Arrangements in CGAL — More Generic than Ever!

Efi Fogel, Ron Wein, Danny Halperin, Idit Haran, and Niv Sabath

{efif,wein,danha,haranidi,sabath}@post.tau.ac.il

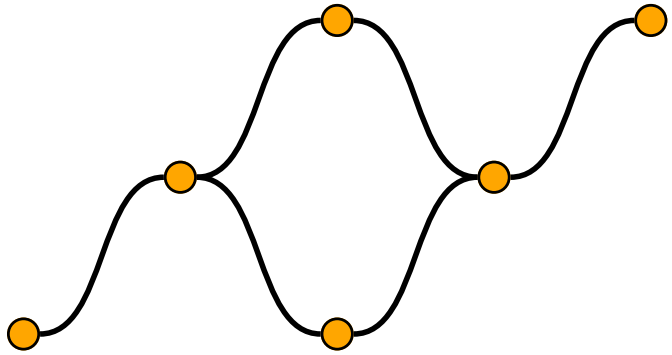
School of computer science, Tel Aviv University

Outline

- Introduction
- Innovations
  - Traits requirements and tags
  - Polyline traits
  - Data traits
- Summary

Planar Maps

Planar graphs that are embedded in the plane

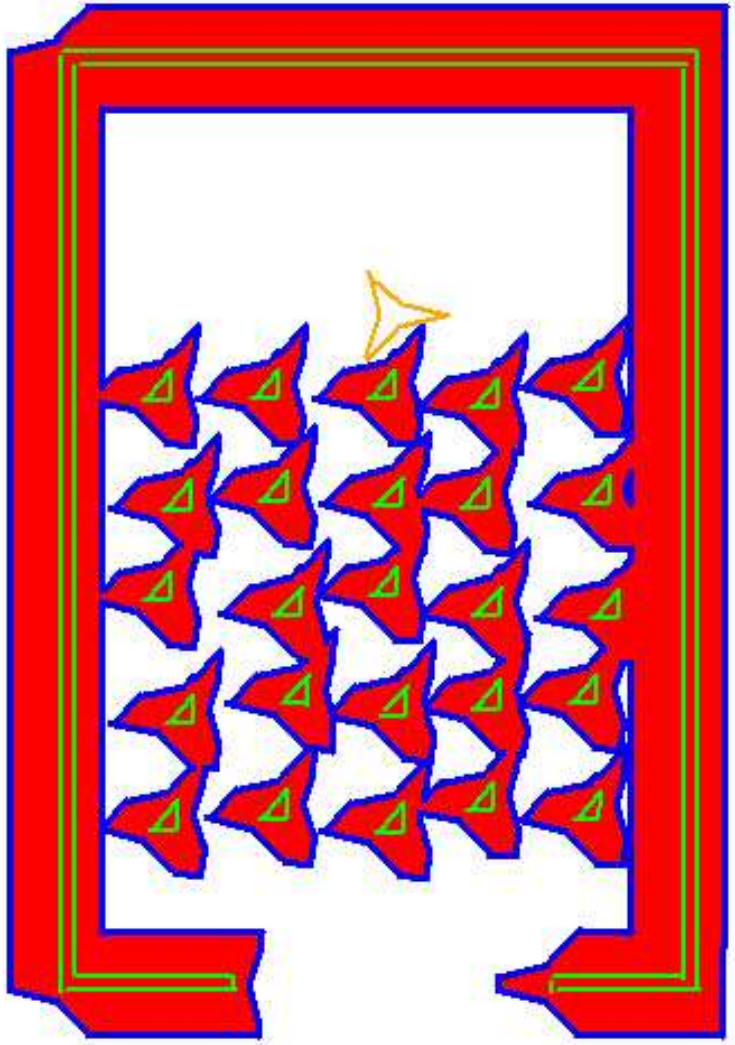


Planar Arrangements

Given a collection Γ of planar curves, the arrangement $\mathcal{A}(\Gamma)$ is the partition of the plane into vertices, edges and faces induced by the curves of Γ

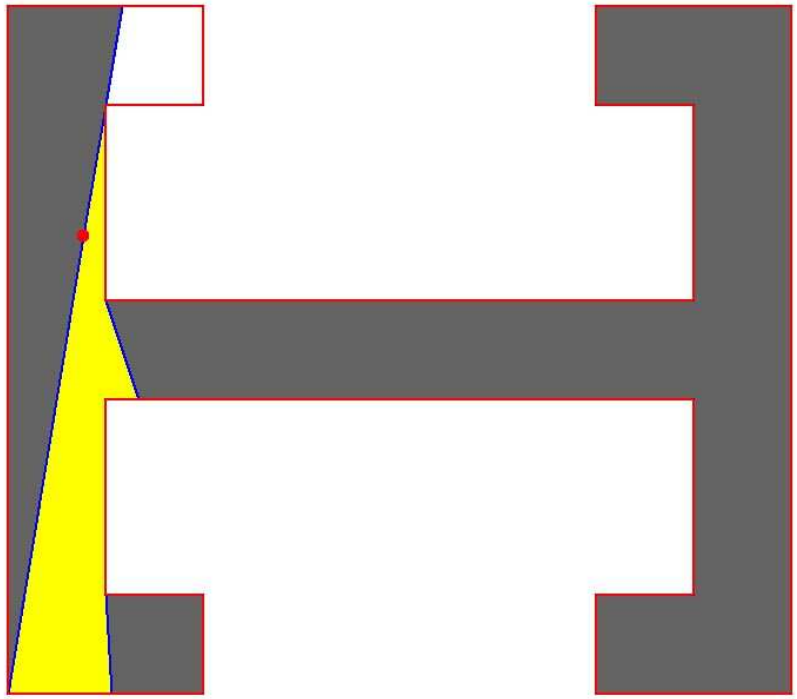
Application — Robot Motion Planning

[Flato, Halperin]



Application — Pursuit-Evasion

[Brian P. Gerkey]

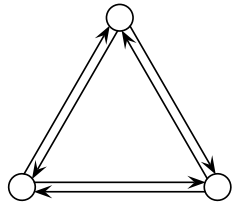


The Package in Brief

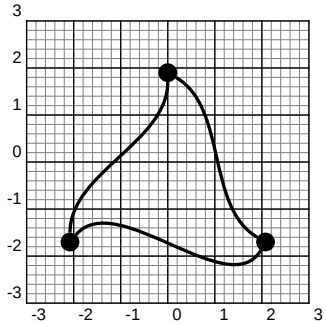
- Goal: construct, maintain, modify, traverse, query and present subdivisions of the plane
- Robust
- Generic
- Handles all degeneracies
- Efficient
- Easy to interface

Hierarchy

- Topological_map
  - Maintains topological maps of finite edges



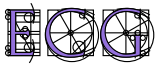
- Planar_map_2
  - Maintains planar maps of interior-disjoint x-monotone curves



- Planar_map_with_intersections_2
  - Maintains planar maps of general curves (may intersect, may be non-x-monotone)
- Arrangement_2
  - Maintains planar maps of intersecting curves along with curve history

Planar Map Functionality

- Creation and destruction
- I/O - selective
  - Save, load, print (ASCII streams)
  - Draw (graphic streams)
- Modification
  - Insertion, removal, split, merge
- Traversal
- Queries
  - Number of vertices, halfedges, and faces
  - Is point in face
  - Point location - various strategies
  - Vertical ray shooting

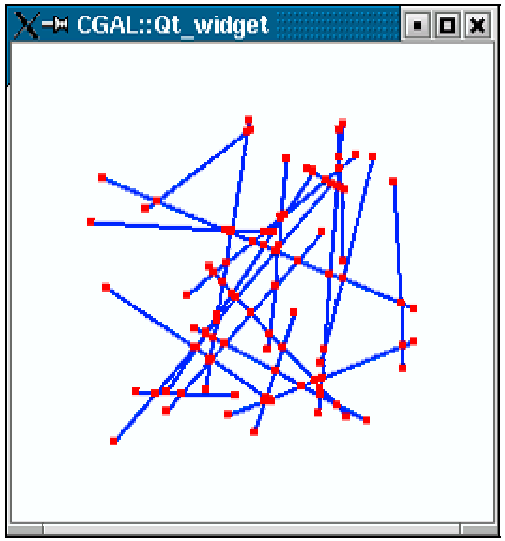


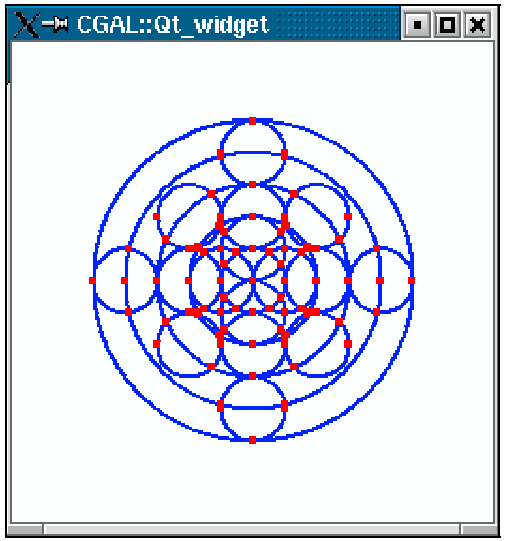
Planar Map Traits

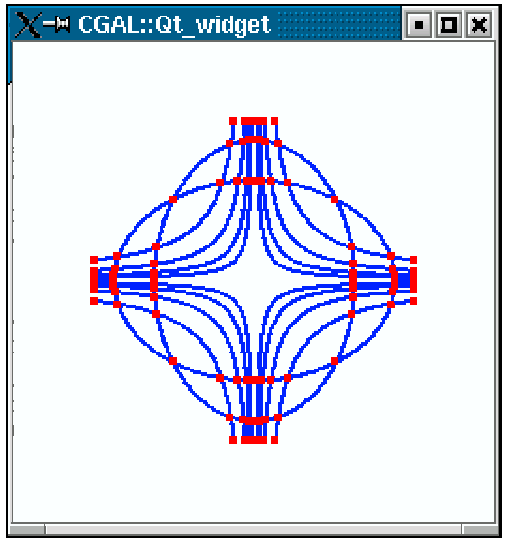
- Geometric interface
- Parameter of package
  - Defines the family of curves in interest
  - Package can be used with any family of curves for which a traits class is supplied
- Aggregate
  - Geometric types (points, curves)
  - Operations over types (accessors, predicates, constructors)

Planar Map Supplied Traits

- Segments
  - Standard
  - Cached – contains the underline line and flags
- Polylines
- Conic arcs (segments of circles, ellipses, parabolas, hyperbolas, and lines)







Planar Map Other Traits

- Circular arcs (Iddo Hanniel)
- Canonical parabola
- Bezier curves - Addaptive Approximation by Polygons
- Conics (Eric Berberich)
- Circular arcs (Monique Teillaud and Sylvain Pion)
- Cubix (Arno Eigenwillig ?)

Planar Map Insertion Operations

- Incremental insert
- Aggregate insert
- Often information is known in advance
  - Containing face
    - Insert in face interior
  - Incident vertices
    - Insert from vertex, between vertices
  - Order around vertex
    - Insert from halfedge target, between halfedge targets

Planar map Traits

`PlanarMapTraits_2` concept requirements

- Types
  - **`X_monotone_curve_2`**
  - **`Point_2`**
- Methods
  1. Compare two points by their x-coordinates or lexicographically
  2. Determine the relative position of a point an a curve
  3. Determine the relative position of two curves at x.
  4. Determine the relative position of two curves to the right or to the left (lexicographically) of a point.

Planar map with intersections Traits

`PlanarMapWithIntersectionsTraits_2` concept

- Types
  - **`Curve_2`**
- Methods
  1. Subdivide a curve into x-monotone curves
  2. Find the intersection point of the two curves to the right or to the left (lexicographically) of a point.
  3. Determine the overlapping subcurve of two curves

Tight and Flexible Traits

- The requirements are designed to be as tight as possible
- Comparing and finding the next intersection from the left
  - are not required when constructing a planar map using the aggregate insert
    - The plane is swept from left to right
  - can be replaced by two other simple functions
    - A pair of functions that reflect a point or a curve about the origin
  - can be dropped all together, and implemented by the other methods
    - perhaps less inefficiently

Using Tags

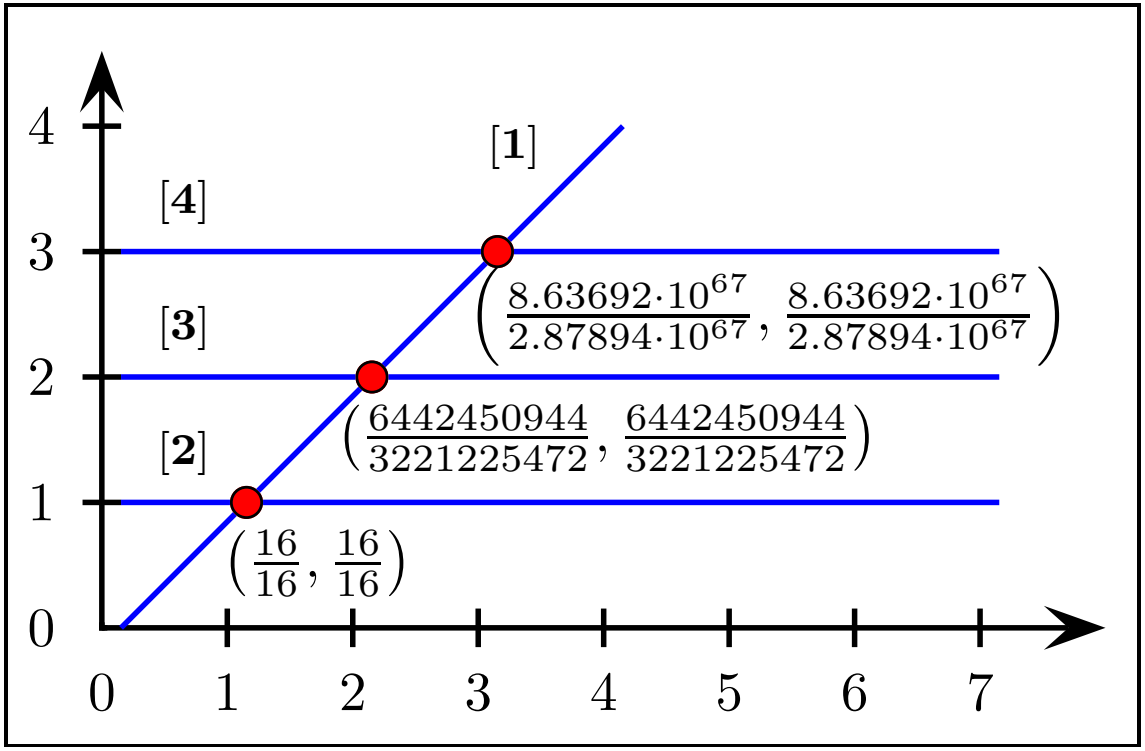
- A traits-class wrapper implements a wider set of methods
- Uses the methods of the wrapped traits-class
- Uses tags to identify the existing methods

  `Has_left_category` —indicates whether the functions that operate “from the right” are implemented

  `Has_reflect_category` —indicates whether the functions that reflect a point or a curve about the origin are implemented

Regular Segment Traits
`Arr_segment_traits_2`

- Parameterized with geometric kernel
- A segment is represented by its 2 endpoints
- Leads to cascaded rep. of intersection points
- Normalization could be expensive



Cached Segment Traits
`Arr_segment_cached_traits_2`

- Parameterized with geometric kernel
- A segment is represented by its 2 endpoints and its underlying line
  - When a segment is split, the underlying line remains the same
  - The underlying line is used to compute intersections
- Faster than the regular traits
  - Especially when constructing an arrangment with many intersections

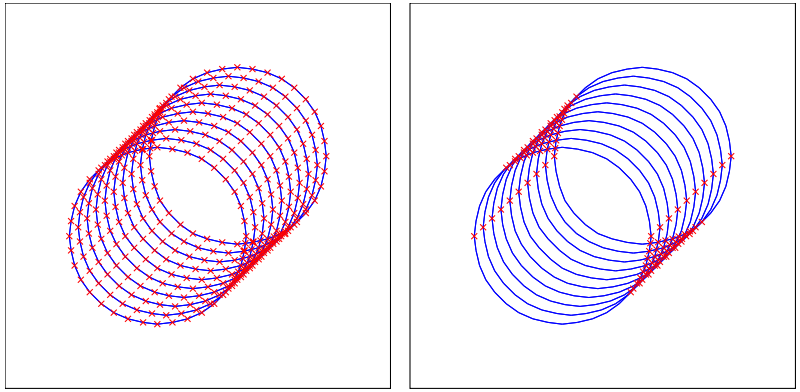
Planar map of Segments Type-Defi nition

```
typedef CGAL::Quotient<CGAL::MP_Float>              NT;
typedef CGAL::Cartesian<NT>                         Kernel;
typedef CGAL::Arr_segment_cached_traits_2<Kernel> Traits;


typedef CGAL::Pm_default_dcel<Traits>               Dcel;
typedef CGAL::Planar_map_2<Dcel,Traits>             Pm;
typedef CGAL::Planar_map_with_intersections_2<Pm> Pmwx;
```

Polyline Traits

- Planar map of piecewise linear segments (polylines)
  - Approximate curves of high degree
  - Useful for testing
    - not necessarily x monotone
    - (Exact) rational number type is sufficient
- Parameterized with a segment traits
- Maintains a `Seg_traits::Curve_2` segment vector
- Underlying segment traits performs geometric ops.



Planar map of Polylines Type-Defi nition

```
typedef CGAL::Quotient<CGAL::MP_Float>             NT;
typedef CGAL::Cartesian<NT>                        Kernel;
typedef CGAL::Arr_segment_cached_traits_2<Kernel> Seg_traits;
typedef CGAL::Arr_polyline_traits_2<Seg_traits>   Traits;

typedef CGAL::Pm_default_dcel<Traits>              Dcel;
typedef CGAL::Planar_map_2<Dcel,Traits>            Pm;
typedef CGAL::Planar_map_with_intersections_2<Pm> Pmwx;
```

Data Traits

- A meta traits
- Parameterized with:
  - A traits class - a model of of the concept **`PlanarMapWithIntersectionsTraits_2`**
  - A data class - contains extraneous data associated with a curve
- The Data is passed from curves to subcurves when
  - A curve is divided into x-monotone curves
  - A curve is split at an intersection point

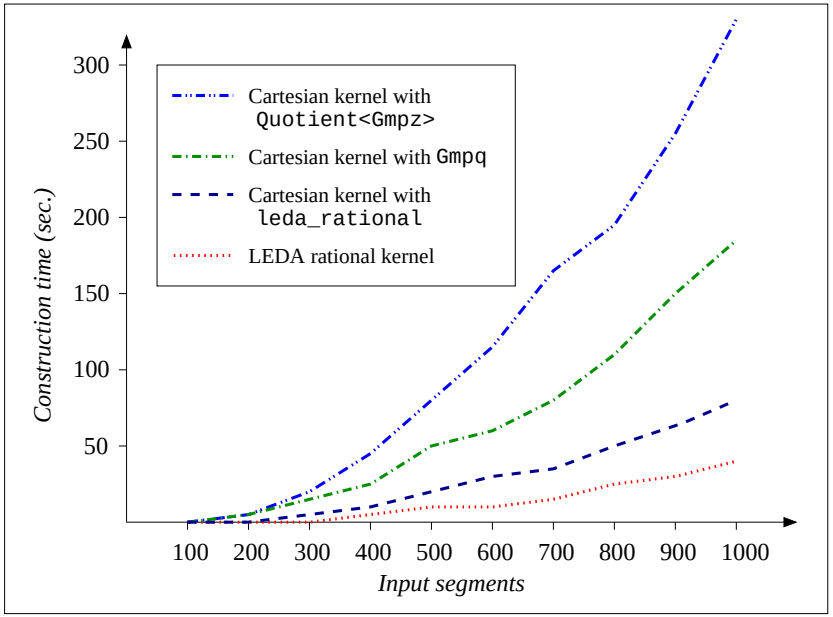
Applications of Data Traits

- An alternative to the `Arrangement_2` class
  - Each curve contains a pointer to the curve it originated from
- An entire hierarchy of planar maps
  - Each x-monotone curve contains a pointer to an x-monotone curve in the planar map above in the hierarchy
- An overlay of rivers and roads

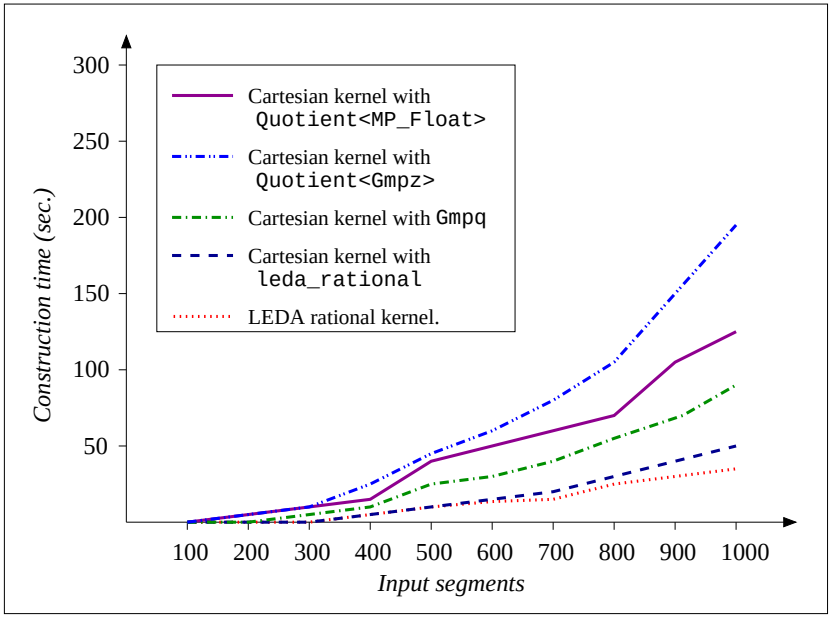
Planar map of Polylines width Data Type-Defi nition

```
struct Data {
  enum {ROAD, RIVER};
  std::string name;
};
typedef CGAL::Quotient<CGAL::MP_Float>                    NT;
typedef CGAL::Cartesian<NT>                               Kernel;
typedef CGAL::Arr_segment_cached_traits_2<Kernel>         Seg_traits;
typedef CGAL::Arr_polyline_traits_2<Seg_traits>           Bas_traits;
typedef CGAL::Arr_curve_data_traits_2<Bas_traits,Data>    Traits;
typedef CGAL::Pm_default_dcel<Traits>                     Dcel;
typedef CGAL::Planar_map_2<Dcel,Traits>                   Pm;
typedef CGAL::Planar_map_with_intersections_2<Pm>         Pmwx;
```

Experimental Results



Using regular traits



Using cached traits

- Using cached traits is more efficient
- It is possible to use `Quotient<MP_float>` with the cached traits