Teil II: Räumliche DBS

1. Einführung in Modellierung räumlicher Datentypen

Spatial database systems and Geographic Information Systems as their most important application aim at storing, retrieving, manipulating, querying, and analysing *geometric data*. Research has shown that special data types are necessary to model geometry and to suitably represent geometric data in database systems. These data types are usually called *spatial data types*, such as *point*, *line*, and *region* but also include more complex types like *partitions* and *graphs* (*networks*). Spatial data types provide a fundamental abstraction for modeling the geometric structure of objects in space, their relationships, properties and operations. Their definition is to a large degree responsible for a successful design of spatial data models and the performance of spatial database systems and exerts a great influence on the expressive power of spatial query languages. This is true regardless of whether a DBMS uses a relational, complex object, object-oriented, or some other kind of data model. Hence, the definition and implementation of spatial data types is probably the most fundamental issue in the development of spatial DBMS. Consequently, their understanding is a prerequisite for an effective construction of important components of a spatial database system (like spatial index structures, optimizers for spatial data, spatial query languages, storage management, and graphical user interfaces) and for a cooperation with extensible DBMS providing spatial type extension packages (like spatial data blades and cartridges).

[aus Tutorial von M.Schneider, Fernuni Hagen: *Spatial Data Types: Conceptual Foundation for the Design and Implementation of Spatial Database Systems and GIS*]

1.1 What are Spatial Data Types (SDTs)?

Spatial data types

- ... are special data types needed to model geometry and to suitably represent geometric data in database systems
- Examples: *point*, *line*, *region*; *partitions* (*maps*), *graphs* (*networks*)
- ... provide a *fundamental abstraction* for modeling the geometric structure of *objects* in space, their relationships, properties, and operations
- ... are an important part of the data model and the implementation of a spatial DBMS

The definition of SDTs

- ... is to a large degree responsible for a successful design of spatial data models
- ... decisively affects the performance of spatial database systems
- ... exerts a great influence on the expressiveness of spatial query languages
- ... should be independent from the data model used by a DBMS

Implications

- An understanding of SDTs is a prerequisite
  - for an effective construction of important components of a spatial database system
    → spatial index structures, optimizers for spatial data, spatial query languages, storage management, graphical user interfaces
  - for a cooperation with extensible DBMS providing spatial type extension packages
    → spatial data blades, cartridges

1.2 What Needs to be Represented?

Two views of spatial phenomena:

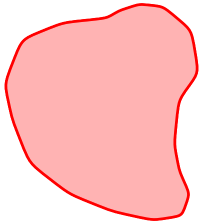
- objects in space (entity-oriented / feature-based view)

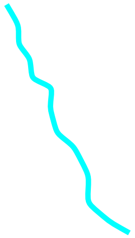
  → vector data, spatial database systems

- space itself (space-oriented / position-based view)

  → raster data, image database systems

Objects in space

city Berlin, pop = 4000000, ..., area = 

highway A45, ..., route = 

Space

Statement about every point in space

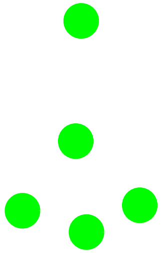
- land use maps (“thematic maps”)
- partitions into states, districts, ...

We consider:

- modeling single, self-contained objects
- modeling spatially related collections of objects

Fundamental abstractions for modeling single, self-contained objects

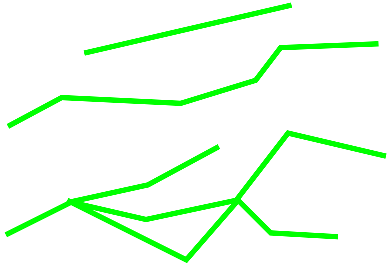
point



city
castle
lighthouse
church

(location of object in space but not its extent relevant)

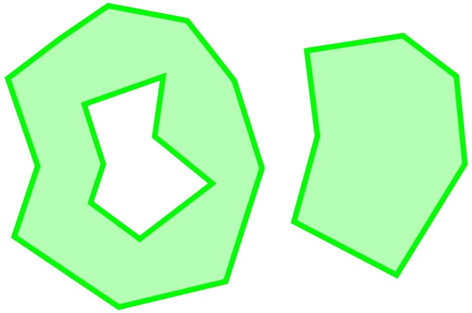
line



highway
river
cable
route

(connections in space, movement through space)

region

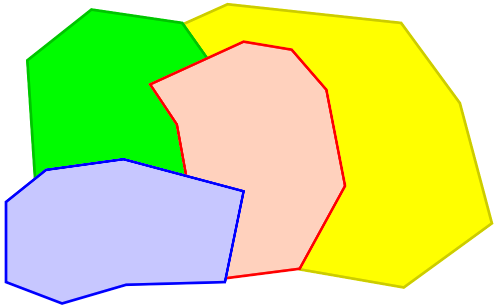


city
lake
district
forest
cornfield

(extent of an object relevant)

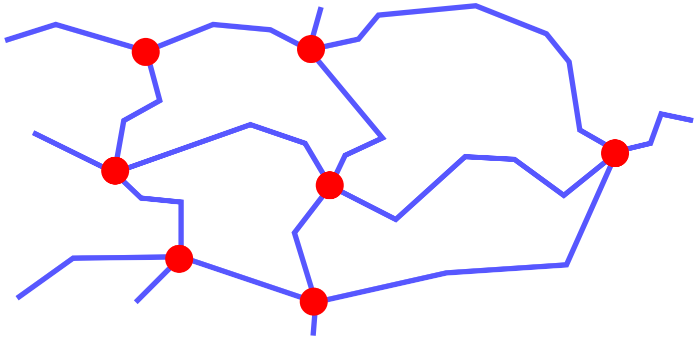
Fundamental abstractions for modeling spatially related collections of objects

partition



land use
districts
wards
countries
speech areas

Spatially embedded *network* (graph)



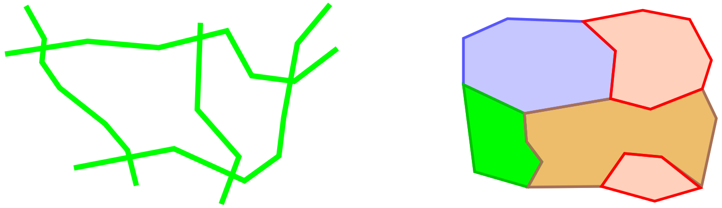
highways
railways
rivers
electricity
phone

Others: nested partitions, digital terrain models

1.3 A Three-Level Model for Phenomena in Space

Structure modeling

Structure objects

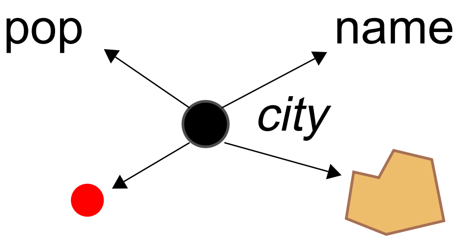


Structure types: sets, sequences, partitions, networks

Operations: overlay, shortest_path

Object modeling

Spatially-referenced objects

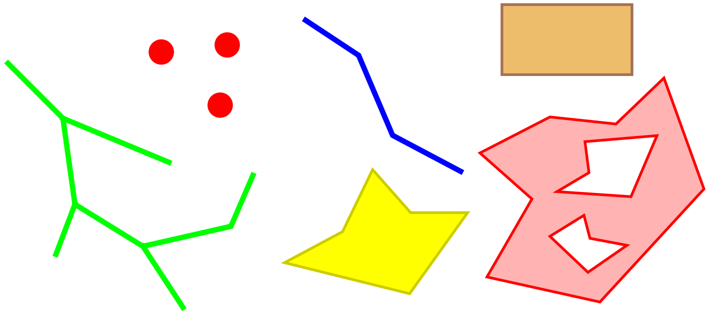


Object types: city, state, river

Operations: lies_in: city → state, flow: river → line

Spatial modeling

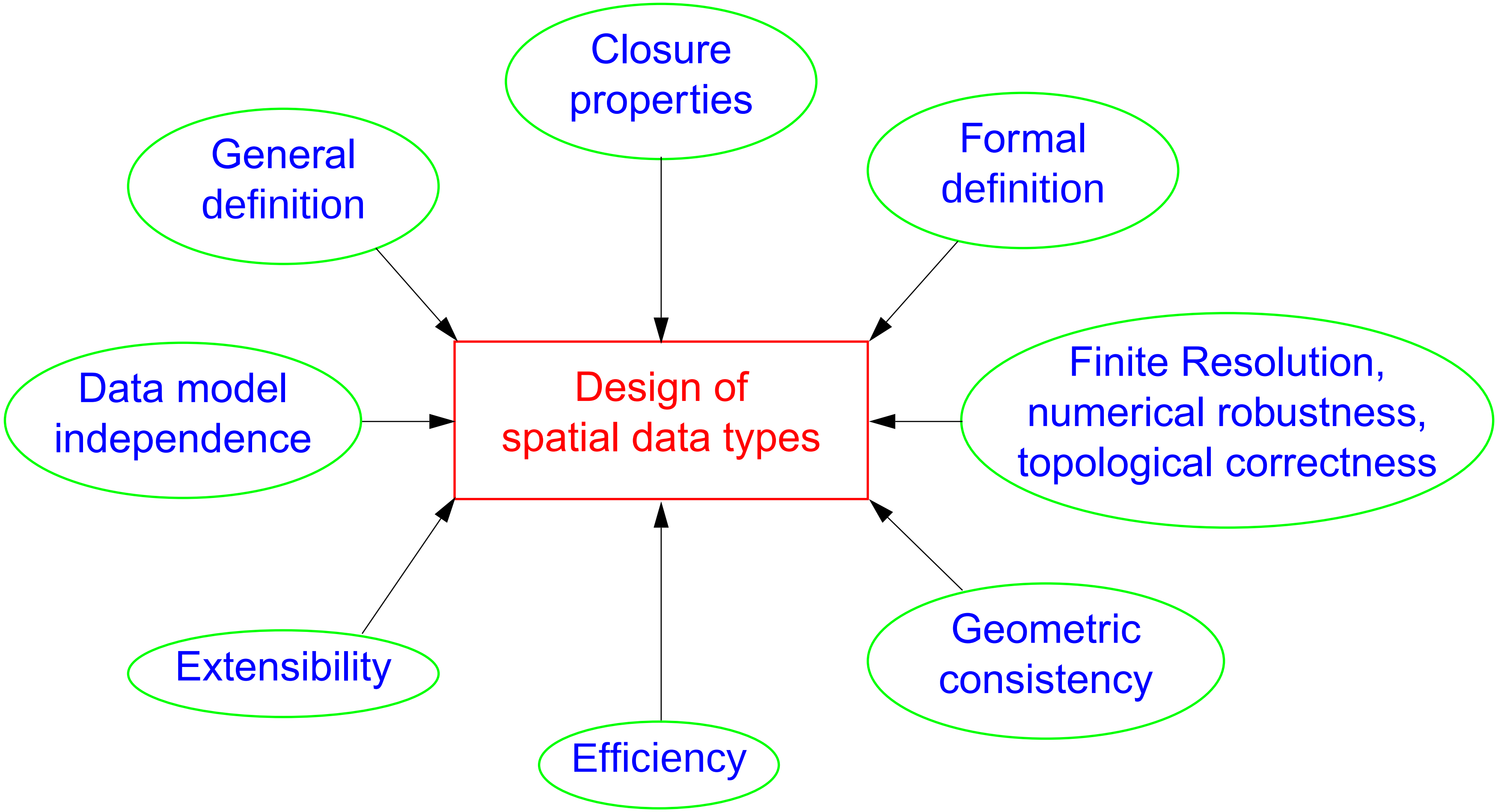
Spatial objects



Spatial data types: point, line, region

Operations: point-in-polygon test, intersection

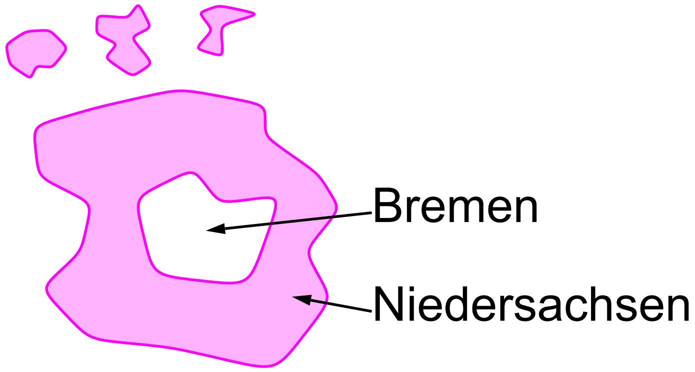
1.4 Design Criteria for Modeling Spatial Data Types



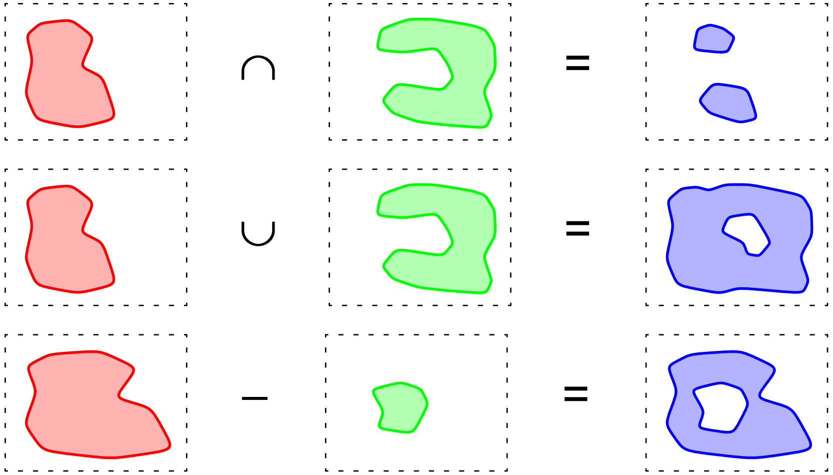
1.5 Closure Properties and Geometric Consistency

General definition/structure of spatial objects

application-driven requirements



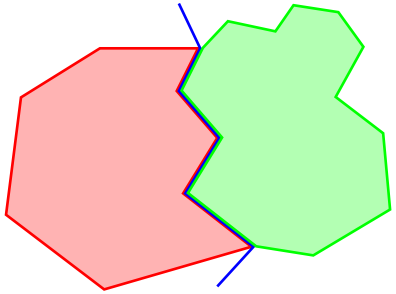
formal requirements



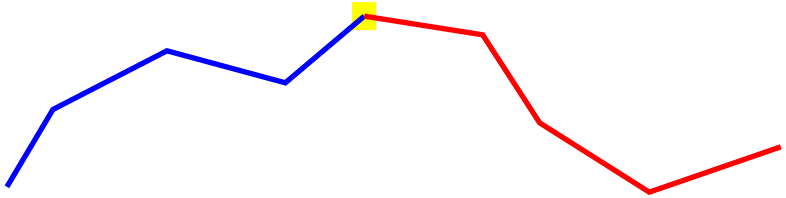
→ spatial objects must be closed under set operations on the underlying point sets

Support of geometric consistency constraints for spatially related objects

adjacent regions



meeting lines



→ SDT definition must offer facilities to enforce such consistency contraints

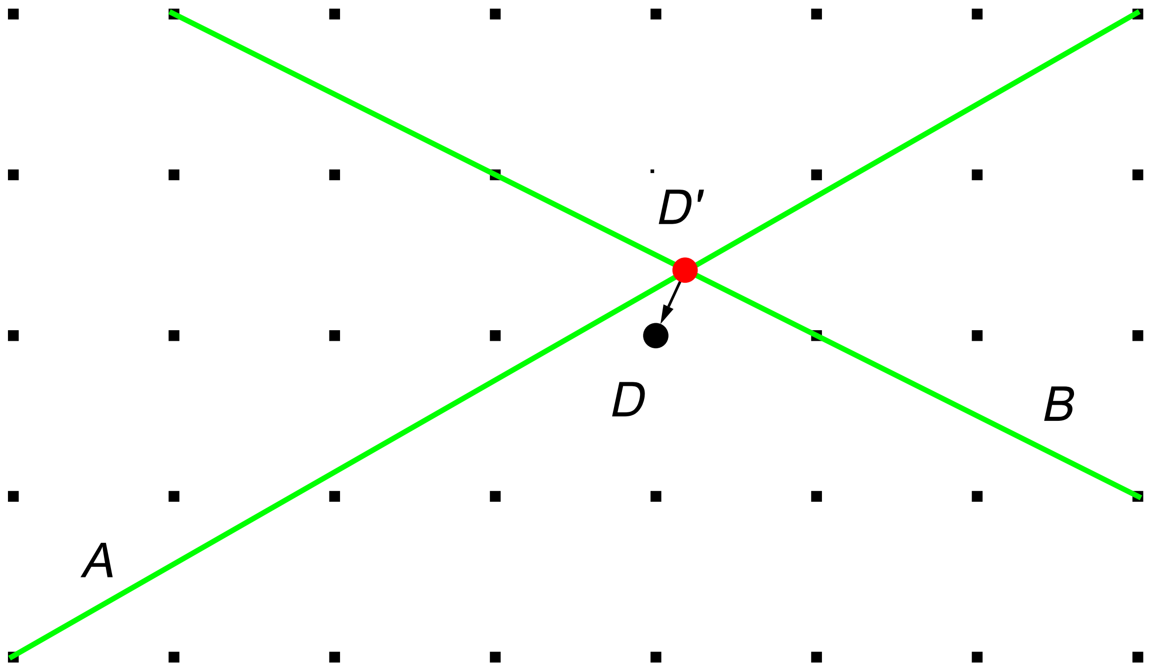
1.6 Organizing the Underlying Space: Euclidean Geometry versus Discrete Geometric Bases

Euclidean space is continuous ($p = (x, y) \in \mathrm{IR}^2$)

- basis of Computational Geometry algorithms

But: computer numbers are finite and discrete ($p = (x, y) \in \mathtt{real} \times \mathtt{real}$)

- → numerical rounding errors → topological inconsistencies and degeneracies



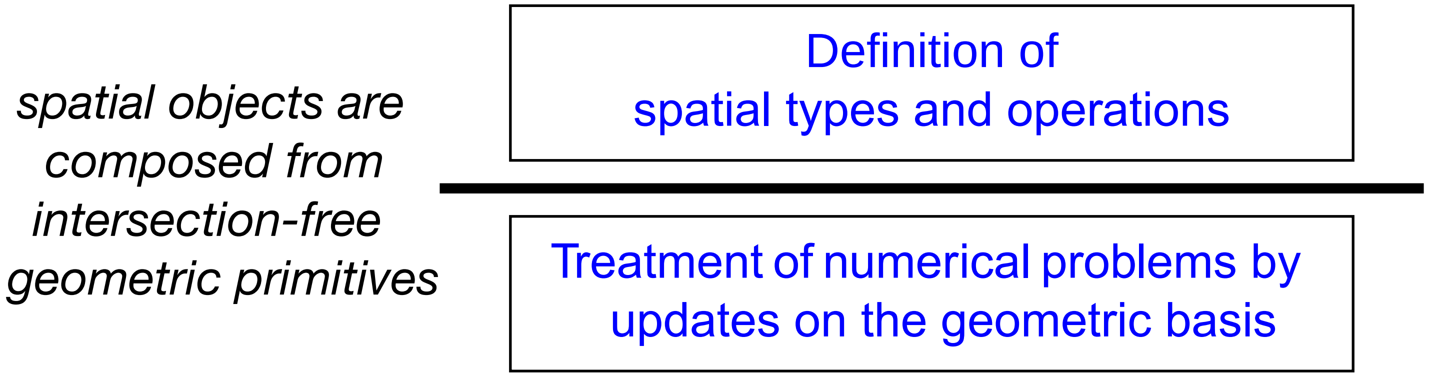
Is *D* *on* *A*?

Is *D* *properly contained* in the area below *A* and *B*?

What happens if there is a segment *C* between *D* and *D'*?

→ formal SDT definitions must bear in mind the finite representations available in computers

Solution: avoid computation of any new intersections within geometric operations



spatial objects are composed from intersection-free geometric primitives

Definition of spatial types and operations

Treatment of numerical problems by updates on the geometric basis

Two approaches for a geometric basis:

- *Simplicial complexes*

  Frank & Kuhn 1986

  Egenhofer, Frank & Jackson 1989

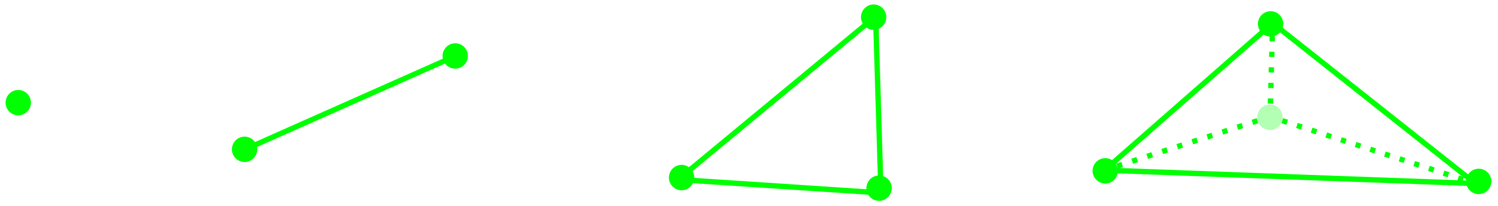
- *Realms*

  Güting & Schneider 1993

  Schneider 1997

Simplicial Complexes

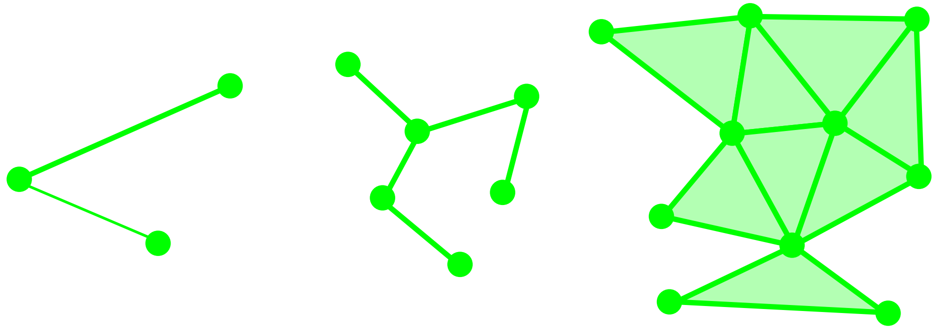
- topological relations are separately recorded and independent of metric positions
- use of *k-simplices* for representing minimal spatial objects of dimension *k*
  - construction rule: *k*-simplex consists of *k*+1 simplices of dimension *k*–1
  - component of a simplex is called *face*



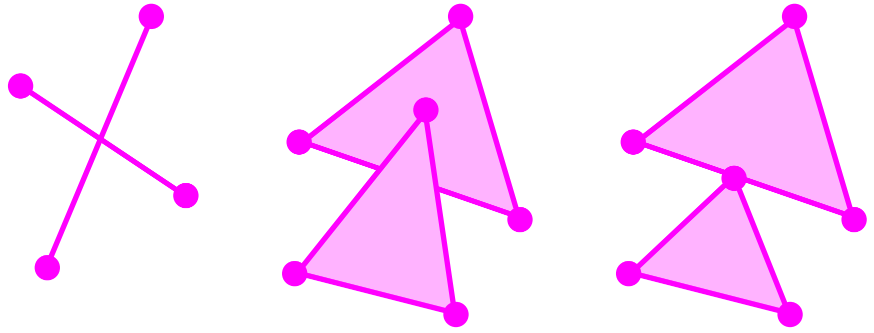
0-simplex 1-simplex 2-simplex 3-simplex etc.

- two completeness principles:
  - completeness of intersection: the intersection of two *k*-simplices is either empty or a face of both simplices
    → no line intersection at points which are not start or end points of the lines, no two geometric objects may exist at the same location (geometry only recorded once)
  - completeness of inclusion: every *k*-simplex is a face of a (k+1)-simplex
    → all points are end points of lines, all lines are boundaries of triangles, etc.; no isolated points, no lines which are not part of a boundary

- *Simplicial complex*: finite set of simplices such that the intersection of any two simplices is empty or a face



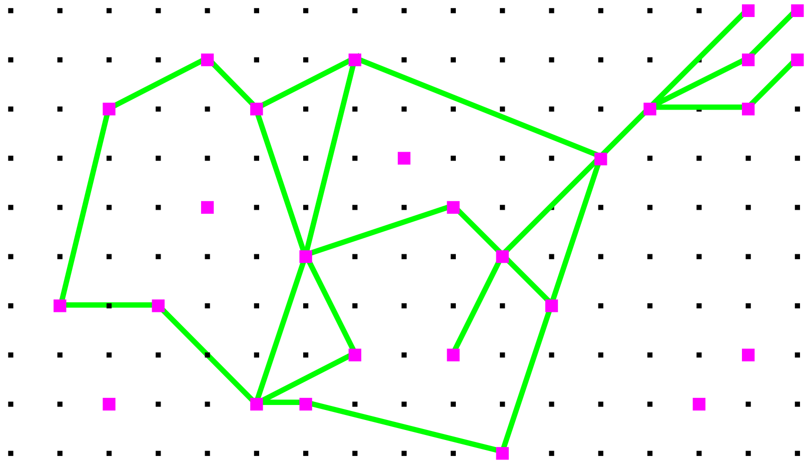
correct simplicial complexes

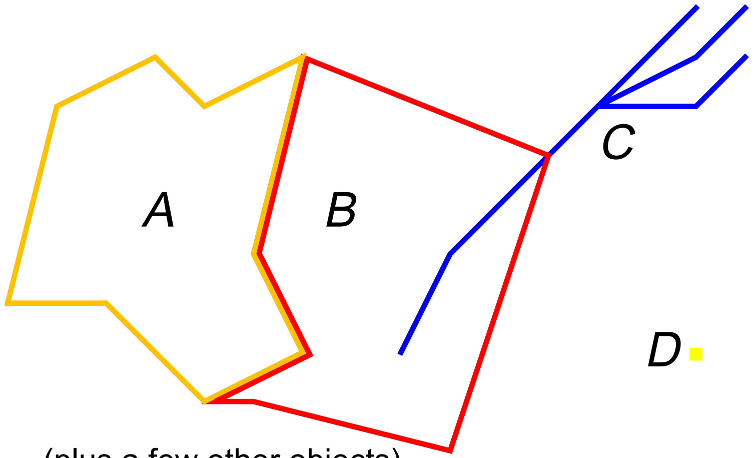


no simplicial complexes

- Advantages
  - maintenance of topological consistency
  - approach fulfills closure properties
- Drawbacks
  - unfortunately: no spatial operations have been defined on top of this approach
  - triangulation of space would lead to space-consuming representations of spatial objects (worse than boundary polygons)
  - no treatment of numerical problems: additional data structures needed to realize metric operations

Realms



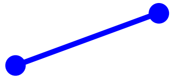


(plus a few other objects)

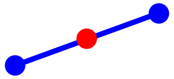
Realm (intuitive notion): description of the complete underlying geometry (all points and lines) of an application or application space

Realm (formally): A finite set of points and *non-intersecting* line segments defined over a grid such that

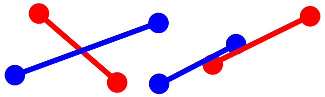
- each point and each end point of a segment is a grid point
- each end point of a segment is also a point of the realm



- no realm point lies within a segment



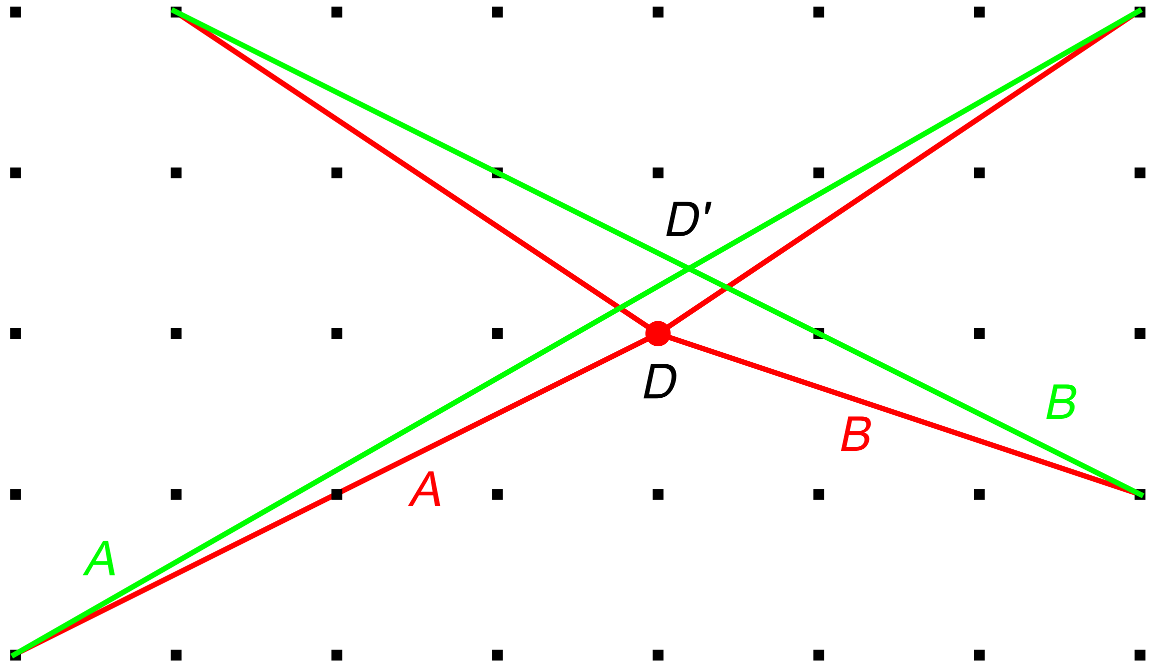
- any two distinct segments do neither properly intersect nor overlap



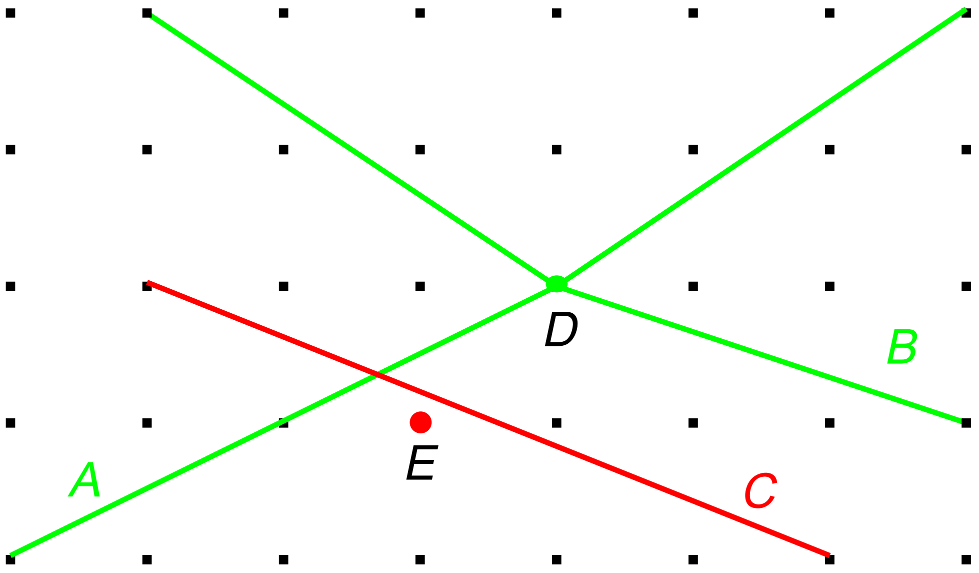
→ A realm is a *spatially embedded planar graph .*

All numerical problems are treated *below* the realm layer:

- input: application data that are sets of points and *intersecting* line segments
- output: “realmified” data that have become acquainted with each other
- basic idea: slightly distort/perturb both segments



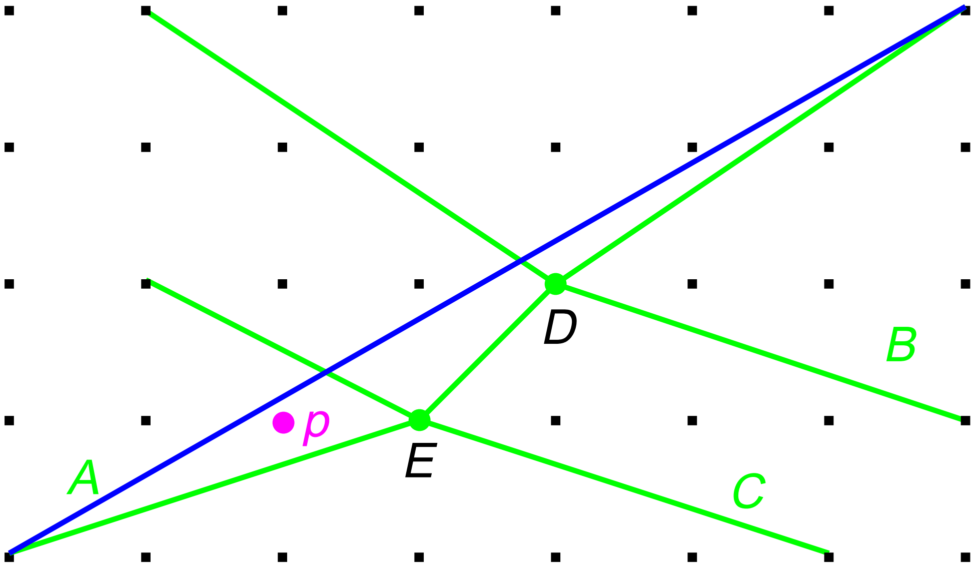
Good solution?



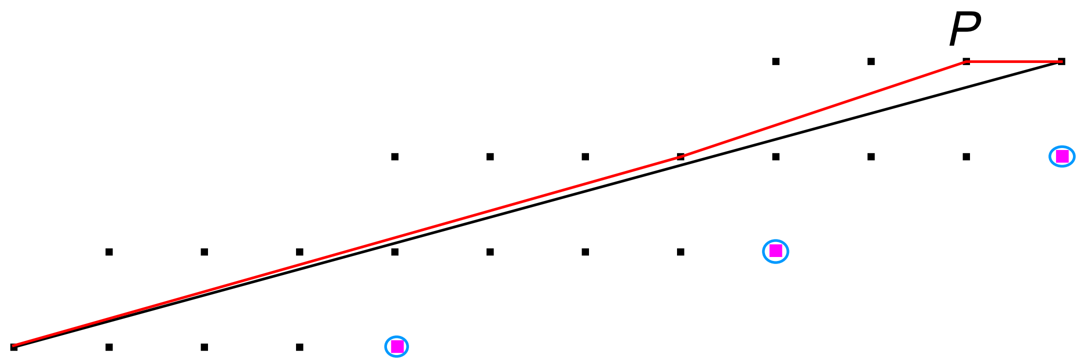
Intersection of segment *A* with some further segment *C*

Observations

- Segments can move far away from their original position by iterated intersections!
- Topological errors can occur: point *p* is now on the wrong side of *A*!



Solution: *redrawing* of segments only within their envelope (Greene & Yao 1986)



Segments

- are "caught" within their envelope (dotted points + auxiliary points ●)
- can never cross a grid point
- *idea:* elastic band in a bed of nails

Advantages of the realm concept:

- definition of distinct SDTs on a common domain, guarantee of closure properties
- protection of geometric computation in query processing from problems of numerical robustness and topological correctness
- enforcement of geometric consistency of related spatial objects

Disadvantages of the realm concept:

- no SDT operations possible that create new geometries (leave the realm closure), e.g., *convex_hull*, *voronoi*
- integration of realms into database systems possible, but somewhat difficult, since propagation of realm updates to realm-based attribute values in database objects needed

1.7 Integrating Spatial Data Types into a DBMS Data Model

Integration of single, self-contained spatial objects

- can be realized in a data model independent way (→ ADTs)

- Basic concept: represent “spatial objects” (i.e., points, lines, regions) by *objects* of the DBMS data model *with at least one SDT attribute*

- DBMS data model must be open for new, user-defined types

  → ADT support, → extensibility

- Example for the relational model:

```
 relation states(sname: string, area: regions, spop: integer)
 relation cities( cname: string, center: point, extent: region,
                  cpop: integer)
 relation rivers(rname: string, flow: line)
```

```
Query: select cname from cities, rivers
        where intersect(extent,flow) and distance(center,flow)<10
```

Integration of spatially related collections of objects

- not data model independent

- partitions
  - set of database objects with *region* attribute?
  - loss of information: disjointness or adjacency of regions cannot be modeled in SDT but support of these integrity constraints by the DBMS needed

- networks
  - not much research on *spatially embedded networks*
  - e.g., Güting 1994: GraphDB with *explicit graphs* integrated into an OO model

1.8 What has to be modelled from an application point of view?

Spatial data types

- self-contained objects: points, lines, regions (one object is a *set* of points, line segments or simple polygons !)
- spatially related collections of objects: partitions, networks

Spatial operations

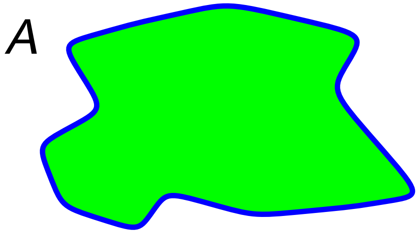
- *spatial predicates* returning boolean values
  - *topological relationships*
    e.g., *equal*, *unequal*, *disjoint*, *adjacent* (*neighboring*), *intersect* (*overlap*), *meet* (*touch*), *inside* (*in*), *outside*, *covered_by*, *contains*
  - metric relationships
    e.g., *in_circle*, *in_window*
  - spatial order and strict order relationships
    e.g., *behind / in_front_of*, *above* / *below*, *over* / *under*, *inside* / *contains*
  - directional relationships
    e.g., *north/south _of*, *left /right _of*

Spatial Operations (*continued*)

- spatial operations returning numbers
  e.g., *area*, *perimeter*, *length*, *diameter*, *dist*, *mindist*, *maxdist*, *direction*, *components* (*cardinality*)
- spatial operations returning new spatial objects
  - object construction operations
    e.g., *union*, *intersection*, *difference*, *convex_hull*, *center*, *boundary* (*border*), *box*
  - object transformation operations
    e.g., *extend*, *rotate*, *translate*
- spatial operations on sets of spatially related objects
  - general operations
    e.g., *voronoi*, *closest*, *compose*, *decompose*
  - operations for partitions
    e.g., *overlay*, *superimposition*, *fusion*, *cover*, *windowing*, *clipping*
  - operations for networks
    e.g., *shortest_path*

Topological Relationships (Egenhofer & Herring 1990, Egenhofer & Franzosa 1991)

- goal: a "*complete*" collection of *topological relationships* between two spatial objects
- topological relationships are invariant under translation, rotation, and scaling
- originally: topol. relationships between two simple, connected regions without holes



boundary ≡ ∂A

interior ≡ A°

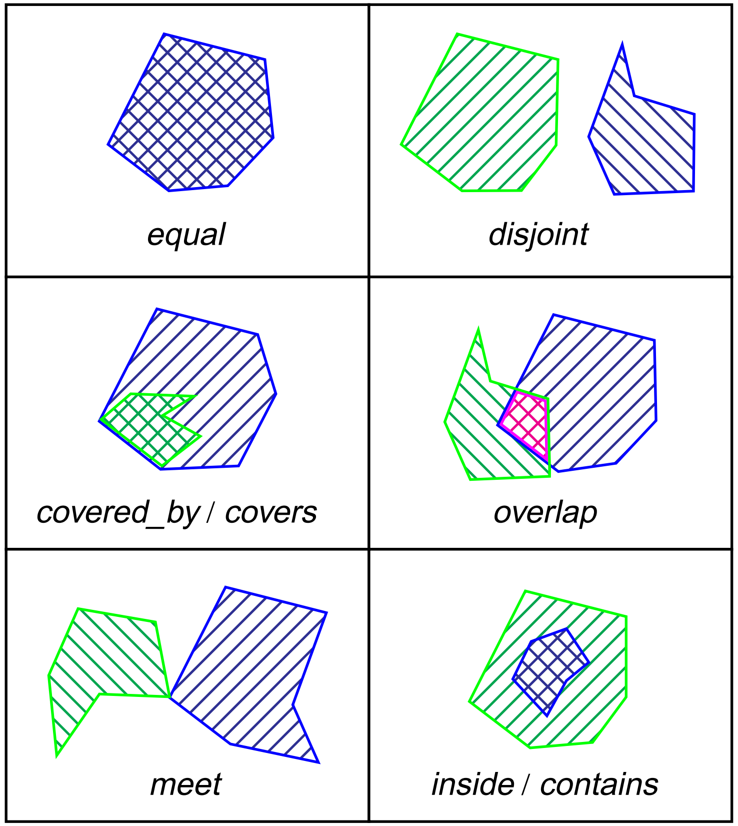
- *4-intersection model*: 4 intersection sets between boundaries and interiors of two objects (see next page)

Extension:

- *9-intersection model* (Egenhofer 1991): consider also intersections of ∂A and A° with the exterior / complement A^{-} ($\rightarrow 9^2 = 81$ combinations, 8 are valid)
- includes point and line objects

- *4-intersection model*

$\partial A \cap \partial B$	$\partial A \cap B^\circ$	$A^\circ \cap \partial B$	$A^\circ \cap B^\circ$	relationship name
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	*A* and *B* are *disjoint*
$\emptyset$	$\emptyset$	$\emptyset$	$\neq\emptyset$	
$\emptyset$	$\emptyset$	$\neq\emptyset$	$\emptyset$	
$\emptyset$	$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	*A contains B* / *B inside A*
$\emptyset$	$\neq\emptyset$	$\emptyset$	$\emptyset$	
$\emptyset$	$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	*A inside B* / *B contains A*
$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	
$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	
$\neq\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	*A* and *B meet*
$\neq\emptyset$	$\emptyset$	$\emptyset$	$\neq\emptyset$	*A* and *B* are *equal*
$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	$\emptyset$	
$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	$\neq\emptyset$	*A covers B* / *B covered_by A*
$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	$\emptyset$	
$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	$\neq\emptyset$	*A covered_by B* / *B covers A*
$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\emptyset$	
$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	$\neq\emptyset$	*A* and *B overlap*



4^2 = 16 combinations, 8 are valid