A Semantics for Version Queries In GIS Databases

Hanna Kemppainen

Finnish Geodetic Institute, Geodeetinrinne 2, FIN-02430 Masala, Finland,
hanna.kemppainen@fgi.fi

Abstract

A formal semantics for spatio-temporal version queries is presented. A version query specifies criteria against which change information is obtained from a database. The query criteria may concern spatial and temporal characteristics and indexed states of objects. The semantics is given using an object calculus, where the calculus terms refer to versioned geographical objects, their changing states and spatial and temporal relationships. The purpose of specifying the semantics of queries is to aid in understanding the exact meaning of spatio-temporal queries better. The work provides a framework for future research in modelling of complex identity-affecting mutations of objects and related queries.
Keywords: spatio-temporal, object-oriented, change, query

1 Introduction

Implementation of a GIS that supports various aspects of object-based change is founded on models of change itself and models of queries to get information of the stored change data. A model of change can be based on an object-oriented model of geographical information where a geographical object consists of spatial, aspatial and temporal components (Worboys, 1992, 1994b). A change in an object may involve the identity, spatial or aspatial attribute values or the thematic content of the object. A software system that supports change should not only be able to store data of the changing states and identities of the objects. It should also provide a query language to specify criteria against which change information is obtained from a database. A query language should provide support for queries where the user wants to find out what happened to an entity at a given time or during a given time interval. The user should also be able to ask what happened to an entity when a related entity changed at a given time, or how an entity changed between two

consecutive changes of a related entity. Such analysis capabilities are not provided by current commercial GIS.

A model of extracting data from a database is often given as an SQL-like query language. Several languages exist that augment the standard SQL with temporal constructs (Snodgrass, 1992). A query language that integrates both spatial and temporal aspects is needed for geographical applications (Egenhofer, 1994), and such languages have been developed (Griffiths *et al*. 2001, Kemppainen, 2001). This paper does not present another SQL-like query language, but examines how the *semantics* of change queries can be expressed. Starting with queries expressed in an SQL-like language, the objective is to look for a semantics that aids in finding a solution to implement them. This can be accomplished if the semantics is given in terms that are near to the concepts used in implementing systems, for example objects and methods. A formal semantics also aids in understanding the exact meaning of the queries.

Specification of the semantics of change queries is studied here using a logical language called spatio-temporal version calculus. The calculus, as any formal language, consists of syntax and semantics. The syntax specifies the symbols that may be used to express spatio-temporal version queries and how those symbols may be arranged to create well-formed query formulas. The semantics of the calculus specifies how meaning is ascribed to the symbols and the well-formed query formulas. The calculus is designed such that the terms of the calculus correspond to geographical objects and their spatial and temporal operations. Given a set of database objects I, the semantics of a query q specifies which objects are to be included in the query result. The calculus approach to specifying semantics of queries is adapted from the field of database theory for relational query languages (Abiteboul *et al*. 1995, Gallaire *et al.* 1984). An object calculus is also suggested as a query language for geographic databases in (Clementini *et al*. 1993).

The remainder of the paper is organized as follows. Related work is discussed in section 0. A note is made in section 0 as to how change data can be stored in GIS systems. The paper presents a semantics for some typical query types (section 0) based on a conceptual model of version data (section 0). The semantics is similar to the relational query semantics, which is reviewed in section 0. The main contribution of the paper is the object calculus and its use in expressing version queries and their semantics. This is the topic of the remaining chapters.

2 Related Work

Geographical change and temporal GIS have been popular research topics for over a decade (Armstrong, 1988, Langran, 1992, Peuquet, 1994, Frank *et al*. 2000). Conceptual models of change have been proposed for object-based representations, where the object mutates to another object or the object state changes (Claramunt *et al*. 1995). Hornsby & Egenhofer developed a change

description language that can be used to graphically represent operations on identity of simple and composite objects (Hornsby *et al*, 1997, 1998). Medak (1999) developed a theory of lifestyles in terms of operations affecting object identity. The theory is presented as an algebra, i.e. as types, operations and axioms that define their behaviour.

Implementation models have been proposed to store change-related information in GIS databases, relational or object-oriented (Langran, 1989, Kemp *et al*. 1992, Tryfona, 1998). Change that does not affect the object identity can be modelled as object versions (Claramunt *et al*. 1995, Wachowicz *et al*. 1994). A unified model to space and time and its implementation using simplicial complexes was introduced in (Worboys, 1994a).

Research in query languages for changing geographical objects has been somewhat less intense compared to the development of conceptual models of change. Typologies of spatio-temporal queries are presented in (Peuquet, 1994, El-Geresy *et al*. 2000). A model of extracting data from a database is often given as an SQL-based query language. Several query languages have been proposed for temporal (Snodgrass, 1992, Chomicki, 1994) and spatial applications (Egenhofer, 1994, Herring, 1987). A recent contribution augments the syntax of an object oriented query language OQL (Cattell *et al*. 2000) with spatial and temporal operations and index numbers on object states (Griffiths *et al*. 2001). An earlier work by the author of this paper developed a similar but SQL-based query language that also contains constructs to refer to the version history of an object (Kemppainen, 2001).

3 Support for Change in GIS

It has been stated that implementation of a temporal GIS is still problematic and support for tracking changes in spatio-temporal databases is not effective (Peuquet, 2001, Griffits *et al*. 2001). Current GISs provide two kinds of mechanisms that support storing change-related data: user-defined attributes and version management. Users may time-stamp their data by storing a temporal value as one of an object's attributes. To track the evolution of a database object one can chain different objects representing the same real world entity. Some systems implement *database versions* rather than *object versions* to provide users with simultaneous access to data. Version management systems (Easterfield *et al*. 1990) consider change as a tree of database alternatives. A database alternative is a database view that corresponds to some state S_i of the database, and each sub-alternative S_{ii} derived from S_i provides another view to the database by updating S_i independently of other sub-alternatives. Object's representation in each of the database alternatives corresponds to a version of an object.

The purpose of such a versioning mechanism is to allow different users to work simultaneously with the same data rather than to provide a representation of the evolution of objects. If, however, database alternatives are used to store evolution

data, a specific meaning is assigned to each database alternative. For example, there might be a cadastral database alternative, utilities alternative, etc. To track the “evolution” of a single object in different alternatives, the alternative tree has to be traversed to extract the object information in each of the alternatives. It is left to the application programmers to extract the version data from different alternatives and make analysis as to what changes the entity has experienced, and how the different versions of the entity relate to other entities.

4 Examples of Queries on Time-Varying Geographical Objects

The starting point for this work is provided by a typology of spatio-temporal queries by Peuquet (1994). The queries are not given with respect to any data model, and no indication is made as to how they should be implemented. The formulation of the queries suggests a natural approach to consider changing geographical phenomena, for example, when making observations of the landscape by eye, or studying maps or plans.

The first class of queries addresses changes in an *object* or *feature*: "How has this object changed or moved over the last five years" and "Where was this object two years ago". These queries involve some real world entity ("this object") that is to be monitored at different points in time. Such an entity is thus identifiable, but not necessarily well defined. The problem is that we should identify exactly the same entity at different time instants.

The second class of queries involves changes in the spatial distribution of an object or a set of objects; for example, "What agricultural areas in January 1980 changed to residential land use areas as of December 1989" and "Did any land use changes occur in this drainage basin between t_1 and t_2". Entities may not be well defined here either. As far as land use areas are concerned, it may be questioned whether a land use area should be considered as an identifiable object at all; rather, such areal phenomena might be abstracted as fields. If they are considered as discrete objects, it should be possible to model *class changes* of objects and provide support for such change queries.

Class III queries address the temporal relationships among multiple geographic phenomena, for example, "Which areas experienced a landslide within one week of a major storm event?" In such a query, both the thematic attribute and the spatial characteristics of objects are related.

Based on this framework, geographical real world entities are considered here as spatio-temporal objects that are elements of some class. The name of the class reveals the thematic content of the object (for example, Road class contains road objects). Objects have aspatial, spatial and temporal characteristics. A change may occur that concerns any apatial or spatial value of the object, or the class of the object may change.

5 Version Model and Model of Version Queries

5.1 Conceptual version model

Modelling of changing geographical entities as versioned objects is suggested in e.g. (Wachowicz *et al*. 1994). A versioned object consists of a sequence of versions. The versioned object denotes the changing real world entity, and each version in the version sequence denotes an entity state during some time interval. Change of an object to another object, or other mutations affecting object identity (Claramunt *et al*. 1995) are not supported. Using this model, a road object *Road-A* would be represented as a versioned object, whose changing states correspond to versions *Road-A-1*, *Road-A-2*, etc. The version scenario is depicted in Fig. 1, where versioned objects are depicted as small squares, connected to a sequence of circles that denote versions.

Possible queries for versioned objects are identified by examining potential relationships between versioned objects and their versions. Different ways of associating versioned objects are depicted using large rectangles in Fig.1. Given versionable objects O_1 and O_2, with a set of versions V_1 and V_2, respectively, a query may involve the location of an element of V_1 in the version sequence (similarly for V_2). Such a query extracts versions using expression such as "get the latest version of the object" or "get the first and the second version of the object", to determine how the entity has changed. Another category of queries relates versions based on their aspatial, spatial, and temporal characteristics.

The limitations of this version scenario are the following: 1) Many geographical phenomena are difficult to identify so that exactly the same entity or phenomenon can be detected, say, years later. 2) Creating new versions causes data duplication, which may result in inefficient systems unless the storage problem can be resolved efficiently.

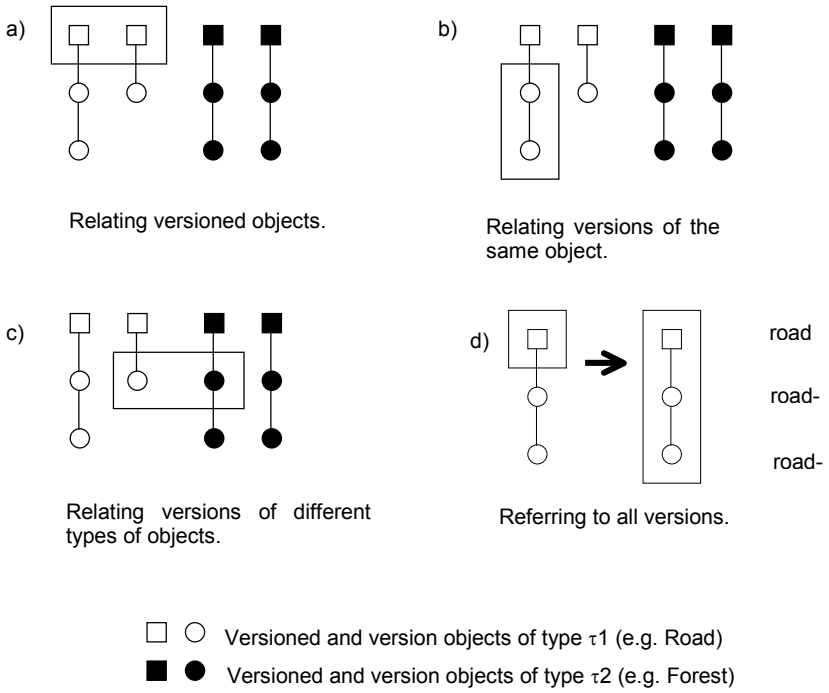

Fig. 1. Relationships between versioned objects

5.2 SQL-Like Version Queries

Version queries are expressed using a *select-from-where* expression, which is the basic construct of SQL, the standard relational query language (explained in e.g. Elmasri *et al*. 1989). Similar expression, but with a somewhat different syntax, is also used in OQL, a standard for object database queries (Cattell *et al*. 2000). The SQL-like version query language developed in (Kemppainen, 2001) is used to express queries whose semantics is studied in this paper.

Let us suppose the data about the versioned objects is stored in a relational database. There are two tables: one for versioned objects and one for versions. A version query would be expressed as:

select versioned_exp | version_exp
from relation_name [,relation_name*]
where select-condition

Here *relation_name* refers to the tables storing data of versioned and version objects. The *select* clause identifies the data that will comprise the query result; for example, *versioned_exp* is an expression that refers to the whole versioned object. In this case, the select clause would be formulated as "*select versioned*", to select

all the versions that exist of the object. The expression might be used in queries such as "Did something change about this object?" The query would return all the versions of the versioned object, for which conditions specified in the select condition apply. A *version_exp* may be of the form 1) "*select version.aspatial*", 2) "*select version.spatial*" or 3) "*select version.temporal*", where *aspatial*, *spatial* and *temporal* denote the corresponding attributes of the version, or 4) "*select version*" if the whole version is asked for.

The select condition is expressed as a combination of version sequence clauses and aspatial, spatial and temporal clauses. An example of a version sequence clause is "*version in (last –1, last)*", to denote the latest version and the one preceding it. A spatial clause uses spatial operators (Egenhofer *et al*. 1991) to evaluate relationships between spatial components of versions, for example, "*version_1.spatial inside version_2.spatial*" to relate spatial components of the two versions using is a spatial operator *inside*. In a similar manner, we express aspatial clauses using standard comparison operators $<, \leq, >, \geq, =, \neq$, and temporal clauses using standard temporal operators (Halpern *et al.* 1991). When relating temporal characteristics of objects, we use temporal expressions to refer to time instants or time intervals given with respect to some calendar value, or *now*. For example, to denote a time interval that started two years ago and ends at the current moment, we would specify "*last ("year", 2, now, "interval")*".

Examples of queries and their semantics are given in section 9.

6 Specification of the Semantics of Queries

The semantics of queries is discussed in terms of relational database theory in (Abiteboul *et al*. 1995, Gallaire *et al*. 1984). These principles are extended to object-oriented databases for versioned geographical objects.

From a logic perspective to databases, one considers a relational database as an interpretation of a first-order language, where the atomic constructs of the language are *n*-ary predicates $P(x_1,\ldots, x_n)$, and equality operator =. A predicate is understood as a tuple in a relation, and is deemed true in the interpretation if the corresponding tuple exist in the database. Queries are expressed as logical formulas. Their truth-value can be determined when we know how the constants of the language are interpreted, and how the variables occurring in the formulas are valuated. The truth of a formula is derived with respect to the values that are stored in the database. An answer to a query q, presented as a logical formula φ, is obtained by finding a valuation of variables in φ that makes φ true.

To put this formally, we consider a relational database with *n*-ary relations R_i. A fact over R_i is an expression of the form $R(a_1,\ldots,a_n)$, where a_i is an alphanumeric value. A relation instance over R is a finite set of facts over R. A database schema **R** consists of the relation names, $\mathbf{R} = \{R_1, \ldots, R_n\}$ and a database instance is a finite set **I** that is the union of relation instances over R_i, for $R_i \in \mathbf{R}$.

Let L be a first order language whose domain is **dom**, an infinite set of alphanumeric values. In standard logic, the constants of the language are used to denote elements of the domain, but it is customary in the database context to consider elements of the domain as the constants of the language also. Predicate symbols $R_1,\ldots, R_n$ denote real predicates among the individuals of the domain. In this setting, a database instance is understood as a finite interpretation **I** of L. This means that the tuples in the database are interpreted as true facts about the universe, and negation of a fact can be derived if the corresponding tuple is not found in the database. The naming of the domain elements is arranged such that each element of the domain is named, and individuals with different names are different.

Queries would be expressed in the relational calculus in the form $q = \{e_1,\ldots, e_m \mid \varphi\}$ where $e_1,\ldots, e_m$ are variables that are free in φ. The semantics of a query is given as an image of database instance **I** under q relative to **d**:

$q_{\mathbf{d}}(\mathbf{I}) = \{\nu([e_1,\ldots, e_n]) \mid$ where ν is the *valuation* over free variables of φ with range contained in **d**, and **I** satisfies φ for $\nu\}$. The underlying domain **d**, with $\mathbf{d} \subseteq$ **dom**, is used to permit us to talk of different underlying domains. The set of all constants occurring in **I** is contained in **d**.

The semantics of spatio-temporal version queries shall be given here in the form of an object calculus. The calculus consists of 1) appropriate domains for versioned geographical objects, 2) an alphabet and rules to use the alphabet to build well-formed query expressions, and 3) specification of truth-values of the calculus expressions with respect to the set of objects that exist in the database.

7 Basic Notions for the Object Calculus

A query q is expressed as $q = \{x_i \mid \varphi\}$, where φ is an object calculus formula and x_i denotes a free variable in φ. The calculus is designed such that the atoms in φ denote meaningful constructs in the geographical application domain. The individuals of the domain are abstracted as complex values and objects.

The calculus is similar to the complex value calculus presented in (Abiteboul *et al*. 1995), extended to incorporate object-oriented characteristics.

7.1 Domains

The domain of the calculus consists of the standard domains **integer**, **string**, **bool** and **float** whose disjoint union is **dom**. Complex value domains are defined recursively using the following abstract syntax: $\tau =$ **dom** $\mid [B_1 : \tau,\ldots, B_k : \tau] \mid \{\tau\}$, where $[B_1 : \tau,\ldots, B_k : \tau]$ denotes a tuple value, and $\{\tau\}$ is a set value. Objects are referred to using elements of **obj**, which is a finite set of object identifiers, **obj** $= \{o_1,\ldots, o_n\}$.

Spatial objects are elements of the spatial domain, with **spatial** $\subseteq$ **obj**. Given an OID assignment π for spatial classes *Point*, *Line* and *Area*, a finite set of spatial objects is denoted by $\pi(Point) \cup \pi(Line) \cup \pi(Area)$. Temporal values are contained in temporal domain **temp**, whose objects can denote time-stamps given in valid time or transaction time. We assume **versioned**, the domain for the object identifiers of versioned objects, with **versioned** $\subseteq$ **obj**, and **version**, the domain for the object identifiers of **version** objects, with **version** $\subseteq$ **obj**. The domains **spatial**, **temp**, **version** and **versioned** are pair-wise disjoint.

The underlying object model abstracts a changing geographical entity as a *versioned object*, whose changing states are described as *versions*. Each object version may have spatial, aspatial and temporal components. For example, changing real world roads would be modelled using *Versioned_Road* class. An object of *Versioned_Road* would consist of an ordered set of *Road_Version* objects, each of which is an aggregation of values coming from the **spatial**, **temporal** and **aspatial** domains. Objects of *Versioned_Road* and *Road_Version* come from **versioned** and **version**, respectively.

7.2 Predicates for Version Queries

Relationships between objects are modelled as predicates $P(o_1,\ldots, o_n)$. For the purposes of defining the semantics of queries, for a given predicate P, it is sufficient to know whether P holds for some valuation of the object variables standing for $o_1,\ldots, o_2$.

To formulate spatial and temporal selection conditions we assume a set of spatial predicates *SP* and temporal predicates *TP*. Comparisons between numeric and textual values are included in the calculus operations. The definition of spatial and temporal relationships is discussed in e.g. (Egenhofer *et al*., 1991, Allen, 1984, Halpern *et al*. 1991).

$SP = \{D, SM, SO, COV, COVB, I, CON, SE\}$
$TP = \{B, TE, TM, TO, D, S, F, AS, AE\}$

Spatial predicates *SP* refer to spatial operators disjoint (*D*), spatial_meet (*SM*), spatial_overlap (*SO*), covers (*COV*), covered_by (*COVB*), inside (*I*), contains (*CON*), and spatial_equal (*SE*). Temporal predicates correspond to temporal operators before (*B*), temporal_equal (*TE*), temporal_meet (*TM*), temporal_overlap (*TO*), during (*D*), starts (*S*), finishes(*F*), at_start (*AS*) and at_end (*AE*).

8 The Object Calculus

The object calculus for expressing version queries is based on the complex value calculus of (Abiteboul *et al*. 1995), extended with some object-oriented notions.

The alphabet of the object calculus (as that of any formal language) consists of constants, variables, logical connectives and quantifiers, and of rules that govern the formulation of well-formed formulas.

Constants and variables For each domain, a countable finite set of constants and variables of that domain is assumed.

Terms are interpreted as standing for an individual of the domain. Constants and variables are terms. An object variable stands for an object, and a tuple variable denotes a complex value that is a tuple. A field A of a tuple variable x (of sort [A : **dom**]) is referred to as $x.A$.

Atomic formulas:

a) A predicate $P(t_1,\ldots, t_n)$ is an atom ($t_1,\ldots, t_n$ are terms).

b) Atoms for simple values: $t = t'$ and $t < t'$ are atoms (t and t' are terms).

c) Atoms for complex values: $t \in t'$, $t \subseteq t'$, t and t' are terms and the appropriate sort restrictions apply ($\in$,$\subseteq$ are many sorted).

d) Object-oriented characteristics of the calculus are supported by 1) value equality $o = o'$, and 2) identity equality $o =_{id} o'$ for object variables o and o'. To compare values returned by methods we have: $t = m(t_1,\ldots, t_n)$ and $m_1(t_1,\ldots, t_n) = m_2(t_1,\ldots, t_m)$ where $t, t_1,\ldots, t_n ,\ldots,t_m$ are terms and m, m_1 *and* m_2 denote method names. The value of $m(t_1,\ldots, t_n)$ is obtained by evaluating the implementation of m under a variable assignment μ. that associates values to $t_1,\ldots, t_n$. The method signature reveals the type of the value which m returns.

Formulas are created from atomic formulas using the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and quantifiers $\forall$ and $\exists$.

8.1 Semantics of the calculus

The truth-value of an atomic formula is defined like the standard relational calculus. For example, an equality atom $\varphi = (s = s')$ is true in an interpretation **I** if the valuations of the two variables are equal, i.e. $\nu(s) = \nu(s')$. We say that **I** satisfies φ for ν, relative to **d**, denoted by $\mathbf{I} \vDash_{\mathbf{d}} \varphi[\nu]$. If φ is formed using logical connectives for example if $\varphi = (\psi \wedge \xi)$, $\mathbf{I} \vDash_{\mathbf{d}} \varphi$ if $\mathbf{I} \vDash_{\mathbf{d}} \psi[\nu']$ and $\mathbf{I} \vDash_{\mathbf{d}} \xi\,[\nu']$, where ν' is a restriction of ν to free variables of the formula. For other forms of φ, we shall only discuss the interesting features that are related to complex values and objects. Given a database instance **I**, $\mathbf{I}(C)$ denotes an instance of class C, and $\mathbf{I}(P)$ denotes an instance of relation P. The truth of a formula φ in **I** is given as follows:

(a) $\varphi = C(o)$ is true if $\nu(o) \in \mathbf{I}(C)$ and ν is a valuation of object variables.

(b) $\varphi = (s = s')$ is true if $\nu(s) = \nu(s')$. s,s' denote variables of any domains (value equality)

(c) $\varphi = (s =_{id} s')$ is true if $\nu(s) =_{id} \nu(s')$. s,s' denote variables of object domains (identity equality)

(d) $\varphi = (s = m(s_1,\ldots,s_n))$ if $\nu(s) = m(\nu(s_1),\ldots, \nu(s_n))$ and $\nu(s)$ complies with the method signature of m under ν, especially we require that $\nu(s)$ is of the correct

type. Similarly, $m_1(s_1,\ldots, s_n) = m_2(s_1,\ldots, s_m)$ if $m_1(\nu(s_1),\ldots, \nu(s_n)) = m_2(\nu(s_1),\ldots, \nu(s_m))$ and the values returned by m_1 and m_2 are of the same type.

(e) $\varphi = (s \in s')$ and $\nu(s) \in \nu(s')$

(f) $\varphi = (s \subseteq s')$ and $\nu(s) \subseteq \nu(s')$

(g) $\varphi = P(o_1,\ldots,o_n)$ if $[\nu(o_1),\ldots, \nu(o_n)] \in \mathbf{I}(P)$

9 The Semantics of Version Queries

The version queries are formalised as $q = \{x \mid \varphi\}$, where φ is an object calculus formula and x is free in φ. For example, to select a versioned object from the database we use a query language statement "*select versioned*" which corresponds to a query $\{r \mid \varphi(r)\}$, where r is a variable ranging over **versioned**, and r is free in φ.

The object calculus formulation of some of the queries discussed in section 0 requires a sample database schema and a set of objects of the schema classes. Let *Versioned_Road, Road_version, Versioned_landuse, Landuse_version* and *Line* denote classes of a schema *S*. A database instance *I* consists of objects belonging to classes of *S*. Methods *M* are used to extract attribute values and versions of objects. A method definition is given as a signature $m : c \times \tau_1 \times \ldots \times \tau_{n-1} \rightarrow \tau_n$, where c is a class name and τ_i is a type over *C*, the set of all class names (Abiteboul *et al*. 1995). This signature is associated with class c; we say that method applies to objects of class c. $\tau_1,\ldots,\tau_{n-1}$ denote parameter types and the method returns a value whose type is τ_n. A coded definition of m is the piece of program code to implement the method.

For example, for class *Road_version* with attributes *aspatial*, *spatial* and *temporal*, method *spatial* gets the value of the spatial attribute: *spatial* : *Road_version* $\rightarrow$ *Spatial*. Temporal component is returned by a method whose signature is *temporal* : *Road_version* $\rightarrow$ *Temporal*. Method *version*(*Versioned*, {**int**}) $\rightarrow$ {version} extracts versions of an instance of a versioned class The method returns a set of versions specified by a set of integers as a method parameter.

A query "Has Road A moved in the last two years?" would be formulated as an SQL-like query as follows:

select version
from *Versioned_Road, Road_version*
where root = "road-A" **and version.temporal during last** ("year", 2, **now**, "interval")

This query extracts the versions of Road A, whose time stamp indicates that the version existed during a time interval specified by **last** ("year", 2, **now**, "interval"), for example [1.1.1999,31.12.2000], if the query was performed at 31.12.2000.

The query extracts versions whose spatial components have to be compared to determine whether a change occurred during the specified interval of time.

The following calculus query gets all versions of Road A:

$$\{v \mid \exists r\ (Versioned\_Road(r) \wedge name(r) = \text{"}road\text{-}A\text{"} \wedge v \in versions(r)\}$$

We restrict the query result further by requiring that the temporal value associated with the version overlaps a temporal constant tc_1, where tc_1 denotes a time interval starting two years ago and ending *now*.

$$\{v \mid \exists r\ (Versioned\_Road(r) \wedge name(r) = \text{"}road\text{-}A\text{"} \wedge v \in versions(r) \wedge D(temporal(v), tc_1)\ )\}$$

Here variable r denotes an object for which predicate $Versioned\_Road(r)$ should hold. Answer to the query is obtained by a valuation for r such that r denotes an object from the *Versioned_Road* class. Let o be such an object. The query also requires that the method *name* applied to o returns "*road-A*". A free variable v is used to denote an object that belongs to the set of versions of o. Finally, temporal predicate D (for "during") is used to determine whether the temporal component of version v takes place during the time interval denoted by a temporal constant tc_1. Once we have found a valuation for r and v such that the formula holds, we have an answer to the query.

In a similar manner, "Where was Road B two years ago?" would be formulated SQL-like as

select spatial
from *Versioned_road*, *Road_version*
where root = "Road B" **and version.temporal temporal_overlap last** ("year", 2, **now**, "instant")

This query extracts the spatial component of the version of Road B that existed two years ago. This is a direct answer to the original query, given as a spatial object.

Formally:

$$\{s \mid \exists r,v\ (Versioned\_Road(r) \wedge name(r) = \text{"}Road\ B\text{"} \wedge v \in versions(r) \wedge D(temporal(v), tc_1) \wedge spatial(v) = s)\}$$

A query "Did any land use changes occur in this drainage basin between Jan 1, 1980 and Dec 31, 1989?" might be interpreted as involving a change in the area extent of the land use areas, or a change in their classification attribute, or whether the particular area object disappeared or new areas were born. The following query finds versions of a land use area 12345 where the value of its the classification attribute changes. The select condition first specifies that we are interested in areas that lie inside "this drainage basin"; that area is referred to using a spatial constant "REF". A negated aspatial clause is then used to specify that the versions whose classification attribute values are different looked for.

select versioned, **version_1, version_2**

from Versioned_Landuse, Landuse_version, Landuse_version
where versioned.id = "12345" **and**
version_1.**spatial inside** "REF" **and**
version_2.**spatial inside** "REF" **and**
not (**version_1**.classification = **version_2**.classification)

In this query, the class *Versioned_Landuse*, and methods defined similarly as in the previous example are assumed. Spatial constant sc_1 represents the reference area REF. *I* is a spatial predicate for "inside". The query is formalised as:

$$\{v_1, v_2 \mid \exists r_1, v_1, v_2, s_1, s_2 \,(\; Versioned\_Landuse(r_1) \wedge id(r_1) = \text{"12345"} \wedge v_1 \in versions(r_1) \wedge v_2 \in versions(r_2) \wedge \neg(v_1 =_{id} v_2) \wedge spatial(v_1) = s_1 \wedge\ spatial(v_2) = s_2 \wedge I(sc_1, s_1) \wedge I(sc_1, s_2)\;)\}$$

10 Concluding Remarks

The purpose of specifying a formal semantics for SQL-like spatio-temporal queries is to aid in understanding the meaning of the queries better, and to outline a solution for their implementation. The semantics of spatio-temporal version queries is developed here using an object calculus. The alphabet of the calculus consists of concepts that can be easily implemented: classes, objects and methods. Given a set of database objects, the answer to a query is obtained by finding a proper valuation for query variables such that the query criteria are fulfilled.

A question then arises, what can we describe using the calculus. The calculus can be used to refer to objects from different classes, and methods to extract their values and other characteristics. Predicates can be used to make statements about relationships between objects Spatial predicates denote spatial conditions between object versions, and temporal predicates are used for evaluating temporal relationship between them. It is thus possible to describe the relationships between versions of objects in terms of their spatio-temporal characteristics, and determine the semantics of such a description with respect to what data exists in a database. What first-order predicate calculus cannot do is to state something about sets of objects as a whole. For that purpose, we should be able to quantify over predicates, not just individual elements of domains.

The formalisation of queries and their semantics provides a framework for research that allows the development of models of discrete change further, for example, to model complex mutations of objects.

Acknowledgements

This work has been carried out under supervision of Professor Tapani Sarjakoski, whose advice and support is gratefully acknowledged. The author is indebted to the anonymous reviewers for the comments that helped in shaping the contribution of the paper.

References

Abiteboul A, Hull R, Vianu V (1995) Foundations of databases. Addison-Wesley

Allen JF (1984) Towards a general theory of action and time. Artificial Intelligence 23(2):123-154

Armstrong M (1988) Temporality in spatial databases. *Proceedings GIS/LIS'88*, San Antonio, Texas, pp 880-889.

Cattell R, Barry DK, Berler M, Eastman, J, Jordan D, Russell C, Schadow O, Stanienda T, Velez F (2000) The Object Data Standard: ODMG 3.0. Morgan Kaufmann Publishers

Chomicki J (1994) Temporal query languages: a survey. In: Proceedings of the First International Conference Temporal Logic. Bonn, Germany, pp 506-534

Claramunt C, Theriault M (1995) Managing time in GIS: an event-oriented approach. In: Proceedings of the International Workshop on Temporal Databases. 17-18 September 1995, Zürich, Switzerland, pp 23-42

Clementini E, Di Felice P (1993) An object calculus for geographic databases. In: Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice, Indianapolis, IN, pp 302-308

Easterfield M, Newell R, Theriault D (1990) Version management in GIS – applications and techniques. In: First European Conference on Geographical Information Systems. Amsterdam, The Netherlands, pp 288-297

Egenhofer M (1994) Spatial SQL: a query and presentation language. IEEE Transactions on Knowledge and Data Engineering 6(1): 86-95

Egenhofer M, Franzosa R (1991) Point set topological relations. International Journal of Geographical Information Systems 5(2):161-174

El-Geresy B, Jones C (2000) Spatio-Temporal Models and Queries in GIS. In: Atkinson P, Martin D (eds) Innovations in GIS 7, Taylor and Francis, pp 27-39

Elmasri R, Navathe S (1989) Fundamentals of Database Systems. Menlo Park, CA, Benjamin/Cummings

Frank AU, Raper J, Cheylan, J-P (eds) (2000) Life and Motion of Socio-Economic Units. GISDATA Volume 8, London, Taylor & Francis.

Gallaire H, Minker J, Nicolas J-M. (1984) Logic and databases: a deductive approach. ACM Computing Surveys 16(2):153-185

Griffiths T, Fernandes A, Paton N, Mason K, Huang B, Worboys M (2001) Tripod: A Comprehensive Model for Spatial and Aspatial Historical Objects. In: 20th International Conference on Conceptual Modeling. Yokohama, Japan, pp 84-102

Halpern J, Shoham Y (1991) A propositional modal logic of time intervals. Journal of the ACM 38(4):935-962

Herring J (1987) TIGRIS: Topologically integrated geographical information system. Proceedings of AUTO-Carto 8. March 1987, Baltimore, MD, pp 282-291

Hornsby K, Egenhofer M (1997) Qualitative representation of change. In: Hirtle S, Frank A (eds) Spatial Information Theory: A Theoretical Basis for GIS, Proceedings of the International Conference COSIT '97. pp 15-33

Hornsby K, Egenhofer M (1998) Identity-Based Change Operations for Composite Objects. In: Proceedings of the 8th International Symposium On Spatial Data Handling. Vancouver, Canada, pp 202-213

Kemp Z, Thearle R (1992) Modeling relationships in spatial databases. In: Proceedings 5th International Symposium on Spatial Data Handling. Charleston SC, pp 313-322

Kemppainen H (2001) A data model and query language for versioned spatio-temporal objects. Submitted to *Geoinformatica* (under review)

Langran G (1989) A review of temporal database research and its use in GIS applications. International Journal of Geographical Information Systems 3(3): 215-232

Langran G (1992) Time in Geographic Information Systems. Taylor & Francis

Medak (1999) Lifestyles – an algebraic approach to change in identity. Spatio-Temporal Database Management, International Workshop STDBM'99. Edinburgh, Scotland pp 19-38

Peuquet D (1994) It's about time: a conceptual framework for the representation of temporal dynamics in geographic information systems. Annals of the Association of American Geographers 84(3):441-461

Peuquet D (2001) Making space for time: issues in space-time data representation. GeoInformatica 5(1):11-32

Snodgrass R (1992) Temporal Databases. International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning. Pisa, Italy, pp 22-64

Tryfona N (1998) Modeling Phenomena in Spatiotemporal Databases: Desiderata and Solutions. In: Database and Expert Systems Applications, 9th International Conference, DEXA '98. Vienna, Austria, pp 155-165

Wachowicz M, Healey R (1994). Towards temporality in GIS. In: Worboys MF (ed) Innovations in GIS 1. Taylor and Francis, London, pp 105-115

Worboys MF (1992) A Generic Model for Planar Geographic Objects. International Journal of Geographical Information Systems 6(5): 353-372

Worboys MF (1994a) A unified model of spatial and temporal information. The Computer Journal 37(1):26-34

Worboys MF (1994b) Object-oriented approaches to geo-referenced information. International Journal of Geographical Information Systems 8(4):385-399