Contents

Constraint-Based Interoperability of Spatiotemporal Databases

Jan Chomicki
Dept. of Computer Science
Monmouth University
West Long Branch, NJ 07764
USA
chomicki@monmouth.edu

Peter Z. Revesz
Dept. of Computer Science and Engineering
University of Nebraska–Lincoln
Lincoln, NE 68588
USA
revesz@cse.unl.edu

Abstract

We propose constraint databases as an intermediate level facilitating the interoperability of spatiotemporal data models. Constraint query languages are used to express translations between different data models. We illustrate our approach in the context of a number of temporal, spatial, and spatiotemporal data models.

1 Introduction

Very large temporal and spatial databases are a common occurrence nowadays. Although they are usually created with a specific application in mind, they often contain data of potentially broader interest, e.g., historical records or geographical data. By *database interoperability* we mean the problem of making the data from one database usable to the users of another. Data sharing between different applications and different sites is often the preferable mode of interoperation. As [29] says 'Efficiency, security and availability all argue for shipping the data to the downstream database rather than providing integrated access to both systems." But sharing of data (and application programs developed around it), facilitated by the advances in network technology, is hampered by the incompatibility of different data models and formats used at different sites. Semantically identical data may be structured in different ways. Also, the expressive power of some data models is limited.

A temporal database may have been built using one of the many temporal extensions of the relational data model (the book [33] describes at least 12 such extensions which are mutually incompatible), using a customized temporal data model, or simply using SQL (or some of *its* extensions). There may be many application programs and complex queries that have been developed for this database. The situation in the area of spatial databases is similar [25, 38], often with a considerable investment in software tools tuned to specific data models.

Temporal and spatial databases share a common characteristic: they contain *interpreted data*, associated with uninterpreted data in a systematic way. For example, a temporal database may contain the historical record of all the property deeds in a city. A spatial database may contain the information about property boundaries. Moreover, as this example shows, spatial and temporal data are often mixed in a single application.

In this research, we propose that *constraint databases* [20] be used as a common language layer that makes the interoperability of different temporal, spatial and spatiotemporal databases possible. Constraint databases generalize the classical relational model of data by introducing *generalized tuples*: quantifier-free formulas in an appropriate constraint theory. For example, the formula $1950 \leq t \leq 1970$ describes the interval between 1950 and 1970, and the formula $0 \leq x \leq 2 \wedge 0 \leq y \leq 2$ describes the square area with corners $(0,0)$, $(0,2)$, $(2,2)$, and $(2,0)$. The constraint database technology makes it possible to finitely represent infinite sets of points, which are common in temporal and spatial database applications. We list below some further advantages of using the constraint database technology:

1. *Wide spectrum of data models.* By varying the constraint theory, one can accommodate a variety of different data models. By syntactically restricting constraints and generalized tuples, one can precisely capture the expressiveness of different models.

2. *Broad range of available query languages.* Relational algebra and calculus, Datalog and its extensions are all applicable to constraint databases. Those languages have well-studied formal semantics and computational properties, and are thus natural vehicles for expressing translations between different data models. Also, constraint query languages may be able to express queries inexpressible in the query languages of the interoperated data models, enhancing in this way the expressive power of the latter. This is more a practical than a theoretical contribution. We simply mean that if, for instance, we have a TQuel database, then translation to a constraint database with dense order constraints allows querying by Datalog, a query language which is more expressive than TQuel. In the paper we show how to enhance, through interoperability, the expressive power of the query language of the spatiotemporal data model of Worboys [37].

3. *Decomposability.* The problem of translating between two arbitrary data models, which is hard, is decomposed into a pair of simpler problems: translating one data model to a class C of constraint databases, and then translating C to the other data model. Also, by using a common constraint basis, we need to specify only $4n$ instead of $n(n-1)$ translations for n different data models.

4. *Combination and interaction of spatial and temporal data within a single framework.* This is an issue of considerable recent interest [13, 16].

In this paper we address the issue of *application-independent* interoperability of spatiotemporal databases. We show that the translations between different data models can be defined independently of any specific application that uses those models. We distinguish between *data* and *query* interoperability. For the former, it is the data that is translated to a different data model, while the latter concerns the translation of queries. The constraint database paradigm is helpful in both tasks. For *data interoperability,* constraint databases serve as a mediating layer and translations between different data models are expressed using constraint queries. For *query interoperability,* it is the constraint query languages themselves that serve as the intermediate layer. In an actual implementation, the presence of a mediating constraint layer may be completely hidden from the user. In this paper we study only *data interoperability.* Query interoperability is a topic of future research.

We show below two scenarios in which our approach may be useful in practice.

Data Casting. The user of a data model Δ_2 wants to query a database D_1 developed under a data model Δ_1. He translates D_1 to a Δ_2-database D_2 (using constraint databases as an intermediate layer) that he can subsequently query using the query language of Δ_2. (As a practical matter, if a user is interested in a query Q_2 in Δ_2, then only the part of the database that is relevant to Q_2 needs to be translated.)

Query Enhancement. The user of a data model Δ_1 wants to augment the power of the query language of Δ_1. For example, this language may be unable to express recursive queries. However, such queries can be formulated in an appropriate constraint query language. Thus whenever the user wants to run such a query on a database D_1, he first translates D_1 to a constraint database, runs the query in the constraint query language on it (using a constraint query engine), and translates the result back to Δ_1. (N.b., interoperating query results is an often neglected aspect of database interoperability.)

The plan of the paper is as follows. In Section 2 we define a very general notion of a data model and introduce a number of data models that will be studied in the rest of the paper: the TQuel data model for temporal databases, the 2-spaghetti model for spatial databases, and two spatiotemporal models (one of which is new). We believe that those models are representative of a large part of spatiotemporal data models that occur in practice. In Section 3 we characterize the expressiveness of the above data models using appropriately defined classes of constraint databases. In Section 4, which is the most technically involved part of the paper, we show that the bulk of the translations between the data models can be expressed using first-order constraint query languages. In particular, we show that the boundary, the vertices and the edges of a spatial object specified by linear arithmetic constraints can be defined using first-order queries with linear arithmetic constraints only. As an application of our techniques, we show in Section 5 how the expressive power of the query language of an existing spatiotemporal data model can be enhanced by data interoperability. In Section 6 we discuss related work. In Section 7 we conclude the paper and point out directions for future work in this area.

2 Data Models

2.1 Database Interoperability

By *database interoperability* we mean the problem of making the data from one database usable to the users of another. There are many possible sources of mismatches between different databases [19]: they may use different data models, the schemas may not match, some data may be missing or inconsistent etc. In this paper we limit our attention to the differences in the data models and are thus concerned with *application-independent* interoperability.

2.2 Basic Notions

A *data model* Δ consists of a set of valid databases $I(\Delta)$ and a set of valid queries $L(\Delta)$. All valid databases are finite. We assume that for every valid database D in a data model Δ, the abstract semantics of D is given as a first-order structure $\theta_\Delta(D)$. (Often, the abstract semantics is not given in the published description of the data model but has to be inferred from it.) We will term D a *concrete* representation of the structure $\theta_\Delta(D)$ (which may be infinite). Examples are given later in this section.

Definition 2.1 Two databases $D_1 \in I(\Delta_1)$ and $D_2 \in I(\Delta_2)$ are *equivalent* if $\theta_{\Delta_1}(D_1) = \theta_{\Delta_2}(D_2)$.

Definition 2.2 [2, 1] The *data expressiveness* of the data model Δ is the set $E_\Delta = \{\theta_\Delta(D) : D \in I(\Delta)\}$.

Currently used data models can substantially differ in terms of data expressiveness. For example, some can only represent finite relations.

Given two data models Δ_1 and Δ_2, there are two fundamentally different ways to query databases from $I(\Delta_1)$ using queries from $L(\Delta_2)$. These two approaches are:

1. *Data interoperability:* For a given database $D_1 \in I(\Delta_1)$ an equivalent database $D_2 \in I(\Delta_2)$ is constructed. Then D_2 can be queried using queries in $L(\Delta_2)$. Data interoperability opens up the data of Δ_1 to the users of Δ_2, making direct data sharing possible.
2. *Query interoperability:* This means that for a given query $Q_2 \in L(\Delta_2)$ an equivalent query $Q_1 \in L(\Delta_1)$ is constructed. Then Q_1 can be evaluated over the given database $D_1 \in I(\Delta_1)$ and the result translated back to $I(\Delta_2)$. Query interoperability enables

the users of Δ_2 to request the evaluation of queries within Δ_1. In this case data sharing is indirect.

Note that the above definitions characterize the *semantics* of data and query interoperability. In any *implementation* of data interoperability only a part of the database that is relevant for the given query will be translated. Note also that query interoperability relies on data interoperability to translate the query result. For some, e.g., Boolean, queries such translation will be trivial.

In this paper we concentrate on *data interoperability.* Note that for the data interoperability between Δ_1 and Δ_2 to be possible the data expressiveness E_{Δ_1} has to be contained in or equal to E_{Δ_2}. Otherwise, a database $D_2 \in I(\Delta_2)$ which is equivalent to a given database $D_1 \in I(\Delta_1)$ may not exist.

2.3 The TQuel Data Model

TQuel is a popular model for representing temporal data. (For the puprose of this paper, we chose TQuel over TSQL2 [32], because TQuel is simpler and TSQL2 is still in flux.) In the TQuel data model each relation contains two special attributes called *From* and *To* to represent *valid time.* The value of these temporal attributes must be integers or the special constants $-\infty$ or $+\infty$. The *From* and *To* values represent the endpoints of an interval. Such intervals in different tuples with identical nontemporal components have to be disjoint. (Another time dimension, transaction time, can also be present and is represented similarly to valid time. For simplicity we do not consider it here.)

The abstract semantics of a TQuel database is a relational database which has the same scheme as the TQuel database except the temporal attributes *To* and *From* are replaced with a single temporal attribute. The abstract semantics is point-based and hides the implementation details, in this case the fact that intervals are used. For each TQuel tuple of the form $r(a_1, \ldots, a_k, b_1, b_2)$ the abstract model contains the tuples

$$r(a_1, \ldots, a_k, b_1), \ldots, r(a_1, \ldots, a_k, b_2).$$

Example 2.1 Table 1 is a TQuel representation of the relation in Table 2.

Name	*Company*	*From*	*To*
Anderson	AT&T	1980	1993
Brown	IBM	1985	1996
Clark	Lotus	1990	1991

Table 1: TQuel DB researcher relation

Name	*Company*	*Year of Employment*
Anderson	AT&T	1980
⋮	⋮	⋮
Anderson	AT&T	1993
Brown	IBM	1985
⋮	⋮	⋮
Brown	IBM	1996
Clark	Lotus	1990
Clark	Lotus	1991

Table 2: DB researcher relation

The semantics of TQuel queries assumes the above abstract, point-based view. For example, a temporal join is implemented using interval intersection. On the other hand, the syntax of TQuel queries refers explicitly to intervals. Various built-in operators that work on intervals, e.g., `overlap`, are provided. The issue of points vs. intervals in temporal query languages is discussed in detail in [8, 11, 34].

2.4 The K-spaghetti Data Model

The K-spaghetti data model [24] is a very popular model for representing K-dimensional geometric objects for CAD (Computer Aided Design) [23] and GIS (Geographic Information Systems) [38]. In GIS applications typically $K = 2$ because the objects of interest are planar, while in CAD applications $K \geq 3$. The basic idea is to provide a general relational representation for geometric objects.

In this paper we concentrate on the 2-spaghetti (planar) data model. In this data model we can represent only spatial objects that are composed of finite unions of closed convex polygons. We further restrict the 2-spaghetti data model. First, we assume that the polygons have been triangulated. In this way polygons can be represented using first-normal form relations with a fixed number of attributes. Each triangle is represented by its three corners. Degenerate triangles (points or segments) can be represented in the same way. There are many good algorithms from computational geometry for triangulating polygons [26]. Second, to simplify the presentation we assume a single thematic attribute which can be conveniently interpreted as object identifier. The issue of K-spaghetti for $K \geq 3$ is addressed in Section 7.

Example 2.2 Let us consider Figure 1. In the 2-spaghetti model the spatial objects in Figure 1 are represented by the relation in Table 3. Note that the rectangle is represented by two and the pentagon by three triangles.

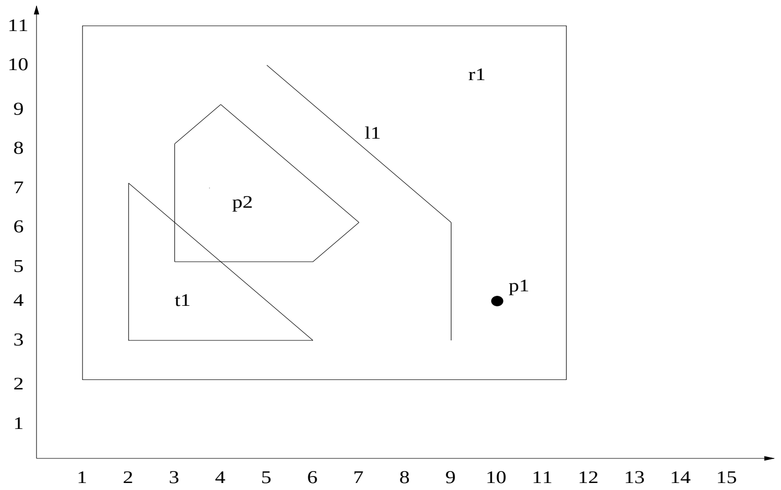

Figure 1: Spatial objects

id	x	y	x'	y'	x''	y''
p1	10	4	10	4	10	4
l1	5	10	9	6	9	6
l1	9	6	9	3	9	3
t1	2	3	2	7	6	3
r1	1	2	1	11	11.5	11
r1	11.5	11	11.5	2	1	2
p2	3	5	3	8	4	9
p2	4	9	7	6	3	8
p2	3	5	7	6	3	8

Table 3: Triangular representation of spatial figure

The abstract semantics of a 2-spaghetti relation r contains all the tuples w'' such that for some tuple $w' \in r$:

- the values of the thematic attribute of w'' and w' are the same,
- w'' has two spatial attributes x and y with values equal to the (x, y)-coordinates of a point within the triangle described by w'.

Thus, similarly to TQuel, it is point-based.

Typically, query languages for a 2-spaghetti data model hide the internal representation of spatial objects, referring explicitly to the objects themselves. Various built-in operators that work on polygons, e.g., `overlap`, are provided. The semantics of those languages is also point-based.

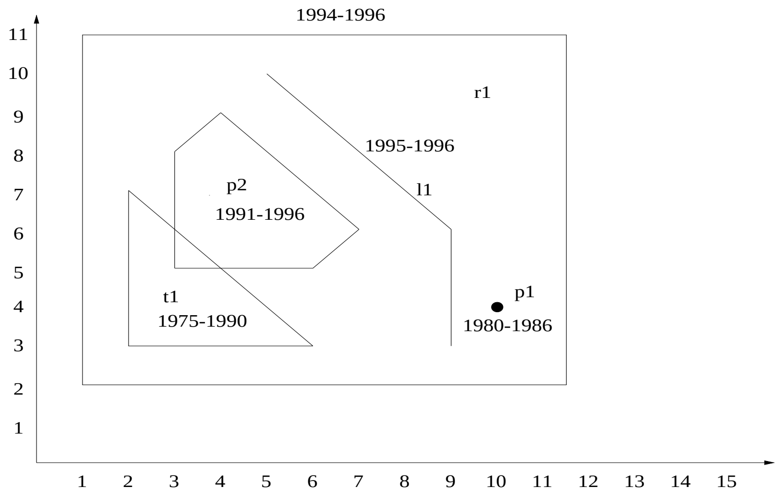

Figure 2: Spatiotemporal objects

2.5 The Spatiotemporal Data Model of Worboys

The spatiotemporal data model of Worboys [37] is a recent example of a data model that can represent both spatial and temporal information about objects in a database. *Worboys relations* (our term) are 2-spaghetti relations where spatial objects are timestamped with temporal extents which can be intervals or finite unions of intervals. Without loss of generality, we consider a specific version of the Worboys model where 2-spaghetti relations are triangular (as above) and temporal extents are single intervals (as in TQuel).

Example 2.3 Let us consider Figure 2. It is like Figure 1, except that we have added a time interval to each spatial object. The interval tells us from which year to which year the object existed (valid time). For each object there could be several such intervals. Figure 2 can be represented using a relation identical to the one in Table 3, except that two temporal attributes, *From* and *To*, are added to encode the appropriate intervals (see Table 4).

The abstract semantics of the Worboys spatiotemporal data model is the cross product of the abstract semantics given for the 2-spaghetti spatial data model and the abstract semantics given for the TQuel temporal data model. Thus, Worboys relations represent abstract relations with one temporal and two spatial dimensions. Moreover, the temporal and the spatial dimensions are independent [6]. Thus, only *discrete* changes, not continuous ones can be represented in this model. For example, the relationship that exists between time and the area covered by an incoming tide cannot be represented using Worboys relations.

The Worboys model, like TQuel, allows one more dimension for time (transaction time) if necessary. Transaction time can be handled like valid time and we do not discuss it here. The query language of the Worboys data model is a variant of relational algebra containing

id	x	y	x'	y'	x''	y''	*From*	*To*
p1	10	4	10	4	10	4	1980	1986
l1	5	10	9	6	9	6	1995	1996
l1	9	6	9	3	9	3	1995	1996
t1	2	3	2	7	6	3	1975	1990
r1	1	2	1	11	11.5	11	1994	1996
r1	11.5	11	11.5	2	1	2	1994	1996
p2	3	5	3	8	4	9	1991	1996
p2	4	9	7	6	3	8	1991	1996
p2	3	5	7	6	3	8	1991	1996

Table 4: Representation in the Worboys Model

the operators for spatial and temporal projection, temporal selection, and a general operator called "β-product" that can be used to simulate spatial selection, union, intersection, and difference. The semantics of the language is point-based.

2.6 The Parametric 2-spaghetti Data Model

This data model is a new model that we introduce in this paper. It generalizes the Worboys data model in a natural way by allowing an interaction between spatial and temporal attributes. Vertex coordinates can now be *linear functions of time.*

Example 2.4 Let us suppose that we have a rectangular area on a shore as shown in Figure 3. A tide is coming in and the water level continuously changes and is shown as a line marked by the time the water rises to that line. The front edge of the tide water is a linear function of time. In this case the area flooded by the water will be a point at 1:00 am, a triangle at 8:00 am, a quadrangle at 10:00 am, and a pentagon at 1:00 pm. This data can be represented as in Table 5. Notice that this table is no longer a standard relation but rather a *parametric* one (with parameter t).

The abstract semantics of a parametric 2-spaghetti relation r is a possibly infinite relation S defined in two stages. First, the set I_r of all the *instantiated tuples* of r is defined. An instantiated tuple w' of r is obtained from some parametric tuple w of r by:

1. substituting for the parameter t a value t_0 which falls between the *From* and *To* values in w,

2. removing the *From* and *To* attributes and adding a new temporal attribute t with the value t_0 (like in the semantics of TQuel).

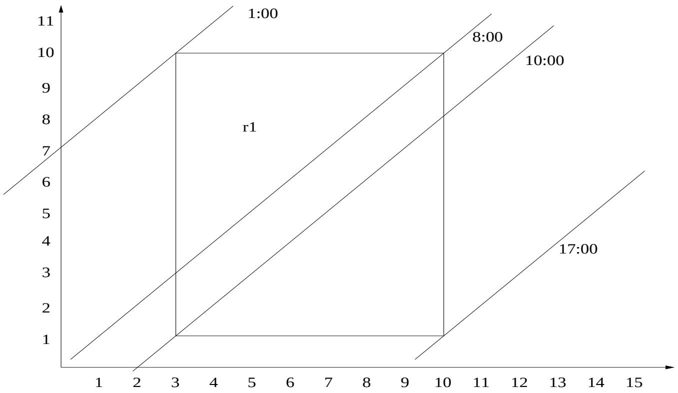

Figure 3: Continuous change

id	x	y	x'	y'	x''	y''	*From*	*To*
r1	3	10	3	10	3	10	1	$+\infty$
r1	3	10	3	11-t	2+t	10	1	8
r1	3	10	3	3	10	10	8	$+\infty$
r1	3	3	10	10	3	11-t	8	10
r1	10	10	3	11-t	10	18-t	8	10
r1	3	3	10	10	3	1	10	$+\infty$
r1	10	10	3	1	10	8	10	$+\infty$
r1	3	1	10	8	t-7	1	10	17
r1	t-7	1	10	18-t	10	8	10	17
r1	3	1	10	1	10	8	17	$+\infty$
r1	10	1	10	1	10	1	17	$+\infty$

Table 5: Parametric Triangular Representation

In every instantiated tuple the spatial attributes denote, as in the 2-spaghetti model, the vertices of a triangle. Thus in the second stage, the abstract semantics of r is defined to contain all the tuples w'' such that for some instantiated tuple $w' \in I_r$:

- the values of the thematic and temporal attributes of w'' and w' are the same,
- w'' has two spatial attributes x and y with values equal to the (x, y)-coordinates of a point within the triangle described by w'.

Query languages for the parametric 2-spaghetti data model are under development. Clearly, temporal selection and projection, spatial selection and projection, and union can be easily defined. However, there is a serious difficulty with defining the join operator because the tuples in this model are not closed with respect to intersection.

Lemma 2.1 The intersection of two parametric 2-spaghetti relations cannot always be represented as a parametric 2-spaghetti relation.

Proof: Let's take the simple case with two input relations R_1 and R_2, each containing a single parametric 2-spaghetti tuple. Let's suppose that R_1 contains the tuple $(3, 3, 10, 10, 3, 11 - t, 9, 10)$ and R_2 contains the tuple $(3, 3, 10, 10, 5, t - 8, 9, 10)$. Therefore, at any instant $t \in [9, 10]$, R_1 describes a triangle ABC and R_2 describes a triangle ACD with corner points $A = (3, 3), B = (3, 11 - t), C = (10, 10)$, and $D = (5, t - 8)$. At any time $t \in [9, 10]$ let the intersection of the triangles ABC and ACD be the triangle ACE. Figure 4 shows the triangles at $t = 9$.

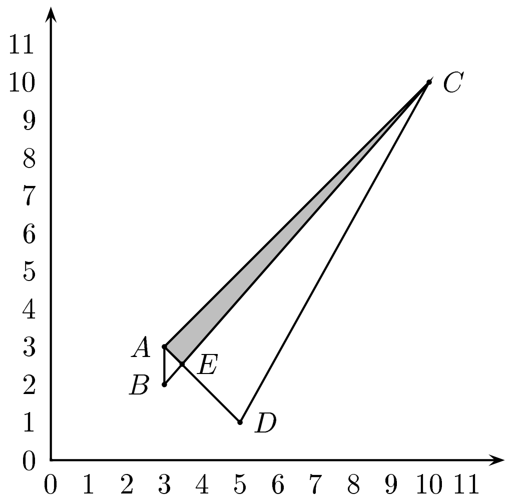

Figure 4: Non-closure under intersection

In order to represent triangle ACE as a parametric tuple, we need to get the coordinate values of the point E. Note that E is the intersection of two straight lines AD and BC. From the points $B = (3, 11 - t)$ and $C = (10, 10)$, we get the linear equation of the line BC, which is: $7y = x(t - 1) + 10(8 - t)$. Similarly, from the points $D = (5, t - 8)$ and $A = (3, 3)$, we get the linear equation of the line AD, which is: $-2y = x(11 - t) + (3t - 39)$. By solving the system consisting of both equations, we obtain

$$x = \frac{113 - t}{75 - 5t}.$$

Thus x cannot be represented as a linear function of t. In fact, the set of E-vertices is described by the quadratic curve:

$$10x^2 - 5xy - 66x + y + 150 = 0.$$

Therefore, in this case to express the intersection of the two relations, we need an infinite number of triangles ACE such that E lies on this quadratic curve. Since the parametric 2-spaghetti model allows only a finite number of tuples, the intersection of R_1 and R_2 cannot be represented within the parametric 2-spaghetti model. $\square$

3 Data Expressiveness

3.1 Constraint Databases

Definition 3.1 [20] Let Φ be the set of atomic constraints of some constraint theory. A *generalized k-tuple* over variables $x_1, \ldots, x_k$ is of the form:

$$r(x_1, \ldots, x_k) :\!\!- \phi_1 \wedge \ldots \wedge \phi_n$$

where r is a relation symbol, and $\phi_i \in \Phi$ for $1 \leq i \leq n$ and uses only the variables $x_1, \ldots, x_k$. A *generalized relation r with arity k* is a finite set of generalized k-tuples with symbol r on left hand side. A *generalized database* is a finite set of generalized relations. $\square$

Definition 3.2 [20]. Let D be the domain over which variables are interpreted. Then the *model of a generalized k-tuple t* with variables $x_1, \ldots, x_k$ is a k-ary relation consisting of all tuples $(a_1, \ldots, a_k) \in D^k$ such that the substitution of a_i for x_i satisfies the body of t.
Such a model may be infinite, e.g., the model of the generalized tuple $x < y$. The *model of a generalized relation* is the union of the models of its generalized tuples.
The *model of a generalized database* is the set of the models of its generalized relations. $\square$

We consider the following classes of constraints:

- linear arithmetic constraints of the form $a_1x_1 + \ldots a_kx_k\theta b$ where a_i and b are rational constants and x_i are variables on rational numbers (θ is one of $=, <, >, \leq, \geq$);
- order constraints $t_1\theta t_2$ over integers where t_1 and t_2 are variables or constants (θ is one of $<, >, \leq, \geq$);
- equality constraints $t_1 = t_2$ over various domains where t_1 and t_2 are variables or constants.

We also consider restricted forms of constraints:

- unary order constraints of the form $x\theta c$ where x is a variable, c a constant, and θ one of $<, >, \leq, \geq$);

- unary equality constraints of the form $x = c$ where x is a variable and c is a constant.

The models of generalized relations are relations. Therefore, in principle relational query languages like relational calculus, relational algebra, Datalog with or without negation can all be applied to constraint databases. The challenge is how to provide appropriate query evaluation methods that handle finite representations in the form of generalized tuples. For example, projection needs to be implemented as quantifier elimination. This problem is addressed in [20] and many subsequent papers.

3.2 Constraint Basis

We use the constraint database framework to characterize data expressiveness of data models.

Definition 3.3 A *constraint basis* of a data model Δ is a class C of generalized databases such that every database in $E_\Delta = \{\theta_\Delta(D) : D \in I(\Delta)\}$ is a model of some generalized database in C and vice versa.

In the following let $\alpha_1(x_1), \ldots, \alpha_n(x_n)$ be unary equality constraints over *thematic* (nonspatial and nontemporal) variables $x_1, \ldots, x_n$, $\phi(t)$ a conjunction of unary order constraints over a *temporal* variable t, and $\gamma(x, y)$ a conjunction of linear arithmetic constraints over *spatial* variables x and y. We assume that the planar object described by $\gamma(x, y)$ is closed and bounded.

Proposition 3.1 The class of generalized databases with relations whose tuples are of the form

$$\alpha_1(x_1) \wedge \cdots \wedge \alpha_n(x_n) \wedge \phi(t)$$

is a constraint basis of the TQuel data model.

Example 3.1 We can represent the relation in Table 2 as the following generalized relation:

$$\begin{array}{lll}
researcher(name, comp, t) & :\!\!- & name = "Anderson", comp = "AT\&T", \\
 & & 1980 \leq t, t \leq 1993. \\
researcher(name, comp, t) & :\!\!- & name = "Brown", comp = "IBM", \\
 & & 1985 \leq t, t \leq 1992. \\
researcher(name, comp, t) & :\!\!- & name = "Brown", comp = "IBM", \\
 & & 1990 \leq t, t \leq 1996. \\
researcher(name, comp, t) & :\!\!- & name = "Clark", comp = "Lotus", \\
 & & 1990 \leq t, t \leq 1991.
\end{array}$$

Note that here two generalized tuples represent Brown's employment at IBM, while in the corresponding TQuel relation there was just one tuple with related information. In general the sets described by different generalized tuples need not be disjoint. For instance, different generalized tuples can represent information coming from different sources (this is common if multiple databases are interoperated).

Proposition 3.2 [35] The class of generalized databases with relations whose tuples are of the form

$$\alpha_1(id) \wedge \gamma(x, y)$$

is a constraint basis of the 2-spaghetti data model (note our restriction to one thematic attribute).

Because $\gamma(x, y)$ is a conjunction of linear arithmetic constraints, the planar object described by $\gamma(x, y)$ is a finite union of convex polygons. Additionally, it is supposed to be closed and bounded. Both conditions are first-order definable with linear constraints, so the above constraint basis is effectively recognizable.

Proposition 3.3 Boundedness and closedness are first-order definable with linear constraints.

Proof (sketch): Boundedness means that the object is entirely contained within some nonempty rectangle whose sides parallel the axes. Interior, exterior and boundary are easily defined using rectangle-sized neighborhoods. For instance, a point is an interior point of an object if there is a nonempty rectangle-sized neighborhood of this point which is entirely contained within the object. Similarly the exterior. The boundary consists of the points that are neither interior nor exterior. Closedness means that every boundary point of an object belongs to the object itself. □

Example 3.2 For the 2-spaghetti model example, we can represent the abstract semantics by the following constraint database:

$$\begin{array}{lcl}
object(id, x, y) & :\!\!- & id = p1, x = 10, y = 4. \\
object(id, x, y) & :\!\!- & id = l1, 5 \leq x, x \leq 9, y = -x + 15. \\
object(id, x, y) & :\!\!- & id = l1, x = 9, 3 \leq y, y \leq 6. \\
object(id, x, y) & :\!\!- & id = t1, 2 \leq x, x \leq 6, 3 \leq y, y \leq 7, y \leq -x + 9. \\
object(id, x, y) & :\!\!- & id = r1, 1 \leq x, x \leq 11.5, 2 \leq y \leq 11. \\
object(id, x, y) & :\!\!- & id = p2, x \geq 3, y \geq 5, y \geq x - 1, y \leq x + 5, y \leq -x + 13.
\end{array}$$

Corollary 3.1 The class of generalized databases with relations whose tuples are of the form

$$\alpha_1(id) \wedge \phi(t) \wedge \gamma(x, y)$$

is a constraint basis of the Worboys spatiotemporal data model. (This means that in such relations the temporal attribute t and the spatial attributes x and y are *syntactically independent* in the sense of [6].)

Example 3.3 We can represent the previous example of Worboys' model as the following generalized database.

$$\begin{array}{lcl}
object(id,x,y,t) & :\!\!- & id = p1, x = 10, y = 4, 1980 \le t, t \le 1986. \\
object(id,x,y,t) & :\!\!- & id = l1, 5 \le x, x \le 9, y = -x + 15, 1995 \le t, t \le 1996. \\
object(id,x,y,t) & :\!\!- & id = l1, x = 9, 3 \le y, y \le 6, 1995 \le t, t \le 1996. \\
object(id,x,y,t) & :\!\!- & id = t1, 2 \le x, x \le 6, y \le -x + 9, 3 \le y, y \le 7, 1975 \le t, t \le 1990. \\
object(id,x,y,t) & :\!\!- & id = r1, 1 \le x, x \le 11.5, 2 \le y, y \le 11, 1994 \le t, t \le 1996. \\
object(id,x,y,t) & :\!\!- & id = p2, x \ge 3, y \ge 5, y \ge x - 1, y \le x + 5, y \le -x + 13, 1991 \le t, \\
 & & t \le 1996.
\end{array}$$

For the parametric 2-spaghetti model we have only one-way containment.

Theorem 3.1 Every generalized database consisting of relations whose tuples are of the form

$$\alpha_1(id) \wedge \psi(x,y,t)$$

where $\psi(x,y,t)$ is a conjunction of linear arithmetic constraints can be represented as a finite parametric relation in the parametric 2-spaghetti model provided for each rational constant t_0, the formula $\psi(x,y,t_0)$ describes a closed bounded polygon. This condition can again be defined as a first-order query with linear arithmetic constraints.

Proof: We sketch the construction of a parametric relation equivalent to the given generalized relation in the above form. We show how to define a parametric tuple corresponding to a finite set of generalized tuples of the form $\psi(x,y,t)$. Such a set represents a polyhedral set P in three dimensions: x, y and t. This set may be unbounded but only in the dimension t.

First, determine the extreme points and the faces of P. Let $t_1,\ldots,t_k$ be the time coordinates of the extreme points of P, sorted in ascending order. For simplicity, we assume they are all different. (Equality is a degenerate case and can be handled similarly. Also, $-\infty$ and $+\infty$ have to be handled in a special way – one cannot speak about edges anymore.)

Repeat the following for every interval $I = [t_i, t_{i+1}]$. Denote by P_I the slice of P that contains all the points of P whose time coordinates are in $[t_i, t_{i+1}]$. P_I is a polyhedral set. Also, all its vertices have t-coordinates equal to t_i or t_{i+1} (by construction). Call the first kind *low* and the second kind *high*.

Determine the edges of P_I, and among those select the edges that connect a low vertex with a high one and lie entirely within P_I (other edges are uninteresting). Each such edge is a segment of an edge of P and thus also a segment of a line which is an intersection of two faces of P. Consequently, it can be described as a system of two linear equations in x, y, and t (the faces are described by single linear equations). From this system obtain the equation relating x and t and the one relating y and t. This gives a parametric formulation for the x- and y-coordinates of the vertices of the finite union of convex polygons G_I described by $\psi(x,y,t)$ for every $t \in I$.

Triangulate G_I and produce a parametric tuple for every obtained triangle. This tuple contains the parametric x- and y-coordinates of the triangle vertices (described above) and the interval I.□

The containment in the other direction does not hold, as shown by the following example.

Example 3.4 Consider the following parametric object with a single tuple (with the same spatial attribute values as in the 4th tuple in Table 5).

id	x	y	x'	y'	x''	y''	*From*	*To*
r1	3	3	10	10	3	11-t	$-\infty$	$+\infty$

This is a parametric triangle whose edge between the points $(3, 11-t)$ and $(10, 10)$ has a slope that depends on t. This edge is contained in a line described by the equation:

$$7y = x(t-1) + 10(8-t).$$

The triangle cannot be specified using linear arithmetic constraints over x, y and t.

In many cases, however, a parametric object can be represented using linear arithmetic constraints.

Example 3.5 To represent Table 5, which is an instance of the parametric 2-spaghetti data model, note that flooded area is always described by the constraint $y \geq x + 8 - t$:

$$flooded(id, x, y, t) \quad :\!\!- \quad id = r1, 1 \leq y, y \leq 10, 3 \leq x, x \leq 10, y \geq x + 8 - t.$$

We conjecture that as long as a relation in the parametric representation contains only objects whose boundaries involve only a finite number of slopes, there is a corresponding representation using linear arithmetic constraints. We think that the conjecture holds not only for triangular representations but also for representations involving rectangles, or higher degree convex polygons. In general, it is an open problem to find and precisely characterize relational representations of spatiotemporal constraint databases.

4 Data Translation Using Constraint Queries

4.1 Constraint Wrapper

The notion of constraint basis defined in the previous section characterizes the *semantics* of a data model. Here we provide a constraint counterpart to the specific *syntax* of relation instances in this model.

Definition 4.1 A *constraint wrapper* for a data model Δ is a syntactically defined class C of generalized databases such that there is a simple correspondence between databases in $I(\Delta)$ and generalized relations in C.

By "simple correspondence" we mean that it is easy to construct the generalized relation in a wrapper of Δ from the instances in $I(\Delta)$ and vice versa. We are intentionally vague here, because we want to allow a broad class of wrappers. There may be more than one constraint wrapper for a data model. Relations corresponding to a constraint wrapper of Δ may have a different arity than those corresponding to a constraint basis of Δ. For example, a constraint wrapper for TQuel consists of generalized relations whose elements are tuples over n data and two temporal variables t_1 and t_2. The temporal constraints in every such tuple are equality constraints of the form $t_i = c$ only.

Now the data translation between two data models Δ_1 and Δ_2 *such that the data expressiveness of* Δ_1 *is contained in or equal to the data expressiveness of* Δ_2 is a composition of translations shown in Figure 5.

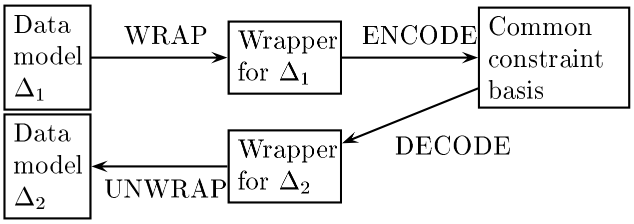

Figure 5: Composition of translations

The WRAP/UNWRAP translations are outside the scope of the constraint database technology, because they are data-model-dependent. The ENCODE/DECODE translations are queries and may be expressible using constraint query languages. We expect many data models to share a common constraint basis. The interoperability of such data models is greatly simplified: instead of constructing $n(n-1)$ direct data translations between every pair of data models, it is enough to construct the WRAP/UNWRAP translations for every model, and ENCODE/DECODE for every constraint wrapper (at most $4n$ translations).

4.2 TQuel

A constraint wrapper for TQuel consists of constraint databases whose relations contain generalized tuples of the form

$$x_1 = c_1 \wedge \cdots \wedge x_n = c_n \wedge t_1 = a \wedge t_2 = b$$

where a and b are integers, $+\infty$ or $-\infty$. A constraint basis for TQuel consists of constraint databases whose relations contain generalized tuples of the form

$$x_1 = c_1 \wedge \cdots \wedge x_n = c_n \wedge \phi(t)$$

where $\phi(t)$ is a conjunction of order constraints over the integers.

Theorem 4.1 The ENCODE/DECODE translations for TQuel can be expressed as first-order constraint queries with order and equality constraints.

Proof: The ENCODE translation has to handle the difference of arities ($n+2$ versus. $n+1$) and eliminate $+\infty$ and $-\infty$. Let P be a generalized relation of the constraint wrapper. We define the corresponding generalized relation of the constraint basis using a relational calculus query $\gamma_P(\bar{x}, t)$ defined as

$$\exists t_1, t_2.(P(\bar{x}, t_1, t_2) \wedge \alpha(t_1, t_2, t))$$

where $\alpha(t_1, t_2, t)$ is

$$\begin{aligned}(t_1 \neq -\infty \ \wedge \ t_2 \neq +\infty \ \wedge \ t_1 \leq t \leq t_2 \vee t_1 \neq -\infty \ \wedge \ t_2 = +\infty \ \wedge \ t_1 \leq t \vee \\ t_1 = -\infty \ \wedge \ t_2 \neq +\infty \ \wedge \ t \leq t_2 \vee t_1 = -\infty \ \wedge \ t_2 = +\infty).\end{aligned}$$

The DECODE translation is more complicated, as it involves coalescing tuples with the same nontemporal components (and nondisjoint intervals) and generating $+\infty$ and $-\infty$ where appropriate. Let R be a generalized relation of the constraint basis. We define the corresponding generalized relation of the constraint wrapper using a (domain) relational calculus query $\beta_R(\bar{x}, t_1, t_2)$ defined as

$$\begin{aligned}\beta_1(\bar{x}, t_1, t_2) \vee \beta_2(\bar{x}, t_2) \ \wedge \ t_1 = -\infty \vee \beta_3(\bar{x}, t_1) \ \wedge \ t_2 = +\infty \vee \\ \beta_4(\bar{x}) \ \wedge \ t_1 = -\infty \ \wedge \ t_2 = +\infty\end{aligned}$$

where the query $\beta_1(\bar{x}, t_1, t_2)$ specifies tuples with bounded intervals:

$$\begin{aligned}\forall t.(t_1 \leq t \leq t_2 \Rightarrow R(\bar{x}, t) \ \wedge \ \exists t_3.(t_3 < t_1 \ \wedge \ \forall t_0.(t_3 \leq t_0 < t_1 \Rightarrow \neg R(\bar{x}, t_0))) \ \wedge \\ \exists t_4.(t_2 < t_4 \ \wedge \ \forall t_0.(t_2 < t_0 \leq t_4 \Rightarrow \neg R(\bar{x}, t_0)))).\end{aligned}$$

The query $\beta_2(\bar{x}, t_2)$ specifies tuples with intervals unbounded to the left, $\beta_3(\bar{x}, t_1)$ tuples with intervals unbounded to the right, and $\beta_4(\bar{x})$ tuples unbounded on both sides. These queries can be defined similarly to $\beta_1(\bar{x}, t_1, t_2)$. $\square$

4.3 2-spaghetti

Lemma 4.1 Assume that a finite union of closed convex polygons is represented as a generalized relation $object(i, x, y)$ with linear arithmetic constraints over x and y (i is the object identifier). The following relations can be defined as first-order queries with linear arithmetic constraints:

- $boundary(i, x, y) \equiv$ the point (x, y) is on the boundary of i,
- $vertex(i, x, y) \equiv$ the point (x, y) is a vertex of i,
- $edge(i, x, y, x', y') \equiv$ the points (x, y) and (x', y') form an edge of the boundary of i.

Proof: In the proof we use rectangles whose sides are parallel to the x- or y-axis. We represent them as quadruples (x, y, x', y') where (x, y) is the bottom left corner and (x', y') is the top right corner.

(*Boundary.*) For the object i each point on the boundary is a point of this object that is not an inside point of i. A point is an inside point of i iff it is inside a rectangle wholly contained within i (see Figure 6):

$$\begin{array}{lcl} inside(i,x,y) & \equiv & object(i,x,y) \land \exists x', y', x'', y''.(x' < x < x'') \land (y' < y < y'') \land \\ & & \forall x''', y'''.((x' < x''' < x'') \land (y' < y''' < y'') \Rightarrow object(i, x''', y''')) \\ boundary(i,x,y) & \equiv & object(i,x,y) \land \neg inside(i,x,y). \end{array}$$

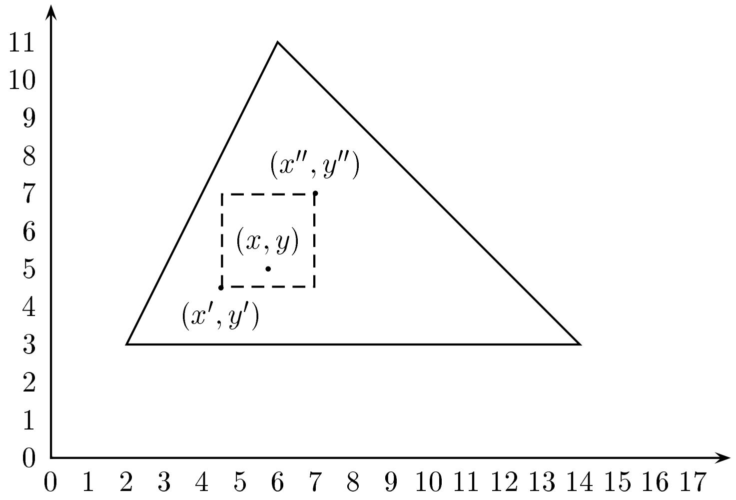

Figure 6: Point inside polygon

(*Vertex and edge.*) We define two auxiliary relations:

- $iso(i, x, y, x', y', z, w, z', w') \equiv$ translating the fragment of the object i contained in the rectangle (x, y, x', y') by the vector $(z - x, w - y)$ gives the fragment of the object i contained in the rectangle (z, w, z', w') (see Figure 7).

$$\begin{aligned} iso(i, x, y, x', y', z, w, z', w') \quad \equiv \quad & z' = x' + z - x \wedge w' = y' + w - y \wedge \\ & \forall s. \forall t. x \leq s \leq x' \wedge y \leq t \leq y' \Rightarrow \\ & (object(i, s, t) \Leftrightarrow object(i, s + z - x, t + w - y)). \end{aligned}$$

- $nc(i, x, y) \equiv (x, y)$ is a boundary point of i but not a vertex of i. The consecutive occurences of the relation iso in the formula correspond to positive slopes, negative slopes, slopes equal to 0, and slopes equal to ∞. Figure 8 shows the case of a positive slope.

$$\begin{aligned} nc(i, x, y) \quad \equiv \quad & boundary(i, x, y) \wedge \exists c > 0. \exists d > 0. \\ & boundary(i, x + c, y + d) \wedge boundary(i, x - c, y - d) \wedge \\ & (iso(i, x - c, y - d, x, y, x, y, x + c, y + d) \vee \\ & iso(i, x, y - d, x + c, y, x - c, y, x, y + d) \vee \\ & iso(i, x - c, y - d, x, y + d, x, y - d, x + c, y + d) \vee \\ & iso(i, x - c, y - d, x + c, y, x - c, y, x + c, y + d)). \end{aligned}$$

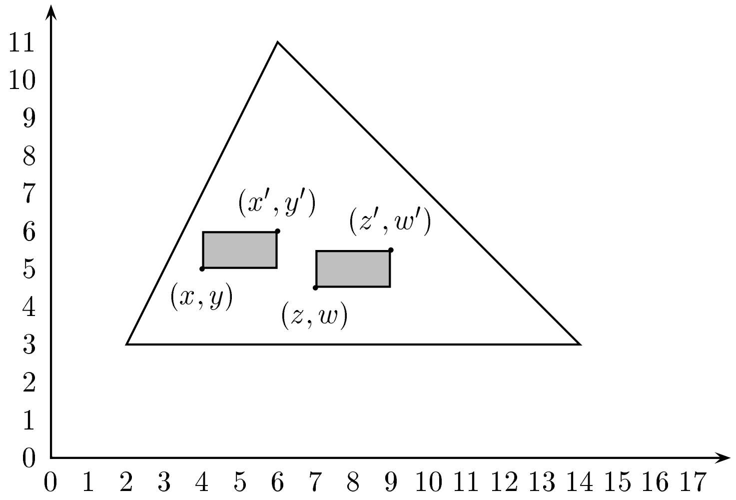

Figure 7: The *iso* relation

Now

$$vertex(i,x,y) \equiv boundary(i,x,y) \wedge \neg nc(i,x,y).$$

Considering again all possible slopes

$$\begin{aligned} edge(i,x,y,x',y') \equiv\ & vertex(i,x,y) \wedge vertex(i,x',y') \wedge \\ & ((x \neq x' \wedge y \neq y' \wedge pos\_slope(i,x,y,x',y')) \vee \\ & (x \neq x' \wedge y \neq y' \wedge neg\_slope(i,x,y,x',y')) \vee \\ & y = y' \wedge horizontal(i,x,y,x',y') \vee \\ & x = x' \wedge vertical(i,x,y,x',y')). \end{aligned}$$

We define only *pos_slope*; the definitions of the remaining predicates are similar. Figure 9 demonstrates how edge is determined for positive slopes. The essential idea behind this definition consists of checking the existence of infinitely many rectangles containing identical (up to a translation) fragments of the object and spanning the object boundary.

$$\begin{aligned} pos\_slope(i,x,y,x',y') \equiv\ & \forall c_1 > 0.\forall d_1 > 0.\exists c_2 > 0.\exists d_2 > 0.\exists s.\exists t.\exists s'.\exists t'. \\ & boundary(i,s,t) \wedge boundary(i,s+c_2,t+d_2) \wedge \\ & x < s \wedge s + c_2 < x + c_1 \wedge y < t \wedge t + d_2 < y + d_1 \wedge \\ & s' + c_2 < x' \wedge s < s' \wedge x' - c_1 < s' \wedge \\ & t' + d_2 < y' \wedge t < t' \wedge y' - d_1 < t' \wedge \\ & \forall g.\forall h.\ 0 < g \leq s' - s \wedge 0 < h \leq t' - t \Rightarrow \\ & iso(i,s,t,s+c_2,t+d_2,s+g,t+h,s+c_2+g,t+d_2+h). \end{aligned}$$

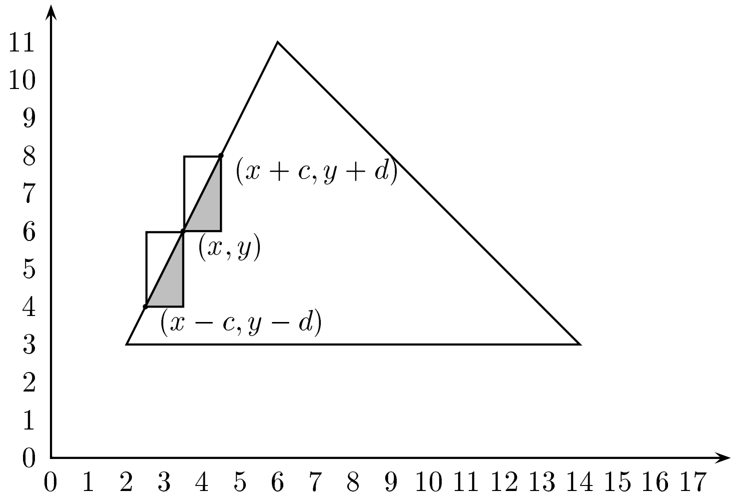

Figure 8: The *nc* relation

This concludes the proof of Lemma 4.1.□

A constraint wrapper for 2-spaghetti consists of databases with relations whose tuples are of the form

$$id = i_0 \wedge x = a \wedge y = b \wedge x' = a' \wedge y' = b' \wedge x'' = a'' \wedge y'' = b''.$$

Theorem 4.2 The DECODE translation for 2-Spaghetti can be expressed as a first-order constraint query with linear arithmetic constraints.

Proof. The constraint basis of the 2-spaghetti model (Proposition 3.2) guarantees that the spatial objects are finite unions of convex polygons. The construction in Lemma 4.1 gives in this case the vertices and the edges of the object. The triangulation (necessary for our 2-spaghetti representation) requires more work. For convex polygons, a triangulation can be easily described in a first-order way by picking an arbitrary vertex v_0 (e.g., the least vertex in the lexicographic ordering of all vertices) and constructing all nondegenerate triangles (v_0, v_1, v_2) such that (v_1, v_2) is an edge. However, for finite unions of such polygons the situation is more complicated. We can view such unions as consisting of finitely many components that are polygons, possibly nonconvex and containing holes. Different components intersect only in finitely many points. We settle for a triangulation in which the triangles are not necessarily disjoint (but they still cover the entire object). First, we determine the *candidate* edges of the triangulation. They have either to be *external* edges as defined by the *edge* predicate in Lemma 4.1 or *internal* ones (contained wholly within the interior of the object). Internal edges can be defined similarly to external ones, except that instead of requiring that the infinitely many rectangles span the boundary we require that they are

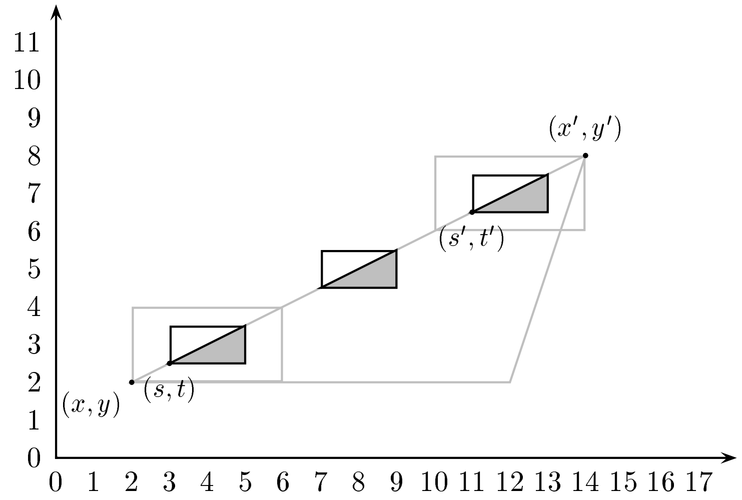

Figure 9: The *edge* relation

wholly contained within the object. Edges that are not external or internal have at least one point that does not belong to the object and are thus useless for triangulation. Second, we construct a superset of the triangulation by taking all triangles with vertices that are vertices of the object and edges that are candidate edges. Such triangles can still contain holes inside. Thus in the third step we check for each triangle obtained whether it has a point which does not belong to the object. If such a point p exists in a triangle, the horizontal line drawn through it intersects two different edges of the triangle. The point p lies then *between* the intersection points. All the above conditions can be defined as first-order formulas with linear arithmetic constraints. $\square$

Remark: The DECODE translation does not necessarily have to be implemented as the constraint query described above. It would be easier to implement the translation by considering each generalized tuple separately. The vertices of the convex polygon corresponding to the tuple can then be constructed using a variety of methods, e.g., linear programming.

Theorem 4.3 The ENCODE translation for 2-spaghetti can be expressed as a first-order constraint query with polynomial inequality constraints.

Proof: The translation of points and line segments is straightforward. Let us look now at the translation of triangles. Let us assume that the vertices of a nondegenerate triangle are $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3). We want to formulate the condition that a point (x, y) is on the same side of the line segment defined by the points (x_1, y_1) and (x_2, y_2) as the third vertex (x_3, y_3) of the triangle (and similarly for the remaining sides). To do this we determine the equation $y = ax + b$ of the line going through the points (x_1, y_1) and (x_2, y_2)

and then formulate the condition as

$$y > ax + b \equiv y_3 > x_3 + b.$$

This can also be expressed as

$$(y_3 - y_1)(x_2 - x_1) > (y_2 - y_1)(x_3 - x_1) \equiv (y - y_1)(x_2 - x_1) > (y_2 - y_1)(x - x_1).$$

Hence let us define

$$\begin{array}{rcl} \mathit{side}(x, y, x_1, y_1, x_2, y_2, x_3, y_3) & \equiv & (((y_3 - y_1)(x_2 - x_1) \geq (y_2 - y_1)(x_3 - x_1) \wedge \\ & & (y - y_1)(x_2 - x_1) \geq (y_2 - y_1)(x - x_1) \vee \\ & & ((y_3 - y_1)(x_2 - x_1) \leq (y_2 - y_1)(x_3 - x_1) \wedge \\ & & (y - y_1)(x_2 - x_1) \leq (y_2 - y_1)(x - x_1))) \end{array}$$

Now each triangle can be translated as

$$\begin{array}{ll} \exists x_1, y_1, x_2, y_2, x_3, y_3. & \mathit{triangle}(i, x_1, y_1, x_2.y_2, x_3, y_3) \wedge \\ & (x_1 \neq x_2 \vee y_1 \neq y_2) \wedge \\ & (x_1 \neq x_3 \vee y_1 \neq y_3) \wedge (x_3 \neq x_2 \vee y_3 \neq y_2) \wedge \\ & \mathit{side}(x, y, x_1, y_1, x_2, y_2, x_3, y_3) \wedge \\ & \mathit{side}(x, y, x_2, y_2, x_3, y_3, x_1, y_1) \wedge \\ & \mathit{side}(x, y, x_3, y_3, x_1, y_1, x_2, y_2) \end{array}$$

The translation query contains quadratic constraints. However, the generalized relation resulting from translating any 2-spaghetti relation will contain only linear constraints. This is because the variables $x_1, x_2, x_3, y_1, y_2, y_3$ will be all replaced by constants coming from the 2-spaghetti relation being translated. □

Theorem 4.4 The DECODE translation for Worboys' spatiotemporal model can be expressed as a first-order constraint query with linear arithmetic constraints. The ENCODE translation for Worboys' spatiotemporal model can be expressed as a first-order constraint query with polynomial arithmetic constraints.

Proof: Consider DECODE first. We need to define maximal intervals $[t_1, t_2]$ such that

$$\mathit{object}(i, x, y, t_1) \wedge \mathit{object}(i, x, y, t_2) \wedge \forall t \in [t_1, t_2].\mathit{object}(i, x, y, t).$$

Maximality is clearly first-order definable using order constraints (see the DECODE mapping for TQuel, Theorem 4.1). For all the elements of each such interval, the corresponding spatial object is the same. So for example it is described by $\mathit{object}(i, x, y, t_1)$. On the basis of this object, one can then define vertices, edges and a triangulation as in Theorem 4.2 (all the predicates will have one extra argument for time). The fact that only finitely many maximal intervals are obtained is guaranteed by the form of the constraint basis of the Worboys model.

The ENCODE construction is a combination of Theorems 4.1 and 4.3. The specific form of the constraint basis for the Worboys model guarantees the independence of spatial and temporal attributes. This allows us to translate the temporal and the spatial attributes separately. □

Theorem 4.5 The DECODE translation for the parametric 2-spaghetti can be expressed as a first-order constraint query with linear arithmetic constraints.

Proof: The method used in the proof of Theorem 3.1 can be expressed as a first-order query with linear arithmetic constraints. First, extreme points and faces of polyhedral sets in three dimensions can be defined analogously to vertices and edges in finite unions in polygons (Lemma 4.1). Second, consecutive intervals of t-coordinates of extreme points can also be defined in a first-order way. Finally, using those definitions one can define slices and their triangulation, as in Theorem 4.2. □

Theorem 4.6 The ENCODE translation within the proof of Theorem 4.3 for the parametric 2-spaghetti can be expressed as a first-order query with polynomial constraints. Moreover, the ENCODE translation can be expressed as a first-order query with *linear* arithmetic constraints if and only if in each parametric 2-spaghetti tuple of the form:

id	x_1	y_1	x_2	y_2	x_3	y_3	*From*	*To*
i_1	a_1t+b_1	a_2t+b_2	a_3t+b_3	a_4t+b_4	a_5t+b_5	a_6t+b_6	t_1	t_2

one of the following conditions holds:

- $a_1 = a_2 = a_3 = a_4 = a_5 = a_6 = 0$, or
- $((b_1 = b_3$ and $b_2 = b_4))$ or $(a_4 - a_2)(b_3 - b_1) = (b_4 - b_2)(a_3 - a_1)$ and $((b_3 = b_5$ and $b_4 = b_6))$ or $(a_6 - a_4)(b_5 - b_3) = (b_6 - b_4)(a_5 - a_3)$ and $((b_1 = b_5$ and $b_2 = b_6))$ or $(a_6 - a_2)(b_5 - b_1) = (b_6 - b_2)(a_5 - a_1)$.

Before giving the proof we sketch the intuition behind the above conditions. The conditions require that the triangles corresponding to different values of t be *self-similar* (see Figure 10). That is, if any side of a triangle grows from t_1 to t_2 by some constant multiplicative factor c, then so do the remaining sides of this triangle. Notice that we ignore the *From* and *To* attributes, as they are irrelevant for the second part of the theorem.

Proof: It is easy to see that the translation results in a first-order query with polynomial constraints. We have to show both directions of the claim for linear constraints.

(If direction:) Suppose that the condition holds. Then the translation within the proof of Theorem 4.3 will result in the following (where $\theta_1, \theta_2, \theta_3$ are either $\leq$ or $\geq$):

$(y-y_1)(x_2-x_1)\theta_1(y_2-y_1)(x-x_1)$
$\vee(y-y_2)(x_3-x_2)\theta_2(y_3-y_2)(x-x_2)$
$\vee(y-y_3)(x_1-x_3)\theta_3(y_1-y_3)(x-x_3)$

Substituting into the first disjunction the components of the parametric 2-spaghetti tuple and assuming $b_2 = b_4$ and $b_1 = b_3$, the above simplifies to

$$(y - a_2 t - b_2)(a_3 - a_1)\theta_1(x - a_1 t - b_1)(a_4 - a_2)$$

which is a linear constraint in x, y and t.

Alternatively, substituting when $(a_4 - a_2)(b_3 - b_1) = (b_4 - b_2)(a_3 - a_1)$ and simplifying we get:

$$(y-a_2t-b_2)((b_3-b_1)[(a_4-a_2)t+(b_4-b_2)])\theta_1(x-a_1t-b_1)((b_4-b_2)[(a_4-a_2)t+(b_4-b_2)]))$$

When $(a_4 - a_2)t + (b_4 - b_2)$ is equal to zero, then we do get the constraint $0\theta_1 0$, which is linear. When it is non-zero, then dividing by it we get:

$$(y - a_2 t - b_2)(b_3 - b_1)\theta_1(x - a_1 t - b_1)(b_4 - b_2)$$

or the same with θ_1 reversed depending whether the value divided by is positive or negative. In both cases we obtain a linear constraint.

Hence if ($b_1 = b_3$ and $b_2 = b_4$) or $(a_4 - a_2)(b_3 - b_1) = (b_4 - b_2)(a_3 - a_1)$, then the first disjunction contains only linear constraints. A similar analysis applies to the remaining cases.

(Only if direction:) Let's consider again Example 3.4. There we have noted that the equation obtained cannot be represented by linear constraints using x, y and t. We also find that $a_6 = -1$ and $b_4 = 10 \neq b_6 = 11$ and $(a_6 - a_4)(b_5 - b_3) = 7 \neq (b_6 - b_4)(a_5 - a_3) = 0$. Therefore, the condition of the theorem fails. $\square$

Example 4.1 As an example let's consider the following parametric 2-spaghetti tuple which satisfies the condition of the second part of Theorem Thus, its encoding can be determined by a query with linear constraints only. Figure 10 shows the self-similar triangles obtained from this tuple for $t = 0$ and $t = 1$.

id	x_1	y_1	x_2	y_2	x_3	y_3	*From*	*To*
i_1	1	t+2	2t+3	1	1	1	0	1

5 Query Enhancement

We show here how constraint databases and constraint query languages can be used to enhance the expressive power of an existing query language. The starting point is the Worboys' spatiotemporal data model. The query language of this model is somewhat limited.

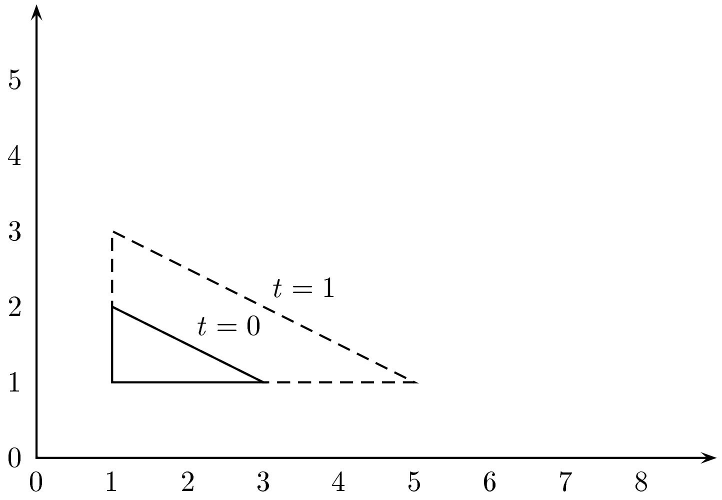

Figure 10: Self-similar triangles

It cannot express non-equijoins. Moreover, the operators cannot produce relations whose abstract semantics involve more than one temporal or more than two spatial dimensions. For instance, the following spatiotemporal queries are not expressible in this language:

- *did John own any piece of land before Paul?*
- *were there two different time instants when John owned the same land?*

On the other hand, the above queries can be easily expressed in relational calculus or algebra (see below). However, relational query languages cannot be directly evaluated on Worboys relations. Such relations need first to be translated to generalized relations with linear arithmetic constraints, as shown in Section 4.

But then a difficulty appears: how to guarantee that the *result* of a relational calculus or algebra query, which is a generalized relation, can be mapped back to a Worboys relation, so that the user deals only with a single data model? The answer consists of several parts. First, one needs the characterize what it means for a generalized relation R to correspond to a Worboys relation. This is the case if the set of spatial attributes of R is *independent*, in the sense of [6], of the set of nonspatial attributes of R (see Figure 11). Second, one needs to define a *safe* subset of relational algebra (or calculus) consisting of operators that preserve that kind of independence. It turns out that the safe subset contains all the operators of relational algebra and the only restriction is that a selection condition cannot mix spatial and nonspatial attributes. However, to prove the safety of this set one has to extend the original set of inference rules for variable independence given in [6]. The modified set of rules is presented in [7].

Worboys model **Linear arithmetic constraint databases**

$$
\begin{tikzcd}
D \arrow[r, "\text{attrs independent}"'] \arrow[d, dashed, "\text{no query}"'] & C_D \arrow[d, "\text{Query } Q"] \\
D' & Q(C_D) \arrow[l, "\text{if attrs independent}"]
\end{tikzcd}
$$

Figure 11: Enhancing Worboys' Query Language

The query

did John own any piece of land before Paul?

can be expressed in relational calculus as:

$$\exists t_1.\exists t_2.\exists x.\exists y.own(John, x, y, t_1) \wedge own(Paul, x, y, t_2) \wedge t_1 < t_2.$$

This query can be translated to relational algebra in a standard way. The resulting algebraic expression has to contain a nonequijoin or a subexpression producing a relation with two temporal dimensions.

The query

were there two different time instants when John owned the same land?

can also be expressed in relational calculus, as:

$$\exists t_1.\exists t_2.\forall x.\forall y.t_1 \neq t_2 \wedge own(John, x, y, t_1) \Leftrightarrow own(John, x, y, t_2).$$

The relational algebra formulation of this query requires a subexpression producing a relation with two temporal dimensions.

6 Related Work

Interoperability between a GIS database and application programs was studied in [30]. Interoperability in the sense of combining spatial and thematic data managers for the implementation of GIS systems was studied in [22]. In this paper, we are addressing a different issue, namely *data interoperability* among various temporal and spatial data models.

Data interoperability of different temporal data models was studied in [18]. The solution proposed there, the provision of a single unifying temporal data model to which other models could be mapped, is not sufficient. This still falls short of defining a *representation-independent* abstract semantics for temporal databases. In fact, the model of [18] uses another notion of concrete temporal database, admittedly simpler and more general than others. This model is still limited in its data expressiveness as it is capable of representing only finite databases. We also think that the model is unnecessarily complicated because it introduces a new notion of "bitemporal element". Moreover, translations between this model and other temporal data models are expressed using an ad-hoc procedural language.

Recent work on the interoperability of temporal databases, e.g., Wang, Jajodia and Subrahmanian [36, 5], addresses similar concerns as the present paper. However, the paper [36] does not address spatial or spatiotemporal database issues and makes very strong assumptions about the concrete temporal databases that are to be interoperated. In particular, such databases have to provide a unified interface. This is not necessary in our approach. Moreover, the data expressiveness of the cited model is limited to sets of finite databases. A follow-up work [5] demonstrates a systematic approach of deriving implicit temporal information from the explicit information stored in a temporal database. Such derivations could very well be incorporated into our framework. For surveys of temporal query languages, see [8, 11].

Spatiotemporal data models and query languages are a topic of growing interest. The paper [37], discussed in section [13] the authors talk about moving points and regions but formally define only the former. In their approach some kinds of continuous change can be modeled using linear interpolation functions. Query language issues are not addressed. In [16] the authors propose a formal spatiotemporal data model based on constraints in which, like in [37], only discrete change can be modeled. An SQL-based query language is also presented. Finally, [10] proposes a general framework for modeling spatiotemporal objects supporting continuous change. None of the approaches considers, however, the issue of database interoperability.

Among the many recent papers on constraint databases we focus on those that are directly relevant to the topic of this paper. [35] brings the first systematic study of linear constraints in spatial database applications and [27] an important classification of spatial query languages in constraint databases. [12] contains, as a corollary, a first-order definition of the polygon wireframe (vertices and edges). This result was obtained independently of the first version of this paper [9]. Moreover, as opposed to [12], our construction is elementary and has a clear geometric intuition. [21] and [15] describe spatial DBMS based on constraints. Other language proposals in this area include [3, 4]. [6, 7] introduces the notion of variable independence constraints and studies their theoretical properties. In the context of spatiotemporal databases, this notion captures discrete change. Further work in this direction includes [17].

7 Conclusions and Future Work

We have presented a novel way to apply the constraint database technology to temporal, spatial and spatiotemporal database applications. Constraint databases are used as a layer mediating between different data models. Constraint query languages are used to formulate the translations between the models.

Implementation. The research reported in this paper is just a first step in the direction of making spatiotemporal databases interoperable. We have developed efficient algorithms for the translations developed here and implemented them. This will be described in a forthcoming paper.

Applications. In addition to the presented scenarios for database interoperability, there remain others to be explored. For example, we have discovered that the parametric 2-spaghetti data model is well suited for the *animation* of spatiotemporal databases because the construction of the explicit geometric representation of each snapshot is very easy.

Generalizations. Based on the framework proposed here, we plan to investigate a broader range of data models, as well as the issue of query interoperability. The spaghetti model can be extended to higher spatial dimensions [38]. In higher dimensions each K-dimensional object is represented as a set of $K-1$-dimensional facets. For representing this containment we need to have a separate table describing the object containment hierarchy. We believe that our techniques generalize to arbitrary fixed spatial (or temporal) dimensions.

Extensibility. Application-dependent interoperability issues of resolving semantic and representational mismatches and conflict detection and resolution have found an elegant formulation using the language of first-order logic [28]. Thus, solutions to these problems can be seamlessly integrated with the techniques we propose for the translation between different data models. The problem of incomplete information can also be addressed in the same framework.

Acknowledgement: An early version of some of the results in this paper appeared in [9]. The work of the first author was supported by NSF grant IRI-9632870. The work of the second author was supported by NSF grants IRI-9632871 and IRI-9625055, and by a Gallup Research Professorship.

References

[1] M. Baudinet, J. Chomicki, and P. Wolper. Temporal Deductive Databases. In A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*, chapter 13, pages 294–320. Benjamin/Cummings, 1993.

[2] M. Baudinet, M. Niézette, and P. Wolper. "On the Representation of Infinite Temporal Data and Queries". In *ACM Symposium on Principles of Database Systems*, 1991.

[3] A. Belussi, E. Bertino, and B. Catania. "Manipulating Spatial Data in Constraint Databases". In M. Scholl and A. Voisard, editors, *International Symposium on Large Spatial Databases*, pages 116–141, Berlin, Germany, 1997. Springer-Verlag, LNCS 1262.

[4] A. Brodsky and Y. Kornatzky. "The LyriC Language: Constraining Objects". In *ACM SIGMOD International Conference on Management of Data*, San Jose, California, 1995.

[5] C. Bettini, X. S. Wang, and S. Jajodia. "Temporal semantic assumptions and their use in databases". *IEEE Transactions on Knowledge and Data Engineering*, 10(2):277–296, 1998.

[6] J. Chomicki, D. Goldin, and G. Kuper. "Variable Independence and Aggregation Closure". In *ACM Symposium on Principles of Database Systems*, Montreal, Canada, June 1996.

[7] J. Chomicki, D. Goldin, and G. Kuper. "Variable Independence in Constraint Databases". In preparation, 1998.

[8] J. Chomicki. "Temporal Query Languages: A Survey". In D.M. Gabbay and H.J. Ohlbach, editors, *Temporal Logic, First International Conference*, pages 506–534. Springer-Verlag, LNAI 827, 1994.

[9] J. Chomicki and P. Z. Revesz. "Constraint-Based Interoperability of Spatiotemporal Databases". In *International Symposium on Large Spatial Databases*, pages 142–161, Berlin, Germany, July 1997.

[10] J. Chomicki and P. Z. Revesz. "A Geometric Framework for Specifying Spatiotemporal Objects". In *International Workshop on Temporal Representation and Reasoning (TIME'99)*, Orlando, Florida, May 1999.

[11] J. Chomicki and D. Toman. "Temporal Logic in Information Systems". In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 3. Kluwer Academic Publishers, Boston, 1998.

[12] F. Dumortier, M. Gyssens, L. Vandeurzen, and D. Van Gucht. "On The Decidability of Semi-Linearity for Semi-Algebraic Sets and Its Implications for Spatial Databases". In *ACM Symposium on Principles of Database Systems*, pages 68–79, Tucson, Arizona, May 1997.

[13] M. Erwig, R. H. Güting, M. M. Schneider, and M. Vazirgiannis. "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases". In *ACM Symposium on Geographic Information Systems*, November 1998.

[14] A. U. Frank, I. Campari, and U. Formentini. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space.* Springer-Verlag, LNCS 639, 1992.

[15] S. Grumbach, P. Rigaux, and L. Segoufin. "The DEDALE System for Complex Spatial Queries". In *ACM SIGMOD International Conference on Management of Data*, pages 213–224, June 1998.

[16] S. Grumbach, P. Rigaux, and L. Segoufin. "Spatio-Temporal Data Handling with Constraints". In *ACM Symposium on Geographic Information Systems*, November 1998.

[17] S. Grumbach, P. Rigaux, and L. Segoufin. "On the Orthographic Dimension of Constraint Databases". In *International Conference on Database Theory*, January 1999. Also INRIA Verso Report 141, 1998.

[18] C. S. Jensen, M. D. Soo, and R. T. Snodgrass. "Unifying Temporal Data Models via a Conceptual Model". *Information Systems*, 19(7):513–547, 1994.

[19] W. Kim, I. Choi, S. Gala, and M. Scheevel. "On Resolving Semantic Heterogeneity in Multidatabase Systems". In W. Kim, editor, *Modern Database Systems*, pages 521–550. Addison-Wesley, 1995.

[20] P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. "Constraint Query Languages". *Journal of Computer and System Sciences*, 51(1):26–52, August 1995.

[21] P. Kanjamala, P.Z. Revesz, and Y. Wang. "MLPQ/GIS: A Geographic Information System using Linear Constraint Databases". In *9th COMAD International Conference on Management of Data*, pages 389–393, Hyderabad, India, December 1998. Tata McGraw Hill.

[22] C. P. Kolovson, M-A. Neimat, and S. Potamianos. "Interoperability of Spatial and Attribute Data Managers: A Case Study". *Proc. Symp. on Spatial Databases*, 239-264, Singapore, June 1993.

[23] A. Kemper and M. Wallrath. "An Analysis of Geometric Modeling in Database Systems". *ACM Computer Surveys*, 19(1), 1987.

[24] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems.* Academic Press, 1992.

[25] J. Paredaens. "Spatial Databases, The Final Frontier". In *International Conference on Database Theory*, pages 14–32, Prague, Czech Republic, January 1995. Springer-Verlag, LNCS 893.

[26] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[27] J. Paredaens, J. Van den Bussche, and D. Van Gucht. "Towards a Theory of Spatial Database Queries". In *ACM Symposium on Principles of Database Systems*, pages 279–288, Minneapolis, Minnesota, 1994.

[28] X. Qian and T. F. Lunt. "Semantic Interoperation: A Query Mediation Approach". Technical Report SRI-CSL-94-02, Computer Science Laboratory, SRI International, April 1994.

[29] A. Rosenthal and L. J. Seligman. "Data Integration in the Large: The Challenge of Reuse". In *International Conference on Very Large Data Bases*, pages 669–675, 1994.

[30] H. J. Schek, and A. Wolf. "From Extensible Databases to Interoperability between Multiple Databases and GIS Applications". *Proc. Symp. on Spatial Databases*, 207-238, Singapore, June 1993.

[31] R. T. Snodgrass. “The Temporal Query Language TQuel”. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[32] R. T. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

[33] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.

[34] D. Toman. “Point vs. Interval-based Query Languages for Temporal Databases”. In *ACM Symposium on Principles of Database Systems*, pages 58–67, Montréal, Canada, June 1996.

[35] L. Vandeurzen, M. Gyssens, and D. Van Gucht. “On the Desirability and Limitations of Linear Spatial Database Models”. In *International Symposium on Large Spatial Databases*, pages 14–28, 1995.

[36] X. S. Wang, S. Jajodia, and V. S. Subrahmanian. “Temporal Modules: An Approach Toward Federated Temporal Databases”. In *ACM SIGMOD International Conference on Management of Data*, pages 227–236, 1993.

[37] M. F. Worboys. “A Unified Model for Spatial and Temporal Information”. *Computer Journal*, 37(1):26–34, 1994.

[38] M. F. Worboys. *GIS: A Computing Perspective*. Taylor&Francis, 1995.