Journal of Universal Computer Science, vol. 16, no. 20 (2010), 2986-3015
submitted: 7/1/10, accepted: 13/9/10, appeared: 1/11/10 © J.UCS

A Geometrically Enhanced Conceptual Model and Query Language

Hui Ma
(School of Engineering and Computer Science
Victoria University of Wellington, New Zealand
hui.ma@ecs.vuw.ac.nz)

Abstract: Motivated by our experiences with spatial modelling for the sustainable land use initiative we present a geometrically enhanced ER model (GERM), which preserves the key principles of entity-relationship modelling and at the same time introduces bulk constructors and geometric features. The model distinguishes between a syntactic level of types and an explicit internal level, in which types give rise to polyhedra that are defined by algebraic varieties. It further emphasises the stability of algebraic operations by means of a natural modelling algebra that extends the usual Boolean operations on point sets.

Key Words: conceptual model, geometric model, spatial model, entity-relationship model, natural modelling algebra, query language

Category: E.m[Data]:Miscellaneous, H.1.10[Information Systems]:Models and Principles

1 Introduction

The goal of our research is to provide a conceptual model supporting geometric modelling. One motivation is the need for spatial data modelling in the context of the sustainable land use initiative (SLUI) [Mackay, 2007], which addresses erosion problems in the New Zealand hill country. At the core of SLUI whole farm plans (WFPs) are produced, which capture farm boundaries, paddocks, etc. and provide information about land use capability (LUC) such as rock, soil, slope, erosion, vegetation, plants, poles, etc. Such plans can then be used to get an overview of erosion and vegetation levels and water quality, and to exploit this information for sustainable land use management.

A number of spatial data models have been proposed in the literature [Guting and Schneider, 1995; Laurini and Thompson, 1992; Peano, 1890], for a brief survey see Section 2. The existing models focus on how to store data and how to efficiently realise storage and operations on such data. However, the purpose of a conceptual model is to serve as communication tool between domain experts and information analysts, that abstracts from implementation-level details of data storage and data access. Based on our experiences with the modelling of WFPs we are convinced that more research has to be done to obtain adequate conceptual models supporting geometric modelling. Our objective is to go even further than geographic information systems, and to support other kinds of geometric modelling in the same manner. For instance, technical constructions such as rotary piston engines can be supported by trochoids, which are plane algebraic curves already known by the Greeks [Brieskorn and Knörrer, 1981]. Bézier curves and patches [Salomon, 2005] are also commonly applied in these applications. Together with hull operators [Hartwig, 1996] they can also be used for 3-D models of hill shapes in WFPs.

In this paper we introduce the geometrically enhanced entity-relationship model (called GERM for short) as our approach to deal with the problems discussed. As the name suggests, our intent is to preserve the aggregation-based approach [Hull and King, 1987] of the original entity-relationship (ER) model by means of higher-level relationship types [Thalheim, 2000], but we enhance roles in relationship types by supporting choice and bulk constructors (sets, lists, multisets). However, different from [Hartmann and Link, 2007] the bulk constructors are not used to create first-class objects, neither is the choice constructor (defining so-called clusters in [Thalheim, 2000]).

Furthermore, we keep the fundamental distinction between data types such as points, polygons, Bézier curves, etc. and concepts. The former ones are used to define the domains of (nested) attributes, while the latter ones are represented by entity and relationship types. That is, a concept such as a paddock is distinguished from the curve defining its boundary. In this way we guarantee a smooth integration with non-geometric data such as farm ownership, processing and legal information that is also relevant for WFPs, but does not pose a novel modelling challenge.

GERM has been developed to support modelling at multiple levels. On a syntactic (or surface) level we provide an extendible collection of data types such as line sequences, polygons, sequences of Bézier curves, or Bézier patches, that have efficient surface representations. For example, a polygon can be represented by a list of points, and a Bézier curve of order n can be represented by $n + 1$ points – the case $n = 2$ captures the most commonly known quadratic Bézier curves that are also supported in LaTeX.

On an explicit (or internal) level we use a representation by polyhedra [Hartwig, 1996] that are defined by algebraic varieties, i.e., sets of zeros of polynomials in n variables. All curves that have a rational parametric representation such as Bézier curves [Salomon, 2005], can be brought into such an "implicit" form, e.g. [Gao and Chou, 1992] describes a method for implicitisation based on Gröbner bases, and many classical curves that have proven their values in landcare for centuries can be represented in this way [Brieskorn and Knörrer, 1981]. This kind of explicit representation bears some similarities to the polynomial model of spatial data introduced in [Paredaens and Kuijpers, 1998].

Adequate support for operations on geometric data, such as interiors, unions, or intersections of point sets, defines a third, derived level. To guarantee the geometric stability of these operations we adopt the idea of natural modelling [Hartwig, 1996] and extend it to the data types provided in GERM.

This paper is organised as follows. In Section 2 we give a brief review of related work in the literature. Section 3 assembles some relevant information on WFPs to set up the context of the study. In Section 4 we introduce the constituents of GERM emphasising the syntactic level. Section 5 outlines an SQL-based language for expressing queries against GERM conceptual schemas. In Section 6 we discuss the internal representation of geometric data by means of algebraic varieties. We illustrate our approach by examples from the modelling of WFPs. Finally, in Section 7 we introduce a natural modelling algebra, and discuss its merits with respect to expressiveness and accuracy.

2 Related Work

While there is a lot of sophisticated mathematics around to address geometric modelling in landcare, and this has a very long tradition as shown in [Brieskorn and Knörrer, 1981], spatial and geometric modelling within conceptual modelling have mainly followed two lines of research – for an overview see [Shekhar and Xiong, 2008]. The first one is based on modelling spatial relationships such as disjointness, touching, overlap, inside, boundary overlap, etc. and functions such as intersection, union, etc. that are used for spatial primitives such as points, lines, polygons, or regions. Extensions have been proposed to the ER model and to UML class diagrams to support spatial data modelling [Shekhar et al., 1997]. However, spatial data types are not considered at the conceptual level. Thanasis et al. [Hadzilacos and Tryfona, 1997] propose the Geo-ER Model that introduces special entity sets, relationships, and new constructs to handle properties associated to objects relating to the objects' position. In [Shekhar et al., 1999] pictograms are added to the common ER model to highlight spatial objects and relationships. Price et al. [Price et al., 2001] deal in particular with part-whole relationships, Ishikawa et al. [Ishikawa and Kitagawa, 2001] apply constraint logic programming to deal with these predicates, Gubiani et al. [Gubiani and Montanari, 2008] propose an approach to support the management of consistency constraints on multiple representations of the same spatial entity, Malinowski et al. [Malinowski and Zimányi, 2007] use spatial integrity constraints to ensure the semantic equivalence of the conceptual and logical schemas. McKenny et al. [McKenny and Schneider, 2007] handle problems with collections, and Chen et al. [Chen and Zaniolo, 2000] use the predicates in an extension of SQL. In [Shekhar et al., 1999], a set of rules are presented to generate spatial columns and data types in SQL DDL from entity pictograms. However, the complexity of rules are not discussed and can be very high.

The work in [Behr and Schneider, 2001; Chen and Zaniolo, 2000] links to the second line of research expressing the spatial relationships by formulae defined on point sets applying basic Euclidean geometry or standard linear algebra, respectively. Likewise, point sets are used in [Stoffel et al., 2007] to express predicates on mesh of polygons in order to capture motion, and Frank [Frank, 2005] classifies spatial algebra operation into local, focal and zonal ones based on whether only values of the same location, of a location and its immediate neighbourhood, or of all locations in a zone, respectively, are combined. Christian et al. [Jensen et al., 2004] extend existing algebraic query language to accommodate spatial values that exhibit partial containment relationships.

Paredaens and Kuijpers [Paredaens, 1995; Paredaens and Kuijpers, 1998] compare five spatial data models: the raster model [Guting and Schneider, 1995] and the Peano model [Peano, 1890], which represent spatial data by finite point sets that are either uniformly or non-uniformly distributed over the plane, respectively, the Spaghetti model [Laurini and Thompson, 1992] based on contours defined as polylines, the polynomial model [Kanellakis et al., 1990; Paredaens et al., 1994] based on formulae that involve equality and inequality of polynomials, and the PLA model [Corbett, 1979], which only uses some kind of topological information without dealing with exact position and shape. While some models lack theoretical foundations, those that are grounded in theory do not bother about efficient implementations.

As we can see from above, spatial models discussed in the literature are not suitable for conceptual modelling of geometric data to provide an abstract view of the an application domain. Widely used conceptual models like the entity-relationship model and UML do not capture some important semantics inherent in spatial modelling [Shekhar et al., 1999]. Geometric data should be included into conceptual models so that it can be conveniently accessed through the surface level, while its internal structure is hidden from the user. In the mean time, the conceptual model for geometric data should be closed under geometric set operations [Schneider, 2009]. Further, the spatial relationships and functions discussed in the literature are in fact derived from underlying representations of point sets, so we need representations on multiple levels as also proposed in [Balley et al., 2004]. Furthermore, when dealing with point sets it is not sufficient to define spatial relationships and functions in a logical way. We also have to ensure "good nature" in the numerical sense, i.e., the operations must be as accurate as possible when realised using floating point arithmetics. For example, [Liu et al., 2005] emphasised several spatial conflicts such as determining the accurate spatial relationship for a winding road along a winding river as opposed to a road crossing a river several times, leading to a classification of line-line relationships. The accuracy problem has motivated a series of modifications to algebras on point sets that enhance the standard Boolean operators [Hartwig, 1996] to make them fit for practical applications of geometric modelling.

3 Whole Farm Plans

As we mentioned this study is motivated by the need for spatial data modelling in the context of the sustainable land use initiative (SLUI). Now let us look in more detail at the information that should be kept for the SLUI program. To manage land in the hill country sustainably, the aim of the SLUI and Whole Farm Plans (WFPs) is to propagate land use change. For the long term the SLUI program has an objective to have 75,000 ha of land improved in the next 10 years. To demonstrate the considerable funds being spent are achieving worthwhile outcomes, WFPs should be monitored to measure the progress towards the objective. For example WFP data, including spatial and non-special data, need to be kept to create WFPs and to analyse the effectiveness of the programme, e.g., the upgraded land area.

During the environmental assessment of farms a series of artifacts is generated, including legal titles and parcels maps, paddock maps, land resource inventories, Land Use Capability (LUC) maps, soil fertility and nutrient maps, pasture production maps, summaries of resource issues, and recommendations for the sustainable management of land resources. We are not aiming to illuminate every detail of this process, but just want to exemplify the kinds of spatial and non-spatial data that need to be captured in Whole Farm Plans. In doing so we shed some light on the features a conceptual modelling language needs to observe to be useful for WFP modelling, and for geometric modelling in general.

Paddock maps: Figure 1 contains an example of paddock maps which shows the boundary of paddocks within a farm. To model the maps we need to consider spatial data as well as non-special data, e.g. an area with paddock code = 'P01' is defined by its paddock boundary which is a polygon. That is, we need a conceptual model to include *Polygon* and *STRING* as abstract data types for paddock boundaries and paddock codes.

LUC maps: Land resources can be described and evaluated according to LUC classification. The LUC classification has three basic components class, subclass and capability unit. The area under study is divided into landcare units by drawing boundaries around areas with similar soil characteristics. A landcare unit polygon represents an area of similar soil characteristics. LUC groups similar landcare units according to their capacity for sustainable production under arable, pastoral, forestry or conservation uses. The LUC unit (e.g. $3w2$) indicates general capability (classes 1 to 8), the major limitations (4 subclass limitations of **w**etness, **e**rosion, **s**oil and **c**limate) and the capability unit (e.g. 2) to link with regional classifications and known best management practices. Figure 2 shows an example of LUC maps. Again to model the information on LUC maps, we need to consider spatial data, e.g., landcare unit boundaries, and non-spatial data, e.g. LUC units. For example, an area on the bottom of the map is defined by a polygon and has a value of $3w2$ for the LUC unit. For the same value of

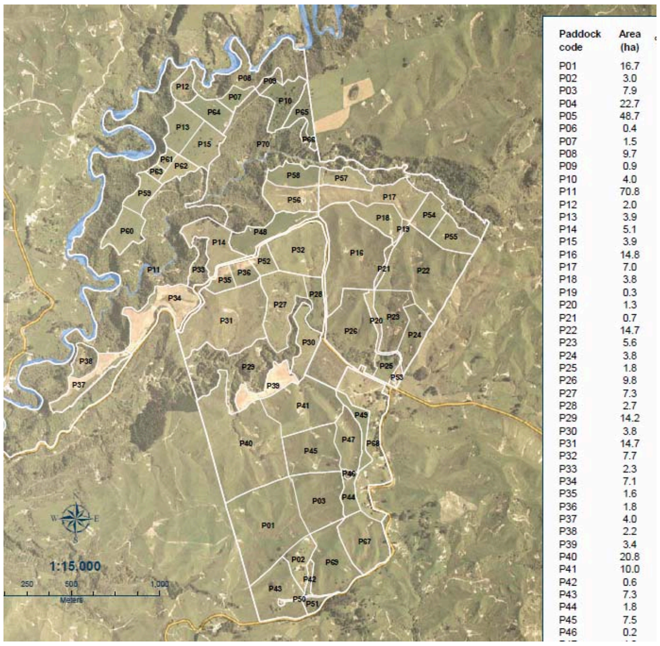

Figure 1: *An Example of a Paddock Map [AgResearch, 2005]*

LUC unit total area can be calculated, e.g. the total area of $3w2$ is 20.4 hectares. To demonstrate the improvement of land use capability, LUC maps produced at different years can be compared.

WFP maps: Based on the assessment of land strengths and weaknesses are given by LUC unit, a catalogue of environmental works is recommended. Recommendations for either land use change or management change are based primarily on the degree, severity, and location of the soil erosion and the resultant effects. A set of maps are created to show the locations of planned work. For example, as shown in Figure 3, the location of tree planting or fence to be built are shown on the maps. To model the data on WFP maps, we need to model the location information of the planned work, presented as polygons, points or arcs, as well as additional information of the planned work which includes the financial support information provided by the regional council.

As we can see from above, WFP data includes classic relational or non-spatial data (attribute data) such as owner information, investment information, land usability data and spatial data (location data) such as paddock boundaries, LUC landcare unit boundaries, buildings, land and water resources, bridges, tracks, track crossings, enhancement plantings, fences, stone pickings. There is a need to integrate the spatial and non-spatial data in the SLUI application so that users

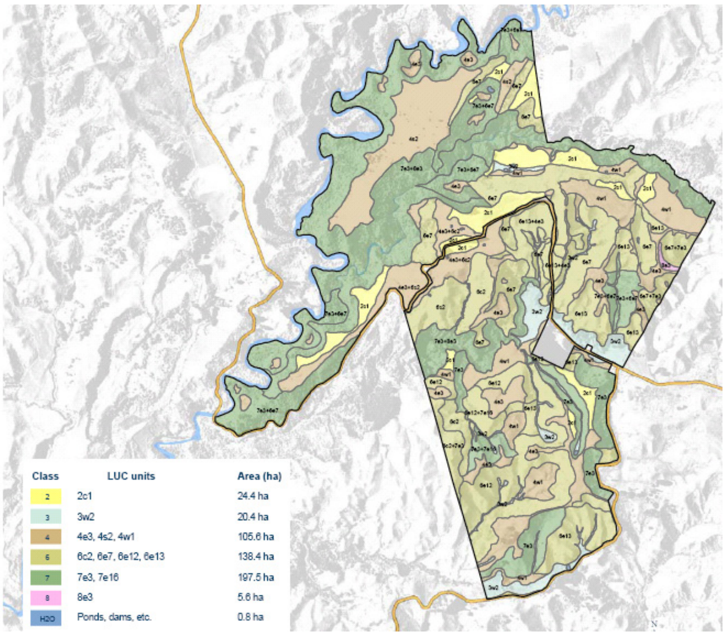

Figure 2: *An Example of a LUC Map [AgResearch, 2005]*

can analyse the effectiveness of the SLUI programme, e.g. by calculating the areas of landcare units that have some LUC unit values improved. Our goal is to support conceptual modelling of WFPs for integration into the SLUI information system by providing representations of the spatial relationships and functions on multiple levels, by providing relationships and functions in a logical way when dealing with point sets, and by providing suitable operations when realised using floating point arithmetic.

4 Geometrically Enhanced ER Model (GERM)

In this section we start with the presentation of GERM focussing on the syntactic (or surface) level, which is what will be needed first for modelling geometrically enhanced applications. We will concentrate on the definition of data and object types and their semantics, but we will dispense with discussing primary keys or other constraints. For attributes we will permit structuring.

4.1 Data Types and Nested Attributes

Definition 1. A *universe* is a countable set $\mathcal{U}$ of simple attributes together with a type assignment tp that assigns to each attribute $A \in \mathcal{U}$ a data type $tp(A)$. $\square$

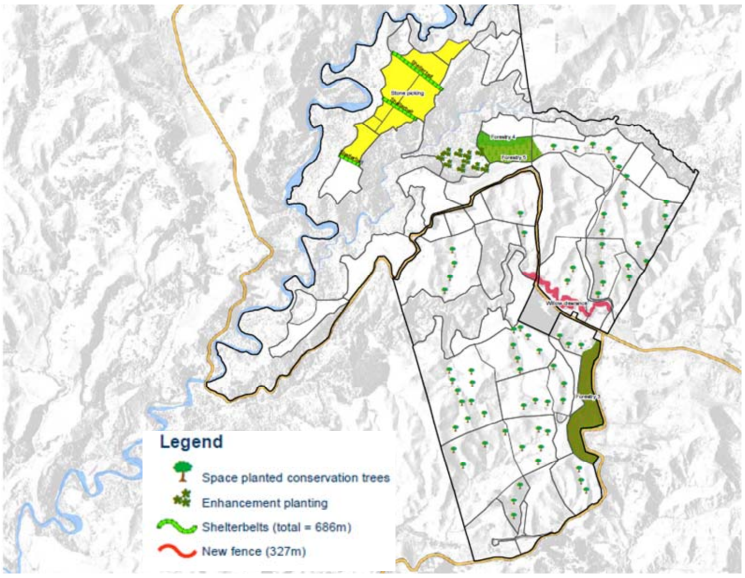

Figure 3: *An Example of a WFP Map [AgResearch, 2005]*

For most purposes, the data type $tp(A)$ assigned to an attribute $A \in \mathfrak{U}$ will be a *base type*, but we do not enforce such a restriction. We do not further specify the collection of base types. Common examples of base types are INT, $FLOAT$, $STRING$, $DATE$, or $TIME$. Each base type t is associated with a countable set of values $dom(t)$ called the *domain* of t. For the base types mentioned here the domains are chosen as usual. For an attribute $A \in \mathfrak{U}$ we let $dom(A) = dom(tp(A))$, and also call $dom(A)$ the *domain* of A.

We use constructors to build complex data types from base types. In particular, we use $(\cdot)$ for record types, $\{\cdot\}$, $[\cdot]$ and $\langle\cdot\rangle$ for finite set, list and multiset types, respectively, $\oplus$ for (disjoint) union types, and $\rightarrow$ for map types. Furthermore, we make use of a trivial type $\mathbb{1}$ with domain $dom(\mathbb{1}) = \{\perp\}$. We obtain *complex data types* t by the production rule

$$t = \mathbb{1} \mid b \mid (a_1 : t_1, \dots, a_n : t_n) \mid (a_1 : t_1) \oplus \cdots \oplus (a_n : t_n) \mid \{t\} \mid [t] \mid \langle t \rangle \mid t_1 \rightarrow t_2$$

where b stands for base types, and $a_1, \dots, a_n$ are pairwise distinct labels chosen from a suitable alphabet. Moreover, we allow complex data types to be named and used in type definitions in the same way as base types with the restriction that cycles are forbidden. Domains associated with complex data types can be obtained in a similar way from the values of the respective base types and $\perp$.

Example 1. We may define named complex data types that can be used for geometric modelling, such as $Point = (x : FLOAT, y : FLOAT)$ for points in the two-dimensional plane, $PolyLine = [Point]$, $Polygon = [Point]$, $Bezier = [Point]$, or $PolyBezier = [Bezier]$. Note, that the examples $PolyLine$, $Polygon$, and $Bezier$ constitute types with identical surface representations, but different geometric semantics (details will be discussed in Section 6). A *polyline* is defined defined piecewise linearly, while a *polygon* is a region with a polyline border. A *Bézier curve* is determined by a finite sequence of points in the plane, too. It passes through the first and last control points and lies within the convex hull of the control points. A *polyBézier curve* is defined piecewise by Bézier curves. We will commonly refer to such data types as *geometric data types*, thus emphasising that they will serve as surface representations for particular geometric features.

The trivial type $\mathbb{1}$ can be used in combination with the union constructor to define *enumerated types*, i.e., types with finite domains, such as $Bool = (\mathbf{T} : \mathbb{1}) \oplus (\mathbf{F} : \mathbb{1})$, $Gender = (\text{male} : \mathbb{1}) \oplus (\text{female} : \mathbb{1})$, or $INT_n = (1 : \mathbb{1}) \oplus \cdots \oplus (n : \mathbb{1})$ for any positive integer n, which gives a domain with values $1, \ldots, n$. The map constructor can be used to define *array types*, such as $Patch = (i : INT_n, j : INT_m) \to Point$ representing Bézier patches. Further examples used for spatial modelling are *vector field types* of different dimensions, such as $\mathit{Vectorfield1} = \{Point\} \to FLOAT$, which is useful for capturing sensor data (e.g., water levels), and $\mathit{Vectorfield2} = \{Point\} \to Point$, which is useful for modelling other measurements (e.g., wind force and direction) by two-dimensional vectors. Finally, $TimeSeries = (d : DATE, t : TIME) \to \mathit{Vectorfield1}$ is useful for modelling a series of observed data over time, thus capturing also temporal aspects of data.

Similar to base types, we may also consider attributes with complex data types assigned to them. Such attributes are known as *nested attributes*, cf. [Thalheim, 2000]. We extend this notion as follows:

Definition 2. The set $\mathcal{A}$ of *(nested) attributes* (over universe $\mathcal{U}$) is the smallest set with $\mathcal{U} \subseteq \mathcal{A}$ satisfying $X(A_1, \ldots, A_n), X\{A\}, X[A], X\langle A\rangle, X_1(A_1) \oplus \cdots \oplus X_n(A_n), X(A_1 \to A_2) \in \mathcal{A}$ with labels $X, X_1, \ldots, X_n$ and $A, A_1, \ldots, A_n \in \mathcal{A}$. $\square$

The type assignment tp extends naturally from $\mathcal{U}$ to $\mathcal{A}$ as follows:

- $tp(X(A_1, \ldots, A_n)) = (a_1 : tp(A_1), \ldots, a_n : tp(A_n))$ with labels $a_1, \ldots, a_n$,
- $tp(X_1(A_1) \oplus \cdots \oplus X_n(A_n)) = (X_1 : tp(A_1)) \oplus \cdots \oplus (X_n : tp(A_n))$,
- $tp(X\{A\}) = \{tp(A)\}$, $tp(X[A]) = [tp(A)]$, $tp(X\langle A\rangle) = \langle tp(A)\rangle$, and
- $tp(X(A_1 \to A_2)) = tp(A_1) \to tp(A_2)$.

4.2 Entity and Relationship Types

Following [Thalheim, 2000] the major difference between entity and relationship types is the presence of components $\rho : O$ (with a role name ρ and a name O of an entity or relationship type) for the latter ones. We will therefore unify the definition, and simply talk of *object types* as opposed to the data types in the previous subsection. We will, however, permit object types to possess *structured components*.

Definition 3. Let $\mathcal{O}$ be a countable set of object type names. The set $\mathcal{C}$ of *component expressions* (over $\mathcal{O}$) is the smallest set containing all object type names $O \in \mathcal{O}$, all list expressions $[C]$, all set expressions $\{C_0\}$, all multiset expressions $\langle C_0 \rangle$, and all union expressions $C_1 \oplus \cdots \oplus C_n$, with $C, C_i \in \mathcal{C}$, but where the C_i are not union expressions. A *structured component* is a pair $\rho : C$ with a *role name* ρ and a component expression $C \in \mathcal{C}$. $\square$

Note that this definition does neither permit record and map constructors in component expressions, nor full orthogonality for set, multiset and union constructors. We excluded the record constructor as it corresponds to *aggregation*, i.e. whenever a component of a relationship type has the structure of a record, it can be regarded as a relationship type for its own sake. We decided to exclude the map constructor since functions on entities and relationships that depend on instances seemed to make very little sense to us. The reason the restricted combinations of the other constructors are the intrinsic equivalences observed in [Hartmann et al., 2004; Sali and Schewe, 2009]. If in $\{C\}$ we had a union component expression $C = C_1 \oplus \cdots \oplus C_n$, this would be semantically equivalent to a record expression $(\{C_1\}, \ldots, \{C_n\})$, to which the argument regarding records can be applied. The same holds for multiset expressions, while nested union constructors can be flattened. In this way we guarantee to deal only with normalised and thus simplified structured components that do not contain any hidden aggregation.

Definition 4. An *object type O of level* $k \geq 0$ consists of a finite set $comp(O) = \{\rho_1 : C_1, \ldots, \rho_n : C_n\}$ of structured components with pairwise different role names $\rho_1, \ldots, \rho_n$, and a finite set $attr(O) = \{A_1, \ldots, A_m\} \subseteq \mathcal{A}$ of nested attributes. Each object type that occurs in any of the component expressions C_i is of level at most $k-1$, and unless $comp(O) = \emptyset$ at least one of these object types must have exactly the level $k-1$.

Note that this definition enforces $comp(O) = \emptyset$ if and only if O is an object type of level 0. Therefore, we call object types of level 0 *entity types*, and object types of level $k > 0$ *relationship types*. For brevity, we use the notation $O = (comp(O), attr(O))$ to define an object type O. $\square$

Example 2. Recall that whole farm plans are developed on the basis of paddock maps. Each paddock belongs to some farm, is identified by its paddock_code, is legally defined by a boundary, and has a particular usage. We can use an object type PADDOCK with structured component $farm$: FARM and with attributes *paddock_code*, *usage*, and *boundary* to model paddocks. That is, $comp(\text{PADDOCK}) = \{farm : \text{FARM}\}$ and $attr(\text{PADDOCK}) = \{paddock\_code,\ usage,\ boundary\}$. Herein, FARM is an entity type and, thus, PADDOCK is a relationship type of level 1. For simplicity, we agree to omit the role name (e.g., farm) if it corresponds to the object type name (e.g., FARM) in the component expression.

Note that while we discarded full orthogonality for component constructors, we did not do this for the nested attributes, leaving a lot of latitude to modelers. The rationale behind this flexibility is that the attributes should reflect pieces of information that are meaningful within the application context. For instance, using a simple attribute *shape* with $tp(shape) = Polygon$ (thus, $shape \in \mathcal{U}$) indicates that the structure of polygons as lists of pairs of floating point numbers is not relevant for the conceptual model of the application, whereas the alternative of having a nested attribute $shape([point(x\text{-}coord, y\text{-}coord)])$ with $tp(x\text{-}coord) = tp(y\text{-}coord) = FLOAT$ would indicate that points and their coordinates are conceptually relevant beyond representing a data type. Notice that the use of nested attributes in object types also gives rise to a generalised notion of primary keys, which however, is beyond the scope of this paper.

Furthermore, the way we define structured components opens the way for object types with alternative or collection-type components. For example, one can define an object type LAND_PARCEL with a set of paddocks as component. However, we do not promote the use of disjoint unions nor sets, lists and multisets for first-class concepts. For example, a set of paddocks will not appear as a stand-alone object type. This differs from [Thalheim, 2000], where disjoint unions (called *clusters*) are used independently from relationship types, and also from [Hartmann and Link, 2007], where this has been extended to sets, lists and multisets. The reason is that such stand-alone constructors are hardly needed in the model, unless they appear within a component of a object type.

4.3 Schemata and Instances

Finally, we put the definitions of the previous subsections together to define schemata and their instances in the usual way.

Definition 5. A *GERM schema* $\mathcal{S}$ is a finite set of object types, such that whenever $\rho_i : C_i \in comp(O)$ is a structured component of some object type $O \in \mathcal{S}$ and the object type name Q appears in O_i, then also $Q \in \mathcal{S}$ holds. □

The definition of a GERM schema covers the syntactic level of our conceptual model. For the semantics, we need to define instances of a GERM schema. We will start with the definition of *objects*. An *interpretation* $\mathcal{I}(O)$ of an object type O is just a finite set of objects of type O. Similar to the extension of *dom* from base types to complex data types, we can also extend the interpretation to structured components. This determines a unique set $\mathcal{I}(C_i)$ of values for any component expression C_i, which is built from interpretations $\mathcal{I}(Q)$ for all Q appearing in C_i.

Definition 6. An *object o* of type O is a mapping defined on $comp(O) \cup attr(O)$ that assigns to each structured component $\rho_i : C_i \in comp(O)$ a value $c_i \in \mathcal{I}(C_i)$, and to each attribute $A_j \in attr(O)$ a value $v_j \in dom(A_j)$. Objects of an entity type (i.e., level $k = 0$) are called *entities*, while objects of a relationship type (i.e., level $k > 0$) are called *relationships*.

For brevity, we use the notation $o = (\rho_1 : c_1, \ldots, \rho_n : c_n, A_1 : v_1, \ldots, A_m : v_m)$ for an object o of type $O = (\{\rho_1 : C_1, \ldots, \rho_n : C_n\}, \{A_1, \ldots, A_m\})$. □

Definition 7. An *instance* $\mathcal{I}$ of a GERM schema $\mathcal{S}$ is an $\mathcal{S}$-indexed family $\{\mathcal{I}(O)\}_{O \in \mathcal{S}}$, such that for each object type $O \in \mathcal{S}$ the set $\mathcal{I}(O)$ is an interpretation of O, and only these sets are used to determine the value sets for structured components. □

It is common practice to visualise entity-relationship schemas as diagrams. This approach has also been used in the presence of higher-level relationships, see [Thalheim, 2000]. We will adopt this approach to our purposes here, and extend it to include a notation for sets. The *GERM diagram* of a GERM schema S is a directed graph with the object types of S as nodes, and with edges from a node O to a node Q whenever Q appears in a structured component $\rho_i : C_i \in comp(O)$. Disjoint unions are indicated by diamonds labelled with $\oplus$, while sets are indicated by diamonds labelled with $\otimes$.

Note that the GERM schema and its visualisation as a diagram are just two different ways of presenting essentially the same information. We can attach labels for attributes to the entity or relationship types to increase the level of detail. For the sake of readability, however, we have omitted them in Figure 4. For the same reason we have omitted role name labels if they correspond to the object type name in a structured component.

4.4 An Example: A GERM Schema for WFP Modelling

Example 3. Let us consider a GERM schema for a WFP as illustrated by its diagram in Figure 4. Among other it reflects geographic information related with the farms that are managed with whole farm plans. The central entity type FARM can have attributes *farm_name*, *owner*, *address*, and *boundary*. The first three

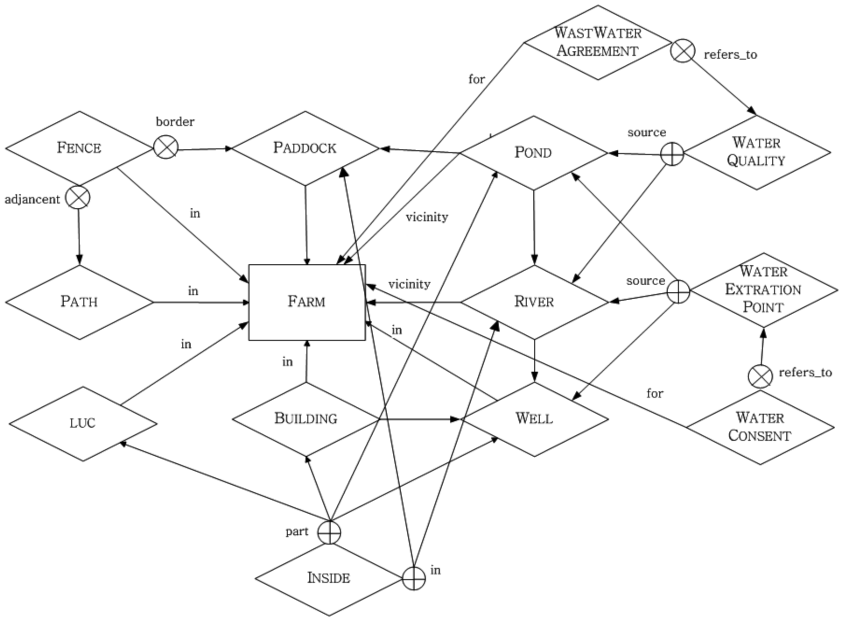

Figure 4: *Example of a GERM diagram for WFP modelling*

have the base type *STRING* assigned to them, while $tp(boundary) = PolyBezier$ is the data type assignment for the last one.

The object type PADDOCK is used to represent the arable land units of a farm (commonly called paddocks), with attributes *paddock_code*, *boundary*, and *usage*. The data types assigned to them are *STRING*, *PolyBezier*, and $(\text{cattle} : \mathbb{1}) \oplus (\text{dairy} : \mathbb{1}) \oplus (\text{hort} : \mathbb{1}) \oplus (\text{sheep} : \mathbb{1}) \oplus \cdots \oplus (\text{other} : \mathbb{1})$, respectively.

The object type LUC is used to capture the properties of landcare units of a farm, with attributes *year*, *boundary*, and *unit(class, sub_class, capability)*. We assigned the data types *YEAR* and *PolyBezier* to the first two attributes, while the third one is a nested attribute formed using a record constructor from the attributes *class* of data type *INT*, *sub_class* of data type $(\text{w} : \mathbb{1}) \oplus (\text{e} : \mathbb{1}) \oplus (\text{s} : \mathbb{1}) \oplus (\text{c} : \mathbb{1})$, and *capability* of data type *INT*. Note that the data type of *sub_class* is an enumerated type whose available values indicate **w**etness, **e**rosion, **s**oil, and **c**limate (for details please see Section 3 again).

The object type BUILDING is used to represent buildings located on a farm, with attributes *base* and *kind*. We assigned a further enumerated type to the non-spatial attribute *kind*, and the data type *Polygon* to the spatial attribute *base*. The object type FENCE has an attribute *shape* of data type *PolyLine*, and two components: a set of PADDOCK objects and a set of PATH objects, referring to the paddocks and paths it is adjacent to. The object type PATH has attributes *width*, *location*, and *shape*. The latter is of data type *PolyBezier*, indicating the course of the path by a polyBézier curve.

The object types River, Pond and Well are used to model water resources on a farm. River has attributes *river_name*, *location*, *left* and *right*. The latter two are of data type *PolyBezier*, to capture the course of the left and right bank of a river. The object type Well possesses attributes *well_number*, *depth*, and *boundary* of data types types *STRING*, *FLOAT*, and *Circle*, respectively. The object type Pond has attributes *pond_name*, and *boundary* of data types *STRING* and *PolyBezier*, respectively. As usual we may assign different data types to attributes (here *boundary*) if they appear in multiple object types.

The relationship type Inside is used to model part-of relationships that are common on a farm. For example, an island may lie in a river, and a well, pond or building may be located in a paddock. Inside has two components, one is a disjoint union of the object types representing parts, while the other is a disjoint union of the object types representing wholes.

A water consent for a farm refers to set of water extraction points, each of them linked to a water resource (called source) such as a river, a well, or a pond. The object type WaterExtractionPoint has attributes *location*, *capacity*, and *minimum*. The data types assigned to them are *Point*, *FLOAT*, and $Month \rightarrow FLOAT$, respectively. *capacity* specifies the amount of water that may be taken out of the source, while *minimum* defines a minimum season-dependent standard that may not be fallen below. The object type WaterConsent has an attribute *allowance* of data type $Month \rightarrow FLOAT$ defining the total amount of water the farm is permitted to use. The object type WaterQuality is used to model the water quality measured for a water resource (called source) such as a river or a pond over time. It has a nested attribute *measurement*$\{$*date* $\rightarrow$ *reading(agent, unit, value)*$\}$ to capture measured levels of oxygen, nitrate, or other agents. The object type WasteWaterAgreement is used to model the contracted minimum and maximum values governing the water quality for a farm.

5 GERM-SQL

Conceptual models are widely used during the analysis and design of data-intensive applications. They provide a high-level documentation of user requirements, thus facilitate the communication between data architects and domain experts during requirements capture. As design artifacts they often define a basis for further development efforts and can be subject to contracts. It is good practice to equip a conceptual modelling language with suitable means for querying conceptual models. For the entity-relationship model, such as query language is SQL/ER [Gogolla, 1994] which was created in the spirit of SQL, but omits some ingredients of classical SQL such as the Group-By-clause. [Abiteboul et al., 1995] introduced O_2SQL that extends SQL to cope with complex data. [Thalheim, 2000] built on the experiences with SQL/ER and O_2SQL when suggesting HERM/SQL for the higher-order entity-relationship model. Note, however, that geometric data types were not present in HERM, so that HERM/SQL does not provide adequate support for querying geometric data on the conceptual level.

On the other hand, there is evidence that the use of SQL for certain spatial queries is inconvenient and error-prone due to the high syntactic complexity [Lorie and Meier, 1984; Westlake and Kleinschmidt, 1990]. There have been considerable efforts to make SQL fit for querying spatial databases, for a summary see [Schneider, 2009]. Examples include Pictorial SQL [Roussopoulos et al., 1988] and Spatial SQL [Egenhofer, 1994] that extends SQL to cope with spatial data on the logical level.

In this section we are concerned with the question of how to extend HERM/SQL to cope with geometric data types. The challenge is to equip the query language with means for describing and composing geometric data that are compatible with the dual representation of geometric data on the surface and the internal level that we use for GERM. Following the extensions of SQL mentioned before, we aim to preserve the Select-From-Where structure of SQL queries. However, additions are necessary to allow the user to pose queries involving geometric features.

Every query Q has GERM schemas $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$ as input and output schema, respectively, and a mapping q that maps instances of $\mathcal{S}_{in}$ to instances of $\mathcal{S}_{out}$. Unless explicitly specified otherwise, the output schema consists of a single object type Ans. Its attributes are given in the Select-clause.

Example 4. The following query asks for the names and boundaries of Golden Kiwi's farms. Its input schema is the illustrated in Figure 4, its output schema contains the object type Ans $= (\emptyset, \{farm\_name, boundary\})$.

`SELECT` *farm_name*, *boundary*
`FROM` Farm
`WHERE` *owner* = 'Golden Kiwi';

The attributes in the Select-clause may be simple or nested attributes. They may be already existing attributes of the object types in the input schema, projections thereof to subattributes, or completely new attributes.

Example 5. To analyse the relationship between water quality and land quality, one often want to find out all the farms that have a particular river going through. To following query asks for all those Golden Kiwi farms that have the Manawatu river going through. Herein, r.vicinity:Farm.*farm_name* denotes the projection of objects of type River to an attribute of its structured component Farm. In

[Thalheim, 2000] such projections are also denoted using *navigation paths* in the entity-relationship diagram. For the formal definition of projections on complex data types using bulk and choice operators, we refer to [Hartmann et al., 2006; Sali and Schewe, 2009].

SELECT r.vicinity:FARM.*farm_name*
FROM RIVER r
WHERE r.*river_name* = 'Manawatu' and r.vicinity:FARM.*owner* = 'Golden Kiwi';

To provide adequate support for geometric data types, we need to include operators acting on the respective domains. Examples are arithmetic operators, such as *area* for the data types *Polygon* and *Circle*, *length* for the data types *PolyLine*, *Bezier* and *PolyBezier*, or *distance* between pairs of points, curves or regions. We further need unary geometric operators, such as *closure*, *interior*, and *border*, and binary geometric operators, such as *union*, *intersection*, and *difference*. A particular challenge that arises here is that the result of applying such operators does not necessarily belong to the same domain anymore. We discuss this issue in more detail in Section 7. Moreover, we may also use aggregation functions, such as *average*, *minimum*, and *sum*.

Example 6. The following query asks for the overall size of Golden Kiwi's farms.

SELECT sum(area(*boundary*)) as *total_land*
FROM FARM
WHERE *owner* = 'Golden Kiwi';

In the Where-clause we may also use Boolean operators, in particular those reflecting topological relationships between points, curves and regions, such as *contains*, *covers*, *meets*, or *disjoint*. Finally, the structure of queries may be extended by further clauses as known from classical SQL, e.g., the Group-By-clause.

Example 7. As part of the Sustainable Land Use Initiative (SLUI), the land quality of participating farms is inspected on an annual basis. The outcomes of this inspection are modelled by the object type LUC. A summary of the outcomes for a particular year according to the LUC classification can be retrieved using the following query.

SELECT ℓ.*unit(class, sub_class, capability)*, sum(area(ℓ.*boundary*))
FROM LUC ℓ
WHERE ℓ.in:FARM.*farm_name* = 'Rainbow Farm' and ℓ.in:FARM.*year* = 2009
GROUP BY ℓ.*unit(class, sub_class, capability)*;

The previous query can easily be extended to retrieve the improvement in land quality over a certain period:

SELECT ℓ.*unit(class, sub_class, capability)*, sum(area(ℓ.*boundary*))
FROM LUC ℓ
WHERE ℓ.in:FARM.*farm_name* = 'Rainbow Farm' and ℓ.in:FARM.*year* = 2009
GROUP BY ℓ.*unit(class, sub_class, capability)*;
MINUS
SELECT ℓ.*unit(class, sub_class, capability)*, sum(area(ℓ.*boundary*))
FROM LUC ℓ
WHERE ℓ.in:FARM.*farm_name* = 'Rainbow Farm' and ℓ.in:FARM.*year* = 2005
GROUP BY ℓ.*unit(class, sub_class, capability)*;

6 Geometric Types and Algebraic Varieties

Usually, the domain $dom(t)$ associated with a data type t defines a set of values and a set operations acting on these values. For geometric data, however, this approach needs to be questioned. For example, a value of data type *Bezier* as defined in the previous section is simply a list of $n+1$ points $\mathbf{p}_0, \dots, \mathbf{p}_n \in \mathbb{R}^2$. However, it defines a Bézier curve of order n in the two-dimensional plane. This calls for an additional *geometric domain* $gdom(t)$, which associates with a geometric data type t a suitable set of point sets in the n-dimensional Euclidean space $\mathbb{R}^n$, together with a mapping $dom(t) \to gdom(t)$. In the following we will concentrate on the case $n = 2$, i.e., we focus on points, curves and regions in the plane, but most definitions are not bound to this restriction. We will use algebraic varieties and polyhedra to define the point sets of interest.

Definition 8. An *(algebraic) variety* V of dimension n is the set of zeroes of a polynomial P in n variables, i.e., $V = \{(x_1, \dots, x_n) \in \mathbb{R}^n \mid P(x_1, \dots, x_n) = 0\}$. A *base polyhedron* H is the intersection of half planes, i.e.,

$$H = \{(x_1, \dots, x_n) \in \mathbb{R}^n \mid P_i(x_1, \dots, x_n) \geq 0 \text{ for } i = 1, \dots, k\}$$

where $P_1, \dots, P_k$ are polynomials in n variables. A *polyhedron* H is the union of a finite set of base polyhedra $H_1, \dots, H_\ell$. □

It is well-known that algebraic varieties in the plane include all classical two-dimensional curves [Brieskorn and Knörrer, 1981]. Note that $P(x_1, \dots, x_n) = 0$ holds if and only if both $P(x_1, \dots, x_n) \geq 0$ and $-P(x_1, \dots, x_n) \geq 0$ hold true, so base polyhedra are straightforward generalisations of algebraic varieties.

If a point set is defined by polynomials as in Definition 8, this is called an *implicit representation* of the point set. Alternatively, point sets are often given by a function $\gamma(u) : \mathbb{R} \to \mathbb{R}^n$ which is called an *explicit* or *parametric* *representation* of the point set [Gao and Chou, 1992]. Note that an parametric representation can always be turned into an implicit one, while the converse is not always possible. For most two-dimensional point sets of interest, however, it is possible to find rational parametric representations [Gao and Chou, 1992].

Example 8. Take the simple example of the unit circle. It has the implicit representation $P(x,y) = x^2 + y^2 = 0$, but also the rational parametric representation $\gamma(u) = (x,y)$ with $x = \frac{2u}{u^2+1}$ and $y = \frac{1-u^2}{1+u^2}$.

Example 9. A *Bézier curve of degree* n is defined by $n+1$ control points $\mathbf{p}_0,\ldots,\mathbf{p}_n$. A parametric representation is $B(u) = \sum_{i=0}^{n} B_{in}(u)\cdot\mathbf{p}_i$ $(0 \le u \le 1)$. Herein B_{in} denotes the i^{th} *Bernstein polynomial of degree* n, which is defined as $B_{in}(u) = \binom{n}{i} u^i(1-u)^{n-i}$.

A Bézier curve of degree 1 is simply a straight line between its two control points $\mathbf{p}_0$ and $\mathbf{p}_1$. A Bézier curve of degree 2 is a parabolic curve connecting $\mathbf{p}_0$ with $\mathbf{p}_2$ such that its tangents in both points pass through $\mathbf{p}_1$. To obtain an implicit representation for it, we can start with the two quadratic equations $x = au^2 + bu + c$ and $y = du^2 + eu + f$ that define $B(u) = (x,y)$. Dividing these equations by a and d, respectively, and subtracting them from each other eliminates the quadratic term u^2. This can then be solved to give u, plugged back in to give x and y leading to a polynomial in x and y of degree 2 that defines the implicitisation of the Bézier curve.

Example 10. A *Bézier patch* is defined by an $(n \times m)$ array of control points $\mathbf{p}_{ij}$. A parametric representation is $P(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{in}(u)\cdot B_{jm}(v)\cdot\mathbf{p}_{ij}$. If we fix $u = 0$ or $v = 0$, we obtain Bézier curves $P(0,v)$ and $P(u,0)$, respectively.

Definition 9. The *geometric domain* $gdom(t)$ associated with a geometric data type t is a set of point sets. Each element of $gdom(t)$ has an implicit representation by a polyhedron $H = H_1 \cup \cdots \cup H_\ell$ with base polyhedra H_i $(i = 1,\ldots,\ell)$ defined by polynomials $P_{i1},\ldots,P_{in_i}$. For each of the polynomials P_{ij}, the algebraic variety defined by P_{ij} has an explicit parametric representation $\gamma_{ij}(\mathbf{u})$, unless this is impossible. $\square$

To obtain suitable polyhedra for two-dimensional regions surrounded by polygons or polyBézier curves one often *triangulates* the region first. Triangulations are a popular topic in computational geometry, and a number of algorithms have been proposed in the literature to find triangulations efficiently.

Note that the union and the intersection of two polyhedra are polyhedra again, but this is not generally true for their difference. Polyhedra are always closed sets with respect to the standard topology on the Euclidean space $\mathbb{R}^n$, but the difference of closed sets is not necessarily closed. We may, however, regain a polyhedron by building its closure. For a geometric domain, it desirable to have operators available to obtain the *closure* $\bar{X}$, *interior* $\mathring{X}$, and the *boundary* ∂X of a point set X in the geometric domain. The semantics of these operators can be defined in the usual way by $\mathring{X} = \{\mathbf{p} \in X \mid \exists r > 0.U_r(\mathbf{p}) \subseteq X\}$, $\partial X = \{\mathbf{p} \mid \forall r > 0.U_r(\mathbf{p}) \cap X \neq \emptyset \neq U_r(\mathbf{p}) - X\}$, and $\bar{X} = X \cup \partial X$. Herein, $U_r(\mathbf{p})$ denotes the open set of radius $r > 0$ with centre $\mathbf{p}$ in the Euclidean space $\mathbb{R}^n$.

7 Mathematical Challenges in 2D Geometric Modelling

Let us concentrate on geometric data types whose associated geometric domains consist of two-dimensional polyhedra. The polynomials P needed to define two-dimensional polyhedron give rise to plane algebraic curves. The association of geometric domains to geometric data types in GERM unlocks the wealth of a wide range of (partly very old) mathematical results that are ready for exploitation in conceptual modelling of geometric data.

7.1 Logical Properties

Paredaens and Kuijpers [Paredaens and Kuijpers, 1998] studied semi-algebraic sets from the point of view of generic database queries. Typical geometric queries would involve semi-algebraic sets A, their interior $\mathring{A}$, boundary ∂A, closure $\bar{A}$ and complement $\mathbb{R} - A$, distances, angles, and also spatial relations as mentioned in the introduction.

The key question addressed in [Paredaens and Kuijpers, 1998] is completeness. In the context of relational databases, a query language is *complete* if it can express all *computable queries*, i.e., all queries that can be expressed by a partial recursive function that is *generic* in the sense that it commutes with all isomorphisms. Isomorphisms for relational databases are defined by permutations acting on the domain of values, which are then extended to tuples and relations in a canonical way.

This approach, however, cannot be transferred to geometrical queries without making changes to the notion of isomorphism, as arbitrary permutation of points in Euclidean space would destroy all geometric properties. Therefore, genericity has been defined with respect to various groups that operate on Euclidean space such as the group T of *translations* and the group I of *isometries*, i.e. transformations that preserve distances. Unfortunately, for many relevant groups G the property of G-genericity is undecidable [Paredaens et al., 1994]. Still, Gyssens et al. [Gyssens et al., 1997] were able to define languages that capture exactly the G-generic queries for some relevant groups such as T and I.

Each such transformation group G expresses the geometric properties we would like to preserve in queries, so the completeness results in [Gyssens et al., 1997] show that for the most relevant cases we can adequately query databases involving geometric objects that are defined by semi-algebraic sets, i.e. we obtain again semi-algebraic sets as the result of such queries. Furthermore, the execution plans of such queries prescribe the level of accuracy that will be required for elementary operations on point sets such as union and intersection.

7.2 Classification of Plane Algebraic Curves

While querying can be effectively expressed, we want to go further and demand that one must also be able to retrieve a surface representation by means of geometric data types in GERM. For that, we require translations from polyhedra to the domain values of a geometric data type t, that is, a mapping from $gdom(t)$ back to $dom(t)$.

For this let us take a closer look at plane algebraic curves, that is, algebraic varieties defined by polynomials with two variables. In order to understand how to approach operations on these objects (and thus the polyhedra defined by them), we may exploit the classification of such curves, for which we consider the degree of the defining polynomial P. If the degree of P is 1, we have $P(x, y) = ax + by + c$ with coefficients $a, b, c \in \mathbb{R}$. Obviously, polynomials of degree 1 define all straight lines in the plane, including the degenerated case of a single point. If the degree of P is 2, we have $P(x, y) = ax^2 + by^2 + cxy + dx + ey + f$ with coefficients $a, b, c, d, e, f \in \mathbb{R}$. It is known that polynomials of degree 2 define all cone sections [Brieskorn and Knörrer, 1981], that is, all circles, ellipses, parabolas, and hyperbolas, including the degenerated cases of straight lines, two straight lines and a single point.

For a semi-algebraic set A defined by formula φ we can then apply de Morgan's laws to obtain a form, in which negation only appears in front of atomic formulae. From the fact that $\neg P(x, y) \geq 0$ is equivalent to $P(x, y) < 0$ and further to $-P(x, y) > 0$ we can further eliminate negation completely for the price of permitting $>$ in our formulae. Using laws of distributivity we can further arrange A as a union of intersections of elementary semi-algebraic sets that are defined by $P(x, y) \geq 0$ or $P(x, y) > 0$. As emphasised in [Ma et al., 2009] most applications of geometric modelling deal with compact sets, i.e., we are mainly interested in elementary sets defined by $P(x, y) \geq 0$. The intersection of such sets will then be defined by sections of plane algebraic curves.

Thus returning to the cases of degree 1 and 2, the result of the most relevant operations union, intersection and difference (modulo boundaries) gives rise to regions with boundaries defined piecewise by straight lines and cone sections. This is in accordance with the geometric data types provided by GERM. In particular, Bézier curves are sections of parabolas.

Furthermore, as non-degenerated cone sections are smooth, we always obtain an explicit parametric representation as discussed in Examples 8 and 9 for circles and parabolas, respectively, and this enables us to easily obtain the parameterisation of sections. If $C(u)$ is a parameterisation of a subsection of a cone section with parameter $u \in [0, 1]$, and $C(u_0)$ and $C(u_1)$ (for $u_0 < u_1$) defines a subsection of this curve, then this subsection can be obtained by a simple linear parameter transformation, i.e., $C'(u) = C((u_1 - u_0) \cdot u + u_0)$ for $0 \leq u \leq 1$ defines the parameterisation of the subsection.

Similarly, the case of degree 3 requires the discussion of the following four cases [Brieskorn and Knörrer, 1981]:

$$xy^2 + ey = ax^3 + bx^2 + cx + d \tag{1}$$

$$xy = ax^3 + bx^2 + cx + d \tag{2}$$

$$y^2 = ax^3 + bx^2 + cx + d \tag{3}$$

$$y = ax^3 + bx^2 + cx + d \tag{4}$$

These equations have already been carefully analysed by Newton using a detailed investigation of 78 cases. Some of these curves were already known by the ancient Greeks, e.g., the *kissoid of Diocles*, the *folium Cartesii* and the *Neilean parabola* defined by the polynomials $y^2(2r - x) - x^3$, $x^3 + y^3 + axy$, and $x^3 - y^2$, respectively.

Note that in the case of degree 3, curves can observe singularities, but nevertheless can be composed piecewise by smooth sections. For these sections we obtain parametric representations, which give rise to explicit parameterisations of their subsections.

7.3 Natural Modelling Algebra

The two layers of GERM support the storage and retrieval of geometric objects within a conceptual model. The challenge is, however, the manipulation of such objects by queries and transactions. For this we now present an algebra on geometric objects. As we always have an internal representation by point sets, we first focus on these.

Standard operations on point sets are of course the Boolean ones, i.e. union, intersection, difference, and complement. In combination with interior, closure and boundary these operations are in principle sufficient to express a lot of relationships between the geometric objects as discussed widely in the conceptual GIS literature (see e.g. [Behr and Schneider, 2001; Shekhar and Xiong, 2008]). For instance, $A - B = \emptyset$ is equivalent to $A \subseteq B$, so we only need difference and an emptiness test. Similarly, $\mathring{A} \cap B = \emptyset \wedge \partial A \cap \partial B \neq \emptyset$ express that A and B touch each other, but do not intersect.

However, relying on the Boolean set operations is insufficient. We have to address two concerns: 1) The *closure problem*: The set of point sets of interest must be closed under the operations. For example applying the operations on polyhedra should also return a polyhedra. We already remarked at the end of the previous section that this is not true for the set difference and complement operation. 2) The *stability problem*: The operations must be numerically stable so that unavoidable numerical error, which are due to the rounding that is necessary when dealing with floating-point representations of real numbers, remains bounded and very small and that no large errors occur.

We can circumvent the closure problem as we are merely interested in point sets "up to their boundary". That is, we will apply an equivalence relation $\sim$ where $A \sim B$ holds for two geometric if and only if $\bar{A} = \bar{B}$. Then each equivalence class has exactly one closed representative: a polyhedron. The problem then is that the Boolean operations do not preserve this equivalence, and we lose some of the properties of a Boolean algebra. However, these properties are volatile anyway due to the necessary modifications that we propose to deal with the stability problem.

For the stability problem, some conceptual modelling experts might argue that this concerns only an implementation. We do not share this opinion as any result obtained by operations of point sets (i.e. the polyhedra on the internal level) must be re-interpreted by a value of the respective geometric data type on the surface level. For instance, the union and intersection of polygons must again be represented as a polygon, having a finite sequence of points as its surface representation. Furthermore, we must be aware that the intersection of two two-dimensional curves may be more than just a discrete set of points when the stability problem is addressed adequately. Thus, stability considerations have an non-negligible impact on the surface level of GERM.

7.4 Modification of Boolean Operations

It is known that Boolean operations on point sets may be instable. For instance, for two straight lines their intersection point may be only obtainable with an intolerable error. This problem occurs, when the angle between the two lines is very small. Our solution will replace the intersection operation by a modified operation, which in this case will enlarge the result. Thus, we actually obtain a point set instead of a single point. The enlargement will depend on the operands, so that for the uncritical cases we almost preserve the Boolean operations.

In general, we use the following new operations on point sets: $A \uplus B = A \cup B \cup q(A, B)$ and $A \Cap B = (A \cap B) \cup q(A, B)$ with a *natural modelling function* q that assigns a point set to a pair of point sets. The name "natural modelling" is adopted from [Hartwig, 1996], as it should reflect properties associated with stability and the original union and intersections operations in a natural way.

We do not modify the complement X' of a set X. With $A \mathbin{\cup\mkern-11mu-} B = (A' \mathbin{\cap\mkern-11mu+} B')'$ and $A \mathbin{\cap\mkern-11mu-} B = (A' \uplus B')'$ we obtain two more modified operations. The simple idea behind these operations is to slightly enlarge (or reduce) unions and intersections in order to cope with the stability problem. The enlargement (or reduction) depends on the arguments; critical operands require larger modifications than uncritical ones.

Definition 10. A function q from pairs of point sets to point sets is called a *natural modelling function* iff it satisfies the following properties for all A, B:

$$q(A, B) = q(B, A) \qquad q(A', B) = q(A, B) \qquad q(A, \emptyset) = \emptyset$$

□

We require q to be symmetric, as the stability problem for building intersections and unions does not depend on the order. Analogously, the potential instability caused by A and B is the same as the one caused by A' and B.

The simple idea behind these operations is to slightly enlarge unions and intersections in order to cope with the accuracy problem. The enlargement depends on the arguments – critical operands require larger modifications than uncritical ones.

Definition 11. The *natural modelling algebra* consists of the set of equivalence classes of polyhedra with respect to $\sim$ and the operations $\uplus$, $\mathbin{\cap\mkern-11mu+}$, $\mathbin{\cap\mkern-11mu-}$ and $\mathbin{\cup\mkern-11mu-}$ with a natural modelling function q. □

Hartwig [Hartwig, 1996] studied the algebraic properties of the *direct* modelling algebra $(\mathcal{P}(E), \uplus, \mathbin{\cap\mkern-11mu+})$ and the *small* modelling algebra $(\mathcal{P}(E), \uplus, \mathbin{\cap\mkern-11mu-})$. In both cases we obtain a *weak Boolean algebra*, i.e., the existence of neutral and inverse elements is preserved, and the de Morgan laws still hold, but other properties of Boolean algebras have been abandoned.

Using the idea of a "natural modelling algebra" an intersection point of two curves may be replaced by several points. In doing so, the desirable property of having the exact intersection is given up in favour to a "global" minimisation of error. For instance, if the union of two regions requires the computation of an intersection point $\mathbf{p}_i$ of two curves γ_1 and γ_2, then the approach may result in two points $\mathbf{q}$ and $\mathbf{r}$ instead, and instead of building subsections of γ_1 and γ_2 involving the point $\mathbf{p}_i$ we may get subsections involving the point $\mathbf{q}$ or $\mathbf{r}$, respectively, and the straight line between $\mathbf{q}$ and $\mathbf{r}$ will be added to the solution.

7.5 Computing with Polyhedra and Surface Representations

The key question is of course how to choose a good natural modelling function q. Before addressing this let us first look at the modified operations on polyhedra. As these are defined by algebraic varieties, it will be decisive (and sufficient) to understand the operations on two half-planes $A = \{(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n) \geq 0\}$ and $B = \{(x_1, \ldots, x_n) \mid Q(x_1, \ldots, x_n) \geq 0\}$. If A and B are plane curves, we have to compute their intersection point(s) in order to determine a surface representation of their union and intersection, respectively. Let us discuss this further for polygons and regions defined by a sequence of Bézier curves.

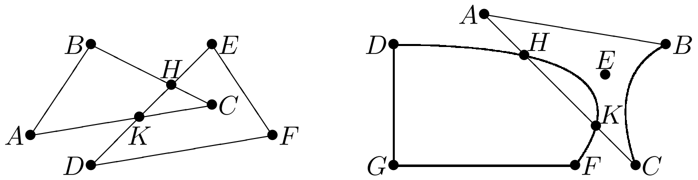

Figure 5: *On the left the intersection of two polygons, on the right the intersection of two regions with a boundary defined by Bézier curves*

Example 11. Let us look at the union / intersection of two polygons depicted on the left in Figure 5, one defined by the points A, B, C, the other one by D, E, F. With $A = (1, 1)$, $B = (3, 4)$, $C = (7, 2)$, $D = (3, 0)$, $E = (7, 4)$, and $F = (9, 1)$ the line through D and E is defined by $P(x, y) = x - y - 3 = 0$, and the line through B and C is defined by $Q(x, y) = x + 2y - 11 = 0$. They intersect in the point $H = (5.66, 2.66)$. This intersection divides the plane into four parts depending on whether $P(x, y)$ and $Q(x, y)$ take positive or negative values.

If we can compute the intersection points H and K, then A, B, H, E, F, D, K defines the surface representation of the union, while K, H, C defines the one of the intersection.

However, the angle between the lines DE and AC at the intersection point K is rather small, which may cause a different result defined by the operations $\uplus$ and $\sqcap$ instead of $\cup$ and $\cap$, respectively. The resulting polygon for the modified union may become $A, B, H, E, F, D, K_1, K_2$, while the resulting polygon for the modified intersection may become K_1', H, C, K_2' with points K_1, K_2, K_1', K_2' in a small neighbourhood of K.

At H the angle between the two intersecting lines is nearly a right angle, so the modified intersection may coincide with the normal one.

Example 12. Look at the two regions defined on the right in Figure 5, both defined by values of the data type *PolyBezier*, the first one by $[(A, B), (B, E, C), (C, A)]$, the second one by $[(D, B, F), (F, G), (G, D)]$. As in the previous example the two intersection points H and K of the line (A, C)

with the Bézier curve (D, E, F) are decisive for the computation of the union and intersection.

With $A = (16, 5)$, $B = (22, 4)$, $E = (20, 3)$, $C = (21, 0)$, $D = (13, 4)$, $F = (19, 0)$, and $G = (13, 0)$ the parametric representation of the Bézier curve can be easily obtained as $B(u) = (-12u^2 + 18u + 13, -4u^2 + 4)$, and the straight line gives rise to $x + y - 21 = 0$. Substituting $B(u) = (x, y)$ in this gives rise to a quadratic equation with the roots $u_{1/2} = \dfrac{9 \pm \sqrt{17}}{16}$, i.e. $u_1 = 0.304$ and $u_2 = 0.82$, which define $H = (17.32, 3.64)$ and $K = (19.69, 1.31)$. Then the union can be represented by $[(A, B), (B, E, C), (C, K), (K, F', F), (F, G), (G, D), (D, D', H), (H, A)]$ of the data type *PolyBezier*, while the intersection is represented by $[(H, H', K), (K, H)]$. Once H and K are known, it is no problem to obtain the necessary points D' and H', as sections of Bézier curves are again Bézier curves.

As in Example 11 the computation of the point K can be expected to be relatively stable, whereas H is not. Using $\uplus$ instead of the usual union, we end up with a modified union represented by $[(A, B), (B, E, C), (C, K), (K, F', F), (F, G), (G, D), (D, D', H_1), (H_1, H_2), (H_2, A)]$ of the data type *PolyBezier*, where H_1 and H_2 are points in the vicinity of H on the Bézier curve and the straight line (H, A), respectively. Analogously, using $\Cap$ instead of $\cap$, we obtain a representation $[(H_2', H_1')(H_1', H', K), (K, H_2')]$ with points H_1', H_2' in the vicinity of H on the Bézier curve and the straight line (H, K), respectively.

7.6 The Choice of the Natural Modelling Function

In view of the discussion in the previous subsection it is sufficient to consider base polyhedra, i.e. if $H = H_1 \cup \dots \cup H_n$ and H' are polyhedra, we define $q(H, H') = \bigcup_{i=1}^{n} q(H_i, H')$. Furthermore, for base polyhedra it is sufficient to consider the boundary, i.e. if H and H' are base polyhedra, we define $q(H, H') = q(\partial H, \partial H')$. In the two-dimensional plane $E = \mathbb{R}^2$ we can therefore concentrate on plane curves. If such a curve γ is defined by a union of (sections of) algebraic varieties, say $V_1 \cup \dots \cup V_n$, then we define again $q(\gamma, \gamma') = \bigcup_{i=1}^{n} q(V_i, \gamma')$. If q is symmetric, the naturalness conditions in Definition 10 are obviously satisfied.

In order to obtain a good choice for the natural modelling function q it is therefore sufficient to look at two curves γ_1 and γ_2 defined by polynomials $P(x, y) = 0$ and $Q(x, y) = 0$, respectively. Let $\mathbf{p}_1, \dots, \mathbf{p}_n$ be the intersection points of these curves – unless $\gamma_1 = \gamma_2$ we can assume that there are only finitely many. Then we define $q(\gamma_1, \gamma_2) = \bigcup_{i=1}^{n} U_i$ with neighbourhood $U_i = U_{\gamma_1, \gamma_2}(\mathbf{p}_i)$ as defined next.

Definition 12. For $\varepsilon > 0$ the ε-*band* of a variety $V = \{(x, y) \mid P(x, y) = 0\}$ is the point set $B_\varepsilon(V) = \{(x', y') \mid \exists(x, y) \in V.|x - x'| < \varepsilon \wedge |y - y'| < \varepsilon\}$. □

8 Conclusion

In this paper we presented the geometrically enhanced ER model (GERM) as our approach to conceptual geometric modelling. GERM preserves aggregation as the primary abstraction mechanism of the entity-relationship model, but loosens the definition of relationship types permitting bulk and choice constructors to be used for components without first-class status of bulk objects. Geometric features in the application domain can be modelled by attributes that have geometric data types assigned to them. This defines a syntactic level of GERM that largely remains within the popular ER framework and thus enables a smooth integration with non-geometric modelling. It also allows users to cope with modelling tasks that involve geometry in a familiar, non-challenging way thereby preserving all the positive experience made with entity-relationship modelling.

The syntactic level is complemented by an internal level that employs algebraic varieties, i.e., sets of zeros of polynomials, to represent geometric features as point sets. The use of such varieties leads to a significant increase in expressiveness way beyond standard approaches that mostly support points, lines and polygons. In particular, common shapes such as circles, ellipses, or Bézier curves and patches are captured in a natural way. However, for polynomials of high degrees we have to face computational problems.

The highly expressive internal level of GERM makes geometric modelling not only very flexible, it is only the basis for an extended algebra that generalises and extends the standard Boolean operators on point sets. By using this algebra, GERM enables a higher degree of accuracy for derived geometric relationships.

Our next short-term goal is to apply GERM to WFP modelling for the Sustainable Land Use Initiative (SLUI). In order to support the wider objectives of this programme, GERM is general enough to capture spatio-temporal data as well. We are also looking for applications beyond GIS. On the theoretical side we plan to investigate further back and forth translations between the syntactic and the internal level of GERM, and special cases of the natural modelling algebra for specific applications. In this sense this paper is only the start of a larger research programme devoted to geometric conceptual modelling.

Achnowledgement

I am grateful to my former colleagues at the Horizons Council for discussions.

References

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

[AgResearch, 2005] AgResearch (2005). Farm plan prototype for SLUI. retrieved online from the New Zealand Association of Resource Management http://www.nzarm.org.nz/KinrossWholeFarmPlan_A4_200dpi_secure.pdf.

[Balley et al., 2004] Balley, S., Parent, C., and Spaccapietra, S. (2004). Modelling geographic data with multiple representations. *International Journal of Geographical Information Science*, 18(4):327–352.

[Behr and Schneider, 2001] Behr, T. and Schneider, M. (2001). Topological relationships of complex points and complex regions. In *Conceptual Modeling – ER*, volume 2224 of *LNCS*, pages 56–69. Springer.

[Brieskorn and Knörrer, 1981] Brieskorn, E. and Knörrer, H. (1981). *Plane Algebraic Curves*. Birkhäuser.

[Chen and Zaniolo, 2000] Chen, C. X. and Zaniolo, C. (2000). SQLST: A spatio-temporal data model and query language. In *Conceptual Modeling – ER 2000*, volume 1920 of *LNCS*, pages 96–111. Springer.

[Corbett, 1979] Corbett, J. P. (1979). Topological principles in cartography. Technical paper 48, US Bureau of the Census.

[Egenhofer, 1994] Egenhofer, M. J. (1994). Spatial sql: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6:86–95.

[Frank, 2005] Frank, A. U. (2005). Map algebra extended with functors for temporal data. In *Perspectives in Conceptual Modeling – ER Workshops*, volume 3770 of *LNCS*, pages 194–207. Springer.

[Gao and Chou, 1992] Gao, X. S. and Chou, S. C. (1992). Implicitization of rational parametric equations. *Journal of Symbolic Computation*, 14:459–470.

[Gogolla, 1994] Gogolla, M. (1994). *Extended Entity-Relationship Model: Fundamentals and Pragmatics*. Springer.

[Gubiani and Montanari, 2008] Gubiani, D. and Montanari, A. (2008). A conceptual spatial model supporting topologically-consistent multiple representations. In *International Conference on Advances in Geographic Information Systems – GIS*, pages 1–10. ACM.

[Guting and Schneider, 1995] Guting, R. H. and Schneider, M. (1995). Realm-based spatial data types: The ROSE algebra. *VLDB J.*, 4:243–286.

[Gyssens et al., 1997] Gyssens, M., Van den Bussche, J., and Van Gucht, D. (1997). Complete geometrical query languages. In *PoDS*, pages 62–67. ACM.

[Hadzilacos and Tryfona, 1997] Hadzilacos, T. and Tryfona, N. (1997). An extended entity-relationship model for geographic applications. *SIGMOD Record*, 26(3):24–29.

[Hartmann and Link, 2007] Hartmann, S. and Link, S. (2007). Collection type constructors in entity-relationship modeling. In Parent, C. et al., editors, *Conceptual Modeling – ER*, volume 4801 of *LNCS*, pages 307–322. Springer.

[Hartmann et al., 2004] Hartmann, S., Link, S., and Schewe, K.-D. (2004). Weak functional dependencies in higher-order datamodels. In *FoIKS*, volume 2942 of *LNCS*, pages 116–133. Springer.

[Hartmann et al., 2006] Hartmann, S., Link, S., and Schewe, K.-D. (2006). Axiomatisations of functional dependencies in the presence of records, lists, sets and multisets. *Theor. Comput. Sci.*, 355(2):167–196.

[Hartwig, 1996] Hartwig, A. (1996). *Algebraic 3-D Modeling*. A. K. Peters.

[Hull and King, 1987] Hull, R. and King, R. (1987). Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):201–260.

[Ishikawa and Kitagawa, 2001] Ishikawa, Y. and Kitagawa, H. (2001). Source description-based approach for the modeling of spatial information integration. In *Conceptual Modeling – ER*, volume 2224 of *LNCS*, pages 41–55. Springer.

[Jensen et al., 2004] Jensen, C. S., Kligys, A., Pedersen, T. B., and Timko, I. (2004). Multidimensional data modeling for location-based services. *VLDB J.*, 13(1):1–21.

[Kanellakis et al., 1990] Kanellakis, P., Kuper, G., and Revesz, P. (1990). Constraint query languages. In *PoDS*, pages 299–313. ACM.

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

[Liu et al., 2005] Liu, W., Chen, J., Zhao, R., and Cheng, T. (2005). A refined line-line spatial relationship model for spatial conflict detection. In *Perspectives in Conceptual Modeling – ER Workshops*, volume 3770 of *LNCS*, pages 239–248. Springer.

[Lorie and Meier, 1984] Lorie, R. and Meier, A. (1984). Using relational DBMS for geographical databases. *GeoProcess.*, 2:243–257.

[Ma et al., 2009] Ma, H., Schewe, K.-D., and Thalheim, B. (2009). Geometrically enhanced conceptual modelling. In *Conceptual Modeling – ER*, volume 5829 of *LNCS*, pages 219–233. Springer.

[Mackay, 2007] Mackay, A. (2007). Specifications of whole farm plans as a tool for affecting land use change to reduce risk to extreme climatic events. AgResearch.

[Malinowski and Zimányi, 2007] Malinowski, E. and Zimányi, E. (2007). Logical representation of a conceptual model for spatial data warehouses. *Geoinformatica*, 11(4):431–457.

[McKenny and Schneider, 2007] McKenny, M. and Schneider, M. (2007). PLR partitions: A conceptual model of maps. In *Advances in Conceptual Modeling –ER Workshops*, volume 4802 of *LNCS*, pages 368–377. Springer.

[Paredaens, 1995] Paredaens, J. (1995). Spatial databases, the final frontier. In Gottlob, G. and Vardi, M. Y., editors, *ICDT*, volume 893 of *LNCS*, pages 14–32. Springer.

[Paredaens and Kuijpers, 1998] Paredaens, J. and Kuijpers, B. (1998). Data models and query languages for spatial databases. *Data and Knowledge Engineering*, 25(1-2):29–53.

[Paredaens et al., 1994] Paredaens, J., Van den Bussche, J., and Van Gucht, D. (1994). Towards a theory of spatial database queries. In *PoDS*, pages 279–288. ACM.

[Peano, 1890] Peano, G. (1890). Sur une courbe qui remplit toute une aire plane. *Mathematishe Annalen*, 36:157–160.

[Price et al., 2001] Price, R., Tryfona, N., and Jensen, C. S. (2001). Modeling topological constraints in spatial part-whole relationships. In *Conceptual Modeling – ER*, volume 2224 of *LNCS*, pages 27–40. Springer.

[Roussopoulos et al., 1988] Roussopoulos, N., Faloutsos, C., and Sellis, T. K. (1988). An efficient pictorial database system for PSQL. *IEEE Trans. Software Eng.*, 14(5):639–650.

[Sali and Schewe, 2009] Sali, A. and Schewe, K.-D. (2009). A characterisation of coincidence ideals for complex values. *Journal of Universal Computer Science*, 15(1):304–354.

[Salomon, 2005] Salomon, D. (2005). *Curves and Surfaces for Computer Graphics*. Springer.

[Schneider, 2009] Schneider, M. (2009). Spatial and spatio-temporal data models and languages. In *Encyclopedia of Database Systems*, pages 2681–2685. Springer.

[Shekhar et al., 1997] Shekhar, S., Coyle, M., D.-R. Liu, B. G., and Sarkar, S. (1997). Data models in geographic information systems. *Communications of the ACM*, 40(4):103–111.

[Shekhar et al., 1999] Shekhar, S., Vatsavai, R. R., Chawla, S., and Burk, T. E. (1999). Spatial pictogram enhanced conceptual data models and their translation to logical data models. In Agouris, P. and Stefanidis, A., editors, *Integrated Spatial Databases, Digital Images and GIS*, volume 1737 of *LNCS*, pages 77–104. Springer.

[Shekhar and Xiong, 2008] Shekhar, S. and Xiong, H., editors (2008). *Encyclopedia of GIS*. Springer.

[Stoffel et al., 2007] Stoffel, E.-P., Lorenz, B., and Ohlbach, H.-J. (2007). Towards a semantic spatial model for pedestrian indoor navigation. In *Advances in Conceptual Modeling – ER Workshops*, volume 4802 of *LNCS*, pages 328–337. Springer.

[Thalheim, 2000] Thalheim, B. (2000). *Entity Relationship Modeling – Foundations of Database Technology*. Springer.

[Westlake and Kleinschmidt, 1990] Westlake, A. and Kleinschmidt, I. (1990). The implementation of area and membership retrievals in point geography using sql. *IEEE Data Eng. Bull.*, 13(3):4–11.