Algorithmic Robotics and Motion Planning

Efi Fogel

Tel Aviv University

CGAL 2D Arrangements

Apr. 9th, 2018

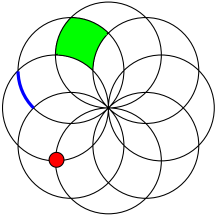
Outline

1. CGAL 2D Arrangements
    - Representation
    - Queries
        - Vertical Decomposition
        - Point Location Queries
    - The Zone Computation Algorithmic Framework
    - The Plane Sweep Algorithmic Framework
    - Arrangement of Unbounded Curves
    - Arrangement-Traits Classes
    - Extending the Arrangement
    - Map Overlay
    - Adapting to BOOST Graphs
    - Arrangement on Surfaces
    - Literature

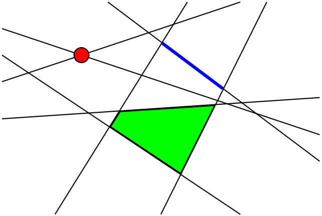
Two Dimensional Arrangements

Definition (Arrangement)

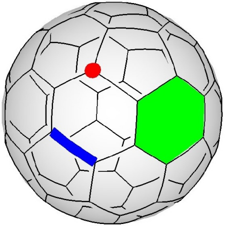
Given a collection $\mathscr{C}$ of curves on a surface, the arrangement $\mathscr{A}(\mathscr{C})$ is the partition of the surface into vertices, edges and faces induced by the curves of $\mathscr{C}$.



An arrangement of circles in the plane.



An arrangement of lines in the plane.



An arrangement of great-circle arcs on a sphere.

Outline

The CGAL `Arrangement_on_surface_2` Package

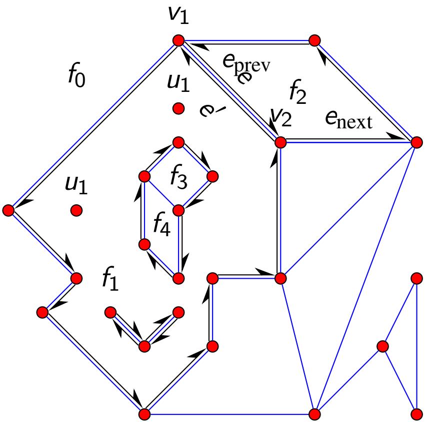
- Constructs, maintains, modifies, traverses, queries, and presents arrangements on two-dimensional parametric surfaces.
- Complete and Robust
  - All inputs are handled correctly (including degenerate input).
  - Exact number types are used to achieve robustness.
- Generic—easy to interface, extend, and adapt
- Modular—geometric and topological aspects are separated
- Supports among the others:
  - various point location strategies
  - zone-construction paradigm
  - sweep-line paradigm
    - ⋆ vertical decomposition
    - ⋆ overlay computation
    - ⋆ batched point location
- Part of the CGAL basic library

Arrangement_2<Traits, Dcel>

- Is the main component in the *2D Arrangements* package.
- An instance of this class template represents 2D arrangements.
- The representation of the arrangements and the various geometric algorithms that operate on them are separated.
- The topological and geometric aspects are separated.
  - The `Traits` template-parameter must be substituted by a model of a geometry-traits concept, e.g., *ArrangementBasicTraits_2*.
    - ⋆ Defines the type `X_monotone_curve_2` that represents x-monotone curves.
    - ⋆ Defines the type `Point_2` that represents two-dimensional points.
    - ⋆ Supports basic geometric predicates on these types.
  - The `Dcel` template-parameter must be substituted by a model of the *ArrangementDcel* concept, e.g., `Arr_default_dcel<Traits>`.

The Doubly-Connected Edge List

- One of a family of combinatorial data-structures called the *halfedge data-structures*.
- Represents each edge using a pair of directed *halfedges*.
- Maintains incidence relations among cells of 0 (vertex), 1 (edge), and 2 (face) dimensions.

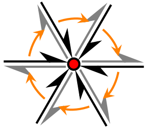


- The target vertex of a halfedge and the halefedge are incident to each other.
- The source and target vertices of a halfedge are adjacent.

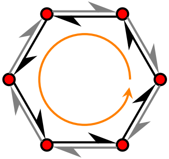
The Doubly-Connected Edge List Components

- Vertex
  - An incident halfedge pointing at the vertex.
- Halfedge
  - The opposite halfedge.
  - The previous halfedge in the component boundary.
  - The next halfedge in the component boundary.
  - The target vertex of the halfedge.
  - The incident face.
- Face
  - An incident halfedge on the outer CCB.
  - An incident halfedge on each inner CCB.
- Connected component of the boundary (CCB)
  - The circular chains of halfedges around faces.

Arrangement Representation



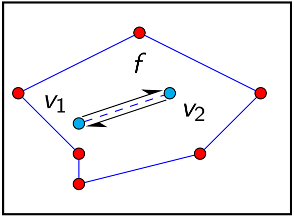
- The halfedges incident to a vertex form a circular list.
- The halfedges are clockwise oriented around the vertex.



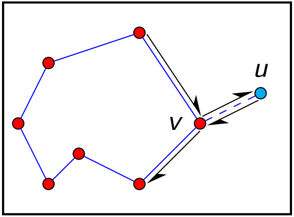
- The halfedges around faces form circular chains.
- All halfedges of a chain are incident to the same face.
- The halfedges are counterclockwise oriented along the boundary.

- Geometric interpretation is added by classes built on top of the halfedge data-structure.

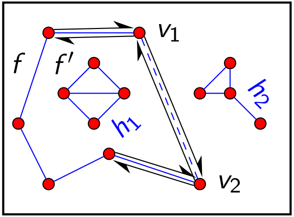
Modifying the Arrangement



Inserting a curve that induces a new hole inside the face f, `arr.insert_in_face_interior(c,f)`.



Inserting a curve from an existing vertex u that corresponds to one of its endpoints, `insert_from_left_vertex(c,v)`, `insert_from_right_vertex(c,v)`.



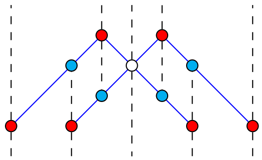
Inserting an x-monotone curve, the endpoints of which correspond to existing vertices v_1 and v_2, `insert_at_vertices(c,v1,v2)`.

- The new pair of halfedges close a new face f'.
- The hole h_1, which belonged to f before the insertion, becomes a hole in this new face.

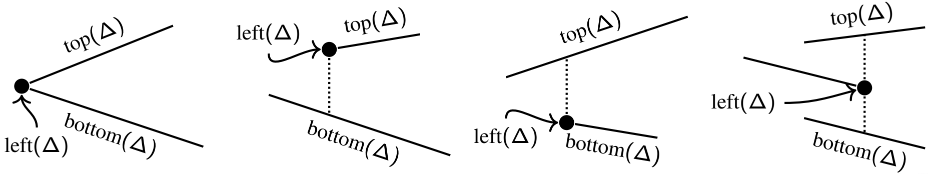
Outline

Vertical Decomposition

- Is a refinement of the original subdivision $\mathscr{A}$ of n edges.
- In the plane
  - Contains $O(n)$ pseudo trapezoids (triangles and trapezoids).
  - A pseudo trapezoid is determined by
    - ★ 2 vertices $\mathrm{left}(\Delta)$ and $\mathrm{right}(\Delta)$, and
    - ★ 2 segments $\mathrm{top}(\Delta)$ and $\mathrm{bottom}(\Delta)$.

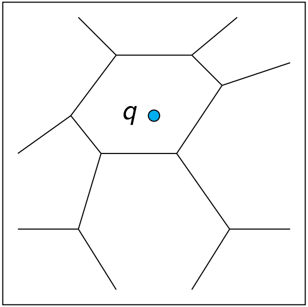


- Generalizes to higher dimensions and arrangements induces by well behaved objects.



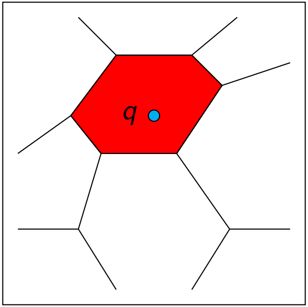
Arrangement Point Location

Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



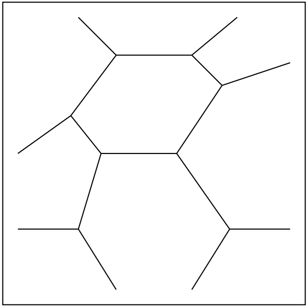
Arrangement Point Location

Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



Arrangement Point Location

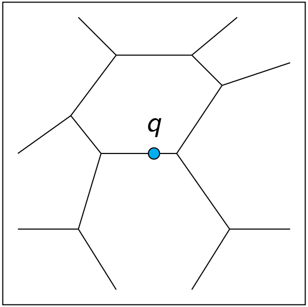
Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



- In degenerate situations the query point can

Arrangement Point Location

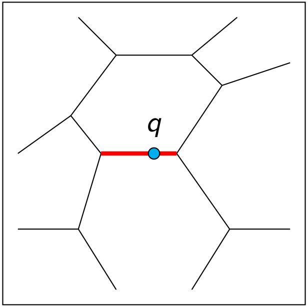
Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



- In degenerate situations the query point can
  - lie on an edge, or

Arrangement Point Location

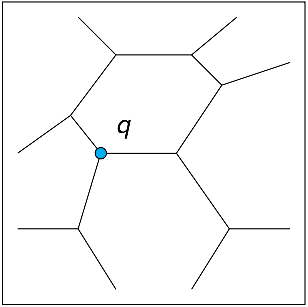
Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



- In degenerate situations the query point can
  - lie on an edge, or

Arrangement Point Location

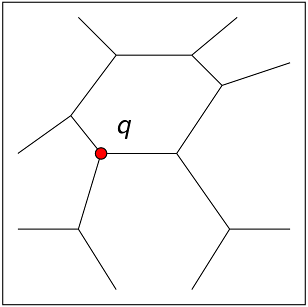
Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



- In degenerate situations the query point can
    - lie on an edge, or
    - coincide with a vertex.

Arrangement Point Location

Given a subdivision A of the space into cells and a query point q, find the cell of A containing q.



- In degenerate situations the query point can
  - lie on an edge, or
  - coincide with a vertex.

Outline

Outline

1. CGAL 2D Arrangements
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
   - Map Overlay
   - Adapting to BOOST Graphs
   - Arrangement on Surfaces
   - Literature

Outline

1. CGAL 2D Arrangements
   - Representation
   - Queries
     - Vertical Decomposition
     - Point Location Queries
   - The Zone Computation Algorithmic Framework
   - The Plane Sweep Algorithmic Framework
   - Arrangement of Unbounded Curves
   - Arrangement-Traits Classes
   - Extending the Arrangement
   - Map Overlay
   - Adapting to BOOST Graphs
   - Arrangement on Surfaces
   - Literature

Outline

Arrangement Geometry Traits

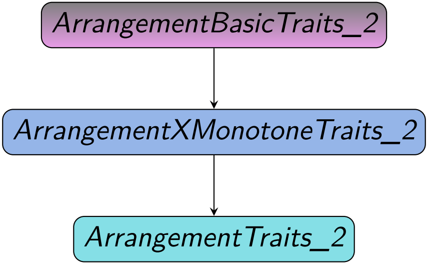
- Separates geometric asspects from topological asspects
  - `Arrangement_2<Traits, Dcel>`—main component.
- Is a parameter of the data structures and algorithms.
  - Defines the family of curves that induce the arrangement.
  - A parameterized data structure or algorithm can be used with any family of curves for which a traits class is supplied.
- Aggregates
  - Geometric types (point, curve).
  - Operations over types (accessors, predicates, constructons).

Arrangement Geometry Traits

- Separates geometric asspects from topological asspects
  - Arrangement_2<Traits , Dcel>—main component.
- Is a parameter of the data structures and algorithms.
  - Defines the family of curves that induce the arrangement.
  - A parameterized data structure or algorithm can be used with any family of curves for which a traits class is supplied.
- Aggregates
  - Geometric types (point, curve).
  - Operations over types (accessors, predicates, constructons).
- Each input curve is subdivided into x-monotone subcurves.
  - Most operations involve points and x-monotone curves.

Arrangement Traits Hierarchy

The traits-concept hierarchy for arrangements induced by bounded curves.



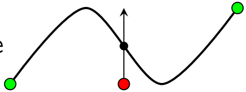
ArrangementBasicTraits_2 Concept

- Types:
  - `Point_2`
  - `X_monotone_curve_2`
- Methods:
  1. `Compare_x_2`—compares the x-coordinates of 2 points.
  2. `Compare_xy_2`—lexicographically compares the x-coordinates of 2 points.
  3. `Equal_2`
     - ★ Are two points represent the same geometric entity?
     - ★ Are two x-monotone curves represent the same geometric entity?
  4. `Construct_min_vertex`—returns the lexicographically smallest endpoint of an x-monotone curve.
  5. `Construct_max_vertex`—returns the lexicographically lasgest endpoint of an x-monotone curve.
  6. `Is_vertical`—determines whether an x-monotone curve is vertical.

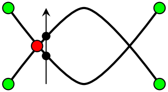
ArrangementBasicTraits_2 Concept (Cont.)

- Methods:

  7. `Compare_y_at_x_2`—determines the relative position of an x-monotone curve and a point.



  8. `Compare_y_at_x_right_2`—determines the relative position of 2 x-monotone curves to the right of a point.



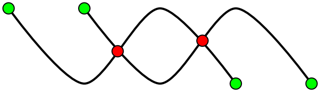
  9. `Compare_y_at_x_left_2`—determines the relative position of 2 x-monotone curves to the left of a point (optional).

- Categories:
  - `Has_left_category`—determines whether the predicate `Compare_y_at_x_left_2` is supported.
  - Determines whether x-monotone curves may reach the corresponding boundary.
    - ★ `Arr_left_side_category`
    - ★ `Arr_right_side_category`
    - ★ `Arr_bottom_side_category`
    - ★ `Arr_top_side_category`

ArrangementXMonotoneTraits_2 Concept

Supporting intersecting bounded curves.

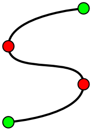
- Methods:
  1. `Split_2`—splits an x-monotone curve at a point into two interior disjoint subcurves.
  2. `Are_mergeable_2`—determines whether two curves can be merged into a single curve.
  3. `Merge_2`—merges two mergeable curves into a single curve.
  4. `Intersection_2`—finds all intersections of 2 x-monotone curves.



ArrangementTraits_2 Concept

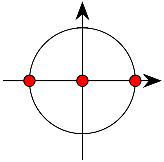
Supporting arbitrary bounded curves.

- Types: Curve_2
- Methods:



1. Make_x_monotone_2—subdivides a curve into x-monotone curves and isolated points.

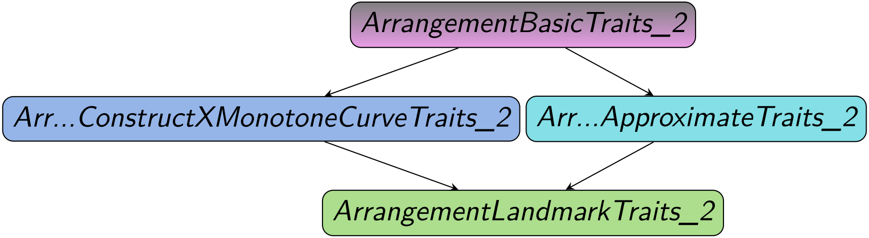
- $(x^2 + y^2)(x^2 + y^2 - 1) = 0$—the defining polynomial of a curve c.
- c comprizes of
  - the unit circle (the locus of all points for which $x^2 + y^2 = 1$) and
  - the origin (the singular point $(0,0)$).



- c is subdivided into two circular arcs and an isolated point.

Arrangement Traits Hierarchy for the Landmark Point Location

The traits-concept hierarchy for arrangements that support the Landmark Point Location strategy.



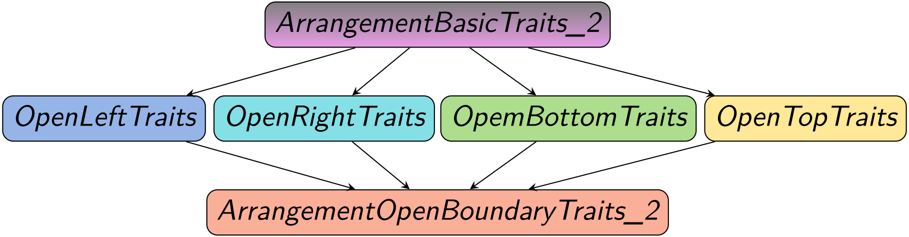
Arrangement *ArrangementLanmarkTraits_2* Concept

Supporting Lanmark point-location strategy.

- *ArrangementConstructXMonotoneCurveTraits_2* Concept
  - Methods:
    1. `Construct_x_monotone_curve_2`—constructs an x-monotone curve connecting two given points.
- *ArrangementApproximateTraits_2* Concept
  - Methods:
    1. `Approximate_2`—approximates the x- and y-coordinates of a given point using the fixed precision number type.

Arrangement Traits Hierarchy for Open Curves

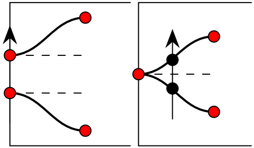
The traits-concept hierarchy for arrangements induced by unbounded curves.



ArrangementOpenBoundaryTraits_2 Concept

Supporting unbounded curves.

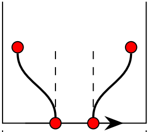
- Methods:
  1. `Parameter_space_in_x_2`—determines the location of the curve end along the *x*-dimension.
  2. `Compare_y_near_boundary_2`—compares the *y*-coordinate of 2 curve ends near their limits.



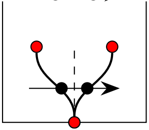
ArrangementOpenBoundaryTraits_2 Concept (cont.)

Supporting unbounded curves.

- Methods:
  1. `Parameter_space_in_y_2` — Determines the location of the curve end along the y-dimension.
  2. `Compare_x_on_boundary_2` — Compare the x-coordinate of 2 curve ends at their limits.



  3. `Compare_x_near_boundary_2` — Compare the x-coordinate of 2 curve ends near their limits.
     - ★ Precondition: the x-coordinate of the curves at their limit is equal.

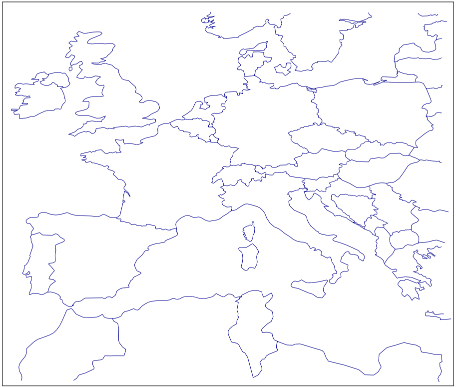


Arrangement Traits Models

- Line segments:
  1. Uses the kernel point and segment types.
  2. Caches the underlying line.
- Linear curves, i.e., line segments, rays, and lines.
- Circular arcs and line segments.
- Conic curves
- Arcs of rational functions.
- Bézier curves.
- Algebraic curves of arbitrary degrees.

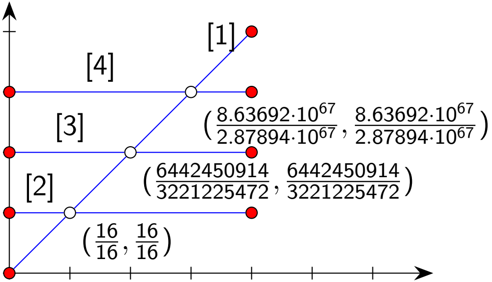
Traits Model: Non Caching Line Segments

- `Arr_non_caching_segment_traits_2<Kernel>`
- Nested point type is `Kernel::Point_2`.
- Nested curve type is `Kernel::Segment_2`.
- Most of the defined operations are delegations of the corresponding operations of the `Kernel` type.



The Effect of Cascading of Intersections

- An arrangement induced by 4 line segments.
- The segments are inserted in the order indicated in brackets.
- The insertion creates a cascading effect of segment intersection.
- The bit-lengths of the intersection-point coords grow exponentially.
- Indiscriminate normalization considerably slows down the arrangement construction.

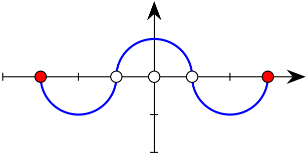


Traits Model: Caching Line Segments

- `Arr_segment_traits_2<Kernel>`
- Nested point type is `Kernel::Point_2` (like `Arr_segment_non_caching_traits_2`)
- A segment is represented by:
  - its two endpoints,
  - its supporting line,
  - a flag indicating whether the segment is vertical, and
  - a flag indicating whether the segment target-point is lexicographically larger than its source.
- Superior when the number of intersections is large.

Traits Model: Polycurves

- `Arr_polycurve_traits_2<SubcurveTraits>`
- A polycurve is a continuous non necessarily linear piecewise curve.
- `SubcurveTraits` must be a model of
  - *ArrangementTraits_2* and
  - *ArrangementDirectionalXMonotoneTraits_2*, and



Traits Model: Arcs of Rational Functions

- `Arr_rational_arc_traits_2<AlgKernel, NtTraits>`

Definition (polynomial)

A polynomial is an expression of finite length constructed from variables and constants, using only the operations of addition, subtraction, multiplication, and non-negative integer exponents.

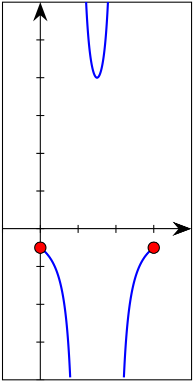
For example: $x^2 - 4x + 7$

- $P(x)$, $Q(x)$ — Univariate polynomials of arbitrary degrees.
- $y = \frac{P(x)}{Q(x)}$ — A rational function.
- The coefficient are rational numbers.
- $[x_{\min}, x_{\max}]$ is an interval over which an arc is defined $\Longrightarrow$ $x_{\min}$ and $x_{\max}$ can be arbitrary algebraic numbers.
- Suports arcs of rational functions.

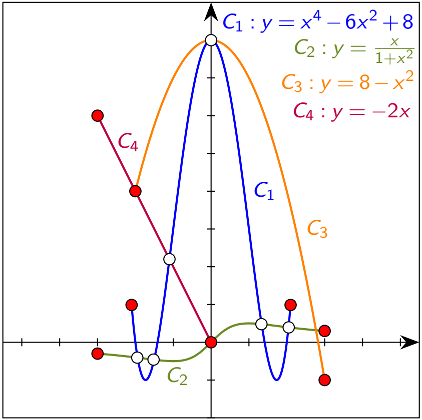
Traits Model: Arcs of Rational Functions (Cont.)

- A rational arc is always x-monotone.
- A rational arc is not necessarily continuous.

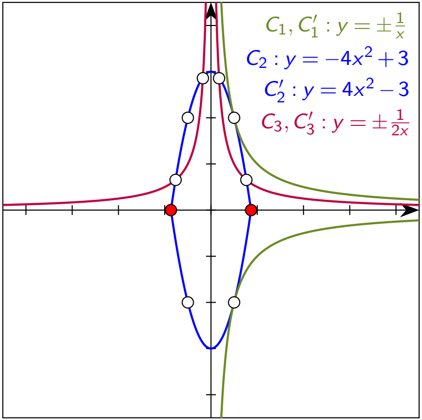
- $y = \frac{1}{(x-1)(2-x)}$ defined over the interval $[0,3]$.
  - Has two singularities at $x = 1$ and at $x = 2$.
  - Is subdivided by `Make_x_monotone_2` into 3 continuous portions defined over the intervals $[0,1)$, $(1,2)$, and $(2,3]$, respectively.



Traits Model: Arcs of Rational Functions (Cont.)



An arrangement of 4 bounded rational arcs (`ex_rational_functions.cpp`).



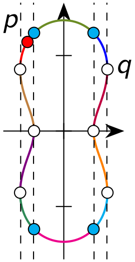
An arrangement of 6 unbounded arcs of rational functions (`ex_unbounded_rational_functions.cpp`

Traits Model: Algebraic Curves

- `Arr_algebraic_segment_traits_2<Coefficient>`

Definition (Algebraic curve)

An algebraic curve is the (real) zero set of a bivariate polynomial $f(x,y)$.



- Supports
  - algebraic curves and
  - continuous x-monotone segments of algebraic curves, which are not necessarily maximal.
  - Non x-monotone segments are not supported.
  - x-monotone segments are not necessarily maximal.
- An Oval of Cassini, $(y^2+x^2+1)^2-4y^2=4/3$.
  - The induced arrangement consists of 2 faces, 10 edges, and 10 vertices.

Arrangement Geometry Traits Models

① — *ArrangementLandmarkTraits_2*
② — *ArrangementTraits_2*
③ — *ArrangementDirectionalXMonotoneTraits_2*
④ — *ArrangementOpenBoundaryTraits_2*.

Model Name	Curve Family	Degree	Concepts	
Arr_non_caching_segment_basic_traits_2	line segments	1	①	
Arr_non_caching_segment_traits_2	line segments	1	①,②,③	
Arr_segment_traits_2	line segments	1	①,②,③	
Arr_linear_traits_2	line segments, rays, and lines	1	①,②,③,④	
Arr_circle_segment_traits_2	line segments and circular arcs	≤ 2	②,③	
Arr_circular_line_arc_traits_2	line segments and circular arcs	≤ 2	②	
Arr_conic_traits_2	circles, ellipses, and conic arcs,	≤ 2	①,②,③	
Arr_rational_function_traits_2	curves of rational functions	≤ 2	①,②,③,④	
Arr_Bezier_curve_traits_2	Bézier curves	$\leq n$	②,③	
Arr_algebraic_segment_traits_2	algebraic curves	$\leq n$	②,③,④	
Arr_polyline_traits_2	polylines	∞	①,②,③	
Arr_polycurve_traits_2	polycurves	∞	①,②,③	
Arr_geodesic_arcs_traits_2	geodesic arcs on sphere	1	①,②,③	

Outline

The Notification Mechanism

Definition (Observer)

An observer defines a one-to-many dependency between objects, so that when one object changes state, all its dependents are notified and updated automatically.

- The *2D Arrangements* package offers a mechanism that uses *observers*.
- The observed type is derived from an instance of `Arr_observer<Arrangement>`.
- The observed object does not know anything about the observers.
- Each arrangement object stores a list of pointers to `Arr_observer` objects.
- The trapezoidal-RIC and the landmark point-location strategies use observers to keep their auxiliary data-structures up-to-date.

Observer Notification Functions

The set of functions can be divided into 3 categories:

1. Notifiers on changes that affect the topological structure of the arrangement. There are 2 pairs (*before* and *after*) that notify when
   - the arrangement is cleared or
   - the arrangement is assigned with the contents of another one.
2. Pairs of notifiers before and after of a local change that occurs in the topological structure.
   - A new vertex is constructed or deleted.
   - An new edge is constructed or deleted.
   - 1 edge is split into 2 edges, or 2 are merged into 1.
   - 1 face is split into 2 faces, or 2 are merged into 1.
   - 1 hole is created in the interior of a face or removed from it.
   - 2 holes are merged into 1, or 1 is split into 2.
   - A hole is moved from one face to another.
3. Notifiers on a structural change caused by a free function. A single pair `before_global_change()` and `after_global_change()`.

Extending all the DCEL Records

- An instance of

  `Arr_extended_dcel<Traits, VertexData, HalfedgeData, FaceData>`

  is a DCEL that extends the vertex, halfedge, and face records with the corresponding types.

```
enum Color {BLUE, RED, WHITE};

typedef CGAL::Arr_extended_dcel<Traits_2, Color, bool, unsigned int>    Dcel;
typedef CGAL::Arrangement_2<Traits_2, Dcel>                              Ex_arrangement_2;
```

```
  Ex_arrangement_2::Vertex_iterator          vit;
  for (vit = arr.vertices_begin(); vit != arr.vertices_end(); ++vit) {
    unsigned int degree = vit->degree();
    vit->set_data((degree == 0) ? BLUE : ((degree <= 2) ? RED : WHITE));
  }
```

```
  std::cout << "The␣arrangement␣vertices:" << std::endl;
  for (vit = arr.vertices_begin(); vit != arr.vertices_end(); ++vit) {
    std::cout << '(' << vit->point() << ")␣-␣";
    switch (vit->data()) {
      case BLUE  : std::cout << "BLUE."  << std::endl; break;
      case RED   : std::cout << "RED."   << std::endl; break;
      case WHITE : std::cout << "WHITE." << std::endl; break;
    }
  }
```

Extending all the DCEL Records (Cont.)

```
#include <string>

#include <CGAL/basic.h>
#include <CGAL/Arr_dcel_base.h>

/*! The map extended dcel vertex */
template <typename Point_2>
class Arr_map_vertex : public CGAL::Arr_vertex_base<Point_2> {
public:
  std::string name, type;
};

/*! The map extended dcel halfedge */
template <typename X_monotone_curve_2>
class Arr_map_halfedge : public CGAL::Arr_halfedge_base<X_monotone_curve_2> {
public:
  std::string name, type;
};

/*! The map extended dcel face */
class Arr_map_face : public CGAL::Arr_face_base {
public:
  std::string name, type;
};

/*! The map extended dcel */
template <typename Traits>
class Arr_map_dcel : public
  CGAL::Arr_dcel_base<Arr_map_vertex<typename Traits::Point_2>,
                      Arr_map_halfedge<typename Traits::X_monotone_curve_2>,
                      Arr_map_face>
{};
```

Outline

Outline

Outline

Outline

Arrangement Bibliography I

Boris Aronov and Dmitriy Drusvyatskiy
Complexity of a Single Face in an Arrangement of s-Intersecting Curves
arXiv:1108.4336, 2011

Jon Louis Bentley and Thomas Ottmann.
Algorithms for Reporting and Counting Geometric Intersections.
IEEE Transactions on Computers, 28(9): 643–647, 1979.

Eric Berberich, Efi Fogel, Dan Halperin, Michael Kerber, and Ophir Setter.
Arrangements on parametric surfaces ii: Concretizations and applications, 2009.
Mathematics in Computer Science, 4(1):67–91,2010.

Ulrich Finke and Klaus H. Hinrichs.
Overlaying simply connected planar subdivisions in linear time.
In *Proceedings of 11th Annual ACM Symposium on Computational Geometry (SoCG)*, pages 119–126. Association for Computing Machinery (ACM) Press, 1995.

Ron Wein, Efi Fogel, Baruch Zukerman, Dan Halperin, and Eric Berberich.
2D Arrangements.
In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 4.4 edition, 2014.
http://www.cgal.org/Manual/latest/doc_html/cgal_manual/packages.html#Pkg:Arrangement2.

David G. Kirkpatrick.
Optimal search in planar subdivisions.
SIAM Journal on Computing. 12(1):28–35,1983.

N. Sarnak and Robert E. Tarjan.
Planar point location using persistent search trees.
Communications of the ACM. 29(7):669–679, 1986.

Kentan Mulmuley.
A fast planar partition algorithm, I.
Journal of Symbolic Computation. 10(3-4):253–280,1990.

Arrangement Bibliography II

Raimund Seidel
A Simple and Fast Incremental Randomized Algorithm for Computing Trapezoidal Decompositions and for Triangulating Polygons.
Computational Geometry: Theory and Applications. 1(1):51–64, 1991.

Olivier Devillers, Sylvain Pion, and Monique Teillaud.
Walking in a triangulation.
International Journal of Foundations of Computer Science. 13:181–199,2002.

Luc Devroye Christophe, Christophe Lemaire, and Jean-Michel Moreau.
Fast Delaunay Point-Location with Search Structures.
In *Proceedings of* 11^{th} *Canadian Conference on Computational Geometry*. Pages 136–141, 1999.

Luc Devroye, Ernst Peter Mücke, and Binhai Zhu.
A Note on Point Location in Delaunay Triangulations of Random Points.
Algorithmica. 22:477–482, 1998.

Olivier Devillers.
The Delaunay hierarchy.
International Journal of Foundations of Computer Science. 13:163-180, 2002.

Sunil Arya
A Simple Entropy-Based Algorithm for Planar Point Location.
ACM Transactions on Graphics. 3(2), 2007

Masato Edahiro, Iwao Kokubo, And Takao Asano
A new Point-Location Algorithm and its Practical Efficiency: comparison with existing algorithms
ACM Transactions on Graphics. 3(2):86–109, 1984.

Micha Sharir and Pankaj Kumar Agarwal
Davenport-Schinzel Sequences and Their Geometric Applications.
Cambridge University Press, New York, NY, 1995.

Arrangement Bibliography III

Bernard Chazelle, Leonidas J. Guibas, and Der-Tsai Le.
The Power of Geometric Duality.
BIT, 25:76–90, 1985.

Herbert Edelsbrunner,
Algorithms in Combinatorial Geometry,
Springer, Heidelberg, 1987.

Mark de Berg, Mark van Kreveld, Mark H. Overmars, and Otfried Cheong.
Computational Geometry: Algorithms and Applications.
Springer, 3^{rd} edition, 2008.

Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir.
On the Zone Theorem for Hyperplane Arrangements.
SIAM Journal on Computing. 22(2):418–429,1993.

Silvio Micali and Vijay V. Vazirani.
An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs.
Proceedings of 21^{st} *Annual IEEE Symposium on the Foundations of Computer Science*, pages 17–27, 1980.

Jack Edmonds.
Paths, Trees, and Flowers.
Canadian Journal of Mathematics, 17:449–467,1965.

Robert Endre Tarjan.
Data structures and network algorithms, Society for Industrial and Applied Mathematics (SIAM), 1983.

Marcin Mucha and Piotr Sankowski.
Maximum Matchings via Gaussian Elimination
Proceedings of 45^{th} *Annual IEEE Symposium on the Foundations of Computer Science*, pages 248–255, 2004.

Arrangement Bibliography IV

Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine.
The Boost *Graph Library*.
Addison-Wesley, 2002

Efi Fogel, Ron Wein, and Dan Halperin.
Cgal *Arrangements and Their Applications, A Step-by-Step Guide*.
Springer, 2012.