Department of Informat cs

Multi-dimensional Time Support for Spatial Data Models

Bjørn Skjellaug and
Arne-Jørgen Berre

Research Report 253

ISBN 82-7368-173-4
ISSN 0806-3036

May 1997

Multi-dimensional Time Support for Spatial Data Models

Bjørn Skjellaug and Arne-Jørgen Berre*
Institutt for informatikk
Universitetet i Oslo
P.O.Box 1080 Blindern, 0316 Oslo, Norway

bjornsk@ifi.uio.no
http://www.ifi.uio.no/~bjornsk/

Research Report 253
ISBN 82-7368-173-4

May 1997

Abstract

Time has been considered from many research perspectives for applications of spatial information. However, this paper presents a generic multi-dimensional time support for spatial data models with an emphasis on geographic data. The generic time support, i.e., the temporal notions and concepts, is based on the same support applied in temporal databases and engineering databases, respectively.

The generic temporal notions and concepts to handle revisions, histories, and versioning of objects are defined, and applied to an example geographic data model (which is related to the standardizations within ISO/TC 211 and OpenGIS). For example, a database designer who wants to keep track of revisions of individual geographic objects can do so by assigning to these objects a single database built-in transaction time dimension. Furthermore, all kinds of objects may be defined to be temporally multi-dimensional, i.e., the database designer could, in principle, apply any number of time dimensions to the data types defined in the database schema. Hence, a value associated with different time dimensions can capture different temporal aspects and possible world semantics. The temporal model is illustrated by examples formulated using ODMG's Object Model and languages.

1 Introduction

Time and temporal data are recognized as important issues by the GIS community. Still, though Thrift in 1977 [20] considered time as an additional dimension to 2D and 3D geographic data, research on spatio-temporal databases was limited before the present decade [1], but over the past few years several workshops and conferences have addressed the issues of time and temporality for geographic data, e.g. [7, 8]. However, spatio-temporal database models (for GIS), e.g. [5, 13, 14, 21], usually lack a conceptual discussions where both the fundamental temporal principles are defined and where these principles are unambiguous translated and applied to a spatio-temporal data management model.

This paper investigates the temporal support from a data modeling and management perspective, and directly address the experiences of other fields, such as temporal databases and

*The authors are also with Department of Distributed Information Systems, SINTEF Telecom and Informatics, Oslo.

engineering databases. Many proposed spatio-temporal data models for GIS are defined and utilized in context of some GIS related applications, e.g. [14, 13, 22], but then the advantage of more generic temporal database notions and concepts seem to be lost. Our approach is solely based on a generic temporal support. Moreover, the fundamental time support for geographic data is uniformly defined for modeling temporality and managing change, including adding, deleting, and/or modifying historic, current and predictive spatial and aspatial data. Whereas Worboys' spatio-temporal model [23] unifies two spatial and two temporal dimensions (i.e., an event time and a database time) into a four dimensional generic model, our approach uniformly incorporates a multi-dimensional temporal support to both spatial and aspatial data.

The temporal model is at this stage not fully formalized—mathematically. The main concepts of temporality introduced, such as *lifespan* and *multi-dimensional timestamp*, generalize those of general purpose temporal data models and capture how facts relate to their trueness in time in various respects. The multi-dimensional temporal support, i.e., an object or a value may relate to several time dimensions, will, for example, mean that different time dimensions describe different temporal aspects or possible worlds semantics of the modeled reality. The multi-dimensional temporal perspective is an important property, here, introduced for geographic data. Temporal databases usually defines two time dimension denoting database times (transaction times) and times for the modeled reality (valid times). Time for geographic data has most often been considered to only be an extra dimension to the spatial domain, extending the model to denote the 3D or 4D domain. However, we show that time is more involved and in fact represents an n-dimensional structure, also making the model n-dimensional. Applications are then not isolated only to adopt one or two temporal perspectives of their data, but may assign to their data any number of times required.

This paper is organized as follows. Section 2 gives a brief introduction to concepts defined for temporal and engineering databases. Section 3 briefly presents the geographic information standardization of ISO/TC211 and clarifies how our work relates to that of ISO/TC211 by introducing for purposes of illustration a simplified feature data model. Then, Section 4 introduces the multi-dimensional temporal model and applies it to the feature data model. Section 5 summarizes the paper and points to future research issues.

2 Temporal and Engineering Databases

This section presents the basic notions of temporal support as defined by a vast set of both relational and object-oriented data models and systems [18, 19]. The time information may be given different semantics depending on the time dimensions supported. Usually, database models define one or two of the following time dimensions: the *valid-time* dimension defines when a fact (e.g. a person's address) was, is, or is believed to be valid (true) in reality. On the other hand the *transaction-time* dimension defines when a fact was or is stored/current in the database.

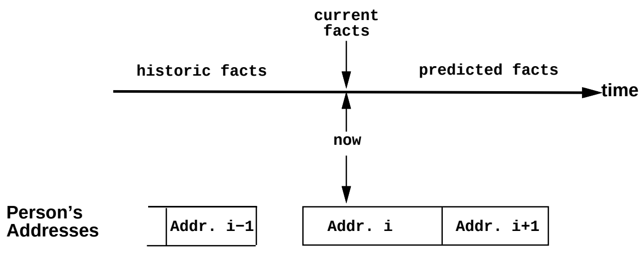

Figure 1: Facts and Times

Thus, an address object could then be regarded as defined over these two dimensions; Each point along the valid-time dimension denotes a domain value for a mapping of the object onto an actual address value, i.e., where the person lived at the specified time (the example of Figure 1); Whereas each point along the transaction-time dimension maps to an actual address current in the database at that time. These two dimensions provides an object with a "real world" history and a modification (revision) history, respectively. Temporal databases, and their temporal DBMS's, provide and utilize the temporal semantics of these dimensions. A user can access both historic and predictive facts, as well as the current facts, managed by the database. Typically, the ability to determine when a certain phenomenon occurred and how many times it independently occurred is fully maintained by the temporal model.

The temporal dimensions are defined in the following, where the domains for valid-time, D_v, and transaction-time, D_t, are, for reasons of simplicity, denoted by a total ordered set of discrete times:

$$\begin{aligned} D_v &= \{t_0, t_1, \ldots, t_i, \ldots, now, \ldots\} \cup \{\infty\}, \text{ where} \\ & \quad \forall t', t'' \in D_v \setminus \{\infty\} \ : \ t' < t'' < \infty \ \vee \ t'' < t' < \infty \ \vee \ t' = t'' < \infty \qquad (1) \\ D_t &= \{t_0, t_1, \ldots, t_i, \ldots, now\}, \text{ where} \\ & \quad \forall t', t'' \in D_t \setminus \{now\} \ : \ t' < t'' < now \ \vee \ t'' < t' < now \vee \ t' = t'' < now \qquad (2) \end{aligned}$$

The valid-time dimension has an upper bound set to ∞, but has also times that goes beyond the current time *now*. The transaction time has *now* as its upper bound. The interpretation of *now*, for both dimensions, is that it is a variable, but at every point in time it denotes the present time. For time values given as time instants, t_k, and time intervals, $I_{[t_k, t_l]}$, with domain over a time dimension, D_x, the following constraints apply, respectively:

$$\begin{aligned} t_k &= t_0 + k \text{ and } t_k = t_{k-1} + 1, \ k \geq 1 \qquad (3) \\ I_{[t_k, t_l]} &= \{t_i \,|\, t_i \in D_x \ \wedge \ t_k \leq t_i \leq t_l \ \wedge \ t_0 \leq t_i\} \qquad (4) \end{aligned}$$

In temporal databases time values are denoting timestamps. Time-instants may be used to denote events without durations and time intervals typically denote state durations. A single timestamp associated with an object (e.g. a record or tuple) defines when this object either exists in the database (transaction-time) or in the reality (valid-time).

The notions of an object's lifespan defines when an object was created, and, eventually, when an object ceases to exist. A lifespan also constrains the timestamps of attribute values associated with an object, i.e. each timestamp denotes a subset of times of the times defined by the lifespan.

In Temporal Relational Databases (TRDB) time is used for both schema versioning, and keeping track of changes to tuples and attributes [19]. A relation scheme definition $R = (A_1, A_2, ..., A_n)$ is an ordered set of attribute names, A_i's. This scheme could be extended with built-in valid-time and transaction-time timestamping, termed bi-temporal timestamps, associated on either the tuple or attribute level, and as defined by the following bi-temporal relation schemes, respectively:

$$\begin{aligned} R_1 &= (A_1, ..., A_n, V \times T) \qquad (5) \\ R_2 &= (\{(A_1, V \times T)\}, ..., \{(A_n, V \times T)\}) \qquad (6) \end{aligned}$$

where in both (5) and (6) the A_i's are user-defined attributes of the relation (including key attributes), $V = [T_{v_s}, T_{v_e}]$ and $T = [T_{t_s}, T_{t_e}]$ are interval timestamps with the T_{x_s} and T_{x_e} denoting start and end times , respectively. Subscripts $_v$ and $_t$ stands for valid and transaction timestamps, respectively. Timestamps are defined over the domains given by (1) and (2) above, and they are constrained by the definitions of (3) and (4), also above.

Temporal object-oriented databases (TODB) also support valid and transaction times, schema versioning and so forth [18]. In addition, a TODB provides the ability both to timestamp objects as well as their individual attributes. An other strength that a TODB easily provides over a TRDB is the notions of object versioning [18]. Object versioning support different and parallel versions of conceptually the same object. A version is a member of the object that it belongs to, and different versions could both define (simultaneous) alternatives and how the object evolves from one version to the next. (Versions can, in some sense, be regarded as different views of an object, both in time and by "value".) The set of versions of an object, typically, constitutes the mathematical concept of a directed acyclic graph (DAG), where each node represents a version. We return to TODB characteristics in Section 4, where Object Data Management Group's Object Model, ODMG–93 [4], is used to define temporality.

3 Modeling Concepts for Geographic Data

As a basis for a geographic data model, we first relate our modeling approach to the ongoing work within ISO/TC211 [12], for the modeling approach, and OpenGIS [16]. Then we show how such models can be mapped onto an implementation model, in this case we use ODMG–93 [4].

3.1 A modeling perspective

The ISO/TC211 general modeling approach is to follow the structure of the viewpoints from the ISO RM ODP(Reference Model on Open Distributed Processing) [9]. This approach will describe a geographic information system through five different viewpoints. The enterprise viewpoint describes the purpose, scope, context and policies of the enterprise/business related to the specified system/service. The information viewpoint is the viewpoints that describes the data models, and structures the information of an enterprise. The computational viewpoint is concerned with the interaction patterns between the components (services) of the system, and reflects the API's that is offered to support the various services defined, e.g. a database interface API. The engineering viewpoint is concerned with the design of distribution-oriented aspects, i.e., the infrastructure required to support distribution. The technology viewpoint is concerned with the mapping to an underlying infrastructure. The technology viewpoint level is where OpenGIS is focusing their implementation specification, which has addressed technology specifications for Microsoft COM/OLE, OMG CORBA, Internet and ODBC.

In the context of this paper we are only concerned with two of these viewpoints, namely the information and technology viewpoints. That is, the former is focusing on the application schemas and meta-schemas for geographic information (illustrated in Figure 2, and the latter is denoted by a mapping of the applications schemas to a representation in the technology viewpoint. In our work we are mapping to a technology implementation based on the ODMG

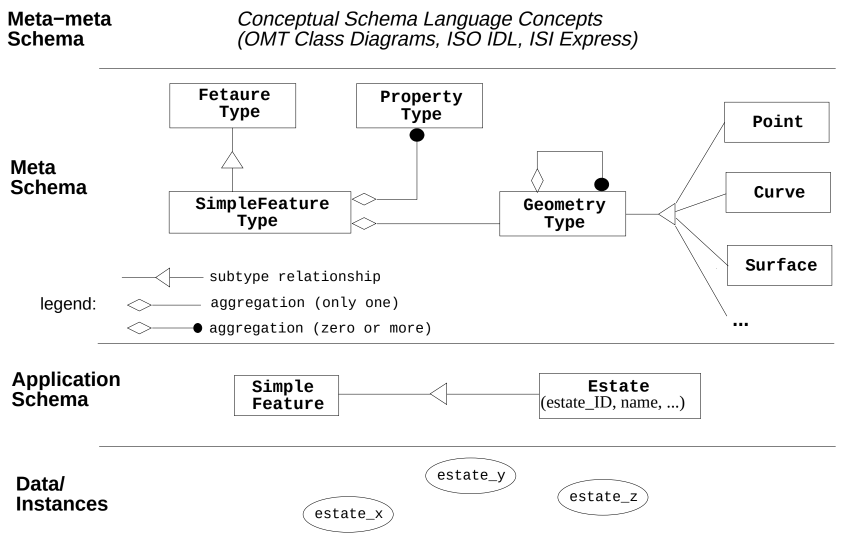

Figure 2: A Simplified Feature Model in the ISO CSMF Layered Architecture

Object Oriented Database Model [4]. Our implementation is based on an object-oriented database that includes a basis support for time-based versioning on the attribute level. Within the ISO/TC211 information viewpoint the proposed modeling approach is to use the concepts in the ISO work on Conceptual Schema Modeling Facility [10], and apply this to the GIS domain. This approach distinguishes between four modeling levels, where each level defines the concepts to be used for definitions on the level below, as illustrated in Figure 2. The topmost meta-meta schema defines the concepts in the chosen conceptual schema language, and is in ISO/TC211 selected to be a model with OMT class diagrams as a graphical syntax, and ISO EXPRESS and ISO IDL as lexical syntax. The meta schema defines generic geographic information concepts and includes spatial and temporal aspects. The application schema defines the concepts to be used in a particular context, and can reuse existing definitions which are maintained in a Feature catalog. The data/instance level contains the actual data.

The meta schema we use is a combination of the generic feature model from ISO/TC211 and the Open Geodata Model from OpenGIS. Figure 2, however, only gives a simplified and brief view of these models. For the data access API we are using the basis services from the ODMG model [4].

In this paper we focus on demonstrating the time support and shows for simplicity only a static, early binding specification, while it is also possible to use a dynamic, late binding access mechanism. The latter would be most suited for the use by generic map editors and browsers. We show next the specifications directly in ODL, the ODMG Object Definition Language, without showing the mapping process from OMT and EXPRESS to ODL.

3.2 A Feature Data Type—FDT

Below we show how the application schema of the information viewpoint, presented in Figure 2, is mapped to a corresponding database schema of the technology viewpoint expressed in ODMG ODL [4]. Note, however, that this mapping is only an example and used for illustrative purposes only.

A data type for geographic data, here called a feature data type—FDT, is an abstract data type—ADT. We present a translation of a FDT from the ODP information viewpoint, as presented above, into the technology viewpoint which in this case is based ODL. (Note that the example shown in Figure 3 is only illustrating one such translation, and is not intended to define what a `SimpleFeature` should look like in a real setting.)

A FDT, e.g. a `SimpleFeature` as shown in Figure 3, captures both spatial and aspatial information of a phenomenon. A FDT could encapsulate specific representational models, such as raster and vector models, supported by an underlying system. Because a FDT is an ADT it provides a means to specify subtype relationships and the characteristics of an interface, such as properties (attributes and relationships), and operations of objects.

The FDT `SimpleFeature` is defined in the technology viewpoint by the ODL specification of Figure 3, but it maintains the modeling structure of the information viewpoint and application schema presented in Figure 2. In Figure 3 the keyword `interface` indicates that

```
interface SimpleFeature :: Feature {
  extent simplefeatures;
  key name;
  attribute string name;
  attribute Geometry spatial;
  relationship Set<Property> properties
               inverse Property::is_property_of;
  centroid (in Geometry, out Point)
           raises (centroid_undefined)};
```

Figure 3: FDT example

we define a FDT `SimpleFeature`, that is a subtype of the FDT `Feature`. This type has an `extent`, meaning that this type has object instances that are managed by its `extent` set named `simplefeatures`. The attribute `name` is defined given the keyword `attribute` and its data type `string`. A key for the type is defined by identifying it by the keyword `key`, and one or several attributes may represent a key attribute. A relationship is defined by denoting the other data type it stands in relationship with (i.e., `Property`), its cardinality (i.e., `Set<...>`), its name (i.e., `properties`), and by denoting the corresponding (i.e., `inverse`) definition of the other data type (i.e., `Property::is_property_of`). An operation signature is defined by a name, here `centroid`, a list of input parameter types, a list of output parameter types, and some exception handling. The parameter lists and exception(s) are indicated by the keywords `in`, `out`, and `raises`, respectively.

4 Extending Spatial Data with Time

This section defines the role and the notions of time suited for incorporating temporal support for spatial data models, with emphasis on the model presented in Section 3. This section also represents the mapping of the information viewpoint of a GIS to the technology viewpoint of the same system. Even though ISO/TC211 defines a temporal model, termed a temporal subschema, in the information viewpoint (a draft standard at this stage) [11], this model does not explicitly define the different time dimensions and temporal aspects a model should associate with feature types and their attributes. In that respect our temporal model, described next, is a concrete proposal for a multi-dimensional temporal support, which we exemplify by ODMG ODL in the technology viewpoint.

4.1 A ODMG based Temporal Geographic Data Model

An example definition of a temporal FDT `Estate` and related definitions are given by the ODL specification of Figure 4. Two temporal notions are introduced to capture the temporal nature of both an object as a whole and its individual properties, i.e., for the management of the temporal extent of an object. These notions are introduced by the type `T_Object` and the qualifier `temporal()` (their respective definitions and ODL specifications are given in Section 4.2.1 and Section 4.2.2). Figure 4 defines at the left a temporal FDT, `T_Feature`,

```
interface T_Feature
          ::T_Object{};


interface T_SimpleFeature
          ::T_Feature,
            SimpleFeature{};
```

```
interface Estate::T_SimpleFeature {
  temporal(extent) estates;
  key estate_ID;
  attribute estate_no estate_ID;
  attribute temporal(string) name;
  attribute temporal(Geometry) spatial
  relationship temporal(Set<Property>)
          properties inverse
          Property::is_property_of;
  centroid(in Geometry, in TIME,
           out Point)
           raises (centroid_undefined)};
```

Figure 4: Estate FDT example

that is a subtype of `T_Object`. `T_SimpleFeature` is, in turn, a subtype of `T_Feature` and `SimpleFeature`. (Recall the sketch of a model presented in Section 3.) These two novel temporal types do not include any definitions of their own, i.e., their `{}` clause is empty, so the definitions inherited through their respective supertypes entirely constitute the specification of these types.

The right hand side of Figure 4 defines the temporal FDT `Estate`. Note that several of the characteristics (attributes, relationships and operations) shown for this type are, in fact, redefinitions and refinements of the characteristics of its static supertype, the FDT `SimpleFeature`, defined in Figure 3. For example, the `name` attribute is refined, i.e., qualified by `temporal()`, simply meaning that this type allows the name attribute of an object to change over time, the different names are maintained by the type, and at each specified time an individual name could be accessed. Similarly, the `estates, spatial` and `properties` characteristics also are refined to be time-varying. However, the `estate_ID` is defined to be static, i.e., it is time-invariant. The key `name` of its supertype `SimpleFeature` is here redefined to be a key `estate_ID`.

The operation `centroid` is now a temporal function in the sense that its domain is refined and reflects the time dependency of an estate's spatial information. The type `TIME` (of the function domain) is a data type with a value domain composed of timestamps defined over the time dimensions (1) and (2), their related constraints (3) and (4), as defined in Section 2.

An estate object could be defined by different (types of) geometries, both simultaneously and at different times. This simply could be done by redefining the `spatial` attribute of the FDT `Estate` of Figure 4 as illustrated by Figure 5. That is, the `spatial` attribute is now a

```
attribute struct<temporal(Raster) image,
                 temporal(Region) area>  spatial;
```

Figure 5: Multiple geometries

record (struct in ODL terms) with its fields defined to be time-variant, meaning that different raster and region values (e.g. rasters and regions are specialized `Geometry` values) capture the estate's time-varying spatial information. For example, the `image` field of the `spatial` record could represent different aerial photos of the estate that document the cultivation of the estate and how this cultivation changes with seasons and over years.

4.2 Multi-dimensional Time support

The above schematically showed how time was incorporated into a geographic data model, and that it did not represent a fundamental different issue and approach compared to that of temporal non-geographic data management briefly presented in Section 2. We argue that the orthogonality of time, to both spatial and aspatial data, makes time uniform to all geographic data. That is, geographic data does not induce fundamental different notions of time and change in this respect, i.e., when defining means for modeling and managing time-varying data by a database. We postulate that the time dimension is "independent" of (i.e., orthogonal to) both spatial dimensions and property data domains. In the following we show that time could easily exhibit a composite structure, i.e., the temporal support of an application could be defined by means of combining the built-in time dimensions and, thereby, utilize the multi-temporal semantics of a model.

The spatio-temporal extended SQL by Böhlen et al. [3], termed STSQL, incorporates multiple spatial and temporal dimension attributes[1] defined for a relation. A relation with both spatial and temporal dimension attributes has, logically speaking, a multi-dimensional *space-timestamp* associated with each object, i.e., tuple. A space-timestamp denotes aspects concerning *where* and *when* of an object. Based on the the notions of *valid* and *transaction timestamps* as presented in Section 2, STSQL generalizes these notions to define *valid* and *transaction spacestamps*. Thus, the spatial aspect of a spacestamp denotes either where in reality an object is true or from where it was registered, respectively. The former is a valid spacestamp the latter a transaction spacestamp. In STSQL also multiple valid-times and transactions-times may be associated with objects. That is, multiple valid-time dimensions

[1]Each dimension attribute captures values of only one kind, i.e., a 1D, 2D or 3D region data type for space and intervals for time.

denote different temporal aspects of an object or possible worlds of an object. For example, say that a forest management application is used for harvest planning and scheduling of forest stands (i.e., well-defined and surveyed regions of forests) [15]. One valid timestamp may denote the aspect of when the data stored about a particular plan of a stand is true in reality, whereas another valid timestamp for the same object may denote the aspect of when the harvest period itself is scheduled. If we have possible world semantics, a third valid timestamp could denote an other period when the harvest is scheduled, because, for example, this time period is estimated based on an alternative set of ecological and economical parameters used by the application.

Section 2 introduced the concepts of valid time and transaction time. In principle there could be several of these times, and even several orthogonal lifespans, associated with objects. In the above we did not discuss in detail what the type `T_Object` is, nor what the qualifier `temporal()` hides. These are, in turn, defined next.

4.2.1 Object temporality

The `T_Object` type introduces the concept of a *lifespan*. (Note that the characteristics defined for `T_Object` are those inherited by `T_Feature` and its subtypes defined in Section 4.1.) The novel approach of this model is that several lifespans of an object are defined. More formally, a **lifespan** L is a subset of the set T [6], i.e., $L = \{t \mid t \in T\}$, where T is a time dimension. In practice, and, since, the time dimensions of (1) or (2) of Section 2 are isomorphic to (a subset of) the natural numbers, we define the lifespans in terms of ("sets" of) interval timestamps, and constrained by (3) and (4) of Section 2. Timestamps are then given by the ODL specifications in Figure 6, i.e., three time-related stamps, namely `VT`, `TT` and `VID`, denoting *valid time-, transaction time-* and *version-stamps*, respectively. Note that valid and transaction

```
interface VT {                          interface TT {
  attribute Valid_dim  v_start;           attribute Trans_dim  t_start;
  attribute Valid_dim  v_end};            attribute Trans_dim  t_end};

interface VID {                         interface T_Object {
  attribute Trans_dim version_time;       attribute List<VT>  valid_lifespan;
  attribute node      version_ID;}        attribute TT        trans_lifespan;
                                          attribute List<VID> version_lifespan;}
```

Figure 6: Timestamps and `T_Object` specification

timestamps are time intervals, both denoting start- and end-times. A version-stamp denotes a transaction time and a version identification. A version-stamp also denotes a node in a structure. In our simplistic approach this structure is linear, i.e., the `List<VID>` specification for `T_Object` in Figure 6. Basically what it does is to denote a database transaction time, termed `version_time`, from which data already stored about an object is to be considered as a separate version identified by a `version_ID` value. That is, data as best known at that time is part of this version, data that is added to the database after this time is considered to be part of the next version of the object, i.e., a `version_time` value, here, simply denotes a database *check point*.

A versioning system that involves a more complex model, such as a directed acyclic graph (DAG), could be introduced either on the object level or on the attribute level, as defined by Wuu and Dayal [24] and by Sciore [17], respectively. A `version_lifespan` could, then, be defined in terms of a data type, `Graph<T>`, that reflects the abstract DAG structure of an object. For example, nodes in a DAG with `version_ID` values 3.2.1 and 3.2.2, respectively, denote nodes as different alternatives derived from the node version denoted by the `version_ID` value 3.2. Note that the transaction time instants associated with the nodes of a DAG instance would then represent a branching time of `version_times`, i.e. imposing a partial order. However, such a DAG would require an other structure than the one defined by `T_Object` above. We return to this issue when we discuss an attribute versioning approach below, where a DAG structure is applied, an approach that in several respects is similar to the approach defined by Sciore [17]. Figure 7 illustrates the notions of an object's lifespans.

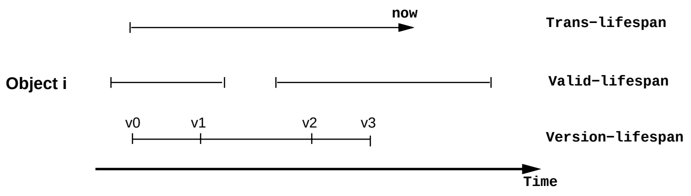

Figure 7: Object Lifespan

"Object i" has three lifespans, one for when it is valid in the reality, one when it is current in the database, and one lifespan for versions. The notion of a valid-lifespan allows an object in reality to die and thereafter be reincarnated at some later time, and even to have a predicted "life". A transaction-time lifespan allows only a contiguous database lifespan of an object, and its end time is always constrained by the time *now*. The version lifespan denotes, besides the different versions of the object, the times when each version becomes current in the database, and, hence, each version is restricted by its corresponding transaction lifespan. Note that after version `v3`, in this example, no version exists.

4.2.2 Attribute temporality

The language construct `temporal()` qualifies attributes of a FDT as temporal, i.e., time-varying, and also manages their time-varying values. This qualifier is similar to that of T_Chimera [2], but our approach generalizes that of T_Chimera to support an arbitrary number of time dimensions, and not only valid-time.

The qualifier `temporal()` should also be regarded as a, possibly, predefined ODMG ADT. Then users could define subtypes of the type `temporal()` with specific requirements. In the estate example we have `attribute temporal(string) name`, where the `name` attribute is a time-varying attribute. An interpretation of this attribute gives a finite set of timestamp-string pairs, such as $\{\langle\tau_1, s_1\rangle, \langle\tau_2, s_2\rangle, ..., \langle\tau_n, s_n\rangle\}$, where $\tau_1, ..., \tau_n$ are, possibly, multi-dimensional timestamps, and $s_1, ..., s_n$ are legal string values.

Formally, each temporal attribute is defined as a pair $(t, f(t))$, where f defines a partial function, $t \in D_x$, and $f(t)$ is the value at time t. To translate this into our estate example we have the `name` attribute interpreted as the set $\{\langle t_1, \texttt{name}(t_1)\rangle, ..., \langle t_n, \texttt{name}(t_n)\rangle\}$, where, here, each t_i denotes a timestamp, and $\texttt{name}(t_i)$ is the function value at time(s) denoted by t_i. Formally, an n-dimensional timestamp t is of a type defined from a cross-product of timestamp types, $T_{x_1} \times T_{x_2} \times ... \times T_{x_n}$, $0 \leq n$, and where any of the T_{x_i}'s could represent valid and transactions timestamps, or some other application or system induced timestamp. The specification in Figure 8 of the temporal type `temporal(T)` includes (only) three time dimensions through a combination of the data types `TIME` and `NODE`. An attribute a of some type `T` qualified by `temporal()` has its time-varying information managed by `o_graph`, i.e., a graph instance, where each node in the graph is denoted by a version ID, `vid`, and a list of records of object values and associated bi-temporal timestamps, termed `o_history`. The operation `history` retrieves the value of a at a specified time for a specified version. Note that an output value, of type `Set<T>`, from `history` could include several values, because, for example, the times denoted by the `TIME` input value could span several or all values of type `T`.

The versioning approach showed here is similar to that of Sciore's approach [17], in that only those attributes that are believed to be time-variant and versioned are qualified by the

```
interface TIME {                     interface temporal(T)
  attribute VT   vt;                   attribute Graph<NODE(T)> o_graph;
  attribute TT   tt};                  history (in      o_history,
                                                in      TIME,
interface NODE(T) {                             in      VID,
  attribute List<struct<                        out     Set<T>)
            T    object,                        raises (object_not_defined_at_
            TIME time>> o_history;                      the_specified_time)};
  attribute VID vid};
```

Figure 8: A `temporal(T)` specification

type `temporal()`. However, these two approaches differs in one important respect, i.e., our approach supports an explicit graph structure, where the type `Graph<>` manages the different versions as nodes of a graph instance. Each node, in turn, manages the different valid and transaction timestamped attributes values. Figure 9 illustrates the notions of valid-time

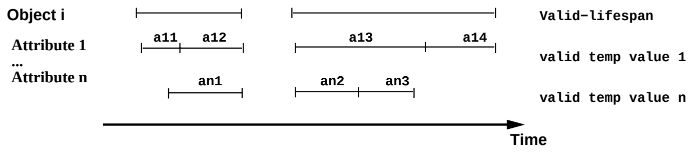

Figure 9: Attribute Lifespan

attributes, showing that each valid-time attribute denotes a sequence of values, where any two values, a_{ij}, of a sequence are both associated with individual, but adjacent or disjoint, time intervals of the corresponding time dimension. Each valid-time attribute is temporally constrained by the valid-time lifespan of its object. That is, each period of validness of an attribute value is a subset of the lifespan of its corresponding object—always. Such constraints will also exist between an object transaction lifespan and its transaction timestamped attributes.

4.3 Object Query Language—OQL

The above described the structure of a spatio-temporal data model. This section presents an example of how queries can be formulated over this model. The language used is still that of ODMG, termed OQL.

Figure 10 demonstrates the formulation of two queries in OQL, both trying to extract the names of neighbor estates to the Royal Palace. The query example to the left is a standard non-temporal OQL issued on a snapshot database (note that both queries invoke the operation `adjacent`, e.g. an operation that `Estate` inherited from `Feature`). Now, envision that the

```
SELECT e.name                            SELECT e.name
FROM   e IN estates                      FROM   e IN estates as_of [1960-1990]
WHERE  e.adjacent ('Royal Palace');      WHERE  e.adjacent ('Royal Palace');
```

Figure 10: OQL examples

previously defined generic temporal data types are all built-in ODMG types, and that we can utilize this temporal support by the OQL. The query example to the right introduces a new syntax construct, i.e., an `as_of` clause, used to denote the valid time period to evaluate the query for. The evaluation of this query would involve the following tasks; Determine all estates from the extent set `estates` that were valid sometime during the specified period; For each of these estates, in turn, check if the estate geometry is adjacent to the `Royal Palace`; Then, for all qualified estates, i.e., the adjacency test yields *true* for at least one time point within the specified period, return names and associated valid-times. Note that several names of an estate may be included in the result if the estate has changed its name during the specified period.

The time scope or reference time is in this query assumed to be for the valid-time dimension, and is denoted by the `as_of` clause. The time scope for transaction time, which is not explicitly specified in this example, is therefore set to the default time *now* (i.e., as best known at present). If we wanted an other referenced transaction time we had to denote this time by an `as_best_known` clause, i.e., what was current in the database at specified previous times. Similarly, the default for versioning are those versions known now, whereas a specific referenced version time should be set by an `as_version` clause.

5 Conclusion

The temporal notions and concepts defined for our model are based on developments within temporal and engineering databases. However, in several respects our model generalizes and redefines the temporal support defined by others, and, thereby, adds novelty to our approach; First, the concept of lifespan by Clifford & Croker [6] is generalized to include multiple lifespans of an object; Second, Sciore's object versioning model [17] applied on versioned attributes is redefined to explicitly denote the structure of a DAG; Third, the valid-time varying nature of an object's attributes, as seen by the $T_$Chimera model by Bertino et al. [2], is generalized to a multi-dimensional structure that is combined with our approach of object versioning on the attribute level; Fourth, multi-dimensional time are applied to both objects and attributes, and not only to objects (i.e., tuples) as by STSQL [3]. Thus, a database designer has, by this approach, the flexibility of assigning whatever temporal built-in or user-defined dimensions required.

The temporal model defines how these notions and concepts are incorporated by two main temporal concepts, termed a qualifier `temporal(T)` and a type `T_Object`, and both manage multi-dimensional time-varying data. We used ODMG's ADT, `interface`, to define these concepts, and, thereby, showed how temporality could be assigned to arbitrary feature types of a geographic data model. Therefore, also a main result of this paper is that we have emphasized and exemplified that general purpose temporal notions and concepts can uniformly be applied to geographic data.

We are implementing our model in an object-oriented database system supporting the ODMG model. This database system has embedded versioning-support based on both named configurations and time. The base time-versioning is on the attribute level. Compared with the work on the temporal subschema in ISO/TC211, our approach can relate to the temporal reference systems that is being defined there, but we in addition address how these concepts can be realized based on general technology for time support. This realization defines both a temporal ODL, with a fundamental set of temporal constraints, and the necessary extensions to the query and modification languages for the manipulation of temporal objects. This includes syntactic extensions, evaluation semantics, and a temporal object algebra. The first author also works on STSQL [3], and will coordinate the object-oriented model presented in this paper with the results of STSQL.

Acknowledgment We like to thank Gunnar Misund and David Skogan for most valuable comments and suggestions on an earlier version of this paper. This research was supported in part by the Norwegian Research Council through grant MOI.31297, the SINTEF projects MultiBed and GISIT, and the ESPRIT project 22.084, DISGIS, founded by the European Commission.

References

[1] K.K. Al-Taha, R. T. Snodgrass, and M. D. Soo. Bibliography on spatiotemporal databases. *International Journal of Geographical Information Systems*, 8(1):95–103, January-February 1994.

[2] E. Bertino, E. Ferrari, and G. Guerrini. A Formal Temporal Object-Oriented Data Model. In *Proceedings of EDBT*, volume 1057 of *Lecture Notes in Computer Science*, pages 342–356. Spinger-Verlag, 1996.

[3] M. Böhlen, C. S. Jensen, and B. Skjellaug. Spatio-Temporal DataBase Support for Legacy Applications. Technical report, Technical report(TR-20), TimeCenter, July 1997.

[4] R. Cattell, editor. *The Object Database Standard: ODMG-93.* Morgan Kaufmann, San Mateo, 1994.

[5] T. S. Cheng and S. K. Gadia. A seamless object-oriented model for spatio-temporal databases. Technical Report TR-92-41, Computer Science Department, Iowa State University, 1992.

[6] J. Clifford and A. Croker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, Los Angeles, CA, February 1987. IEEE Computer Society Press.

[7] A. U. Frank, I. Campari, and U. Formentini, editors. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[8] A. U. Frank and W. Kuhn, editors. *Spatial information theory : a theoretical basis for GIS – COSIT'95*, volume 988 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[9] ISO RM ODP. *ISO/IEC 10746 - Open Distributed Processing*, 1995.

[10] ISO/SC21/WG3/N2039. *CSMF - Conceptual Modeling Facilities (CSMF), CD for ISO/IEC 14481*, 1996.

[11] ISO/TC211. *ISO 15047-8 Temporal Subschema - Working Draft Version 1.0*, May 1996. `http://-www.statkart.no/isotc21`.

[12] ISO/TC211/WG1 N040.01. *ISO 15056-1 - Reference Model*, January 1996. `http://www.-statkart.no/isotc211`.

[13] G. Langran. States, Events, and Evidence: The Principle Entities of a Temporal GIS. In *GIS/LIS '92*, pages 417–425, 1992.

[14] G. Langran. *Time in Geographic Information Systems.* Taylor & Francis Ltd., 1992.

[15] G. Misund, B.S. Johansen, G. Hasle, and J. Haukland. Integration of geographical information tecnology and constraint reasoning — a promissing approach to forest management. Technical Report STF33A 95009, SINTEF Applied Mathematics, Oslo, Norway, June 1995.

[16] Open GIS Consortium. *The OpenGIS Guide - and abstract specification*, January 1996. `http://-www.opengis.org`.

[17] E. Sciore. Versioning and Configuration Management in an Object-Oriented Data Model. *The VLDB Journal*, 3(1):77–106, 1994.

[18] B. Skjellaug. Temporal Data: Time and Object Databases. Research Report 245, Department of Informatics, University of Oslo, April 1997. ISBN 82-7368-160-2.

[19] B. Skjellaug. Temporal Data: Time and Relational Databases. Research Report 246, Department of Informatics, University of Oslo, April 1997. ISBN 82-7368-161-0.

[20] N. Thrift. *An Introduction to Time Geography*, volume 13 of *Concepts and Techniques of Modern Geography.* Geo-Abstracts, University of East Anglia, Norwich, England, 1977.

[21] M. F. Worboys. A Model for Spatio-temporal Information. In *Proceedings of the International Symposium on Spatial Data Handling*, pages 602–611, 1992.

[22] M. F. Worboys. Object-oriented Models of Spatio-temporal Information. In *GIS/LIS '92*, pages 825–834, 1992.

[23] Michael F. Worboys. A unified model for spatial and temporal information. *The Computer Journal*, 37(1):27–34, 1994.

[24] G. T. J. Wuu and U. Dayal. *A Uniform Model for Temporal and Versioned Object-oriented Databases*, chapter 10, pages 230–247. Benjamin/Cummings, 1993.