

MODELING AND QUERYING HISTORY OF MOVEMENT: OVERVIEW

Modeling and Querying History of Movement

Basic abstractions in spatial databases: point, line, region.

- **Point**: An entity for which only the position in space is relevant.
- **Region**: An entity for which also the extent is relevant.
- **Line**: Usually ways for moving, connections in space.

We focus on *moving point* and *moving region*.

Moving points: people, animals, stars, cars, planes, ships, ...

Moving regions: forests, forest fires, oil spills, countries, hurricanes, tribes, armies, ...

Support all kinds of queries about such moving objects.

An Approach Based on Abstract Data Types

Database: collection of object classes. *Objects* have associated (attribute) *values* of arbitrary data types (generic view valid in all data models, e.g. relational, object-relational, object-oriented, ...).

Objects

- are born
- die

Perhaps

- may be reborn
- “redie”

repeatedly.

→ Associate set of validity (time) intervals with objects.

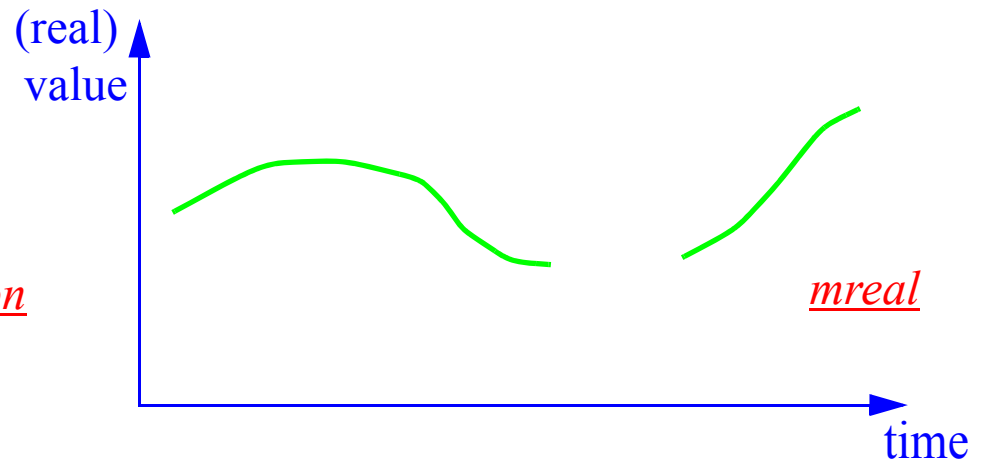
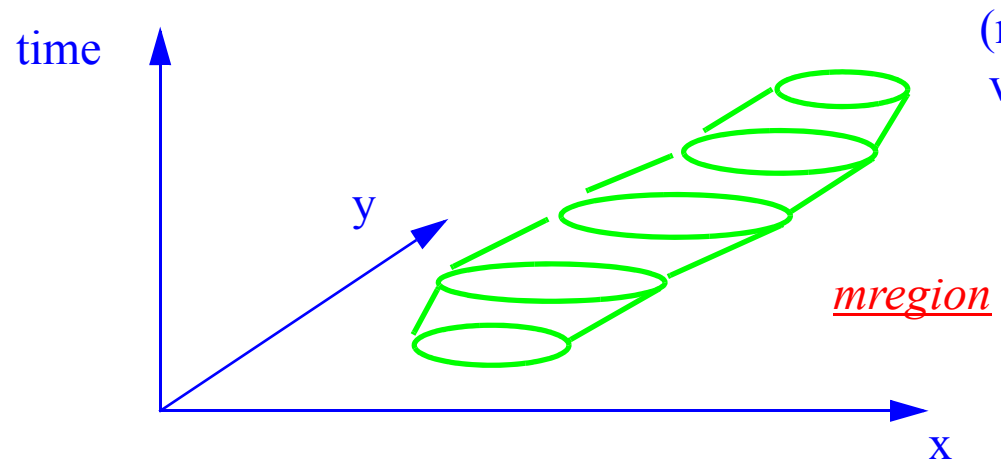
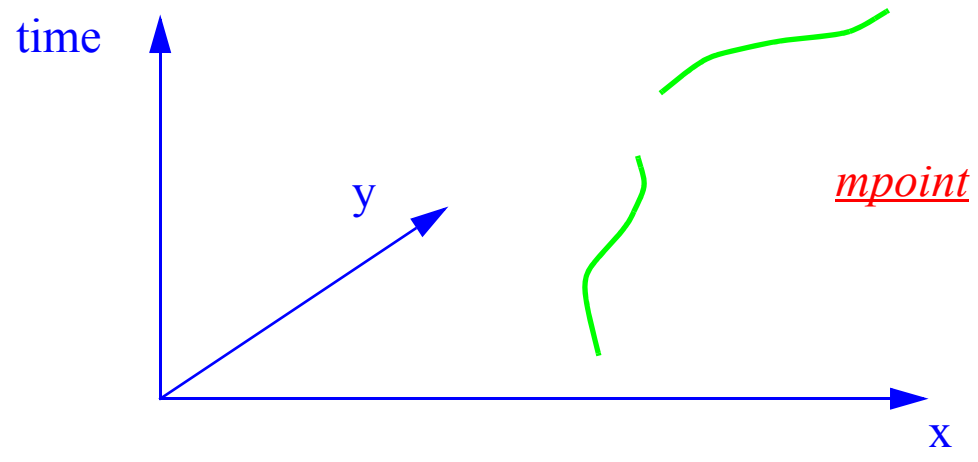
Alternative approach based on Abstract Data Types:
use a standard model and work just with ADTs.

Overview of Types and Operations

Values: Spatio-temporal + (1D-temporal) data types

- $\underline{mpoint} = \underline{time} \rightarrow \underline{point}$
- $\underline{mregion} = \underline{time} \rightarrow \underline{region}$ continuous changes
- $\underline{mreal} = \underline{time} \rightarrow \underline{real}$

- $\underline{mint} = \underline{time} \rightarrow \underline{int}$
- $\underline{mbool} = \underline{time} \rightarrow \underline{bool}$ discrete changes
- $\underline{mstring} = \underline{time} \rightarrow \underline{string}$



Overview of Types and Operations (cont.)

Some Operations

$\underline{moving}(\alpha) \times \underline{time}$	$\rightarrow \alpha$	atime
$\underline{moving}(\alpha)$	$\rightarrow \alpha$	minvalue, maxvalue
$\underline{moving}(\alpha)$	$\rightarrow \underline{time}$	start, stop
$\underline{moving}(\alpha)$	$\rightarrow \underline{real}$	duration
α	$\rightarrow \underline{moving}(\alpha)$	const

$\underline{mpoint} \times \underline{mpoint}$	$\rightarrow \underline{mreal}$	mdistance
$\underline{mpoint} \times \underline{mregion}$	$\rightarrow \underline{mpoint}$	intersection

\underline{mpoint}	$\rightarrow \underline{line}$	trajectory
$\underline{mregion}$	$\rightarrow \underline{region}$	traversed
$\underline{point} \times \underline{region}$	$\rightarrow \underline{bool}$	inside
\underline{line}	$\rightarrow \underline{real}$	length

Overview of Types and Operations (cont.)

Example Queries

flights (id: *string*, from: *string*, to: *string*, route: *mpoint*)

“Give me all flights from Düsseldorf that are longer than 5000 kms.”

```
SELECT id  
FROM flights  
WHERE from = "DUS" AND  
      length(trajectory(route)) > 5000
```

“Which destinations can be reached from San Francisco within 2 hours?”

```
SELECT to  
FROM flights  
WHERE from = "SFO" AND duration(route) <= 2.0
```

Overview of Types and Operations (cont.)

Example Queries (cont.)

“Find all pairs of planes that during their 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(mdistance(A.route, B.route)) < 0.5
```

```
airport (code: string, location: point)  
weather (kind: string, area: mregion)
```

“Which flights went through a snow storm?”

```
SELECT id  
FROM flights, weather  
WHERE kind = "snow storm" AND  
  duration(intersection(route, area)) > 0
```

Overview of Types and Operations (cont.)

Example Queries (cont.)

“Which airports were affected by snow storms?”

```
SELECT DISTINCT from  
FROM airport, weather  
WHERE kind = "snow storm" AND  
    inside(location, traversed(area))
```

“Which airports were *most* affected by snow storms?”

```
SELECT code,  
    SUM(duration(intersection(const(location), area)))  
    AS storm_hours  
FROM airport, weather  
WHERE kind = "snow storm"  
GROUP BY code  
HAVING storm_hours > 0 ORDER BY storm_hours
```

Overview of Types and Operations (cont.)

Implementation concept:

ADT extension package (“datablade”) to an extensible (OR, OO) DBMS.

In particular, representations of attribute values may be of varying and arbitrary size.

Work program:

- Design a system of types and operations (a many-sorted algebra)
which level of abstraction?
- Design data structures for the types and algorithms for the operations
- Implement DBMS extension package

Abstract vs. Discrete Models

In any case: Design a many-sorted algebra. Two steps:

1. Invent a signature (types + operations)

sorts **operators**

2. Define semantics: Carrier sets for the sorts, functions for the operators

Abstract Continuous Infinite	Concrete Discrete Finite
A <u>region</u> is a closed subset of \mathbb{R}^2 with non-empty interior	A region is a set of polygons with holes
A <u>line</u> is a curve in \mathbb{R}^2 , i.e., a continuous function $f: [0, 1] \rightarrow \mathbb{R}^2$	A <u>line</u> is a polyline (e.g., a list of line segments)

Abstract Continuous Infinite	Concrete Discrete Finite
A moving point (<i>mpoint</i>) is a function from time into <i>point</i> values	A moving point is a polyline in 3D space. ... is a cubic spline in 3D
An <i>mreal</i> is a function from time into <i>real</i> values	... is a piece-wise quadratic polynomial
A moving region (<i>mregion</i>) is a function from time into <i>region</i>	... is a polyhedron in 3D ... is a sequence of affine mappings

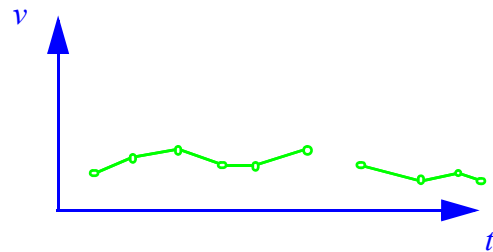
Abstract Model: Definitions in terms of infinite sets allowed. Don't care about finite representation.

Discrete Model: Only definitions in terms of finite representations are allowed.

Abstract vs. Discrete Models (cont.)

Difficulties in Discrete Modeling

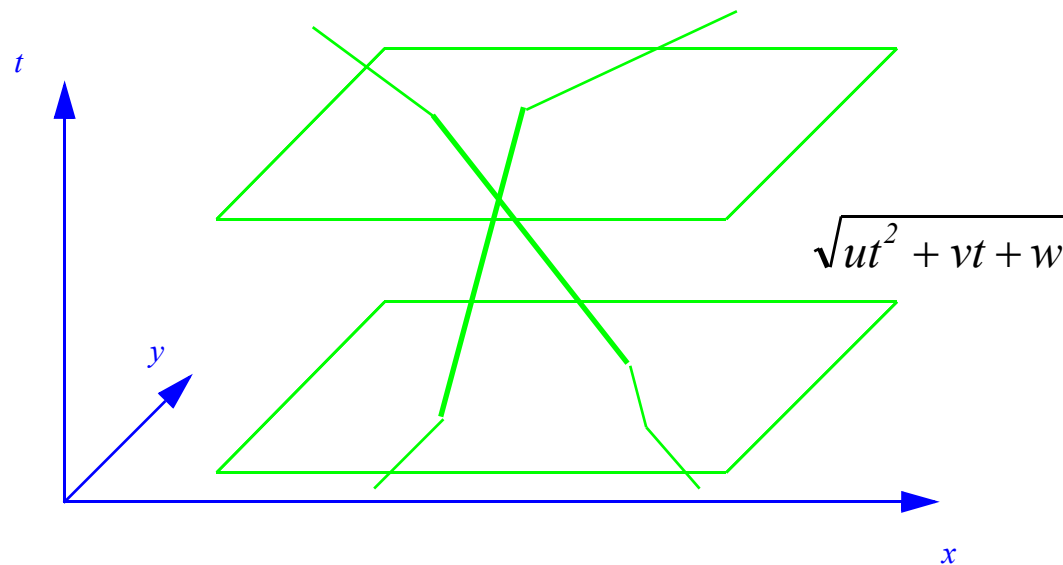
Model for *mreal*?



Now consider the operation:

$$\underline{mpoint} \times \underline{mpoint} \rightarrow \underline{mreal} \quad \mathbf{mdistance}$$

Difficulties in Discrete Modeling (cont.)



Find a (piece-wise) fixed size representation for *mreal* that is closed under **derivative**!

Representations for *mregion*:

- *polyhedral* model can represent arbitrary growth/shrinking and changes of shape, but can only very badly represent rotation of fixed-shape regions
- *sequence of affine mapping* model can represent rotation of fixed shape regions cleanly, but not non-linear growth or intersection with other regions.

Abstract vs. Discrete Models (cont.)

Abstract models are

- + simple, uniform
- + generic (admit several discrete implementations)
- not (directly) implementable

Discrete models

- are more complex, heterogeneous
- represent specific choices not suitable for all applications
- + offer a direct implementation

Conclusion: Proceed in two steps.

1. Design *abstract model* as a target.
2. Design one or more *discrete models* as instantiations of the abstract model (and then implement these).

Language Embedding of Abstract Data Types

Some facilities assumed available in DBMS query language:

Assignments

```
LET <name> = <query>
```

Multistep queries

```
<assignment>; ... <assignment>; <query>
```

Conversion between sets of elements and atomic values

```
ELEMENT (<query>)
```

```
SET (<attrname>, <value>)
```

```
ELEMENT(SELECT salary FROM employee WHERE name = 'John Smith') * 2
```

```
SET(name, 'John Smith')
```

name
'John Smith'

Defining derived values

```
LET <name> = <functional expression>  
<functional expression> ::= FUN <parameter list> <expression>  
LET square = FUN (m:int) m*m;
```

Defining aggregate functions

```
LET <name> = AGGREGATE(<operator>, <neutral element>)  
  
employee(name: string, salary: int, permanent: bool)  
  
LET all = AGGREGATE(and, TRUE);  
SELECT all(permanent);
```

For example, defining the union of all regions in a relation:

```
country(name: string, area: region)  
  
union: region x region -> region  
  
LET sum = AGGREGATE(union, TheEmptyRegion);  
LET Europe = ELEMENT(SELECT sum(area) FROM country)
```