1

Toward Spatiotemporal Patterns

Martin Erwig

School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon 97331, USA, erwig@cs.orst.edu

Abstract Existing spatiotemporal data models and query languages offer only basic support to query changes of data. In particular, although these systems often allow the formulation of queries that ask for changes at particular time points, they fall short of expressing queries for sequences of such changes.

In this chapter we propose the concept of spatiotemporal patterns as a systematic and scalable concept to query developments of objects and their relationships. Based on our previous work on spatiotemporal predicates, we outline the design of spatiotemporal patterns as a query mechanism to characterize complex object behaviors in space and time. We will not present a fully-fledged design. Instead, we will focus on deriving constraints that will allow spatiotemporal patterns to become well-designed composable abstractions that can be smoothly integrated into spatiotemporal query languages.

Spatiotemporal patterns can be applied in many different areas of science, for example, in geosciences, geophysics, meteorology, ecology, and environmental studies. Since users in these areas typically do not have extended formal computer training, it is often difficult for them to use advanced query languages. A visual notation for spatiotemporal patterns can help solving this problem. In particular, since spatial objects and their relationships have a natural graphical representation, a visual notation can express relationships in many cases implicitly where textual notations require the explicit application of operations and predicates. Based on our work on the visualization of spatiotemporal predicates, we will sketch the design of a visual language to formulate spatiotemporal patterns.

1.1 Introduction

Our world is changing at an increasing pace causing the validity intervals of information to shrink. Therefore, the *change* of information becomes important information itself—an important information resource that should be exploitable by query languages. In particular, information about the position and extent of objects is of high interest in many areas of science and elsewhere (vehicle tracking, history, security, health care, ecology, weather prediction, urban planning, and so on). In all these areas relevant information has to be extracted from huge data sets. Beyond data mining that often serves the purpose of identifying interesting objects in spatial or spatiotemporal data sets, queries are of interest that find out about relationships between objects once they have been identified. In the realm of spatio*temporal* databases this means to find out, in particular, about the *changes* that objects and their relationships undergo. Moreover, many applications require not only the detection of one single change, but rather look for sequences of changes describing particular *developments*.

Existing spatiotemporal database systems and query languages offer only basic support to query changes of data. Most of these systems allow the formulation of queries that ask for changes at particular time points. However, it is often very difficult, if not impossible, to express queries for sequences of such changes. In other words, existing query languages do not offer a systematic, scalable concept to query developments of objects and relationships; instead they require the user to encode these by a number of individual conditions. This method is awkward, in particular, because it also requires the formulation of additional side conditions. Moreover, this approach does not work in queries for arbitrary numbers of changes.

A concept of *spatiotemporal patterns* and their integration into spatiotemporal query languages can help to close this gap. Spatiotemporal patterns enable the formulation of queries about complex object developments; their integration into query languages allows the formulation of queries that are currently not possible. Moreover, spatiotemporal patterns can also simplify the formulation of queries that are possible with existing languages.

Beyond the integration of spatiotemporal patterns into a textual query language we will also discuss the design of a visual query language and a corresponding user interface to support the formulation of spatiotemporal patterns and queries. This is of particular importance because most users of spatiotemporal data (for example, scientists) do not have a formal computer education and do not know how to use a query language, in particular, query languages for complex data like spatiotemporal data. Moreover, the fast growing set of available spatial and spatiotemporal data and its wide dissemination through the Internet increases the class of possible users for these data. Offering ordinary users access to spatiotemporal data is therefore becoming a more and more important issue that should be addressed by developing a visual query language and a corresponding user interface.

In the remainder of this chapter we first motivate the need for spatiotemporal patterns in Section 1.2. We discuss related work on spatiotemporal data models and query languages in Section 1.3. We review the notion of spatiotemporal objects and their integration into databases in Section 1.4. Spatiotemporal predicates are described in Section 1.5. Spatiotemporal patterns will be discussed in Section 1.6, and the design of a visual query language is sketched in Section 1.7. Finally, we present some conclusions in Section 1.8.

1.2 The Need for Spatiotemporal Patterns

Queries about changing objects are becoming more and more important. One reason might be that the world is changing at a constantly increasing pace so that information about a situation at a particular instant in time has a decreasing half-life. In other words, the intervals for which certain pieces of information remain valid become shorter and shorter. Therefore, the *change* of information becomes important information itself—an important information resource that should be exploitable by query languages. In the area of temporal databases this issue has been addressed for quite some time now [4, 60, 63].

Spatiotemporal Data

A particularly important kind of information change is that of objects' locations and/or spatial extensions. Many phenomena in all areas of natural sciences involve the change of spatial information. A few example are given below.

- Meteorology: all kinds of weather data, moving storms, tornados, developments of high pressure areas, movement of precipitation areas, changes in freezing level, droughts, El Niño effects, ...
- Biology: animal movements, mating behavior, species relocation and extinction, ...
- Crop sciences: seasonal grasshopper infestation, harvesting, soil quality changes, land usage management, ...
- Forestry: forest growth, forest fires, hydrology patterns, canopy development, planning tree cutting, planning tree planting, ...
- Medicine: patients' cancer developments, supervising developments in embryology, ...
- Geophysics: earthquake histories, volcanic activities and prediction, ...

A characteristic feature of most of these examples is that changes are of continuous nature. Moreover, many human-related activities are related to changes in spatial information as well:

- People: movements of terrorists/criminals/spies, movement of people in emergency situations, pedestrian patterns/habits, ...
- Cars/trucks/taxis: tracking, rerouting, fleet management, ...
- Urban planning: parcel management, development of social areas, urban economics, tourism planning, bus routes, ...
- Crime/disaster prevention: risk area analyses, resource allocations (police, health care, fire stations), ...
- History: country expansions, reunifications, tribe movements, ...
- Military: missile tracking, troop movements, ...
- Planes/ships: routes, detours, ...

- Ecology: causal relationships in environmental changes, tracking down pollution incidents, ...

In particular, since mobility is constantly increasing, we can observe an increase in the time dependency of spatial information related to human activities. Many of the second group of examples represent continuous changes, some are discrete in nature.

In addition to the shown individual areas, combinations of phenomena are also of interest, for example, what changes in forests can be linked to which kind of animal behavior, which weather developments are responsible for grasshopper infestation, etc. Moreover, some combinations pose particular planning challenges, for example, extreme weather events require rerouting of cars, planes, and ships.

Spatiotemporal Patterns

The shown examples reveal one important characteristic of change in spatial data that makes it interesting for data processing by scientists and others: interesting phenomena are those that are not random but rather follow certain rules. In other words, applications are interested in spatial data changes that exhibit a certain regular structure.

Regular structures in space and time, in particular, repeating structures, are often called *patterns*. Patterns that describe changes in space and time are referred to as *spatiotemporal patterns* [48].

Not surprisingly, the notion of spatiotemporal patterns occurs in many different areas of nature [44] and sciences [17], in particular, in geosciences [49] and geophysics [57], meteorology [46, 68], ecology [73] and environmental studies [70], and even embryology [65]. A collection of applications in ecology and related sciences and the challenges they pose for computer science has been gathered during a recent NSF workshop [45]. An area of growing importance is that of crime detection and prevention, in which spatiotemporal patterns also play an important role, for example, in crime-pattern analysis with the help of GIS [72, 14, 53], tracking criminals [52], and crime prevention [37].

To make the following discussion concrete, we use a couple of example queries that could arise in different application areas. Here we will comment on the requirements that a data model and query language must have to support these applications. In Section 1.3 we will discuss how most of the existing approaches fall short of supporting these requirements in some way or another. The first example relates to the environment.

> **Q1.** Find all satellites that were able to observe an oil spill at least four times for at least 5 minutes each time.

This innocent looking query demands a lot from a data model and query language. For example, in addition to moving points the data model must also support evolving *regions*. Moreover, the movement of the satellite and the oil spill are *continuous*.

Next we consider some examples of spatiotemporal queries that can be relevant for crime prevention. Suppose we want to find out suspicious movements of, say cars. With regard to security of the public mail system we might want to pose the following query.

> **Q2.** Find all cars that stopped at two or more mailboxes in a short period of time.

Why would anyone *not* put all their mail into the first mailbox? Therefore, the above query gives hints for suspicious behavior. To answer this query a language must be able to express *repetitions* of *patterns* of *changes*, for example, in speed or topological relationships (here: meeting a mailbox). Moreover, the language must be able to express *time conditions* on the duration of certain predicates. Another example is to identify suspicious behavior near sites of high public interest (like Disneyland):

> **Q3.** Find cars that surrounded an area of interest multiple times or at a significantly less than average speed.

In addition to the need for a repeated pattern, this query requires the possibility to express a constraint on an object's movement by another object's spatial attribute (here, boundary).

A spatiotemporal database system should be able to answer queries like these. However, as we will discuss next, most existing systems and language proposals are rather limited in their support for querying spatiotemporal patterns and cannot express queries as the ones shown.

Limitations of Existing Approaches

One goal of spatiotemporal databases research is to enable the intelligent use of the collected data. In addition to the challenges of efficient storage and indexing spatiotemporal data [50, 1], a main problem is that of extracting relevant and useful information from the data. Two complementing areas of database research address two different aspects of this problem.

(1) *Data mining* [16], which is also sometimes referred to as *knowledge discovery* [32], is concerned with the discovery of patterns in large data sets. Data mining often uses statistical analyses and methods to extract interesting constellations/arrangements of points in a usually very huge data space. In particular, spatiotemporal data mining [69, 6, 51] has the goal of identifying interesting structures and patterns in spatiotemporal data sets. In many cases, spatiotemporal data mining is used to identify spatiotemporal objects. Once found, these objects of interest can the be stored in a database where they can be processed further. In GIS systems the so-called *field view* [55, 36] of spatial and spatiotemporal data is used. Data mining techniques can be used to move

from this low-level field view of raw data to the higher-level view of objects. This conception was prevailing at a recent NSF workshop on spatiotemporal data models for scientific applications [45].

(2) *Query Languages* work on data that is well structured and that is stored in rather fixed formats obeying schema definitions and possibly additional integrity constraints. The distinctive feature of query languages (compared to data mining) is the assumption that the knowledge in a database is already present, whereas the goal of data mining is to discover it. Query languages are used to find out further relationships among objects.

Although considerable research has been performed in recent years on data models and query languages, most of them are not capable of expressing the example queries shown above.

What are the reasons for this situation? We believe that there are two major problems of data models and query languages that are responsible for the gap between research and applications: (1) Lack of expressiveness and (2) high complexity, which has a particularly bad impact on the usability.

Problem 1: Lack of Expressiveness. There have been quite a few proposals for spatiotemporal query languages. However, most languages are able to express only a small fraction of the queries that are possible on spatiotemporal data and that are needed to solve application problems like the ones indicated above. For example, data models that are only concerned with moving objects, such as [62, 71], cannot express queries that involve changing regions, such as Q1. Models that are restricted to discretely changing objects, such as [10, 74, 54, 9, 41, 18], cannot express queries about continuous objects; again, query Q1 relies on exploiting the continuous nature of the involved objects. None of the existing data models can express queries about changing relationships of objects in a satisfactory way, as it is required for query Q2. In particular, queries like Q3 that ask for a repetition of certain patterns are either impossible to express or lead to incredibly complex ad hoc solutions. Although constraint databases promise a declarative and expressive approach to spatiotemporal databases [12], actual approaches to spatiotemporal query languages have been described only in [39, 41, 11]. The DEDALE system [39, 41] can due to its embedding of constraint objects into relations only deal with discretely changing spatiotemporal objects. The proposal of Chomicki [11] is based on constraints in the $n + 1$-dimensional space (n space dimension plus one time dimension) and corresponds more closely to the field view, so that it is not really clear whether applications involving many attributes (spatial and others) can be modeled well. In any case, the constraint approach like all other proposals lacks abstractions to express patterns of changes.

A possible route for a solution to the last problem is given by the concept of *developments* [24] and *spatiotemporal predicates* [27], described in Section 1.5, which form the basis for spatiotemporal patterns, discussed in Section 1.6.

Problem 2: Complexity and Lack of Usability. A data model and query language for spatiotemporal data must be necessarily more complex than, for example, SQL, because SQL is a language for querying flat relations over atomic data, whereas spatiotemporal databases have to deal with spatial data (point, lines, and regions) and their possible changes. It seems that the goal of simplicity contradicts the goal of expressiveness because one might think in order to express more advanced queries one needs more and complex operations in the query language. Indeed, extending, for example, relational algebra by spatial data types, increases the number of operations significantly because the added data types require their own operations [42]. This is even more the case when considering more complex data types for spatiotemporal objects [22]. Apparently, the constraint approach does not suffer from an inflation of operations. However, it is not immune to an increase in complexity because the domains over which constraints are formulated become more complex [31]. Although the vocabulary of constraint languages remains very much unaffected, queries grow complex, too, because without a mechanism to express patterns of changes, the individual changes have to be spelled out in detail, which leads to an inflation of conditions comparable to the inflation of joins found in relational approaches.

The problem of usability is of growing importance because the fast growing number of data sources and the raising availability through the Internet causes the target group of possible users also to grow. For many of these possible users languages like SQL are already too complex since most of them do not have formal computer science training at all; the requirement to learn a formal language with keywords, binding rules, etc. is in many cases too high a barrier.

Visual query languages and visual query interfaces to spatiotemporal databases can help solving this problem. Visual query languages are particularly promising for spatial databases because spatial data has a direct and intuitive visual representation that can be exploited for formulating queries in a more direct way. As we will discuss in Section 1.7, we can also obtain a representation of spatiotemporal data that is effective for expressing spatiotemporal queries in a simple and direct way.

1.3 Related Work

Some early models for spatiotemporal data have been proposed starting in the mid-90s. In [74] a spatial data model has been generalized to become spatiotemporal. Spatiotemporal objects are defined as so-called *spatio-bitemporal complexes* whose spatial features are described by simplicial complexes and whose temporal features are given by bitemporal elements attached to all components of simplicial complexes. On the other hand, temporal data models have been generalized to become spatiotemporal and include variants of Gadia's temporal model [34] which can be found in [10, 7]. In [66] a discrete snapshot model is described. The main drawback of all these approaches is that they are incapable of modeling *continuous* changes of spatial objects over time. An extension of the snapshot model toward capturing continuous changes has been described by Yeh and Cambray [75, 76] who introduce the notion of *behavioral time sequences.* In this approach, which is based on the work described in [59], each element of such a sequence contains a geometric value, a date, and a behavioral function, which describes the evolution up to the next element of the sequence.

The constraint-based approach to modeling spatiotemporal data considers spatiotemporal objects as point sets in a multi-dimensional space [40]. The description of objects is given by logical formulas (constraints). Moreover, queries are also expressed by logical formulas. An embedding of constraints into a relations is performed in [39] where also a SQL-like query language is proposed. In other work [12] spatiotemporal objects are represented by spatial objects plus affine geometric transformations to describe their evolution. Although the logical/constraint-based approach is very general, it is not clear, for example, whether it can be efficiently implemented. Other possible problems are the handling of objects that require non-linear constraints and the treatment of metric operations (for example, computing distances).

A logic-based approach to model predicates on (spatio)temporal data is offered through temporal logic. In particular, Allen's work [3] is centered around the definition of a predicate $\mathit{Holds}(p, i)$ that asserts that a property p is true during a time interval i. Combinations of predicates can be constructed through logical predicates and quantifiers. The work is based on his earlier work [2] where he has identified thirteen possible relationships between intervals, such as *before*, *equal*, *meets*, *overlaps*, and *during*. Allen's temporal logic is solely based on time intervals and does not include time points. Galton [35] has extended Allen's approach to the treatment of temporally changing topological relationships. Topological relationships are based on the RCC model [15, 56]. In contrast to Allen, Galton also takes time points into account. In specifying changes of spatial situations he uses the notion of a *fluent* or *state*, which can have different values at different times. Whereas Galton takes into account time points, he does not consider geometric points. We believe that it is important to have spatial as well as time points available in a data model. First, spatial points are needed in many applications. Second, having also spatial points provides some form of symmetry to the existence of points in the time domain. Third, spatiotemporal values can be projected to spatial values by function application at time points. This feature is absolutely essential and much needed in spatiotemporal query languages.

Our approach is based on (1) the notion of *temporal objects* [30] and (2) the concept of *spatiotemporal data types* [21, 22, 43].

The notion of a *temporal object* is based on the observation that anything that changes over time can be expressed as a function over time. A temporal version of an object of type α is then given by a function from *time* to α. Spatiotemporal objects like *moving points* and *evolving regions* are regarded as special instances of temporal objects where α is a spatial data type like *point* or *region*. Spatiotemporal objects are encapsulated in data types that offer tailor-made operations for them, which facilitates their integration as complex values into databases [64]. The ADT approach has several advantages. First, employing ADTs is more expressive than relying on attribute-timestamped models, let alone tuple-timestamped models, since continuous changes can be modeled [22]. Second, a definition of ADT values is valid regardless of a particular DBMS data model and query language. The reason is that ADT values are *not* modeled by concepts of a DBMS data model and that they therefore do not depend on them. This facilitates an easy and elegant conceptual integration of ADT values into relational, complex-object, object-oriented, or other data models and query languages.

ADT operations can be used within queries to construct new, derived spatiotemporal objects. In particular, boolean-valued operations can be used as predicates to formulate conditions in queries. However, like for all other existing query language approaches, this approach is basically suited to express queries about changes at single time points, so that a combination of conditions is required to express multiple changes. What makes this worse is the fact that, in general, additional side conditions are needed in such queries to express temporal ordering of the formulated change events.

Our work on spatiotemporal predicates [27] enables more general queries about developments [28, 24]. The idea is to specify and query changes in topological spatial relationships. Using point-set topology as a formal framework, the 9-intersection model [19] provides a canonical collection of topological predicates for each combination of spatial types. Based on the nine topologically invariant intersections of boundaries, interiors, and exteriors of the two participating objects, these predicates are mutually exclusive and cover each possible topological situation between two objects. The relevance of spatiotemporal predicates is demonstrated in [24] where we have identified an important new class of spatiotemporal queries, which is concerned with developments of spatial objects over time, which means queries ask especially about changes in spatial relationships over time. A macro mechanism is provided which allows the user to build more and more complex spatiotemporal predicates starting with a small set of elementary ones. We have demonstrated how these concepts can be realized in a relational model and how SQL can be appropriately extended to a spatiotemporal query language called *STQL* to enable the querying of developments [28].

Our work on spatiotemporal predicates has also inspired other researchers to consider the idea of spatiotemporal patterns. In [18] an approach is sketched where the developments of objects can be queried using a regular expression over one predicate. Although the approach is interesting, the expressiveness is rather limited. First, only discretely changing objects are considered. Second, since regular expressions can be formed only over one predicate at a time, queries that ask for a development described by two different predicates (such as, *Inside meet Disjoint*, see below), *cannot* be expressed. This means that the described approach is strictly less expressive than spatiotemporal predicates as defined in [27]. Finally, the approach does not support any of the additional expressiveness, such as time constraints on the duration of predicates, that is possible in the approach presented here. On the other hand the work of [18] and the described application provide a strong evidence that spatiotemporal predicates form a solid basis for the development of spatiotemporal patterns.

We have addressed the problem of user-friendly specifications of temporally changing topological situations and predicates in [25]. For this purpose, we have defined a *visual language* which extends existing concepts for visual spatial query languages, which are only capable of querying *static* topological situations. In [26] the design of the visual language is motivated in detail and a user interface is defined. The visual notation is generalized to deal with undefined parts of object developments in [29]. That paper also defines the translation into our formal model of spatiotemporal predicates. The visual notation can be used directly as a visual query interface to spatiotemporal databases, or it can provide predicate specifications that can be integrated into textual query languages leading to heterogeneous languages [23].

1.4 Spatiotemporal Objects and Databases

Spatial data types like *point*, *line*, and *region* have turned out to be a fundamental abstraction for modeling the two-dimensional structure of geometric entities, their properties, relationships, and operations [42, 58]. Points are elements of the Euclidean plane. Lines are two-dimensional curves. Regions describe point sets with a two-dimensional extent and are bounded by lines which in this context are called *boundaries*. To avoid anomalies and ensure well-definedness of operations formal definitions of spatial data types are often based on point-set topology [33] and on regularization functions [67].

Spatiotemporal objects are obtained by considering the changes of spatial objects over time. They can be conveniently described by *temporal functions* [30, 21], which are simply functions from time into the corresponding spatial domain. For example, the movement of a satellite (projected onto the earth's surface) can be regarded as a function $Sat : time \rightarrow point$, and the development of an oil spill can be regarded as a function $Spill : time \rightarrow region$ [20]; see Figure 1.1. We can apply functions like *Sat* or *Spill* to time values and obtain the position of the satellite or the extent of the oil spill at a particular time as spatial data.

Of course, *spatial* temporal data is just a particular example of temporal data in general. For example, we can also deal with temporally changing numbers through a type like $time \rightarrow num$. Such a temporal number could give the size of the oil spill depending on the time. Using the notational convention that temporal versions of data types have their letter capitalized, we have, for example, $Sat : Point$ and $Spill : Region$. We can define functions on spatiotemporal objects, for example, a function *Area* that computes the time-dependent area of an evolving region:

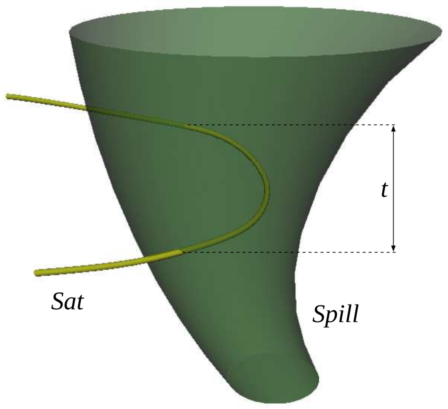

Figure 1.1. Satellite movement and oil spill development.

$$Area : Region \rightarrow Num$$

If we apply *Area* to *Spill*, we obtain a time-dependent number reflecting the temporal development of the size of the oil spill. The design of these spatiotemporal data types has been described in [22]. A categorization and formal definition of operations has been performed in [43].

By encapsulating (spatio)temporal data in data types like *Point*, *Region*, and *Num* we can directly integrate them into data models, such as the relational data model. For example, we can define two relations containing information about satellites and pollution incidents as follows.

$$\begin{array}{l} Satellites(sname : string, Pos : Point) \\ Pollutions(pname : string, Reg : Region) \end{array}$$

The relation *Satellites* contains for each satellite one tuple consisting of the satellite's name and its movement, which is given by the attribute *Pos* that stores a moving point. Similarly, an evolving region is associated with each pollution incident through the attribute *Reg* in the relation *Pollutions*. The temporal changes are completely encapsulated by the types *Point* and *Region*. The operations offered by these data types enable us to pose spatiotemporal queries in a straightforward way. For example, we have an operation *Intersection* that computes the intersection of two spatiotemporal objects. In the case of a moving point and an evolving region this intersection is given by those parts of the moving point that lie inside the evolving region. As a query example we ask which satellites were able to observe which pollution incidents for a certain amount of time by the following query formulated in the SQL-like language STQL [20, 28].

```
SELECT sname, pname
  FROM Satellites, Pollutions
 WHERE duration(Intersection(Pos, Reg)) > 3 hours
```

The query works as follows: For all pairs of satellites and pollution incidents we consider the intersection of their spatiotemporal attributes and select those pairs for which the lifetime (indicated by t in Figure 1.1) of the moving point resulting from the intersection is larger than 3 hours. In queries we use dimension conversion operators like $\mathit{hours} : \mathit{num} \rightarrow \mathit{duration}$ in postfix notation. The type *duration* is isomorphic to the real numbers, but it contains values describing the duration of time intervals. The *duration* function computes the total duration of the domain of a temporal function.

1.5 Generalizing Spatiotemporal Predicates

Although many interesting queries can be expressed by the approach described in the Section 1.4, there are also many queries that cannot be (easily) formulated. One large class of such queries are queries asking for changes in relationships between objects. For example, we might be interested in finding ships that were inside the oil spill and eventually left the spill. (This query could point to the possible polluter.) Using a spatiotemporal intersection predicate yields also all ships that entered, crossed, touched, etc. the spill and is simply not precise enough to find the desired information. Although we can express the query by adding a couple of further explicit conditions on the ship's trajectory (see for example, [28]), the query becomes incredibly complex and almost unreadable. Moreover, the situation gets much worse for more complex developments, which indicates that the available query language abstractions are not expressive enough. The same observation holds for other data models and query languages. For example, in SQL^{ST} [9] queries suffer from an inflation of joins.

What are Spatiotemporal Predicates?

With the introduction of *spatiotemporal predicates* [24] we can describe precisely the developments of objects and their relationships in a relatively simple way. A (binary) spatiotemporal predicate P is a function that maps a pair of spatiotemporal objects to a boolean value, that is, $P : \alpha \times \beta \rightarrow \mathit{bool}$ for $\alpha, \beta \in \{\mathit{Point}, \mathit{Region}\}$. An example is the predicate *Inside* (defined for the two types $\mathit{Point} \times \mathit{Region} \rightarrow \mathit{bool}$ and $\mathit{Region} \times \mathit{Region} \rightarrow \mathit{bool}$) that checks whether the first argument is always inside the evolving region. Another example is *Disjoint* that yields true for two spatiotemporal objects that have no points in common. Complex spatiotemporal predicates can be built by regular expressions over a basic set of spatial and spatiotemporal predicates. For example, the predicate describing a "leaving" development can be defined as:

$$\mathit{Leaves} := \mathit{Inside}\ \mathit{meet}\ \mathit{Disjoint}$$

The *meet* predicate is a spatial predicate that applies to two spatial objects and yields true when they touch each other. The semantics of a "string" of spatiotemporal and spatial predicates is formally defined in [27]. Intuitively, the meaning is that one predicate has to hold after the other. In other words, for a sequence of alternating spatiotemporal and spatial predicates $P_1\, p_1\, P_2\, \ldots\, p_{k-1}\, P_k$ to hold for an object (pair), there has to be a connected sequence of time intervals and instants $I_1, t_1, I_2, \ldots, t_{k-1}, I_k$ such that each spatiotemporal predicate P_j holds for I_j and each spatial predicate p_j holds at t_j. Hence, *Leaves* is true for two spatiotemporal objects P_1 and P_2 if for some time P_1 was inside P_2, then touched the border, and finally was disjoint from P_2. In addition to predicate "concatenation" there are many other predicate compositions, such as alternation, reflection, and repetition, which have nice properties (proved in [27]) that can be used, for example, for optimization.

Assume we have the following schema for the relation *Ships*.

$$\mathit{Ships}(\mathit{sname} : \mathit{string}, \mathit{Pos} : \mathit{Point})$$

With the spatiotemporal predicate *Leaves* we can formulate the query for ships leaving the oil spill as follows.

```
SELECT sname
  FROM Ships, Pollutions
 WHERE pname = 'The Big Spill' AND Pos Leaves Reg
```

This example demonstrates how spatiotemporal predicates offer a basic abstraction for expressing conditions on sequences of changes.

Spatiotemporal Predicates Based on Numerical Operations

The concept of spatiotemporal predicates is based on the notion of changes in *topological* relationships between spatiotemporal objects. A logical next step is the generalization to *numerical* predicates. For example, with spatiotemporal predicates we can currently not formulate a query that asks for regions (like oil spills) that after a period of growth were only shrinking (to check, for example, a successful treatment).

The modular, two-step definition of spatiotemporal predicates indicates possibilities for generalizations: A spatiotemporal predicate P is obtained from a spatial predicate p by (1) *lifting* it to a function $\uparrow p$ that computes for two spatiotemporal objects a temporal boolean that gives the predicate value for each time point and (2) aggregating all the boolean values using $\forall$ quantification into one boolean value. For example, consider the spatial predicate $\mathit{inside} : \mathit{point} \times \mathit{region} \to \mathit{bool}$ that tells whether or not a point is inside a region. The lifted function $\uparrow\mathit{inside} : \mathit{Point} \times \mathit{Region} \to \mathit{Bool}$ computes for a moving point and an evolving region a temporal boolean that tells for each time point whether or not the point is inside the region. The temporal aggregation of all these boolean values yields true if the point was always inside the evolving region.[1]

If we now consider numerical functions, such as *area* or *distance*, their lifted versions $\uparrow area$ and $\uparrow distance$ yield temporal numbers, which can be aggregated in two ways into booleans. First, we can apply a lifted numerical numerical predicate (such as a function checking for "> 3"). This results in a temporal boolean that can be aggregated further by $\forall$ as already described. With that approach, however, we are not able to check, for example, growth since boolean values are obtained for individual time values, independently from other time values. However, we have to compare the numbers of different time values in order to identify a growth trend in time. This can be performed by an operator Δ that maps any temporal object of type α (if a total ordering is defined on α) to an object of type $\{negative, none, positive\}$ that gives for each time the change in the value at that point. Thus, Δ is basically a differential operator. Now, "growth" can be checked by comparing the result of Δ to *positive* and aggregating with $\forall$. We can generalize spatiotemporal predicates to work for numerical functions by parameterizing the $\forall$ quantification by a function that maps non-boolean values into booleans. This function will be lifted and applied to the temporal object just before the $\forall$ aggregation is performed. For example, to define a predicate for checking the growth of a particular *Region* value R, we have to map R into a temporal number by applying *Area* and then determining the change at each time point by Δ. Next we can define a predicate *grows* as follows.

$$grows(x) :\Longleftrightarrow x = positive$$

This predicate can be applied to all values returned by Δ and aggregated with $\forall$. We combine the last two steps in the definition of a parameterized *Always* operation that takes two parameters: (i) the predicate to be applied (here *grows*) and (ii) the spatiotemporal object to which the first parameter expression is applied (here: $\Delta(Area(R))$). Then we find out whether or not R always grows through the expression:

$$Always(grows, \Delta(Area(R)))$$

The precise definition of *Always* has to take into account different lifetimes of objects, which is discussed in detail in [27]. Another issue to investigate is the result type of Δ. For some applications the type $\{negative, none, positive\}$ is not general enough. It seems that we rather need Δ to be of type $\{negative, none, positive\} \times \alpha$. Then, for numerical functions Δ gives the change rate (for example, speed). However, for spatial types the situation is more involved. For example, when we apply Δ to an evolving region, we might get marked regions *and* curves as a result.

[1] There are some subtle details regarding the time intervals over which the aggregation has to range. These considerations are not important here; a detailed analysis can be found in [27].

Adding Time Constraints

Another limitation of spatiotemporal predicates is demonstrated by the query Q1: While we can express the multiple-time-observation condition with the help of repetition and the definition of a *Cross* predicate, the minimum time constraint *cannot* be expressed by the current form of spatiotemporal predicates. Similarly, the query Q2 about suspicious mailbox visitors requires a constraint on the duration of the validity of a spatiotemporal predicate.

One possible way to extend spatiotemporal predicates by temporal constraints is to define an operation that can superimpose these constraints on spatiotemporal predicates. Such an operation, say *For*, would take a predicate on durations and would restrict a spatiotemporal predicate further by this condition. If STP denotes the type of all spatiotemporal predicates (that is, $STP = \alpha \times \beta \rightarrow \mathit{bool}$ for $\alpha, \beta \in \{\mathit{Point}, \mathit{Region}\}$), then *For* has the type $\mathit{For} : (\mathit{duration} \rightarrow \mathit{bool}) \times STP \rightarrow STP$. With this operation we can define two spatiotemporal predicates for expressing the minimum-observation-time requirement and the minimum-number-of-visits requirement, respectively.

$$\begin{aligned}
&\mathit{IntersectFor}(d, P, Q) := \mathit{For}(\geq d, \mathit{Intersection})(P, Q)\\
&\mathit{AtLeastFour}(P) := P\ (\neg P)\ P\ (\neg P)\ P\ (\neg P)\ P^{+}
\end{aligned}$$

Note that we are using partial function application as a convenient means to formulate duration predicates. For example, the expression "$\geq d$" is a short form for the function f that is defined by $f(x) = x \geq d$. The advantage of using expressions like "$\geq d$", in which the $\geq$ function is only partially applied to its second argument, is that we do not have to introduce a name for that function (for a detailed discussion, see [5]). Moreover, note that *For* is a *higher-order* predicate, that is, it takes two predicates as arguments and returns a predicate as a result. This fact also explains the notation $\mathit{For}(\geq d, \mathit{Intersection})(P, Q)$: $\mathit{For}(\geq d, \mathit{Intersection})$ is the application of *For* to its two argument predicates, and the result is a predicate that is applied to the pair of objects (P, Q).

The alternating sequence of predicates and their negation in the definition of *AtLeastFour* is required to describe the repeated satisfaction of P on arbitrary but non-overlapping time intervals. Using the two predicates we could express query Q1 as follows.

```
SELECT sname
  FROM Satellites, Pollutions
 WHERE pname = 'The Big Spill' AND
       AtLeastFour(IntersectFor(5 min, Pos, Reg))
```

With the *For* operator we can also express query Q2. Assume that we have two relations *Cars* and *Mailboxes* with the following schemas.

$$\begin{aligned}
&\mathit{Cars}(\mathit{license\text{-}plate} : \mathit{string}, \mathit{Location} : \mathit{Point})\\
&\mathit{Mailboxes}(\mathit{id} : \mathit{string}, \mathit{pos} : \mathit{point})
\end{aligned}$$

Since mailboxes usually do not move, their position is represented by a spatial point and not a moving point. To express the spatiotemporal condition of "meeting", we have to lift the mailbox points into moving points that actually do not move. We can use the already mention lifting operation "↑", which, when applied to objects and not functions, yields a constant temporal function that returns the argument object for each time point. For example, for a given point $p : \mathit{point}$, the expression $\uparrow p$ (of type *Point*) denotes a stationary position. The query Q2 can now be expressed as follows.

```
SELECT  license-plate
  FROM  Cars, Mailboxes
 WHERE  For(≤ 4 hours, AtLeastTwo(For(≥ 2 min, Meet)(Location, ↑pos)))
```

This query works as follows. The innermost *For* refines the *Meet* predicate to hold for at least two minutes. The *AtLeastTwo* requires an occurrence of this predicate at least twice. Finally, the outermost *For* constrains this whole predicate to hold only if the total time is less than four hours. In this example, we observe the need for another operation *AtLeastTwo* or a generalization *AtLeast* that takes an integer parameter. We will discuss a generalization of these function in Section 1.6.

We can also see that the definition of *For* is not general enough. To express more complex conditions to refine spatiotemporal predicates it seems that *For* needs access to the spatiotemporal objects that are under consideration by the predicates to be refined. Then we can express constraints on speed (which is required for the query Q3) and other object properties. Moreover, we also have to consider versions of *For* that take a predicate on time instants and constrains spatial predicates (when used inside developments).

1.6 Spatiotemporal Patterns

The extended form of spatiotemporal predicates outlined in Section 1.5 can be used to define a language of spatiotemporal patterns.

One limitation of spatiotemporal predicates is demonstrated by the definition of the predicate *AtLeastFour*, which shows that a more detailed control over repetition is needed to express developments/queries that mention an explicit number of repetitions. This is one of many ways to compose individual predicates into *spatiotemporal patterns*.

As far as repetition is concerned, we can define in addition to the Kleene star operation a combinator *Repeat* that takes a number k and a spatiotemporal predicate P and returns a spatiotemporal predicate that is true if the k-fold composition/sequence of P is true. It is interesting to observe that although we can easily express a combinator *AtLeast* as a derived predicate based on *Repeat*, namely simply by

$$\mathit{AtLeast}(P, k) := \mathit{Repeat}(P, k)\ P^*$$

it is not possible to define the combinator *AtMost* (without using recursion).

Sequential composition, repetition, and alternation (see [27]) of spatiotemporal predicates are just three simple example combinators to express structure for different relationships among spatiotemporal objects. In more advanced applications we might have to find out about more complex relationships among different parts of a development. The following query illustrates this aspect.

Q4. Which forests suffered from multiple fires of increasing duration?

We can express the multiple-fire constraint by an *AtLeast* operator similar to what we have done for Q1 and Q2. However, the second part of the query refers to the duration of the predicates, but not just with a condition that is evaluated for each predicate individually, but rather with a condition that relates different subpredicates. Here, it is the condition that the duration of any intersection predicate is greater than the duration of the preceding one. Since the *For* combinator is only able to constrain individual predicates, we need a more general mechanism.

On possibility is to allow aggregations over spatiotemporal predicate sequences, that is, their values and time periods. By working in a generalized model of numerical predicates, values and time periods can be exposed as part of the evaluation of a spatiotemporal predicate as a sequence of (pairs of) objects and the corresponding predicate whose evaluation on the objects has contributed to the overall result. In the example query Q4 this would be a list of forest/fire pairs plus the predicates *Intersection*, $\neg$*Intersection*, *Intersection*. (Here we assume for simplicity to having just one fire object.) Then we can imagine a function that scans this list, computes the relevant information for each element (here, the duration of the first and third object pairs), and performs an aggregation, which in this case could be simply a test whether the computed list of durations is increasing.

This approach generalizes the notion of spatiotemporal predicates from mere predicates that yield a boolean value to object filters or object partitioning functions. It is the generalization of the result type that "opens" spatiotemporal predicates for further computations. Two open questions in this context are:

- How can the notion of spatiotemporal predicates be generalized without compromising their current useful applications?
- What is a suitable language to specify aggregations over spatiotemporal predicate results?

Eventually, this generalization might lead to hierarchical sequences since multiple aggregations could follow one another offering the opportunity for another aggregation. In other words, spatiotemporal patterns provide a way to describe hierarchical or nested spatiotemporal predicates. One way of describing such hierarchies is to use a grammar formalism where terminals represent basic spatiotemporal predicates and nonterminals represent spatiotemporal patterns that can be expanded according to the grammar rules into arbitrarily nested expressions of spatiotemporal predicates. Evaluating such a general form of spatiotemporal patterns means to parse predicate sequences according to a grammar that defines a pattern. The current version of spatiotemporal predicates result as a special case by not using recursion in grammars, which leads to regular expressions.

Another issue to consider is the ability to existentially quantify objects used in development queries. Under the current interpretation, a spatiotemporal predicate is a binary predicate that can be applied to two spatiotemporal objects and evaluates to either true or false. For example, a spatiotemporal predicate that is able to express query Q4 can succeed only for a forest and *one particular* fire that repeatedly affected that forest. However, the query is rather concerned with the forest being affected repeatedly by *any* fire, that is, it should be possible to satisfy the condition by different fires. One possibility to allow for such a generalization is to introduce local quantifiers (existential and universal) into the language for specifying predicates. Nonquantified variables can be assumed to be universally quantified by default to make this a conservative extension. Another possibility is to introduce on the conceptual level "relation sequences" so that instances for a predicate can be chosen independently from different relations.

In summary, spatiotemporal patterns seem to be a logical next step following spatiotemporal predicates in the definition of more expressive query predicates. Their integration into spatiotemporal query languages, such as STQL, can extend the expressiveness of such query languages considerably.

1.7 Visual Query Interfaces

Spatiotemporal predicates allow the formulation of many queries in a rather direct way by expressing spatiotemporal relationships directly on attributes of the involved objects. However, as we have seen in Section 1.6, more advanced queries that require spatiotemporal patterns quickly tend to become quite complex. Moreover, the fast growing set of available spatial and spatiotemporal data and its wide dissemination through the Internet increases the class of possible users for these data. Offering ordinary users access to spatiotemporal data is therefore becoming a more and more important issue that can be addressed by developing a visual query language and a corresponding user interface. In particular, it is important to make databases accessible to users without formal training in databases and query languages, such as scientists, administrators, or other end users. In many cases, these users do not have the time or are not willing to learn a formal query language. From this point of view, even spatiotemporal predicates embedded into a query language like STQL [24, 28] is beyond what most of these users can deal with.

A visual notation for spatiotemporal predicates can help solving this problem. In particular, since spatial objects and their relationships have a natural graphical representation, a visual notation can express relationships in many cases implicitly where textual notations require the explicit application of operations and predicates. A similar situation arises in the case of spatiotemporal objects when the time dimension is visualized as one spatial dimension [25]. However, special care must be taken to keep the notation simple enough to avoid overspecifications by exposing too many details. The idea of our approach is to represent traces of object developments by two-dimensional pictures. The relationships of the drawn objects can then be interpreted as spatiotemporal predicates. An example is shown in Figure 1.2 that specifies the *Leaves* predicate. (The interpretation of pictures like these and their formal translation into spatiotemporal predicates is described in detail in [29].)

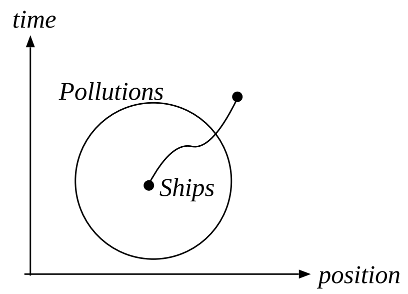

Figure 1.2. Visual Specification of the *Leaves* predicate.

Additional improvements and simplifications can be achieved by embedding the query notation in a tailor-made user interface that supports the creation and adaptation of queries by direct manipulation of the objects whose relationships represent spatiotemporal predicates and patterns [26]. Moreover, by selecting attributes from a database schema and associating them with the drawn objects, pictures can not only define spatiotemporal predicates, but also complete queries. For example, if the moving point in Figure 1.2 is associated with the *Pos* attribute of the *Ships* relation and the circle is associated with the *Reg* attribute of the *Pollutions* relation, the picture describes the following query and can be automatically translated into it.

```
SELECT sname, pname
  FROM Ships, Pollutions
 WHERE Pos (Inside meet Disjoint) Reg
```

Concreteness and directness are two strategies that have been used successfully by some end-user programming languages [8], and they should be employed in visual query interfaces for spatiotemporal databases as well. Concreteness has to do with working with values instead of with abstract variables. The strategy of directness is particularly important in our design plans. The term *directness* as used in the HCI community expands on the term *direct manipulation*, first coined by Shneiderman to describe three principles: continuous representation of the objects of interest, physical actions or presses of labeled buttons instead of complex syntax, and rapid incremental reversible operations whose effect on the object of interest is immediately visible [61]. Hutchins, Hollan, and Norman expand on these notions, suggesting that directness is inversely proportional to the cognitive effort needed to use the interface [47]. They describe directness as having two aspects. The first aspect is the distance between one's goals and the actions required by the system to achieve those goals. We argue that distance is small in our approach since the objects to be queried are displayed and directly manipulated on the screen. For example, the goal "describe a ship leaving a region" is expressed by a picture in which the trace of the ship start inside the region and ends outside of it, instead of requiring a textual expression involving predicate and operator applications. Green and Petre enumerate several examples showing the unfortunate lack of this aspect of directness (termed *closeness of mapping* in their work) in commonly used programming languages [38]. The second aspect is a feeling of direct engagement: "the feeling that one is directly manipulating the objects of interest". The notion of aiming for directness as a programming language design goal has in recent years begun to influence other kinds of end-user programming languages and domain-specific languages as well. In our approach direct engagement is supported by the user interface, which lets users place objects on a drawing region and move them around to create traces. Simultaneously to the user action, the spatiotemporal predicate that is specified through the current picture is displayed in a small area of the screen. This enables the user to move objects and observe at the same time the effect on the specified predicate. An additional benefit is that users can learn about the formal, textual notation by examples if they want to.

To extend the developed visual query notation (and the user interface) to cope with more general forms of spatiotemporal predicates and patterns, a number of issues have to be addressed.

One example is to add conditions to query pictures. Consider, for example, a refinement of the visual query in Figure 1.2 that is obtained by adding a constraint on the time the ship needed to leave the pollution region. We can express this by projecting two points onto the time axis and adding a condition to it. The resulting visual query asks only for those ships that have left the region within 30 minutes, see Figure 1.3.

This visual query will then be translated into the following STQL query.

$$\begin{array}{rl}\text{SELECT} & \mathit{sname}, \mathit{pname} \\ \text{FROM} & \mathit{Ships}, \mathit{Pollutions} \\ \text{WHERE} & \mathit{Pos}\ (\mathit{For}(<30\ \mathit{min}, \mathit{Inside})\ \mathit{meet}\ \mathit{Disjoint})\ \mathit{Reg}\end{array}$$

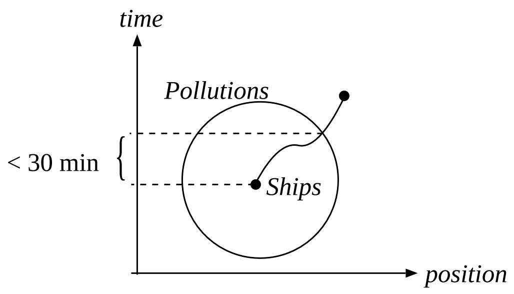

Figure 1.3. Refining the *Leaves* predicate with a condition.

Two other useful extensions are to express numerical spatiotemporal predicates and to allow for hierarchical specifications. To use numerical spatiotemporal we need additional visual elements to refer to numerical spatiotemporal operations like *Distance* to be able to express numerical constraints on developments. In order to be able to express general spatiotemporal patterns that require nested sequence conditions we need a mechanism to refer to visual specifications from within other specifications. Nesting of visual expressions can sometimes simplify the notation considerably [13]. Coupled with zooming capabilities on the interface level, this approach can lead to convenient specification of hierarchical predicates and patterns.

1.8 Conclusions

We have introduced a concept of spatiotemporal patterns that can be employed in query languages to express queries about the development of spatiotemporal objects and their relationships. To help coping with the complexity of the query language we also have sketched the design of a visual query language that makes the expressiveness of spatiotemporal patterns accessible to a large group of users.

We believe that providing effective access to spatiotemporal data through expressive query languages is an important step to help scientists and ordinary users to exploit the information that is contained in fast growing spatiotemporal data sets. Moreover, we believe that the concept of spatiotemporal patterns can provide a key to solving this problem. A group of researchers at the recent NSF workshop on "Workshop on Spatio-Temporal Data Models of Biogeophysical Fields for Ecological Forecasting" has concluded the same thing [45] (emphasis added):

> [...] there is a critical need for theories and tools that will enable efficient and reliable characterization of *spatio-temporal patterns* contained in image time series.

The general goal of the described work can be summarized by the term "universal geo-data access". First, the developed query language concepts can help to improve the ability to extract information from large geo-data sets. Second, through the development of visual query interfaces, the access to geo data is provided to a large group of users.

References

1. P. K. Agarwal, L. Arge, and J. Erickson. Indexing Moving Points. In *19th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 175–186, 2000.
2. J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26:832–843, 1983.
3. J. F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154, 1984.
4. G. Ariav. An Overview of TQuel. *ACM Transactions on Database Systems*, 11(4):499–527, 1986.
5. R. S. Bird. *Introduction to Functional Programming Using Haskell.* Prentice-Hall International, London, UK, 1998.
6. T. Bittner. Rough Sets in Spatio-Temporal Data Mining. In *Int. Workshop on Temporal, Spatial and Spatio-Temporal Data Mining*, LNAI 2007, pages 89–104, 2001.
7. M. H. Böhlen, C. S. Jensen, and B. Skjellaug. Spatio-Temporal Database Support for Legacy Applications. In *ACM Symp. on Applied Computing*, pages 226–234, 1998.
8. M. M. Burnett. Visual Programming. In Webster, J. G., editor, *Encyclopedia of Electrical and Electronics Engineering*, pages 215–283. John Wiley & Sons, 1999.
9. C. X. Chen and C. Zaniolo. SQLST: A Spatio-Temporal Data Model and Query Language. In *19th Int. Conf. on Conceptual Modeling*, LNCS 1920, pages 96–111, 2000.
10. T. S. Cheng and S. K. Gadia. A Pattern Matching Language for Spatio-Temporal Databases. In *ACM Conf. on Information and Knowledge Management*, pages 288–295, 1994.
11. J. Chomicki. Querying Spatiotemporal Databases. In *NSF/BDEI Workshop on Spatio-Temporal Data Models of Biogeophysical Fields for Ecological Forecasting*, San Diego Supercomputer Center, La Jolla, CA, USA, 2002.
12. J. Chomicki and P. Z. Revesz. A Geometric Framework for Specifying Spatiotemporal Objects. In *6th Int. Workshop on Temporal Representation and Reasoning*, pages 41–46, 1999.
13. W. Citrin, R. Hall, and B. Zorn. Programming with Visual Expressions. In *11th IEEE Symp. on Visual Languages*, pages 294–301, 1995.
14. M. Craglia, P. Haining, and P. Wiles. A Comparative Evaluation of Approaches to Urban Crime Pattern Analysis. *Urban Studies*, 37(4):711–729, 2000.
15. Z. Cui, A. G. Cohn, and D. A. Randell. Qualitative and Topological Relationships in Spatial Databases. In *3rd Int. Symp. on Advances in Spatial Databases*, LNCS 692, pages 296–315, 1993.

16. D. J. Hand and H. Mannila and P. Smyth. *Principles of Data Mining.* MIT Press, Cambridge, MA, 2001.
17. D. Walgraef. *Spatio-Temporal Pattern Formation: With Examples from Physics, Chemistry, and Material Science.* Springer Verlag, New York, NY, 1997.
18. N. Djafri, A. A. A. Fernandez, N. P. Paton, and T. Griffiths. Spatio-Temporal Evolution: Querying Patterns of Change in Databases. In *10th ACM Int. Symp. on Advances in Geographic Information Systems*, 2002.
19. M. J. Egenhofer and R. D. Franzosa. Point-Set Topological Spatial Relations. *Int. Journal of Geographical Information Systems*, 5(2):161–174, 1991.
20. M. Erwig. Design of Spatio-Temporal Query Languages. In *NSF/BDEI Workshop on Spatio-Temporal Data Models of Biogeophysical Fields for Ecological Forecasting*, San Diego Supercomputer Center, La Jolla, CA, USA, 2002.
21. M. Erwig, R. H. Güting, M. Schneider, and M. Vazirgiannis. Abstract and Discrete Modeling of Spatio-Temporal Data Types. In *6th ACM Symp. on Geographic Information Systems*, pages 131–136, 1998.
22. M. Erwig, R. H. Güting, M. Schneider, and M. Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *GeoInformatica*, 3(3):269–296, 1999.
23. M. Erwig and B. Meyer. Heterogeneous Visual Languages – Integrating Visual and Textual Programming. In *11th IEEE Symp. on Visual Languages*, pages 318–325, 1995.
24. M. Erwig and M. Schneider. Developments in Spatio-Temporal Query Languages. In *IEEE Int. Workshop on Spatio-Temporal Data Models and Languages*, pages 441–449, 1999.
25. M. Erwig and M. Schneider. Visual Specifications of Spatio-Temporal Developments. In *15th IEEE Symp. on Visual Languages*, pages 187–188, 1999.
26. M. Erwig and M. Schneider. Query-By-Trace: Visual Predicate Specification in Spatio-Temporal Databases. In H. Arisawa and T. Catarci, editors, *Advances in Visual Information Management – Visual Database Systems*, pages 199–218. Kluwer Academic Publishers, Boston, MA, 2000.
27. M. Erwig and M. Schneider. Spatio-Temporal Predicates. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):881–901, 2002.
28. M. Erwig and M. Schneider. STQL: A Spatio-Temporal Query Language. In R. Ladner, K. Shaw, and M. Abdelguerfi, editors, *Mining Spatio-Temporal Information Systems*, chapter 6, pages 105–126. Kluwer Academic Publishers, Norwell, MA, 2002.
29. M. Erwig and M. Schneider. A Visual Language for the Evolution of Spatial Relationships and its Translation into a Spatio-Temporal Calculus. *Journal of Visual Languages and Computing*, 14(2):181–211, 2003.
30. M. Erwig, M. Schneider, and R. H. Güting. Temporal Objects for Spatio-Temporal Data Models and a Comparison of Their Representations. In *Int. Workshop on Advances in Database Technologies*, LNCS 1552, pages 454–465, 1998.
31. G. M. Kuper, L. Libkin, J. Paredaens, editor. *Constraint Databases.* Springer-Verlag, Berlin, 2000.
32. G. Piatetsky-Shapiro and W. Frawley. *Knowledge Discovery in Databases.* MIT Press, Cambridge, MA, 1991.
33. S. Gaal. *Point Set Topology.* Academic Press, 1964.

34. S. K. Gadia and S. S. Nair. Temporal Databases: A Prelude to Parametric Data. In A. U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*, pages 28–66. The Benjamin/Cummings Publishing Company, 1993.
35. A. Galton. Towards a Qualitative Theory of Movement. In *2nd Int. Conf. on Spatial Information Theory*, LNCS 988, pages 377–396, 1995.
36. A. Galton. A Formal Theory of Objects and Fields. In *5th Int. Conf. on Spatial Information Theory*, LNCS 2205, pages 458–473, 2001.
37. E. Geake. How PCs Predict where Crime will Strike. *New Scientist*, 140(1896):17, 1993.
38. T. R. G. Green and M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages and Computing*, 7(2):131–174, 1996.
39. S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. The DEDALE Prototype. In G. M. Kuper, L. Libkin, J. Paredaens, editor, *Constraint Databases*, chapter 17, pages 365–382. Springer-Verlag, Berlin, 2000.
40. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. In *6th ACM Int. Symp. on Advances in Geographic Information Systems*, pages 106–111, 1998.
41. S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-Temporal Data Handling with Constraints. *GeoInformatica*, 5:95–115, 2001.
42. R. H. Güting. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. In *Int. Conf. on Extending Database Technology*, LNCS 303, pages 506–527, 1988.
43. R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A Foundation for Representing and Querying Moving Objects. *ACM Transactions on Database Systems*, 25(1):1–42, 2000.
44. H. W. Mielke. *Patterns of Life: Biogeography of a Changing World.* Unwin Hyman, Inc., Winchester, MA, 1989.
45. G. M. Henebry, J. Chomicki, T. Fountain, and K. J. Ranson, editors. *NSF/BDEI Workshop on Spatio-Temporal Data Models of Biogeophysical Fields for Ecological Forecasting*, San Diego Supercomputer Center, La Jolla, CA, USA, April 2002.
46. M. P. Hoerling and A. Kumar. Atmospheric Response Patterns Associated with Tropical Forcing. *Journal of Climate*, 15(16):2184–2203, 2002.
47. E. Hutchins, J. Hollan, and D. Nomran. Direct Manipulation Interfaces. In D. Norman and S. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*, pages 87–124. Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986.
48. S. Imfeld. *Time, Points and Space – Towards a Better Analysis of Wildlife Data in GIS.* Dissertation, University of Zürich, 2000.
49. L. Knopoff, A. Gabrielov, and M. Ghil, editors. *IMA Workshop on Spatio-Temporal Patterns in the Geosciences*, University of Minnesota, Minneapolis, MN, USA, September 2001.
50. G. Kollios, D. Gunopulos, and V. J. Tsotras. On Indexing Mobile Objects. In *18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 261–272, 1999.
51. E. Mesrobian, R. Muntz, E. Shek, S. Nittel, M. La-Rouche, M. Kriguer, C. Mechoso, J. Farrara, P. Stolorz, and H. Nakamura. Mining Geophysical Data for Knowledge. *IEEE Expert*, 11(5):34–44, 1996.

52. T. Miller. Computers Track the Criminal's Trail. *American Demographics*, 16(1):13–14, 1994.
53. G. Naggle. Urban Crime: A Geographic Appraisal. *Geographical Magazine*, 67(4):56–57, 1995.
54. D. J. Peuquet and N. Duan. An Event-Based Spatiotemporal Data Model (ESTDM) for Temporal Analysis of Geographical Data. *Int. Journal of Geographical Information Systems*, 9(1):7–24, 1995.
55. Peuquet, D. J. *Representations of Space and Time.* The Guilford Press, New York, NY, 2002.
56. J. Renz and B. Nebel. On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. *Artificial Intelligence*, 108(1-2):69–123, 1999.
57. B. Romanowicz. Spatiotemporal Patterns in the Energy Release of Great Earthquakes. *Science*, 260(5116):1923–1926, 1993.
58. M. Scholl and A. Voisard. Thematic Map Modeling. In *1st Int. Symp. on Large Spatial Databases*, LNCS 409, pages 167–190, 1989.
59. A. Segev and A. Shoshani. Logical Modeling of Temporal Data. In *ACM SIGMOD Conf. on Management of Data*, pages 454–466, 1987.
60. A. Segev and A. Shoshani. A Temporal Data Model Based on Time Sequences. In A. U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*, pages 248–270. The Benjamin/Cummings Publishing Company, 1993.
61. B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Computer*, 16(8):57–69, 1983.
62. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Int. Conf. on Data Engineering*, pages 422–432, 1997.
63. R. T. Snodgrass, editor. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, Boston, MA, 1995.
64. M. Stonebraker. Inclusion of New Types in Relational Database Systems. In *Int. Conf. on Data Engineering*, pages 262–269, 1986.
65. T. Suzue and Y. Shinoda. Highly Reproducible Spatiotemporal Patterns of Mammalian Embryonic Movements at the Developmental Stage of the Earliest Spontaneous Motility. *European Journal of Neuroscience*, 11(8):2697–2710, 1999.
66. Y. Theodoridis, T. Sellis, A. Papadopoulos, and Y. Manolopoulos. Specifications for Efficient Indexing in Spatiotemporal Databases. In *10th Int. Conf. on Scientific and Statistical Database Management*, pages 123–132, 1998.
67. R. B. Tilove. Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE Transactions on Computers*, C-29:874–883, 1980.
68. Y. M. Tourre, B. Rajagopalan, and Y. Kushnir. Dominant Patterns of Climate Variability in the Atlantic Ocean during the Last 136 Years. *Journal of Climate*, 12(8):2285–2299, 1999.
69. I. Tsoukatos and D. Gunopulos. Efficient Mining of Spatiotemporal Patterns. In *7th Int. Symp. on Spatial and Temporal Databases*, LNCS 2121, pages 425–442, 2001.
70. M. A. Tumeo and M. K. Larson. Movement of Fuel Spills in the Ross Ice Shelf. *Antarctic Journal of the United States*, 29(5):373–374, 1994.
71. M. Vazirgiannis and O. Wolfson. A Spatiotemporal Query Language for Moving Objects. In *7th Int. Symp. on Spatial and Temporal Databases*, LNCS 2121, pages 20–35, 2001.

72. F. Wang and W. W. Minor. Where the Jobs Are: Employment Access and Crime Patterns in Cleveland. *Annals of the Association of American Geographers*, 92(3):435–451, 2002.
73. T. Weigand, K. A. Moloney, and S. J. Milton. Population Dynamics, Disturbance, and Pattern Evolution: Identifying the Fundamental Scales of Organization in a Model Ecosystem. *The American Naturalist*, 152(3):321–337, 1998.
74. M. F. Worboys. A Unified Model for Spatial and Temporal Information. *The Computer Journal*, 37(1):25–34, 1994.
75. T. S. Yeh and B. de Cambray. Time as a Geometric Dimension for Modeling the Evolution of Entities: A 3D Approach. In *Int. Conf. on Integrating GIS and Environmental Modeling*, 1993.
76. T. S. Yeh and B. de Cambray. Modeling Highly Variable Spatio-Temporal Data. In *6th AustraliAsian Database Conf.*, pages 221–230, 1995.