

Spatial Abstract Data Types (ADTs)

The previous section dealt with the internal representation of spatial objects. In this section, we also take a look at the geographic properties of objects in terms of a *reference schema* and *reference queries*.

From this, we then develop spatial ADTs, with a focus on

- extending data models with spatial ADTs
- designing spatial ADTs
- exploring topological predicates

Additional literature:

- [RSV02] Chapter 3 (Sections 3.1-3.3)
- Markus Schneider: "*Spatial Data Types: Conceptual Foundation for the Design and Implementation of Spatial Database Systems and GIS*", tutorial given at 6th International Symposium on Advances in Spatial Databases (SSD'99), 1999.

<http://www.cise.ufl.edu/~mschneid/Service/Tutorials/TutorialSDT.pdf>

Reference Schema and Queries

- A *conceptual spatial database schema* describes at an abstract level one or many *themes* for representing a set of geographic objects in a geographic application, including relationships among objects.
- For this, standard modeling frameworks can be used, such as UML.
- For a spatial database schema, then reference queries are described that specify typical queries in the application(s) to be built.

Example reference schemas:

1. Administrative units (countries, states, counties,...)
2. Highway network among cities (highways, sections, cities,...)
3. Land use (areas in the plane, type of use,...)

For each schema, a set of reference queries is determined:

- Alphanumerical queries (only refer to descriptive components)
- Spatial queries (operations that apply to one or more spatial objects)

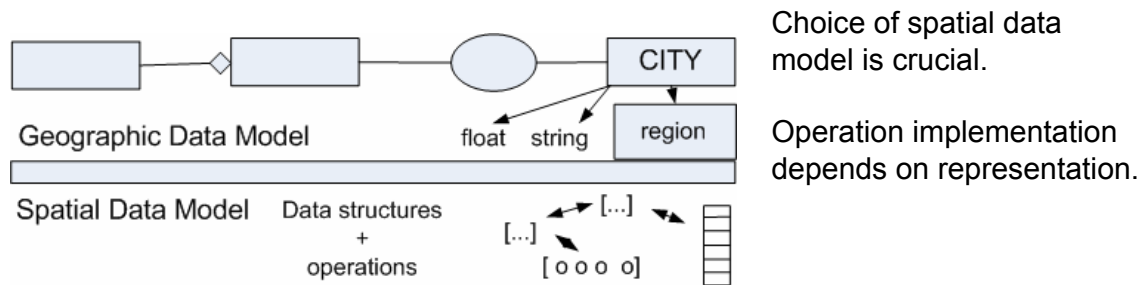
Spatial Abstract Data Types

An ADT is an abstract view of objects: a set of operations is defined on a set of objects of a given type.

The general idea is to hide implementation details of the data type from the user (or programmer) and only allow access through operations.

⇒ *encapsulation*

The ADT approach gives rise to distinguishing between *geographic data models* and *spatial data models*.



Choosing Appropriate Types

There are numerous candidates for geometric types: point, polyline, complex polyline, polygon, polygon set, mixed,

There is a trade-off between the *modeling power* captured by a definition of an ADT and the *constraints* imposed on a chosen representation. For example, what properties is a polyline supposed to have?

ADTs also need to be evaluated in terms of

- *space complexity*: the amount of storage needed to represent objects and relationships between objects
- *Time complexity*: the amount of time algorithms need to compute properties of objects and collections of objects that are not explicitly represented.

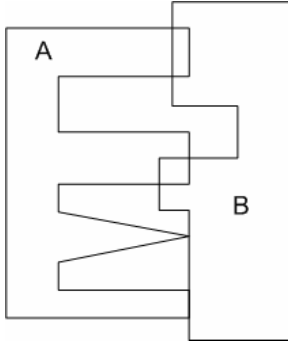
Also:

- Realization should allow for an *unambiguous representation of subdivisions*.

Defining ADT Operations

By no means simple....

Result of operations must be either atomic (real, Boolean,...) or any of the spatial abstract base types.



What is the result of intersecting two polygons, i.e., $A \cap B = ??$ ☞

Closure properties are important!

The result of an operation needs to be enforced, and operations have to comply with the type system defined.

Spatial ADT Signatures

In the following, each operation has a *signature*, which states the type of each of its arguments and of its result.

We assume four spatial ADTs: point, polyline, region, and rectangle.

The region ADT includes the following operations:

- *PointInRegion*: $region \times point \rightarrow bool$
- *Overlaps*: $region \times region \rightarrow bool$
- *OverlapsRectangle*: $region \times rectangle \rightarrow bool$
- *Clipping*: $region \times rectangle \rightarrow region$
- *Intersection*: $region \times region \rightarrow region$
- *Meets*: $region \times region \rightarrow bool$
- *Area*: $region \rightarrow real$
- *RegionUnion*: $\{region\} \rightarrow region$

Spatial ADT Signatures (2)

The line ADT includes the following operations:

- *PointOnLine*: $line \times point \rightarrow bool$
- *Length*: $line \rightarrow real$
- *OverlapsLR*: $line \times region \rightarrow bool$

The point ADT includes the following operations:

- *Distance*: $region \times point \rightarrow real$

The above lists are not exhaustive but would support typical queries corresponding to the reference schemas. However, some perhaps useful operations cannot be defined (examples ↗)

In the following, we further explore the issue of spatial data type design.

Designing Spatial ADTs

Recall that type definitions are strongly application driven. A SDBMS, however, should support much more general sets of types and operations, providing a wide range of functions for *data representation* and *data manipulation*.

In the following, we define a (non-exhaustive) list of operations, with a particular focus on managing geospatial data; then we focus more on the semantics. We assume that at least one argument is of a spatial data type.

Unary operations with Boolean result.

This operation tests a spatial object for a given property.

SimpleLine: $line \rightarrow bool$

ClosedLine: $line \rightarrow bool$

Convex: $polygon \rightarrow bool$

(*IsSquare*: $rectangle \rightarrow bool$)

(*IsRectangle*: $polygon \rightarrow bool$)

Operators

Unary operations with scalar result:

area, length, diameter, perimeter, components (cardinality), direction,...

Unary operations with spatial result:

There is a large number of operations of different nature in this class.

1. *Topological transforms*, i.e., homeomorphisms that preserve topological relationships. That is, all of the continuous, bijective mappings $\mathbf{R}^2 \rightarrow \mathbf{R}^2$.

Example operators include *rotation, translation, scale change, ...*



2. *Dimension reduction*, i.e., operators that transform a d -dimensional objects into a $d-1$ -dimensional object (examples ↗)
3. *Object extraction*, including operations like *minimum bounding rectangle, region centroid* etc. (examples ↗)

Operators (2)

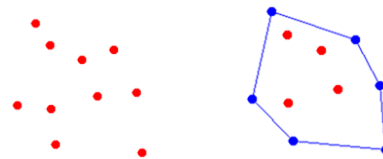
N-ary operations with spatial result (assuming a homogeneous input set):

ConvexHull: point \rightarrow polygon*

PointSCenter: point \rightarrow point*

Voronoi: point \rightarrow polygon**

others ??? ↗



Binary operations with spatial result:

Most frequently used operations for spatial queries are *set operations*, which apply to objects represented as (infinite) sets of points.

An operand can be a single object or a set of objects of the same type.

*L1Intersect: line \times line \rightarrow point**

L2Intersect: line \times line \rightarrow point**

L3Intersect: line \times line* \rightarrow point**



We'll look at these types of operators in more detail later...

Operators (3)

Binary operations with Boolean result: (aka *spatial predicates*)

They provide the basis for spatial querying and more particularly for *spatial selection* and *spatial join*.

1. *Topological predicates*, which are invariant under topological transforms, including *intersect*, *contains*, *adjacent*, or *is_enclosed_by*.

A *point query* tests whether a point is contained by a single object or set of objects, e.g., *contains: point × polygon → bool*

A *range query* tests whether an object intersects a given region, e.g., *intersect: line × region → bool*

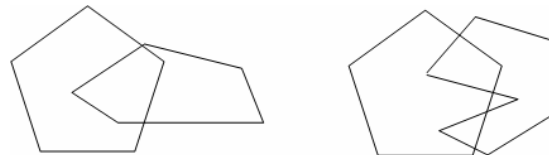
2. *Direction predicates*, such as *above*, *North of*,....

3. *Metric predicates*, e.g., testing whether the distance between two point is less than a given value.

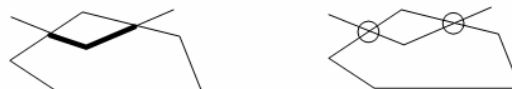
Binary operations with scalar results, e.g., distance between two points, min/max distance, ...

Semantics of Operators

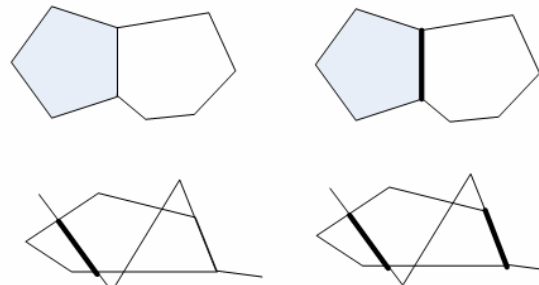
What is the meaning of an operation (in a given dimension)?



Should an operator preserve the dimensions of its operands?



An intersection operator that preserves the dimensions of its operands is called *regularized*. For example, the intersection of two convex polygons is always a *single convex polygon*.



Cases for intersections in the plane

Semantics of Operators (2)

First of all, the result of operations should satisfy one of the existing types. One (minor) problem is that a result can be a set of objects of an existing type (e.g., the intersection of a polyline with a polygon, depending on how intersection is defined).

A larger type system thus seems appropriate, provided it at least incorporates the notion of a *set*. An approach that chooses a type large enough to remain closed under a set of common operations is called a *weak type-based approach*. In this approach, there is one type *geometry*, which is a set of either points, lines, or polygons.

⇒ Does not take advantage of the power of *strong typing*.

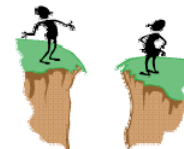


If a type system is not appropriately defined, it might be necessary to access the internal structure of objects, and thus violate encapsulation.

Topological Predicates

Exploring spatial relationships is a challenging task involving geography, cognitive sciences etc. (“far from”, “meets”).

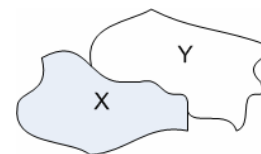
A formal definition of spatial relationships is crucial to clarify the understanding and implementation!



In the following, we introduce a *computational method* for reasoning about binary topological relations between spatial regions and *deduce consistency* of complete and incomplete topological information.

We use set theory as basis. Let $points(x)$ denote the set of points that belong to a spatial region x . Then

- $x=y \equiv points(x) = points(y)$
- $x \neq y \equiv points(x) \neq points(y)$
- $x \text{ inside } y \equiv points(x) \subseteq points(y)$
- $x \text{ outside } y \equiv points(x) \cap points(y) = \emptyset$
- $x \text{ intersects } y \equiv points(x) \cap points(y) \neq \emptyset$



Minimal and complete??

4-Intersection Scheme

The point set approach has been augmented with the notion of *boundary* and *interior* of spatial objects to distinguish between the topological predicates *meets* and *overlap*. The goal is to have a “complete” collection of topological relationships.

By comparing whether or not boundaries and interiors of two spatial regions intersect, four relations can be identified, known as *4-intersection scheme*.

For this, the common concepts of point set topology with open and closed sets are used.

(Def) A point set S in \mathbf{R}^2 is *open* if for each point $p \in S$, there exists an $\varepsilon \in \mathbf{R}$, $\varepsilon > 0$, such that the disc with radius ε and center p is contained in S . (recall the notion of open ball/disc from Section 2.1)
 S is *closed* if $\mathbf{R}^2 - S$ is open.

4-Intersection Scheme (2)

Interior, complement, and boundary of an object in 2-dimensional space that is described by a non-empty point set A :

- interior of A ($\equiv A^\circ$) is the union of all open sets in A
- closure of A ($\equiv \bar{A}$) is the intersection of all closed sets that contain A
- complement of A ($\equiv C_A$) with respect to \mathbf{R}^2 is the set of all points in \mathbf{R}^2 not contained in A
- boundary of A ($\equiv \delta A$) is the intersection of the closure of A and the closure of the complement of A



A° , δA , and C_A are mutually exclusive; union = ...

Given two spatial regions (objects), the idea is to find out what topological relationships between them can be deduced from the relationships between their interiors and boundaries.

4-Intersection Scheme (3)

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

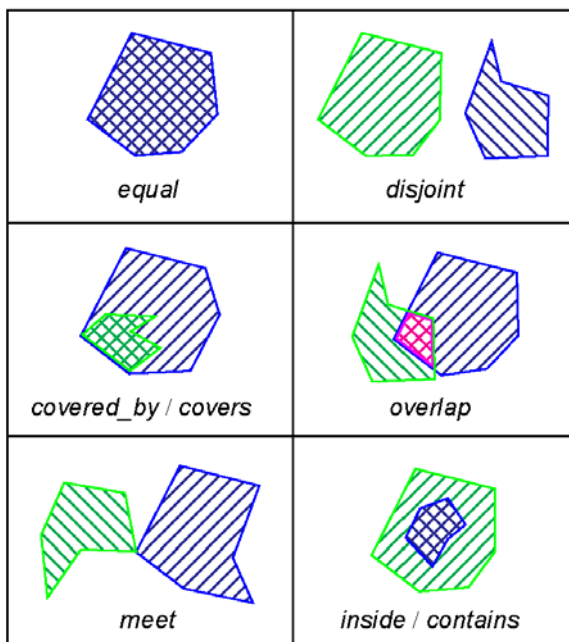
[taken from Schneider's tutorial]

Given two spatial objects A and B:
There are 2^4 combinations between the interiors A° , B° and boundaries δA , δB .

Only eight of those are possible.

What if A and B are polygons with holes.....??

4-Intersection Scheme (4)



[taken from Schneider's tutorial]

Note that because of different topological spaces, the same configuration may show different relations between spatial regions as the notion of boundary and interior may vary.

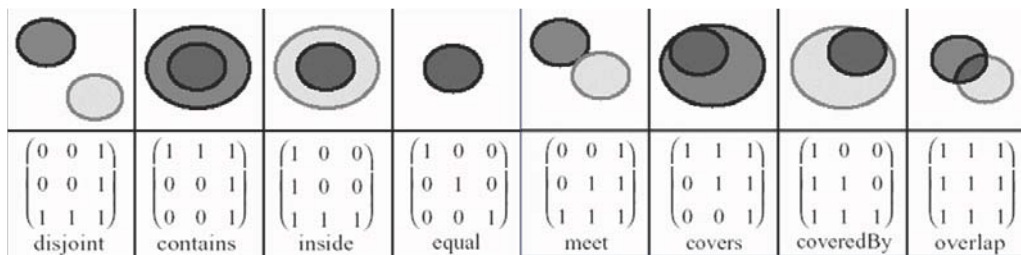
Example: what is the interior of a line in 2-dimensional space?

9-Intersection Scheme

Given 2-dimensional point sets A and B, the topological relationships can also be described by the *9-intersection scheme* of A's boundary, interior, and *complement* with the boundary, interior, and *complement* of B.

Representation as a 3×3 matrix:

$$I_9(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \delta B & A^\circ \cap C_B \\ \delta A \cap B^\circ & \delta A \cap \delta B & \delta A \cap C_B \\ C_A \cap B^\circ & C_A \cap \delta B & C_A \cap C_B \end{pmatrix}$$



Remarks

There has been plenty of work on defining spatial relationships

- The 4-intersection scheme has been introduced by Egenhofer and Herring in 1990.
- This scheme has been extended to higher dimensions by Egenhofer and Franzosa in 1991.
- Worboys and Bofakas (1993) describe a point set topology approach to define complex spatial relations with holes and islands to any finite level.
- Belussi, Bertino, & Catania (1997), and Grumbach, Rigaux & Segoufin (1998) describe a linear constraint approach, based on the spatial constraint model.
- Güting & Schneider (1995) present the ROSE algebra, which is based on a discrete geometric basis (system of realm-based spatial data types).

Again, see Schneider's tutorial for an overview.