

QUERY-BY-TRACE: VISUAL PREDICATE SPECIFICATION IN SPATIO-TEMPORAL DATABASES

Martin Erwig and Markus Schneider
FernUniversität Hagen, Praktische Informatik IV
58084 Hagen, Germany
{erwig, markus.schneider}@fernuni-hagen.de

Abstract In this paper we propose a visual interface for the specification of predicates to be used in queries on spatio-temporal databases. The approach is based on a visual specification method for temporally changing spatial situations. This extends existing concepts for visual spatial query languages, which are only capable of querying static spatial situations. We outline a preliminary user interface that supports the specification on an intuitive and easily manageable level, and we describe the design of the underlying visual language. The visual notation can be used directly as a visual query interface to spatio-temporal databases, or it can provide predicate specifications that can be integrated into textual query languages leading to heterogeneous languages.

Key Words Spatio-Temporal Queries, Visual Predicate Specification, Visual Database Interface

1. INTRODUCTION

Spatio-temporal databases deal with spatial objects that change over time (for example, they move or they grow): cars, planes, people, animals, ..., storms, lakes, forests, etc. Hence, database systems, in particular, spatial and temporal database systems, and geographical information systems (GIS) need to be extended to handle this kind of information. Of particular interest is, of course, the development of simple but powerful query languages that allow one to ask for changes in spatial relationships, for instance: “Has a tornado ever crossed Iowa?” or “Which planes were able to avoid a certain blizzard?”. A formal foundation for these kinds of queries is given by *spatio-temporal predicates* (Erwig et al., 1999e). Whereas it is possible to identify a relatively small set of spatial predicates (Egenhofer et al., 1991), it is almost impossible to do so in the spatio-temporal case, simply because there are too many of them. Thus, there is a very strong need for a simple way of specifying spatio-temporal situations, and a visual notation can be extremely helpful here.

We will propose a visual language for spatio-temporal predicates. The main idea is to represent a spatio-temporal object (such as a car or a storm) in a two-dimensional way by its trace. The intersections of such a trace with another object's trace is interpreted and translated into a sequence of predicates, called *development*, that can then be used, for example, to query spatio-temporal databases. This interpretation is described in (Erwig et al., 1999d).

The described visual notation can be employed in several ways. One application is, as already mentioned, to realize a visual query interface to spatio-temporal databases. But we can also use pictures of this language as specifications for (complex) spatio-temporal predicates, which can then be used in arbitrary query languages. One interesting possibility is to use a well-accepted textual query language like SQL, extend it by spatio-temporal objects and predicates (Erwig et al., 1999b), and use pictures to represent predicates in WHERE clauses. This leads then to a heterogeneous visual language (Erwig et al., 1995).

The paper is structured as follows: after commenting on related work in the next section, we demonstrate in Section 3 as an application of visual development specifications a visual query interface to a spatio-temporal database. In Section 4 we describe how spatio-temporal data can be modeled. In particular, we explain the notions of spatio-temporal objects, predicates, and developments. In Section 5 we then explain and motivate the design of our visual notation for developments. Finally, conclusions are given in Section 6.

2. RELATED WORK

The similarity of spatial and temporal phenomena has been recognized for a long time in the literature. Both phenomena deal with “spaces” or “dimensions” of some kind and are thus closely related. Recently, research efforts have led both in spatial and in temporal data modeling to an increased interest in integrating both directions into a common research branch called *spatio-temporal data modeling* and in constructing *spatio-temporal data bases*. Their underlying basic entities are called *spatio-temporal objects* and are ubiquitous in everyday life. Consider the flight of an airplane, the migration of whales, the raging of a storm, or the spreading of a fire region. Characteristic features of all these objects are that they are *spatial entities changing over time* and that these changes are *continuous*. Changes refer, for example, to motion, shrinking, growing, shape transformation, splitting, merging, disappearing, or reappearing of spatio-temporal objects. In particular, the capability of incorporating continuous change of spatial objects over time belongs to the most challenging requirements of spatio-temporal data models.

In the meanwhile, some data models for spatio-temporal databases have already been proposed. In (Worboys, 1994) a spatial data model has been generalized to become spatio-temporal: spatio-temporal 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 spatio-temporal and include variants of Gadia's temporal model (Gadia et al., 1993) which are described in (Cheng et al., 1994, Böhlen et al., 1998). The main drawback of all these approaches is that ultimately they are incapable of modeling continuous changes of spatial objects over time.

Our approach to dealing with spatio-temporal data takes a more integrated view of space and time and includes the treatment of *continuous* spatial changes. It introduces the concept of *spatio-temporal data types* (Erwig et al., 1998a, 1999a). These data types are designed as *abstract data types* whose values can be integrated as complex entities into databases (Stonebraker, 1986) and whose definition and integration into databases is independent of a particular DBMS data model.

The definition of a temporal object (Erwig et al., 1998b) in general is motivated by 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 . Spatio-temporal objects are regarded as special instances of temporal objects where α is a spatial data type like *point* or *region*. A point (representing an airplane, for instance) that changes its location in the Euclidean plane over time is called a *moving point*. Similarly, a temporally changing region (representing a fire area, for instance) is a region that can move and/or grow/shrink and whose components can split or merge. We call such an object an *evolving region*.

Similar to our approach, in (Yeh et al., 1993, 1995) based on the work in (Segev et al., 1993) so-called *behavioral time sequences* are introduced. Each element of such a sequence contains a geometric value, a date, and a *behavioral function*, the latter describing the evolution between two consecutive elements of the sequence. Whereas this approach mainly focuses on representational issues and advocates the three-dimensional object view of spatio-temporal objects, we are particularly interested in an algebraic model of general spatio-temporal data types including a comprehensive collection of spatio-temporal operations. Nevertheless, behavioral time sequences could be used as representations for our temporal objects.

Temporal changes of spatial objects induce modifications of their mutual topological relationships over time. For example, at one time two spatio-temporal objects might be disjoint whereas some time later they might intersect. These modifications usually proceed continuously over time but can, of course, also have a discrete property. We already have devised and formally defined a concept for such *spatio-temporal relationships* which are described by so-called *spatio-temporal predicates* (Erwig et al., 1999e). We call a sequence of spatial and spatio-temporal predicates a *development*.

Since we are dealing with predicates, it is not surprising that logic-based approaches are related to our work. Allen (Allen, 1984) defines a predicate $Holds(p,i)$ which asserts that a property p is true during a time interval i . Galton (Galton, 1995) has extended Allen's approach to the treatment of temporally changing two-dimensional topological relationships. Topological predicates are taken from the RCC model (Cui et al., 1993) which comes to similar results as Egenhofer's 9-intersection model which is briefly discussed below. In contrast to these approaches, we have pursued a hybrid approach taking into account elements from temporal logic and elements from point set theory and point set topology. The main reason for not taking a purely logic approach is the intended integration of spatio-temporal objects and predicates into spatio-temporal databases and query languages. These require concrete representations for spatio-temporal objects and besides predicates the possibility of constructing new objects through spatio-temporal operations. Hence, efficiency, in particular, for the evaluation of spatio-temporal queries, is indispensable.

Our work on spatio-temporal predicates is based on Egenhofer's 9-intersection model (Egenhofer et al., 1991) for topological predicates between spatial objects in two-dimensional space. The goal of this model is to provide a canonical collection of topological relationships for each combination of spatial types. The model rests on the nine possible intersections of boundary, interior, and exterior of a spatial object with the corresponding parts of another object. Each intersection is then tested with regard to the topologically invariant criteria of emptiness and non-emptiness. From the total number of possible topological constellations only a certain subset makes sense depending on the combination of spatial objects just considered. For two regions, eight meaningful constellations have been identified which lead to the eight predicates called *equal*, *disjoint*, *coveredBy*, *covers*, *intersect*, *meet*, *inside*, and *contains*. For a point and a region we obtain the three predicates *disjoint*, *meet*, and *inside*. For two points we get the two predicates *disjoint* and *meet* (which corresponds to equality). For each group all predicates are mutually exclusive. They are also complete in the sense that they cover all possible topological constellations under the assumptions of the 9-intersection model.

There exist several approaches to visual query languages for spatial databases, for example, (Aufaure-Portier, 1995, Calcinelli et al., 1994, Egenhofer, 1996, Lee et al., 1995). Common to all these approaches is that they allow to query only *static* spatial situations, that is, they can express queries like "Retrieve all airports in Ohio". A characteristic of the involved objects "airport" and "Ohio" is that these objects rarely change their location and/or extent. There are also a few approaches to querying image sequences (Arndt et al., 1989, Del Bimbo et al., 1995, Walter et al., 1992). However, the goal of these proposals is mainly to facilitate queries on video databases and not the querying of spatial (or spatio-temporal) databases. Since video data is largely

unstructured (just a sequence of images), all these approaches have to be concerned with additional symbolic representations for the stored images to enable queries. Our visual notation is translated into sequences of predicates that can be directly checked for the database representation of the spatio-temporal objects. A short, preliminary proposal for the visualization idea that is developed in this paper has been presented in (Erwig et al., 1999c).

3. QUERY-BY-TRACE

The following scenario illustrates how our visual notation can be employed for querying developments in spatio-temporal databases. We give a rough outline of the interaction a user may perform when visually specifying queries. Our goal is to interactively and graphically produce a sketch from which a spatio-temporal predicate can be derived.

The user interface *Query-By-Trace (QBT)* allows a comfortable specification of developments. It incorporates an editor component to draw specifications. The horizontal dimension is the x -axis; the vertical dimension describes time. The top of the editor provides two menus, one for moving points and one for evolving regions. Assuming a relational setting, both menus show the available attributes related to spatio-temporal objects in the database together with the corresponding relation names in brackets. In our example we use an environmental database containing weather and flight information. Assume that a user asks for all flights crossing hurricanes. The user selects from the

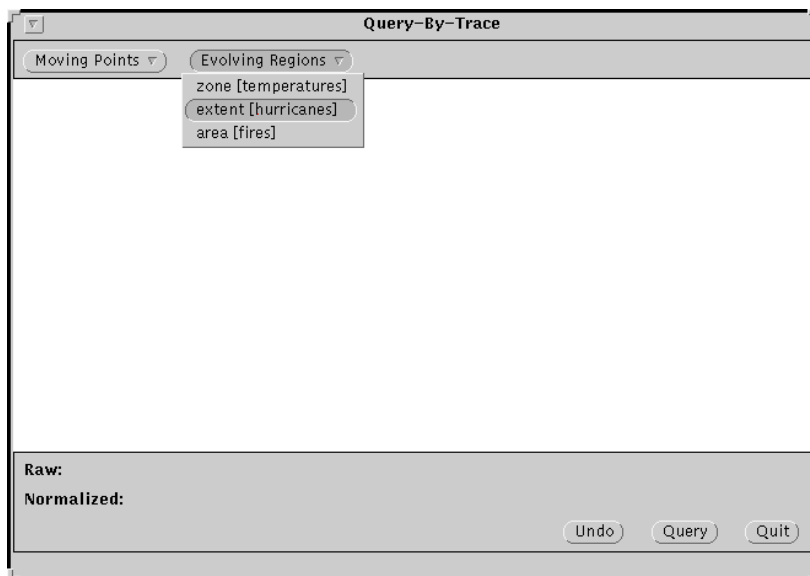


Figure 1 Selecting First Object

menu *Evolving Regions* the attribute *extent* of the relation *hurricanes* (see Figure 1) and clicks at a desired position on the canvas of the editor. The result of this action is a circle labeled with the name of the selected relation (see Figure 2).

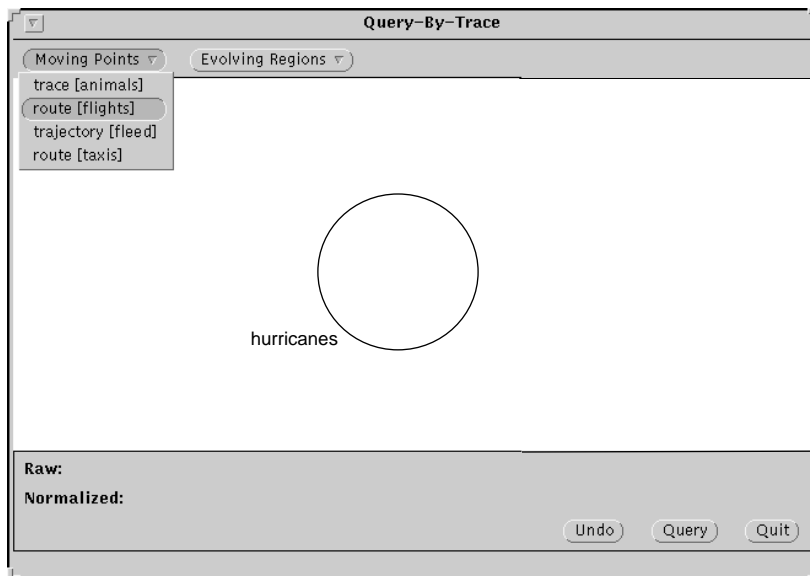


Figure 2 Selecting Second Object

Two things are striking. First, we can observe that the circle and the vertical line, respectively, are *static*. Since we are going to investigate the topological relationships between two moving objects, it is not decisive whether both objects or only one object moves due to the independence of metric and distance properties. It is only necessary that one object moves to be able to describe and visualize the process of the temporal evolution of the spatial relationships between the objects. Second, we do not need to model the real extent, shape, and location of an evolving region and the exact location of a moving point over time. We can abstract from these aspects since we are only interested in specifying topological relationships, which is a task for which we do not need any metric information.

Depending on the next selection, the kind of query is determined: if another region is chosen, a development between two regions is being specified; otherwise a point/region development is going to be sketched. If instead of a moving region a moving point were selected at the beginning, the user is only allowed to select another point, and a development between two moving points would be specified.

In our example the user now selects from the menu *Moving Points* the attribute *route* of the relation *flights*. This indicates that the user is interested in specifying the development between an evolving region and a moving

point. The second selection always creates a point or a circle that can be moved over the canvas. The user next draws a crossing situation which requires the following interactions: a click at a desired position outside the circle produces the starting point. The system determines the initial relationship of this point with respect to the evolving region and displays the spatial predicate *disjoint* in the two message lines at the bottom of the editor. Now the user drags the mouse from bottom to top from the starting point towards the circle. Note that during the specification process it is not possible to drag the mouse cursor below the current position since a spatial object cannot move backward in time. As soon as the mouse cursor leaves the starting point, the name of the spatio-temporal predicate *Disjoint* is added to the message lines. The fact that a spatial predicate is constant for a certain period is registered in the message lines by a spatio-temporal predicate indicated by an initial capital letter (for example, *Disjoint*, *Meet*).

We distinguish between the *raw mode* and the *normalized mode* of a development specification. The *raw mode* corresponds to the original definition of a development as an alternating sequence of spatial and spatio-temporal predicates. The *normalized mode* introduces simplifications to make the specification more readable for the user. One of these simplifications is that a spatial predicate (like *disjoint*) followed or preceded by its corresponding spatio-temporal predicate (like *Disjoint*) can be abbreviated to the spatio-temporal predicate. Hence, in the second message line we only see the *Disjoint* predicate so far, see Figure 3.

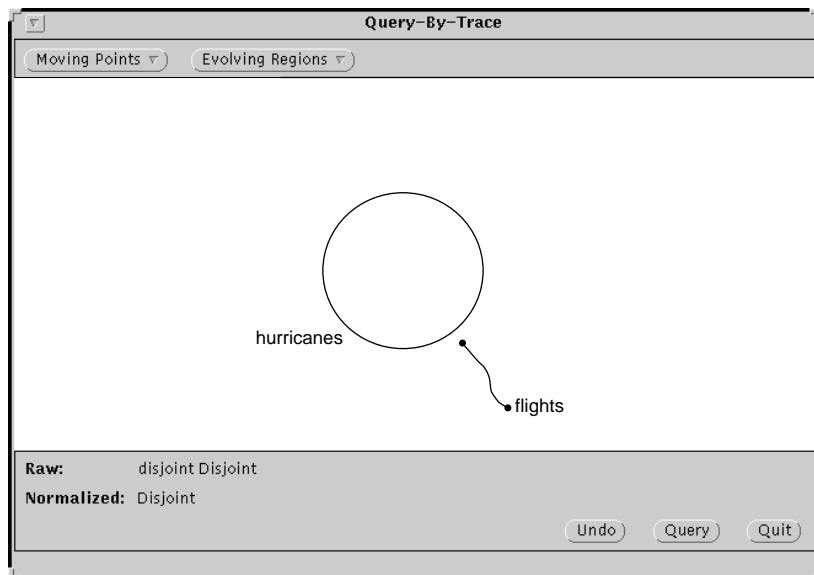


Figure 3 Raw and Normalized Modes

While moving the mouse, the system draws the trace of the point and steadily watches for possible changes in the topological relationship. Each change is recorded in the message lines. The user now continues to move the cursor towards the circle and then traverses it. So far the user has specified an *Enter* situation, that is, the moving point at some time has met the circle and has been inside the circle since then, see Figure 4. Afterwards the user drags the

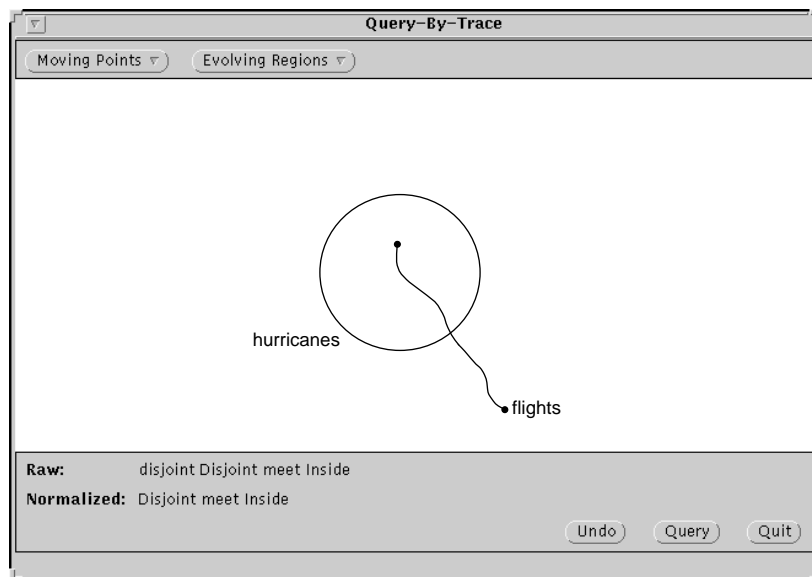


Figure 4 Dragging Sample Objects

mouse to an end point outside the circle and releases the mouse button. The final picture is shown in Figure 5.

If the second selection of a moving object is also an evolving region, the development between two evolving regions shall be specified. The user can in this case move a second circle which is smaller than the first one. The trace consists of two disjoint curves spanning a corridor which the moved circle traverses while being dragged from a start position to its end position. The two predicates *meet* and *coveredBy* (describing the situations when the circles touch externally, respectively, internally) are called *instant predicates* since only they can be valid at an instant. They can, of course, also be valid for some period (*Meet*, *CoveredBy*). To distinguish these two cases interactively, the drawing of *Meet* and *CoveredBy* is supported by holding down the shift-key during dragging. The movement of the mouse is then restricted to go along the border of the constant object until the shift-key is released again. At the end of a dragging transaction, both the visual specification and the spatio-temporal predicate sequence are immediately available.

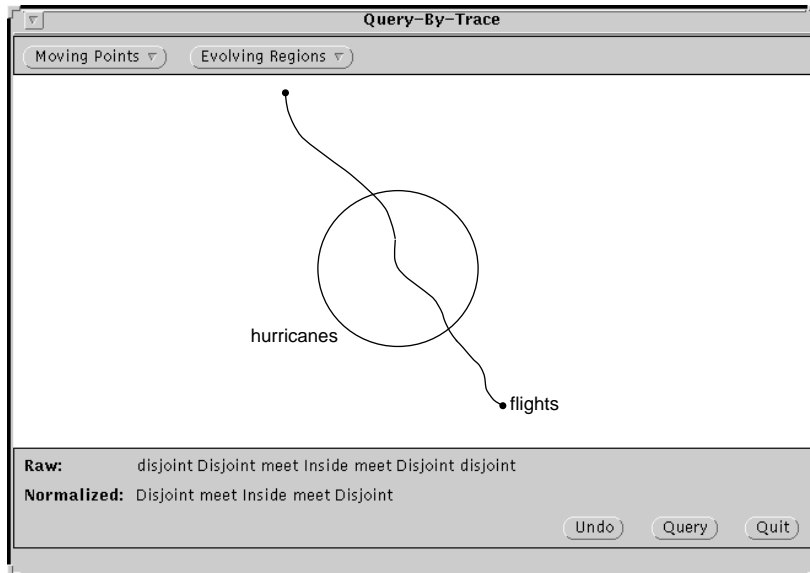


Figure 5 Final QBT-Specification of the *Cross* Predicate

An example for two moving points is given in (Erwig et al., 1999d). We believe that this user interface is intuitive and easy to use because the user acts (via the mouse) as a moving object that behaves exactly in the way as the drawn spatio-temporal predicate demands it. In other words, the user action precisely conforms to, or satisfies, the specification that is drawn.

4. SPATIO-TEMPORAL OBJECTS, PREDICATES, AND DEVELOPMENTS

In this section we will review some of the formal foundations and sketch our definition of spatio-temporal objects (Section 4.1), our concept of spatio-temporal predicates (Section 4.2), and our specification mechanism for spatio-temporal developments (Section 4.3).

4.1. SPATIO-TEMPORAL OBJECTS

One of our design goals is to define a spatio-temporal data model that is independent of a specific DBMS data model. This is achieved by encapsulating spatio-temporal data types into abstract data types which comprise a comprehensive collection of operations and predicates. Assuming a relational setting, for instance, we can then embed spatio-temporal data types in the same way

like types for integers, reals, booleans, or strings as attribute types in a relation, that is, the relation has only a container function to store attribute data in tuples.

The design of our model for spatio-temporal data is as follows: for compatibility with smoothly changing spatio-temporal objects we choose a continuous model of time, that is, $time = \mathbb{R}$. The temporal version of a value of type α that changes over time can be modeled as a *temporal function* of type

$$\tau(\alpha) = time \rightarrow \alpha$$

We have used temporal functions as the basis of an algebraic data model for *spatio-temporal data types* (Erwig et al., 1998a, 1999a) where α is assigned a spatial data type like *point* or *region*. For example, a point that changes its location over time is an element of type $\tau(point)$ and is called a *moving point*. Similarly, an element of type $\tau(region)$ is a region that can move and/or grow/shrink. It is called an *evolving region*. Currently, we do not consider a temporal version of lines, mainly because there seem to be not many applications of moving lines. A reason might be that lines are themselves abstractions or projections of movements and thus not the primary entities whose movements should be considered. In any case, however, it is principally possible to integrate moving lines in much the same way as moving points if needed. In addition, we also have changing numbers and booleans, which are essential when defining operations on temporal objects. For instance, we could be interested in computing the (time-dependent) distance of an airplane and a storm. This could be achieved by an operation:

$$Distance : \tau(point) \times \tau(region) \rightarrow \tau(real)$$

The example demonstrates the concept of *temporal lifting* avoiding an inflation of operation names and definitions: we can, in principle, take almost any non-temporal operation (like $distance : point \times region \rightarrow real$) and “lift” it so that it works on temporal objects returning also a temporal object as a result. More precisely, for each function $f : \alpha_1 \times \dots \times \alpha_n \rightarrow \beta$ its corresponding lifted version

$$\uparrow f : \tau(\alpha_1) \times \dots \times \tau(\alpha_n) \rightarrow \tau(\beta)$$

is defined by:

$$\uparrow f(S_1, \dots, S_n) := \{(t, f(S_1(t), \dots, S_n(t))) \mid t \in time\}$$

Hence, we can derive temporal operations rather automatically. For example, we obtain $Distance = \uparrow distance$.

4.2. SPATIO-TEMPORAL PREDICATES

Temporal lifting is, of course, also applicable to spatial predicates. Consider the spatial predicate

$$inside : point \times region \rightarrow bool$$

The lifted version of this predicate has the type

$$\uparrow inside : \tau(point) \times \tau(region) \rightarrow \tau(bool)$$

with the meaning that it yields *true* for each time at which the point is inside the region, *undefined* whenever the point or the region is undefined, and *false* in all other cases. We see that the lifted version is not a predicate since it yields a temporal boolean and not a (flat) boolean what we would expect from a predicate.

Our understanding of spatio-temporal predicates is the following: a spatio-temporal predicate is essentially a function that aggregates the values of a spatial predicate as it evolves over time. Thus, a spatio-temporal predicate is a function of type $\tau(\alpha) \times \tau(\beta) \rightarrow bool$ for $\alpha, \beta \in \{point, region\}$.

If we consider the definition of $\uparrow inside$, we can define two spatio-temporal predicates *sometimes-inside* and *always-inside* that yield true if $\uparrow inside$ yields true at some time, respectively, at all times. Whereas the definition for *sometimes-inside* is certainly reasonable, the definition for *always-inside* is questionable since it yields false whenever the point or the region is undefined. This is not what we would expect. For example, when the moving point has a shorter lifetime than the evolving region and is always inside the region, we would expect *always-inside* to yield true. Actually, we can distinguish different kinds of “forall” quantifications that result from different time intervals over which aggregation can be defined to range. In the case of *inside* the expected behavior is obtained if the aggregation ranges over the lifetime of the first argument, the moving point. This is not true for all spatial predicates. In fact, it depends on the nature and use of each individual predicate. For example, two spatio-temporal objects are considered as being *always-equal* only if they are equal on both objects’ lifetimes, that is, the objects must have the same lifespans and must be always equal during these.

In order to be able to concisely build spatio-temporal predicates, we use the following general syntax: $Q_{op}.p$ where $Q \in \{\forall, \exists\}$, $op \in \{\cap, \cup, \pi_1, \pi_2\}$ is a function mapping two sets into a new set (π_i simply takes the i th argument set), and p is a spatial predicate. Such an expression then denotes the spatio-temporal predicate:

$$\lambda(S_1, S_2).Q t \in op(dom(S_1), dom(S_2)).p(S_1(t), S_2(t))$$

This means that, for example, $\forall_{\pi_1}.inside$ denotes the spatio-temporal predicate

$$\lambda(S_1, S_2).\forall t \in dom(S_1).inside(S_1(t), S_2(t))$$

In general, $\lambda(x_1, x_2, \dots).e$ denotes a function that takes arguments x_1, x_2, \dots and returns a value determined by the expression e . So the above expression denotes a function that takes two arguments S_1 and S_2 and yields the boolean value denoted by the \forall -expression.

With this notation we can give the definitions for the spatio-temporal versions of the eight basic spatial predicates (for two regions):

$$\begin{aligned} Disjoint & := \forall_{\cap}.disjoint \\ Meet & := \forall_{\cup}.meet \\ Overlap & := \forall_{\cup}.overlap \\ Equal & := \forall_{\cup}.equal \\ Covers & := \forall_{\pi_2}.covers \\ Contains & := \forall_{\pi_2}.contains \\ CoveredBy & := \forall_{\pi_1}.coveredBy \\ Inside & := \forall_{\pi_1}.inside \end{aligned}$$

For a moving point and a moving region we have just the three basic predicates *Disjoint*, *Meet*, and *Inside*, which are defined as above. For two moving points we have the basic predicates *Disjoint* and *Meet*, which are also defined as above. The chosen aggregations are motivated and discussed in detail in (Erwig et al., 1999e).

4.3. DEVELOPMENTS

Now that we have basic spatio-temporal predicates, the question is how to combine them in order to capture the change of spatial situations. That is, the issue is how to specify *developments*. In order to temporally compose different spatio-temporal predicates, we need a way to restrict the temporal scope of basic spatio-temporal predicates to specific intervals. This can be obtained by *predicate constrictions* (note that $S|_I$ denotes the partial function that yields $S(t)$ for all $t \in I$ and is undefined otherwise): let I be a (half-) open or closed interval. Then

$$P_I := \lambda(S_1, S_2).P(S_1|_I, S_2|_I).$$

Now we can define the *composition* of predicates as follows:

$$\begin{aligned} P \text{ until } p \text{ then } Q & := \\ & \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)) \wedge P|_{]-\infty, t[}(S_1, S_2) \wedge Q|_{]t, \infty[}(S_1, S_2) \end{aligned}$$

When we now consider how spatial situations can change over time, we observe that certain relationships can be valid only for a period of time and not for only a single time point (given that the participating objects do exist for a period of time) while other relationships can hold at instants as well as on time intervals. Predicates that can hold at time points and intervals are: *equal*, *meet*, *covers*, *coveredBy*; these are called *instant predicates*. For example, an airplane and a hurricane can meet at a certain instant or for a whole period. Predicates that can only hold on intervals are: *disjoint*, *overlap*, *inside*, *contains*; these are called *period predicates*. For example, it is not possible for an airplane to be disjoint from a hurricane only at one point in time; they have the inherent property to be disjoint for a period.

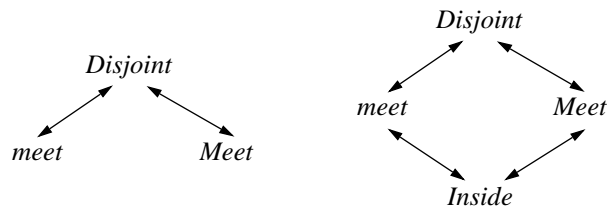
It is now interesting to see that in satisfiable developments instant and period predicates always occur in alternating sequence. For example, it is not possible that two continuously changing spatio-temporal objects satisfy *Inside* immediately followed by *Disjoint*. In contrast, *Inside* first followed by *meet* (or *Meet*) and then followed by *Disjoint* can be satisfied. Hence, developments are represented by alternating sequences of spatio-temporal predicates and spatial predicates and are written down by juxtaposition (in this paper). A more formal treatment of compound spatio-temporal predicates and developments is given in (Erwig et al., 1999e). Our example of a flight running into a hurricane can now be formulated as the composition:

Disjoint until meet then Inside

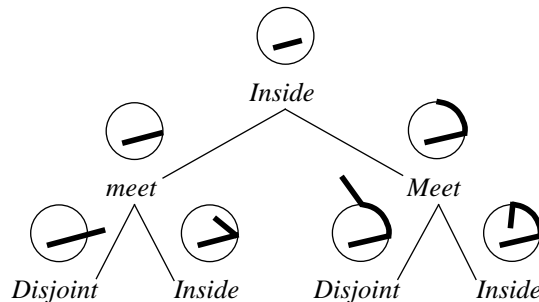
Since predicate composition is associative, we can abbreviate nested compositions by writing down simply a sequence of the spatio-temporal and spatial predicates, that is, we can simply write *Disjoint meet Inside* for the above example. We introduce the name *Enter* for it to reuse it later. A flight running out of a hurricane can be characterized by *Leave := Inside meet Disjoint*. A flight that traverses a hurricane can be described by *Disjoint meet Inside meet Disjoint* using basic spatio-temporal predicates or shorter as *Enter Leave* using derived predicates; we introduce the name *Cross* for it. Note that spatial predicates and their corresponding spatio-temporal predicates (like *meet* and *Meet*) that occur next to each other in a development can be merged to the respective spatio-temporal predicate. We list a few further examples for two evolving regions:

Enter := *Disjoint meet Overlap coveredBy Inside*
Leave := *Inside coveredBy Overlap meet Disjoint*
Cross := *Enter Leave*
Touch := *Disjoint meet Disjoint*
Bypass := *Disjoint Meet Disjoint*
Graze := *Disjoint meet Overlap meet Disjoint*

In order to assess the expressiveness of our visual notation we can ask which developments are possible at all and which developments can be specified by our visual language. Possible topological changes or transitions of spatio-temporal objects over time can be visualized in so-called *development graphs* whose vertices are labeled either with a spatial, that is, an instant, predicate or with a basic spatio-temporal predicate. Hence, each vertex models a time point or a time interval in which the corresponding predicate is valid. An edge (p, q) represents the transition from a predicate p to a predicate q and stands for $p \rightarrow q$. A path (p_1, p_2, \dots, p_n) within the graph describes a possible temporal development $p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_n$ of topological relationships between two spatial objects. For the point/point and for the point/region case we obtain the following two development graphs:



Starting, for example, with *Inside* in the point/region case, we obtain seven possible development paths not properly containing cycles¹:



Since the development graph is symmetric in this case (each of the four vertices can be selected as the start vertex of a path), we obtain a total of 28 paths. This means, there are 28 distinct temporal evolutions of topological changes between a moving point and an evolving region without repetitions. For each alternative we could define an own spatio-temporal predicate. In the point/point case we get 13 possible development paths. The development graph for the region/region case yields not less than 2198 paths and thus possible predicates (Erwig et al., 1999e). It is shown in Figure 6.

1. More precisely, *quasi-cycles*, see (Erwig et al., 1999e).

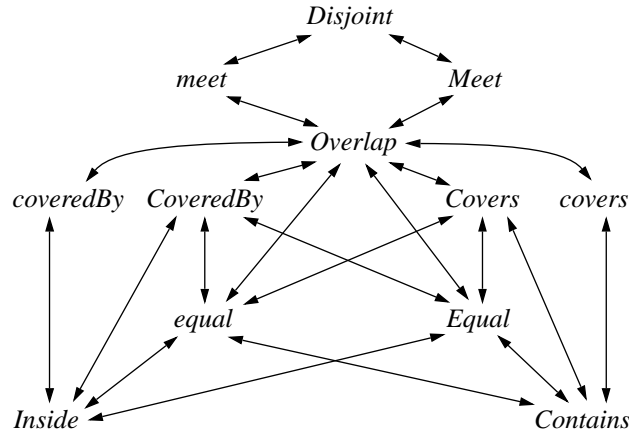


Figure 6 Region/Region Development Graph

There are some constraints imposed by our visual notation which restrict the possible development paths that can be expressed by a visual specification; consequently, they lead to a restriction of the development graph. These constraints are: (i) the sizes of the static circle and the moved circle are fixed, (ii) the static circle is larger than the moved circle, and (iii) our visual notation contains an implicit ordering of both circles, that is, the smaller moved circle symbolizes always the first argument of a predicate and the larger constant circle stands always for its second argument. These constraints lead to the following restrictions of the development graph.

First, from the three pairs *coveredBy/covers*, *CoveredBy/Covers*, and *Inside/Contains* only one relationship per pair, namely *coveredBy*, *CoveredBy*, and *Inside*, can be represented in our visual development specifications. Hence, we can remove the vertices *covers*, *Covers*, and *Contains* and their incident edges².

Second, four transitions in the graph, namely from *CoveredBy* and from *Inside* to *equal* and *Equal*, respectively, solely result from a growing or shrinking of one object. Since we cannot alter the proportions neither of the static circle nor of the moved smaller circle, the vertices *equal* and *Equal* cannot be reached by *CoveredBy* and *Inside* so that we can take away the corresponding four edge.

Third, the transitions between *Overlap* and *equal* and between *Overlap* and *Equal* do not require growing or shrinking. But the prerequisite for this transition is that the static circle and the moved circle have the same size, and just this is excluded by our visual notation. From an *Overlap* situation we can never come to an *equal* or an *Equal* situation so that the two corresponding edges must be removed. Because the vertices *equal* and *Equal* are isolated

². Note that this restriction could be dropped if we would allow that the moved circle can be made larger than the constant circle.

now, we can remove them, too. We obtain the following final development graph shown in Figure 7. In this restricted graph, only 87 different paths not containing quasi-cycles are possible.

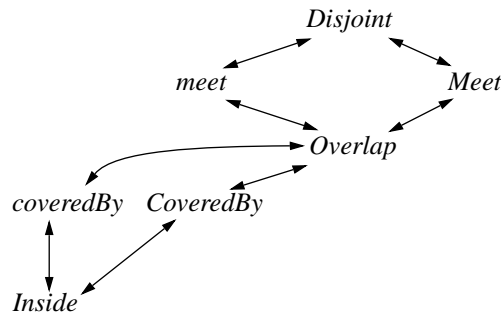


Figure 7 Simplified Development Graphs

All finite paths that can be obtained by this development graph can be specified with our visual language for the region/region case.

5. VISUAL SPECIFICATIONS OF DEVELOPMENTS

In this section we give a collection of design decisions that eventually lead to a simple and intuitive, yet powerful, two-dimensional visual language.

The first design decision is essential to obtain an integrated notation for spatial and temporal aspects:

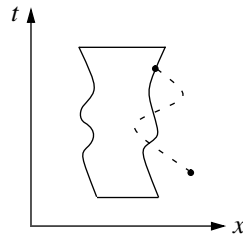
- (1) Represent the temporal dimension geometrically. This leads in a first step to a three-dimensional model of spatio-temporal objects.

Now we could stop here and use 3D pictures to specify developments, but there are two main reasons for not doing so: first, drawing three-dimensional pictures is much more difficult than drawing 2D pictures. In particular, without specialized user input devices, it can become quite tedious to generate 3D drawings with mouse and keyboard. Such a drawing interface is also very hard to implement; it must offer many options to the user and is thus again more difficult to learn and to apply than a two-dimensional language. Second, three-dimensional illustrations of developments are overdetermined in the sense that they display (i) growing/shrinking and movement of regions and (ii) relative positions of the beginnings and endings of objects' lifetimes. (The first point will be discussed in more detail below.) Such overspecifications are generally undesirable since they complicate the understanding of visual notations because the user has to sort out much visual information that has no meaning for her specification.

The second design decision is essentially a step to reduce overdetermination:

- (2) Abstract from exact positions/extents, and reduce two-dimensional geometric objects to one-dimensional ones. Use the y -axis to represent the temporal dimension.

This essentially means to “forget” about the y -axis with regard to spatial information, and to represent a point as a point on the x -axis and a region as an x -interval. Thus, the y -axis can capture the temporal aspect of spatio-temporal objects so that a moving point is represented by a line and an evolving region is represented by a region as shown below:



This picture describes a moving point that enters a region, then leaves the region and finally stops on the region’s border. It is striking that the sketched movement/shrinking/growing of the interval representing the evolving region does not contribute anything to this specification, that is, it would be as well possible to use a plain rectangle representing a stationary/constant region. The reason is that we are only specifying topological relationships, and thus we need only information about the relative positions of objects with respect to each other. In particular, we need not be concerned about the exact position or size information of objects.³

This leads to the third design decision:

- (3) Represent the evolving region in the definition of a point/region predicate (respectively, one evolving region in the definition of a region/region predicate) simply as a circle. Likewise, represent one of the two moving points in a point/point spatio-temporal predicate as a vertical line.

This leads to an easy to understand notation. For instance, the point/region predicate *Bypass* can be specified as shown in Figure 8.

It remains to be explained how the second evolving region in the specification of a region/region predicate is represented. We do that analogous to the representation of moving points: display two objects (showing the moving object’s first and last position) connected by a trace specifying the object’s

³. Actually, this is not the whole truth: for some spatio-temporal developments, growing and shrinking is essential, but these cases are rare, and the complexity of an extension of the visual notation would not be justified by the relatively small gain in expressiveness, see Section 4.

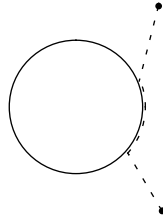
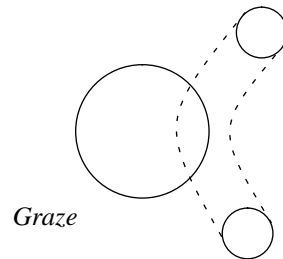


Figure 8 Visual Specification of Bypass

movement. The initial and the final object are given by two circles (that are smaller than the constant circle). The trace is depicted for a moving point by a dotted line, and for a moving region we use two dotted lines. For example the predicate *Graze* is drawn as follows:



The first and the last shown position of the evolving region are disjoint from the constant region, and thus they both represent the predicate *disjoint*. The trace represents the sequence *Disjoint meet Overlap meet Disjoint*. This is because the left trace border intersects the constant region in exactly two points and the right trace border does not intersect the constant region at all. Hence, altogether this picture denotes the predicate *Graze*. Some variations are shown below.

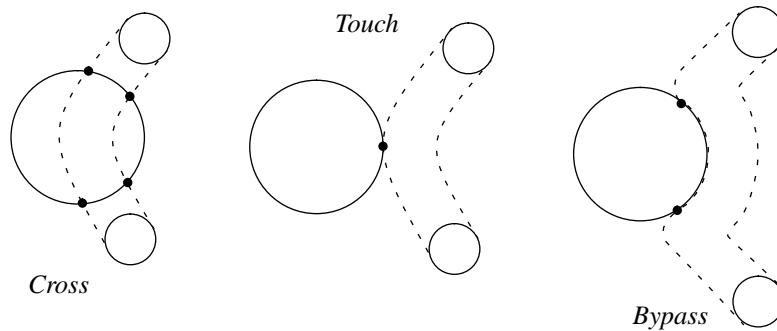


Figure 9 More Spatio-Temporal Predicates

Note that the exact interpretation can always be inferred from the intersections of the trace borders with the static circle. This is explained in (Erwig et al., 1999d).

6. CONCLUSIONS

We have demonstrated how a simple two-dimensional visual language can be used to express predicates on spatio-temporal objects. This language can be well used as a query interface to spatio-temporal databases. Having a precise semantics, the visual notation can also serve as a formal language to communicate and reason about spatio-temporal situations in general.

REFERENCES

- Allen, J.F. (1984) Towards a General Theory of Action and Time, *Artificial Intelligence* 23, 123-154.
- Arndt, T. and Chang, S.K. (1989) Image Sequence Compression by Iconic Indexing, *IEEE Workshop on Visual Languages*, 177-182.
- Aufaure-Portier, M.A. (1995) A High Level Interface Language for GIS, *Journal of Visual Languages and Computing* 6, 167-182.
- Böhlen, M.H., Jensen, C.S. and Skjellaug, B. (1998) Spatio-Temporal Database Support for Legacy Applications, *ACM Symp. on Applied Computing*, 226-234.
- Calcinelli, D. and Mainguenaud, M. (1994) Cigales, a Visual Query Language for a Geographical Information System: the User Interface, *Journal of Visual Languages and Computing* 5, 113-132.
- Cheng, T.S. and Gadia, S.K. (1994) A Pattern Matching Language for Spatio-Temporal Databases, *ACM Conf. on Information and Knowledge Management*, 288-295.
- Cui, Z. Cohn, A.G. and Randell, D.A. (1993) Qualitative and Topological Relationships in Spatial Databases, *Int. Symp. on Advances in Spatial Databases*, LNCS 692, 296-315.
- Del Bimbo, A., Vicario, E. and Zingoni, D. (1995) Symbolic Description and Visual Querying of Image Sequences Using Spatio-Temporal Logic, *IEEE Transactions on Knowledge and Data Engineering* 7(4), 609-621.
- Egenhofer, M.J. (1996) Spatial-Query-By-Sketch, *IEEE Symp. on Visual Languages*, 60-67.
- Egenhofer, M.J. and Franzosa, R.D. (1991) Point-Set Topological Spatial Relations, *Int. Journal of Geographical Information Systems*, 161-174.
- Erwig, M. and Meyer, B. (1995) Heterogeneous Visual Languages – Integrating Visual and Textual Programming, *IEEE Symp. on Visual Languages*, 318-325.
- Erwig, M., Güting, R.H., Schneider, M. and Vazirgiannis, M. (1998a) Abstract and Discrete Modeling of Spatio-Temporal Data Types, *6th ACM Symp. on Geographic Information Systems*, 131-136.
- Erwig, M., Schneider, M. and Güting, R.H. (1998b) Temporal Objects for Spatio-Temporal Data Models and a Comparison of Their Representations, *Int. Workshop on Advances in Database Technologies*, LNCS 1552, 454-465.
- Erwig, M., Güting, R.H., Schneider, M. and Vazirgiannis, M. (1999a) Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases, *GeoInformatica* 3(3), 269-296.
- Erwig, M. and Schneider, M. (1999b) Developments in Spatio-Temporal Query Languages, *IEEE Int. Workshop on Spatio-Temporal Data Models and Languages*, 441-449.
- Erwig, M. and Schneider, M. (1999c) Visual Specifications of Spatio-Temporal Developments, *IEEE Symp. on Visual Languages*, 187-188.

- Erwig, M. and Schneider, M. (1999d) Visual Specifications of Spatio-Temporal Developments, Technical Report 259, FernUniversität Hagen.
- Erwig, M. and Schneider, M. (1999e) Spatio-Temporal Predicates, Technical Report 262, FernUniversität Hagen.
- Gadia, S.K. and Nair, S.S. (1993) Temporal Databases: A Prelude to Parametric Data, *Temporal Databases: Theory, Design, and Implementation* (eds. A.U. Tansel et al.), 28-66.
- Galton, A. (1995) Towards a Qualitative Theory of Movement, *Int. Conf. on Spatial Information Theory*, LNCS 988, 377-396.
- Lee, Y.C. and Chin, F.L. (1995) An Iconic Query Language for Topological Relationships in GIS, *Journal of Visual Languages and Computing* 9, 25-46.
- Segev, A. and Shoshani, A. (1993) A Temporal Data Model Based on Time Sequences, *Temporal Databases: Theory, Design, and Implementation* (eds. A.U. Tansel et al.), 248-270.
- Stonebraker, M. (1986) Inclusion of New Types in Relational Database Systems, *Int. Conf. on Data Engineering*, 262-269.
- Walter, I.M., Sturm, R. and Lockemann, P.C. (1992) A Semantic Network Based Deductive Database System for Image Sequence Evaluation, *2nd IFIP Working Conf. on Visual Database Systems*, 251-276.
- Worboys, M.F. (1994) A Unified Model for Spatial and Temporal Information, *The Computer Journal* 37(1), 25-34.
- Yeh, T.S. and de Cambray, B. (1993) Time as a Geometric Dimension for Modeling the Evolution of Entities: A 3D Approach, *Int. Conf. on Integrating GIS and Environmental Modeling*.
- Yeh, T.S. and de Cambray, B. (1995) Modeling Highly Variable Spatio-Temporal Data, *6th AustraliAsian Database Conference*, 221-230.

BIOGRAPHIES

Martin Erwig received the Diploma degree in computer science from the University of Dortmund, Germany, in 1989. After that he joined the University of Hagen, Germany, where he received his Ph.D. degree (Dr. rer. nat.) in 1994 and his Habilitation in 1999. He is currently on his move to Oregon State University where he is taking an associate professorship. His research interests are in functional programming, visual languages, and spatial databases. Currently, his work in functional programming is focused on abstract data types, in particular, graphs, and his research in spatial databases is guided by a functional modeling approach to spatial and spatio-temporal data types.

Markus Schneider received the Diploma degree in computer science from the University of Dortmund, Germany, in 1990, and the Ph.D. degree (Dr. rer. nat.) in computer science from the University of Hagen, Germany, in 1995. He is currently a research assistant (lecturer) at that university. His research interests were first related to the design and implementation of graphical user interfaces for spatial database systems. Since then, he has worked on the design and implementation of spatial data types (geo-relational algebra, ROSE algebra, realms). Currently, he is interested in research on data modeling for vague or fuzzy spatial objects, on spatio-temporal data modeling, and on partitions in spatial database systems. Moreover, he deals with the design and implementation of data structures and geometric algorithms for these topics.