Data Models and Query Languages for Spatiotemporal Databases

Jan Chomicki
Dept. CSE
University at Buffalo
State University of New York
http://www.cse.buffalo.edu/~chomicki

ICLP 2001 Workshop "Complex Reasoning on Geographical Data," December 2, 2001, Paphos, Cyprus.

Panta rei kai ouden menei.

(All is in flux; nothing abides.)

Heraclitus (540-480 B.C.)

Spatiotemporal phenomena

What is changing **where** and **how**.

What:

- $0D$ points
- $1D$ lines
- $2D$ regions
- $3D$ volumes.

Where:

- in $1D$ space (line)
- in $2D$ space (plane)
- in 3D space.

How:

- continuous **movement**
- continuous **evolution**
- discrete **evolution**
- birth, death, split, merge....

Examples

Transportation: truck or ship movement, airplane flights.

Natural disasters: oil spills, forest fires.

Ecology: species migration, habitat or land cover changes.

Climate: season or vegetation changes.

Society and economy: urban growth, land use changes, epidemics.

Ownership or administrative changes.

Plan of the talk

1. Abstract spatiotemporal objects
2. ADT-based approach
3. Constraint-based approach
4. Moving objects databases
5. Discussion

Abstract spatiotemporal objects

Time isomorphic to reals ($\mathbf{R}$).

n-dimensional abstract spatiotemporal object w:

$w \subseteq \mathbf{R}^n \times \mathbf{R}$.

It should be *well-behaved*:

- *Slice regularity*: $w_{t_0} = \{(x_1, \ldots, x_n) | (x_1, \ldots, x_n, t_0) \in w\}$ should be a "familiar" spatial object (point, region, polygon...)
- *Almost-continuity*: the function $f(t) = w_t$ has only finitely many discontinuity points in which it is still right-continuous

Closure

A class $\mathcal{C}$ of spatiotemporal objects is **closed** under an operation h of arity k if for every $w_1 \in \mathcal{C}, \ldots, w_k \in \mathcal{C}$, also $h(w_1, \ldots, w_k) \in \mathcal{C}$.

What operations:

- set-theoretic: intersection, union, difference
- temporal/spatial selection
- temporal/spatial projection.

Often easy to obtain for classes of "familiar" spatial objects, much harder (or even impossible) for natural classes of spatiotemporal objects.

Defining and querying spatiotemporal objects

ADT approach [Güting et al., 1997-]:

- specification: sufficiently expressive operation signatures
- representation: many possible
- closure needs to be established for every representation
- fits well with SQL3

Constraint database approach [Kanellakis et al., 1990-]:

- representation using first-order formulas
- closure: a property of the logical theory
- fits well with relational calculus and algebra

The ADT approach [Güting et al., TODS 2000]

Spatial objects:

- **point(s)**
- **line**: finite set of continuous curves
- **region**: finite set of bounded, connected subsets of the plane.

Operations:

- **set-thoretic**: intersection, union, difference
- **aggregate**: min, max, avg, center, area, volume,...
- **metric**: distance
- **topological** predicates: containment, adjacency,...
- **direction** predicates
- ...

Temporal lifting

Temporal types:

for every type τ with domain $A_\tau$the type $\mu[\tau]$ consists of partial functions $\mathbf{R} \to A_\tau$

Examples: $\mu(point)$, $\mu(region)$, $\mu(real)$.

An operation

$$h : \tau_1 \times \cdots \times \tau_k \to \tau_0$$

is lifted to

$$\uparrow h : \mu[\tau_1] \times \cdots \times \mu[\tau_k] \to \mu[\tau_0]$$

in a natural way:

$$\uparrow h(w_1, \ldots, w_k) = \{(t, h(w_1(t), \ldots, w_n(t)))|t \in \mathbf{R}\}.$$

Example SQL3 query

We will use the same name for an operation on spatial objects and its temporally lifted version.

"Find all pairs of planes that during the flight came closer to each other than 500 meters."

```
SELECT A.id, B.id
FROM Flights A, Flights B
WHERE A.id <> B.id
  AND minvalue(distance(A.route,B.route)) < 500
```

This is an example of a **spatiotemporal join**.

Intersection

A spatiotemporal join may require the construction of the **intersection** of two spatiotemporal objects.

Example.

Object o_1: the immediate proximity area around a moving ship.

Object o_2: a rock, a shallow or another natural hazard.

Object $o_1 \cap o_2$: the danger zone.

Concrete representation [Forlizzi et al., SIGMOD 2000]

Spatial objects:

- region ≡ finite set of **polygons with holes**

Spatiotemporal (evolving) objects:

- **sliced** representation (finitely many slices)
- the coordinates of moving points within a slice are **linear** functions of time
- segments may degenerate but **cannot rotate** within a slice
- spatiotemporal object ≡ **polyhedron**

ADT approach: conclusion

Representing **change**:

- continuous
- discrete (constant functions of time)

Closure:

- *union*: trivial
- *intersection*, *difference*, *selection*, *projection*: properties of polyhedra

Object-orientation:

- object types, operation signatures
- objects easy to implement in an object-relational data model but ...
- no natural restriction on movement that guarantees polyhedral structure
- not clear how to *specialize* spatiotemporal objects:
  - restricted regions: rectangles,...
  - restricted evolution: translations, isometries,...

Query language

SQL3:

- good *integration* with SQL3, e.g., aggregation
- *formal semantics*?
- extensions:
  - *temporal composition* of spatial predicates [Erwig& Schneider, TKDE]:

    `disjoint` $\triangleright$ `meets` $\triangleright$ `inside`

Query expressiveness:

- depends on the set of operations
- no formal *expressiveness model*:
  - how to show that a query *cannot be expressed*?
  - how to *compare* with other approaches?

Query optimization:

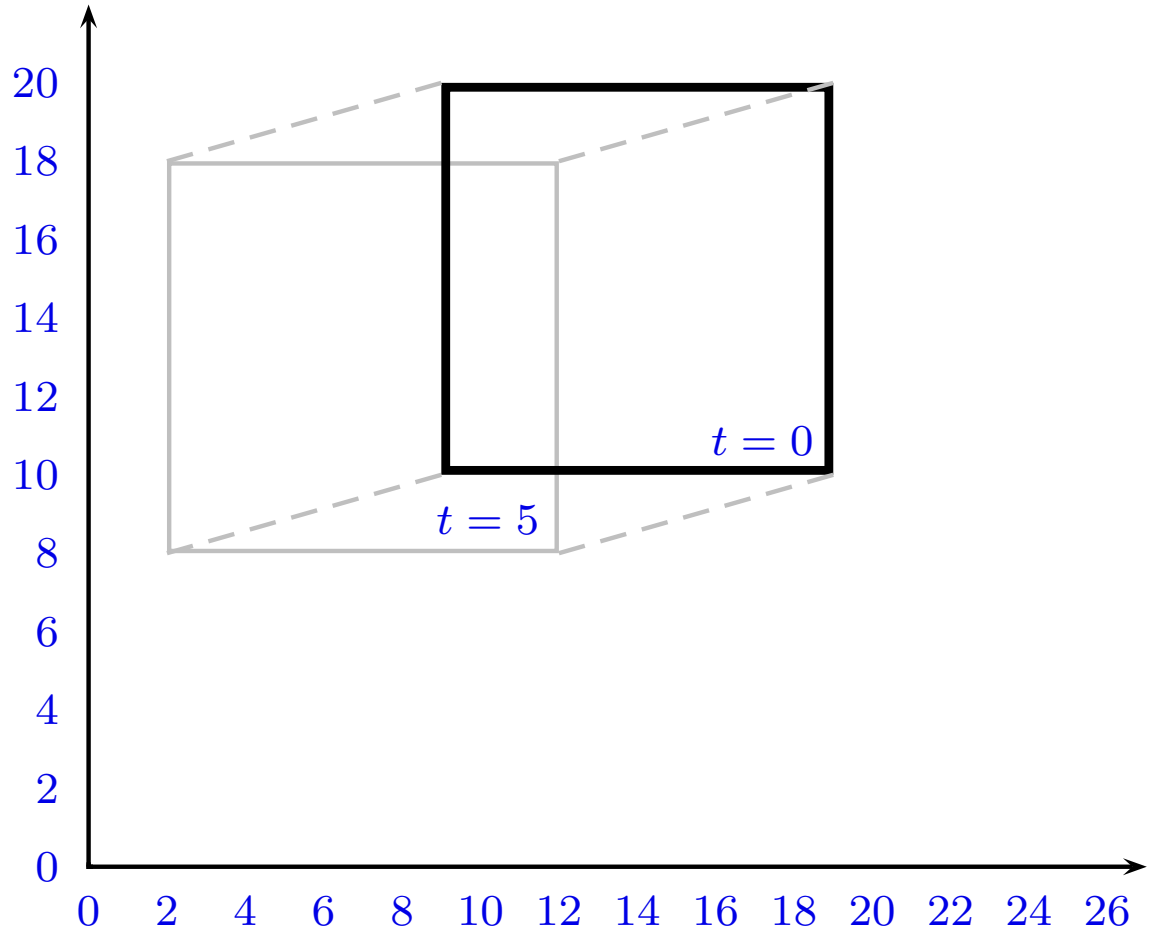
- queries with ADT operations notoriously *hard to optimize*.

Parametric spatiotemporal objects [Chomicki & Revesz, TIME 1999]

Atomic geometric object consists of:

- *reference spatial object*
- *reference time*
- *time domain* (interval)
- *space transformation function*, parameterized by time.

Molecular geometric object: a finite set of atomic objects with disjoint domains.


20
18
16
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
$t = 0$
$t = 5$

Transformation functions

Affine: $f(\bar{x}, t_0) = A_{t_0}\bar{x} + B_{t_0}$ where

$$A = \begin{bmatrix} f_1(t) & f_3(t) \\ f_4(t) & f_2(t) \end{bmatrix}, B = \begin{bmatrix} g_1(t) \\ g_2(t) \end{bmatrix}$$

Scaling:

$$A = \begin{bmatrix} f_1(t) & 0 \\ 0 & f_2(t) \end{bmatrix}$$

Translation:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

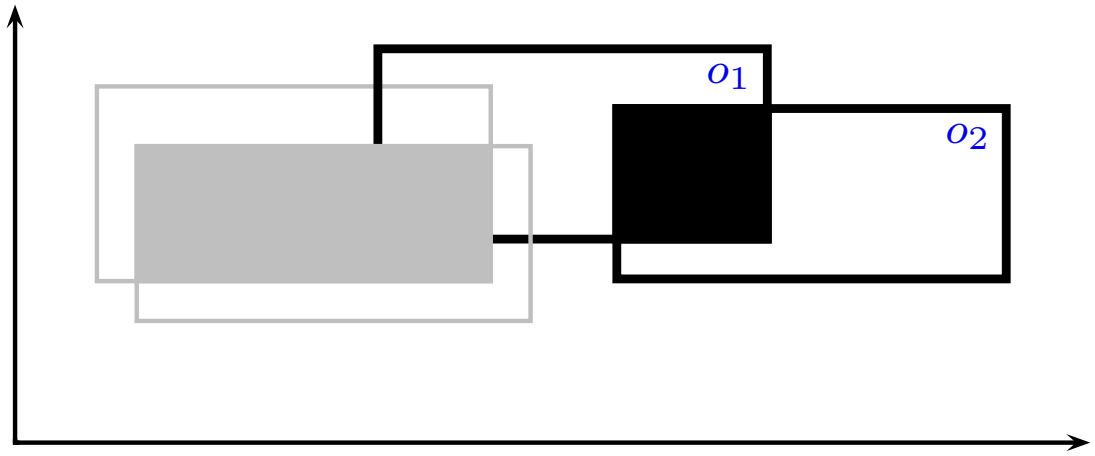
Identity.

Component functions: rational, polynomial, linear.

Rectangle intersection

Spatiotemporal objects defined by **rectangles** and **scaling** are closed under intersection:

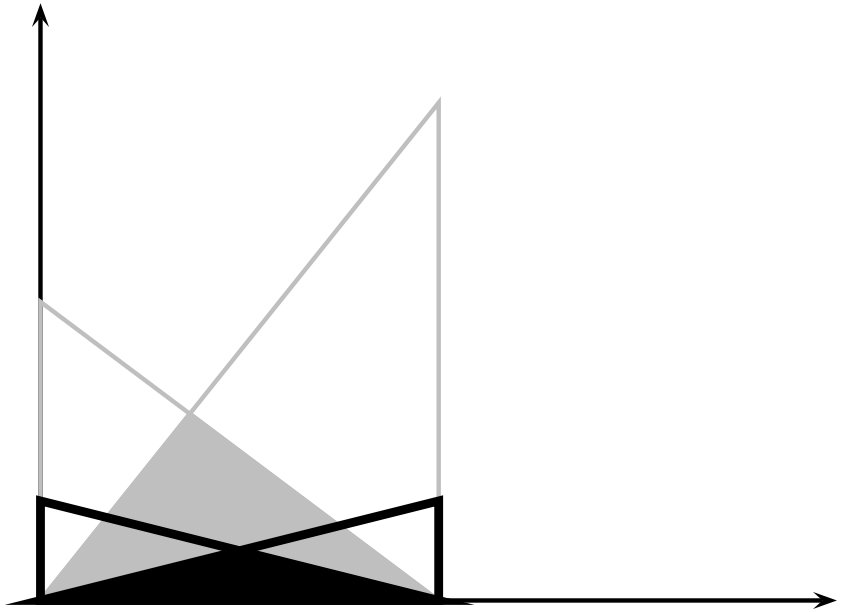
- linear scaling [Chomicki & Revesz]
- polynomial and rational scaling [Haesevoets & Kuijpers, TIME 2000]



Polygon intersection

Spatiotemporal objects defined by **polygons** and **scaling** are not closed under intersection:

- linear and polynomial scaling [Chomicki & Revesz]
- rational scaling [Haesevoets & Kuijpers].



Parametric objects: conclusion

Closure:

- union: trivial
- intersection, difference: some cases
- dealing with **non-closure**:
  - restrict objects to polyhedrons
  - construct objects as AND-OR-NOT trees (as in CSG)

Approach developed independently but may be viewed as a **refinement** of the ADT approach.

Constraint databases [Kanellakis, Kuper & Revesz, 1990]

Relational data model with **infinite, finitely representable relations**:

- constraint tuples: **conjunctions** of constraints
- constraint relations: finite sets of constraint tuples (interpreted as **disjunctions**).

Constraints (atomic formulas):

- polynomial or linear arithmetic inequalities
- order or periodicity constraints
- ...

Spatiotemporal objects:

- in 2D: constraint relations with polynomial or linear arithmetic constraints over x, y and t
- example:

$$t \geq 0 \wedge t \leq 5 \wedge x + y \leq t \wedge x \geq 0 \wedge y \geq 3$$

Relational algebra for constraint databases

Selection $\sigma_C(R)$: conjoin C with every constraint tuple in R.

Cartesian product $R \times S$: take $t \wedge s$ for every $t \in R$ and $s \in S$.

Union $R \cup S$: union of sets of constraint tuples R and S.

Difference $R - S$:

1. write down the complement of S as a formula
2. put it in disjunctive normal form
3. conjoin every resulting constraint tuple with every tuple in R in turn.

Projection $\pi_X(R)$: apply quantifier elimination to every constraint tuple in R in turn.

Constraint databases: conclusion

Closure:

- for all set-theoretic operations, projection and selection closure easily follows
- problems with aggregation, distance

Object-orientation;

- essentially untyped
- no natural notion of object, subclass etc.

Expressiveness:

- constraint relations with linear arithmetic constraints capture all polyhedra
- no exact correspondence with natural classes of parametric objects

Query languages:

- relational calculus and algebra, Datalog (termination!)
- simple formal semantics
- query expressiveness extensively studied
- implementation requires a special constraint engine.

Query expressiveness

Open issue:

Can every spatiotemporal SQL3 query be written as a constraint query and vice versa?

Arrival of spring query:

Find the regions where the spring arrived earlier than a year before.

Using constraints:

$$\exists t, t'.[t < t' < t + 365 \wedge S(t, x, y) \wedge \neg S(t - 1, x, y) \wedge S(t', x, y) \wedge \neg S(t' - 1, x, y)].$$

Using SQL3:

???

Variable independence

Constraint relations can express both **continuous** and **discrete** change but...

...the class of constraint relations expressing discrete change **is not easy to pin down.**

Independence of space and time:

> constraint relation has to be equivalent to one in which all the tuples are of the form
>
> $$\phi_1(x, y) \wedge \phi_2(t).$$

Important in practice: DÉDALE, SQLST.

The story of variable independence

1. [Chomicki, Goldin & Kuper, PODS'96]:
   - definition of *variable independence*
   - application to *"safe" aggregation*
   - proof (incorrect) of *decidability* of variable independence for linear arithmetic constraints
2. [Grumbach, Rigaux & Ségoufin, ICDT'99]:
   - notion of *orthographic dimension*, closely related to variable independence.
   - application to reducing the *complexity* of query evaluation
   - another proof (again incorrect) of *decidability* of variable independence for linear arithmetic constraints
3. [Libkin, ICALP'00]:
   - first correct proof of *decidability* of variable independence for polynomial and linear arithmetic constraints

Representing spatiotemporal phenomena using ADTs or constraints

Problems with **real spatiotemporal data**:

(A) data comes as **discrete observations**:

- within a snapshot (TINs)
- in different snapshots

(B) data lacks clearly **identifiable** and **delineated** objects

(C) data does not have regular (**polyhedral**) 3D structure

Solution to (A) and (C):

1. convert each snapshot to a set of polygons
   - intrasnapshot interpolation can be also expressed using constraints
2. interpolate/approximate between the snapshots.

$\Rightarrow$ the ADT or constraint approach more suitable to construct **approximations**.

Object definition:

- large **collections** of points
- complex **conditions**: $temperature > 32F$
- results of scientific analysis **programs**.

Will relational databases and relational query languages be still useful in that context?

$$\exists t, t'.[t < t' < t + 365 \wedge S(t, x, y) \wedge \neg S(t - 1, x, y) \wedge S(t', x, y) \wedge \neg S(t' - 1, x, y)].$$

Other challenges

Data models:

- *type* systems
- representing *uncertainty*: *speed between 40 and 60 mph*
- *integrating* different representations
- resolving *inconsistencies*

Query languages and interfaces:

- multidimensional aggregation (*spatiotemporal OLAP*)
- *visualization*
- *animation*:
  - *explicit* representations (ADTs) better than *implicit* ones (constraints)

Databases with moving objects

[Wolfson et al., 1997-; Su et al., 2001-].

Moving object:

- parametric spatiotemporal object
- reference spatial object is a *point*
- satisfies *motion continuity*
- component functions (*motion vector*) infinitely differentiable

Moving object database (MOD):

- finite set of moving objects
- the instant NOW

Querying MOD

Operations:

- location
- direction, distance, length,...
- spatial/spatiotemporal predicates
- speed, acceleration,...

Temporal dimension:

- queries about the *past*: *"where was truck #123 at 5pm yesterday?"*
- queries about the *present*
- queries about the *future*

Query languages:

- SQL3 [Forlizzi et al., SIGMOD 2000]
- relational calculus with built-in functions [Su et al., SSTD 2001]
- temporal logic [Wolfson et al., ICDE 1997]

Location issues

Uncertainty:

- object information may be out of date
- *certain/possible* query answers
- probability distributions

Updates:

- cost vs. imprecision tradeoff
- not practical to report every change to the motion vector: *a winding road*

Commercial technology: Qualcomm Omnitracs, Mobitrac.

Some outstanding issues in MOD

Modeling movement:

- **1.5D**: movement on fixed road networks
- what instead of precise location of an object it is enough to know whether it will arrive to some location by a certain **deadline**?

Query processing:

- queries with **uncertainty** factors
- instantaneous vs. **continuous** queries
- location **sampling**
- **conflict** resolution

A final look

MOD is now a separate research area:

- important practical applications
- specific technical issues: precision/uncertainty/probability
- specialized query languages
- specialized indexing techniques

Can the success of MOD be replicated?

Bottom-up approach:

- adding spatiotemporal constructs to existing GIS, in response to applications' demands
- problems with generality, interoperability etc.

Top-down approach:

- design a general model with clean semantics based, for example, on constraint databases
- will anyone use it in practice?

Adapt **general** query languages to a **broad spectrum** of spatiotemporal data.