Spatial and Moving Objects Databases: 
State of the Art and Future Research Challenges

Markus Schneider
University of Florida
Department of Computer & Information Science & Engineering
Abstract

Spatial databases are full-fledged databases that, in addition, enable the storage, retrieval, manipulation, querying, and analysis of geometries like points, lines, and regions representing, for example, the geometries of cities, roads, and states respectively. More complex examples are spatial partitions representing spatial subdivisions like the counties in Florida and the election districts in Gainesville, and spatial graphs representing spatial networks like transportation networks and pipeline systems. Moving objects databases also deal with geometries but focus on the change of their location and/or shape and/or extent over time. Examples are moving points representing cell phone users, moving lines representing traffic congestions, and moving regions representing hurricanes. The objective of this tutorial is to highlight the state of the art of spatial and moving objects databases, and indicate their future research challenges. The focus is on data models, querying, data structures, algorithms, and system architectures. The tutorial will show that spatial data types and spatiotemporal data types provide a fundamental abstraction for modeling the geometric structure of objects in space, the temporally evolving structure of objects in space and time, their relationships, properties, and operations. These data types develop their full expressive power if they are integrated as abstract data types into databases and their query languages.
Outline

1. Introduction
2. Spatial Data Modeling
3. Spatiotemporal Data Modeling
4. Open Research Topics
Outline

1. Introduction
2. Spatial Data Modeling
3. Spatiotemporal Data Modeling
4. Open Research Topics
Outline - Introduction

- Spatial and spatiotemporal data are ubiquitous
- Application examples
- Spatial and moving objects databases as interdisciplinary research fields
- Colloquial example queries
- Definition of spatial and spatiotemporal database systems
Spatial and Spatiotemporal Data are Ubiquitous (I)

- 80% of all available data have either an explicit or an implicit geographical reference

- Explicit geographical references
  - Actual geometries (spatial structures) of geographical objects
  - Examples: city boundaries, lakes, railroads, landmarks, school districts

- Implicit geographical references
  - Textual references to the names and descriptions of geographic objects
  - Examples: zip codes, street names, city names, land parcel numbers
  - Geocoding: Google Map, MapQuest, Bing Maps, etc.
Spatial and Spatiotemporal Data are Ubiquitous (II)

- Adding time lets spatial objects move
  - Continuous evolution of geometries over time: movement, motion
  - Examples: hurricanes, traffic, whales, glaciers

- Conclusions
  - Efficient and user friendly handling of large spatial data volumes is an important task of database technology
  - Understanding of spatial and spatiotemporal data is needed
Applications for Spatial Databases (I)

Data management foundation for Geographical Information Systems (GIS)

DNA helix

Spatial analysis
Applications for Spatial Databases (II)

The brain's complex network of 70 billion neurons and thousands of kilometers of circuits

The Bubble complex (the Hodge object), a star cluster, in NGC 6946

A street map
Applications for Moving Objects Databases (I)

Traffic congestion research

Car navigation system

Migration routes of whales
Applications for Moving Objects Databases (II)

Hurricane research

Material flow in transportation pipelines (electricity, water, oil)

Movement of planets
Introduction

Spatial and Moving Objects Databases as Interdisciplinary Research Fields

- Database Systems
- Geographical Information Systems (GIS)
- Geosciences
- Cognitive Science
- Linguistics
- Mathematics
- Computational Geometry
- Artificial Intelligence
- Spatial and Spatiotemporal Data Warehousing
- Spatial and Spatiotemporal Data Mining
- Spatial and Moving Objects Database Systems

Spatial and Moving Objects Databases: State of the Art and Future Research Challenges
Introduction

User Groups and Spatial Queries of Interest (I)

**Mobile phone user:** Where is the nearest gas station? Is there a pet-food vendor on my way home?

**Army field commander:** Has there been any significant enemy troop movement since last night?

**Insurance risk manager:** Which houses on the Mississippi River are most likely to be affected by the next great flood?

**Medical doctor:** Based on this patient’s Magnetic Resonance Imaging (MRI), have we treated somebody with a similar condition?

**Molecular biologist:** Is the topology of the amino acid biosynthesis gene in the genome found in any other sequence feature map in the database?

**Astronomer:** Find all blue galaxies within two arcmin of quasars.

**Climatologist:** How can I test and verify my new global warming model?
User Groups and Spatial Queries of Interest (II)

**Pharmaceutical Researcher:** Which molecules can dock with a given molecule based on geometric shapes?

**Sports:** Which seats in a baseball stadium provide best view of pitcher and hitter? Where should TV camera be mounted?

**Corporate supply manager:** Given trends about our future customer profile, which are the best places to build distribution warehouses and retail stores?

**Transport specialist:** How should the road network be expanded to minimize traffic congestion?

**Urban sprawl specialist:** Is new urban land development leading to the loss of rich agricultural land.

**Ski resort owner:** Which mountains on our property are ideal for a beginner’s ski run?

**Farmer:** How can I minimize the use of pesticide on my farm?

**Golf entrepreneur:** Where do I build a new golf course which will maximize profit given the constraints of weather, Environmental Protection Agency pesticide regulations, the Endangered Species Act, property prices, and the neighborhood demographic profile.

**Emergency service:** Where is the person calling for help located? What is the best route to reach her?
### User Groups and Spatiotemporal Queries of Interest (I)

<table>
<thead>
<tr>
<th>Moving Point Entities</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>People: politicians, terrorists, criminals</td>
<td>- When did Bush meet Arafat?  &lt;br&gt; - Show the trajectory of Lee Harvey Oswald on November 22, 1963.</td>
</tr>
<tr>
<td>Animals</td>
<td>- Determine trajectories of birds, whales, . . .  &lt;br&gt; - Which distance do they traverse, at which speed? How often do they stop?  &lt;br&gt; - Where are the whales now?  &lt;br&gt; - Did their habitats move in the last 20 years?</td>
</tr>
<tr>
<td>Satellites, spacecraft, planets</td>
<td>- Which satellites will get close to the route of this spacecraft within the next four hours?</td>
</tr>
<tr>
<td>Cars: taxi cabs, trucks</td>
<td>- Which taxi is closest to a passenger request position?  &lt;br&gt; - Which routes are used regularly by trucks?  &lt;br&gt; - Did the trucks with dangerous goods come close to a high-risk facility?</td>
</tr>
<tr>
<td>Airplanes</td>
<td>- Were any two planes close to a collision?  &lt;br&gt; - Are two planes heading toward each other (going to crash)?  &lt;br&gt; - Did planes cross the air territory of state X?  &lt;br&gt; - At what speed does this plane move? What is its top speed?  &lt;br&gt; - Did Iraqi planes pass the 39th degree?</td>
</tr>
<tr>
<td>Ships</td>
<td>- Are any ships heading toward shallow areas?  &lt;br&gt; - Find &quot;strange&quot; movements of ships, indicating illegal dumping of waste.</td>
</tr>
<tr>
<td>Military vehicles: rockets, missiles, tanks, submarines</td>
<td>- All kinds of military analyses.</td>
</tr>
</tbody>
</table>
## User Groups and Spatiotemporal Queries of Interest (II)

<table>
<thead>
<tr>
<th>Moving Region Entities</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countries</td>
<td>What was the largest extent ever of the Roman Empire?</td>
</tr>
<tr>
<td></td>
<td>On which occasions did any two states merge (e.g., reunification)?</td>
</tr>
<tr>
<td></td>
<td>Which states split into two or more parts?</td>
</tr>
<tr>
<td></td>
<td>How did the Serb-occupied areas in former Yugoslavia develop over time? When was the maximal extent reached?</td>
</tr>
<tr>
<td>Forests, lakes</td>
<td>How fast is the Amazon rain forest shrinking?</td>
</tr>
<tr>
<td></td>
<td>Is the Dead Sea shrinking?</td>
</tr>
<tr>
<td></td>
<td>What is the minimal and maximal extent of river X during the year?</td>
</tr>
<tr>
<td>Glaciers</td>
<td>Does the polar ice cap grow? Does it move?</td>
</tr>
<tr>
<td></td>
<td>Where must glacier X have been at time Y (backward projection)?</td>
</tr>
<tr>
<td>Storms</td>
<td>Where is the hurricane heading? When will it reach Florida?</td>
</tr>
<tr>
<td>High/low pressure areas</td>
<td>Where do they go? Where will they be tomorrow?</td>
</tr>
<tr>
<td>Scalar functions over space (e.g., temperature)</td>
<td>Where has the 0-degree boundary been last midnight?</td>
</tr>
<tr>
<td>People</td>
<td>Movements of the Celts in the second century B.C.</td>
</tr>
<tr>
<td>Troops, armies</td>
<td>Hannibal traversing the Alps. Show his trajectory. When did he pass village X?</td>
</tr>
<tr>
<td>Cancer</td>
<td>Can we find in a series of X-ray images a growing cancer? How fast does it grow? How big was it on June 1, 1995?</td>
</tr>
<tr>
<td>Continents</td>
<td>History of continental shift</td>
</tr>
<tr>
<td>Diseases</td>
<td>Show the area affected by mad cow disease for every month in 1998.</td>
</tr>
<tr>
<td>Oil spills</td>
<td>Which parts of the coast will be touched tomorrow?</td>
</tr>
</tbody>
</table>
What is a Spatial Database System?

- A spatial database system is a database system.
- It offers spatial data types (SDTs) in its data model and query language.
- It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial joins.

What is a Moving Objects Database System?

- A **moving objects database system** is a spatial database system.
- In addition, it offers **spatiotemporal data types (STDTs)** for moving objects (including operations and predicates) in its data model and query language.
- It supports spatiotemporal data types in its implementation, providing at least **spatiotemporal indexing** and efficient algorithms for **spatiotemporal joins**.
Goals of this Tutorial

- Provide an overview and understanding of the essential concepts of spatial and moving objects database systems
  - What is special with spatial (geometric) data?
  - What is special with spatiotemporal data?

- Emphasize the importance of spatial and spatiotemporal data types to model and query (moving) geometries in databases

- Modeling and conceptual aspects are the focus of this tutorial but not implementation aspects
Outline

1. Introduction
2. Spatial Data Modeling
3. Spatiotemporal Data Modeling
4. Open Research Topics
Outline – Spatial Data Modeling

- How spatial data should not be represented
- Spatial data types
- Geometric set operations
- Metric operations
- Topological relationships
- Directional relationships
- Spatial querying
How Spatial Data should NOT Be Represented (I)

- Example: Census block, and how we would perhaps like to store it in a table

- Problem: *Polyline* is not a built-in data type
**How Spatial Data should NOT Be Represented (II)**

- First solution ever proposed: Create a set of tables with shared attributes

<table>
<thead>
<tr>
<th>Census_blocks</th>
<th>Polygon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Area</td>
</tr>
<tr>
<td>340</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge-name</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
How Spatial Data should NOT Be Represented (III)

Problems

- Polyline representation is *scattered* over several (!) tables
- How do we know that these four tables represent a polyline?
- The relational algebra and SQL do not provide geometric operations
- SQL queries, if possible, are low-level queries
- Even if so, it would not be clear how to apply these geometric operations to the tables

Only chance

- Expensive joins needed to obtain full information about the polyline
- All tuples have to be loaded into main memory for further processing
- Tuples have to be transformed into a geometric data structure
- Geometric operations have to be applied to geometric data structure
How Spatial Data should NOT Be Represented (IV)

Conclusions

- This solution is highly inefficient
- Table representation does not allow the application of geometric operations.
- Whole approach is unusable
- An object concept is needed
- Spatial and spatiotemporal data are/form complex application objects and are not an accumulation of their constituent parts
- The relational data model and relational database systems cannot and should not be used to represent complex application objects like spatial and spatiotemporal data
What are Spatial Data Types? (I)

- Spatial data types
  - are special data types needed to model geometry and to suitably represent geometric data in database systems
  - Examples: point, line, region; partition (map), graph (network)
  - 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
What are Spatial Data Types? (II)

Conclusions

- An understanding of SDTs 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, graphical user interfaces
  - for a cooperation with extensible DBMS providing spatial type extension packages like spatial data blades, cartridges

- The definition and implementation of spatial data types is probably the most fundamental issue in the development of spatial database systems
Which Spatial Data Need to Be Represented? (I)

- Two kinds of representations of spatial phenomena (vector/raster debate)
  - objects in space (entity-oriented / feature-based view)
    - vector data, boundary representation
    - spatial database systems
  - space itself (space-oriented / position-based view)
    - raster data, bitmap data, pixel data
    - image database systems

- We consider
  - modeling single, self-contained spatial objects
  - modeling spatially related collections of spatial objects
Which Spatial Data Need to Be Represented? (II)

Fundamental abstractions for modeling single, self-contained objects

<table>
<thead>
<tr>
<th>point</th>
<th>line</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>city</td>
<td>highway</td>
<td>city</td>
</tr>
<tr>
<td>castle</td>
<td>river</td>
<td>lake</td>
</tr>
<tr>
<td>lighthouse</td>
<td>cable</td>
<td>district</td>
</tr>
<tr>
<td>church</td>
<td>route</td>
<td>forest</td>
</tr>
</tbody>
</table>

(location of object in space but not its extent relevant)  
(connections in space, movement through space)  
(extent of an object relevant)

Fundamental abstractions for modeling spatially related collections of objects

- partition
- Spatially embedded network (graph)

land use  
districts  
wards  
countries  
speech areas  

- highways  
- railways  
- rivers  
- electricity  
- phone

Others: nested partitions, digital terrain models
A Three-Level Model for Phenomena in Space

**Structure objects**
- Structure types: sets, sequences, partitions, networks
- Operations: overlay, shortest_path

**Spatially-referenced objects**
- Object types: city, state, river
- Operations: lies_in: city → state, flow: river → line

**Spatial objects**
- Spatial data types: point, line, polygon
- Operations: point-in-polygon test, intersection

Spatial and Moving Objects Databases: State of the Art and Future Research Challenges
Abstraction Levels for Data Type Design and Implementation

- Abstract Level
  - Spatial/ST data types
  - Abstract line, region, volume, surface, moving point, moving region
  - Definitions based on mathematics (e.g., point set theory, point set topology, algebraic topology)
  - Spatial/ST operations and predicates
  - No consideration of implementation aspects

- Discrete Level
  - Finite representations of spatial data types
    - Discrete data structures for SDTs/STDTs
    - Discrete line, region, volume, surface, moving point, moving region
  - Algorithms for spatial/ST operations and predicates on finite representations

- Implementation Level
  - Geometric data structures
    - In a programming language (C++, Java)
    - Array structures, no pointer structures
  - Geometric algorithms
    - Methods from Computational Geometry
Design Criteria for Modeling Spatial Data Types

- Data model independence
- General definition
- Formal definition
- Topological correctness
- Numerical robustness
- Enabling finite representations
- Geometric consistency
- Closure properties
- Efficiency
- Extensibility

Design of Spatial Data Types
Closure Properties

- General definition/structure of spatial objects

→ Spatial objects must be closed under set operations on the underlying point sets
Geometric Consistency

- Support of geometric consistency constraints for spatially related objects

→ SDT definition must offer facilities to enforce topological consistency constraints
Spatial Data Types – Intuitive Definition (I)

- **Data type point**: A (complex) point object consists of a finite number of single points.
- **Data type line**: A (complex) line object consists of a finite number of blocks (connected components). Each block consists of a finite number of disjoint or meeting continuous curves.
- **Data type region**: A (complex) region object consists of a finite number of disjoint or 0-meet faces (sub-regions). Each face consists of an outer cycle and a finite (zero, one, or more) number of disjoint or 0-meet hole cycles (holes).
Spatial Data Types – Intuitive Definition (II)

- Advantages of these definitions
  - Intersections of two line objects can be represented as a single complex point object
  - No dependence of the object-oriented type constructors (set, list, array) of the data model of the deployed database system
  - No distribution of the single points of the result in several tuples of a relational table
  - Several disjoint rivers can be represented as a single line object (waterways)
  - Multiple components and holes in regions are needed since
    - this guarantees the closure properties of spatial data types
    - this is required by spatial applications (e.g., Italy with mainland, offshore islands, and the Vatican)
Why Do We Need a Formal Definition of SDTs?

- Better understanding of the complex semantics of spatial objects and operations at the discrete level
- Formal definition of SDTs should be directly usable for a formal definition of corresponding spatial operations and predicates
- Clarity and consistency of the conceptual user view of SDTs
- A first step towards a standardization of spatial data types (achieved)
- Formal specification of SDTs for a possible implementation at the implementation level
Point Set Theory (I)

- Basic assumption: space is composed of infinitely many points (Euclidean plane $\mathbb{R}^2$) and contains a set of spatial objects.
- Each spatial object $Y$ can be regarded as the point set occupied by that object ($Y \subseteq \mathbb{R}^2$).
- Use of set operations $\cup$, $\cap$, $-$ for constructing new spatial objects.
- Possible anomalies.

Diagram showing various spatial anomalies such as missing point, boundary part, dangling line, dangling point, and missing line.
Point Set Theory (II)

- Possible anomalies (continued)

- Ambiguities when defining topological relationships
  - Let $A$ and $B$ two spatial objects, and let \textit{points} be a function that provides the point set of a spatial object
  - $A = B := \text{points}(A) = \text{points}(B)$
    - $A$ inside $B := \text{points}(A) \subseteq \text{points}(B)$
    - $A$ intersects $B := \text{points}(A) \cap \text{points}(B) \neq \emptyset$
  - The definitions of $=$ and \textit{inside} are both covered by the definition of \textit{intersects}
Point Set Topology (I)

- Same basic assumptions as point set theory but investigates topological structures of a point set
  
  *boundary* ($\partial Y$), *interior* ($Y^\circ$), *closure* ($\overline{Y}$), *exterior* ($Y^-$)

\[
\overline{Y} = Y^\circ \cup \partial Y \quad Y^\circ \cap \partial Y = \emptyset \quad Y^- \cap \partial Y = \emptyset \quad Y^\circ \cap Y^- = \emptyset
\]

\[
\mathbb{R}^2 = Y^\circ \cup \partial Y \cup Y^-
\]

- Point set topology investigates properties that are independent of an underlying distance or coordinate measure (metric) and that are preserved under continuous topological transformations
Point Set Topology (II)

- **Regularization** of point sets to avoid anomalies
- Spatial regions are defined as regular closed sets
- \( Y \) is a regular closed set if \( Y = \overline{Y} \)
- Effect of the *interior* function: elimination of dangling points, dangling lines and boundary parts
- Effect of *closure* function: elimination of cuts and punctures by appropriately adding points, adding missing boundary points
- Regularization function \( reg(Y) := \overline{Y} \)
- Geometric set operations are equated with regular set operations (\( A, B \) regular closed sets)

\[
A \oplus B := reg(A \cup B) \quad A \otimes B := reg(A \cap B) \quad A \ominus B := reg(A - B)
\]
Formal Definition of Spatial Data Types

- The spatial data type *point* is defined as

  \[ \text{point} = \{ P \subseteq \mathbb{R}^2 \mid P \text{ is finite} \} \]

- The spatial data type *line* is defined as

  \[ \text{line} = \{ L \subseteq \mathbb{R}^2 \mid (i) \quad L = \bigcup_{i=1}^{n} f_i([0, 1]) \text{ with } n \in \mathbb{N}_0 \]
  \[\quad (ii) \quad \forall 1 \leq i \leq n : f_i : [0, 1] \rightarrow \mathbb{R}^2 \text{ is a continuous mapping} \]
  \[\quad (iii) \quad \forall 1 \leq i \leq n : |f_i([0, 1])| > 1 \}. \]

- The spatial data type *region* is defined as

  \[ \text{region} = \{ R \subseteq \mathbb{R}^2 \mid (i) \quad R \text{ is regular closed} \]
  \[\quad (ii) \quad R \text{ is bounded} \]
  \[\quad (iii) \quad \text{The number of connected sets of } R \text{ is finite} \} \]
Geometric Operations (I)

- Geometric union, geometric intersection, and geometric difference (geometric set operations) of two spatial objects of the same spatial type
  - union, intersection, difference: $\alpha \times \alpha \rightarrow \alpha$ for all $\alpha \in \{\text{point, line, region}\}$

- Formal definition the geometric set operations for $\alpha = \text{region}$
  - $\text{union}(A, B) := A \oplus B$
  - $\text{intersection}(A, B) := A \otimes B$
  - $\text{difference}(A, B) := A \ominus B$
Geometric Operations (II)

- Geometric intersection of two spatial objects of different spatial data types
  
  \[ \text{intersection: } \alpha \times \beta \rightarrow \alpha \text{ for all } \alpha, \beta \in \{\text{point, line, region}\} \text{ and } \alpha \neq \beta \]

- Computation of the common boundary of two spatial objects
  
  \[ \text{commonBorder: } \alpha \times \beta \rightarrow \text{line} \text{ for all } \alpha, \beta \in \{\text{line, region}\} \text{ and } \alpha \neq \beta \]

- Computation of the boundary of a spatial object
  
  - boundary: region \rightarrow \text{line}
  - boundary: line \rightarrow \text{point}
Geometric Operations (III)

- Computation of the common boundary points of two spatial objects
  \[ \text{commonPoints}: \alpha \times \beta \rightarrow \text{point} \text{ for all } \alpha, \beta \in \{\text{line, region}\} \text{ and } \alpha \neq \beta \]

- Computation of the center of a spatial object
  \[ \text{center}: \alpha \rightarrow \text{point} \text{ for all } \alpha \in \{\text{point, line, region}\} \]
Geometric Operations (IV)

- Computation of the **convex hull** of a point object
  \[ \text{convexHull}: \text{point} \rightarrow \text{region} \]

- Computation of the **minimum bounding box** (rectangle) of a spatial object, that is, its smallest enclosing and axis-parallel rectangle
  \[ \text{box}: \alpha \rightarrow \text{region} \text{ for all } \alpha \in \{\text{point, line, region}\} \]
Geometric Operations (V)

- Zooming a spatial object by a factor
  \[ \text{scale} : \alpha \times \text{float} \rightarrow \alpha \text{ for all } \alpha \in \{\text{point, line, region}\} \]

- Rotation of a spatial object around a point by a given angle
  \[ \text{rotate} : \alpha \times \text{point} \times \text{float} \rightarrow \alpha \text{ for all } \alpha \in \{\text{point, line, region}\} \]

- Moving a spatial object by a defined vector
  \[ \text{translate} : \alpha \times \text{point} \rightarrow \alpha \text{ for all } \alpha \in \{\text{point, line, region}\} \]

- Creation of a region object (a buffer) that contains all points in a certain distance from a spatial input object (a negative distance lets the input object shrink)
  \[ \text{bufferZone} : \alpha \times \text{float} \rightarrow \text{region} \text{ for all } \alpha \in \{\text{point, line, region}\} \]
Geometric Operations (VI)

- **Voronoi** operation: For a given set $S$ of points in the plane, the **Voronoi diagram** associates with each point $p$ from $S$ the region consisting of those points of the plane that are closer to $p$ than to any other point in $S$.

![Voronoi diagram](image)

- The **closest** operation yields a point object whose point components are nearest to a given single **reference point**.

\[
\text{closest} : \text{point} \times \text{point} \rightarrow \text{point}
\]

![Closest point](image)
Metric Operations (I)

- Computation of the area of a region object
  \[\text{area}: \text{region} \rightarrow \text{float}\]

- Computation of the length of a region boundary
  \[\text{perimeter}: \text{region} \rightarrow \text{float}\]

- Computation of the length of a line object
  \[\text{length}: \text{line} \rightarrow \text{float}\]

- Computation of the largest Euclidean distance between any of two points of a spatial object
  - \[\text{diameter}: \text{region} \rightarrow \text{float}\]
  - \[\text{diameter}: \text{line} \rightarrow \text{float}\]
  - \[\text{diameter}: \text{point} \rightarrow \text{float}\]
  - In summary: \[\text{diameter}: \alpha \rightarrow \text{float} \text{ for all } \alpha \in \{\text{point, line, region}\}\]
Metric Operations (II)

- Ideas for new unary distance operations
  - $\text{minIntraDist}$,
  - $\text{maxIntraDist}$,
  - $\text{minInterDist}$,
  - $\text{maxInterDist}: \alpha \rightarrow \text{float}$ for all $\alpha \in \{\text{point, line, region}\}$

- Minimum/maximum Euclidean distance computation between two spatial objects
  - $\text{dist, maxDist}: \alpha \times \beta \rightarrow \text{float}$ for all $\alpha, \beta \in \{\text{point, line, region}\}$

- Computation of the direction between two single points as the angle $0 \leq \alpha < 360$ between them
  - $\text{direction}: \text{point} \times \text{point} \rightarrow \text{float}$

- Number of components (points, blocks, faces) of a spatial object
  - $\text{noOfComps}: \alpha \rightarrow \text{integer}$
Spatial Partitions and Their Operations (I)

- **Spatial partition**: subdivision of the plane into pairwise disjoint regions where each region is associated with an attribute (label) having a simple or even complex structure.

- **Partition constraints** maintain topological relationships between regions:
  - Interiors of different region objects must be disjoint.
  - Different region objects are adjacent (they meet) or they are disjoint.

- Important example of a spatial connectivity structure.
Spatial Partitions and Their Operations (II)

- The overlay operation transparently lays two partitions of different themes on top of each other and combines them into a new spatial partition.
The *superimposition* operation superimposes the spatial objects of a spatial partition on another spatial partition in order to cover and erase parts of the other partition.
Spatial Partitions and Their Operations (IV)

- The *fusion* operation merges the region objects of a spatial partition on the basis of the equality of a non-spatial attribute. For each group of equal non-spatial objects, a spatial object is created as the geometric union of the set of spatial objects of each group.

- Example: Compute the spatial partition with the regions of the same land use.
Spatial Partitions and Their Operations (V)

- The *cover* operation yields a spatial partition with a single region object as the geometric union of all region objects of a spatial partition.
Spatial Partitions and Their Operations (VI)

- The *windowing* operation retrieves those regions of a spatial partition whose intersection with a given window (rectangle, circle, polygon) is not empty.
Spatial Partitions and Their Operations (VII)

- The *clipping* operation selects those parts of a spatial partition which lie inside a given window (rectangle, circle, polygon).

- Observation: The formal definition of all application-specific operations on spatial partitions can be derived from three fundamental operations:
  - intersection
  - relabel
  - refine
Spatial Networks and Their Operations (I)

- **Spatial network**: spatially embedded graph in the plane consisting of pairwise disjoint, continuous lines where each line is associated with an attribute (label) having a simple or even complex structure.

- The *shortest path* (*shortest route*) operation determines the shortest path between any two network points with respect to:
  - network distance
  - time
  - slope
  - any weight on which a distance metric can be defined
  - a combination of weights
Spatial Networks and Their Operations (II)

- *Windowing* and *clipping* again, but now applied to spatial networks (Euclidean range query)
Spatial Networks and Their Operations (III)

- The operations *spatial network union*, *spatial network intersection*, and *spatial network difference* compute the union, intersection, and difference of two spatial networks of the same label type or different label types.
  - *Spatial network intersection*: Given a bus route network and a taxi network. Which roads do they share?
  - *Spatial network union*: Given a bus route network and a metro network. Which locations can a passenger reach?
  - *Spatial network difference*: Given a bus route network and a taxi network. Which locations can taxis but not buses reach?

- The operation *spatial network buffer* considers a spatial network as a line object and computes a buffer around it.
  - Example: Given a bus route network. Compute the region that represents its “zone of attraction”.

Spatial Data Modeling
Spatial Networks and Their Operations (IV)

- The operation `get graph` extracts the graph structure of a spatial network.
- The operation `network distance` determines the distance between two points in a network.
  - Important for `network range queries`: Determine the spatial network that has a distance $d$ from a network point $s$.
- The operation `length` sums up the length of all edges of a spatial network.
  - What is the overall length of the bus network in Gainesville?
- The operation `maxDist` determines the maximum distance within a spatial network.
  - What is the farthest distance in the bus network?
- The operation `get transfer points` determines the common points of two different spatial networks where one can transfer from one network into the other.
- Many more operations …
Spatial Networks and Their Operations (V)

- Spatial network modeling in a database context
  - Very few interesting models exist but they are incomplete
  - Still a lot of effort needed

- Spatial network representation in a database context
  - Poor: network data are spread over a large collection of tables
  - Expensive joins needed to bring data together
  - Network operations cannot be applied to these representations

- Spatial network querying in a database context
  - Formulation of network queries is difficult
  - No spatial network queries possible since network operations are missing
  - High-level spatial network query language is missing

- Spatial network algorithms in a database context
  - A number of efficient spatial network algorithms is available
  - However: they depend on a particular spatial network database system
Spatial Databases cannot Store Spatial Partitions and Spatial Networks

- Both are highly complicated connectivity structures
- Both have complex operations involving large volumes of data
- Both are *not* first-class citizens in spatial databases
Topological Relationships (I)

- **Topological relationships** characterize the relative position of two spatial objects towards each other.

- **Examples:**
  - meet
  - disjoint
  - inside

- Needed for **spatial joins** and **spatial selections**

  ```sql
  SELECT cname, sname
  FROM cities, states
  WHERE loc inside territory
  ```
Topological Relationships (II)

- Requirements of a model for topological relationships
  - All topological relationships must be mutually exclusive
  - The set of topological relationships must be complete

- Proposed models:
  - 9-Intersection Matrix model (9IM)
    - based on point set theory and point set topology
    - takes into account the topological invariants of the intersections of the boundary, interior, and exterior of one spatial object with the corresponding components of another spatial object
  - Region Connection Calculus (RCC)
    - based on spatial logic, “pointless” approach
    - axiomatic approach to formulating topological relationships and reasoning about spatial data
Topological Relationships (III)

- 9-intersection matrix \((A, B \in \{\text{point, line, region}\})\)

\[
\begin{pmatrix}
A^o \cap B^o \neq \emptyset & A^o \cap \partial B \neq \emptyset & A^o \cap B^- \neq \emptyset \\
\partial A \cap B^o \neq \emptyset & \partial A \cap \partial B \neq \emptyset & \partial A \cap B^- \neq \emptyset \\
A^- \cap B^o \neq \emptyset & A^- \cap \partial B \neq \emptyset & A^- \cap B^- \neq \emptyset
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 1
\end{pmatrix}
\quad
\begin{pmatrix}
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix}
\quad
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix}
\quad
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

- disjoint
- meet
- overlap
- equal
- inside
- contains
- covers
- coveredBy
- meet
- inside / contains
Topological Relationships (IV)

- Finding topological relationships: proof-by-constraint-and-drawing

- Phase 1
  - Starting point: $2^9 = 512$ possible matrices
  - Find constraint rules that must hold for valid topological relationships
    - Example: the intersection of the exteriors of two spatial objects is always non-empty (eliminates 256 matrices immediately)
    - Constraints are type-combination specific
  - Apply constraint rules one after the other and remove matrices that do not fulfill them

- Phase 2
  - For the remaining matrices, draw prototypical spatial configurations in the Euclidean plane to verify the existence of the corresponding topological relationships
Topological Relationships (V)

- Number of topological relationships for two simple spatial objects

<table>
<thead>
<tr>
<th></th>
<th>Simple Point</th>
<th>Simple Line</th>
<th>Simple Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple point</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Simple line</td>
<td>3</td>
<td>33</td>
<td>19</td>
</tr>
<tr>
<td>Simple region</td>
<td>3</td>
<td>19</td>
<td>8</td>
</tr>
</tbody>
</table>

- Number of topological relationships for two complex spatial objects

<table>
<thead>
<tr>
<th></th>
<th>Complex Point</th>
<th>Complex Line</th>
<th>Complex Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex point</td>
<td>5</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>Complex line</td>
<td>14</td>
<td>82</td>
<td>43</td>
</tr>
<tr>
<td>Complex region</td>
<td>7</td>
<td>43</td>
<td>32</td>
</tr>
</tbody>
</table>
Example 1: Topological relationships between two complex point objects

- Matrix 1:
  
  \[
  \begin{pmatrix}
  0 & 0 & 1 \\
  0 & 0 & 0 \\
  1 & 0 & 1 
  \end{pmatrix}
  \]

- Matrix 2:
  
  \[
  \begin{pmatrix}
  1 & 0 & 0 \\
  0 & 0 & 0 \\
  0 & 0 & 1 
  \end{pmatrix}
  \]

- Matrix 3:
  
  \[
  \begin{pmatrix}
  1 & 0 & 0 \\
  0 & 0 & 0 \\
  1 & 0 & 1 
  \end{pmatrix}
  \]

- Matrix 4:
  
  \[
  \begin{pmatrix}
  1 & 0 & 1 \\
  0 & 0 & 0 \\
  0 & 0 & 1 
  \end{pmatrix}
  \]

- Matrix 5:
  
  \[
  \begin{pmatrix}
  1 & 0 & 1 \\
  0 & 0 & 0 \\
  1 & 0 & 1 
  \end{pmatrix}
  \]

- **component of point A**
- **component of point B**
- **common component of both points**
Topological Relationships (VII)

- Example 2: Topological relationships between two complex region objects
Topological Relationships (VIII)

- Problem: too many topological relationships that the user cannot handle
- Solution: clustering rules that put similar topological relationships together into a topological cluster predicate

<table>
<thead>
<tr>
<th>Cluster predicate</th>
<th>Point/Point</th>
<th>Line/Line</th>
<th>Region/Region</th>
<th>Point/Line</th>
<th>Point/Region</th>
<th>Line/Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>disjoint_c</td>
<td>1</td>
<td>1–4</td>
<td>1</td>
<td>1, 2</td>
<td>1</td>
<td>1, 2</td>
</tr>
<tr>
<td>meet_c</td>
<td>–</td>
<td>5–32</td>
<td>2–4</td>
<td>3–6</td>
<td>2, 3</td>
<td>3–13</td>
</tr>
<tr>
<td>covers_c</td>
<td>–</td>
<td>49, 52, 69, 72</td>
<td>11, 21, 24</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>coveredBy_c</td>
<td>–</td>
<td>37, 38, 41, 42</td>
<td>6, 8, 9</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>inside_c</td>
<td>3</td>
<td>34, 35, 39, 40</td>
<td>7</td>
<td>7, 8, 11, 12</td>
<td>4, 6</td>
<td>15, 17, 28, 31</td>
</tr>
<tr>
<td>contains_c</td>
<td>4</td>
<td>43, 46, 63, 66</td>
<td>19</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>overlap_c</td>
<td>5</td>
<td>44, 45, 47, 48, 50, 51, 53–62, 64, 65, 67, 68, 70, 71, 73–82</td>
<td>10, 12–18, 20</td>
<td>9, 10, 13, 14</td>
<td>5, 7</td>
<td>14, 16, 26, 27, 29, 30</td>
</tr>
<tr>
<td>equal_c</td>
<td>2</td>
<td>–</td>
<td>5</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Directional Relationships (I)

- **Directional relationships** characterize the directional orientation of two spatial objects towards each other
  - *absolute* directional relationships (cardinal directions): *north_of* and *southwest_of* with respect to a given reference or coordinate system
  - *relative* directional relationships: *in_front* and *left* with respect to an observer position

- Needed for **spatial joins** and **spatial selections**

```
states(sname: varchar(50), territory: region)
SELECT s.sname
FROM states s,
     ( SELECT territory
          FROM states
          WHERE sname = 'Italy') t
WHERE s.territory northwest_of t.territory
```
Directional Relationships (II)

- **Objects Interaction Matrix (OIM) model**
  - Computational model for cardinal directions
  - Cognitive, subjective models can be built on top

- **Two phases**
  - Tiling Phase
  - Interpretation Phase

![Diagram showing the process of OIM Generation and interpretation phases with arrows indicating direction relationships]

\[ \text{dir}(A,B) \quad \text{dir}(B,A) \]
Directional Relationships (III)

- Tiling Phase
  - Determine Objects Interaction Grid (OIG) through partitioning lines
  - Determine Objects Interaction Grid Space (OIGS)
Directional Relationships (IV)

- **Tiling Phase (continued)**
  - 9 possible sizes of OIGs
  - $m \times n$-OIG for $m, n \in \{1, 2, 3\}$

![Diagrams of 3x3-OIG, 3x2-OIG, 3x1-OIG, 2x3-OIG, 2x2-OIG, 2x1-OIG, 1x3-OIG, 1x2-OIG, 1x1-OIG]
Directional Relationships (V)

- **Tiling Phase (continued)**
  - Map OIG into an **Objects Interaction Matrix** (here: $3 \times 3$-matrix)

  \[
  OIM(A, B) = \begin{pmatrix}
  \iota(A, B, t_{1,1}) & \iota(A, B, t_{1,2}) & \iota(A, B, t_{1,3}) \\
  \iota(A, B, t_{2,1}) & \iota(A, B, t_{2,2}) & \iota(A, B, t_{2,3}) \\
  \iota(A, B, t_{3,1}) & \iota(A, B, t_{3,2}) & \iota(A, B, t_{3,3}) 
  \end{pmatrix}
  \]

- **Objects Interaction Function**

  \[
  \iota(A, B, t_{i,j}) = \begin{cases}
  0 & \text{if } A^o \cap t^o_{i,j} = \emptyset \land B^o \cap t^o_{i,j} = \emptyset \\
  1 & \text{if } A^o \cap t^o_{i,j} \neq \emptyset \land B^o \cap t^o_{i,j} = \emptyset \\
  2 & \text{if } A^o \cap t^o_{i,j} = \emptyset \land B^o \cap t^o_{i,j} \neq \emptyset \\
  3 & \text{if } A^o \cap t^o_{i,j} \neq \emptyset \land B^o \cap t^o_{i,j} \neq \emptyset
  \end{cases}
  \]
Directional Relationships (VI)

- Tiling Phase (continued)
  - Numbers of possible $m \times n$-matrices
    
    $\begin{array}{c|ccc}
    \times & 1 & 2 & 3 \\
    \hline
    1 & 4 & 16 & 64 \\
    m & 2 & 16 & 256 & 4096 \\
    3 & 64 & 4096 & 262144 \\
    \end{array}$

  - Total: 270756

- Numbers of valid OIMs for two complex/simple region objects
  
  $\begin{array}{c|ccc}
  \times & 1/1 & 6/6 & 8/6 \\
  \hline
  1 & 1/1 & 6/6 & 8/6 \\
  m & 2 & 6/6 & 84/68 & 216/124 \\
  3 & 8/6 & 216/124 & 1132/464 \\
  \end{array}$

  - Total: 1677/805
Directional Relationships (VII)

- **Interpretation Phase**
  - Determine the location of each part of $A$ and $B$ in $OIM(A, B)$
  - Example

![Diagram of directional relationships]

- $loc(A, OIM(A, B)) = \{(1, 1), (1, 2), (2, 1)\}$
- $loc(B, OIM(A, B)) = \{(2, 3), (3, 2), (3, 3)\}$
Directional Relationships (VIII)

- Interpretation Phase (*continued*)
  - Set of basic cardinal directions \( CD = \{N, NW, W, SW, S, SE, E, NE, O\} \)
  - Determine the cardinal directions between the parts of \( A \) and \( B \) by an interpretation function \( \psi \)

\[
\psi(((i, j), (i', j'))) = \begin{cases} 
N & \text{ if } i < i' \land j = j' \\
NW & \text{ if } i < i' \land j < j' \\
W & \text{ if } i = i' \land j < j' \\
SW & \text{ if } i > i' \land j < j' \\
S & \text{ if } i > i' \land j = j' \\
SE & \text{ if } i > i' \land j > j' \\
E & \text{ if } i = i' \land j > j' \\
NE & \text{ if } i < i' \land j > j' \\
O & \text{ if } i = i' \land j = j' 
\end{cases}
\]
Directional Relationships (IX)

- Interpretation Phase (continued)
  - Interpretation table for the interpretation function $\psi$

<table>
<thead>
<tr>
<th>$(i, j)$</th>
<th>(1,1)</th>
<th>(1,2)</th>
<th>(1,3)</th>
<th>(2,1)</th>
<th>(2,2)</th>
<th>(2,3)</th>
<th>(3,1)</th>
<th>(3,2)</th>
<th>(3,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>O</td>
<td>W</td>
<td>W</td>
<td>N</td>
<td>NW</td>
<td>NW</td>
<td>N</td>
<td>NW</td>
<td>NW</td>
</tr>
<tr>
<td>(1,2)</td>
<td>E</td>
<td>O</td>
<td>W</td>
<td>NE</td>
<td>N</td>
<td>NW</td>
<td>NE</td>
<td>N</td>
<td>NW</td>
</tr>
<tr>
<td>(1,3)</td>
<td>E</td>
<td>E</td>
<td>O</td>
<td>NE</td>
<td>NE</td>
<td>N</td>
<td>NE</td>
<td>NE</td>
<td>N</td>
</tr>
<tr>
<td>(2,1)</td>
<td>S</td>
<td>SW</td>
<td>SW</td>
<td>O</td>
<td>W</td>
<td>W</td>
<td>N</td>
<td>NW</td>
<td>NW</td>
</tr>
<tr>
<td>(2,2)</td>
<td>SE</td>
<td>S</td>
<td>SW</td>
<td>E</td>
<td>O</td>
<td>W</td>
<td>NE</td>
<td>N</td>
<td>NW</td>
</tr>
<tr>
<td>(2,3)</td>
<td>SE</td>
<td>SE</td>
<td>S</td>
<td>E</td>
<td>E</td>
<td>O</td>
<td>NE</td>
<td>NE</td>
<td>N</td>
</tr>
<tr>
<td>(3,1)</td>
<td>S</td>
<td>SW</td>
<td>SW</td>
<td>S</td>
<td>SW</td>
<td>SW</td>
<td>O</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>(3,2)</td>
<td>SE</td>
<td>S</td>
<td>SW</td>
<td>SE</td>
<td>S</td>
<td>SW</td>
<td>E</td>
<td>O</td>
<td>W</td>
</tr>
<tr>
<td>(3,3)</td>
<td>SE</td>
<td>SE</td>
<td>S</td>
<td>SE</td>
<td>SE</td>
<td>S</td>
<td>E</td>
<td>E</td>
<td>O</td>
</tr>
</tbody>
</table>
Directional Relationships (X)

- Interpretation Phase (continued)
  - Example (continued)
    - Determine the cardinal directions $\text{dir}(A, B)$ between $A$ and $B$
      $$\text{dir}(A, B) = \psi(\text{loc}(A, OIM(A, B)), \text{loc}(B, OIM(A, B)))$$
Directional Relationships (XI)

- Interpretation Phase (continued)
  - Example (continued)

\[
\text{dir}(A, B) = \psi(\text{loc}(A, OIM(A, B)), \text{loc}(B, OIM(A, B))) \\
= \psi(\{(1, 1), (1, 2), (2, 1)\}, \\
\{(2, 3), (3, 2), (3, 3)\}) \\
= \{\psi((1, 1), (2, 3)), \psi((1, 1), (3, 2)), \\
\psi((1, 1), (3, 3)), \ldots, \psi((2, 1), (3, 3))\} \\
= \{\text{NW}, \text{N}, \text{W}\}
\]

\[
\begin{pmatrix}
1 & 1 & 0 \\
1 & 0 & 2 \\
0 & 2 & 2
\end{pmatrix}
\]

\[
\text{dir}(B, A) = \psi(\text{loc}(B, OIM(A, B)), \text{loc}(A, OIM(A, B))) \\
= \psi(\{(2, 3), (3, 2), (3, 3)\}, \\
\{(1, 1), (1, 2), (2, 1)\}) \\
= \{\psi((2, 3), (1, 1)), \psi((2, 3), (1, 2)), \\
\psi((2, 3), (2, 1)), \ldots, \psi((3, 3), (2, 1))\} \\
= \{\text{SE}, \text{S}, \text{E}\}
\]
Directional Relationships (XII)

- Existential predicates

\[
\begin{align*}
A & \text{ north\_of } B \quad {\text{def}} \quad N \in \text{dir}(A, B) \\
A & \text{ west\_of } B \quad {\text{def}} \quad W \in \text{dir}(A, B) \\
A & \text{ south\_of } B \quad {\text{def}} \quad S \in \text{dir}(A, B) \\
A & \text{ east\_of } B \quad {\text{def}} \quad E \in \text{dir}(A, B) \\
A & \text{ origin\_with } B \quad {\text{def}} \quad O \in \text{dir}(A, B) \\
A & \text{ northwest\_of } B \quad {\text{def}} \quad NW \in \text{dir}(A, B) \\
A & \text{ southwest\_of } B \quad {\text{def}} \quad SW \in \text{dir}(A, B) \\
A & \text{ northeast\_of } B \quad {\text{def}} \quad NE \in \text{dir}(A, B) \\
A & \text{ southeast\_of } B \quad {\text{def}} \quad SE \in \text{dir}(A, B)
\end{align*}
\]

- \( \text{dir}(A, B) = \{NW, N, W\} \) means

\[
A \text{ west\_of } B \land A \text{ northwest\_of } B \land A \text{ north\_of } B \land \\
\neg(A \text{ northeast\_of } B) \land \neg(A \text{ east\_of } B) \land \neg(A \text{ southeast\_of } B) \land \\
\neg(A \text{ south\_of } B) \land \neg(A \text{ southwest\_of } B) \land \neg(A \text{ origin\_with } B)
\]
Directional Relationships (XIII)

- Similarly-oriented directional predicates

\[ A\ \text{northern}\_of\ B \overset{\text{def}}{=} A\ \text{north}\_of\ B \lor A\ \text{northwest}\_of\ B \lor A\ \text{northeast}\_of\ B \]

- Strict directional predicates

\[ A\ \text{strictly\_north}\_of\ B \overset{\text{def}}{=} A\ \text{north}\_of\ B \land \\
\neg(A\ \text{south}\_of\ B) \land \neg(A\ \text{west}\_of\ B) \land \\
\neg(A\ \text{east}\_of\ B) \land \neg(A\ \text{northwest}\_of\ B) \land \\
\neg(A\ \text{northeast}\_of\ B) \land \neg(A\ \text{southwest}\_of\ B) \land \\
\neg(A\ \text{southeast}\_of\ B) \land \neg(A\ \text{origin}\_with\ B) \]

- Surround directional predicates

\[ A\ \text{surronds}\ B \overset{\text{def}}{=} A\ \text{north}\_of\ B \land A\ \text{northeast}\_of\ B \land A\ \text{east}\_of\ B \land \\
A\ \text{southeast}\_of\ B \land A\ \text{south}\_of\ B \land A\ \text{southwest}\_of\ B \land \\
A\ \text{west}\_of\ B \land A\ \text{northwest}\_of\ B \land A\ \text{origin}\_with\ B \]
Spatial Querying (I)

- Tables
  - CREATE TABLE Country(name varchar(30), pop int, shape region);
  - CREATE TABLE River(name varchar(30), shape line);
  - CREATE TABLE City(name varchar(30), pop int, shape point);

- Q1: Find the names of all countries that are neighbors of the USA.
  
  `SELECT C1.name AS neighbors
  FROM Country C1, Country C2
  WHERE C1.shape meets C2.shape AND C2.name = 'USA'
  `

- Q2: The St. Lawrence River can supply water to cities that are within 300 km. Determine those cities and compute their distance from the river.
  
  `SELECT C.name, dist(C.shape, R.shape) AS distance
  FROM City C, River R
  WHERE C1.shape inside bufferZone(R.shape, 300) AND R.name = 'St. Lawrence River'`
Spatial Querying (II)

Q3: List the length of the rivers in each of the countries they pass through together with the river name and the country name.

```
SELECT R.name, C.name,
       length(intersection(R.shape, C.shape)) AS rLength
FROM River R, Country C
WHERE R.shape intersects C.shape
```

Q4: List all countries, ordered by the number of adjacent countries.

```
SELECT C.name, COUNT(C1.name) AS noOfNeighbors
FROM Country C, Country C1
WHERE C.shape meets C1.shape
GROUP BY C.name
ORDER BY noOfNeighbors DESC
```
Spatial Querying (III)

- Q5: List the names of the pairs of all countries that are separated by exactly one other country.

\[
\text{SELECT } \quad \text{C1.name, C3.name} \\
\text{FROM } \quad \text{Country C1, Country C2, Country C3} \\
\text{WHERE } \quad \text{C1.shape meets C2.shape AND C2.shape meets C3.shape} \\
\quad \text{AND C1.shape disjoint C3.shape}
\]

- Q6: List the names of all countries that are north of “Mexico”.

\[
\text{SELECT } \quad \text{C.name} \\
\text{FROM } \quad \text{Country C, Country C1} \\
\text{WHERE } \quad \text{C.shape north_of C1.shape AND C1.name = ‘Mexico’}
\]

Effect of strictly_north_of, northern_of, strictly_northern_of?
Spatial Querying (IV)

- **Spatial data types**
  - are specified as *abstract data types (ADTs)*
  - ADTs are used as *attribute types* in relational (or other) schemas
  - Internals of ADTs are invisible to the user
  - ADT implementations can be exchanged without the need to redefine database schemas and reformulate queries

- **Spatial operations and predicates**
  - are *high-level methods*
  - are embedded into an extension of the SQL query language
  - Algorithms for the operations and predicates are invisible to the user
  - Algorithms for the operations and predicates can be exchanged without the need to reformulate queries since the interfaces (signatures) stay the same
Outline

1. Introduction
2. Spatial Data Modeling
3. Spatiotemporal Data Modeling
4. Open Research Topics
Outline – Spatiotemporal Data Modeling

- What are Spatiotemporal Data Types?
- Uncertain Moving Objects
- Predictive Moving Objects
- Spatiotemporal Partitions and Moving Objects in Spatial Networks
- Spatiotemporal Operations
- Spatiotemporal Predicates
- Spatiotemporal Querying
What are Spatio-temporal Data Types? (I)

- **Spatiotemporal data types** are data types to represent moving objects.
- A **moving object** represents the *continuous* evolution of a spatial object over time.

**Moving point**
- Only time-dependent location is of interest
- Examples: eyes of tropical cyclones, cell phone users, terrorists, whales

**Moving region**
- Also the time-dependent shape and/or areal extent is of interest
- Examples: hurricanes, forest fires, oil spills, diseases, glaciers

**Moving line**
- The time-dependent shape and/or linear extent is of interest
- Examples: traffic jams, front of an army; boundary of any moving region
What are Spatiotemporal Data Types? (II)

- Formally: Let $\alpha \in \{\text{point}, \text{line}, \text{region}\}$, and let $\text{time}$ be the data type to represent time instants. Then a (historical) moving object $m$ is a function of a spatiotemporal data type $\tau(\alpha) = \text{time} \rightarrow \alpha$.
  - $hmpoint = \tau(\text{point}) = \text{time} \rightarrow \text{point}$
  - $hmregion = \tau(\text{region}) = \text{time} \rightarrow \text{region}$
  - $hmline = \tau(\text{line}) = \text{time} \rightarrow \text{line}$
Uncertain Moving Objects

- A moving object $m \in \tau(\alpha)$ is a partial function and can have definition gaps.
- If the maximum velocity of $m$ is known, the uncertainty region of $m$ can be computed as a space-time prism.
Predictive Moving Objects (I)

- Many applications are interested in a forecast of the (near) future temporal evolution of a spatial object (e.g., hurricane research, meteorology, fire management, disaster management, disease control).
- Example: Hurricane Ernesto (August 2012)
- Prediction is application-specific and afflicted with uncertainty.
- Geometric aspect can be represented by spatiotemporal data types $\tau(\alpha)$. 

![Diagram of potential positions over time](image)
Predictive Moving Objects (II)

- Uncertainty aspect represented by a confidence distribution function that is assigned an domain/application-specific distribution function
- Nature of these functions: probabilistic, fuzzy, possibilistic, rough, …
- *Balloon* metaphor taken from hurricane research
  - Past movement of a hurricane resembles the string of a balloon
  - Future prediction of a hurricane resembles the body of a balloon
- *Balloon object* = historical moving object + predictive moving object (spatiotemporal snapshot)
- Observation: $\text{dim(historical moving object)} \leq \text{dim(predictive moving object)}$
Predictive Moving Objects (III)

- **Moving balloon objects** represent the dynamics of moving objects as a continuous sequence of balloon objects.
- Past movement is updated (historical accumulation).
- New predictions and employed prediction models are possible due to new insights and input factors.
- Quality of predictions can be checked.

![Diagram of Predictive Moving Objects](image)
Spatiotemporal Partitions and Moving Objects in Spatial Networks

- Spatiotemporal partitions (ongoing research)
  - represent the temporal evolution of spatial partitions
  - Maintain partition constraints over time
  - Examples: change of temperature zones (weather maps on TV), temporal evolution of icing, development of unemployment rate

- Moving objects in spatial networks (ongoing research)
  - Moving objects are spatially constrained
  - Movement of spatial objects depends on speed, slope, number of lanes, other moving objects, and so on
  - Examples: car navigation, location-based services, mobile computing, traffic jams, pipelines (water, oil, power), Internet
  - Continuous range queries interesting for car navigation
    Example: Permanently report the three nearest restaurants while driving
Spatiotemporal Operations (I)

Operations for projection to domain/range

- The operation *deftime* returns the times for which a moving object (seen as a function) is defined.
  
  \[
  \text{deftime} : \tau(\alpha) \rightarrow \text{periods} \quad \text{[periods} \equiv \text{type for interval sets]} \]

- The operation *rangevalues* returns the values assumed over time as a set of intervals.
  
  \[
  \text{rangevalues} : \tau(\alpha) \rightarrow \text{range}(\alpha) \quad \text{[1D]} \]

- The operation *locations* determines those components of the projection of a discretely moving point into the plane that form a *point* object.
  
  \[
  \text{locations} : \tau(\text{point}) \rightarrow \text{point} \]

- The operation *trajectory* determines those components of the projection of a continuously moving point into the plane that form a *line* object.
  
  \[
  \text{trajectory} : \tau(\text{point}) \rightarrow \text{line} \]
Spatiotemporal Operations (II)

- Operations for projection to domain/range (*continued*)
  - The operation *routes* returns the projection of a discretely moving line object.
  \[
  \text{routes} : \tau(\text{line}) \rightarrow \text{line}
  \]
  - The operation *traversed* returns the projection of a continuously moving line object.
  \[
  \text{traversed} : \tau(\alpha) \rightarrow \text{region} \quad \text{for } \alpha \in \{\text{line}, \text{region}\}
  \]
  - The operation *inst* yields the time component of a \((\text{time}, \text{value})\) pair.
  \[
  \text{inst} : \text{intime}(\alpha) \rightarrow \text{time}
  \]
  - The operation *val* yields the value component of a \((\text{time}, \text{value})\) pair.
  \[
  \text{val} : \text{intime}(\alpha) \rightarrow \alpha \\
  \text{[intime} = \text{time} \times \alpha]\]
Spatiotemporal Operations (III)

- Operations for interaction with domain/range
  - The operation *atinstant* restricts a moving object to a given time instant, resulting in a \((time, value)\) pair (time slice operator).
    \[
    atinstant : \tau(\alpha) \times time \rightarrow intime(\alpha)
    \]
  - The operation *atperiods* restricts a moving object to a given set of time intervals.
    \[
    atperiods : \tau(\alpha) \times periods \rightarrow \tau(\alpha)
    \]
  - The operations *initial* and *final* return the first and last \((time, value)\) pair, respectively.
    \[
    initial, final : \tau(\alpha) \rightarrow intime(\alpha)
    \]
  - The operation *present* checks whether a moving object exists at a given time instant, or is ever present during a given set of time intervals, respectively.
    \[
    present : \tau(\alpha) \times time \rightarrow bool
    \]
    \[
    present : \tau(\alpha) \times periods \rightarrow bool
    \]
Spa
ti
tem
to
ral Ope
ra
tions (IV)

- Operations for interaction with domain/range (*continued*)
  - The operation *at* restricts a moving object to a value in the range.
    
    \[ at : \tau(\alpha) \times \alpha \rightarrow \tau(\alpha) \text{ [1D]} \]
    
    \[ at : \tau(\alpha) \times \text{range}(\alpha) \rightarrow \tau(\alpha) \text{ [1D]} \]
    
    \[ at : \tau(\alpha) \times \text{point} \rightarrow \tau(\text{point}) \text{ [2D]} \]
    
    \[ at : \tau(\alpha) \times \beta \rightarrow \tau(\gamma) \text{ [2D]} \quad \text{with } \gamma = \text{if } \dim(\alpha) \leq \dim(\beta) \text{ then } \alpha \text{ else } \beta \]

- The operations *atmin* and *atmax* restrict a moving object to the times when it is minimal or maximal with respect to the total order on this space.
  
  \[ atmin, atmax : \tau(\alpha) \rightarrow \tau(\alpha) \text{ [1D]} \]

- The operation *passes* checks whether a moving object ever assumed (one of) the value(s) given as a second argument.
  
  \[ passes : \tau(\alpha) \times \beta \rightarrow \text{bool} \quad \text{with } \gamma = \text{if } \dim(\alpha) \leq \dim(\beta) \text{ then } \alpha \text{ else } \beta \]
Spatiotemporal Operations (V)

- Derivable spatiotemporal operations by temporal lifting: automatically derive spatiotemporal operations from spatial operations based on a clear semantics

- Examples
  - \( \text{dist: point} \times \text{region} \rightarrow \text{real} \)
    \[ \text{Dist: } \tau(\text{point}) \times \tau(\text{region}) \rightarrow \tau(\text{real}) \]
    \[ \text{Dist: } \text{hmpoint} \times \text{hmregion} \rightarrow \text{hmreal} \]
  - \( \text{intersection: line} \times \text{region} \rightarrow \text{line} \)
    \[ \text{Intersection: } \tau(\text{line}) \times \tau(\text{region}) \rightarrow \tau(\text{line}) \]
    \[ \text{Intersection: } \text{hmline} \times \text{hmregion} \rightarrow \text{hmline} \]
  - \( \text{area: region} \rightarrow \text{real} \)
    \[ \text{Area: } \tau(\text{region}) \rightarrow \tau(\text{real}) \]
    \[ \text{Area: } \text{hmregion} \rightarrow \text{hmreal} \]
Spatiotemporal Operations (VI)

Let \( f: \alpha_1 \times \ldots \times \alpha_n \rightarrow \beta \) be a spatial operation with \( \alpha_i \in \{\text{point, line, region}\} \) and \( \beta \in \{\text{point, line, region, int, real, bool}\} \). Its corresponding temporally lifted version, that is, its corresponding spatiotemporal operation is defined by

\[
\uparrow f: \tau(\alpha_1) \times \ldots \times \tau(\alpha_n) \rightarrow \tau(\beta)
\]

with

\[
\uparrow f(S_1, \ldots, S_n) := \{(t, f(S_1(t), \ldots, S_n(t))) \mid t \in \text{time}\}
\]

Examples

- \( \text{Dist} = \uparrow \text{dist} \)
- \( \text{Intersection} = \uparrow \text{intersection} \)
- \( \text{Area} = \uparrow \text{area} \)
Spatiotemporal Predicates (I)

- Problem: Temporal lifting of spatial predicates does not lead to predicates

- Example:
  - Spatial predicate: \( \text{inside} : \text{region} \times \text{region} \rightarrow \text{bool} \)
  - Temporally lifted spatial predicate:
    \( \uparrow\text{inside} : \tau(\text{region}) \times \tau(\text{region}) \rightarrow \tau(\text{bool}) \)

- A lifted spatial predicate yields a defined value only on the intersection of the domains of two moving objects.

- A spatiotemporal predicate is a function of type \( \tau(\alpha) \times \tau(\beta) \rightarrow \text{bool} \) for \( \alpha, \beta \in \{\text{point, line, region}\} \).
Spatiotemporal Predicates (II)

- **Basic spatiotemporal predicates**
  - defined by temporal lifting and temporal aggregation
  - each predicate has an own “preferred” temporal aggregation
  - Examples
    - *disjoint*: disjointedness required only on the common lifetime
    - *inside*: containment required only with respect to the lifetime of the first operand object
    - Define default expected aggregation behavior

- A spatiotemporal predicate describes the *continuous development* of the topological relationships of two moving objects over time.
Spatiotemporal Predicates (III)

- Example: airplane (red) crosses a hurricane (blue)
Spatiotemporal Predicates (IV)

Let $P$ be a basic spatiotemporal predicate, $S_1$ and $S_2$ be two moving objects, and $I$ be a (half-) open or closed time interval. Then predicate constriction is defined as

$$P_I(S_1, S_2) := P(S_1|_{I}, S_2|_{I})$$

Example (plane $F$, hurricane $H$)

- $\text{Inside}(F, H) = false$
- $\text{Inside}|_{t_2, t_3}(F, H) = true$

Two classes of topological predicates

- Instant predicates can be true for an instant time (but also for a period of time)
  - $equal$, $meets$, $covers$, $coveredBy$
- Period predicates can only hold for a period of time
  - $disjoint$, $overlap$, $inside$, $contains$
Spatiotemporal Predicates (V)

- A spatiotemporal predicate is an alternating temporal sequence of topological relationships that hold over time intervals or at time points.

- A spatiotemporal predicate is an alternating temporal sequence of topological relationships and basic spatiotemporal predicates.

- Symbol “>>” is temporal composition operator.

- Examples of spatiotemporal predicates for a moving point object and a moving region object:
  - \( \text{Enter} := \text{Disjoint} >> \text{meet} >> \text{Inside} \)
  - \( \text{Cross} := \text{Disjoint} >> \text{meet} >> \text{Inside} >> \text{meet} >> \text{Disjoint} \)
  - \( \text{Leave} := \text{rev(Enter)} \)
  - \( \text{Cross} := \text{Enter} >> \text{Leave} \)
  - \( \text{TempLeave} := (\text{Meet} >> \text{Disjoint})^* >> \text{Meet} \)
Spatiotemporal Predicates (VI)

- Examples of spatiotemporal predicates for two moving region objects

  - Touch := Disjoint >> meet >> Disjoint
  - Snap := Disjoint >> Meet
  - Release := Meet >> Disjoint
  - Bypass := Snap >> Release
  - Excurse := Meet >> Disjoint >> Meet
  - Into := meet >> Overlap >> coveredBy
  - OutOf := rev(Into)
Spatiotemporal Predicates (VII)

- Examples of spatiotemporal predicates for two moving region objects (continued)

  - \( \text{Enter} := \text{Disjoint} \gg \text{Into} \gg \text{Inside} \)
  - \( \text{Leave} := \text{rev}(\text{Enter}) \)
  - \( \text{Cross} := \text{Enter} \gg \text{Leave} \)
  - \( \text{Melt} := \text{Disjoint} \gg \text{meet} \gg \text{Overlap} \gg \text{Equal} \)
  - \( \text{Separate} := \text{rev}(\text{Melt}) \)
  - \( \text{Spring} := \text{equal} \gg \text{Overlap} \gg \text{meet} \gg \text{Disjoint} \)
  - \( \text{Graze} := \text{Disjoint} \gg \text{meet} \gg \text{Overlap} \gg (\text{CoveredBy} \gg \text{Overlap})^* \gg \text{meet} \gg \text{Disjoint} \)
Spatiotemporal Querying (I)

- Tables
  - CREATE TABLE Flight(id varchar(30), route hmpoint);
  - CREATE TABLE Weather(name varchar(30), kind varchar(50),
                            extent hmregion);

- Q1: What are the departure and arrival times of flight LH 257, and how long is the part of the route of this flight that lies within France?
  CREATE TABLE France AS
    (SELECT * FROM States WHERE sname = 'France');
  SELECT min(deftime(route)) AS departureTime,
         max(deftime(route)) AS arrivalTime,
         length(trajectory(Intersection(route, ^territory))) AS lenInFr
  FROM Flight, France
  WHERE id = 'LH 257';
Spatiotemporal Querying (II)

- **Q2**: When and where did flight LH 257 enter the territory of France?
  ```sql
  SELECT initial(at(route, territory)) AS entry,
         inst(entry) AS entryTime,
         val(entry) AS entryLocation,
  FROM Flight, France
  WHERE id = 'LH 257';
  ```

- **Q3**: Determine the time periods when snow storm ‘Lizzy’ consisted of exactly three separate areas.
  ```sql
  CREATE TABLE Lizzy AS
  (SELECT extent
   FROM weather
   WHERE name = "Lizzy" and kind = "snow storm");
  SELECT deftime(at(noOfComps(extent) = 3, TRUE))
  FROM Lizzy;
  ```
Spatiotemporal Querying (III)

Q4: Where was United Airlines flight 207 at time 8:00 am?

```
SELECT val(atinstant(route, 8:00)) AS locAtEight
FROM Flight
WHERE id = 'UA 207';
```

Q5: Where was United Airlines flight 207 between 7:00 am and 9:00 am?

```
SELECT trajectory(atperiods(route, 7:00..9:00)) AS loc
FROM Flight
WHERE id = 'UA 207';
```

Q6: Which planes ran into a hurricane?

```
SELECT id
FROM Flight, Weather
WHERE kind = 'hurricane' AND
route Disjoint >> meet >> Inside extent;
```
Spatiotemporal Querying (IV)

- DEFINE Enters AS Disjoint $\triangleright\triangleright$ meet $\triangleright\triangleright$ Inside;

- Q7: Which planes ran into a hurricane (reformulation)?

```
SELECT id
FROM Flight, Weather
WHERE kind = 'hurricane' AND
route Enters extent;
```

- Other definitions of complex spatiotemporal predicates

```
DEFINE Leaves AS rev(Enters);
DEFINE Crosses AS Enters $\triangleright\triangleright$ Leaves;
DEFINE Bypass AS Disjoint $\triangleright\triangleright$ Meet $\triangleright\triangleright$ Disjoint;
```
Spatiotemporal Querying (V)

Q8: What is the number of planes that were entering snow storms or fog areas?

SELECT count(*) as num
FROM Flight, Weather
WHERE kind = 'snow storm' OR kind = 'fog'
GROUP BY route Enters extent;

CREATE TABLE Bird(swarm varchar(40), movement hmpoint);

Q9: Which swarms fly together, then take different routes for some time, and finally meet again?

DEFINE Remeets AS _ >> Meet >> Disjoint >> Meet >> _;

SELECT A.swarm, B.swarm
FROM Bird AS A, Bird AS B
WHERE A.movement Remeets B.movement;
Outline

1. Introduction
2. Spatial Data Modeling
3. Spatiotemporal Data Modeling
4. Open Research Topics
Outline – Open Research Topics

- Topics not covered in this tutorial
- Research challenges
Topics Not Covered in this Tutorial

- Goal of the tutorial: Learn concepts first
- Data structure design for spatial and spatiotemporal data types
- Geometric algorithm design for spatial and spatiotemporal operations and predicates (require methods from Computational Geometry)
- Efficient implementation techniques for spatial and spatiotemporal joins
- Spatial and spatiotemporal query processing
- Spatial and spatiotemporal indexing
Research Challenges (I)

- Spatial partitions and spatial networks as *first class citizens* in spatial databases (Map Algebra)
  - Query languages for spatial partitions and spatial networks
Research Challenges (II)

- Vague and fuzzy spatial objects

- Some models for fuzzy spatial objects available (especially from GIS field)

- Problem: Implementation of fuzzy spatial objects
Research Challenges (III)

- Three-dimensional spatial objects

- Applications: Meteorology, GIS, soil science, earth science

- Problem: Computational Geometry is immature with respect to three-dimensional data handling
Research Challenges (IV)

- Moving objects (3D + time)

- Applications: Meteorology, hurricane research, tsunami research, earthquake research, volcano research

- Construction of moving objects from gridded sensor data (icing problem)
Research Challenges (V)

- STU objects (Space + Time + Uncertainty)

- Applications: Modeling of natural phenomena in general, geosciences, hurricane research, meteorology
Thank You for Your Attention and Interest!