Towards a Spatio-Temporal OQL for the Four Dimensional Spatial Database System Hawks

Susumu Kuroki
Akifumi Makinouchi

Kensaku Ishizuka

Department of Intelligent Systems
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka 812-81, Japan
tel: +81-92-642-3883, fax: +81-92-632-5204
e-mail: kuroki@is.kyushu-u.ac.jp, ishizuka@db.is.kyushu-u.ac.jp, akifumi@is.kyushu-u.ac.jp

Abstract

This paper discusses a spatio-temporal object query language(OQL), which treats spatial data and temporal data in the same way. Here, we address spatial, temporal and spatio-temporal predicates and operators and then show the queries in the spatio-temporal OQL and the ones in the internal expression of the spatio-temporal database system Hawks. This language is going to be implemented in INADA/ODMG, which is a database programming language based on C++ and provides C++ bindings of the ODMG-93, the Object Database Standard.

The expressions, in which spatio-temporal objects are defined as the figures of the four dimensional topological space resulted from the direct product of the three dimensional space and time, are used to retrieve the spatio-temporal objects which satisfy the condition.

1 Introduction

A spatial query language is an essential tool for interacting with spatial databases. A spatial query is to retrieve spatial data, which describe information that pertains to the space occupied by objects in a database. Spatial data are, for example, points, lines, rectangles, polygons, surfaces, and volumes.[1] Spatial queries posed to map databases are often range queries, which are to retrieve spatial data in the space spanned by the region given by a user.

A temporal query language is also an essential tool for interacting with temporal databases. A temporal query is to retrieve temporal data, which are related to the time during which objects exist in a temporal database. Temporal data are time instances and time intervals.

Spatial query languages and temporal query languages are often studied and used in seperate ways. However, it is time to study them in an integrated way, because new database applications such as computer animation and virtual reality use spatio-temporal queries to retrieve spatial objects which, for example, intersect other spatial objects. Consequently, temporal sequences of spatial queries are asked to databases when the spatial aspects and temporal aspects are not stored in an integrated way. This is often the case with the systems that store spatial data in a temporal layer structure. In such systems, when spatial objects in the condition of a query change thier locations as time passes, the temporal sequence of the outputs of the queries is also time-dependent and obtained by integrating the answers of the spatial queries and temporal queries after the each answers are obtained.

However, the integration of the answers is a very tedious procedure and we are now designing and implementing the spatio-temporal data representation model Universe, which gives users to model time-varying spatio-temporal objects in an unified way so that the integration process is avoided.

In order to process spatio-temporal queries in an integrated way, we have started designing and implementing the four dimensional spatial database system Hawks[5, 6]. It is a spatio-temporal database system and to be implemented in INADA object-oriented database programming language[2]. The database is based on the spatio-temporal data representation model Universe[5]. The model provides users an uniform way of defining, modifying and querying spatio-temporal objects. In the next section, we explain the classes and methods of the spatio-temporal objects so as to show the spatio-temporal representation of the Universe.

2 Representation of Spatio-Temporal Objects in Universe

The model Universe uses the mathematical concept of simplices and simplicial complexes to describe spatiotemporal objects[3]. Here, we assume that the spatial objects are in a three dimensional Euclidean space R^3. They may move in R^3 and may change their topologies, too.

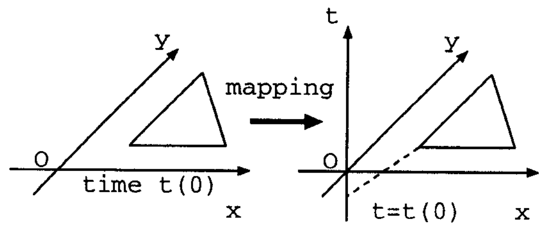

Figure 1: Representation of an instantaneous spatial object. A spatial object in R^3 at time t(0) is mapped into a topological figure in T.

To model the motion of spatial objects, we introduce topological space T(x, y, z, t) which is the direct product of the spatial coordinates (x, y, z) $\in R^3$ and the temporal coordinate t.

First, a spatial object O $\in R^3$ at time instance t is mapped into the topological space T by extending its coordinate (x, y, z) to (x, y, z, t). For example, a triangle *Triangle1* in R^3 at time $t = t_0$ whose vertices are $P_0(x_0, y_0, z_0)$, $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$ is represented as a triangle *Triangle*1* in T, whose vertices are $P_0^*(x_0, y_0, z_0, t_0)$, $P_1^*(x_1, y_1, z_1, t_0)$, $P_2^*(x_2, y_2, z_2, t_0)$(see Figure1).

Then, when the spatial object O in R^3 moves during the time interval I$[t_0, t_1]$, its movement over the interval I induces its own figure *STO* in T. Let us denote the object in R^3 at time t and its representation in T as O(t) and $O^*(t)$, respectively. Then the figure STO is deined as $\bigcup_{t \in [t_0, t_1]} O^*(t)$, the union of the object $O^*(t)$ in T over time interval I.

To represent such kinds of objects in T, we use simplicial complexes. A simplicial complex is a finite set of simplices such that if there exists a pair of simplicies, the intersection of the two is their faces, which are spanned by a subset of their own vertices. A simplex is the minimum convex object whose vertices are linearly independent. Instances of simplices are points(0-dimensional), lines(1-dimensional), triangles(2-dimensional), tetrahedra(3-dimensional) and 4-simplices. They are primitive objects that are used to construct topological figures in T. They play their roles similar to the primitives used in constructive solid geometry. Figure3 and Figure4 illustrate instances of simplices and a simplicial complex, respectively.

In Universe, there are three kinds of spatial data classes to represent points, simplices, and complexes. The structure of the classes are given in the appendix A. The class Complex is used to define spatio-temporal objects which are composed of simplices which are the primitives of spatio-temporal objects. Thus, an instace of the class Complex has membership relationship of the primitives. Also, an instace of the class Simplex has membership relationship of its own vertices. An instace of the class Point has its location in the topological space T. A point object is

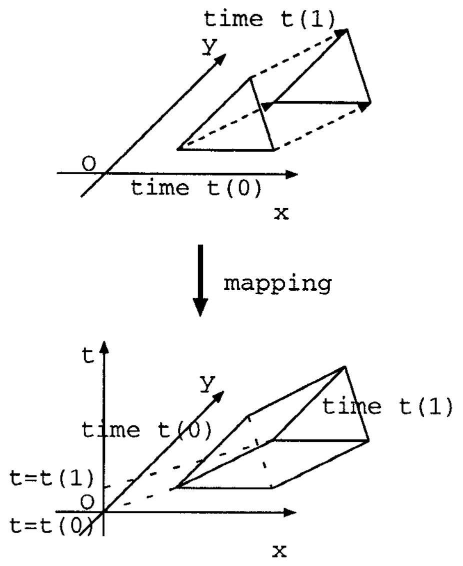

Figure 2: Representation of a time-varying spatial object, i.e. a spatio-temporal object. A spatio-temporal object in R^3 is mapped into a topological figure in T. Here, a translating triangle over time interval [t(0), t(1)] is illustrated.

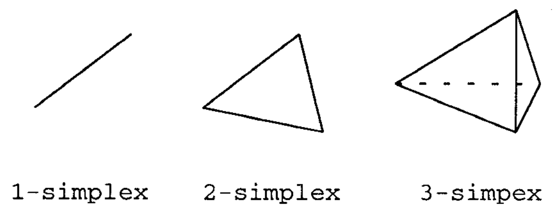

Figure 3: Simplices. When the dimensionality of a simplex is to be explicitly expressed, the term k-simplex is used where k is the dimension of the simplex.

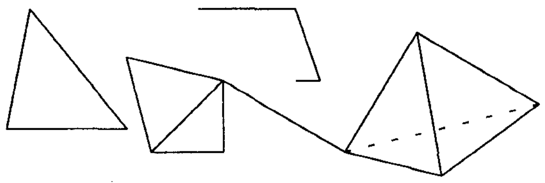

Figure 4: An instance of simplicial complexes. A simplicial complex does not have to be continuous. In addtion, a simplicial complex does not need to be composed of the simplices of the same dimension. In this case, lines(1-simplices), triangles(2-simplices) and tetrahedron(3-simplex) are the member of the set.

linked bidirectionally to the simplices which it is one of the vertices of. Similarly, a simplex object is bidirectionally linked to some instaces of class Complex based on its membership relationship.

3 Spatio-Temporal Queries in OQL

In this section, we give some examples of spatial, temporal, and spatio-temporal queries represented in OQL[4] using spatial, temporal, and spatio-temporal predicates, and operators. OQL is the object database standard and the database programming language INADA provides OQL interface. In the following examples, we use spatial predicate intersect(), temporal predicate overlap(), and spatio-temporal predicate collide() to describe spatial, temporal, and spatio-temporal relationships. The semantics of the predicates intersect() and overlap() are same as the traditional ones. The predicate collide() is true when the two objects meet in some place at some time. These predicates are to be designed and implemented to help users to define typical spatio-temporal queries in the same way as they do with spatial query languages and temporal query languages.

1. *Query 1*

   First of all, an example of spatial queries is addressed. The example is a range query and it retrieves spatial objects such that they intersect a region *range1*: $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1]$ in R^3 specified by the query. This query is expressed in OQL as follows. Here, *SpatioTemporalObjects* is a set of spatio-temporal objects, which have Universe representation. Note that this query does not specify temporal condition at all hence the semantics of the query is to select every objects that intersect with the region *range1* at some time.

   ```
   select c
   from c in SpatioTemporalObjects
   where c.intersect(range1)=true
   ```

2. *Query 2*

   The second one is an example of temporal queries. The query retrieves spatial objects such that they exist at least one moment of the time interval *range2* $[t_0, t_1]$. Here, we use the temporal predicate overlap() to specify a temporal condition. The semantics of the query is to select every objects that exist at least one moment during the time interval *range2* in some place.

   ```
   select c
   from c in SpatioTemporalObjects
   where c.overlap(range2)=true
   ```

3. *Query 3*

   The third one is a kind of spatio-temporal queries and selects objects which intersect a time-invariant *range1* during the time period *range2* $[t_0, t_1]$. Here, the spatial operation `intersection(`*object1*`)` and temporal operation `time_interval(` *object2*`)` return the intersection of c and *object1* and the time interval of *object2*, respectively.

   ```
   select c
   from c in SpatioTemporalObjects
   where time_interval(c.intersection(ran-
   ge1)).overlap(range2)=true in
   (select x
    from x in SpatioTemporalObjects
    where x.intersect(range1))
   ```

4. *Query 4*

   The last one is also an instace of spatio-temporal queries. This selects spatio-temporal objects such that they collide spatio-temporal object *range3*, which moves in R^3 during its overall lifetime interval $[t_0, t_1]$

   ```
   select c
   from c in SpatioTemporalObjects
   where c.collide(range3)=true
   ```

4 Translation from Spatio-Temporal OQL to Universe OQL

In this section, we give a rough sketch of a query mapping module. This module maps the Spatio-Temporal OQL queries into the Universe OQL queries, that is, the ones whose expressions are in the the form of the predicates, operetaions of the Universe data representaion written in the C++-based database programing language INADA. In order to illustrate the functionality of the preprocessor, we address the relation between their inputs and outputs of the module.

We have shown four kinds of the spatio-temporal queries in the previous section and we use them as examples of the inputs.

Spatial query processing in Hawks consists of three procedures as follows(Figure5).

Preprocessing procedure is creating four dimensional objects in T by mapping spatial, temporal, and spatio-temporal conditions of the queries in R^3

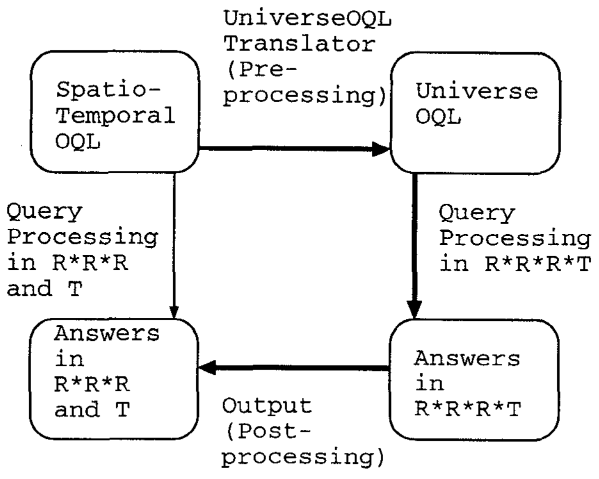
UniverseOQL Translator (Pre-processing)
Spatio-Temporal OQL
Universe OQL
Query Processing in R*R*R and T
Query Processing in R*R*R*T
Answers in R*R*R and T
Answers in R*R*R*T
Output (Post-processing)

Figure 5: Spatio-Temporal Query Processing in Hawks

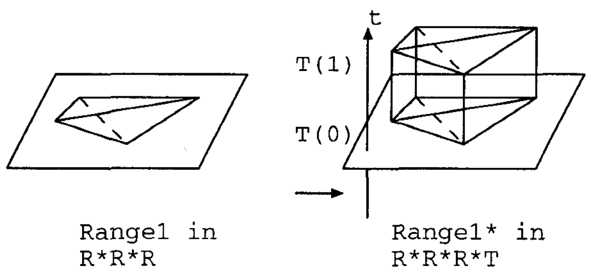

Figure 6: Query regionl RANGE1

into the four dimensional objects in T. Assume that the spatial domain and the temporal domain are $[X_0, X_1] \times [Y_0, Y_1] \times [Z_0, Z_1]$ and $[T_0, T_1]$, respectively. In such a case, the condition in the *Query 1* is transformed into the region *RANGE1*: $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1] \times [T_0, T_1]$ because the condition has nothing to do with the temporal dimension. Similarly, the conditions in *Query 2*, *Query 3*, and *Query 4* are transformed into *RANGE2*: $[X_0, X_1] \times [Y_0, Y_1] \times [Z_0, X_1] \times [t_0, t_1]$, *RANGE3*: $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1] \times [t_0, t_1]$, and *RANGE4*: $\bigcup_{t \in [t_0, t_1]} range3^*(x, y, z, t)$, respectively. These mappings are illustrated in Fugure6 and Figure7.

According to this transformation, the *Query k*(k = 1, 2, 3, 4) in the previous section are transformed into the same representation in Universe OQL, where the parameter *RANGEk* is query-dependent(Figure8). Semantics of the queries are quite different from each other, but their expressions in Universe OQL are the same. The differnece of the semantics is mapped into the difference of query regions **Rangek**. Therefore, we can treat spatial query, temporal query, and spatio-

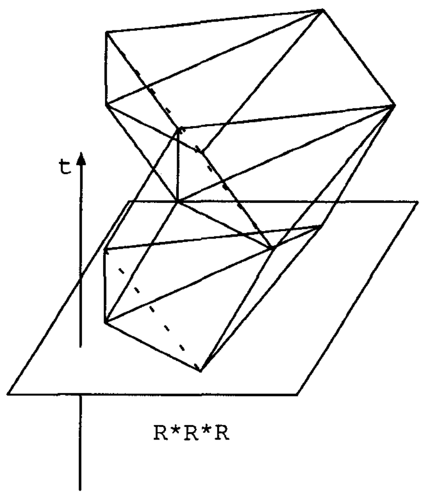

Figure 7: Query region RANGE4. Temporal changes in query regions such as moving objects, topological changes such as split, fusion, metamorphoses are expressed in the shape of the figures in T.

Query k

```
Set<Ref<Complex>>     COMPLEXES;
Ref<Complex>          Rangek;

oql(answer,  "select c
 from c in COMPLEXES
 where c.intersect($1k) = true",
 Rangek);
```

Figure 8: Universe OQL

temporal queries in the same way in which the conditions of the queries are expressed with generalized predicates and operators provided by the Universe.

Then Universe OQLs are processed with four dimensional computational geometrical methods, which are performed with the predicates and operators provided by the Universe.

After that, the result are postprocessed with the operators such as `cross_section()` and `projection()` to interprete four dimensional objects from the users' point of view, apart from the Universe representation. The spatio-temporal operators such as `cross_section()` and `projection()`, which calculates the `cross_section()` in R^3 of the object by intersecting the object and a plane $\pi : t = t_0$ in T to know the location and shape of the objects at time t_0, and union of the `cross_section(t)` in R^3 over time interval of the object, respectively. The operator `cross_section` is similar to the snapshot operator in temporal databases, but the operator `cross_section` can be applied to any moment, not the current time.

5 Related Work

One of the most related works on the spatio-temporal logic is STL[7], that is going to be discussed. STL is based on Temporal Logic and Symbolic Projection. The logic is used to describe contents of image sequences using Symbolic Projection. Symbolic Projection is a tool for encoding the locations of the objects in a scene in , for example, 2D-strings such as (i, j) where (i, j) indicates the location of the rectangles partitioning the image. Using this expression, spatio-temporal objects at time instance t is expressed as a set of the indicaters such as (i, j). Using the indicators, the objects are expressed as a set of rectangles whose edges are parallel to one of the axis. This description reduces spatial logic into the interval one. However, our expression is based on simplicial complexes and the logic is not a combination of the interval logic. This differentiate our expression from the ones in STL.

In addition, spatial relationship of the two objects in a scene are described in the form of assertions beforehands. Consequently, spatial reasoning about the scene is limited to what is described by a set of assertion. On the other hand, our method stores every objects as a four dimensional figures in a database without giving a limited set of assertions and spatial reasoning and retrieval of the spatial objects in a scene is possible. This also differentiate ours from STL. And spatial relationships are limited to topological ones such as overlaps and adjacent. As a result, metric queries such as nearest neighbour query cannot be answered in STL.

6 Conclusion

In this paper, we have addressed the spatio-temporal OQL and its translator into the Universe data representation which we are going to develop for our spatio-temporal database system Hawks. We are going to make the expressions in the spatio-temporal OQL common and familiar to the users so that users can write their queries. In addition, we are going to make the expressions in Universe universal to define spatio-temporal objects and queries. The expressions in Universe is based on the mathematical concept of simplicial complexes, which is an effective tool for modeling the figures in the four dimensional topological space.

Transforming the spatial, temporal, and spatio-temporal conditions into figures in the topological four dimensional space is the heart of the idea and we are now designing it eagerly. The transformation can make complicated spatio-temporal queries such as nested spatial and temporal queries into simpler spatio-temporal ones. This is one of the features of our approach.

Acknowledgments

This research was supported in part by The Ministry of Education, Science, Sports and Culture, Japan under Grant-in-Aid for Scientific Research on Priority Areas(Grant No. 08244105).

References

[1] H. Samet. Spatial Data Models and Query Processing. *Modern Database Systems*(W. Kim ed.). Addison-Wesley Publishing Co., 1995.

[2] M. Aritsugi, and A. Makinouchi. Design and Implementation of Multiple Type Objects in a Persistent Programming Language. Proc. IEEE 19th Int. Computer Software and Application Conf.(COMPSAC'95). pp.70-76, 1995.

[3] M.J. Egenhofer, A.U. Frank, and J.P. Jackson. A Topological Data Model for Spatial Databases. Proc. 1st Symp. SSD'89. pp.271-286. 1989.

[4] R.G.G. Cattell (ed.). *The Object Database Standard: ODMG-93 Release 1.2.* Morgan Kaufmann, 1996.

[5] S. Kuroki and A. Makinouchi. Design of the Spatio-Temporal Data Model Universe using Simplicial Complexes. IPSJ SIG Notes 109-37. 1996(in Japanese).

[6] H. Horinokuchi, S. Kuroki, and A. Makinouchi. Design and Implementation of R*-tree for Spatiotemporal Index. Proc. IPSJ Int. Symp. Next-Generation Information Systems and Technologies. 1997(to appear).

[7] A. Del Bimbo, E. Vicario, and D. Zingoni. Symbolic Description and Visual Querying of Image Sequences Using Spatio-Temporal Logic. IEEE Trans. Knowledge and Data Eng., 7(4). pp.609-622. 1995.

A Classes in Universe

The structure of the classes in Universe are given below.

```
class Point: public Persist_Object{
 public:
 //Attribute
 int id_number;
 double x[4];
 //Relationship
 d_Rel_List<Simplex, _has_point>
 is_a_point_of;
 };

class Simplex: public d_Object{
 public:
 //Attribute
 int id_number;
 int dimension;
 //Relationships
 d_Rel_List<Point,  _is_a_point_of>
 has_point;
 d_Rel_Ref<Complex,  _contains>
 belongs_to;
 //Operation
 Simplex& operator+
=(const d_Ref<Point>  p);
```

```
 Simplex& operator-
=(const d_Ref<Point>  p);
 Point* coordinate();
 //Spatial Set Operation
 Complex& Intersection(const Simplex s);
 Complex& Difference(const Simplex s);
 Complex& Union(const Simplex s);
 //Spatial Relational Operation
 int intersect(const Simplex s);
 int disjoint(const Simplex s);
 int contain(const Point p);
 //Spatial Aggregate Predicate
 double Volume();
 };

class Complex: public d_Object {
 public:
 //Attribute
 int id_number;
 //Relationship
 d_Rel_List<Simplex, _belongs_to> contains;
 //Operation
 Complex& operator+
        =(const d_Ref<Simplex> s);
 Complex& operator-
        =(const d_Ref<Simplex> s);
 //Spatial Set Operation
 //(1)The case of having no common
      part between Complex and Complex c
 Complex& plus(const Complex c);
 Complex& minus(const Complex c);
 //(2)The case of having common
      parts between Complex and Complex c
 Complex& Intersection(const Complex c);
 Complex& Difference(const Complex c);
 Complex& Union(const Complex c);
 //4D Geometric Operation
 Complex& Cross_section(const Simplex s);
 Complex& Projection(const Simplex s);
 //Spatial Relational Predicate
 int intersect(const Complex c);
 int contain(const Complex c);
 int disjoint(const Complex c);
 //Spatial Aggregate Predicate
 double Distance(const Complex c);
 double Volume();
 };
```