Visual Specification of Spatio-Temporal Developments

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

Abstract

We introduce a visual language for the specification of temporally changing spatial situations. The main idea is to represent spatio-temporal (ST) objects in a two-dimensional way by their trace. The intersections of these traces with other objects are interpreted and translated into sequences of spatial and spatio-temporal predicates, called developments, that can then be used, for example, to query spatio-temporal databases.

1. ST Objects, Predicates, and Developments

Spatio-temporal objects like flying airplanes, migrating whales, or spreading fires have the characteristic property that they are *spatial entities changing over time* and that these changes are *continuous*. For the modeling of these data the concept of *spatio-temporal data types* [1] has been introduced which includes the treatment of continuous spatial changes. This concept regards ST objects as functions from time to space. In this way, a *moving point* (for example, representing an airplane) is given as a function from time to a spatial data type *point*; it describes a point that changes its location over time. An *evolving region* (for example, representing a raging storm) is given as a function from time to a spatial data type *region*; it describes a temporally changing region that can move and/or grow/shrink and change its shape. ST objects can also disappear and reappear. A very instructive view of ST objects is to consider them as purely geometric, *three-dimensional* objects, that is, the time axis is regarded as a third geometric dimension. An evolving region is then represented as a volume in 3D space, and a moving point is visualized as a 3D curve.

Temporal changes of ST objects, in particular, induce temporal changes of their mutual topological relationships. For example, at one time two evolving regions might be disjoint where some time later they might intersect. An observation is that during certain time periods the topological relationships between both objects are constant and that at certain points in time these relationships change. Consequently, we obtain an alternating sequence of time intervals and time points at which the topological relationships between both objects are constant. Hence, a *spatio-temporal relationship* is nothing but a sequence of spatial relationships that hold over time intervals or at time points.

Topological relationships between two spatial objects are well known; for two regions there exists a canonical collection of eight topological predicates called *equal*, *meet*, *covers*, *coveredBy*, *disjoint*, *overlap*, *inside*, and *contains*. But only the first four predicates can be true at an instant in time; they are called *instant predicates*. The other four predicates can only hold for a period of time; they are called *period predicates*.

If a spatial predicate is valid for each point in time of a certain time interval, we obtain a basic *spatio-temporal predicate* which is written like the corresponding spatial predicate but with an initial capital letter. Such a predicate can be regarded as a function taking two spatio-temporal objects as arguments and yielding a boolean value as a result. Basic spatio-temporal predicates can now be combined to more complex predicates describing *developments* [2]. Developments are represented by alternating sequences of spatio-temporal predicates and spatial predicates and are written down by juxtaposition. Consider the example of an airplane running into a hurricane. This development can be formulated as *Disjoint meet Inside* which we abbreviate by *Enter*. A flight running out of a hurricane can be characterized by

Leave := *Inside meet Disjoint*

A flight traversing a hurricane can be either written as *Enter Leave* using derived predicates or it can be denoted as

Cross := *Disjoint meet Inside meet Disjoint*

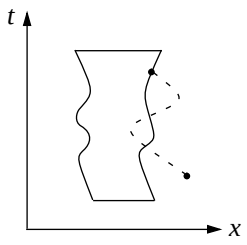
Note that instant predicates and their corresponding spatio-temporal predicates (like *meet* and *Meet*) that occur next to each other in a development can be and have been merged to the respective spatio-temporal predicate. Other examples (to be visualized below) are:

Touch := *Disjoint meet Disjoint*
Bypass := *Disjoint Meet Disjoint*
Graze := *Disjoint meet Overlap meet Disjoint*

2. Visual Specifications of Developments

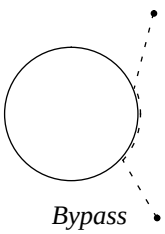
Due to the strong analogy between ST objects and 3D objects we could simply use 3D pictures to specify developments, but there are several reasons for not doing so: first, drawing three-dimensional pictures is much more complex than drawing 2D pictures, especially, when there are no specialized user input devices available. Second, such a drawing interface is also much more difficult to implement; it must offer more options to the user and is thus again more difficult to learn and to apply than a two-dimensional language. Third, 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. 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.

Therefore, we abstract from exact positions/extents, and reduce two-dimensional geometric objects to one-dimensional ones. Then we can use the *y*-axis to represent the temporal dimension. In other words, we "forget" about the *y*-axis with regard to spatial information and 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:

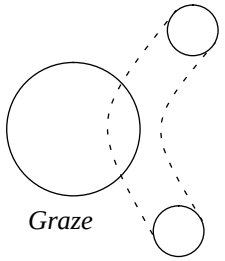


This picture describes a moving point that enters a region, then leaves the region and finally stops on the region's border. It is striking that the sketched movement/ shrinking/growing of the interval representing the evolving region does not contribute anything to this specification. The reason is that we are only specifying topological relationships, and thus we need only information about the relative positions of objects with respect to each other. Therefore, we can always display one of the two objects as a constant, non-moving spatial object.

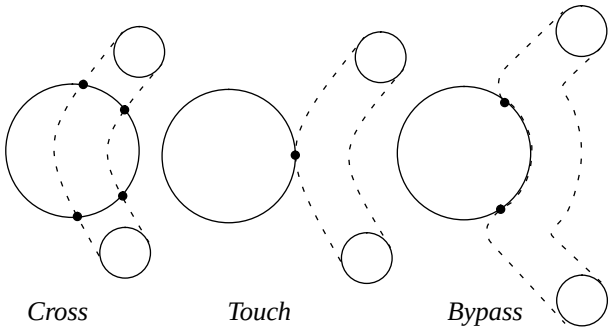
We represent the evolving region in the definition of a point/region predicate (respectively, one evolving region in the definition of a region/region predicate) simply as a (fixed) circle. Likewise, we represent one of the two moving points in a point/point spatio-temporal predicate as a vertical line. This leads to a notation which is easy to understand. For instance, the point/region predicate *Bypass* can be specified as shown on the right.


Bypass

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 the last position) connected by a trace specifying the object's 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:


Graze

The first and 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. Some variations are shown below.


Cross *Touch* *Bypass*

Note that the exact interpretation can always be inferred from the intersections of the trace borders with the static circle. This is explained in the full paper [3].

3. Conclusions

We have shown a simple two-dimensional visual language to express queries 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.

4. References

[1] Erwig, M., Güting, R.H., Schneider, M. & Vazirgiannis, M.: Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases, *GeoInformatica 3*(3), 1999, to appear.

[2] Erwig, M. & Schneider, M.: Developments in Spatio-Temporal Query Languages, *IEEE Int. Workshop on Spatio-Temporal Data Models and Languages*, 1999, to appear.

[3] Erwig, M. & Schneider, M.: Visual Specification of Spatio-Temporal Developments, Report, FernUniv. Hagen, 1999.