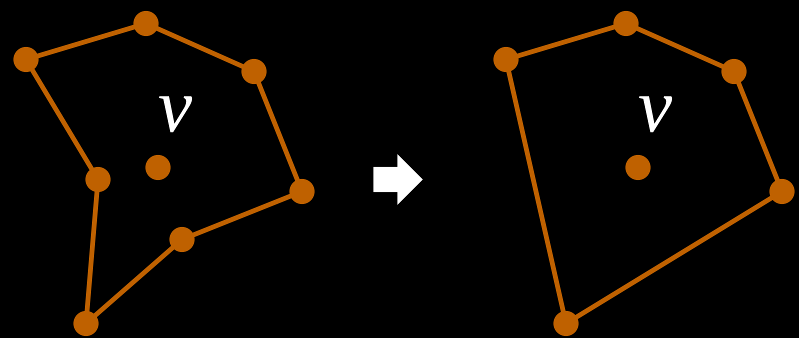
☆ In a 2D triangulation, v's link is a 1D triangulation.



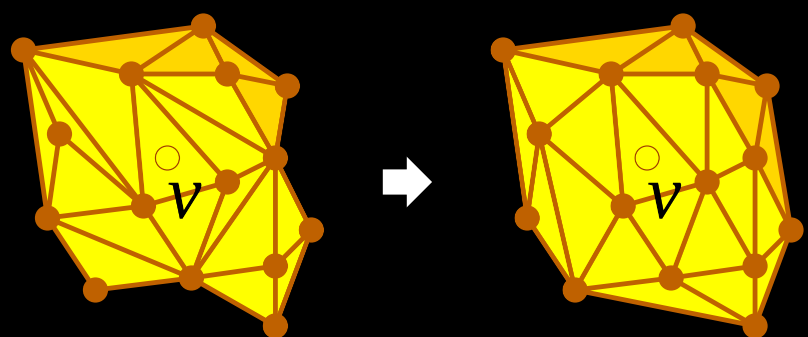
☆ In a 3D triangulation, v's link is a 2D triangulation.

Idea #3

Instead of computing convex cones from scratch, compute them from the starting cones by running star flipping *recursively,* one dimension down.



☆ In a 2D triangulation, v's link is a 1D triangulation.

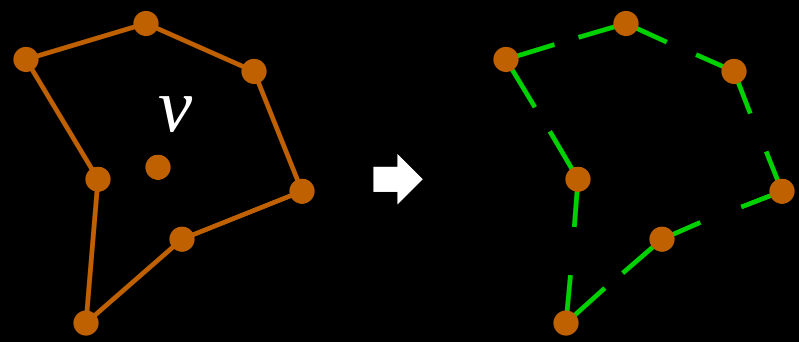


☆ In a 3D triangulation, v's link is a 2D triangulation.

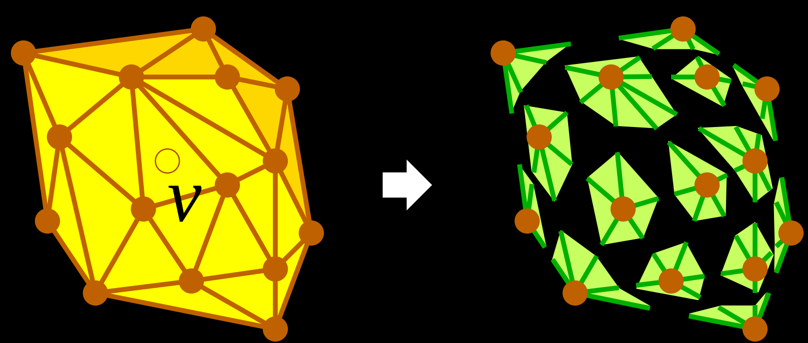
Idea #3

The representation is recursive as well.

Each link triangulation is a collection of stars.



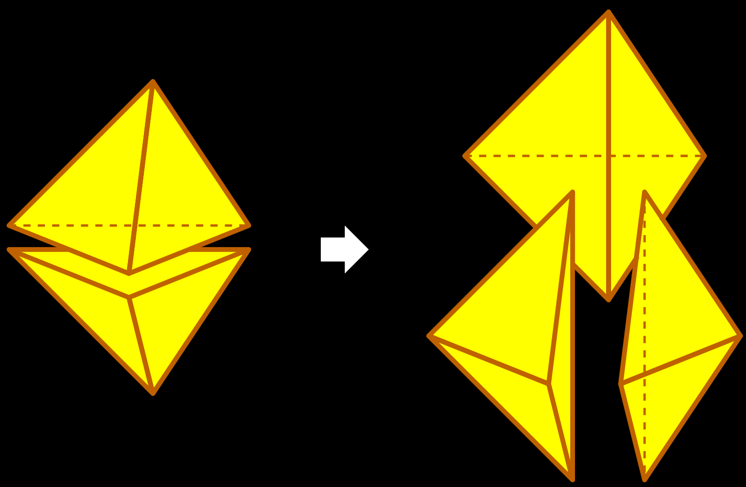
In a 2D triangulation, v’s link is a 1D triangulation.

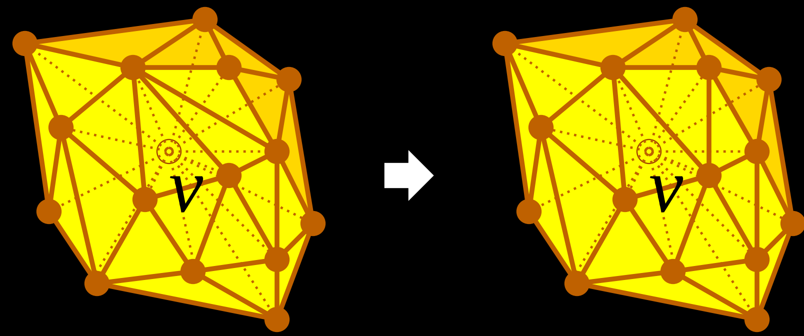


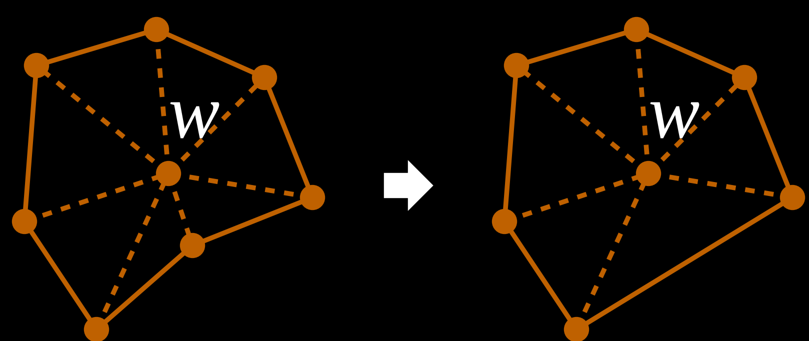
In a 3D triangulation, v’s link is a 2D triangulation.

Idea #4

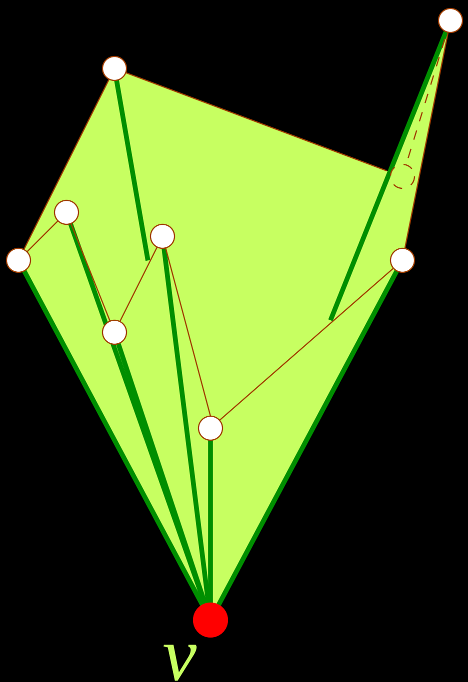
Do as much classic flipping as possible before resorting to star flipping – at every level of the recursion.

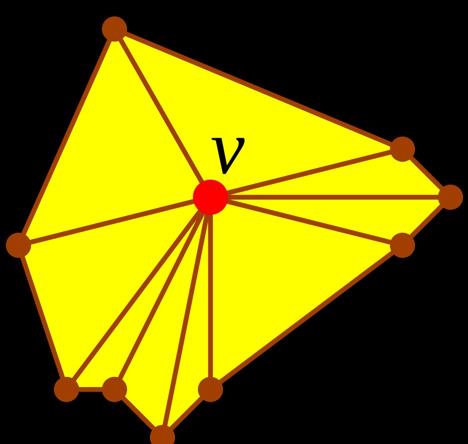




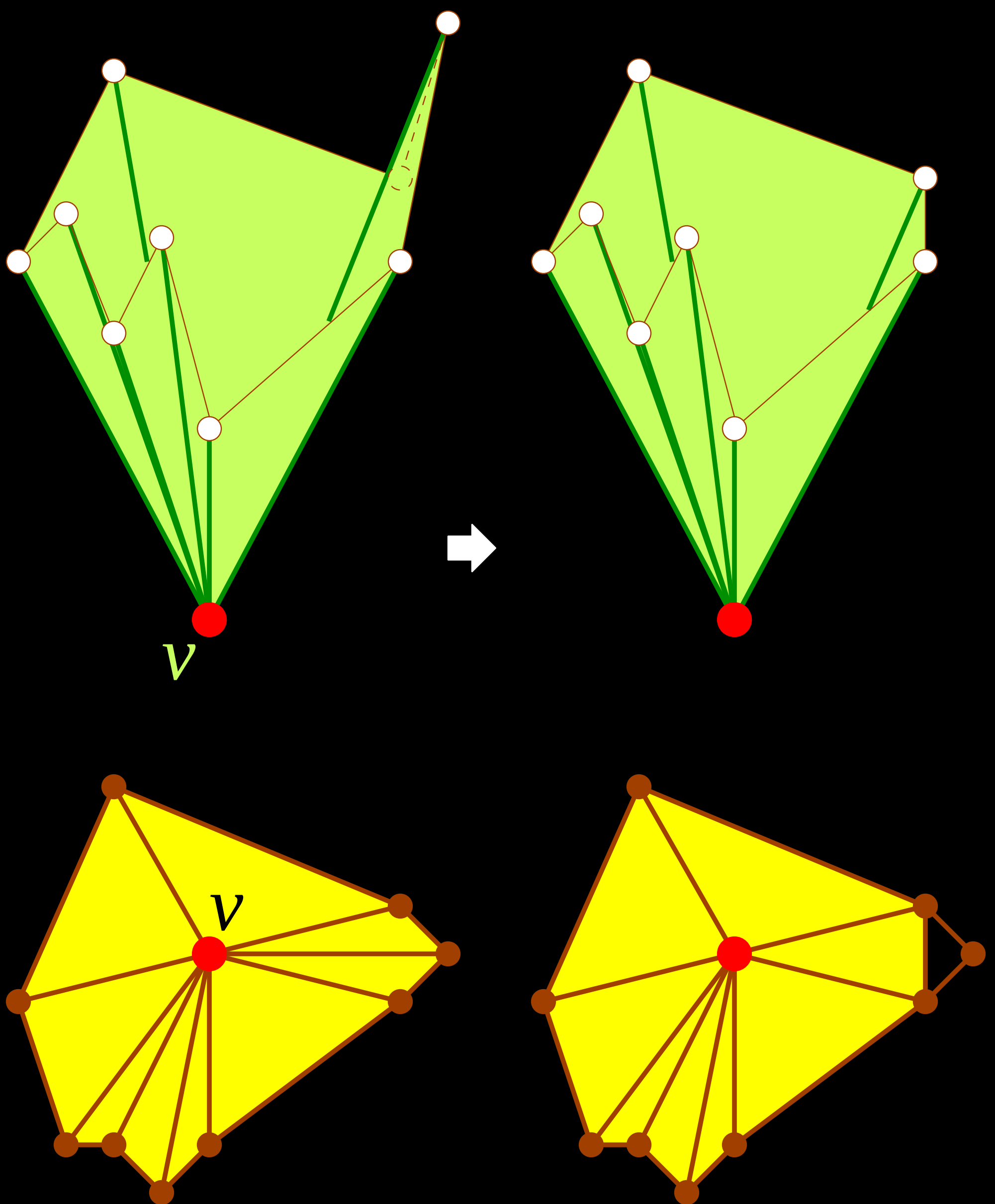


Each Vertex Has a Starting Cone

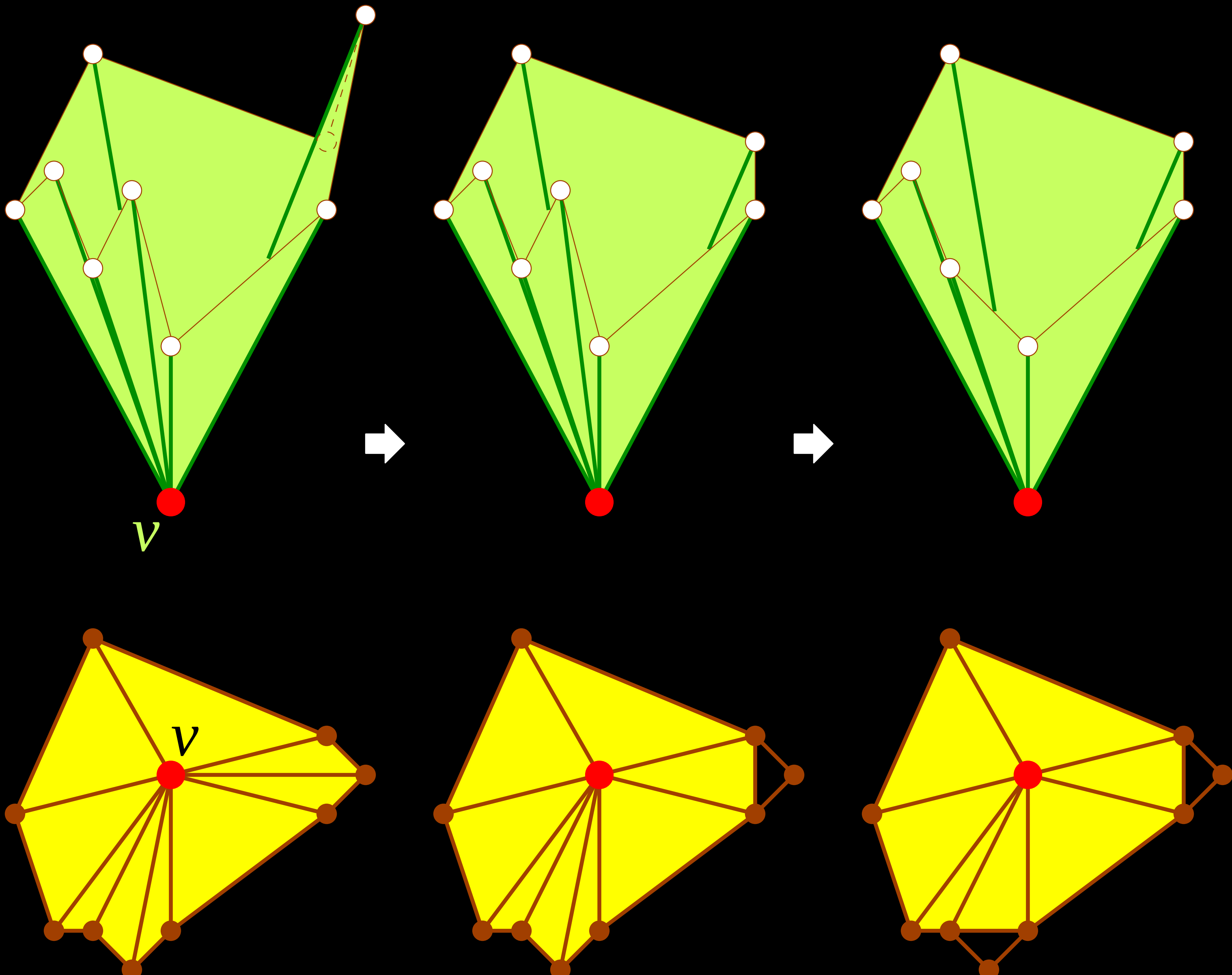




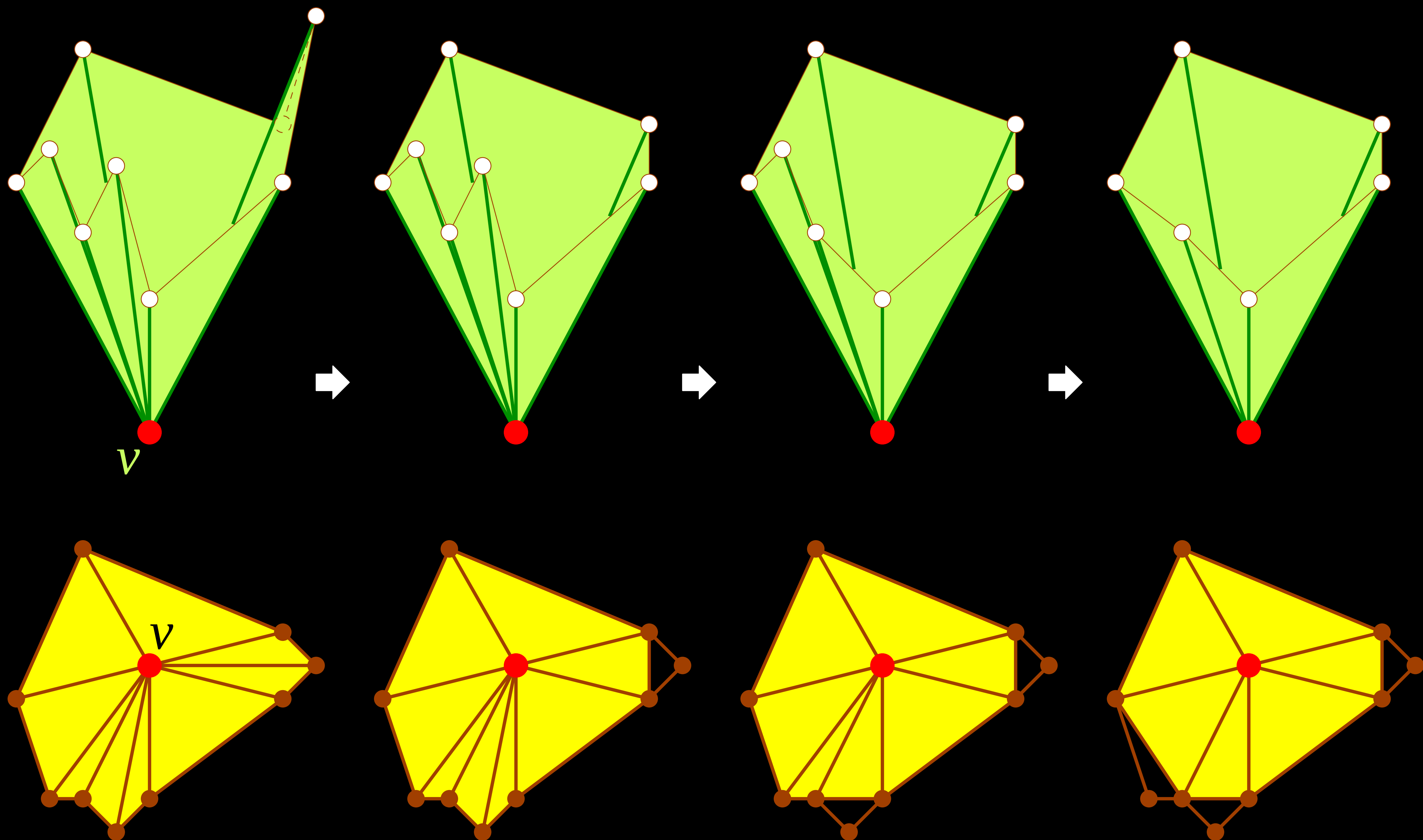
Each Vertex Has a Starting Cone



Each Vertex Has a Starting Cone



Each Vertex Has a Starting Cone



Why Star Flipping?

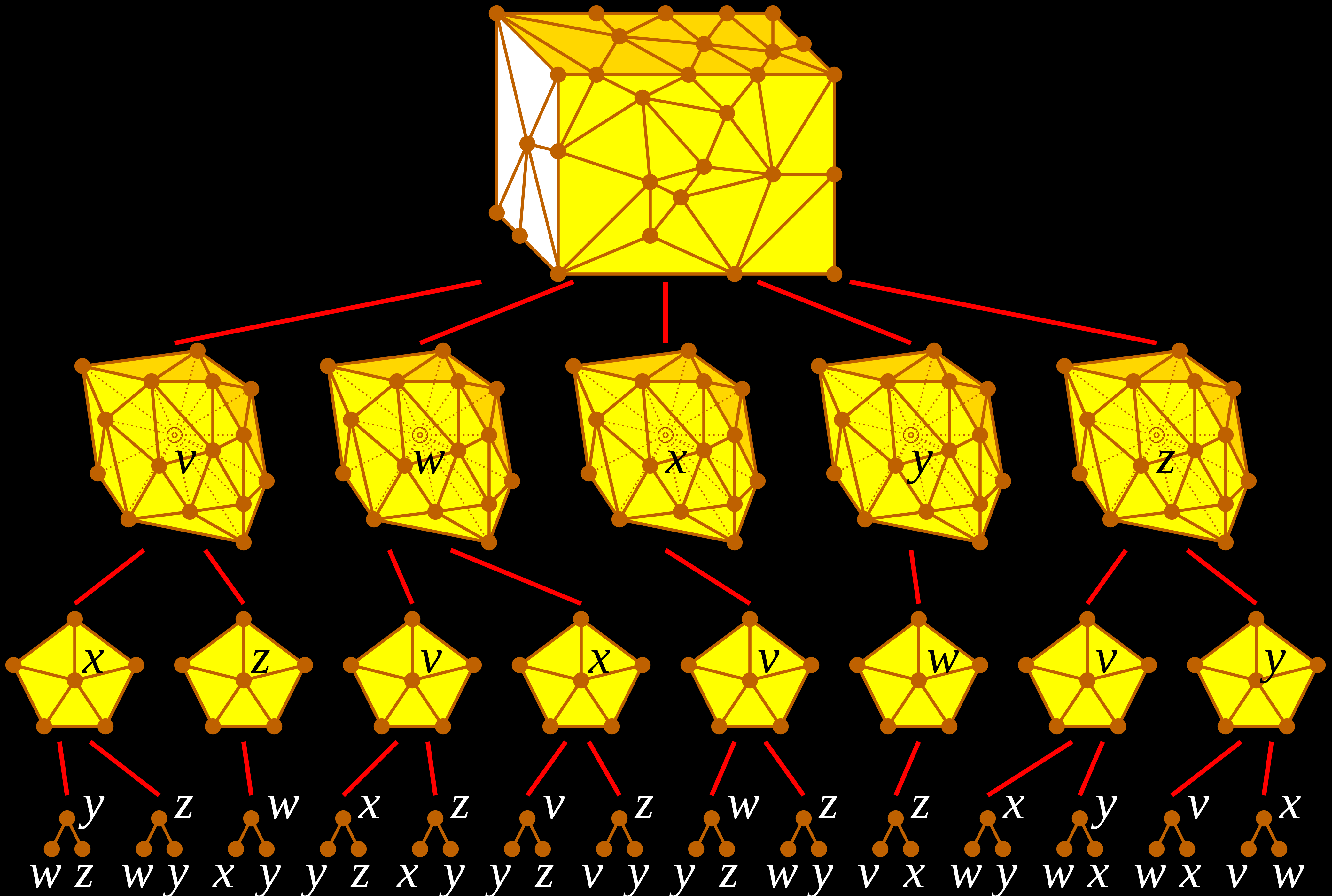
- ✫ Might be faster than star splaying when initial triangulation is “nearly Delaunay.”

Why Star Flipping?

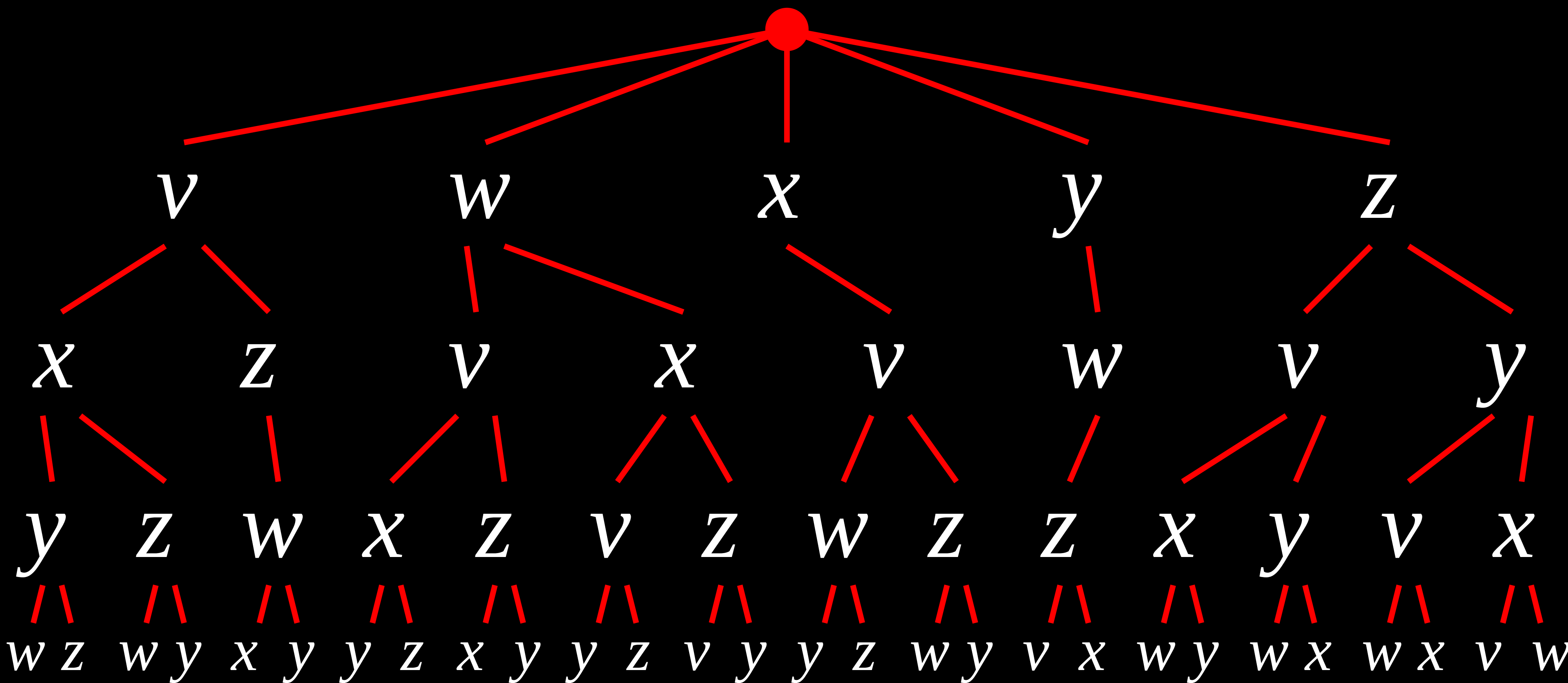
- ✫ Might be faster than star splaying when initial triangulation is “nearly Delaunay.”
- ✫ Can deal with *constrained* triangulations (upcoming).

Tangulations

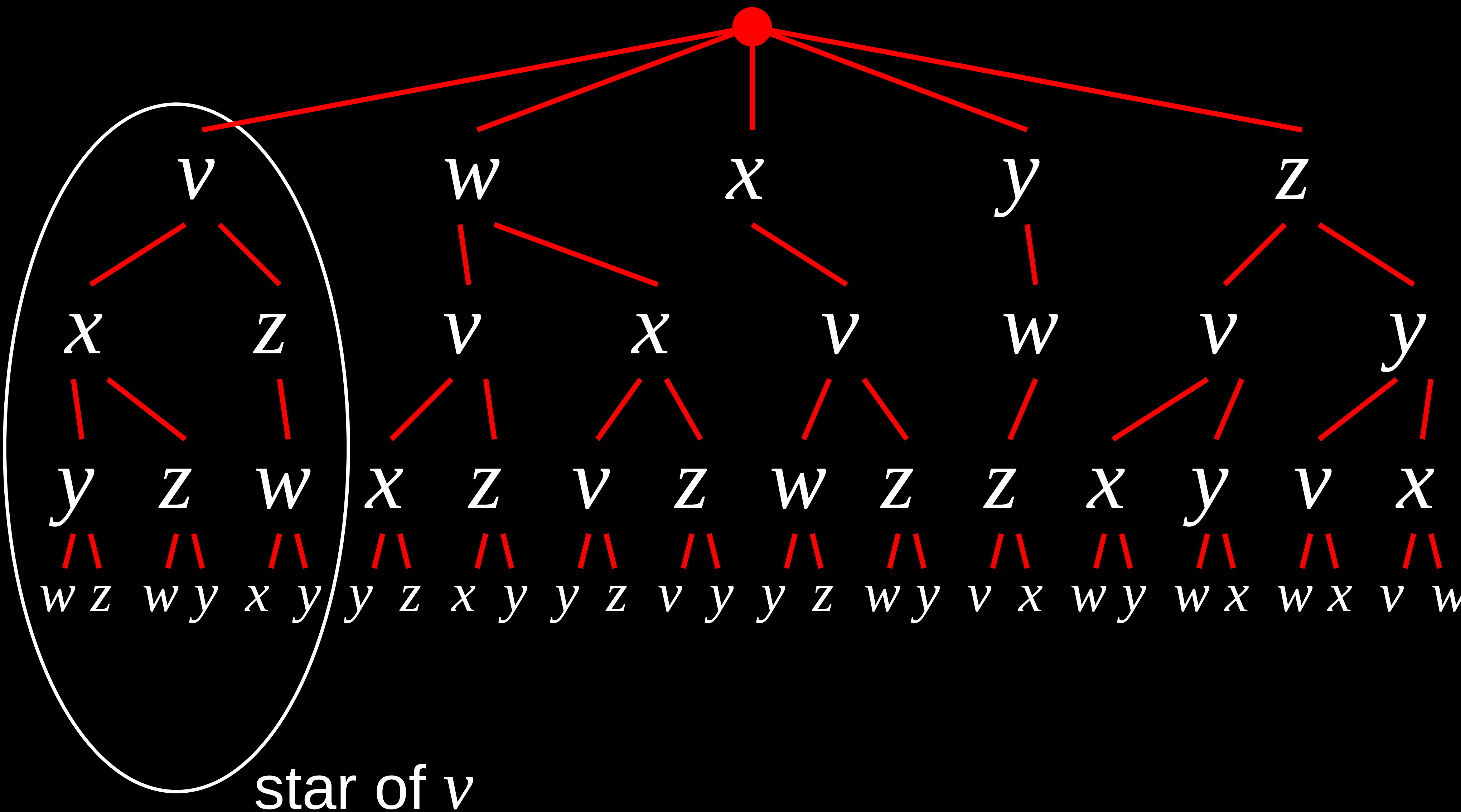
Recursive Representation



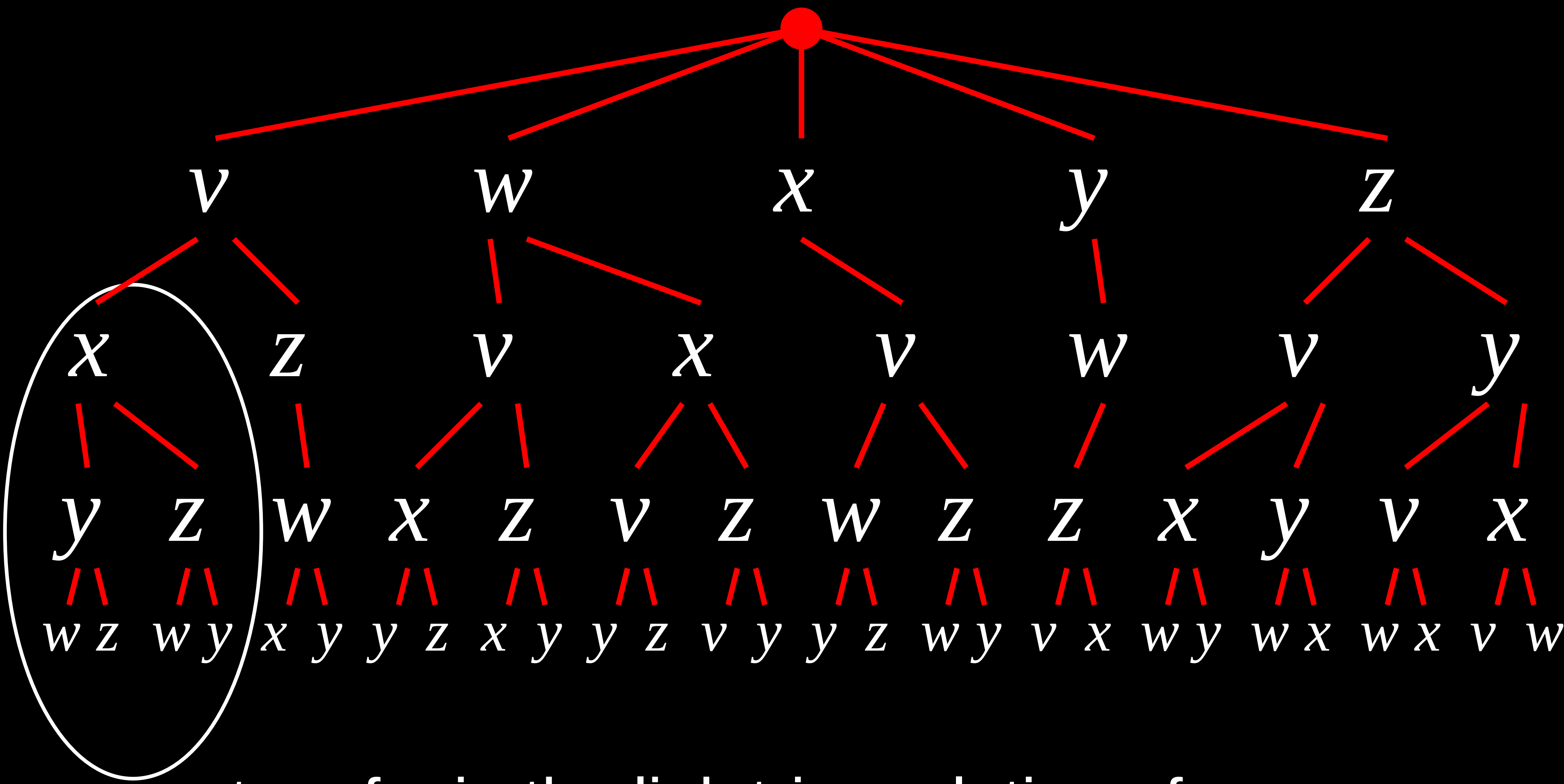
Recursive Representation



Recursive Representation

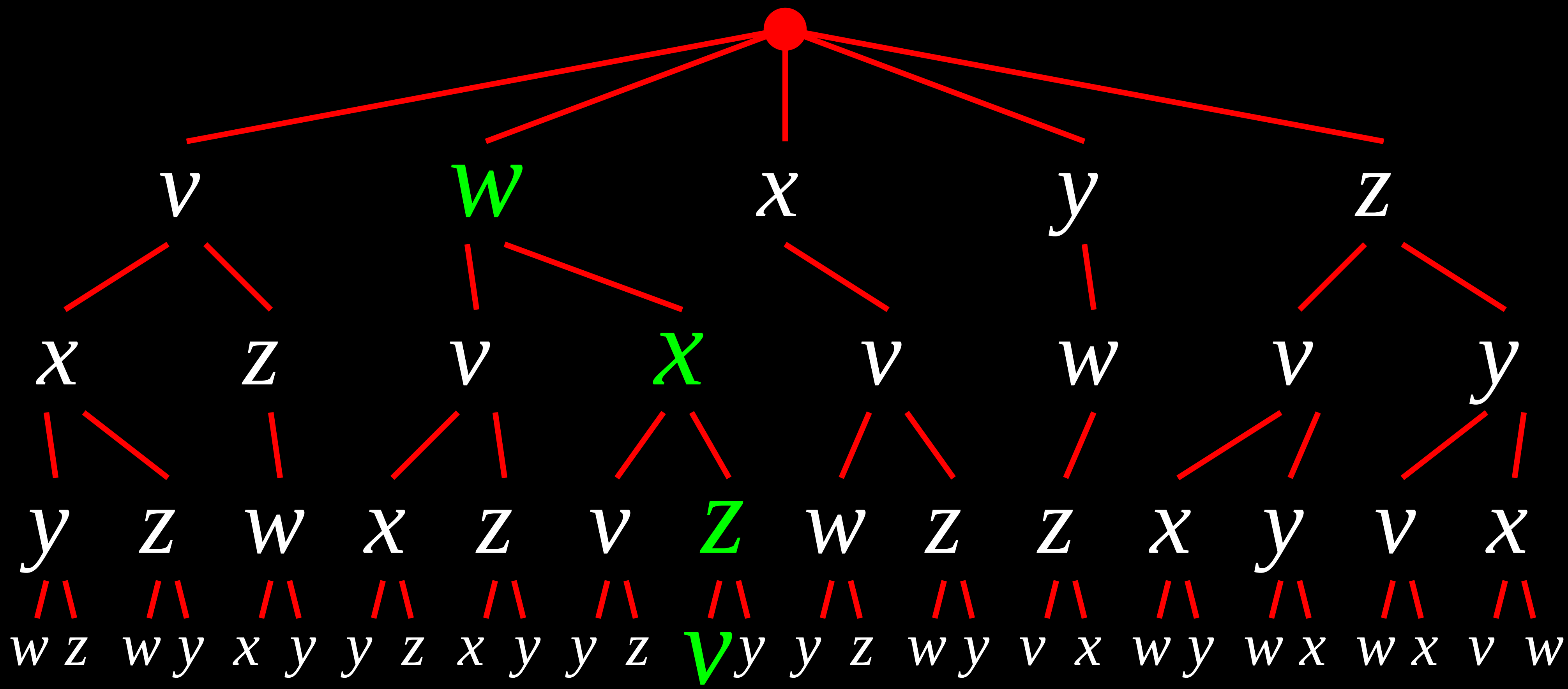


Recursive Representation



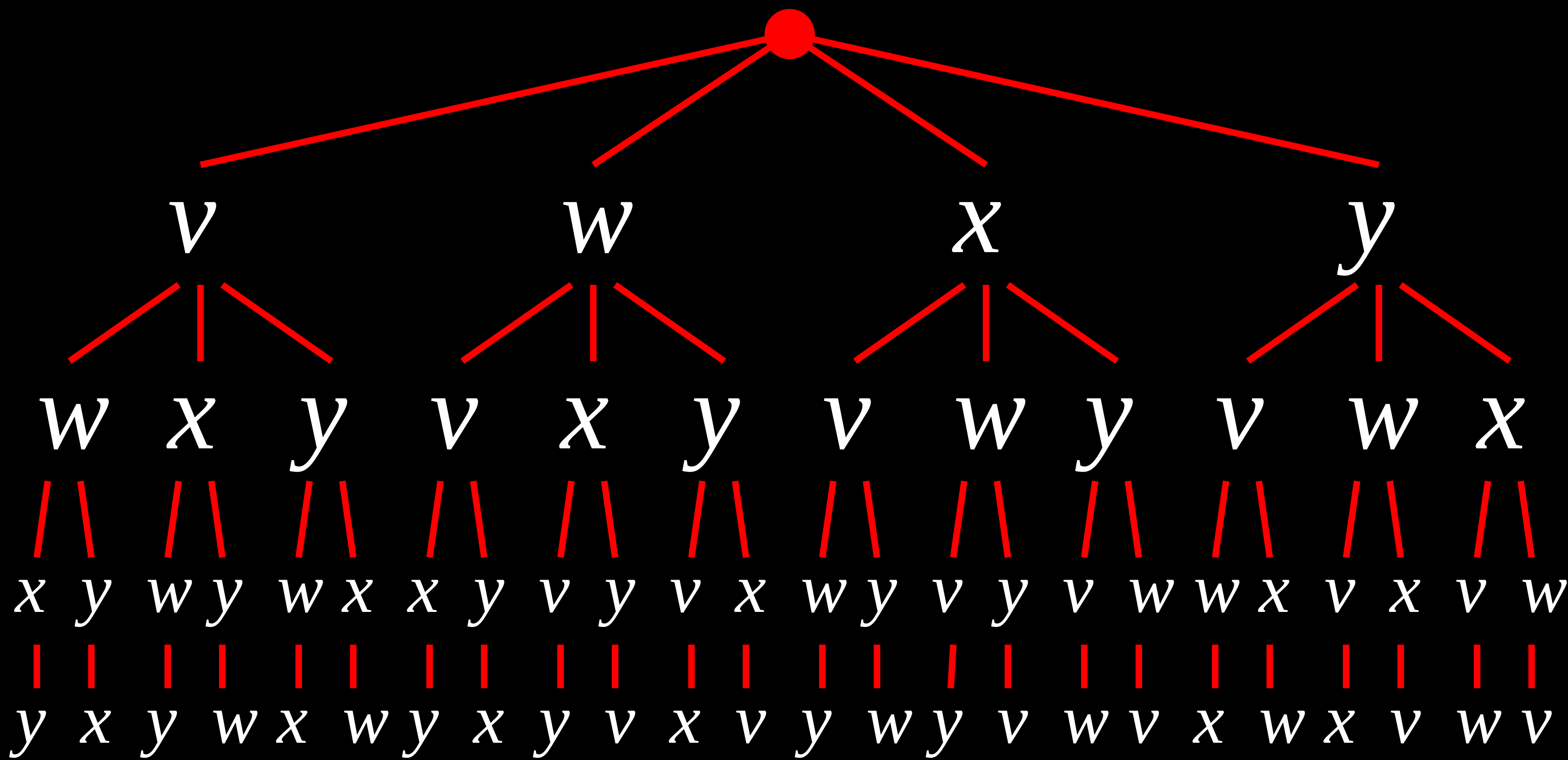
star of x in the link triangulation of v

Recursive Representation (Trie)



ordered tetrahedron: *wxzv*

Recursive Representation (Trie)



- ✫ If all stars are consistent, each simplex appears in every permutation.
- ✫ Unordered tetrahedron *vwxy* is represented by the 24 ordered tetrahedra above.

Tangulation (Definition)

- A set of ordered simplices.

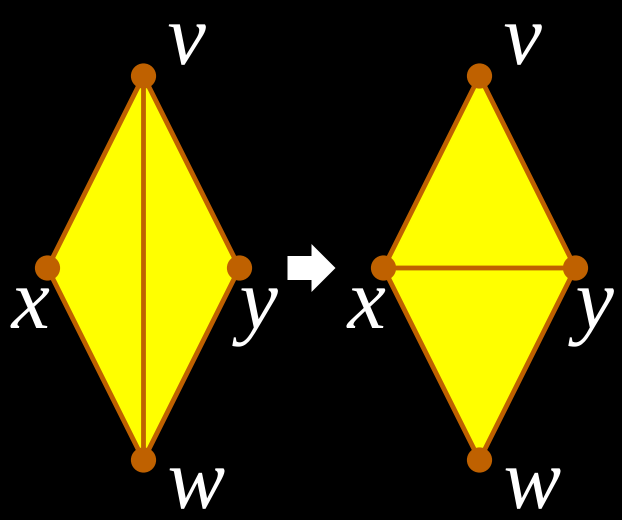
Tangulation (Definition)

- ✫ A set of ordered simplices.

- ✫ If a tangulation contains a simplex, it contains every prefix of that simplex.

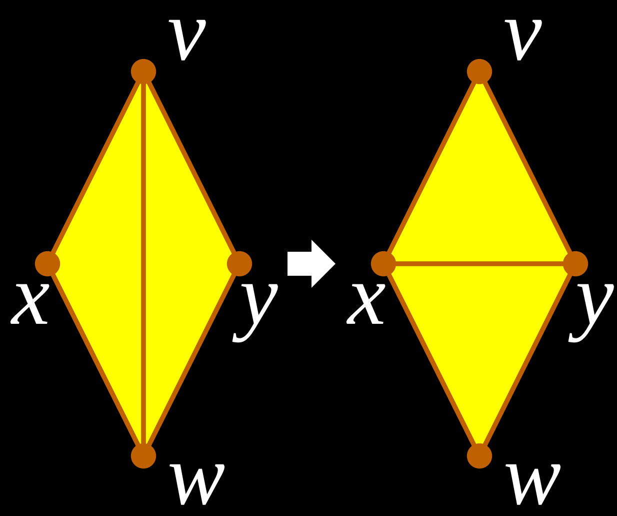
  i.e. if a tangulation contains ordered tetrahedron $vwxy$, it also contains v, vw, vwx.

Flips in Tangulations



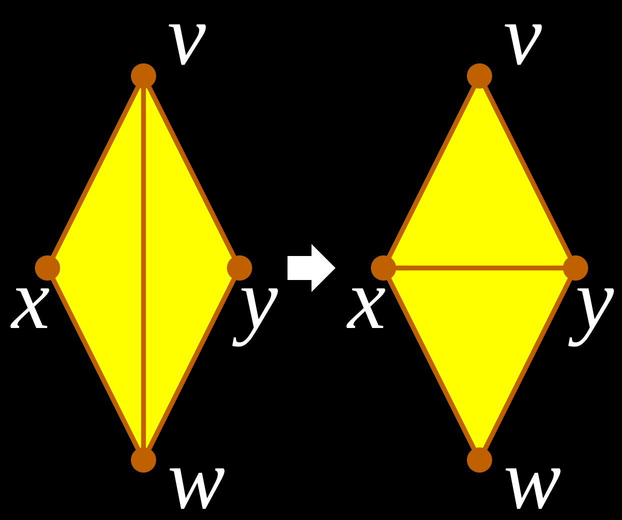
☆ Traditional flip replaces unordered triangles *vwx*, *vwy* with *vxy*, *wxy*.

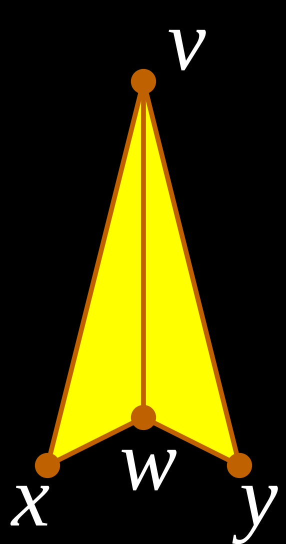
Flips in Tangulations



- ☆ Traditional flip replaces unordered triangles *vwx*, *vwy* with *vxy*, *wxy*.
- ☆ Tangulation flip replaces *vwx*, *vxw*, *xvw*, *xwv*, *wvx*, *wxv*, *vwy*, *vyw*, *wvy*, *wyv*, *yvw*, *ywv* with *vxy*, *vyx*, *xvy*, *xyv*, *yvx*, *yxv*, *wxy*, *wyx*, *xwy*, *xyw*, *ywx*, *yxw*.

Flips in Tangulations

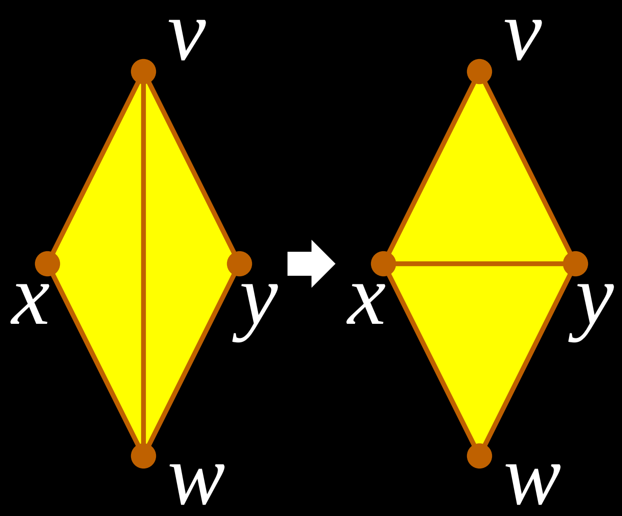


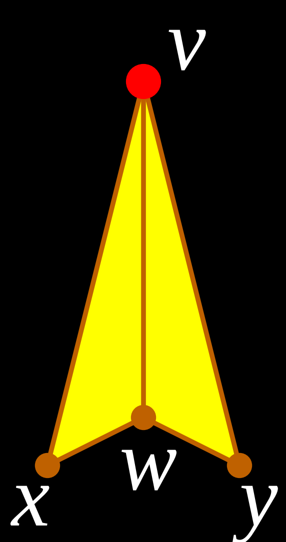


- ☆ Traditional flip replaces unordered triangles vwx, vwy with vxy, wxy.

- ☆ Tangulation flip replaces vwx, vxw, xvw, xwv, wvx, wxv, vwy, vyw, wvy, wyv, yvw, ywv with vxy, vyx, xvy, xyv, yvx, yxv, wxy, wyx, xwy, xyw, ywx, yxw.

Flips in Tangulations

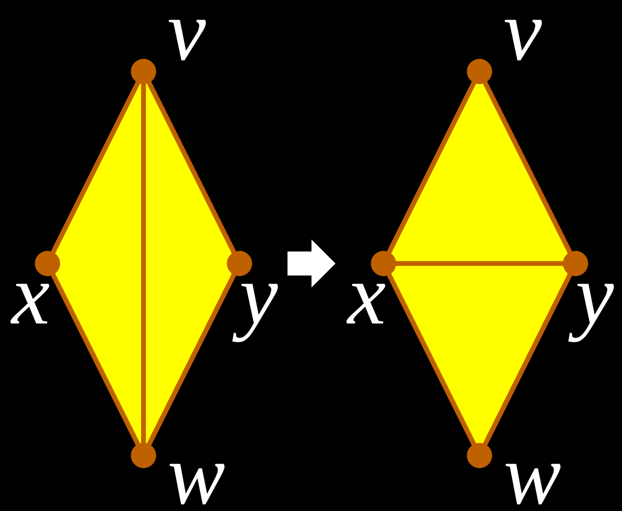




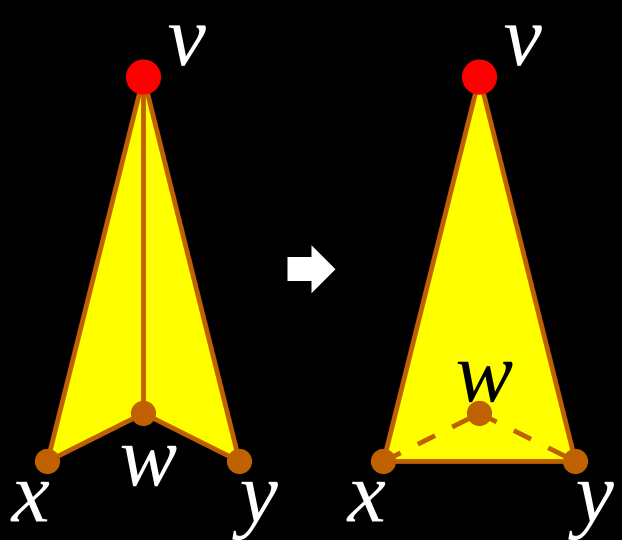
☆ Traditional flip replaces unordered triangles vwx, vwy with vxy, wxy.

☆ Tangulation flip replaces
vwx, vxw, $\cancel{xvw}$, $\cancel{xwv}$, $\cancel{wvx}$, $\cancel{wxv}$,
vwy, vyw, $\cancel{wvy}$, $\cancel{wyv}$, $\cancel{yvw}$, $\cancel{ywv}$
with
vxy, vyx, $\cancel{xvy}$, $\cancel{xyv}$, $\cancel{yvx}$, $\cancel{yxv}$,
$\cancel{wxy}$, $\cancel{wyx}$, $\cancel{xwy}$, $\cancel{xyw}$, $\cancel{ywx}$, $\cancel{yxw}$.

Flips in Tangulations



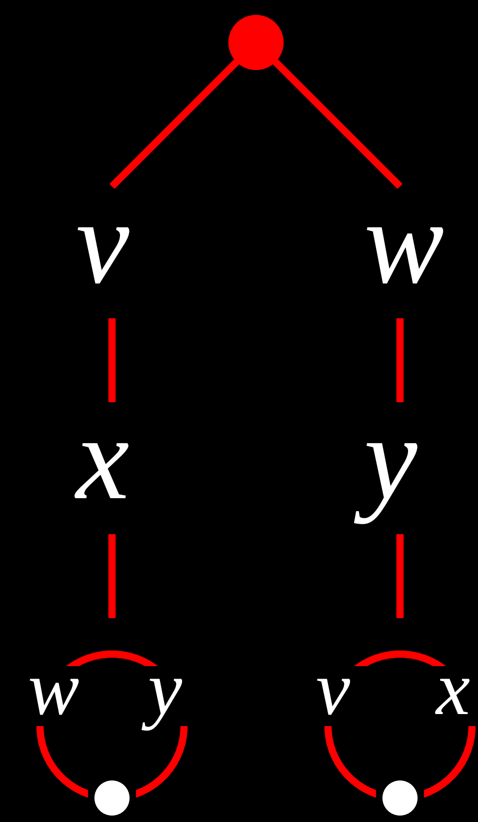
☆ Traditional flip replaces unordered triangles vwx, vwy with vxy, wxy.



☆ Flip in v's star replaces ordered triangles vwx, vxw, vwy, vyw with vxy, vyx.

Tangulation Data Structure

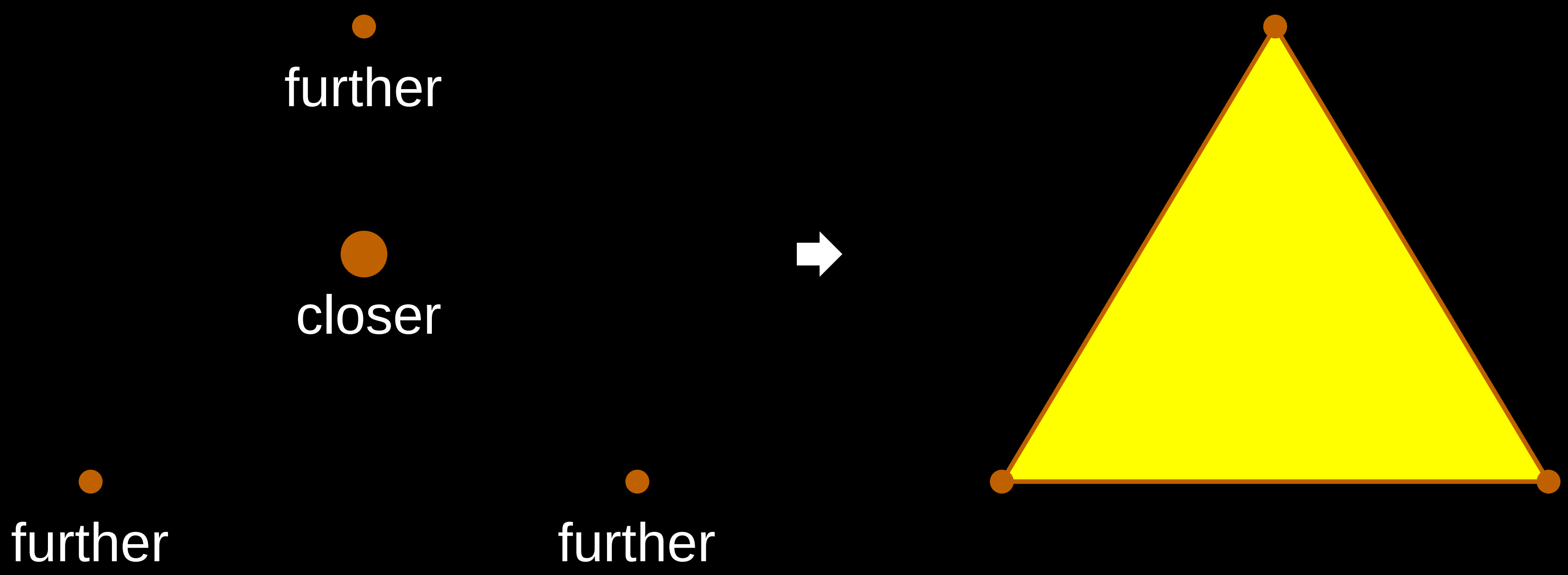
- ✫ Storage optimizations take advantage of symmetry.
- ✫ Proposed data structure reduces to Blandford, Blelloch, Cardoze, and Kadow [IMR 2003] when all stars are mutually & internally consistent.



Unordered tetrahedron *vwxy*

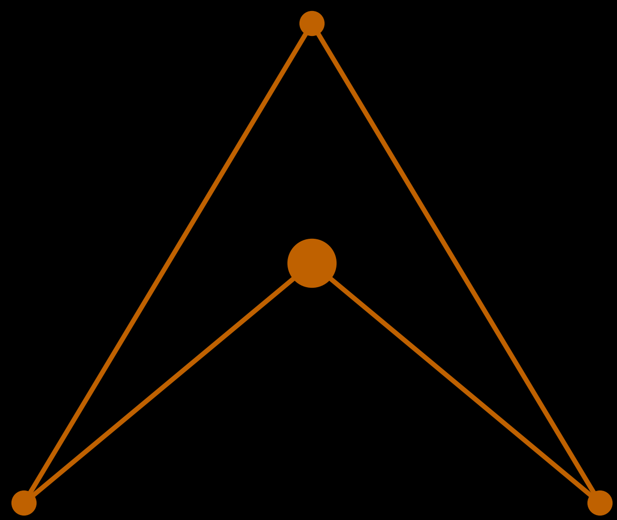
Constrained Tangulations

Weighted Delaunay Triangulation



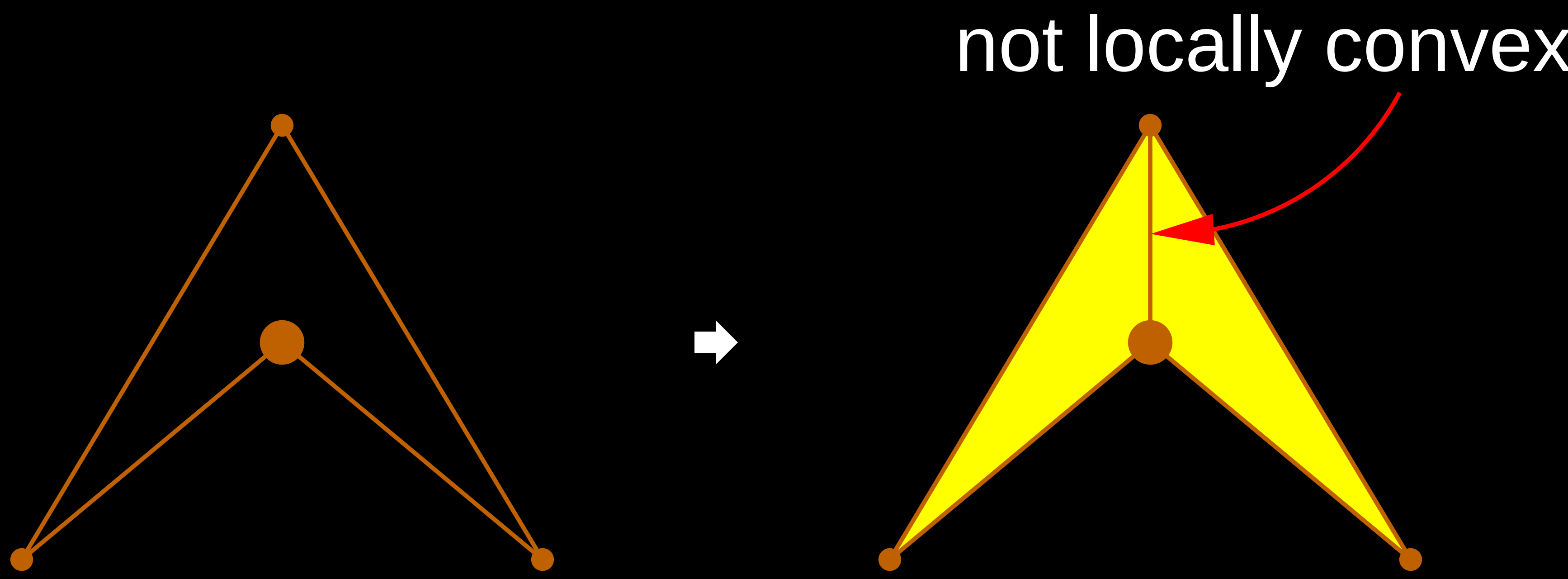
Lower convex hull is a triangle.

Weighted CDT (constrained Delaunay triangulation)

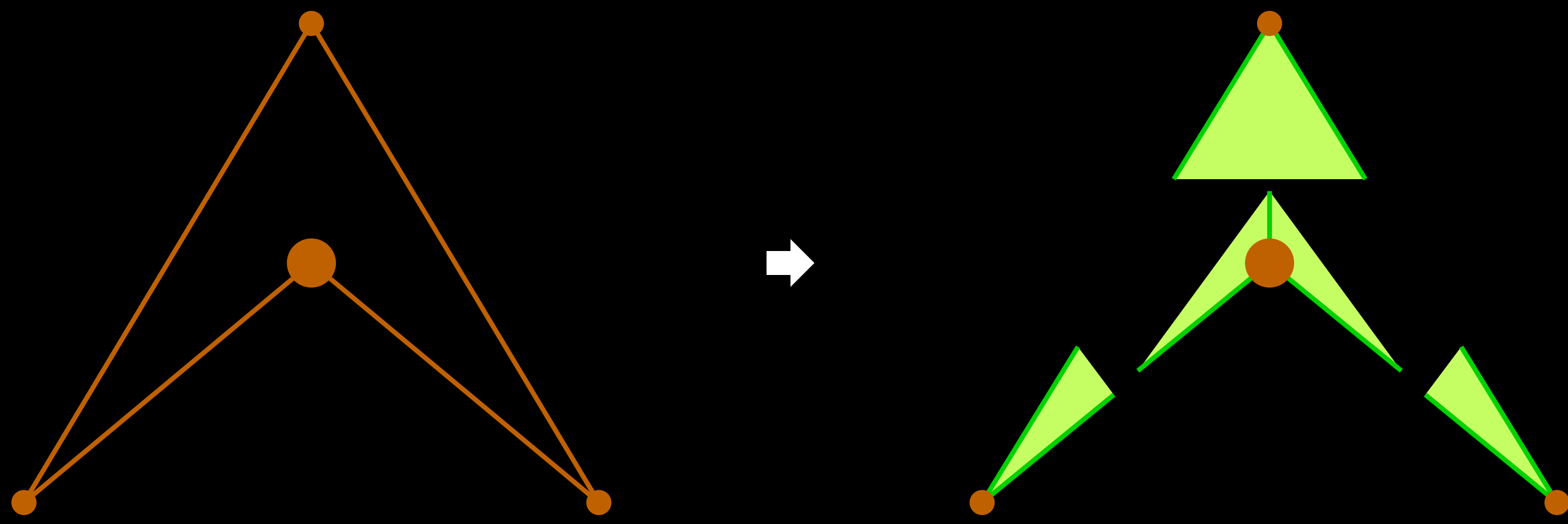


➡ nonexistent

Weighted CDT (constrained Delaunay triangulation)



Constrained Tangulations



Every (weighted) domain has a constrained "Delaunay" tangulation.

Incremental CDT Update