Evaluation of Spatio-Temporal Predicates on Moving Objects

Markus Schneider*
University of Florida
Department of Computer & Information Science & Engineering
Gainesville, FL 32611, USA
mschneid@cise.ufl.edu

Abstract

Moving objects databases *managing spatial objects with continuously changing position and extent over time have recently found large interest in the database community. Queries about* moving objects *become particularly interesting when they ask for* temporal changes in the topological relationships *between evolving spatial objects. A concept of* spatio-temporal predicates *has been proposed to describe these relationships. The goal of this paper is to design efficient algorithms for them so that they can be used in spatio-temporal joins and selections. This paper proposes not to design an algorithm for each new predicate individually but to employ a* generic algorithmic scheme *which is able to cover present and future predicate definitions.*

1. Introduction

Geometries evolving continuously over time, so-called *moving objects* [3], have recently received a lot of interest. Two important abstractions are *moving points* like vehicles or animals, for which the time-dependent position is relevant, and *moving regions* like hurricanes or oil spills, for which also the time-varying shape and the extent are important. Temporal evolutions of spatial objects induce modifications of their mutual topological relationships over time, called *developments*. *Spatio-temporal predicates* [1] have been proposed to ask for these time-varying relationships.

The main goal of this paper is to design efficient algorithms for these predicates. This task is of great interest, e.g., to evaluate the refinement step of spatio-temporal joins and selections in queries. Developments as sequences of topological predicates, which are valid for a period or at a time instant, describe *topological patterns* over time and cannot be expressed as compositions of user-defined functions. We propose a *generic algorithmic scheme* that even enables the computation of spatio-temporal (ST) predicates that will be defined in the future.

Section 2 reviews moving objects and spatio-temporal predicates. Section 3 explains the algorithmic scheme for evaluating developments. Section 4 addresses our implementation framework.

2. Moving Objects and ST Predicates

The structure of moving objects as the operands of ST predicates can be described by so-called *unit representations* [2]. The basic idea is to disassemble the temporal evolution of a spatial object into a sequence of slices called *units* such that within each unit this evolution can be described by some kind of "simple" function. Each unit is associated with a time interval representing its validity. For a *moving unit point*, the simple function is a pair $(x(t), y(t))$ of linear functions describing the evolution of x- and y-coordinates in the *unit interval*. For each *moving unit region*, the simple function is a collection of *moving unit segments* representing the evolution of the boundaries of a 2D region object in the unit interval. Each moving unit segment is given by two moving unit points as endpoints.

A development given by a ST predicate is a sequence of well known topological relationships that alternately hold over time intervals or at time points. Consider the query whether an airplane *crossed* a certain storm (Figure 1). The ST predicate *Cross* checks the validity of several relationships during a series of events (described by *topological predicates*) and periods (described by capitalized *basic ST predicates*). We have to examine the existence of a constellation when the airplane and the storm were *disjoint* for a while, when afterwards they *met* at some time, when then the plane was *inside* the storm for a while, when after that the plane again *met* the border of the storm at some time, and when finally the plane and the storm were *disjoint* again. We can observe that during certain time periods the topological relationships between both objects remain constant and that at certain instants these relationships change.

*This work was partially supported by the National Science Foundation under grant number NSF-CAREER-IIS-0347574.

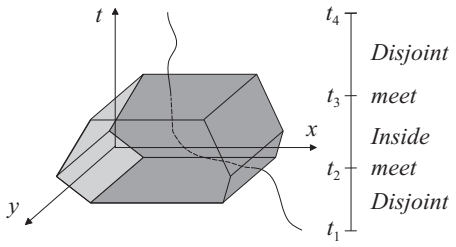

Figure 1. A schematic illustration of the *Cross* development between a moving point and a moving region.

3. Evaluation of Developments

The central idea is to accommodate the unit representations of two moving objects and to derive their development. Our algorithmic scheme includes the following steps:

1. *Time-synchronized interval refinement*. Since matching, i.e., equal, unit intervals can usually not be found in both operand objects, an interval refinement is computed covering all start and end points of unit intervals of both operands. An optimization is based on the fact that a spatio-temporal predicate is only defined at time instants and during periods in which *both* operand objects are defined. Hence, those time intervals can be skipped in which only one of the operands is defined. Finally, both operand objects are synchronized in the sense that for each unit in the first object there exists a matching unit in the second object with the same, possibly reduced unit interval, and vice versa (Figure 2).

2. *Function-valued interval refinement*. Each pair of matching units in both objects has possibly to be further refined depending on the evolutions represented by the two unit functions. This is the case if both moving unit objects intersect or touch each other and thus change their topological relationship (see the intersecting unit functions in the intervals w_1 and w_3 of Figure 2). Hence, this evaluation step depends on the combination of operand types considered. We have to discriminate the cases of an intersection between two moving unit points, between two moving unit regions, and between a moving unit point and a moving unit region. The idea is not to explicitly compute the intersections but to logically infer them. As a result, we obtain the finest interval granularity needed for the determination of the topological relationship between two matching units.

3. *Development determination*. For each pair of matching units we determine the unique, basic spatio-temporal or topological relationship called *unit development*. The central idea here is to represent a unit development as a *string pattern*. We then sequentially collect all unit developments (concatenate all string patterns) and in total obtain the development of the two operand objects of the spatio-temporal query predicate.

4. *Pattern matching with query predicate*. This step includes a (string) pattern matching process between the development computed and the query predicate asking for evaluation. It yields a boolean value.

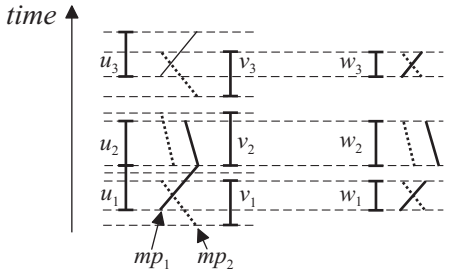

Figure 2. Time-synchronized refinement of the unit interval sequences of moving points.

4. Implementation Framework

The implementation of the algorithmic scheme is part of our *Spatio-Temporal ALgebra* (*STAL*). STAL is a self-contained software library in C++ and implements our spatio-temporal data types, operations, and predicates. It is used as a software extension package integrated into extensible database systems. The communication with a DBMS is provided by a *DBMS-specific adapter* outside of STAL.

The construction of STAL is one of the research goals of the *Space-Time-Uncertainty* (*STU*) project whose overall objective it is to incorporate and combine the three fundamental features of *space*, *time*, and *uncertainty* into the next generation of Geographical Information Systems.

References

[1] M. Erwig and M. Schneider. Spatio-Temporal Predicates. *IEEE Trans. on Knowledge and Data Engineering*, 14(4):1–42, 2002.

[2] L. Forlizzi, R. Güting, E. Nardelli, and M. Schneider. A Data Model and Data Structures for Moving Objects Databases. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 319–330, 2000.

[3] R. Güting, M. Böhlen, M. Erwig, C. Jensen, N. Lorentzos, M. Schneider, and M.Vazirgiannis. A Foundation for Representing and Querying Moving Objects. *ACM Trans. on Database Systems*, 25(1):881–901, 2000.