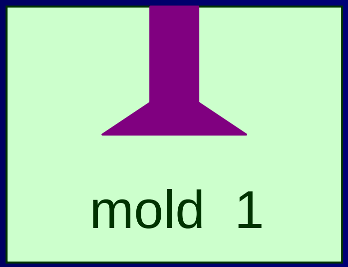
Outline

I. The molding problem

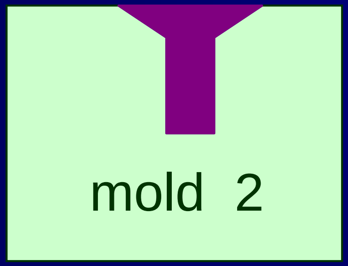
II. Problem transformation

III. Intersection of half-planes

I. The Problem of Molding

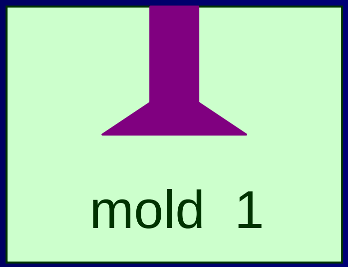


Does a given object have a mold from which it can be removed?

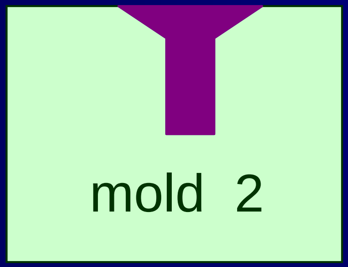


I. The Problem of Molding

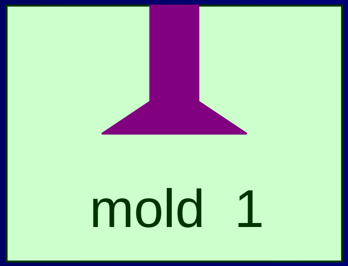
Does a given object have a mold from which it can be removed?



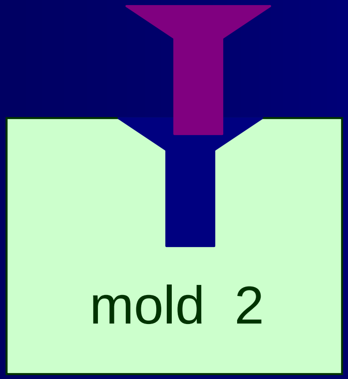
object not removable



I. The Problem of Molding



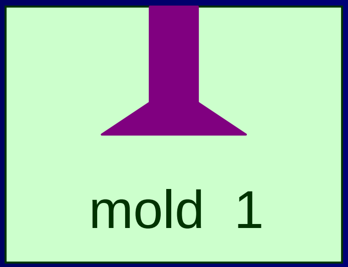
object not
removable



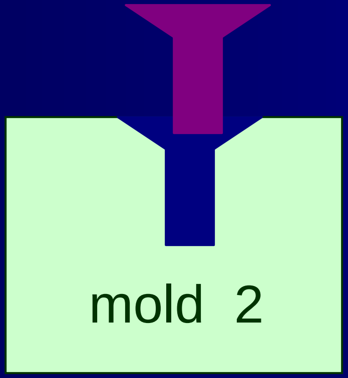
object
removable

Does a given object have a mold from which it can be removed?

I. The Problem of Molding



object not
removable



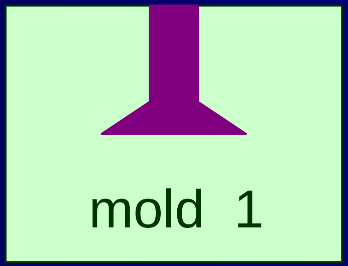
object
removable

Does a given object have a mold from which it can be removed?

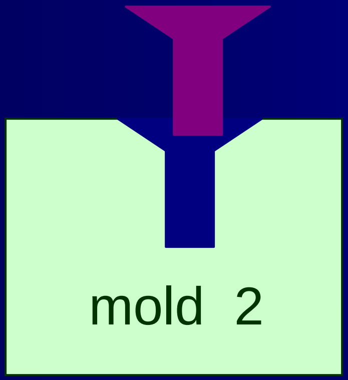
Assumptions

- The object is polyhedral.

I. The Problem of Molding



object not removable



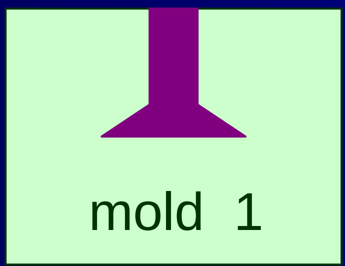
object removable

Does a given object have a mold from which it can be removed?

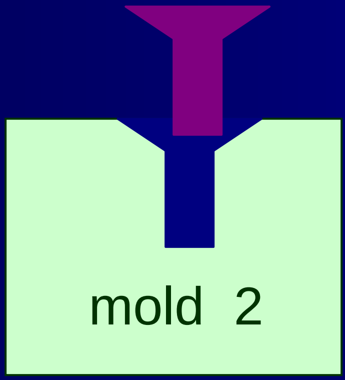
Assumptions

- The object is polyhedral.
- The mold has only one piece.

I. The Problem of Molding



object not removable



object removable

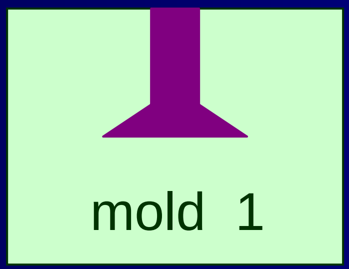
Does a given object have a mold from which it can be removed?

Assumptions

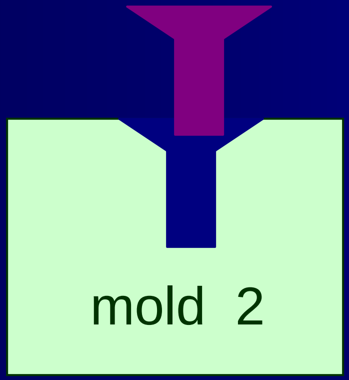
- The object is polyhedral.
- The mold has only one piece.

Spherical objects cannot be manufactured using a mold of one piece.

I. The Problem of Molding



object not removable



object removable

Does a given object have a mold from which it can be removed?

Assumptions

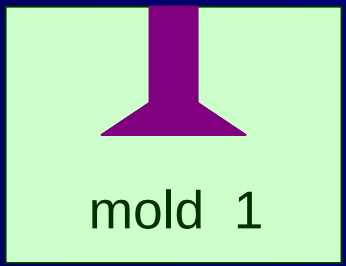
- The object is polyhedral.
- The mold has only one piece.

  Spherical objects cannot be manufactured using a mold of one piece.

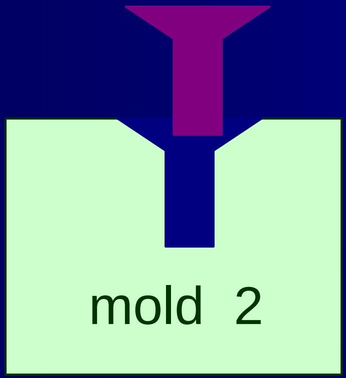
- The object should be removed by only a single translation.

I. The Problem of Molding

Does a given object have a mold from which it can be removed?



object not removable



object removable

Assumptions

- The object is polyhedral.
- The mold has only one piece.

  Spherical objects cannot be manufactured using a mold of one piece.

- The object should be removed by only a single translation.

  Real screws cannot be removed by just a translation.

Castability

How to choose the orientation?

Castability

How to choose the orientation?

- The object has a horizontal *top facet* – the only one not in contact with the mold.

Castability

How to choose the orientation?

- The object has a horizontal *top facet* – the only one not in contact with the mold.
- # possible orientations = # facets.

Castability

How to choose the orientation?

- The object has a horizontal *top facet* – the only one not in contact with the mold.
- # possible orientations = # facets.

  Because every facet may become horizontal.

Castability

How to choose the orientation?

- The object has a horizontal *top facet* – the only one not in contact with the mold.

- # possible orientations = # facets.

  Because every facet may become horizontal.

An object is *castable* if it is removable from its mold for one of the orientations.

Castability

How to choose the orientation?

- The object has a horizontal *top facet* – the only one not in contact with the mold.

- # possible orientations = # facets.

  Because every facet may become horizontal.

An object is *castable* if it is removable from its mold for one of the orientations.

How to determine that the object is castable?

Castability

How to choose the orientation?

- The object has a horizontal *top facet* – the only one not in contact with the mold.
- # possible orientations = # facets.

  Because every facet may become horizontal.

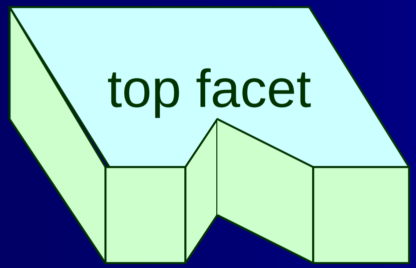
An object is *castable* if it is removable from its mold for one of the orientations.

How to determine that the object is castable?

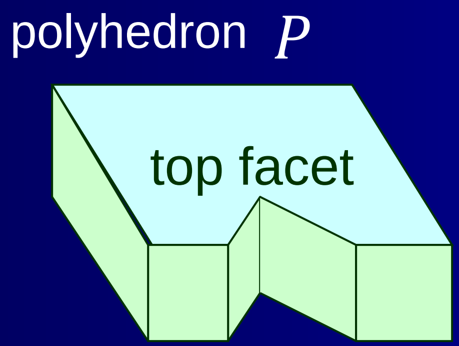
For each potential orientation, determine whether there exists a direction along which the object can be removed from the mold.

Making Things More Precise

polyhedron P

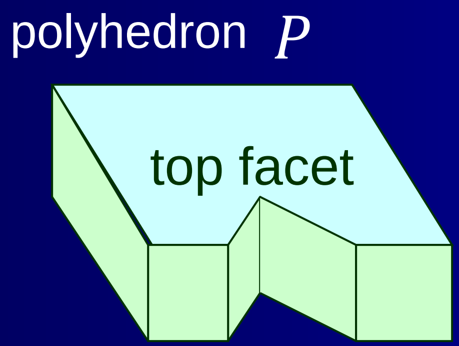


Making Things More Precise



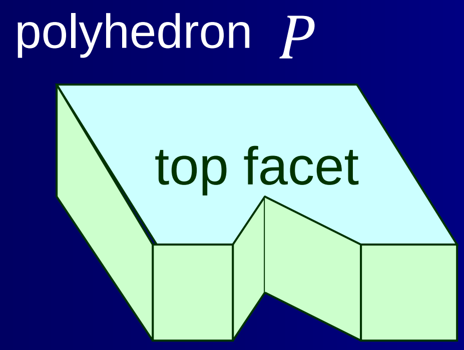
- **The mold is a rectangular block with a concavity that exactly matches the polyhedron.**

Making Things More Precise



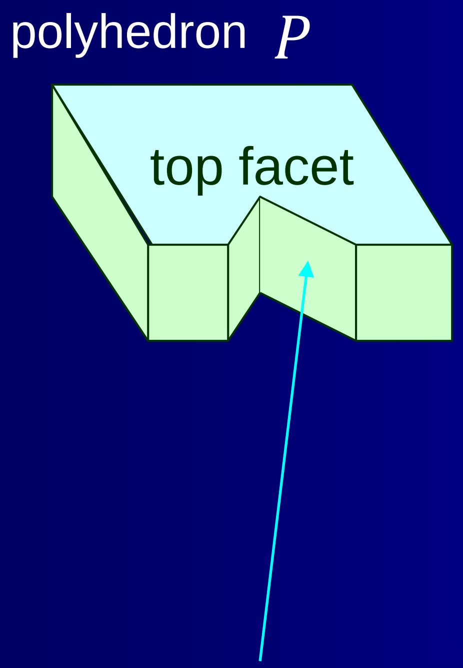
- **The mold is a rectangular block with a concavity that exactly matches the polyhedron.**
- **Its topmost facet is *horizontal* and chosen to be xy-plane.**

Making Things More Precise



- **The mold is a rectangular block with a concavity that exactly matches the polyhedron.**
- **Its topmost facet is *horizontal* and chosen to be xy-plane.**
- **Top facet of the polyhedron is coplanar with the xy-plane.**

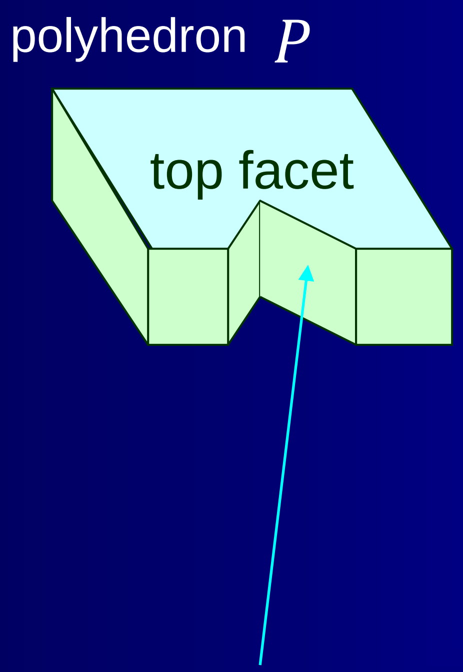
Making Things More Precise



- **The mold is a rectangular block with a concavity that exactly matches the polyhedron.**
- **Its topmost facet is *horizontal* and chosen to be xy-plane.**
- **Top facet of the polyhedron is coplanar with the xy-plane.**

Ordinary facet f: a facet of the polyhedron that is not the top.

Making Things More Precise

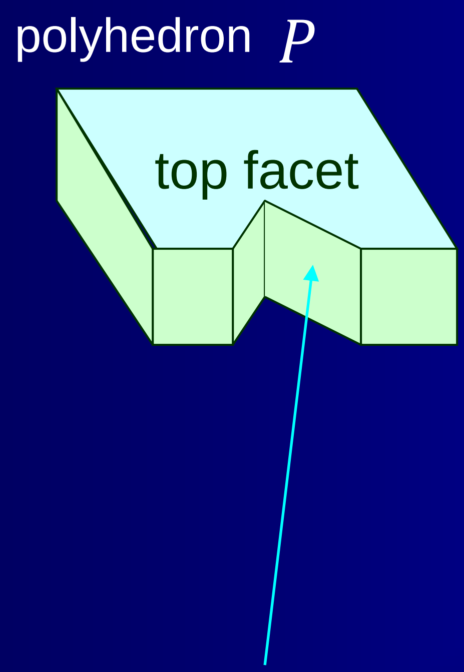


- The mold is a rectangular block with a concavity that exactly matches the polyhedron.
- Its topmost facet is *horizontal* and chosen to be xy-plane.
- Top facet of the polyhedron is coplanar with the xy-plane.

Ordinary facet f: a facet of the polyhedron that is not the top.

F : the facet in the mold that corresponds to f.

Making Things More Precise



- **The mold is a rectangular block with a concavity that exactly matches the polyhedron.**
- **Its topmost facet is *horizontal* and chosen to be xy-plane.**
- **Top facet of the polyhedron is coplanar with the xy-plane.**

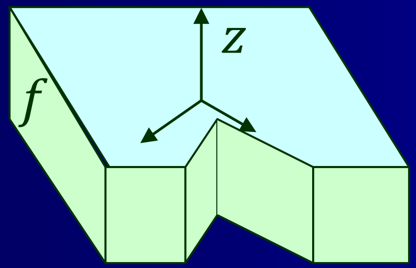
Ordinary facet f: a facet of the polyhedron that is not the top.

F : the facet in the mold that corresponds to f.

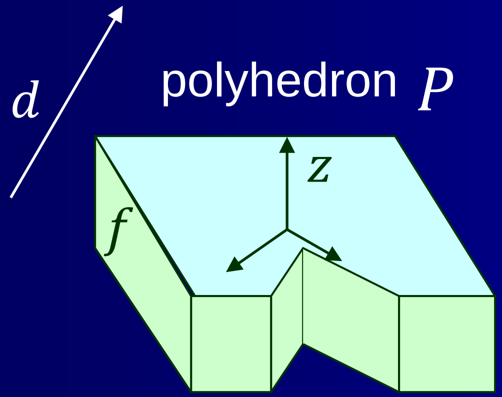
Problem **Decide whether a direction d exists such that P can be translated to infinity without colliding with the mold.**

Necessary Condition for Removal

polyhedron P

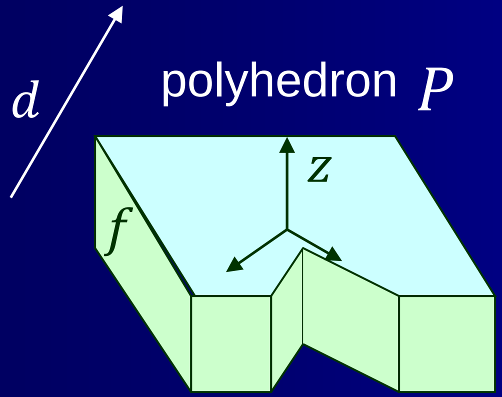


Necessary Condition for Removal



d has a positive z component.

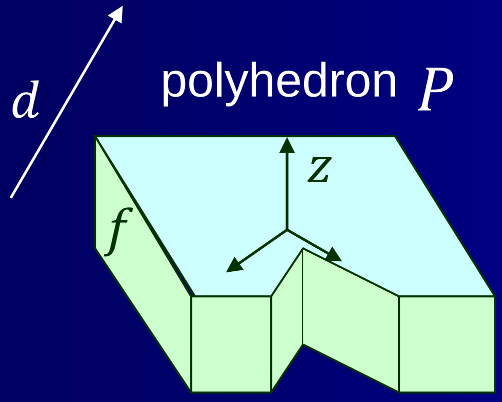
Necessary Condition for Removal



d has a positive *z* component.

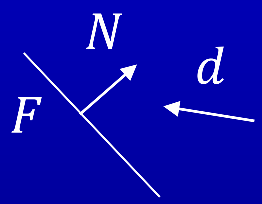
The translation of a facet *f* cannot penetrate into the corresponding facet *F* of the mold.

Necessary Condition for Removal



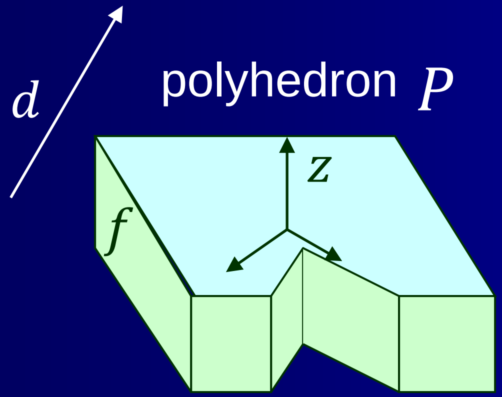
d has a positive z component.

The translation of a facet f cannot penetrate into the corresponding facet F of the mold.



F blocks the translation if d forms an angle $> \pi/2$ with its outward normal N.

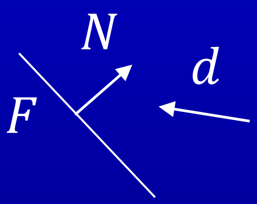
Necessary Condition for Removal

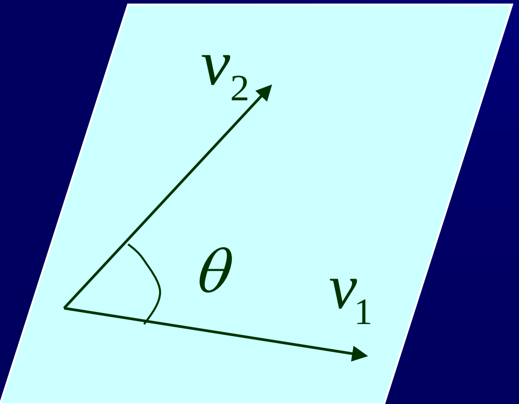


d has a positive z component.

The translation of a facet f cannot penetrate into the corresponding facet F of the mold.

⇨ F blocks the translation if d forms an angle $> \pi/2$ with its outward normal N.



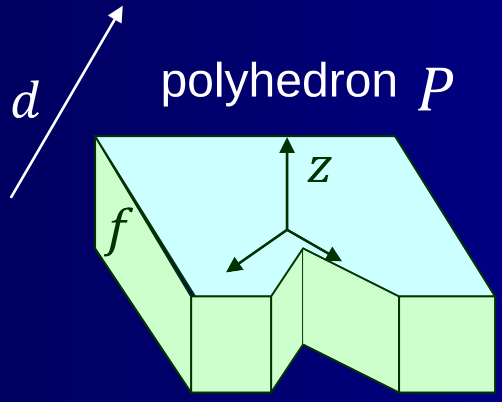


angle between two vectors

$$\cos\theta = v_1 \bullet v_2$$

θ chosen to be in $[0, \pi]$.

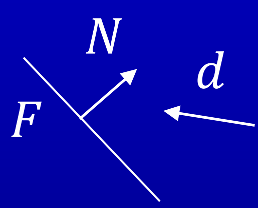
Necessary Condition for Removal



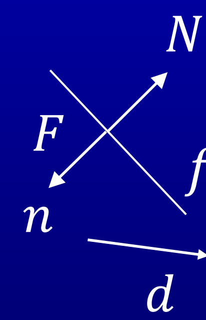
d has a positive z component.

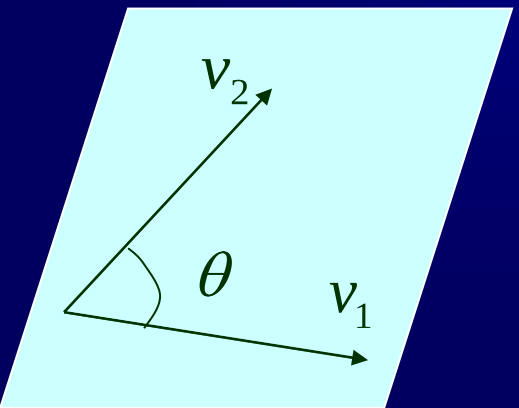
The translation of a facet f cannot penetrate into the corresponding facet F of the mold.

F blocks the translation if d forms an angle $> \pi/2$ with its outward normal N.



F does not block the translation if d forms an angle $\leq \pi/2$ with its outward normal N.



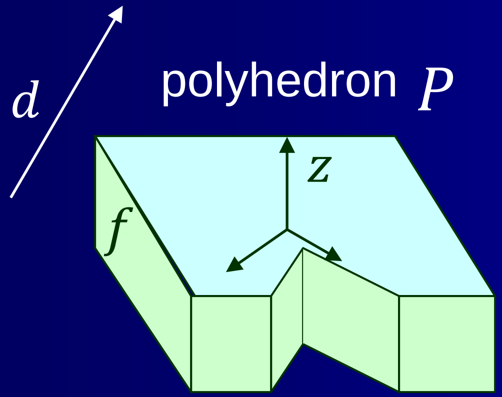


angle between two vectors

$$\cos\theta = v_1 \bullet v_2$$

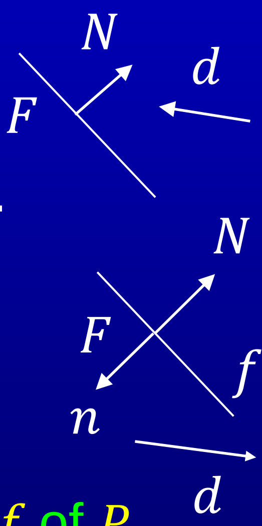
θ chosen to be in $[0, \pi]$.

Necessary Condition for Removal



d has a positive z component.

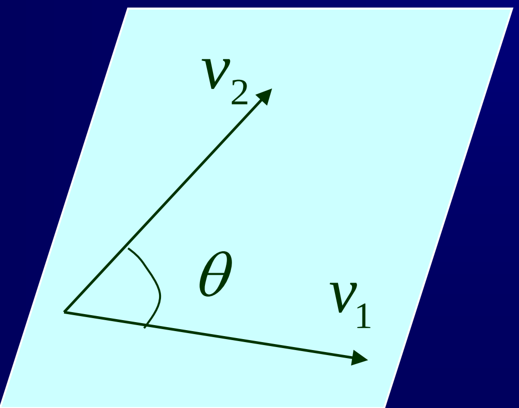
The translation of a facet f cannot penetrate into the corresponding facet F of the mold.



⇒ F blocks the translation if d forms an angle $> \pi/2$ with its outward normal N.

F does not block the translation if d forms an angle $\leq \pi/2$ with its outward normal N.

⇒ d must make an angle $\geq \pi/2$ with the outward normal $n = -N$ of every facet f of P.

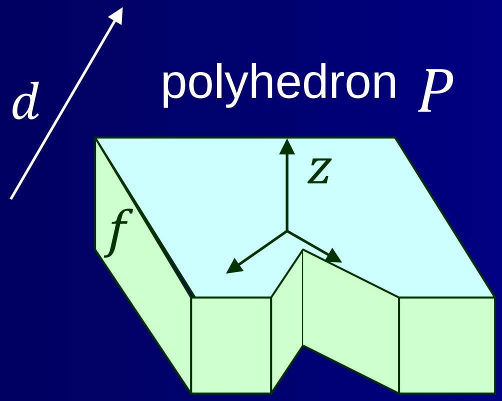


angle between two vectors

$$\cos\theta = v_1 \bullet v_2$$

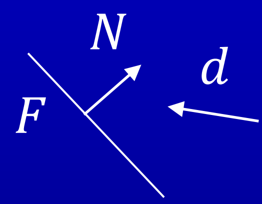
θ chosen to be in $[0, \pi]$.

Necessary Condition for Removal



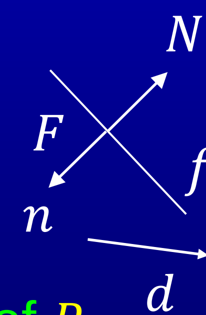
d has a positive z component.

The translation of a facet f cannot penetrate into the corresponding facet F of the mold.



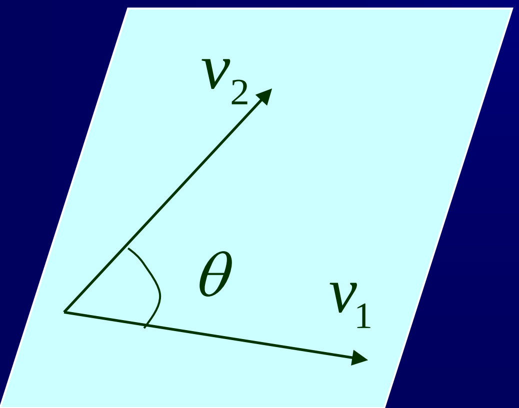
➡ F blocks the translation if d forms an angle $> \pi/2$ with its outward normal N.

F does not block the translation if d forms an angle $\leq \pi/2$ with its outward normal N.



➡ d must make an angle $\geq \pi/2$ with the outward normal $n = -N$ of every facet f of P.

(necessary condition)



angle between two vectors

$$\cos\theta = v_1 \bullet v_2$$

θ chosen to be in $[0, \pi]$.

Also a Sufficient Condition

Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Also a Sufficient Condition

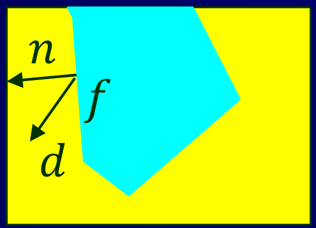
Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.

Also a Sufficient Condition

Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.



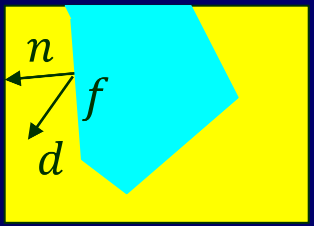
Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

Also a Sufficient Condition

Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.

Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

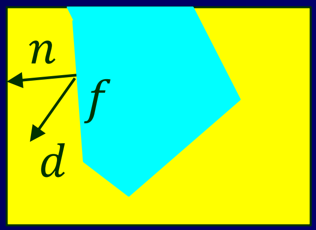


Then any interior point of f collides with the mold in the translation.

Also a Sufficient Condition

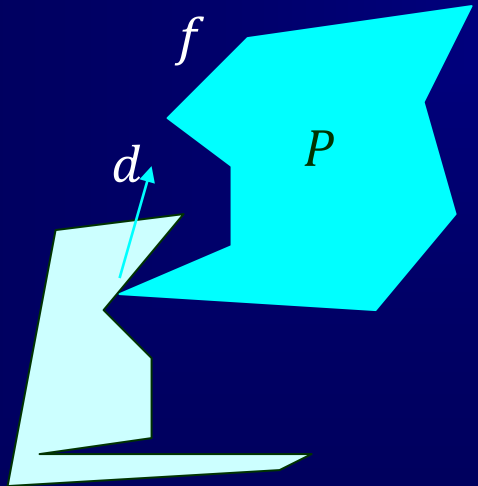
Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.



Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

Then any interior point of f collides with the mold in the translation.



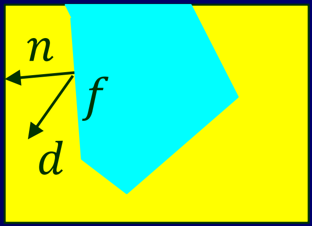
($\Leftarrow$) By contradiction.

Suppose P collides with the mold translating in direction d.

Also a Sufficient Condition

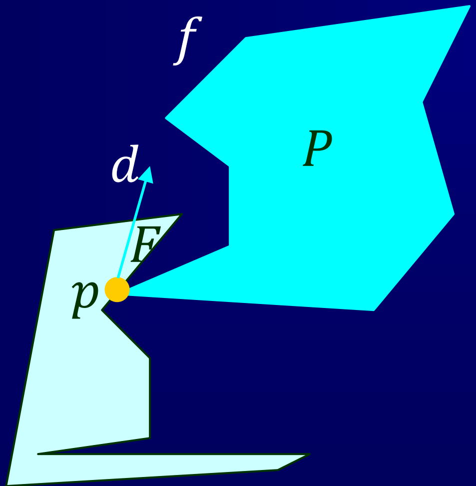
Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.



Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

Then any interior point of f collides with the mold in the translation.



($\Leftarrow$) By contradiction.

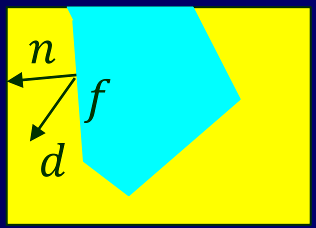
Suppose P collides with the mold translating in direction d.

Let p be the point on P that collides with a facet F of the mold. p is about to move into the interior.

Also a Sufficient Condition

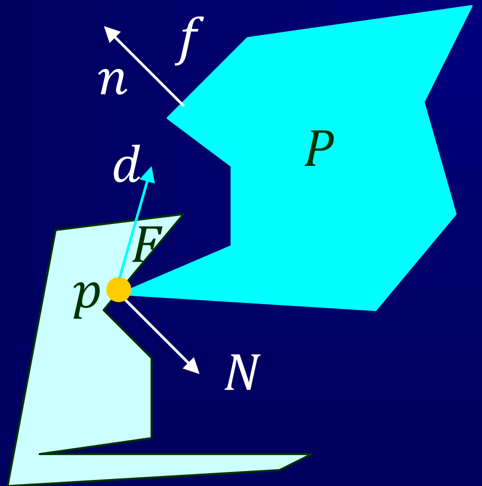
Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.



Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

Then any interior point of f collides with the mold in the translation.



($\Leftarrow$) By contradiction.

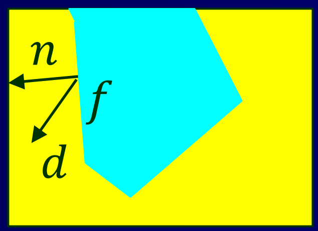
Suppose P collides with the mold translating in direction d.

Let p be the point on P that collides with a facet F of the mold. p is about to move into the interior.

Also a Sufficient Condition

Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

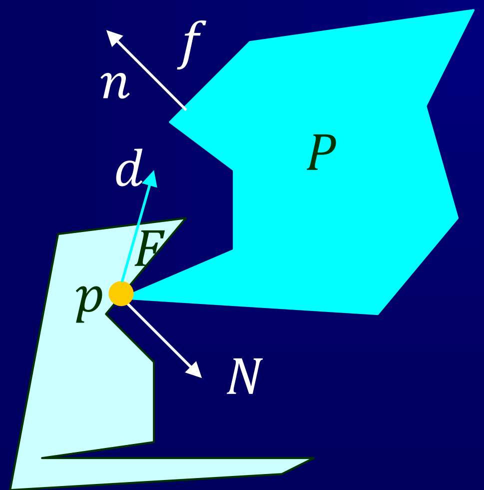
Proof ($\Rightarrow$) By contradiction.



Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

Then any interior point of f collides with the mold in the translation.

($\Leftarrow$) By contradiction.



Suppose P collides with the mold translating in direction d.

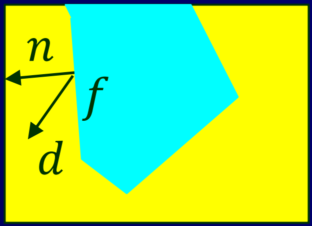
Let p be the point on P that collides with a facet F of the mold. p is about to move into the interior.

The outward normal N of F makes an angle $> \pi/2$ with d.

Also a Sufficient Condition

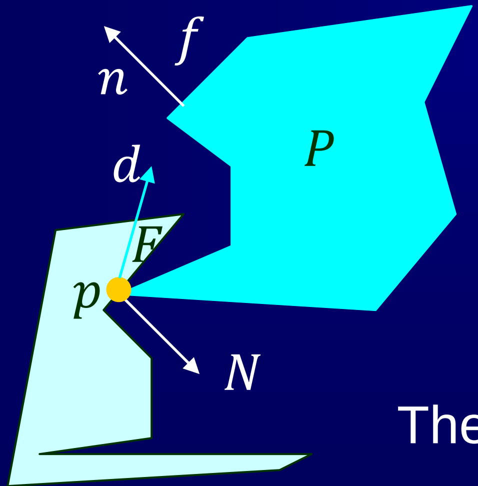
Theorem The polyhedron can translate out of the mold in a direction d if and only if d makes an angle $\geq \pi/2$ with the outward normal of every facet of the polyhedron except the top one.

Proof ($\Rightarrow$) By contradiction.



Suppose d makes an angle $< \pi/2$ with the outward normal n of some facet f.

Then any interior point of f collides with the mold in the translation.



($\Leftarrow$) By contradiction.

Suppose P collides with the mold translating in direction d.

Let p be the point on P that collides with a facet F of the mold. p is about to move into the interior.

The outward normal N of F makes an angle $> \pi/2$ with d.

The outward normal n of f makes an angle $< \pi/2$ with d.

One Translation vs. Multiple Translations

Polyhedron removable by a sequence of translations.

One Translation vs. Multiple Translations

Polyhedron removable by a sequence of translations.

➡ There exists at least one direction d which makes an angle $\geq \pi/2$ with the outward normal of every polyhedron face.

One Translation vs. Multiple Translations

Polyhedron removable by a sequence of translations.

- There exists at least one direction d which makes an angle $\geq \pi/2$ with the outward normal of every polyhedron face.

- Removable along the direction d.

One Translation vs. Multiple Translations

Polyhedron removable by a sequence of translations.

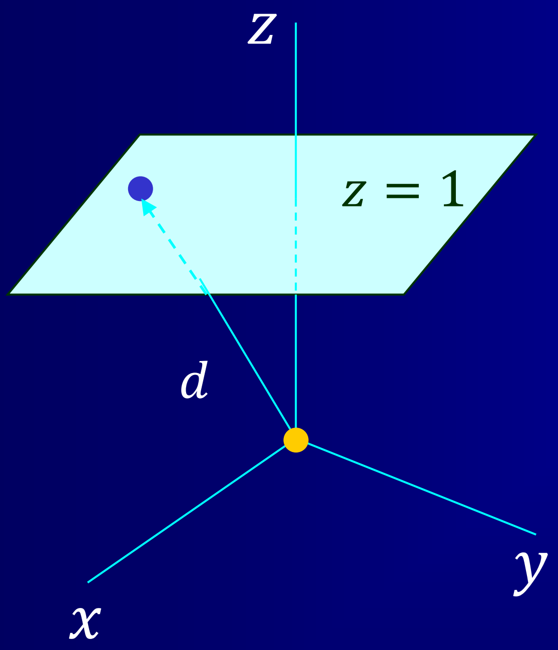
➡ There exists at least one direction d which makes an angle $\geq \pi/2$ with the outward normal of every polyhedron face.

➡ Removable along the direction d.

Allowing for multiple translations does not help.

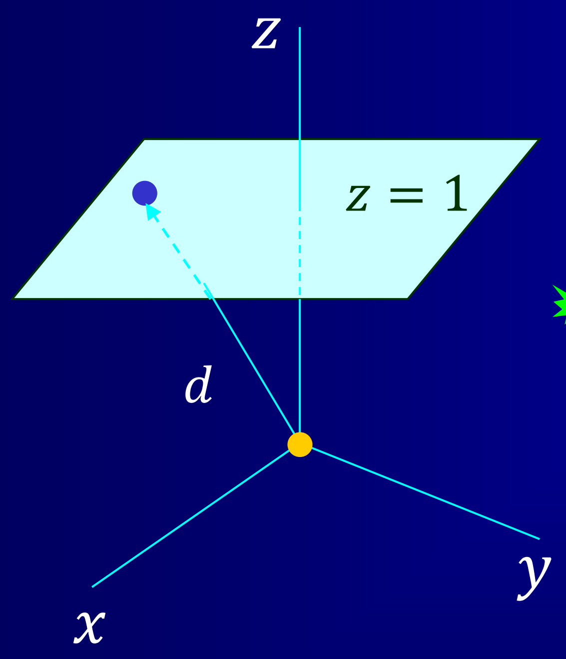
II. Representing a Direction

d to make an angle $\geq \pi/2$ with the normal of every facet.



II. Representing a Direction

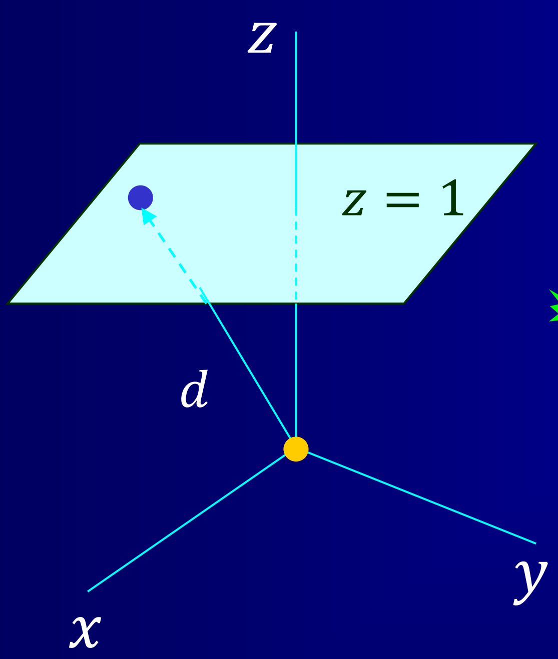
d to make an angle $\geq \pi/2$ with the normal of every facet.



- Every point $(x, y, 1)$ represents a direction.

II. Representing a Direction

d to make an angle $\geq \pi/2$ with the normal of every facet.

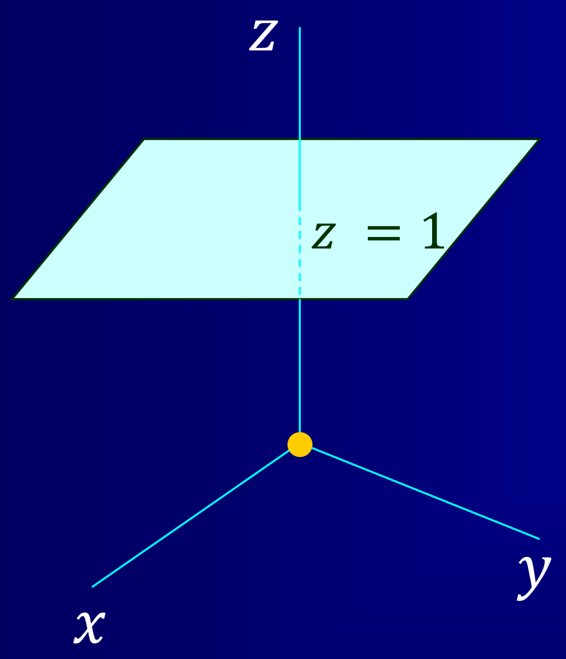


- Every point $(x, y, 1)$ represents a direction.

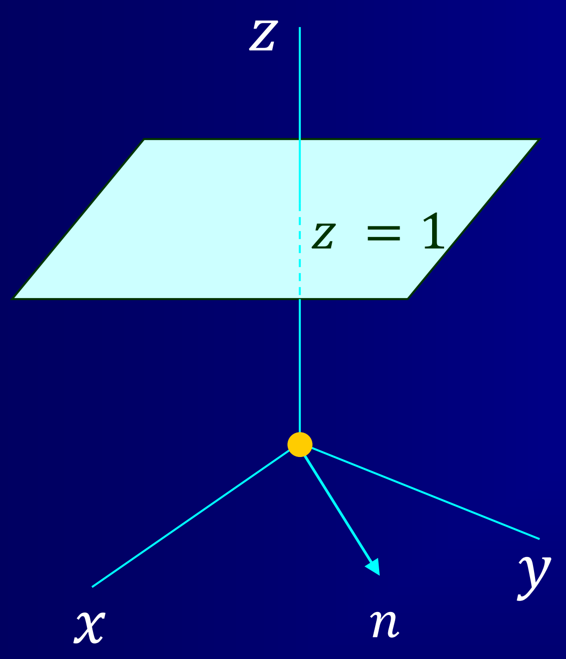
- The set of all directions with a positive z component is represented by the plane $z = 1$.

Geometric Interpretation

Let $d = (d_x, d_y, 1)$.



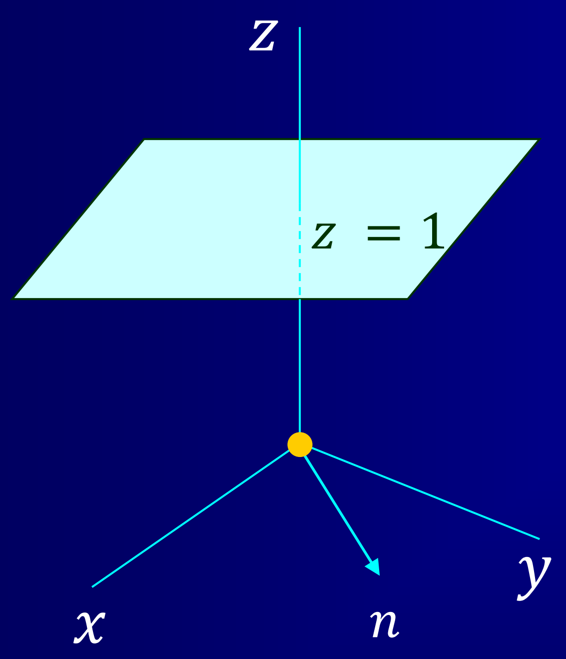
Geometric Interpretation



Let $d = (d_x, d_y, 1)$.

Let $n = (n_x, n_y, n_z)$ be the outward normal of one facet. Then

Geometric Interpretation



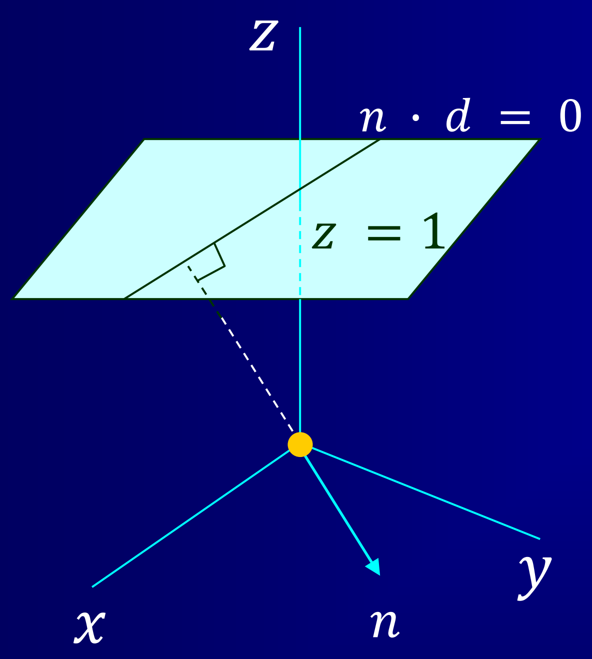
Let $d = (d_x, d_y, 1)$.

Let $n = (n_x, n_y, n_z)$ be the outward normal of one facet. Then

$$n \cdot d = n_x d_x + n_y d_y + n_z \leq 0$$

variables

Geometric Interpretation



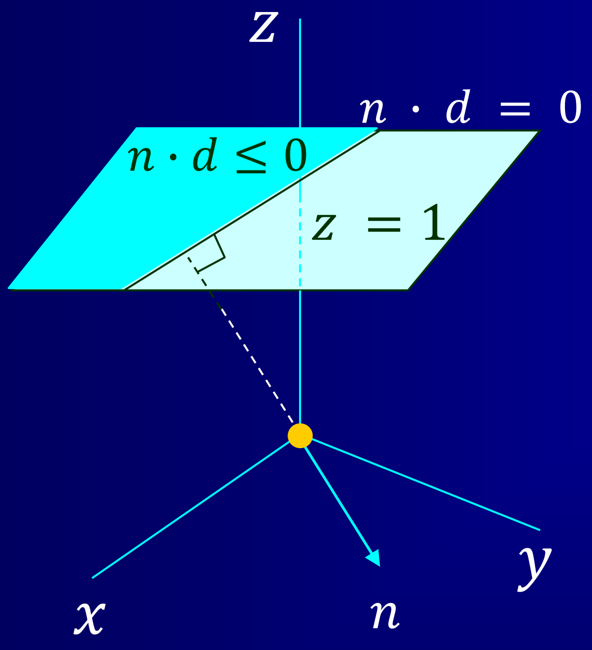
Let $d = (d_x, d_y, 1)$.

Let $n = (n_x, n_y, n_z)$ be the outward normal of one facet. Then

$$n \cdot d = n_x d_x + n_y d_y + n_z \leq 0$$

variables

Geometric Interpretation



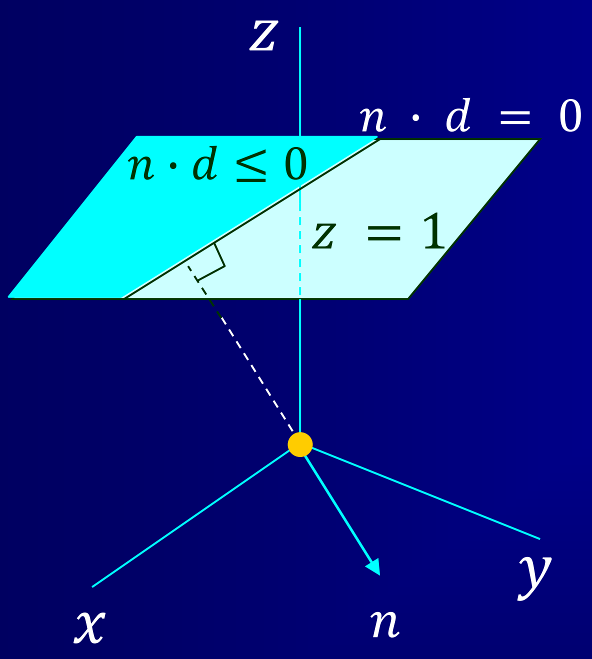
Let $d = (d_x, d_y, 1)$.

Let $n = (n_x, n_y, n_z)$ be the outward normal of one facet. Then

$$n \cdot d = n_x d_x + n_y d_y + n_z \leq 0$$

variables

Geometric Interpretation



Let $d = (d_x, d_y, 1)$.

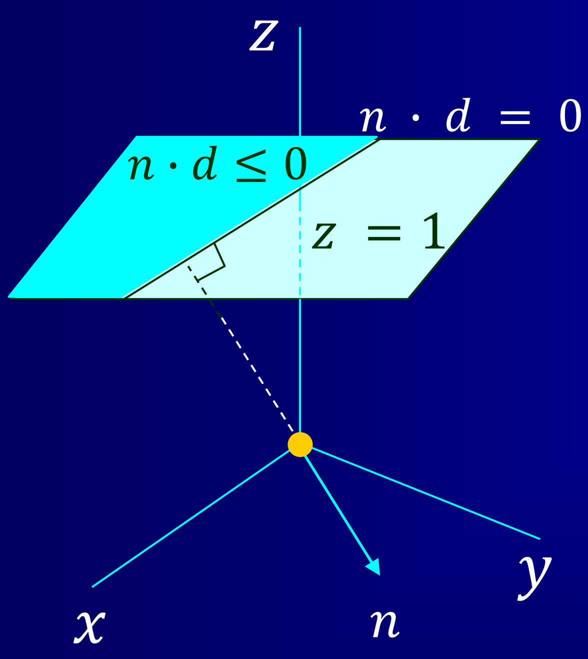
Let $n = (n_x, n_y, n_z)$ be the outward normal of one facet. Then

$$n \cdot d = n_x d_x + n_y d_y + n_z \leq 0$$

variables

- An area to one side of the line $n \cdot d = 0$ on the plane $z = 1$ (the d_x-d_y plane)

Geometric Interpretation



Let $d = (d_x, d_y, 1)$.

Let $n = (n_x, n_y, n_z)$ be the outward normal of one facet. Then

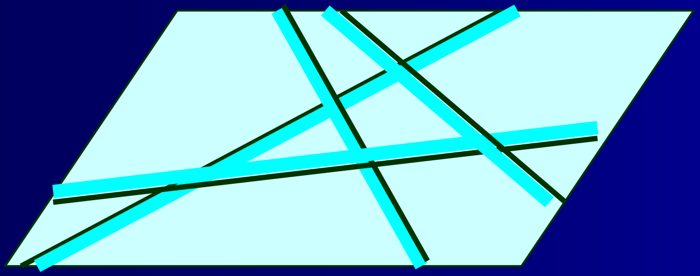
$$n \cdot d = n_x d_x + n_y d_y + n_z \le 0$$

variables

- An area to one side of the line $n \cdot d = 0$ on the plane $z = 1$ (the d_x-d_y plane)

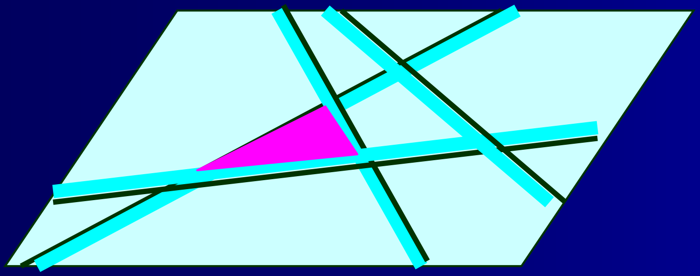
- When the facet is horizontal ($n_x = n_y = 0$), the constraint is either true for all d or false for all d depending on n_z (easy to verify).

Geometric Formulation



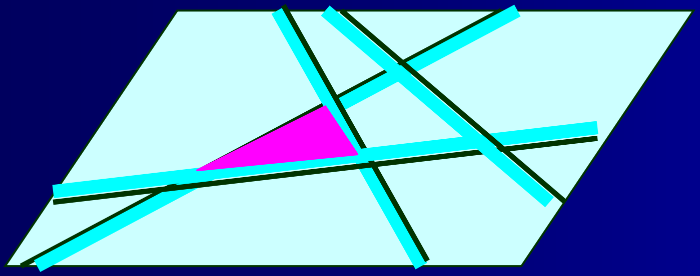
Every non-horizontal facet defines a closed half-plane of $z = 1$.

Geometric Formulation



Every non-horizontal facet defines a closed half-plane of $z = 1$.

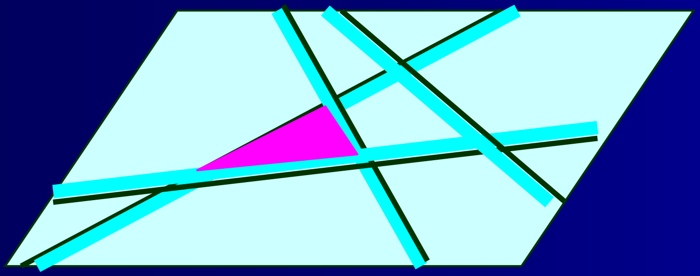
Geometric Formulation



Every non-horizontal facet defines a closed half-plane of $z = 1$.

The ***intersection*** of all such half-planes is the set of points that correspond to a direction in which the polygon can be removed.

Geometric Formulation

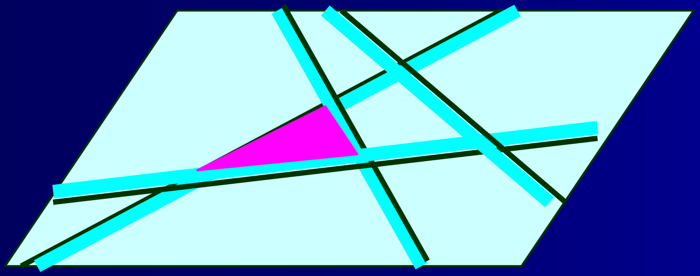


Every non-horizontal facet defines a closed half-plane of $z = 1$.

The ***intersection*** of all such half-planes is the set of points that correspond to a direction in which the polygon can be removed.

Given a set of half-planes, compute their common intersection.

Geometric Formulation



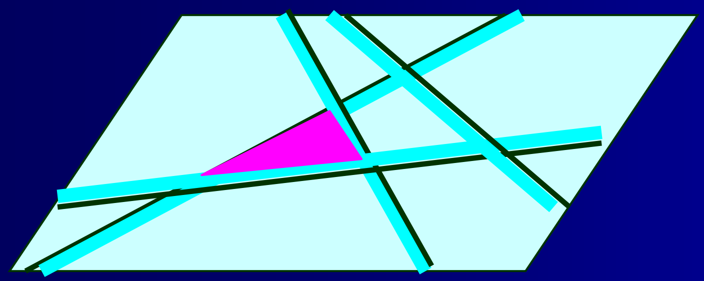
Every non-horizontal facet defines a closed half-plane of $z = 1$.

The ***intersection*** of all such half-planes is the set of points that correspond to a direction in which the polygon can be removed.

Given a set of half-planes, compute their common intersection.

Castability test: Enumerate all facets as top facet.

Geometric Formulation



Every non-horizontal facet defines a closed half-plane of $z = 1$.

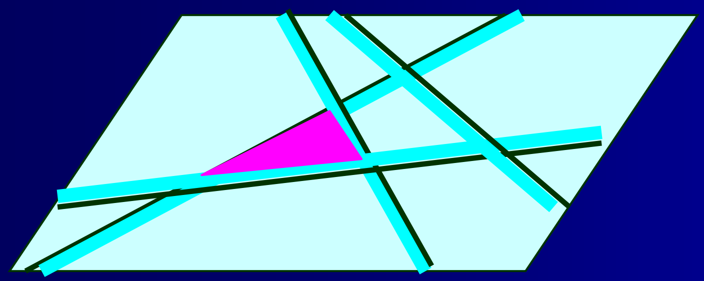
The ***intersection*** of all such half-planes is the set of points that correspond to a direction in which the polygon can be removed.

Given a set of half-planes, compute their common intersection.

Castability test: Enumerate all facets as top facet.

★ This can be done in expected time $O(n^2)$ and $O(n)$ storage.

Geometric Formulation



Every non-horizontal facet defines a closed half-plane of $z = 1$.

The ***intersection*** of all such half-planes is the set of points that correspond to a direction in which the polygon can be removed.

Given a set of half-planes, compute their common intersection.

Castability test: Enumerate all facets as top facet.

- ★ This can be done in expected time $O(n^2)$ and $O(n)$ storage.
- ★ If P is castable, a mold and a removal direction can be computed within the same time bound.

III. Intersection of Half-Planes

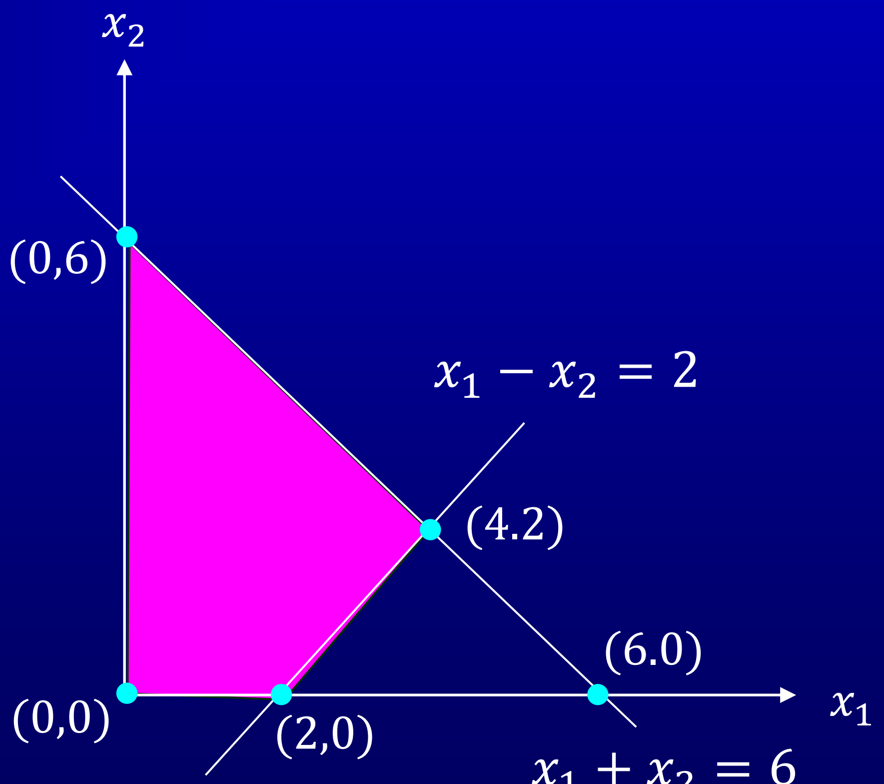
4 half-planes:

$$x_1 - x_2 \leq 2$$

$$x_1 + x_2 \leq 6$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$



General Problem

n half-planes:

$$h_i : \; a_i x + b_i y \le c_i \qquad 1 \le i \le n$$

each a convex set!

General Problem

n half-planes:

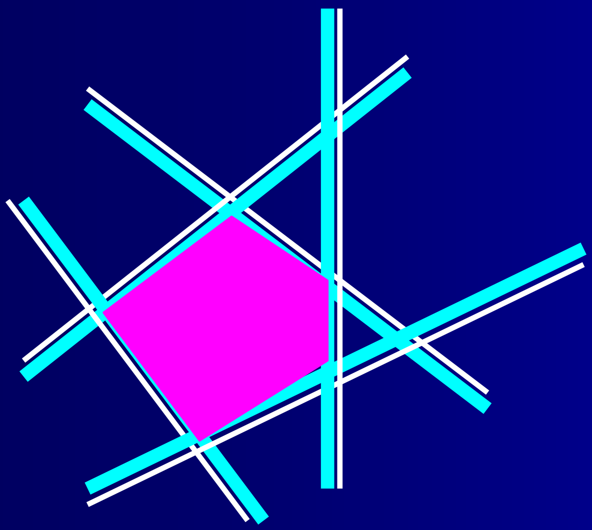
$$h_i : a_i x + b_i y \leq c_i \qquad 1 \leq i \leq n$$

each a convex set!

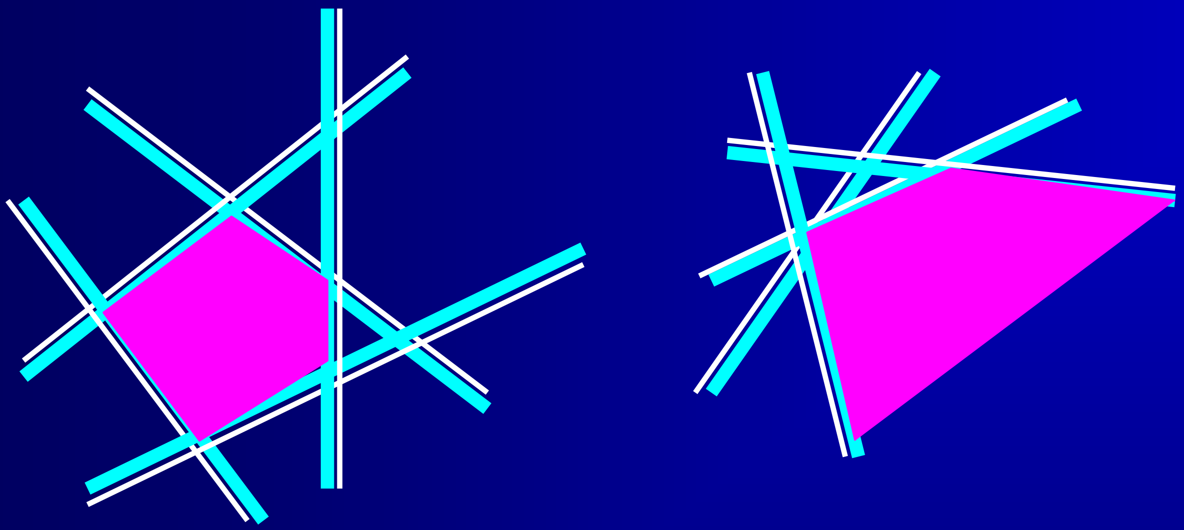
Their intersection must be a convex set:

- a convex polygonal region
- $\leq n$ edges
- possibly unbounded
- possibly degenerating into a line, segment, a point, or an empty set

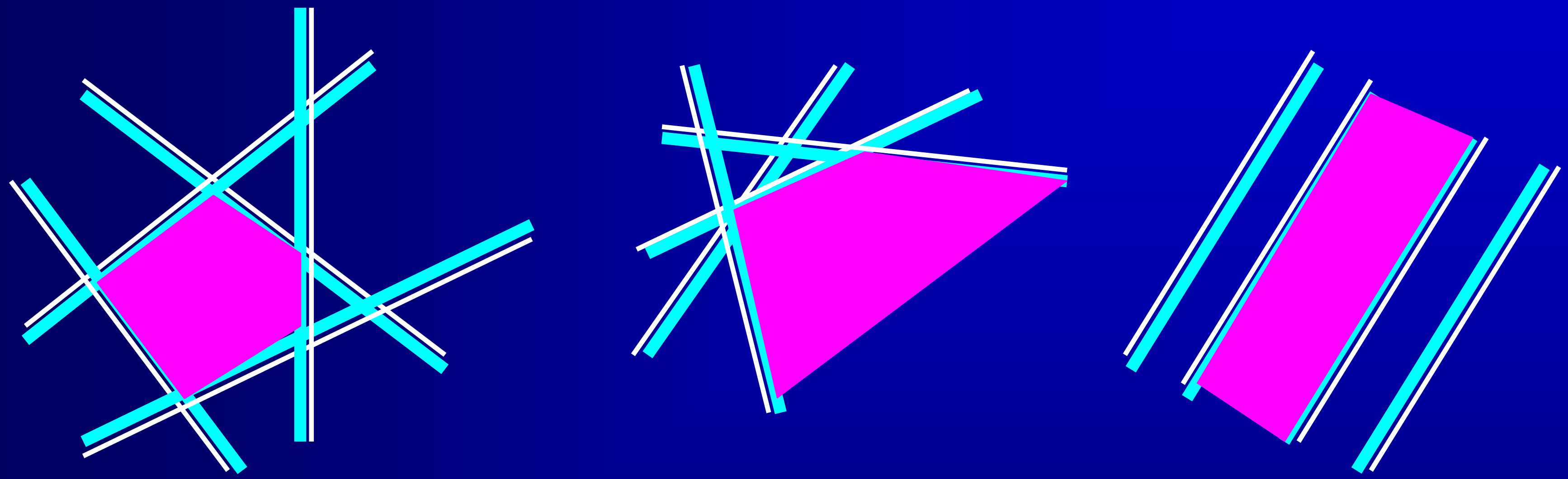
Some Possible Cases



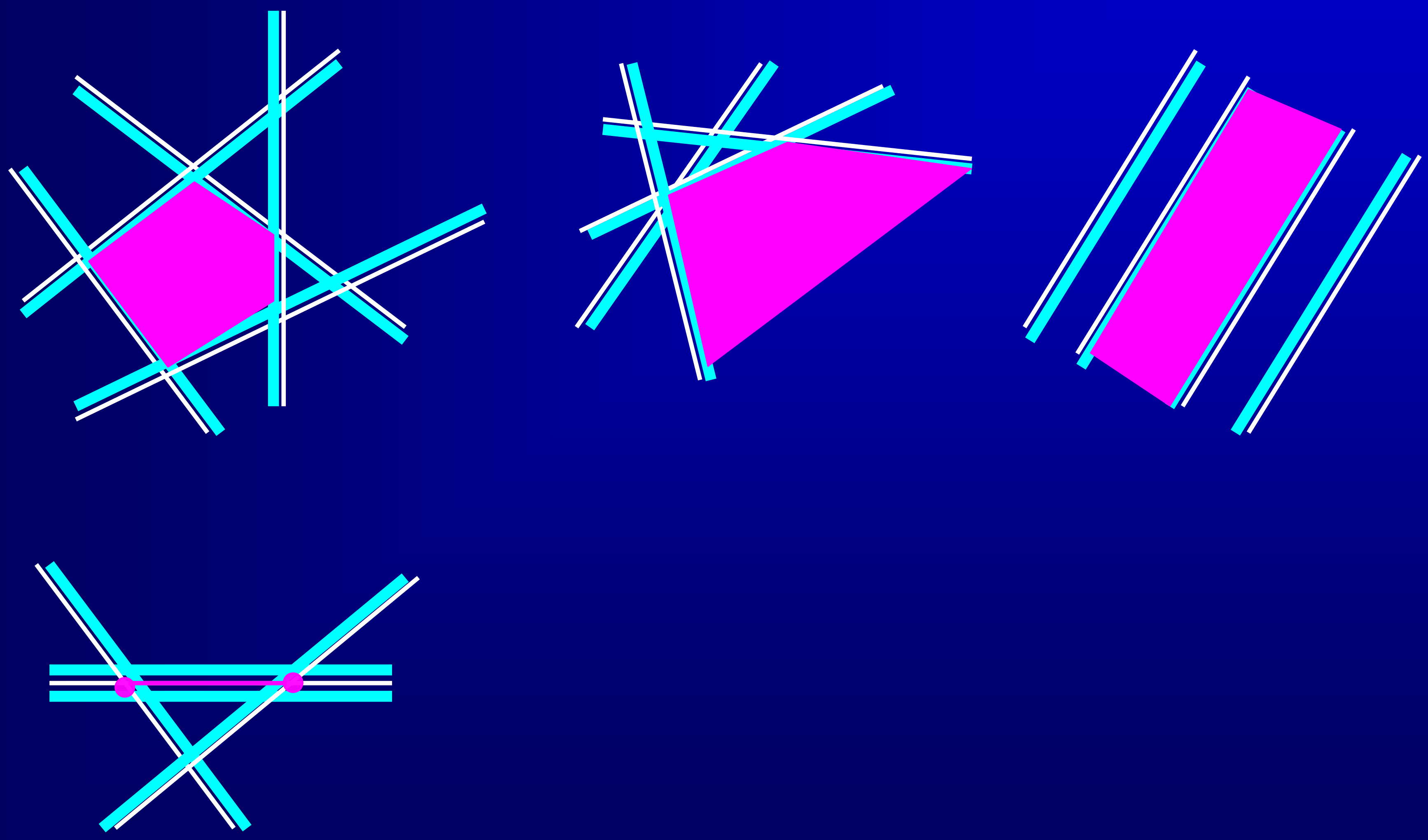
Some Possible Cases



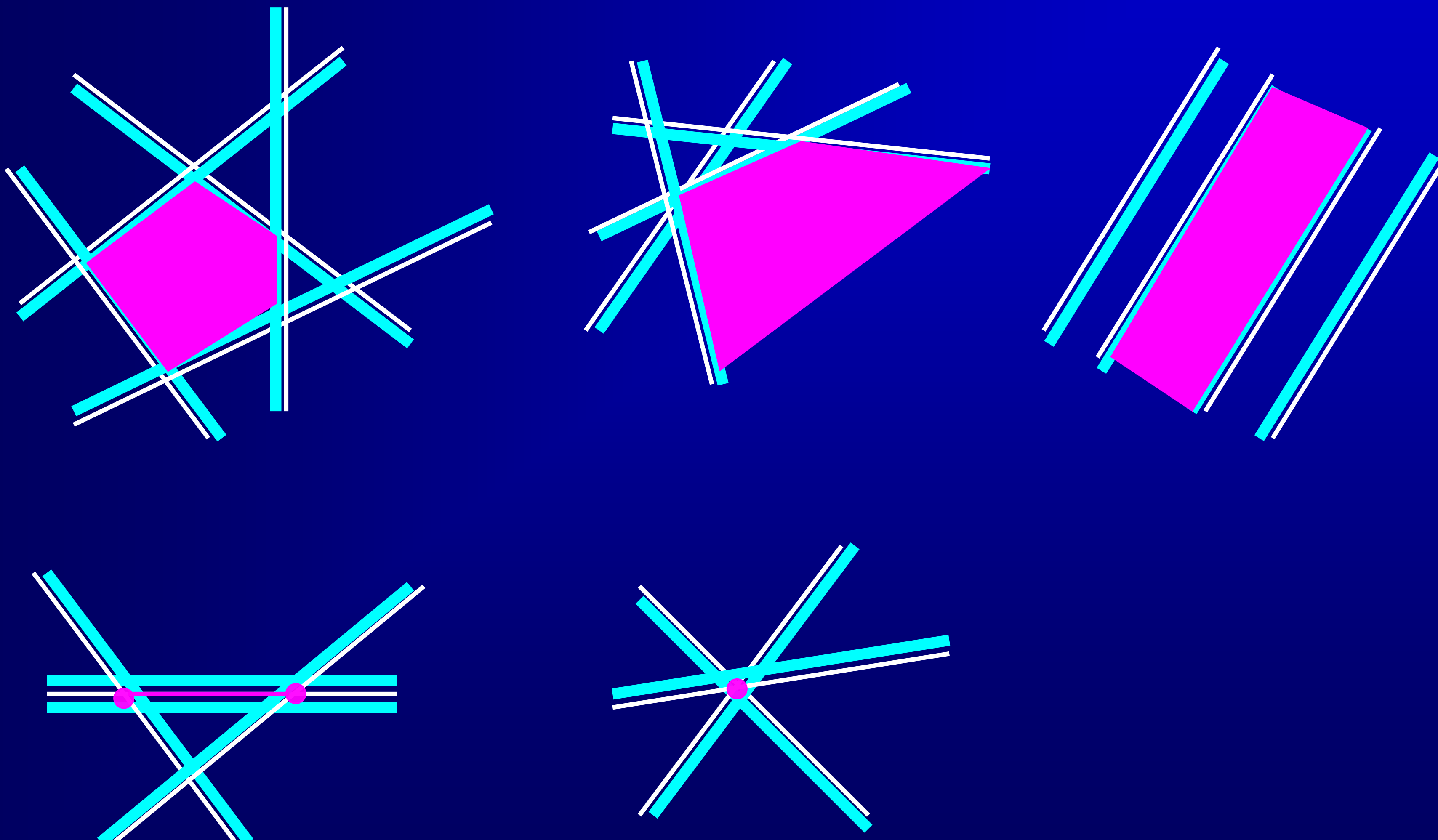
Some Possible Cases



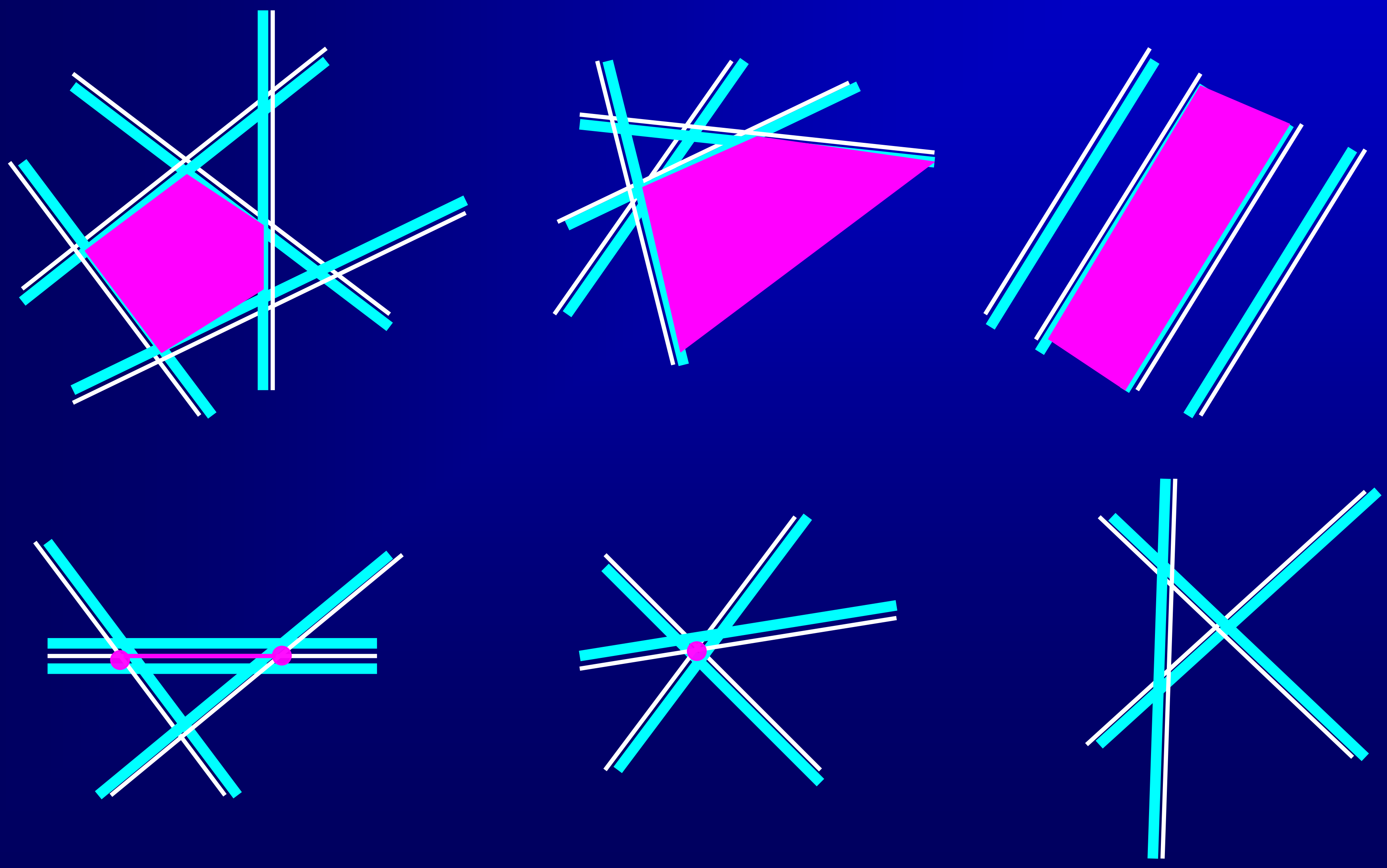
Some Possible Cases



Some Possible Cases



Some Possible Cases



A Divide-and-Conquer Algorithm

```
IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region C = ∩_{h∈H} h
1. if |H| = 1
2.      then C ← the unique half-plane h ∈ H
3.      else split H into sets H1 and H2 of size ⌈n/2⌉ and ⌊n/2⌋
4.           C1 ← IntersectHalfplane(H1)
5.           C2 ← IntersectHalfplane(H2)
6.           C  ← IntersectConvexRegion(C1, C2)
```

Intersection of Convex Regions

Intersection of Convex Regions

The intersection of two polygons in $O((n+k)\log n)$ time.

Intersection of Convex Regions

The intersection of two polygons in $O((n + k)\log n)$ time.

#vertices

Intersection of Convex Regions

The intersection of two polygons in $O((n + k) \log n)$ time.

#vertices #intersections

Intersection of Convex Regions

The intersection of two polygons in $O((n+k)\log n)$ time.

n: #vertices k: #intersections

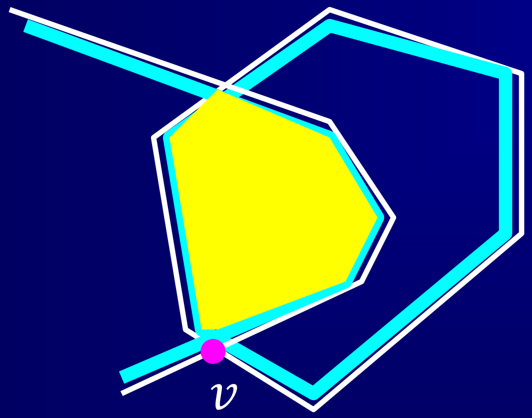
Modify the algorithm to intersect two convex regions.

Intersection of Convex Regions

The intersection of two polygons in $O((n + k) \log n)$ time.

#vertices #intersections

Modify the algorithm to intersect two convex regions.

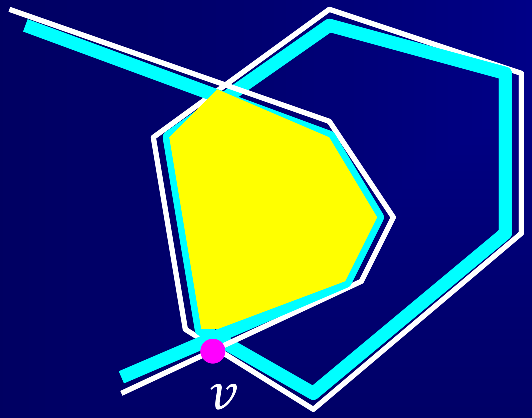


Intersection of Convex Regions

The intersection of two polygons in $O((n + k) \log n)$ time.

#vertices #intersections

Modify the algorithm to intersect two convex regions.



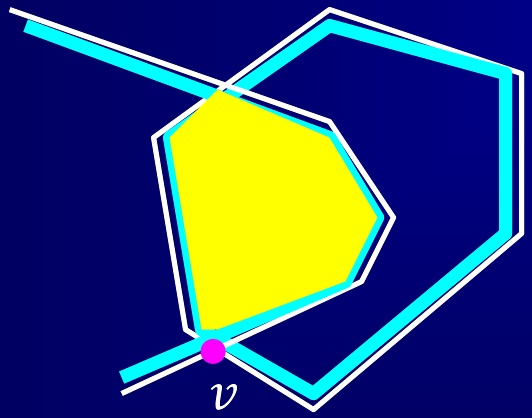
- Every intersection v of an edge of one region with an edge of the other must be a vertex of the intersection region.

Intersection of Convex Regions

The intersection of two polygons in $O((n+k)\log n)$ time.

#vertices #intersections

Modify the algorithm to intersect two convex regions.



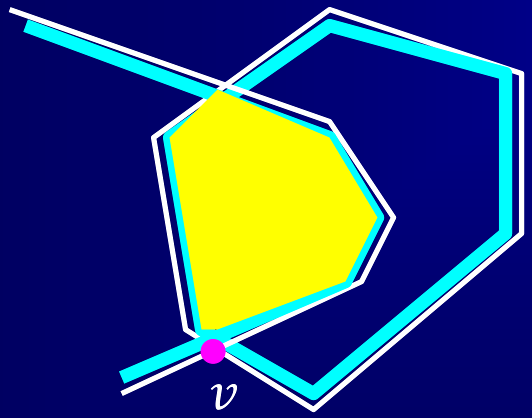
- Every intersection v of an edge of one region with an edge of the other must be a vertex of the intersection region.
- The intersection region has $\leq n$ edges and vertices.

Intersection of Convex Regions

The intersection of two polygons in $O((n + k) \log n)$ time.

#vertices #intersections

Modify the algorithm to intersect two convex regions.



- Every intersection v of an edge of one region with an edge of the other must be a vertex of the intersection region.
- The intersection region has $\leq n$ edges and vertices.

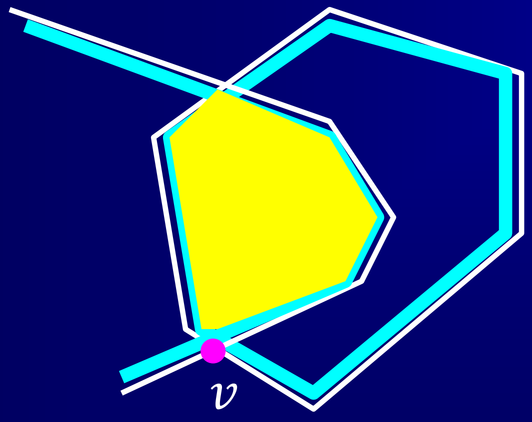
$\Rightarrow k \leq n$

Intersection of Convex Regions

The intersection of two polygons in $O((n + k)\log n)$ time.

#vertices (← n) #intersections (← k)

Modify the algorithm to intersect two convex regions.



- Every intersection v of an edge of one region with an edge of the other must be a vertex of the intersection region.
- The intersection region has $\le n$ edges and vertices.

$\Rightarrow k \le n$

$\Rightarrow$ IntersectConvexRegion takes time $O(n \log n)$.

The Recurrence

Let $T(n)$ be the running time.

```
IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region C = ∩ h
1. if |H| = 1
2.      then C ← the unique half-plane h ∈ H
3.      else split H into sets H1 and H2 of size ⌈n/2⌉ and ⌊n/2⌋
4.           C1 ← IntersectHalfplane(H1)
5.           C2 ← IntersectHalfplane(H2)
6.           C  ← IntersectConvexRegion(C1, C2)
```

The Recurrence

Let $T(n)$ be the running time.

```
IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region C = ∩ h
1. if |H| = 1
2.      then C ← the unique half-plane h ∈ H
3.      else split H into sets H_1 and H_2 of size ⌈n/2⌉ and ⌊n/2⌋
4.           C_1 ← IntersectHalfplane(H_1)        // T(n/2)
5.           C_2 ← IntersectHalfplane(H_2)
6.           C  ← IntersectConvexRegion(C_1, C_2)
```

The Recurrence

Let $T(n)$ be the running time.

IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region $C = \cap h$
1. if $|H| = 1$
2. then $C \leftarrow$ the unique half-plane $h \in H$
3. else split H into sets H_1 and H_2 of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$
4. $C_1 \leftarrow$ IntersectHalfplane(H_1) // $T(n/2)$
5. $C_2 \leftarrow$ IntersectHalfplane(H_2) // $T(n/2)$
6. $C \leftarrow$ IntersectConvexRegion(C_1, C_2)

The Recurrence

Let $T(n)$ be the running time.

IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region $C = \cap h$
1. if $|H| = 1$
2. then $C \leftarrow$ the unique half-plane $h \in H$
3. else split H into sets H_1 and H_2 of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$
4. $C_1 \leftarrow$ IntersectHalfplane(H_1) // $T(n/2)$
5. $C_2 \leftarrow$ IntersectHalfplane(H_2) // $T(n/2)$
6. $C \leftarrow$ IntersectConvexRegion(C_1, C_2) // $O(n \log n)$

The Recurrence

Let $T(n)$ be the running time.

```
IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region C = ∩ h
1. if |H| = 1
2.      then C ← the unique half-plane h ∈ H
3.      else split H into sets H_1 and H_2 of size ⌈n/2⌉and ⌊n/2⌋
4.           C_1 ← IntersectHalfplane(H_1)          // T(n/2)
5.           C_2 ← IntersectHalfplane(H_2)          // T(n/2)
6.           C  ← IntersectConvexRegion(C_1, C_2) // O(n log n)
```

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ & \end{cases}$$

The Recurrence

Let $T(n)$ be the running time.

IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region $C = \cap h$
if $|H| = 1$
  then $C \leftarrow$ the unique half-plane $h \in H$
  else split H into sets H_1 and H_2 of size $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$
    $C_1 \leftarrow$ IntersectHalfplane(H_1) // $T(n/2)$
    $C_2 \leftarrow$ IntersectHalfplane(H_2) // $T(n/2)$
    $C \leftarrow$ IntersectConvexRegion(C_1, C_2) // $O(n \log n)$

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n \log n) + 2T(n/2), & \text{if } n > 1. \end{cases}$$

The Recurrence

Let $T(n)$ be the running time.

```
IntersectHalfplane(H)
Input: A set H of n half-planes in the plane
Output: The convex polygon region C = ∩ h
1. if |H| = 1
2.      then C ← the unique half-plane h ∈ H
3.      else split H into sets H_1 and H_2 of size ⌈n/2⌉ and ⌊n/2⌋
4.           C_1 ← IntersectHalfplane(H_1)          // T(n/2)
5.           C_2 ← IntersectHalfplane(H_2)          // T(n/2)
6.           C  ← IntersectConvexRegion(C_1, C_2)   // O(n log n)
```

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n\log n) + 2T(n/2), & \text{if } n > 1. \end{cases}$$

$\Rightarrow$ $T(n) = O(n\log^2 n)$

Improvement

Can we do better?

Improvement

Can we do better?

- The subroutine for intersection of convex regions is a transplant from that for intersecting two simple polygons.

Improvement

Can we do better?

- The subroutine for intersection of convex regions is a transplant from that for intersecting two simple polygons.
- We made use of the *convexity* in our analysis.

Improvement

Can we do better?

- The subroutine for intersection of convex regions is a transplant from that for intersecting two simple polygons.
- We made use of the *convexity* in our analysis.
- But we haven't taken full advantage of convexity yet …

Improvement

Can we do better?

- The subroutine for intersection of convex regions is a transplant from that for intersecting two simple polygons.
- We made use of the *convexity* in our analysis.
- But we haven’t taken full advantage of convexity yet …

Yes!

Improvement

Can we do better?

- The subroutine for intersection of convex regions is a transplant from that for intersecting two simple polygons.
- We made use of the *convexity* in our analysis.
- But we haven’t taken full advantage of convexity yet …

Yes!

Assumption (non-degeneracy):

The regions to be intersected are 2-dimensional.

Improvement

Can we do better?

- The subroutine for intersection of convex regions is a transplant from that for intersecting two simple polygons.
- We made use of the *convexity* in our analysis.
- But we haven't taken full advantage of convexity yet …

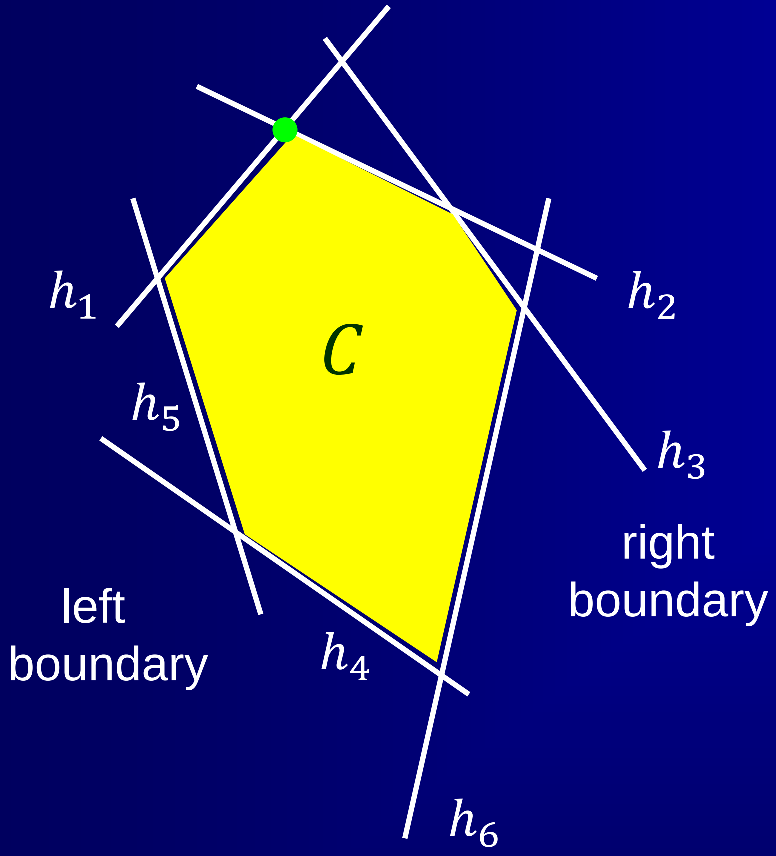
Yes!

Assumption (non-degeneracy):

The regions to be intersected are 2-dimensional.

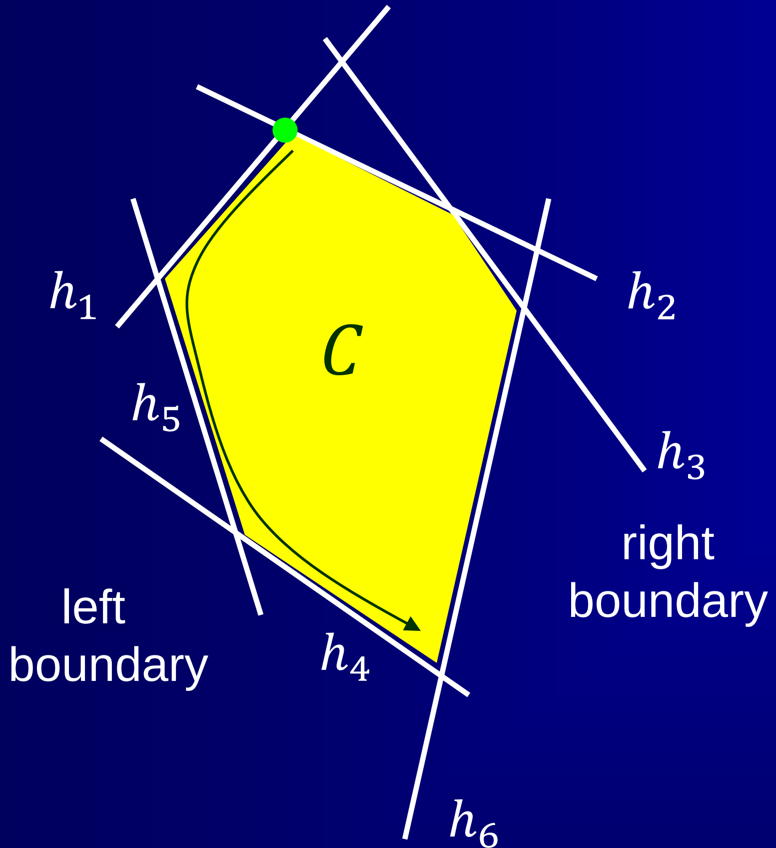
(The degenerate cases are easier.)

Representing a Convex Region



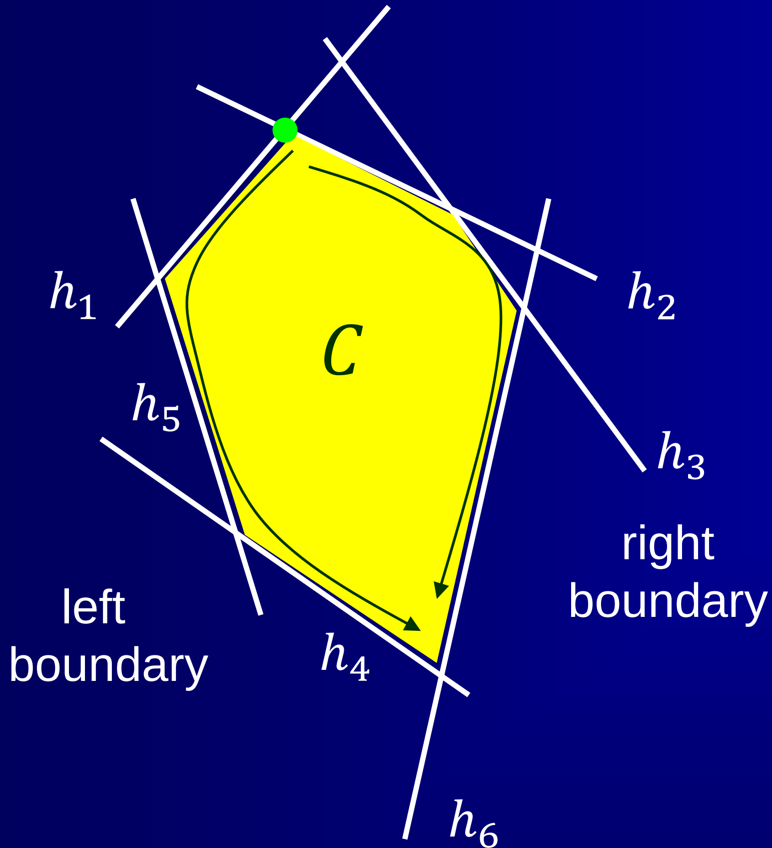
Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

Representing a Convex Region



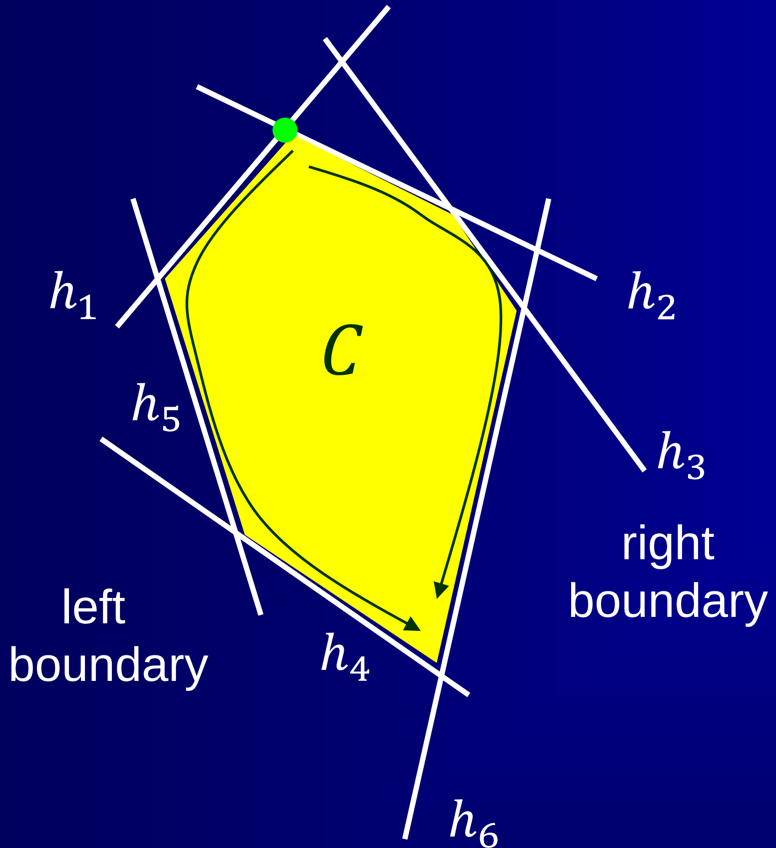
Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

Representing a Convex Region



Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

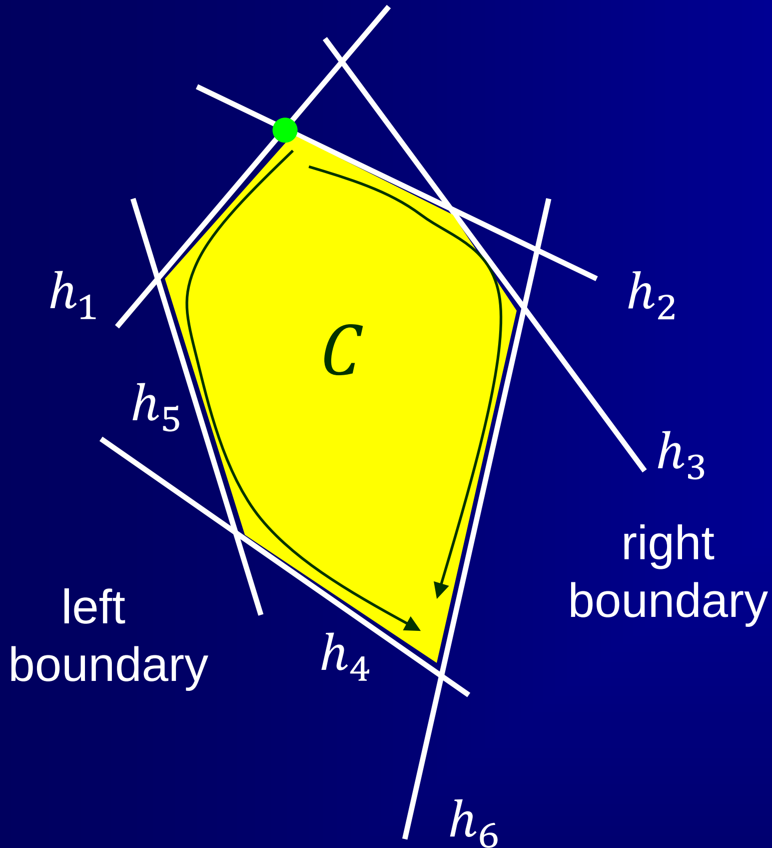
Representing a Convex Region



Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

Denote the two lists by L and R.

Representing a Convex Region



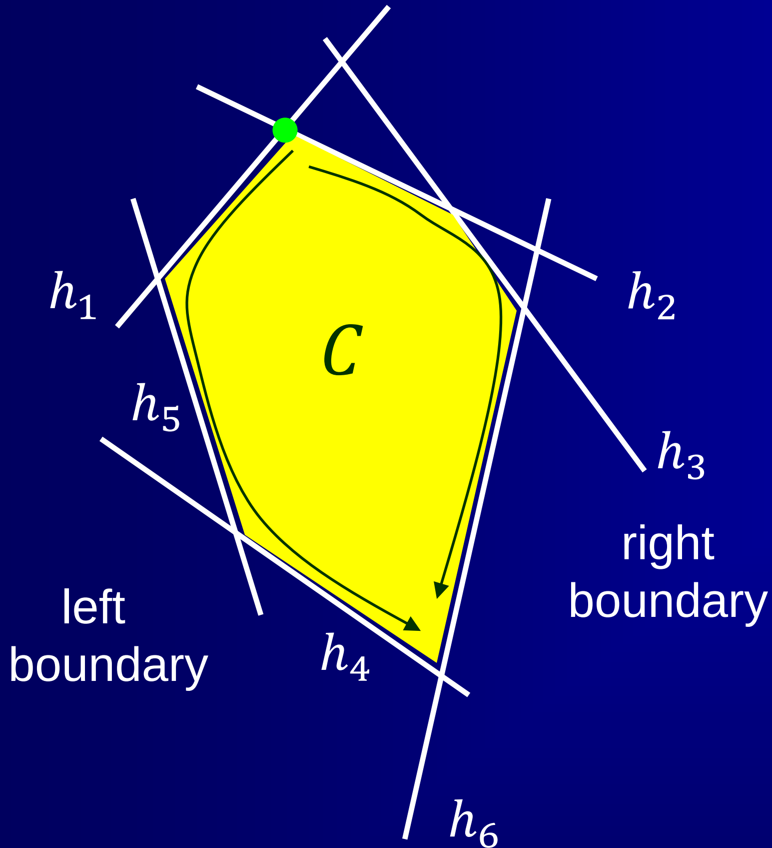
Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

Denote the two lists by L and R.

$L(C)$: h_1, h_5, h_4

$R(C)$: h_2, h_3, h_6

Representing a Convex Region



Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

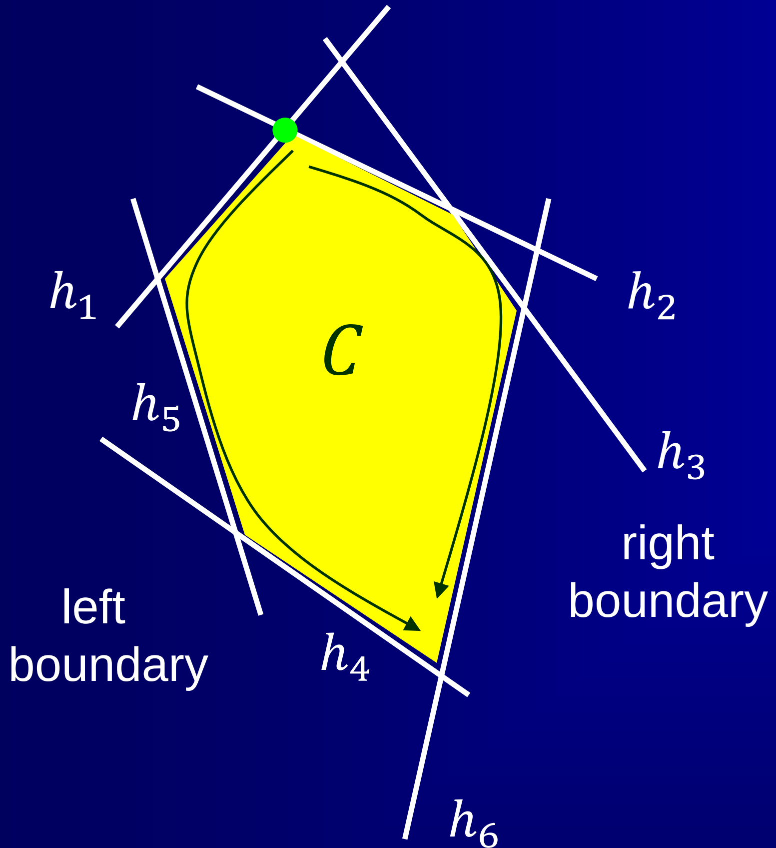
Denote the two lists by L and R.

Vertices can be easily computed by intersecting consecutive bounding lines.

$L(C)$: h_1, h_5, h_4

$R(C)$: h_2, h_3, h_6

Representing a Convex Region



$L(C)$: h_1, h_5, h_4

$R(C)$: h_2, h_3, h_6

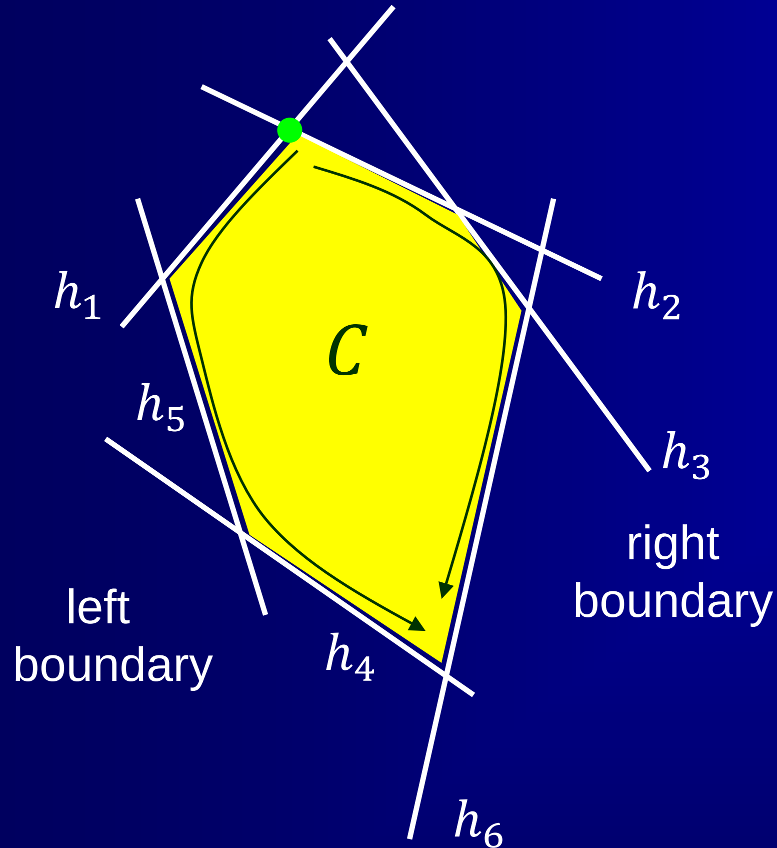
Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

Denote the two lists by L and R.

Vertices can be easily computed by intersecting consecutive bounding lines.

So they are not stored explicitly.

Representing a Convex Region



Left and right boundaries as *sorted* lists of half-planes during traversals from top to bottom.

Denote the two lists by L and R.

Vertices can be easily computed by intersecting consecutive bounding lines.

So they are not stored explicitly.

$L(C)$: h_1, h_5, h_4

$R(C)$: h_2, h_3, h_6

A horizontal edge, if exists, belongs to the left boundary if bounding C from above and to the right boundary otherwise.

Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

- At most four edges intersecting the sweep line.

  *l_e_C*1, *r_e_C*1, *l_e_C*2, *r_e_C*2

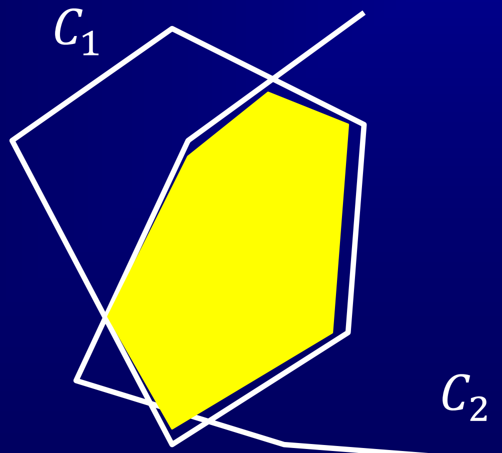
Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

- At most four edges intersecting the sweep line.

  *l*_*e*_*C*1, *r*_*e*_*C*1, *l*_*e*_*C*2, *r*_*e*_*C*2



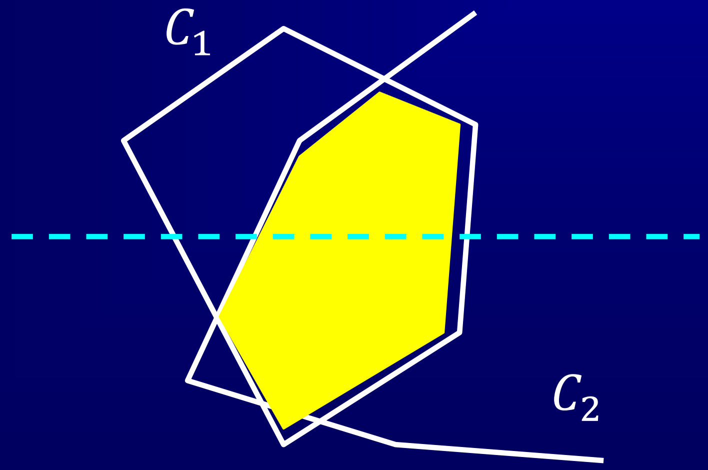
Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

- At most four edges intersecting the sweep line.

  *l_e_C*1, *r_e_C*1, *l_e_C*2, *r_e_C*2



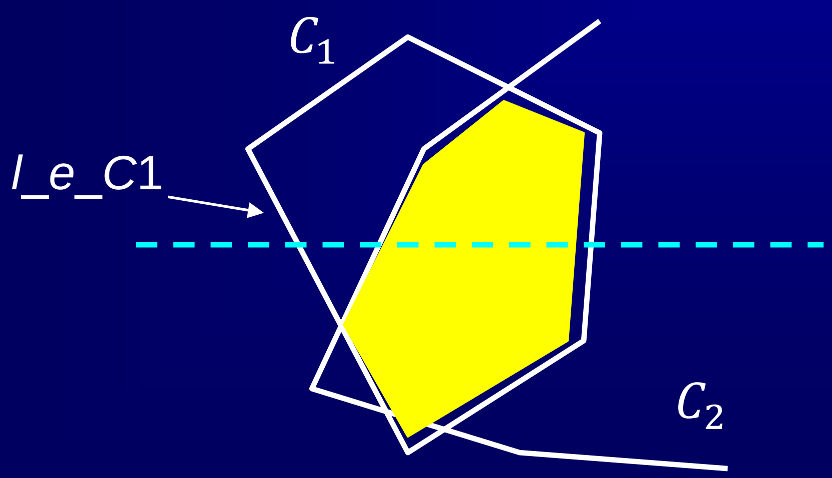
Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

- At most four edges intersecting the sweep line.

*l*_*e*_*C*1, *r*_*e*_*C*1, *l*_*e*_*C*2, *r*_*e*_*C*2



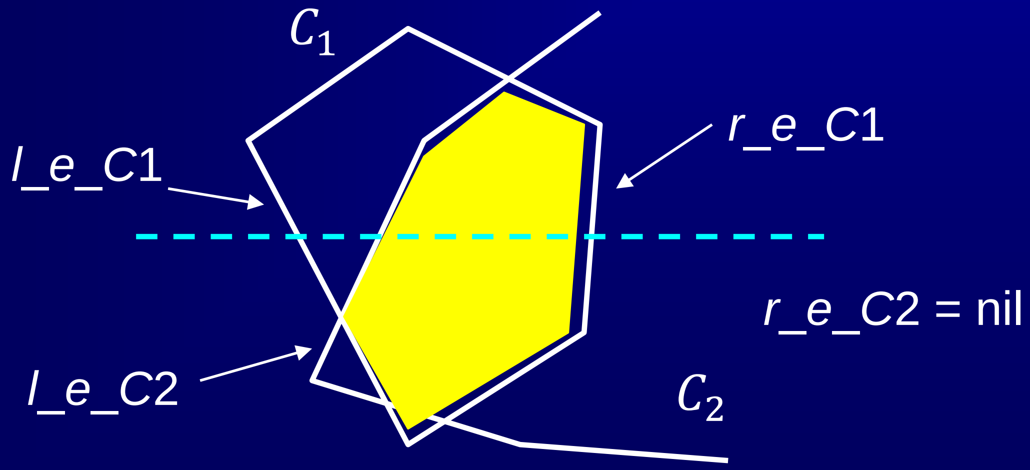
Plane Sweep Again

Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

- At most four edges intersecting the sweep line.

*l_e_C*1, *r_e_C*1, *l_e_C*2, *r_e_C*2



Plane Sweep Again

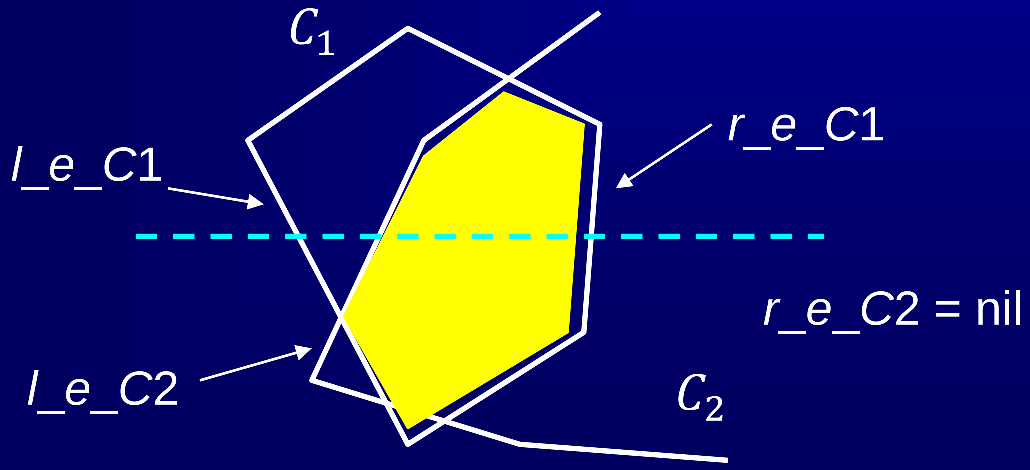
Assumption: no horizontal edge (easy to dealt with if not true).

Sweep downward to merge two convex regions C_1 and C_2.

- At most four edges intersecting the sweep line.

  *l_e_C*1, *r_e_C*1, *l_e_C*2, *r_e_C*2

- Corresponding pointer is set to nil if no intersection.



No Event Queue

Start at

the y-coordinate of the highest vertex of the two chains, or ∞ if one chain has one edge extending upward.

No Event Queue

Start at

the y-coordinate of the highest vertex of the two chains,
or ∞ if one chain has one edge extending upward.

Next event point:

No Event Queue

Start at

the y-coordinate of the highest vertex of the two chains, or ∞ if one chain has one edge extending upward.

Next event point:

Highest of the *lower* endpoints of the four edges that intersect the sweep line.

No Event Queue

Start at

the y-coordinate of the highest vertex of the two chains,
or ∞ if one chain has one edge extending upward.

Next event point:

Highest of the *lower* endpoints of the four edges that intersect the sweep line. $O(1)$

No Event Queue

Start at

the y-coordinate of the highest vertex of the two chains, or ∞ if one chain has one edge extending upward.

Next event point:

Highest of the *lower* endpoints of the four edges that intersect the sweep line. $O(1)$

The new edge e is one of the following:

1. part of C_1 and on the left chain
2. part of C_1 and on the right chain
3. part of C_2 and on the left chain
4. part of C_2 and on the right chain

No Event Queue

Start at

the y-coordinate of the highest vertex of the two chains, or ∞ if one chain has one edge extending upward.

Next event point:

Highest of the *lower* endpoints of the four edges that intersect the sweep line. $O(1)$

The new edge e is one of the following:

1. part of C_1 and on the left chain
2. part of C_1 and on the right chain
3. part of C_2 and on the left chain
4. part of C_2 and on the right chain

Left Boundary of Chain 1

p: upper endpoint of e.

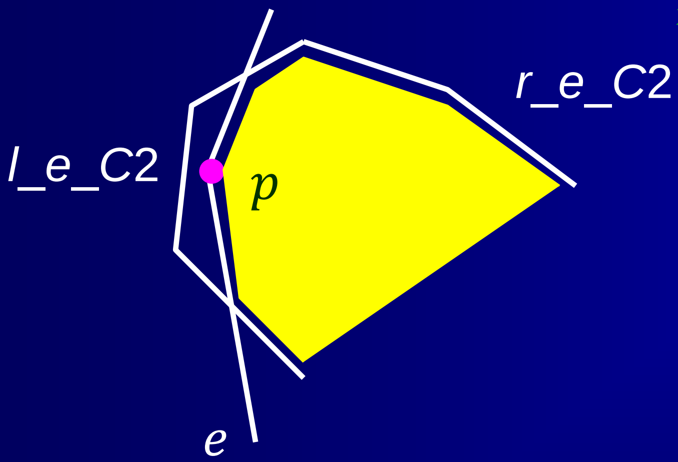
Three possible cases involving e and p in the intersection C:

Left Boundary of Chain 1

p: upper endpoint of e.

Three possible cases involving e and p in the intersection C:

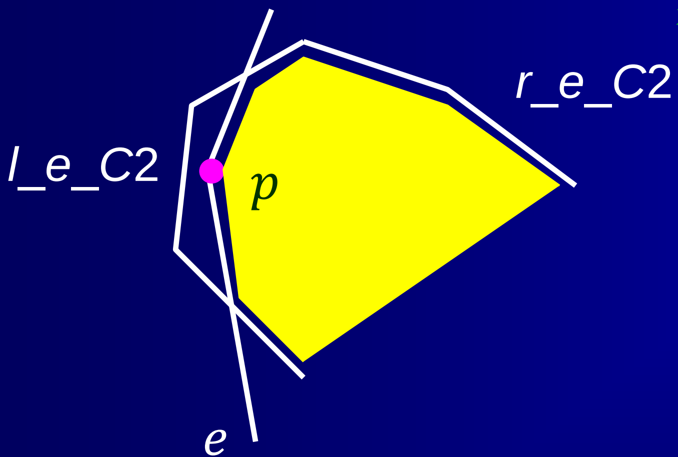
- p lies inside C_2



Left Boundary of Chain 1

p: upper endpoint of e.

Three possible cases involving e and p in the intersection C:

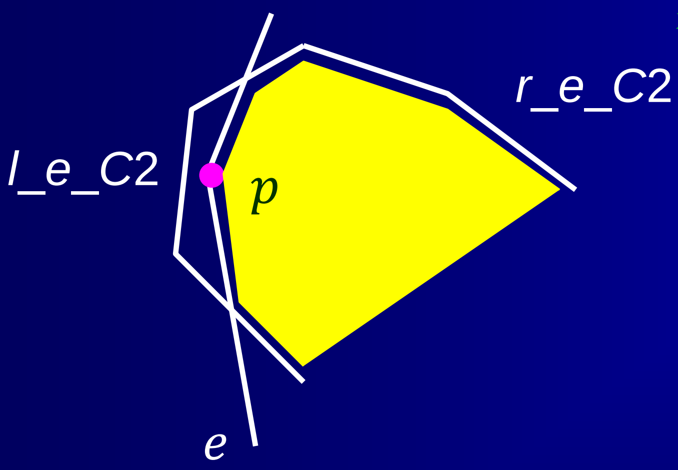


- p lies inside C_2
  - ⇒ C has an edge with p as the upper endpoint.

Left Boundary of Chain 1

p: upper endpoint of e.

Three possible cases involving e and p in the intersection C:



- p lies inside C_2

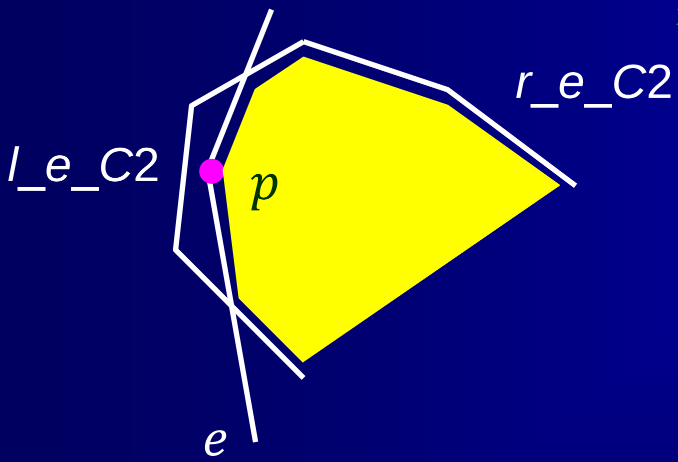
→ C has an edge with p as the upper endpoint.

This can be determined by checking whether p is between *l_e_C2* and *r_e_C2*.

Left Boundary of Chain 1

p: upper endpoint of e.

Three possible cases involving e and p in the intersection C:



- p lies inside C_2

➡ C has an edge with p as the upper endpoint.

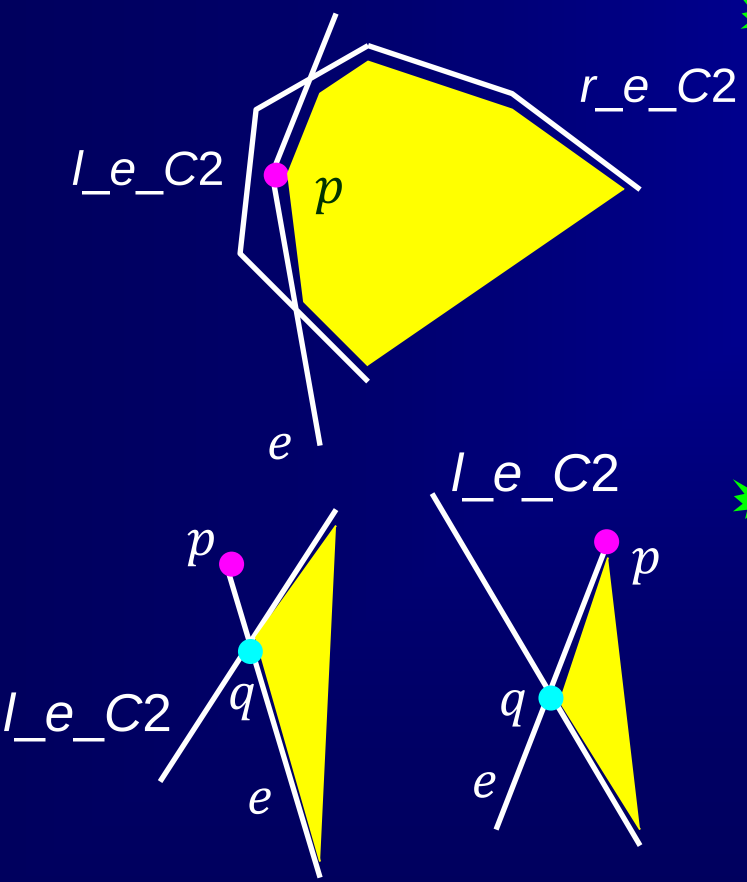
This can be determined by checking whether p is between *l_e_C2* and *r_e_C2*.

Add the half-plane with e part of its boundary to the list L.

Left Boundary of Chain 1

p: upper endpoint of e.

Three possible cases involving e and p in the intersection C:



- p lies inside C_2

  ⇒ C has an edge with p as the upper endpoint.

  This can be determined by checking whether *p* is between *l_e_C2* and *r_e_C2*.

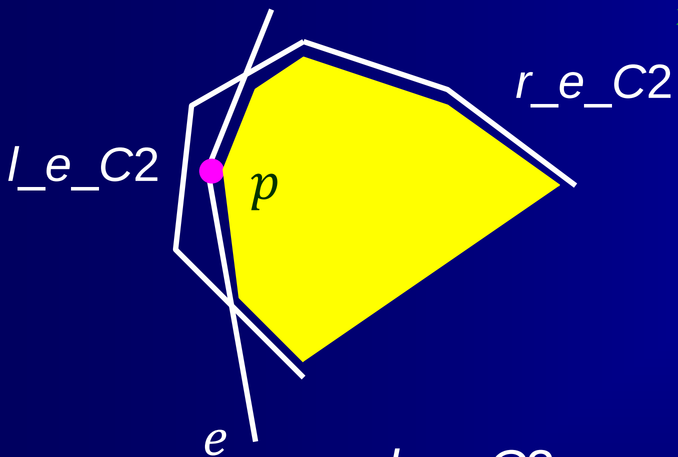
  Add the half-plane with e part of its boundary to the list $\boldsymbol{L}$.

- e intersects *l_e_C2*.

Left Boundary of Chain 1

p: upper endpoint of e.

Three possible cases involving e and p in the intersection C:

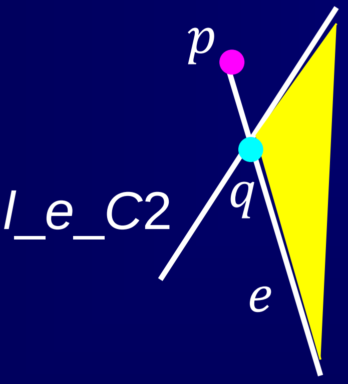


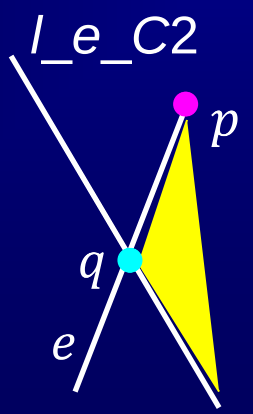
- p lies inside C_2

  ➡ C has an edge with p as the upper endpoint.

  This can be determined by checking whether p is between l_e_C2 and r_e_C2.

  Add the half-plane with e part of its boundary to the list L.





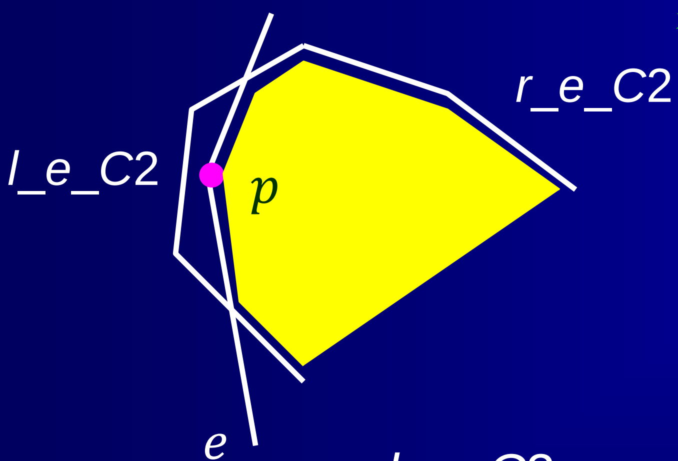
- e intersects l_e_C2.

  ➡ The intersection q is a vertex of C.

Left Boundary of Chain 1

p: upper endpoint of e.

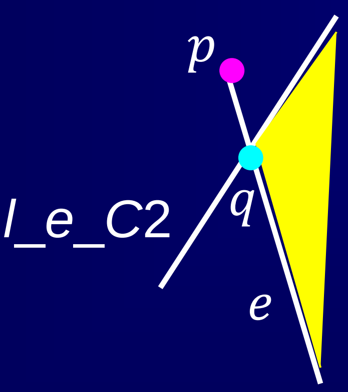
Three possible cases involving e and p in the intersection C:

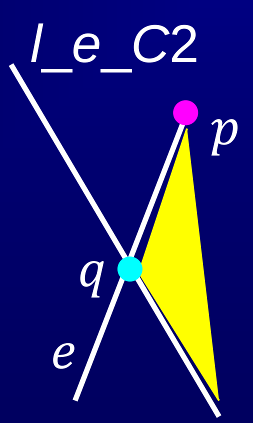


- p lies inside C_2
  - ➡ C has an edge with p as the upper endpoint.

    This can be determined by checking whether p is between *l_e_C2* and *r_e_C2*.

    Add the half-plane with e part of its boundary to the list $\boldsymbol{L}$.



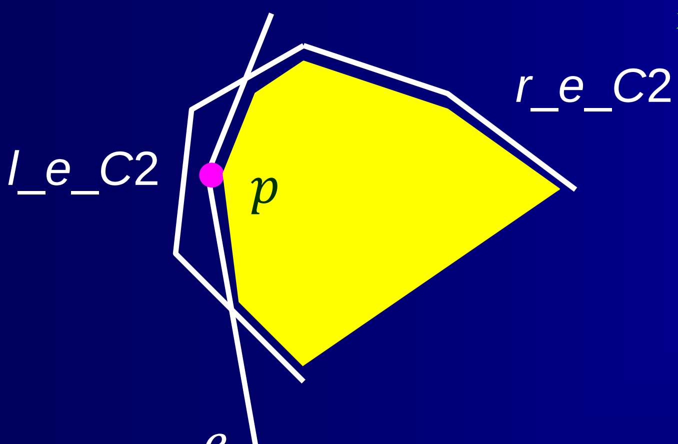


- e intersects *l_e_C2*.
  - ➡ The intersection q is a vertex of C.
  - ➡ The edge of C starting at q is part of either e (p outside of C_2) or *l_e_C2* .

Left Boundary of Chain 1

p: upper endpoint of e.

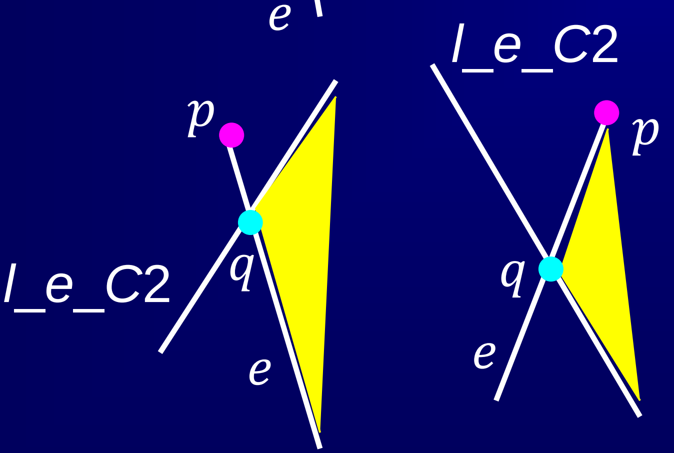
Three possible cases involving e and p in the intersection C:



- p lies inside C_2
  - ➡ C has an edge with p as the upper endpoint.

    This can be determined by checking whether p is between *l_e_C2* and *r_e_C2*.

    Add the half-plane with e part of its boundary to the list $\boldsymbol{L}$.



- e intersects *l_e_C2*.
  - ➡ The intersection q is a vertex of C.
  - ➡ The edge of C starting at q is part of either e (p outside of C_2) or *l_e_C2* .

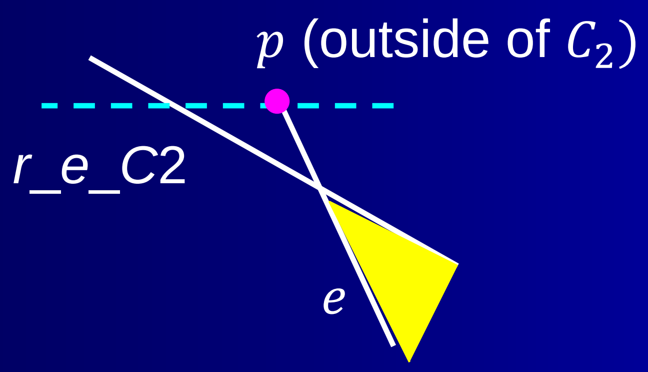
  Add the appropriate edge(s) to the list $\boldsymbol{L}$.

cont'd

- e intersects *r_e_C*2.

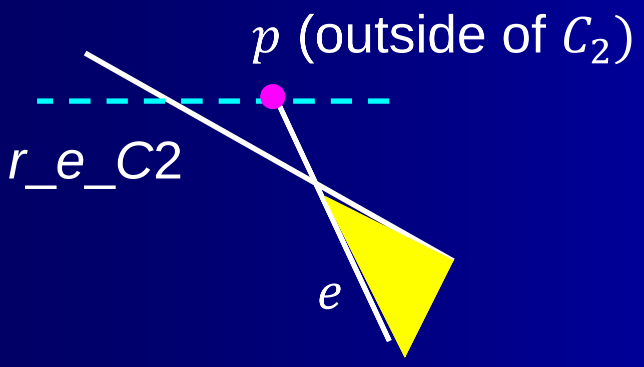
cont'd

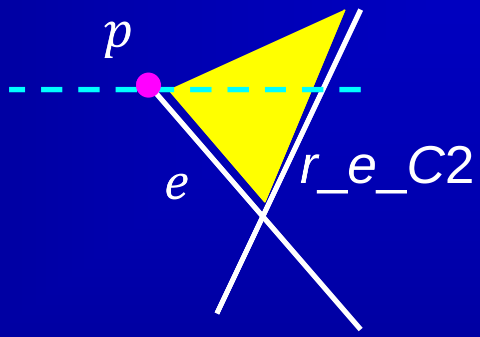
- e intersects *r_e_C*2.



cont'd

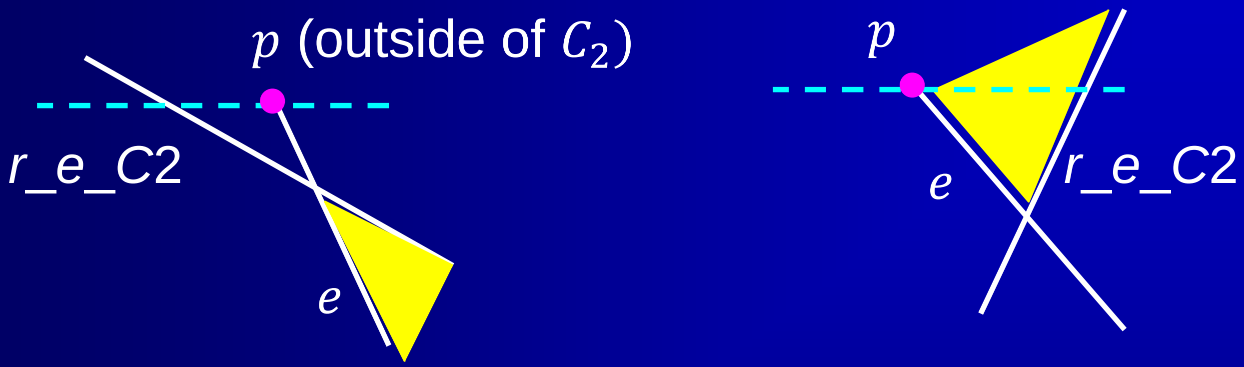
- e intersects *r_e_C*2.





cont'd

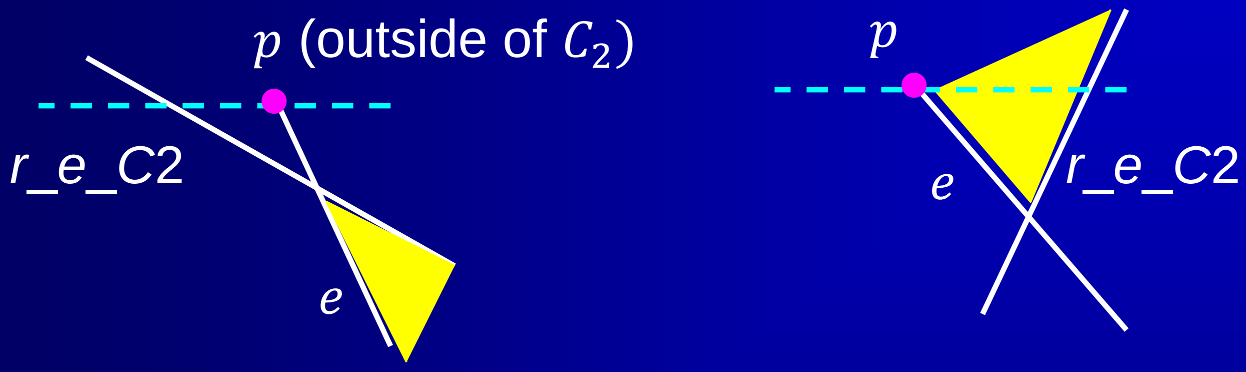
- e intersects *r_e_C*2.



➡ Each of e and *r_e_C2* contributes an edge to C at the intersection.

cont'd

- e intersects *r_e_C*2.

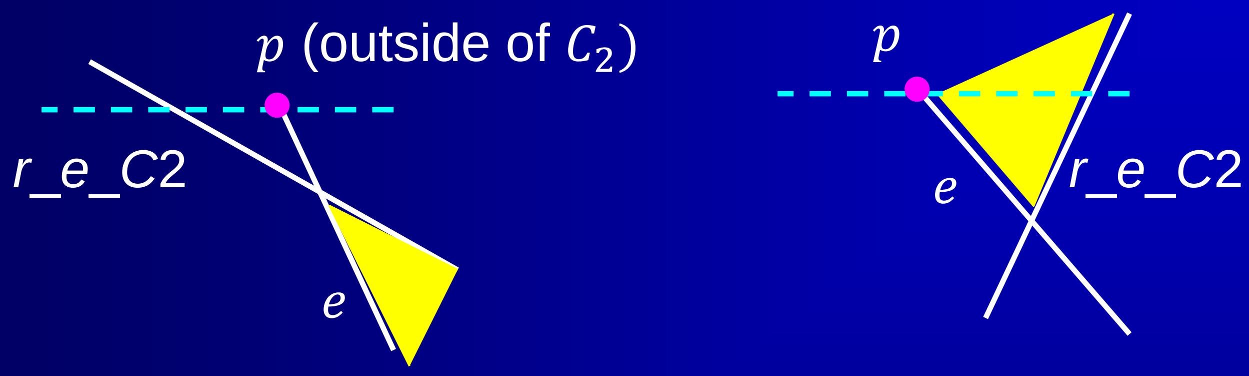


→ Each of e and *r_e_C2* contributes an edge to C at the intersection.

Case 1. The new edges start at the intersection.

cont'd

- e intersects *r_e_C*2.

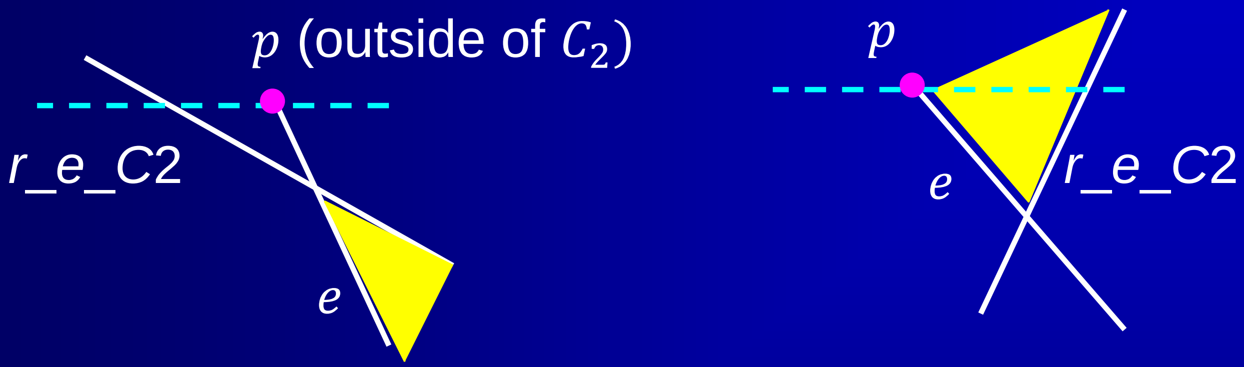


➡ Each of e and *r_e_C2* contributes an edge to C at the intersection.

Case 1. The new edges start at the intersection.
Add the half-plane defining e to L and
the one defining *r_e_C*2 to R.

cont'd

- e intersects *r_e_C*2.



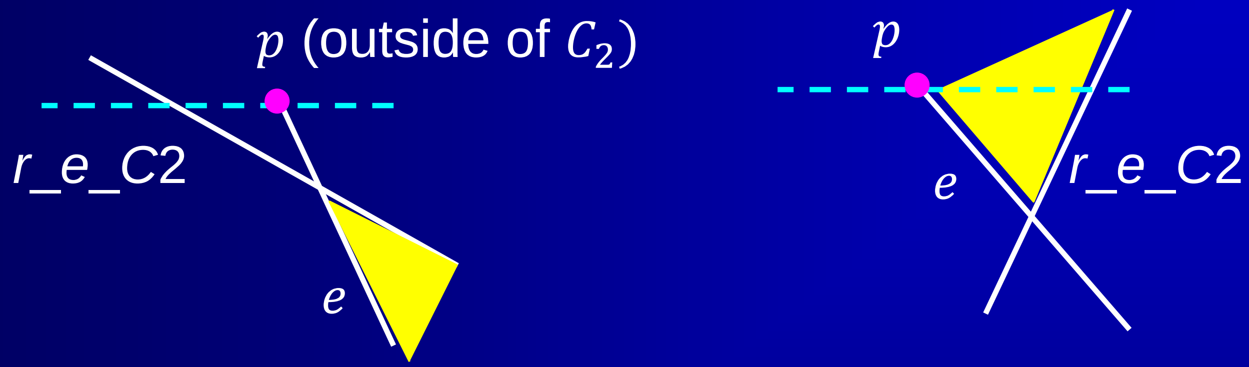
➡ Each of e and *r_e_C2* contributes an edge to C at the intersection.

Case 1. The new edges start at the intersection.
Add the half-plane defining e to $\boldsymbol{L}$ and the one defining *r_e_C*2 to $\boldsymbol{R}$.

Case 2. The new edges end at the intersection.

cont'd

- e intersects *r_e_C*2.



→ Each of e and *r_e_C2* contributes an edge to C at the intersection.

Case 1. The new edges start at the intersection.
Add the half-plane defining e to $\boldsymbol{L}$ and the one defining *r_e_C*2 to $\boldsymbol{R}$.

Case 2. The new edges end at the intersection.

Do nothing because these edges have been discovered.

Running Time

It takes $O(1)$ time to handle an edge.

Running Time

It takes $O(1)$ time to handle an edge.

Intersection of two convex polygonal regions takes $O(n)$ time.

Running Time

It takes $O(1)$ time to handle an edge.

Intersection of two convex polygonal regions takes $O(n)$ time.

Recurrence for the total running time:

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + 2T(n/2), & \text{if } n > 1. \end{cases}$$

Running Time

It takes $O(1)$ time to handle an edge.

Intersection of two convex polygonal regions takes $O(n)$ time.

Recurrence for the total running time:

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + 2T(n/2), & \text{if } n > 1. \end{cases}$$

$$\Rightarrow \quad T(n) = O(n \log n)$$

Running Time

It takes $O(1)$ time to handle an edge.

Intersection of two convex polygonal regions takes $O(n)$ time.

Recurrence for the total running time:

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + 2T(n/2), & \text{if } n > 1. \end{cases}$$

$$\Rightarrow T(n) = O(n \log n)$$

Theorem The common intersection of n half-planes in the plane can be computed in $O(n \log n)$ time and $O(n)$ storage.