Temporal and Spatio-Temporal Data Models and Their Expressive Power*

Martin Erwig & Markus Schneider & Ralf Hartmut Güting
FernUniversität Hagen, Praktische Informatik IV
D-58084 Hagen, Germany
[erwig, markus.schneider, gueting]@fernuni-hagen.de

Abstract: Currently, there are strong efforts to integrate spatial and temporal database technology into *spatio-temporal* database systems. This paper views the topic from a more fundamental perspective and makes several contributions. First, it reviews existing temporal and spatial data models and presents a completely new approach to temporal data modeling based on the very general notion of *temporal objects*. The definition of temporal objects is centered around the observation that anything that changes over time can be expressed as a function over time. For the modeling of spatial objects the well known concept of *spatial data types* is employed. As specific subclasses, *linear* temporal and spatial objects are identified. Second, the paper proposes the database embedding of temporal objects by means of the *abstract data type* (*ADT*) approach to the integration of complex objects into databases. In spatial database technology, spatial data types have already been successfully integrated as ADTs for attributes in relational schemas for a while now. Furthermore, we make statements about the expressive power of different temporal and spatial database embeddings. Third, we consider the combination of temporal and spatial objects into *spatio-temporal objects* in (relational) databases. We explain various alternatives for spatio-temporal data models and databases and compare their expressive power. Spatio-temporal objects turn out to be specific instances of temporal objects.

1 Introduction

In the past, in spite of many similarities, research in spatial and temporal data models and databases has largely developed independently. Spatial database research [Gü94] has focused on modeling, querying, and integrating geometric and topological information in databases. Temporal database research [TCG+93] has concentrated on modeling, querying, and recording the temporal evolution of facts under different notions of time (valid time, transaction time) and thus on extending the knowledge stored in databases about the current and past states of the real world. Currently, there are increasing integration efforts striving for a combination of space and time in "spatio-temporal data models and databases". We have already discussed evolving problems and have presented a first approach for a data model in [EGSV97]. A few other papers already exist that deal with the integrated modeling of space and time, for instance, [Wo94,YC93], but they do not address the embedding in databases.

This paper now views the topic from a more fundamental perspective and makes several contributions. First, we present a very general model of *temporal objects* whose definition is based on the observation that anything that changes over time can be expressed as a function over time. Hence, a (simplified) definition of a temporal object is that of a function mapping time to a certain codomain under some constraints. For the modeling of spatial objects we employ the well known concept of *spatial data types* (e.g. [SV89, SH91, GS95, Sc97]) like those for points, lines, and regions. These have been identified as appropriate and efficient abstractions for modeling the geometric structure of spatial phenomena as well as their relationships, properties, and operations. There are also powerful logical approaches to the modeling of geometry and topology [CCR93, PGB94]. Nevertheless, we base our definition of spatial objects on the point set approach and on point set topology [Ga64]. A spatial object is assumed to be represented by a generally infinite point set with certain properties from which different structures like the boundary or the interior can be identified. There are mainly two reasons for this way of modeling: spatial objects modeled by the point set approach are efficiently implementable and can be easily embedded in a (relational) database. This leads us to the specific subclasses of *linear temporal and spatial objects*. In the case of linear temporal objects the temporal evolution of an entity is described by some kind of "linear" transition from one state to the next; in the

* This research was partially supported by the CHOROCHRONOS project, funded by the EU under the Training and Mobility of Researchers Programme, Contract No. ERB FMRX-CT96-0056.

case of linear spatial objects the linearity is given through polygonal approximations. Moreover, the vast majority of optimization methods and indexing techniques builds upon such linear representations. Besides, this approach fits very nicely with the functional model for our temporal objects.

Second, we propose the database embedding of temporal objects in the spirit of the *abstract data type* (*ADT*) approach applied to the integration of complex objects into databases [SRG83, St86]. So far, temporal databases have been essentially based on atomic standard data types extended by an explicitly or implicitly given type for time. Tuples are associated with time stamps; each tuple describes the validity and the features of a fact or object. We show that temporal databases based on the ADT approach are more powerful than current ones and that a simple extension compensates this difference for a particular class of temporal objects. In spatial database technology, for quite some time, spatial data types have been integrated as ADTs for attributes in relational schemas (e.g. [Gü88, SH91, Sc97]). We show that spatial databases supporting the ADT approach have the same expressive power as those decomposing spatial objects into a set of tuples in flat relations [BS77, Ro87].

Third, the variety of temporal and spatial data models offers a broad spectrum of integration options for temporal and spatial objects into *spatio-temporal objects* in (relational) databases. We discuss this design space and explain different alternatives for spatio-temporal data models and databases. Spatio-temporal objects turn out to be specific instances of temporal objects. In this paper we will ignore line objects and only deal with the temporal behavior of points and regions which leads to *moving points* and *moving regions* (this is motivated in [EGSV97]). We compare the expressive power of the different temporal, spatial, and spatio-temporal data models. For spatio-temporal data models we establish an expressive power lattice. An interesting visualization of spatio-temporal objects for the linear case is a three-dimensional view in the form of *3D-polylines* for moving points and in the form of *polyhedra* for moving regions [EGSV97], respectively. We show that, surprisingly, the polyhedra model is not comparable to any model of our lattice.

The rest of the paper is structured as follows: Sections 2 and 3 define a model for spatial and temporal objects, respectively, and describe an appropriate representation of these objects in databases. Besides, they present statements about the expressive power of the respective temporal and spatial modeling alternatives. Section 4 deals with the design space obtained when combining spatial and temporal objects. Various spatio-temporal data models are explained, and a number of relationships between the different models are shown. Section 5 concludes the paper.

2 Spatial Objects and Spatial Databases

2.1 A Model of Spatial Objects

Spatio-temporal objects describe geometric changes of spatial objects over time. We first deal with the spatial aspect of spatio-temporal objects and focus on spatial features and their integration into spatial databases. In order to find a basis for discussion we view spatial features on three abstraction levels: the raw level, the model level, and the representation level. The *raw level* is centered around the concept of a *point set*. Space is assumed to be composed of infinitely many points; it corresponds to the n-dimensional Euclidean space $\mathbb{R}^n$, $n \in \mathbb{N}$. Each spatial feature S is regarded as an arbitrary, possibly infinite, point set $S \subseteq \mathbb{R}^n$. Hence, the raw or mathematical type of the set of all spatial features is $2^{\mathbb{R}^n}$. This perspective is very general in the sense that spatial features carry no special semantic information and that nothing is said about their formal definition and their representation. In the sequel we will only be interested in two-dimensional Euclidean space $\mathbb{R}^2$.

The *model level* gives semantics to point sets and forms the interface between the very general raw level and the concrete representation level. We consider only decidable point sets; otherwise there is no possibility to represent them on computers. Another important requirement is, of course, that the selected model is implementable at the representation level and that its objects can be embedded in a relational database. Besides, we would like to compare and combine this model with our temporal model discussed in the next section. As mentioned in the introduction, we here take an abstract data type (ADT) approach and use *spatial data types* for points and regions as appropriate abstractions of spatial phenomena; for reasons given in [EGSV97] we will not consider lines. Elements of spatial data types are called *spatial objects*. The set of all spatial objects is denoted by SO, and we also speak of the SO-model.

The *representation level* offers implementation concepts for spatial objects. Its main goal is the provision of finite object representations that can be efficiently stored in and retrieved from a database. Since it is usually difficult to represent arbitrarily shaped spatial objects, we here employ *linear* approximations of spatial objects. These have straightforward representations as we will see below. We call the objects of this level *linear spatial objects* and denote the set of all these objects by $\overline{\text{SO}}$. Out of several implementation alternatives, our selection is to approximate regions by sets of polygons possibly with polygonal holes. Points remain unchanged on this level.

On the model level we consider SO = *Point* ∪ *Region* where *Point* and *Region* are the set of all point and region objects, respectively. Points are the atoms of our underlying space, and hence a point is an element of $\mathbb{R}^2$. A *region* is a finite set of disjoint, closed, and connected areal components possibly with disjoint holes. This model is very general and closed under (appropriately defined) geometric union, intersection, difference, and complement operations. It allows regions to contain holes and islands within holes to any finite level.

A mathematical definition for regions can be based on point set topology which provides several operators identifying different structures of a point set (like its *interior*, *boundary*, and *closure*). A problem of point sets is that they can have geometric anomalies like isolated or dangling line or point features and missing lines and points in the form of cuts and punctures. From an application point of view, these anomalies are undesirable and avoided by the concept of regularity. A set $Y \subseteq \mathbb{R}^2$ is called *regular closed* if $Y = closure(interior(Y))$. Hence, we can define a *regularization function* which maps a set Y to its corresponding regular closed set: $reg(Y) := closure(interior(Y))$. A region is then defined as a regular closed set. Geometric operations on regions correspond to regular set operations on regular closed sets. Their definition preserves regularity and is reduced to the regularization of the result of the corresponding set operations on two regular closed sets [Ti80, ES97].

The $\overline{\text{SO}}$-model can now be defined as a linear specialization of the SO-model. Obviously, point objects remain unchanged. If we speak about regular closed sets, their topology is irrelevant. But if we intend to represent regular closed sets by finite means, we are forced to identify their structure. We are now interested in regions where the boundary of each component and each hole is formed by a sequence of straight line segments. To simplify our technical exposition in the sequel, we confine ourselves to a region model which defines a region as a single component without holes; the extension to general regions is straightforward (definitions for the general case can be found in [WB93, Sc97]). The (restricted) set *Polygon* of all polygons can then be defined as follows:

$$
\begin{aligned}
Polygon = \{P \in Region \mid \exists\, p_0, p_1, ..., p_{n\text{-}1} \in boundary(P) :&\\
&(1)\ \forall\, i \in \{0, ..., n\text{-}1\} : s_i = \{rp_i + (1\text{-}r)\, p_{(i+1) \bmod n} \mid r \in \mathbb{R}, 0 \le r \le 1\}\\
&(2)\ \forall\, 0 < i\text{+}1 < j < n : s_i \cap s_j = \varnothing\\
&(3)\ \forall\, i \in \{0, ..., n\text{-}1\} : s_i \text{ and } s_{i+1} \text{ are not collinear}\\
&(4)\ boundary(P) = \textstyle\bigcup_{0 \le i \le n\text{-}1} s_i\ \}
\end{aligned}
$$

Condition (1) describes the boundary as a sequence of non-intersecting segments. This definition provides a very efficient and finite description formalism for polygons which is known as *boundary representation*. Each polygon is described either as a sequence $(p_0, ..., p_{n\text{-}1})$ of points or as a sequence $(s_0, ..., s_{n\text{-}1})$ of segments with the semantics described above. Condition (2) requires that non-consecutive segments are disjoint. Condition (3) takes care of the minimality of the selected points p_i of the boundary to achieve a space-efficient representation. Condition (4) says that the union of all segments just forms the boundary. It is obvious that linear spatial objects are a strict subset of (general) spatial objects:

Lemma 2.1. $\overline{\text{SO}} \subset \text{SO}$.

Proof. Take circles and ellipses as examples for spatial objects bounded by curves. It is clear that these objects cannot be represented but only approximated by boundary representations. ■

2.2 Representation of Spatial Objects in Databases

We now consider the embedding of spatial objects in relational databases. A *relation scheme R* is written as $R(A_1 : D_1, ..., A_n : D_n)$ where the A_i are the *attributes* of R. For a relation $r : R(A_1 : D_1, ..., A_n : D_n)$ holds: $r \subseteq D_1 \times D_2 \times ... \times D_n$. Tuples are described in the form $(A_1 = x, A_5 = b, ...)$; only the values of interest are shown.

Principally, there are two methods of integrating spatial objects into relational databases. The first method is to embed spatial objects directly as ADTs, i.e., a single attribute value contains a complete spatial object: $R(S : \alpha, ...)$ for $\alpha \in$ GEO = {*Point*, *Region*, *Polygon*}. This applies to the SO and $\overline{\text{SO}}$ model. The corresponding relational data models are called SO-REL and $\overline{\text{SO}}$-REL; they denote the set of relations with at least one attribute of a (linear) spatial data type. For simplicity we assume that each such relation has exactly one spatial attribute. The second method leads us to S-REL which denotes the set of relations modeling spatial features only with atomic standard attribute types. S-REL documents the beginning of spatial database research [BS77, Ro87]. A polygon is represented as a set of tuples each storing the coordinates of two points representing a segment of the boundary representation of a polygon: $R(X_1 : \mathbb{R}, Y_1 : \mathbb{R}, X_2 : \mathbb{R}, Y_2 : \mathbb{R}, ...)$. If the values of X_1 and X_2 and the values of Y_1 and Y_2 are the same, the tuple represents a point.

For a later comparison of S-REL with SO-REL and $\overline{\text{SO}}$-REL, we define mappings from S-REL to $\overline{\text{SO}}$ and vice versa. We first give the representation of a linear spatial object as a relation $r \in$ S-REL. For this purpose we define a function *decompose* which transforms the representation $(p_0, ..., p_{n-1})$ of a polygon P (where $p_i = (x_i, y_i)$) into a relation, i.e., into a set of tuples, of type R.

$$decompose(P) = \bigcup_{0 \le i \le n-1} \{(X_1 = x_i, Y_1 = y_i, X_2 = x_{(i+1) \bmod n}, Y_2 = y_{(i+1) \bmod n})\}$$

The relation representing a linear spatial object o is given by:

$$\rho_S(o) = \begin{cases} \{(X_1 = x, Y_1 = y, X_2 = x, Y_2 = y)\} & \text{if } o = (x, y) \in Point \\ decompose(o) & \text{otherwise} \end{cases}$$

Now we look at the other case and assume that $r = \{(X_1 = x_0, Y_1 = y_0, X_2 = x_1, Y_2 = y_1, ...), ..., (X_1 = x_{n-1}, Y_1 = y_{n-1}, X_2 = x_0, Y_2 = y_0, ...)\} : R(X_1, Y_1, X_2, Y_2 : \mathbb{R}, ...) \in$ S-REL is a (sub-) relation containing only tuples describing one spatial object. Since ρ_S is injective, which means that different polygon objects have different representations in S-REL, we may uniquely assign a semantically correct polygon representation in S-REL to a polygon object. Let β be a function which given a set $L = \{s_0, ..., s_{n-1}\}$ of segments representing a polygon computes its boundary point set:

$$\beta(L) = \bigcup_{0 \le i \le n-1} \{rp_i + (1-r)\, p_{(i+1) \bmod n} \mid s_i = (p_i, p_{(i+1) \bmod n}), r \in \mathbb{R}, 0 \le r \le 1\}$$

The spatial object denoted by relation r is then given by:

$$\sigma_S(r) = \begin{cases} (x_1, y_1) \in Point & \text{if } |r| = 1 \wedge x_1 = x_2 \wedge y_1 = y_2 \\ P \in Polygon : boundary(P) = \beta(\bigcup_{0 \le i \le n-1} \{(x_i, y_i), (x_{i+1}, y_{i+1})\}) & \text{otherwise} \end{cases}$$

If r does not represent a proper polygon, we define $\sigma_S(r) = \perp$.

2.3 Expressiveness of Spatial Data Models

We are now able to compare the different spatial data models. As a first result, we show that S-REL and $\overline{\text{SO}}$-REL have the same expressive power.[1] The second result shows that SO-REL is more powerful than both S-REL and $\overline{\text{SO}}$-REL.

The main difference between $\overline{\text{SO}}$-REL/SO-REL and S-REL is that a tuple in $\overline{\text{SO}}$-REL/SO-REL is decomposed into a set of tuples (= sub-relation) in S-REL. We can define two simple transformations that map between $\overline{\text{SO}}$-REL and S-REL. Let $\nu[AS : A; f]$ be the well-known *nest* operator from the NF[2] relational model [SS86] that takes in addition to the set of attributes AS to be nested a function f that is applied to each resulting sub-relation r'. Then $f(r')$ is stored under the attribute A (instead of r'). Now any relation $r : R(X_1, Y_1, X_2, Y_2 : \mathbb{R}, ...) \in$ S-REL can be mapped to an equivalent relation $s \in \overline{\text{SO}}$-REL by

$$s = \nu[\{X_1, Y_1, X_2, Y_2\} : P; \sigma_S](r)$$

where $P : \alpha$ for $\alpha \in \{Point, Polygon\}$ is a new attribute for the resulting spatial objects. Similarly, the modified *unnest* operator $\mu[A : AS; f](r)$ applies the function f to the value of attribute A of each of r's tuples and produces a relation of schema AS that is embedded into r. Any relation $s \in \overline{\text{SO}}$-REL can then be transformed into an equivalent relation $r \in$ S-REL by

$$r = \mu[P : \{X_1, Y_1, X_2, Y_2\}; \rho_S](s)$$

We let $\nu_S(\text{S-REL}) = \{\nu[\{X_1, Y_1, X_2, Y_2\} : P; \sigma_S](r) \mid r \in \text{S-REL}\} - \{\perp\}$ (removing results of σ_S for non-polygon relations) and $\mu_S(\overline{\text{SO}}\text{-REL}) = \{\mu[P : \{X_1, Y_1, X_2, Y_2\}; \rho_S](s) \mid s \in \overline{\text{SO}}\text{-REL}\}$ and obtain:

1. There are, however, differences with regard to their efficiency.

Theorem 2.1. ν_S(S-REL) = $\overline{\text{S}}$O-REL and μ_S($\overline{\text{S}}$O-REL) = S-REL.

Proof. Since ρ_S is a total function, it is clear that each $\overline{\text{S}}$O-REL can be transformed into a corresponding S-REL. Since ρ_S is injective, the representation in S-REL for a polygon object is unique. Hence, the set of all semantically correct polygon representations in S-REL (this is just the set on which σ_S is defined) can be mapped to a polygon object. ■

If we drop the linear restriction of spatial objects, the ADT approach is more general. As a direct conclusion of Lemma 1 we obtain:

Theorem 2.2. $\overline{\text{S}}$O-REL ⊂ SO-REL.

3 Temporal Objects and Temporal Databases

3.1 A Model of Temporal Objects

When defining a model for temporal objects one has to decide about a model of time. There are quite engaged discussions in the temporal database community about this issue, in particular, whether time is discrete or continuous, and there seems to be no unique approach agreed upon yet. We choose – mainly for consistency with the spatial domains – time to be continuous, i.e., *time* = $\mathbb{R}$. Now anything that changes over time can be expressed as a function over time, i.e., the temporal version of objects of a type α is given by a function of type $time \rightarrow \alpha$, called a *temporal function*. The type of all (partial) temporal functions is simply:

$$\phi(\alpha) = time \rightarrow \alpha$$

We consider changes over time as a first-order property, i.e., we do not model changes of an already changing object so that type expressions like $\phi(\phi(point))$ are not legal.[2]

On the model level we have to deal with representations that are computationally tractable. This means that for an arbitrary temporal function $f \in \phi(\alpha)$ we can determine the value of f at any time of its domain. Thus, we restrict $\phi(\alpha)$ to computable functions. It is also important to be able to compute values of the inverse function, i.e., ask for the times at which a temporal object took a specific value. Further restrictions result from the need to integrate temporal objects into the relational model and from compatibility with the chosen model for spatial objects. Thus, we restrict the domain of ϕ to finite sets of time points and intervals. For any type α that has a total order < (and equality =) we define the type of non-empty (open and closed) intervals over α as follows:

$$\iota(\alpha) = \cup\{\{[x, y],]x, y], [x, y[,]x, y[\} \mid x, y \in \alpha\} - \{\varnothing\} \text{ where}$$
$$[x, y] = \{a \in \alpha \mid x \le a \le y\},\ ]x, y] = \{a \in \alpha \mid x < a \le y\},\ [x, y[= \{a \in \alpha \mid x \le a < y\}, \text{ etc.}$$

This way we can encode continuous parts of ϕ's domain by intervals, i.e., the domain of a temporal object is simply given by a finite set of pairwise disjoint intervals (any time point t can well be represented by a degenerated interval $[t, t] = \{t\}$.) We can now at the model level define the type constructor for *temporal objects* simply as:

$$\tau(\alpha) = \iota(time) \rightarrow \phi(\alpha)(= \iota(time) \rightarrow time \rightarrow \alpha)$$
$$\text{where } \forall\ \omega \in \tau(\alpha) : (1)\ \forall\ I, J \in dom(\omega) : I \cup J \notin \iota(time)$$
$$(2)\ \forall\ I \in dom(\omega) : dom(\omega(I)) = I$$

This means a temporal object ω is defined on a set of pairwise disjoint and non-adjacent intervals and associates with each interval of its domain a (partial) temporal function whose domain is just that interval. The set of all temporal objects is denoted by TO, and we also speak of the TO-model. There are at least two reasons for considering only *linear* temporal functions as a further restriction of temporal objects: (i) it is difficult to compute with general functions, and (ii) a linear temporal function has a straightforward representation, which is particularly important for the integration into relations: store the function values of the boundaries of intervals and use predefined interpretations for deriving function values for the interior of intervals.

2. Actually, the type $\phi(\phi(point))$ can be viewed as denoting velocities of moving points.

In order to formalize the notion of linearity we consider argument types α that have a certain algebraic structure (we call these types *linear smooth*): There must be a non-trivial type $\Lambda(\alpha) \subseteq \alpha \to \alpha$ of functions on α for which two conditions hold: first, a scalar multiplication is defined, i.e., $\forall f \in \Lambda(\alpha), r \in \mathbb{R}: r{\cdot}f : \alpha \to \alpha$ is a well-defined function (with $1{\cdot}f = f$). Second, the function $\Delta : \alpha \times \alpha \to \Lambda(\alpha)$ yields for two values $x, y \in \alpha$ a function δ which captures the "difference" between x and y; in particular, $\delta(x) = y$ must hold. Then, by virtue of the scalar multiplication, values in the interior of an interval can be computed by δ. For instance, for $\alpha = \mathbb{R}$ the usual linear transition from x to y is captured by $\Delta(x, y) = \lambda x'.x'+(y\text{-}x)$ where scalar multiplication is defined as: $r{\cdot}(\lambda x'.x'+(y\text{-}x)) = \lambda x'.x'+r{\cdot}(y\text{-}x)$. The reason why Δ is defined to return a function and not simply a difference value is that the linear interpretation for $\alpha \in$ GEO is given by affine mappings (see Section 4), and for these the functional view is much more easy to handle than the value approach.

Now the *linear temporal object* ($\overline{\text{TO}}$) model can be defined as a linear specialization of the TO-model (very much like $\overline{\text{SO}}$ is a specialization of SO). By $\llbracket t_1, t_2 \llbracket$ we denote an arbitrary open, closed, or semi-open time interval, and we let $\|t\| = (t\text{-}t_1)/(t_2\text{-}t_1)$. We say that a temporal function $f : time \to \alpha$ is *k-piecewise linear* if:

$$\begin{aligned}&\exists\, k \in \mathbb{N}: dom(f) = \cup_{1 \le i \le k} I_i \text{ with } I_i = \llbracket t_1, t_2 \llbracket \wedge \forall 1 \le i < j \le k: I_i \cap I_j = \emptyset \\ &\qquad \wedge \forall\, I_i: |I_i| > 1 \wedge \forall\, t \in I_i: f(t) = (\|t\|{\cdot}\Delta(x, y))\,(x) \\ &\qquad \text{where } x = \lim_{n\to\infty} f(t_1+1/n) \text{ and } y = \lim_{n\to\infty} f(t_2\text{-}1/n)\end{aligned}$$

A k-piecewise linear function is always also $(k+1)$-piecewise linear. To get a canonical (and efficient) representation we look for minimal decompositions of intervals. Therefore, we say that f is *minimally decomposed* (or *maximally piecewise*, or just *k-piecewise*) if f is k-piecewise linear, but not $(k\text{-}1)$-piecewise linear. Then the *minimal decomposition* of f is defined as the partition $\pi(f) = \{(I_i, f|_{I_i}) \mid 1 \le i \le k\}$ where $f|_D = \{(x, f(x)) \mid x \in D\}$. Now the type of linear temporal objects is defined by the type constructor $\overline{\tau}$ as follows.

$$\begin{aligned}\overline{\tau}(\alpha) = \{\omega \in \tau(\alpha) \mid\ & (1)\ \alpha \text{ is linear smooth} \\ & (2)\ \forall\, I \in dom(\omega): |I| = 1 \vee \omega(I) \text{ is } k\text{-piecewise}\}\end{aligned}$$

Note that we cannot simply restrict ω to be linear on each of its intervals, we rather have to refine this condition to finite partitions of each interval I because ω might have different linear behaviors on I. Therefore, we have used the notion of piecewise linearity. It is obvious that linear temporal objects are a strict subset of (general) temporal objects:

Lemma 3.1. $\overline{\text{TO}} \subset \text{TO}$.

Proof. Consider, for example, a temporal object containing a temporal function $f(t) = x_0{\cdot}t^2$. It is clear that f cannot be represented by a finite set of linear pieces. This shows the inclusion. ■

3.2 Representation of Temporal Objects in Databases

The integration of temporal objects into relational databases can be done principally in two ways: temporal objects can be embedded directly as ADTs, i.e., a single attribute contains a complete temporal object: $R(O : \tau(\alpha), \ldots)$. This applies to the TO and $\overline{\text{TO}}$ models. The corresponding data models are called TO-REL and $\overline{\text{TO}}$-REL, and they denote the set of relations with at least one attribute being of a (linear) temporal object type. For simplicity we assume in the sequel that each such relation has exactly one temporal attribute.[3] In contrast, T-REL denotes relations with only atomic attribute types (including *time*). T-REL as defined below gives a unifying view on different traditional tuple-timestamped[4] models of temporal databases. Each temporal object is represented by a set of tuples each storing a value of type α, a time stamp, and a flag B indicating the future value behavior: $R(A : \alpha, T : time, B : \{d, c, l\}, \ldots)$. B specifies the values in between two time stamps, i.e, given two tuples $(A = x, T = t_1, B = b, \ldots)$ and $(A = y, T = t_2, \ldots)$ of a relation $r \in$ T-REL where $\forall\, (T = t_3, \ldots) \in r: t_3 < t_1 \vee t_3 > t_2$, the value of A at any time $t_1 < t < t_2$, denoted by $A(t)$, is:

3. The general case requires that the time domains of different temporal objects have to be "synchronized" by finding a common interval refinement when mapping to T-REL. This is not difficult, but makes the definitions longer.
4. In contrast, attribute-timestamped models like that of [SS93] correspond more closely to the ADT view.

Interpretation	*Definition*	*b*	*Name*
completely undefined	$A^d(t) = \bot$	d	*discrete*
valid up to the next definition	$A^c(t) = x$	c	(*stepwise*) *constant*
changes continuously	$A^l(t) = (\lVert t \rVert \cdot \Delta(x, y))\ (x)$	l	*linear*

(In the informal model of [YC93] *spline interpolation* is suggested as another interpretation.) To be compatible with TO and $\overline{\text{T}}$O we consider a further flag $C : \mathbb{B}$ for distinguishing closed and open intervals: $C = true \Leftrightarrow A$ is a valid value at T (otherwise, A is used only for deriving values in the interior of the preceeding and/or the following interval).

In order to compare T-REL with TO-REL and $\overline{\text{T}}$O-REL we define the temporal object represented by a relation from T-REL. Let $r = \{(A = a_1, T = t_1, B = b_1, C = c_1, \ldots), \ldots, (A = a_n, T = t_n, B = b_n, C = c_n, \ldots)\} : R(A : \alpha, T : time, B : \{d, c, l\}, C : \mathbb{B}, \ldots) \in$ T-REL be a (sub-) relation containing only tuples describing one temporal object (i.e., the projection to all attributes $\notin \{A, T, B, C\}$ yields a relation of a single tuple). First, we derive the set of temporal functions for the intervals represented in r:

$$\Phi(r) = \{\{(t, A^{b_i}(t)) \mid t_i < t < t_{i+1}\} \cup \{(t_j, a_j) \mid c_j = true, j \in \{i, i{+}1\}\} \mid 1 \le i < n\}$$

Next we have to map each interval to its corresponding temporal function. We get: $\{(dom(f), f) \mid f \in \Phi(r)\}$. Note that this is *not* yet the final temporal object, since there are, in general, more intervals in the T-REL representation than in the corresponding temporal object. Therefore, we have to normalize by merging temporal functions on adjacent intervals. This can be done by the function γ:

$$\gamma(\omega) = \begin{cases} \gamma(\{(I \cup J, f \cup g)\} \cup \omega') & \text{if } \exists\, \omega' : \omega = \{(I, f), (J, g)\} \cup \omega' \text{ with } I \cup J \in \iota(time) \\ \omega & \text{otherwise} \end{cases}$$

Hence, the temporal object denoted by relation r is finally given by:

$$\sigma_T(r) = \gamma(\{(dom(f), f) \mid f \in \Phi(r)\})$$

For relations r that do not properly represent temporal objects $\sigma_T(r)$ is undefined, i.e., $\sigma_T(r) = \bot$.

Next we have to define the representation of a linear temporal object ω as a relation $r \in$ T-REL. Therefore, we first partition each interval of ω's domain into maximal sub-intervals so that the corresponding temporal function is linear on each of these sub-intervals. We obtain this through the minimal decomposition π. Note carefully, that we cannot represent "linear functions followed by a jump", but only jumps after stepwise constant parts because we have spent for each interval only one attribute of type α. This means that we cannot represent a function that evolves linearly from x to y and continues with $z \neq y$. Thus, the representation function ρ described in the sequel is only partially defined.

Consider a k-piecewise temporal function f. Let $dom(f) = \cup_{1 \le i \le k} I_i$ with $I_i = [\![t_{i,1}, t_{i,2}]\!]$, $x_i = \lim_{n \to \infty} f(t_{i,1} + 1/n)$, and $y_i = \lim_{n \to \infty} f(t_{i,2} - 1/n)$. If $x_i = y_i$, let $b_i = c$. Otherwise, if $y_i = x_{i+1}$, then $b_i = l$. Otherwise, $\rho''(f)$ (see below) is undefined. Then

$$\rho''(f) = \{(A = x_i, T = t_{i,1}, B = b_i, C = (t_{i,1} \in I_i)) \mid 1 \le i \le k\} \cup \{(A = y_k, T = t_{k,2}, B = d, C = (t_{k,2} \in I_k))\}$$

Now the relation representing any temporal function of a temporal object is given by:

$$\rho'(f) = \begin{cases} \{(A = f(t), T = t, B = d, C = true)\} & \text{if } dom(f) = \{t\} \\ \rho''(f) & \text{otherwise} \end{cases}$$

Finally, the relation representing a complete linear temporal object is:

$$\rho_T(\omega) = \cup_{(I, f) \in \omega}\, \rho'(f)$$

3.3 Expressiveness of Temporal Data Models

Next we can compare the different temporal data models. First, we show that $\overline{\text{T}}$O-REL is more expressive than T-REL, but with two simple extensions T-REL becomes equivalent to $\overline{\text{T}}$O-REL. This means that the ADT approach is essentially equivalent to simple temporal relational models as far as linear temporal behavior is concerned. We also show that, in general however, TO-REL *is* more powerful than both $\overline{\text{T}}$O-REL and T-REL.

The difference between $\overline{\text{T}}$O-REL and T-REL lies essentially in the fact that one tuple in $\overline{\text{T}}$O-REL is represented by a set of tuples (= sub-relation) in T-REL. We can define two simple transformations to map between $\overline{\text{T}}$O-REL and T-REL. Again we use the *nest* and *unnest* operators ν and μ. Now any relation $r : R(A : \alpha, T : time, B : \{d,c,l\}, C : \mathbb{B}, \ldots) \in$ T-REL can be transformed into an equivalent relation $s \in$ $\overline{\text{T}}$O-REL simply by

$$s = \nu[\{A, T, B, C\} : O; \sigma_T](r)$$

Likewise, any relation $s \in$ $\overline{\text{T}}$O-REL can be transformed into a T-REL relation r by:

$$r = \mu[O : \{A, T, B, C\}; \rho_T](s)$$

Let ν_T(T-REL) = $\{\nu[\{A, T, B, C\} : O; \sigma_T](r) \mid r \in$ T-REL$\}$ - $\{\perp\}$, and let μ_T(T-REL) = $\{\mu[O : \{A, T, B, C\}; \rho_T](r) \mid r \in$ $\overline{\text{T}}$O-REL$\}$. Now we first have:

Theorem 3.1. ν_T(T-REL) $\subset$ $\overline{\text{T}}$O-REL.

Proof. Since σ_T is a total function, it is clear that each T-REL can be transformed into a corresponding $\overline{\text{T}}$O-REL. The fact that the inclusion is proper is grounded in the partiality of ρ_T: since each linear function followed by a jump cannot be represented by a T-REL, there are more $\overline{\text{T}}$O-RELs than T-RELs. ■

There is an even more important difference between T-REL and $\overline{\text{T}}$O-REL that gets lost by lifting T-REL to the ADT-level of $\overline{\text{T}}$O-REL: non-temporal attributes in a $\overline{\text{T}}$O-REL exist independently from the domain of the temporal attribute. In contrast, an additional (implicit) rule would be needed to distinguish temporal attributes from non-temporal ones in T-REL. This reflects the fact that attribute-timestamped temporal models are, in general, more expressive than tuple-timestamped models.

If we extend T-REL by storing an additional α-attribute (and a second C-flag specifying the definedness at the end of intervals), we can actually represent all linear temporal objects in flat relations. Let us call such a model T^+-REL. (Of course, we have to redefine and extend the ρ_T and σ_T transformations, too.) Then:

Theorem 3.2. ν_T(T^+-REL) = $\overline{\text{T}}$O-REL and μ_T($\overline{\text{T}}$O-REL) = T^+-REL.

Still the ADT-approach is more general when we do not restrict ourselves to linear behaviors. As a direct corollary of Lemma 3.1 we obtain:

Theorem 3.3. $\overline{\text{T}}$O-REL $\subset$ TO-REL.

4 Spatio-Temporal Data Types and Data Models

Now that we know how to model spatial and temporal objects and how to integrate them into databases we can consider their combination.

4.1 Landscape of Spatio-Temporal Data Models

A straightforward approach is indicated by the fact that τ is a type constructor: it is obvious to apply τ to types from GEO to immediately obtain *spatio-temporal objects* (STO). The types of this model comprise moving objects, i.e., MOV = $\{\tau(Point), \tau(Region), \tau(Polygon)\}$. Again we can restrict ourselves to linear objects, both for the temporal and the spatial component, and obtain the following models and types:

Model	*linear component*	*Types*
STO	–	τ(*Point*), τ(*Region*)
$\overline{\mathrm{S}}\mathrm{TO}$	*spatial*	τ(*Point*), τ(*Polygon*)
$\mathrm{S}\overline{\mathrm{T}}\mathrm{O}$	*temporal*	$\bar{\tau}$(*Point*), $\bar{\tau}$(*Region*)
$\overline{\mathrm{ST}}\mathrm{O}$	*spatial & temporal*	$\bar{\tau}$(*Point*), $\bar{\tau}$(*Polygon*)

Note that before we can apply $\bar{\tau}$ to either geometric type $\alpha \in$ GEO we have to ensure that these are all linear smooth. Therefore, we have to identify reasonable types $\Lambda(\alpha)$. The choice here is not unique, but for points arbitrary vector movements, and for regions and polygons *affine mappings* provide well-understood and general models of geometric transformations that are also amenable to scalar multiplication and to the difference operator Δ. Actually, scalar multiplication is already defined for both vectors and affine mappings. The Δ operation is defined for points as $\Delta(p, q) = \lambda p'.p'+(q\text{-}p)$ where "+" and "-" are usual vector addition and subtraction. Thus Δ simply records the vector that translates p to q. The scalar multiplication is defined as $r\cdot(\lambda p'.p'+(q\text{-}p)) = \lambda p'.p'+r\cdot(q\text{-}p)$, and thus the intermediate positions of a point moving from p (directly) to q all lie on the straight line connecting p and q. For polygons (and regions) the difference is defined component-wise.[5] For two polygons we have: $\Delta(P, Q) = \lambda p.H\cdot p+v$ where the matrix

$$H = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and the vector } v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

contain altogether six variables that are fully determined by three pairs of corresponding points from P and Q as follows. For each two corresponding points $p = (x, y)$ and $q = (x', y')$ we know:

$$x' = a\cdot x+b\cdot y+v_x \text{ and}$$
$$y' = c\cdot x+d\cdot y+v_y$$

For three different pairs of points we thus obtain six equations which are sufficient to compute the parameters a, b, c, d, v_x, and v_y. Scalar multiplication is defined as $r\cdot(\lambda p.H\cdot p+v) = \lambda p.(r\cdot H)\cdot p+r\cdot v$. Now we can also see why we do not take affine mappings for points, but just vector translation: since it is not possible to infer affine transformation from just two points, it would be impossible to define Δ.

In the above table we have only listed ADTs. However, when we consider the integration into relations we can also as a further alternative distinguish the encoding of objects by a set of tuples. This applies to the spatial as well as to the temporal object part. We get the following eight modeling combinations where for each model we give its name and the attribute types[6] of spatio-temporal objects (see table below).

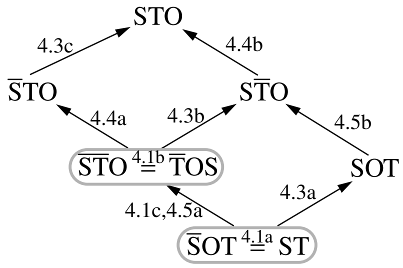
	TO		$\overline{\mathrm{T}}\mathrm{O}$		T (k snapshots)	
SO	τ(*Region*)	STO	$\bar{\tau}$(*Region*)	$\mathrm{S}\overline{\mathrm{T}}\mathrm{O}$	$Region^{(k)}$	SOT
$\overline{\mathrm{S}}\mathrm{O}$	τ(*Polygon*)	$\overline{\mathrm{S}}\mathrm{TO}$	$\bar{\tau}$(*Polygon*)	$\overline{\mathrm{ST}}\mathrm{O}$	$Polygon^{(k)}$	$\overline{\mathrm{S}}\mathrm{OT}$
S (m segments)			[$\bar{\tau}(Line)^m$]	$\overline{\mathrm{T}}\mathrm{OS}$	$Line^{(k\cdot m)}$	ST

Apart from the already mentioned "full" ADT versions (i.e., spatial and temporal objects are both integrated as ADTs), the model SOT ($\overline{\mathrm{S}}\mathrm{OT}$) denotes a model where (linear) spatial objects are integrated into a tuple-timestamped temporal database: for each snapshot the currently valid version of the spatial object is stored. This is indicated by the exponent (k) expressing that there are k tuples representing all the k snapshots. Similarly, $\overline{\mathrm{T}}\mathrm{OS}$ denotes the model where spatial objects (i.e., polygons) are encoded by m tuples – for each segment one tuple –, and the temporal behavior is given by linear temporal objects. This means each evolving polygon is represented by m temporal objects storing the behavior of each segment. This seems to be a rather unrealistic model (mostly because temporal object models do not exist so far), however, SOT and $\overline{\mathrm{S}}\mathrm{OT}$ are conceivable, since spatial object models do already exist. So these two models describe the option of simply combining existing database technol-

5. We notice that the number of components cannot change for linear areas. So we cannot model the splitting or merging of regions.
6. We give only the type for areal objects; for points all entries in the first (second) column are τ(*Point*) ($\bar{\tau}$(*Point*)), and the third column always contains $Point^{(k)}$.

ogy for spatial and temporal databases. We have omitted a possible model TOS of unconstrained temporal objects storing segment representations of polygons, since it seems to be rather difficult to “synchronize” arbitrary temporal line objects so that they always complement into a proper polygon. Finally, the most simple model that does not use ADTs at all is the ST model which represents a changing polygon by $k{\cdot}m$ tuples where m tuples representing one polygon snapshot get a common time stamp.

4.2 Expressive Power of Spatio-Temporal Models



All the different models presented above form a hierarchy which we will describe next. Actually, we obtain an expressive power lattice as shown on the right. An arrow from model A to model B means “A is less expressive than B”, i.e., $A \subset B$. The edge labels serve as indices to the corresponding theorems. First, we combine and generalize the results of Theorems 2.1 and 3.1. Apart from aggregating spatial and temporal objects in an ST-REL separately (by means of ν_S and ν_T), we can also consider the quasi-simultaneous aggregation ν_{ST}(ST-REL) = $\nu_T(\nu_S$(ST-REL)). μ_{T^+} is the extension of μ_T into a total function mapping to the extended representation ST$^+$-REL, and ν_{T^+} is the corresponding extension of ν_T. We also use ν_{ST^+}(ST$^+$-REL) = $\nu_{T^+}(\nu_S$(ST$^+$-REL)).

Theorem 4.1. (a) ν_S(ST-REL) = $\overline{\mathrm{S}}\mathrm{OT}$-REL
(b) $\nu_{ST^+}(\mu_{T^+}(\overline{\mathrm{T}}\mathrm{OS}$-REL)) = $\overline{\mathrm{S}}\overline{\mathrm{T}}\mathrm{O}$-REL
(c) ν_T(ST-REL) $\subset$ $\overline{\mathrm{T}}\mathrm{OS}$-REL
(d) ν_{ST}(ST-REL) $\subset$ $\overline{\mathrm{S}}\overline{\mathrm{T}}\mathrm{O}$-REL

Proof. Part (a) follows directly from Theorem 2.1. Part (b) deserves some explanations: First, we cannot simply apply ν_S to $\overline{\mathrm{T}}\mathrm{OS}$-REL because the spatial attributes are hidden in temporal objects, so we have to unpack them beforehand. However, we must be very careful here: since μ_T is not totally defined on $\overline{\mathrm{T}}\mathrm{OS}$-REL (ST-REL is a true subset of $\overline{\mathrm{T}}\mathrm{OS}$-REL) we have to map to the extended representation ST$^+$-REL by means of the extended unnesting function μ_{T^+}. We then know: $\mu_{T^+}(\overline{\mathrm{T}}\mathrm{OS}$-REL) = ST$^+$-REL. Now we can aggregate the spatial objects and get (from Theorem 2.1 or part (a)): ν_S(ST$^+$-REL) = $\overline{\mathrm{S}}\mathrm{OT}^+$-REL. Finally, we can aggregate the temporal objects and obtain (as a corollary of Theorem 3.2): $\nu_{T^+}(\overline{\mathrm{S}}\mathrm{OT}^+$-REL) = $\overline{\mathrm{S}}\overline{\mathrm{T}}\mathrm{O}$-REL. Part (c) follows directly from Theorem 3.1, and (d) follows from (b) and (c). ■

We also have corresponding results for ST$^+$-REL:

Theorem 4.2. (a) ν_{T^+}(ST$^+$-REL) = $\overline{\mathrm{T}}\mathrm{OS}$-REL
(b) ν_{ST^+}(ST$^+$-REL) = $\overline{\mathrm{S}}\overline{\mathrm{T}}\mathrm{O}$-REL

Theorems 4.1 and 4.2 express relationships of the flat relational model w.r.t. (linear) ADT models. Next we show relationships that result from the polygons being special cases of general regions.

Theorem 4.3. (a) $\overline{\mathrm{S}}\mathrm{OT}$-REL $\subset$ SOT-REL
(b) $\overline{\mathrm{S}}\overline{\mathrm{T}}\mathrm{O}$-REL $\subset$ $\mathrm{S}\overline{\mathrm{T}}\mathrm{O}$-REL
(c) $\overline{\mathrm{S}}\mathrm{TO}$-REL $\subset$ STO-REL

This theorem is essentially a corollary of Lemma 2.1. Similarly, as a corollary of Lemma 3.1 we obtain the following result expressing that relations with linear temporal objects are less expressive than relations with general temporal objects:

Theorem 4.4. (a) $\overline{\mathrm{S}}\overline{\mathrm{T}}\mathrm{O}$-REL $\subset$ $\overline{\mathrm{S}}\mathrm{TO}$-REL
(b) $\mathrm{S}\overline{\mathrm{T}}\mathrm{O}$-REL $\subset$ STO-REL

And finally, as a corollary of Theorems 3.1 and 3.2, we obtain (note that part (a) is actually equivalent to Theorem 4.1 (c)):

Theorem 4.5. (a) $\nu_T(\overline{\mathrm{S}}\mathrm{OT}\text{-REL}) \subset \overline{\mathrm{ST}}\mathrm{O}\text{-REL}$
(b) $\nu_T(\mathrm{SOT}\text{-REL}) \subset \mathrm{S}\overline{\mathrm{T}}\mathrm{O}\text{-REL}$
(c) $\nu_{T^+}(\overline{\mathrm{S}}\mathrm{OT}^+\text{-REL}) = \overline{\mathrm{ST}}\mathrm{O}\text{-REL}$
(d) $\nu_{T^+}(\mathrm{SOT}^+\text{-REL}) = \mathrm{S}\overline{\mathrm{T}}\mathrm{O}\text{-REL}$

It is very instructive to imagine spatio-temporal objects as 3D-objects. Then the different models presented relate directly to different features and restrictions of 3D-objects. For instance, STO describes rather arbitrary volumes (or curves in the case of points), whereas $\overline{\mathrm{S}}\mathrm{TO}$ is restricted to region objects with polygonal faces parallel to the *x*-*y* plane. $\mathrm{S}\overline{\mathrm{T}}\mathrm{O}$ ($\overline{\mathrm{ST}}\mathrm{O}$) restricts STO ($\overline{\mathrm{S}}\mathrm{TO}$) further to straight translations and scalings plus rotations w.r.t. the *t*-axis (for points: translations only). Two severe restrictions of $\overline{\mathrm{ST}}\mathrm{O}$ (that result from affine mappings) are: (i) the number of components cannot change, and (ii) the number of vertices of polygons cannot change. When considering linear representations (to facilitate efficient computations) and 3D-objects, we can also imagine moving regions being represented by *polyhedra*. It is then interesting to note that polyhedra are not comparable in expressive power to $\overline{\mathrm{ST}}\mathrm{O}$: polyhedra cannot represent rotations, but they can well model changes in the numbers of components and polygon vertices.

5 Conclusions

We have presented a new model for temporal objects and temporal databases that, in particular, offers quite different modeling options for spatio-temporal databases. The investigation of the relative expressive powers of the different models gives a clear picture of the relationships between these models. In particular, it can be seen that, compared with the traditional (flat) view of temporal databases, the ADT approach is more versatile and offers much more control over temporal behavior, even for linearly constrained objects. Future work should consider other specific spatio-temporal object models (such as polyhedra) in more detail.

References

[BS77] R.R. Berman & M. Stonebraker. GEO-QUEL: A System for the Manipulation and Display of Geographic Data. *Computer Graphics*, vol. 11, no. 2, pp. 186-191, 1977.

[CCR93] Z. Cui, A.G. Cohn & D.A. Randell. Qualitative and Topological Relationships in Spatial Databases. *3rd Int. Symp. on Advances in Spatial Databases* (*SSD '93*), LNCS 692, pp. 296-315, 1993.

[EGSV97] M. Erwig, R.H. Güting, M. Schneider & M. Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *Technical Report 224*, FernUniversität Hagen, 1997.

[ES97] M. Erwig & M. Schneider. Vague Regions. *5th Int. Symp. on Advances in Spatial Databases* (*SSD '97*), LNCS 1262, pp. 298-320, 1997.

[Ga64] S. Gaal. *Point Set Topology.* Academic Press, 1964.

[GS95] R.H. Güting & M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, vol. 4, pp. 100-143, 1995.

[Gü88] R.H. Güting. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. *Int. Conf. on Extending Database Technology* (*EDBT '88*),pp. 506-527, 1988.

[Gü94] R.H. Güting. An Introduction to Spatial Database Systems. *VLDB Journal*, vol. 3, no. 4, pp. 357-399, 1994.

[PBG94] J. Paredaens, J. van den Bussche & D. van Gucht. Towards a Theory of Spatial Database Queries. *13th ACM Symp. on Principles of Database Systems* (*PODS '94*), pp. 279-288, 1994.

[Ro87] J.W. van Roessel. Design of a Spatial Data Structure Using the Relational Normal Forms. *Int. Journal of Geographical Information Systems*, vol. 1, no. 1, pp. 33-50, 1987.

[Sc97] M. Schneider. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*. LNCS 1288, Springer-Verlag, 1997.

[SH91] P. Svensson & Z. Huang. Geo-SAL: A Query Language for Spatial Data Analysis. *2nd Int. Symp. on Advances in Spatial Databases* (*SSD '91*), pp. 119-140, 1991.

[SRG83] M. Stonebraker, B. Rubenstein & A. Guttmann. Application of Abstract Data Types and Abstract Indices to CAD Data Bases. *ACM/IEEE Conf. on Engineering Design Applications*, pp. 107-113, 1983.

[SS86] H.-J. Schek & M.H. Scholl. The Relational Model with Relation-Valued Attributes. *Information Systems*, vol. 11, no. 2, pp. 137-147, 1986.

[SS93] A. Segev & A. Shoshani: A Temporal Data Model Based on Time Sequences. Chapter 11 of [TCG+93], pp. 248-270, 1993.

[St86] M. Stonebraker. Inclusion of New Types in Relational Data Base Systems. *2nd Int. Conf. on Data Engineering*, pp. 262-269, 1986.

[SV89] M. Scholl & A. Voisard. Thematic Map Modeling. *1st Int. Symp. on the Design and Implementation of Large Spatial Databases* (*SSD '89*), pp. 167-190, 1989.

[TCG+93] A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev & R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings Publishing Company, 1993.

[Ti80] R.B. Tilove. Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE Transactions on Computers*, vol. C-29, pp. 874-883, 1980.

[WB93] M.F. Worboys & P. Bofakos: A Canonical Model for a Class of Areal Spatial Objects. *3rd Int. Symp. on Advances in Spatial Databases* (*SSD '93*), LNCS 692, pp. 36-52, 1993.

[Wo94] M.F. Worboys: A Unified Model for Spatial and Temporal Information. *The Computer Journal*, vol 37, no.1, pp. 27-34, 1994.

[YC93] T.S. Yeh & B. de Cambray: Time as a Geometric Dimension for Modeling the Evolution of Entities: A 3D Approach. *2nd Int. Conf. on Integrating GIS and Environmental Modeling*, 1993.