SNAL: Spatial Network Algebra for Modeling Spatial Networks in Database Systems

Lin Qi, Huiyuan Zhang and Markus Schneider
Department of Computer & Information Science & Engineering, University of Florida, Gainesville, U.S.A.

Keywords: Spatial Networks, Spatial Database, Abstract Model, Design Criteria, Spatial Network Algebra.

Abstract: Spatial networks such as road networks, river networks, telephone networks, and power networks are ubiquitous spatial concepts deployed, for example, in route planning, communication services, high voltage grid topology analysis, and utility management. Current database systems are unable to efficiently handle, represent, store, query, and manipulate large spatial networks. Moreover, data models of spatial networks in a database context are rare due to their inherently complex nature. This paper offers a conceptual foundation called *Spatial Network Algebra* (*SNAL*) for designing, characterizing, and representing spatial networks. A general-purpose abstract model is proposed as a specification for a later implementation of spatial networks in different environments such as spatial database systems and GIS.

1 INTRODUCTION

A *spatial network* is a *spatially embedded and labeled graph*. Infrastructures like road networks, power networks, and river networks are examples of spatial networks. Advances in and analytical applications around such networks have boosted urbanization and infrastructure development and produced large volumes of spatial data describing the complex inner structure and connectivity features of these networks.

Given the fact that existing data models, query languages as well as geo-information aware database systems hardly offer adequate and sophisticated support for the modeling and querying of spatial networks, we propose to build up an explicit object-oriented data model and query language. This data model integrates an explicit modeling and querying of networks into a standard database environment. Database support is essential to store the large volumes of spatial network data and to utilize them in various GIS applications in an efficient way. Providing a *spatial data type* for spatial networks by means of an *abstract data type* that is integrated into a database system would also allow a user to query and manipulate spatial networks in a database setting by high-level operations defined on them. In other words, spatial networks would become ”first class citizens” in the database allowing SQL queries to run on spatial networks and take advantage of the spatial network operations and predicates.

In this paper, we focus on developing an abstract model of spatial networks without any regard to implementation details. This step involves providing a formal definition of spatial networks based on mathematical concepts instead of an intuitive description. The existence of a formal definition that precisely describes the properties of spatial networks is essential for defining operations and predicates on them. In other words, the model will perform as a *type system* for spatial networks, which we call *Spatial Network Algebra* (*SNAL*). The abstract data model is based on point set theoretical and topological concepts. We conceptually associate all points of the Euclidean plane with thematic values. All points with equal thematic values are part of the same component of the network (e.g., all points with the same interstate name “I75”). Interior and exterior components of a network can be distinguished based on these thematic values. Further, this model will act as a specification for the eventual implementation of spatial networks in databases. In particular, this approach enables us to consider attribute values of single points in the network (*space based view*) but also provides access to collection of points having equal attributes values (*object based view*).

The rest parts of this paper are organized as follows: Section 2 discusses the related models of spatial networks and compares them with our approach. Our abstract data model of spatial networks is introduced in Section 3. Finally, Section 4 concludes the paper.

Qi, L., Zhang, H. and Schneider, M.
SNAL: Spatial Network Algebra for Modeling Spatial Networks in Database Systems.
In *Proceedings of the 2nd International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2016)*, pages 145-152
ISBN: 978-989-758-188-5

2 RELATED WORK

Conceptual data models for spatial networks can be broadly classified into three categories: *graph-oriented* models, *hybrid* models, and *network-oriented* models.

One reasonable and natural idea is to model spatial networks as *graphs* to capture their structure and connectivity. Since nodes and arcs correspond to the vertices and edges of a graph, the planar embedding of the node-arc data models in GIS ensure topological consistency and regard a spatial network as a directed graph. This concept has been taken in Güting (1994), Erwigm and Güting (1994), Brinkhoff (2002), Gupta et al. (2004), Jeung et al. (2010). The model in Güting (1994) is designed for embedding graphs in databases and not specifically for spatial networks. A multi-level order-sorted algebra described in Erwigm and Güting (1994) shared a similar idea with our proposed model which allows users to express a query at a high level of abstraction. The authors in Jensen et al. (2003) point out that the graph modeling of a spatial network is not appropriate as it does not present a realistic representation of enough complexity in the real world. Besides, graph-oriented modeling of spatial networks are unable to represent the spatial embedding of a network Qi and Schneider (2012).

The *hybrid* modeling of spatial networks is an alternative approach that introduces spatial embedding to each vertex in a graph-oriented model. The work in Scheider and Kuhn (2008) models road networks as *partially embedded graphs* and defines road components based on their properties. The path-based approach Krogh et al. (2014) and the PARINET approach Popa et al. (2011) model the network in a hybrid manner that supports segments, edges, and route abstractions. These partial graph-oriented approaches lose the geometric information in the networks and only maintain the topology of the network. Moreover, this solution does not allow spatial networks to have attributes of their own. Though effective, it has been shown that this method of modeling a spatial network is neither elegant nor robust Miller and Shaw (2001).

Network-oriented models are geometry-based models for spatial networks. The most well known network model is Güting et al. (2006) which creates an explicit network data model to be used specifically for an implementation of moving objects in a network. This model treats a spatial network as a set of routes and a set of junctions between routes. A recent advance in Ding et al. (2015) introduces a new *network-matched* mechanism to derive moving objects trajectories. Since both of the models focus on modeling of moving objects in road networks, they are heavily biased towards road networks with specific features, for example, to distinguish between simple and dual (divided) roads and thus cannot serve as generic network models. Our emphasis in this paper is on an increase of the functionality and elegance at the conceptual level, the propagation of the abstract data type approach in a database context, and the possibility of query support.

3 FORMAL DEFINITION OF AN ABSTRACT SPATIAL NETWORK MODEL

This section provides our complete abstract data model for spatial networks, *SNAL*, which is the extension of the spatial network specification in Qi et al. (2015). Let $(X, \mathcal{T})$ be a *topological space* Dugundi (1966) with topology $\mathcal{T} \subset 2^X$. For each 3D spatial data type A, we will specify the topological notations of *boundary* (∂A), *interior* (A^o), *exterior* (A^-), and *closure* ($\bar{A}$). To give structural definitions, we need to first present some basic notations and definitions.

Function and Range. The application of a function $f : A \to B$ is a set of values $S \subset A$ defined as $f(S) := \{f(x) | x \in S\}$. If $f(S)$ returns a singleton set, we will then write $f[S]$ to denote the single element, i.e. $f(S) = \{y\} \Rightarrow f[S] = y$. The inverse function of f is defined as $f^{-1}(y) := \{x \in A | f(x) = y\}$. Since f is not always a *bijection*, therefore the signature of f^{-1} is $f^{-1} : B \to 2^A$ where 2^A is the *power set* of A, it is sometimes denoted as $P(A)$ or $\mathcal{P}(A)$. It is important to note that f^{-1} is a total function and that f^{-1} applied to a set yields a set of sets. We also define a *range* function of a function $f : A \to B$ that returns the set of all elements that f returns for an input set A as $rng(f) := f(A)$.

Label Type. In a spatial network, each point is associated with a *label*. A label is consist of two parts: *spatial label* (*sl*) and *thematic label* (*tl*). The spatial label specifies the ownership of the point. For example, the spatial labels of the simple points that belong to the same area are identical. Each point also carries certain thematic information, such as the oil pipe diameter in a pipeline network, ship capacity in a river network or speed limit in a road network at this specific point. During abstract modeling, spatial label plays an important role and is frequently referenced. From this point forward, we use the term *label* to refer to a *spatial label* that is attached to each point.

We call a type that contains labels of the same kind as *label type*. We assume that each label type A contains an element $\perp_A$ that represents the undefined

value. It is called the *exterior* label, and the outside area of a network is labeled by it. For the Cartesian product of two label types A and B, we let $\perp_{A\times B} = (\perp_A, \perp_B)$, and for the union of A and B, we equate $\perp_A$, $\perp_B$, and $\perp_{A\cup B}$. Further, if no ambiguities can arise, we omit the type index and simply use $\perp$.

Spatial Mapping. Given label type A, let $\mathbb{L} = 2^A$ be the corresponding label space. A *spatial mapping* of type A is then defined as a total function $\pi : \mathbb{R}^3 \to \mathbb{L}$. Let A be the label type, and π be the spatial mapping with an arbitrary spatial partition McKenney and Schneider (2007), the *area* and *border* of π are defined as

$$\gamma(\pi) = \pi^{-1}(rng(\pi) \cap \{X \in 2^A |\ card(X) = 1\}); \quad (1)$$

$$\omega(\pi) = \pi^{-1}(rng(\pi) \cap \{X \in 2^A |\ card(X) > 1\}). \quad (2)$$

where $\gamma(\pi)$ represents the *area* and $\omega(\pi)$ represents the *border* respectively. The labels on the borders are modeled using the power set 2^A: a *area* of π is a block mapped to a singleton set, as opposed to a *border* of π which is a block that is mapped to a subset of A containing two or more elements. As an example, for type $A = \{a, b, \perp\}$, the range function gives us $rng(\pi) = \{\{a\}, \{b\}, \{\perp\}, \{a, b\}, \{a, \perp\}, \{b, \perp\}, \{a, b, \perp\}\}$. Therefore, the areas of π are the blocks labeled $\{a\}$, $\{b\}$, and $\{\perp\}$ while the borders are the blocks labeled $\{a, b\}$, $\{a, \perp\}$, $\{b, \perp\}$, and $\{a, b, \perp\}$. The structural definitions are given based on appropriate spatial mapping function. To sum up, the goal is to identify a spatial mapping that assigns appropriate thematic information (i.e. appropriate areas and borders) to every point in $\mathbb{R}^3$ space. Such a spatial mapping represents a spatial network.

Now we start introducing the comprising data types that belong to the spatial networks, we will begin from the simplest type to more complex ones.

3.1 Poi3D

A single point in 3D space ($\mathbb{R}^3$) is of spatial data type *Poi3D*. Spatial data type *Poi3D* is defined as

$$Poi3D = (x, y, z) \text{ where } x, y, z \in \mathbb{R} \quad (3)$$

Each *Poi3D* is consist of an ordered triple of coordinates: x, y, and z in Euclidean plane. We say that two *Poi3D*'s p and q *coincide* if, and only if, $x_p = x_q \wedge y_p = y_q \wedge z_p = z_q$. If two points do not coincide, they are called *disjoint*. The topological notations of a simple point $p = (x, y, z)$ are defined as: (1) $\partial p = \varnothing$; (2) $p^o = \{p\}$; (3) $p^- = \mathbb{R}^3 - (\partial p \cup p^o) = \mathbb{R}^3 - \{p\}$; (4) $\bar{p} = (\partial p \cup p^o) = \{p\}$. In addition, for a simple point type *Poi3D*, the structural definition would be an image of this point in label space with a certain label type that is a subset of 2^A.

3.2 Point3D

A value of type *Point3D* is defined as a finite set of isolated *Poi3D*'s in 3D space. Spatial data type *Point3D* is formally defined as

$$Point3D = \{P \subset \mathbb{R}^3 \mid 0 \leq card\{P\} < \infty \wedge P \text{ is distinct}\} \quad (4)$$

where $card\{P\}$ is the cardinality of set P that measures the total number of simple points of the set. We call *Point3D* a *complex point*. For a complex point P, if $card\{P\} = 1$, it then represents a singleton set that only includes one *Poi3D*. If $card\{P\} = 0$, set P becomes an *empty set* $\varnothing$. An empty set can be admitted from geometric operations such as intersection and union. For example, the intersection of two *Point3D*'s with no points in common yields the empty set. The topological components of a complex object is always the union of its consisting simple objects. The topological notations of a point $P = \{p_1, p_2, ..., p_n\}$ are defined as: (1) $\partial P = \cup_{i=1,2,...,n} \partial p_i = \cup_{i=1,2,...,n} \varnothing = \varnothing$; (2) $P^o = \cup_{i=1,2,...,n} p_i^o = \cup_{i=1,2,...,n} \{p_i\} = P$; (3) $P^- = \mathbb{R}^3 - (\partial P \cup P^o) = \mathbb{R}^3 - P$; (4) $\bar{P} = \partial P \cup P^o = P$.

3.3 Curve3D

The value type *Curve3D* is defined as the image of continuous mapping from 1D to 3D space. In particular, the domain in 1D space is usually taken as a closed interval $[0, 1]$ without losing generality. This is because any finite domain of closed interval can be mapped to $[0, 1]$ and therefore is homogeneous to $[0, 1]$. The shape of the curve can be straight or arbitrarily twisted in 3D plane given the condition that it is not *self-intersected*. Due to this fact, we call such kind of curve as a *simple curve*.

Spatial data type *Curve3D* is formally defined as

$$\begin{aligned} Curve3D = \{f([0,1]) \mid \\ (i)\ f([0,1]) \to \mathbb{R}^3 \text{ is a continuous mapping} \wedge \\ (ii)\ \forall a, b \in (0,1) : a \neq b \Rightarrow f(a) \neq f(b) \wedge \\ (iii)\ \forall a \in \{0,1\}, \forall b \in (0,1) : f(a) \neq f(b)\} \end{aligned} \quad (5)$$

where condition (i) specifies the continuous mapping $f : [0, 1] \to \mathbb{R}^3$, i.e. $\forall u \in [0, 1]$, $\exists v = f(u)$ where v is of type *Poi3D*. Condition (ii) is a strong constraint that the images of two different values within $(0, 1)$ must also be different. This means any intermediate point of the curve must be disjoint from all other intermediate points. This ensures that *Curve3D* does not intersect with itself. From condition (iii), the *end points* of the curve ($f(0)$ and $f(1)$) cannot coincide with any of the interior point along the same curve. However, this condition does not prohibit $f(0) = f(1)$. In the

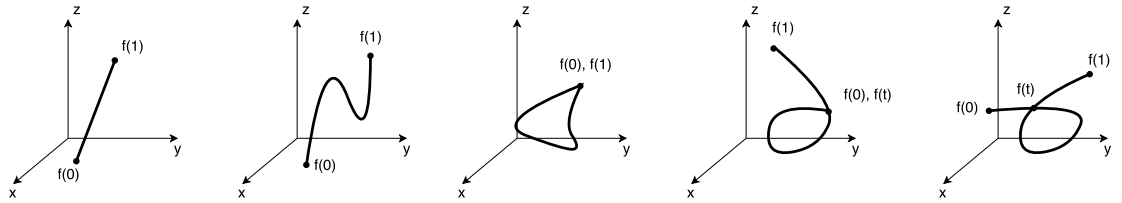

Figure 1: Examples ((a)(b)(c)) and Counter-examples ((d)(e)) of *Curve3D*'s.

case that end points coincide, the curve is called a *loop curve* or *closed curve*. Figure 1(a), 1(b), and 1(c) are typical examples of a spatial data type *Curve3D*. However, Figure 1(d) is not of type *Curve3D* since it allows the equality of an interior point with an end point. Figure 1(e) is also not a *Curve3D* as it intersects itself.

For a simple curve c with the mapping function f, the boundary of c is defined as the set of its end points, i.e. $\partial c = \{f(0), f(1)\}$. It is denoted that $e(c) = \{f(0), f(1)\}$. Note that this set can be a singleton set in the case of a loop curve. The interior of the curve includes all points contained in the curve except for the end points: $c^o = c - e(c)$. For the closure we obtain $\bar{c} = \partial c \cup c^o = c$. Therefore the exterior is defined as $c^- = \mathbb{R}^3 - c$.

Topological Relationships between *Curve3D*'s. Based on the 9-Intersection Model Egenhofer and Herring (1990), the topological relationships between two simple curves c_i and c_j are included in the following matrix:

$$R(c_i, c_j) = \begin{bmatrix} c_i^o \cap c_j^o & c_i^o \cap \partial c_j & c_i^o \cap c_j^- \\ \partial c_i \cap c_j^o & \partial c_i \cap \partial c_j & \partial c_i \cap c_j^- \\ c_i^- \cap c_j^o & c_i^- \cap \partial c_j & c_i^- \cap c_j^- \end{bmatrix}$$

Each intersection will be characterized by a value of *empty* ($\varnothing$) or *non-empty*($\neg\varnothing$). Therefore there can be $2^9 = 512$ different configurations. But not all of the configurations are realistic and meaningful. We are more interested in the following relationships which will be utilized in further spatial data type definitions:

1. *disjoint*: c_i and c_j are called *disjoint* if they do not have any points in common. To define it formally, we state $\forall a \in [0,1],\ \neg\exists b \in [0,1],\ s.t.\ f(a) = g(b)$;
2. *meet*: c_i *meets* c_j if they have only one shared end point. The equivalent conditions are
   (i) $\forall a, b \in (0,1),\ s.t.\ f(a) \neq g(b)\ \wedge$
   (ii) $\forall a, b \in \{0,1\},\ s.t.\ f(a) = g(b)\ \wedge\ f(|1-a|) \neq g(|1-b|)$;
3. *quasi-disjoint*: Two curves c_i, c_j are called *quasi-disjoint*, if their interiors do not intersect. They are allowed to meet in the endpoints. It is formally defined with the following conditions:
   (i) $\forall a, b \in (0,1),\ s.t.\ f(a) \neq g(b)\ \wedge$
   (ii) $\forall a \in (0,1), b \in \{0,1\},\ s.t. f(b) \neq g(a)\ \wedge g(b) \neq f(a)$;
4. *touch*: c_i *touches* c_j if, and only if
   (i) $\forall a, b \in (0,1),\ s.t.\ f(a) \neq g(b)\ \wedge$
   (ii) $\exists a \in (0,1),\ s.t.\ f(0) = g(a)\ \vee f(1) = g(a)\ \vee f(a) = g(0) \vee f(a) = g(1)$.

3.4 Branch3D

If we would represent curves in Figure 1(d) and (e) based on the abstract data type *Curve3D*, we need to separate the whole line into sub-curves so that each sub-curve is a *Curve3D* and they are interconnected. For example, the curve in Figure 1(d) can be decomposed into two simple curves: $f(0)$ to $f(t)$, and $f(t)$ to $f(1)$ while the curve in Figure 1(e) can be decomposed into three simple curves: $f(0)$ to $f(t)$, $f(t)$ to $f(1)$, and the loop with the coincide end point $f(t)$. Based on this decomposition, given the set of the simple curves $c_1, c_2, ..., c_n$ that contribute to the whole line, it is observed that

1. there can be more than two curves that form the shape;
2. no curves are disjoint with all other curves;
3. adjacent curves quasi-disjoint with each other with at least one coincide end point.

Geometrically, we call the set of these connected simple curves as *Branch3D*. Let C be the set of all simple curves in $\mathbb{R}^3$.

Spatial data type *Branch3D* is formally defined as

$$\begin{aligned} Branch3D = \{ & \cup_{i=1}^{n} c_i \mid \\ & (i)\ n \in \mathbb{N}, \forall i \in \{1,2...,n\} : c_i \in C\ \wedge \\ & (ii)\ \forall i \in \{1,2,...,n\}, \exists j \neq i,\ s.t.\ c_i\ meets\ c_j\ \wedge \\ & (iii)\ \forall i \neq j,\ c_i\ and\ c_j\ are\ quasi-disjoint\ \wedge \\ & (iv)\ n = 1 \vee \\ & \quad card(\{f_i \mid \forall k \in \{0,1\}, \\ & \quad card(\{f_j \mid j \in \{1,2,...,i-1,i+1,...,n\} \wedge \\ & \quad (f_j(0) = f_i(k) \vee f_j(1) = f_i(k))\}) = 1\}) \geq 2\} \end{aligned} \tag{6}$$

where condition (i) ensures all components of *Branch3D* are simple curves; condition (ii) posts the restriction that for any curve, there exists another one

that meets this curve in the same branch; condition (iii) emphasizes the non-intersection relationships between all curves; condition (iv) makes sure that each *Branch3D* should have at least two curves that have one of the end points isolated and does not belong to any other curves. In other words, each branch cannot be wrapped by two loop curves.

Figure 2(a) and (b) gives two examples of data type *Branch3D*. However, Figure 2(c) does not form a *Branch3D* as it breaks condition (iv).

Let B be the set of all *Branch3D*'s. For a branch $b = \{\cup_{i=1}^{n} c_i\} \in B$ with corresponding mappings $f_1, f_2, ..., f_n$, the topological notations are defined as: (1) As $\cup_{i=1}^{n} \partial c_i = \cup_{i=1}^{n} e(c_i) = E(b)$, note that we need to exclude the internal end points from $E(b)$ to formulate the boundary. Therefore, $\partial b = E(b) - \{p \in E(b) \mid card(\{f_i \mid i \in \{1,2,...,n\} \wedge (f_i(0) = p \vee f_i(1) = p)\}) \neq 1\}$; (2) The interior of the branch includes all points from all curves except for boundary points, i.e. $b^o = b - \partial b$; (3) The closure of the branch is $\bar{b} = \partial b \cup b^o = b$; (4) The exterior of the branch is then defined as $b^- = \mathbb{R}^3 - b$ since $\mathbb{R}^3$ is the embedding space.

As for *Branch3D* data type, we will now give the corresponding structural definition. For a branch $b = \{\cup_{i=1}^{n} c_i\}$, let $A = \{A_i\}, 1 \leq i \leq n$ be the corresponding label type for each of the curve. Therefore

$$interior : \forall p \in c_i^o \in b, \pi(p) = \{A_i\}; \quad (7)$$

$$boundary : \forall p \in \partial b, card(\pi(p)) > 1; \quad (8)$$

$$exterior : \forall p \in b^-, \pi(p) = \{\perp\}. \quad (9)$$

Label space $\mathbb{L} = 2^A = \{\{A_1\}, ..., \{A_n\}, \{A_1, A_2\}, \{A_1, A_3\}, ..., \{A_1, A_2, A_3\}, ..., \{A_1, A_2, ..., A_n\}, \{\perp\}\}$. The interior of *Branch3D* will be mapped to singleton set in label space excluding $\{\perp\}$. Formula (8) confines that any point that belongs to the boundary of b will be mapped to a subset of 2^A with more than one element. The exterior is labelled with $\{\perp\}$. As shown in Figure 2(d), the branch object is consist of five curves with labels A, B, C, D, and E. The boundary points are mapped to the label space with more than one element while the exterior of the branch is mapped to $\{\perp\}$. Note that the structural definition of *Curve3D* is a special case of *Branch3D*.

3.5 Hub3D

Hubs are important components of spatial networks. Generally speaking, hubs are locations and areas that connect different branches together and may also function as a boundary of the spatial network. In our abstract model, the concept of hub represents a geometric layout and its coverage. Examples of hubs are numerous, from metro stations in road networks to lakes in in-land river networks, from power switch in grid networks to airport in aviation networks. Note that the hubs can be formed by humans or by nature, as long as they belong to the spatial network.

Geometrically, *Hub3D*'s are three-dimensional regions that connect with one or more branches. And in our model, we do not emphasize (or intend) to model the internal structures of hubs. Thus, *Hub3D* is seen as a black box. Before giving the definition of *Hub3D*, we need to define a *simple region* and *Hub2D* data types in 2D space. In the simplest case, a region object consists of a single connected component and is called a *simple region*. The point set in the 2D Euclidean space representing a simple region object is connected, closed, and bounded. The formal definitions of these features can be found in Liu and Schneider (2010). Due to space limit, we skip the definition of *SRegion2D*, *SRegion3D* and move to the definition of *Hub2D* and *Hub3D*. These omitted definitions can be found in our next extension.

In general, *Hub2D* is allowed to contain *none*, *one*, or *more than one* holes by condition (i). Condition (ii) ensures that the holes are contained in the region object and we do not allow holes to intersect each other (in the case of intersection, we could combine two holes into one). Condition (iv) makes *Hub2D* different from a normal region in the sense that it requires the region to connect to a certain branch object at only one end point. There is no limitation of the number of branches that can connect to the region, as long as they touch the region on the boundary. Based on previous definitions, now we are able to define *Hub3D* object as

$$\begin{aligned} Hub3D = \{h(U) \mid \\ (i)\ \forall U \in HUB2D,\ h : U \to \mathbb{R}^3 \wedge \\ (ii)\ h\ is\ a\ continuous\ mapping\} \end{aligned} \quad (10)$$

The above definition is a *lift* from *Hub2D* to $\mathbb{R}^3$. Formally, we can summarize and give the definition of spatial data type *Hub3D* as

$$\begin{aligned} Hub3D = \{R \subset SR3D \mid \\ (i)\ \forall R_i \in SR3D, i \in \mathbb{N}, \exists Q \in SR3D,\ s.t. \\ R = Q - \cup_{i=0}^{n} R_i\ where\ n \in \mathbb{N} \wedge \\ (ii)\ \forall i \in \{0,1,...,n\}, Q\ contains\ R_i \wedge \\ (iii)\ \forall i \neq j,\ R_i\ and\ R_j\ are\ disjoint \wedge \\ (iv)\ \exists b \in B, card(\{p \in \mathbb{R}^3 \mid \\ (\exists curve\ c \in b, f_c(0) = p \vee f_c(1) = p) \wedge \\ p \in \partial R\}) = 1\} \end{aligned} \quad (11)$$

Similar to *Hub2D*, *Hub3D* type can also contain holes or cavities inside. And condition (iv) requires *Hub3D* to connect to at least one branch in $\mathbb{R}^3$.

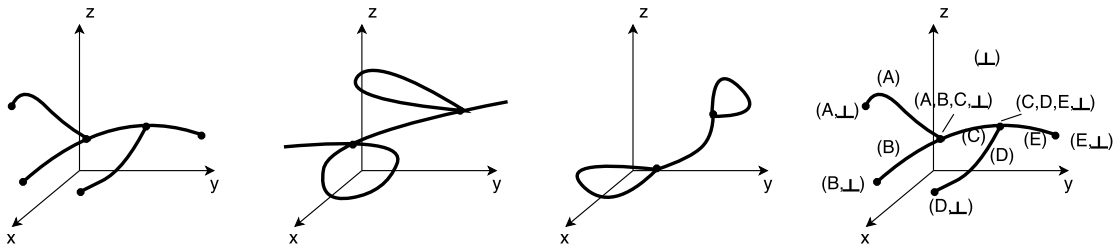

Figure 2: Examples ((a)(b)(d)) and Counter-example ((c)) of *Branch3D*'s.

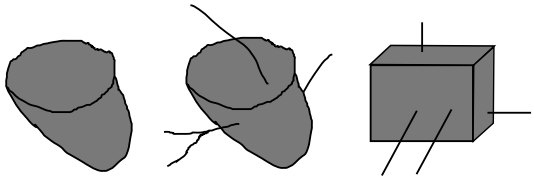

Figure 3: (a) *SRegion3D* Example; (b)(c) *Hub3D* Examples.

Figure 3(a) depicts a simple region in $\mathbb{R}^3$. The shape of a hub can be arbitrary, such as a naturally formulated lake (Figure 3(b)); it can also be regulated by humans, like a pipeline hub or metro station (Figure 3(c)). Hub entities have a common feature that they are not isolated in the network otherwise they themselves form networks since they are not connected to outside world. For example, a lake that belongs to a river network has at least one river stream flowing to/out from it; a pipeline hub gathers the pipelines from different directions, process, filter, and then transfers out gas/oil; a metro station connects to multiple lines of traffic and public transportation networks.

3.6 Junction3D

In spatial networks, there can be multiple pathways that connect one point to another. Each pathway can be a simple curve in 3D space, or a series of curves, i.e. we can use *Branch3D* or more generally, *Trunk3D* data type to represent a pathway. There are certain important points in the network layout that are connected with a number of pathways, we refer to those points as *junctions*. A junction in a 3D network is denoted as of type *Junction3D*.

The definition of a *Junction3D* from a geometric perspective is

$$\begin{aligned} Junction3D = \{J \subset \mathbb{R}^3 \mid \\ \forall p \in J, card(\{t \mid t \in T, \exists c \in t,\ s.t. \\ (f_c(0) = p \vee f_c(1) = p)\}) > 3\} \end{aligned} \tag{12}$$

where C is the set of all simple curves in $\mathbb{R}^3$. From the definition, we require that the number of distinct curves that connect to the junction point should be more than three. If there are less or equal to three curves connecting to a point, that point is not seen as a *junction* in our model as it could also be an internal point in the *Branch3D* object. As shown in Figure 5, point A is not a junction as it only connects to three curves. Similar case holds for point B. Point C and D are junctions since they are linked with more than three simple curves. For points with degree of three, we name them as *fork* points, or *forks* for short. We use $J(\pi)$ to represent the set of junctions for the spatial mapping π on-wards.

3.7 Trunk3D

After giving the definitions of branches, hubs, and junctions, now we are able to define a more complex spatial data type, *Trunk3D*. A collection of pathways between two junctions or hubs forms a *bunch of branches*, which we refer to as a *Trunk3D*.

The formal definition of spatial data type *Trunk3D* is given as

$$\begin{aligned} Trunk3D = \{\cup_{i=1}^{n} b_i \mid \\ (i)\ n \in \mathbb{N}, \forall i \in \{1,2...,n\} : b_i \in B \wedge \\ (ii)\ \forall i,j \in \{1,2,...,n\}, \forall a,b \in (0,1),\ s.t. \\ \forall c_1 \in b_i, c_2 \in b_j : f_{c_1}(a) \neq f_{c_2}(b) \wedge \\ (iii)\ \forall i,j \in \{1,2,...,n\}, \forall a \in (0,1), \forall b \in \{0,1\}, \\ s.t.\ \forall c_1 \in b_i, c_2 \in b_j : \\ f_{c_1}(b) \neq f_{c_2}(a) \wedge f_{c_2}(b) \neq f_{c_1}(a) \wedge \\ (iv)\ \forall p \in \partial t,\ t\ is\ a\ Trunk3D, \\ (\exists b \in B,\ s.t.\ p \in \bar{b} \wedge b \notin t) \vee (p \in J(\pi)) \vee \\ (\exists h \in HUB3D,\ s.t.\ p \in \partial h) \vee (p \in \lambda(\pi))\} \end{aligned} \tag{13}$$

where B is the set of branches in $\mathbb{R}^3$. Recall that two curves that belong to one branch *meet* each other if they have only one shared end point. However, we lift this constraint in defining *Trunk3D*. As in condition (ii) and (iii), all branches are required to not intersect with each other if they belong to the same trunk, but they are allowed to formulate loops. Condition (iv) states that for any end points of the trunk,

they have to either *touch* or *meet* the other *branch*, or they belong to the *boundary* of a *hub*, or they are *junctions*, or they belong to the boundaries of the network, which is denoted by $\lambda(\pi)$. $\partial Trunk3D$ represents the boundary of the trunk.

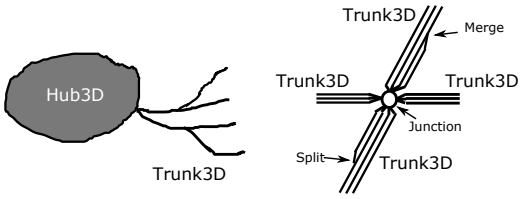

Figure 4: Examples *Trunk3D*'s.

Examples of *Trunk3D*'s are shown in Figure 4. In Figure 4(a), the branches originated from the hub object form a single *Trunk3D* object. We call the points that a *Trunk3D* touches a *Hub3D* as *estuary points*. Figure 4(b) depicts a junction in a road network. The two roads that intersect at this junction are modeled into four pieces of *Trunk3D*'s. Having the *Trunk3D* definition is important to precisely model the geometrical aspects of spatial networks. For example, in a pipeline network, between two interchange stations, there can be multiple pipelines connecting them. These pipelines are usually parallel to each other and do not intersect each other. Since their start points coincide, and end points also coincide, it is impossible to model these lines as a normal *Line3D* object Schneider (1997) or a *Branch3D* object. Another example is in road networks. For a highway or an interstate, it usually consists of multiple lanes. In our abstract model, we use data type *Branch3D* to model the lanes. Lanes can merge or split, but if we treat each lane as an individual branch, the topological relationships of these branches are that they do not intersect each other. We treat those lanes as a new type of spatial data type, i.e. *Trunk3D*. In addition, in road networks, we model junctions (such as the locations of traffic lights) as points, therefore the lanes that connect with junctions are constricted to meet at junctions and then spread out again (Figure 4(b)).

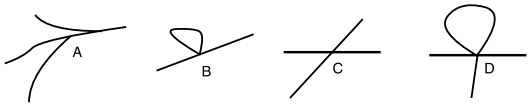

Figure 5: Examples and Counter-examples of *Junction3D*'s.

Let T be the set of all *Trunk3D*'s in $\mathbb{R}^3$. For a trunk $t = \{\cup_{i=1}^{n} b_i\} \in Trunk3D$ where $b_i \in B$, the topological notations of this trunk is defined as: (1) Let $E(t) = \cup_{i=1}^{n} \partial b_i = \cup_{i=1}^{n} \cup_{j=1}^{m} e(c_j^{b_i})$, note that we need to exclude the internal end points from $E(t)$ to formulate the boundary of t. Therefore, $\partial t = E(t) - \{p \in E(t) \mid \forall c \in C, \neg\exists b \in t,\ s.t.\ c \in b \wedge (f_c(0) = p \vee f_c(1) = p)\}$; (2) The interior of the trunk includes all points from all belonging branches except for boundary points, i.e. $t^o = t - \partial t$; (3) The closure of the trunk is $\bar{t} = \partial t \cup t^o = t$; (4) The exterior of the trunk is then defined as $t^- = \mathbb{R}^3 - t$ since $\mathbb{R}^3$ is the embedding space.

3.8 Snet3D

With all previous definitions of different components of the network, now we give the formal definition of the spatial network data type, *Snet3D*.

The spatial data type *Snet3D* for spatial networks is defined as

$$Snet3D = \{S \subset \mathbb{R}^3 \mid \forall u \in S, \exists v = u,\ s.t.\ v \in HUB3D \vee v \in T\} \quad (14)$$

where $HUB3D$ is the set of hubs, and T is the set of trunks. The definition states that for any point that belongs to the spatial network in 3D space, it should either belong to any hubs or trunks in the network. Geometrically, $S = HUB3D \cup T$.

For a spatial network s, the topological notations are specified as follows: (1) The boundary of s is defined as $\partial s = \partial HUB3D \cup \partial T = (\cup_i \partial h_i) \cup (\cup_j \partial t - J - \{p \mid \exists h \in HUB3D, \exists t \in T,\ s.t.\ p \in \partial h \wedge p \in \partial t\})$, which is the union of the boundary of all hubs and trunks excluding junctions and estuary points; (2) The interior of s is then defined as $s^o = s - \partial s$; (3) The closure of a spatial network s is $\bar{s} = s^o \cup \partial s = s$; (4) The exterior of s is defined as $s^- = \mathbb{R}^3 - s$.

Finally, we give the structural definition of spatial networks . Let A be the label type, and π be the spatial mapping

$$interior : \pi^o = \cup_{\{s \in \gamma(\pi) | \pi[s] \neq \perp\}} s\}; \quad (15)$$

$$boundary : \partial\pi = \cup_{b \in \omega(\pi)} b\}; \quad (16)$$

$$exterior : \pi^- = \pi^{-1}(\{\perp\}) = \cup_{\{s \in \gamma(\pi) | \pi[s] = \perp\}} s\}. \quad (17)$$

Recall from (1)(2) that the *areas* of π is defined as $\gamma(\pi) = \pi^{-1}(rng(\pi) \cap \{X \in 2^A |\ card(X) = 1\})$ and the *borders* of π is defined as $\omega(\pi) = \pi^{-1}(rng(\pi) \cap \{X \in 2^A |\ card(X) > 1\})$. The *interior* of π (formula (18)) is defined as the union of π's areas. The *boundary* of π (formula (19)) is defined as the union of the borders of π. And the *exterior* of π is the block mapped to $\perp$.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an abstract model of spatial networks. This model is based on point set theory and topology that offers a formal data type definition of spatial networks in the form of spatial mapping. The model provides rich details that are generic to all kinds of spatial networks.

Future work includes the completion of the abstract model with a comprehensive set of spatial networks operations and predicates. These operations and predicates along with the data types definitions form the foundation of the type system of spatial networks. The only trouble with abstract models is that we cannot store and manipulate them in computers. The challenge of representing and storing abstract spatial data types is addressed by the next level of modeling: discrete modeling. We can view discrete models as *approximations*, finite descriptions of the infinite shapes we are interested in. In spatial databases there is the same problem of giving discrete representations for in principle continuous shapes Erwig et al. (1998); there almost always linear approximations have been used. Hence, a region is described in terms of polygons and a curve in space (e.g. a river) by a polyline. Linear approximations are attractive because they are easy to handle mathematically; most algorithms in computational geometry work on linear shapes such as rectangles, polyhedra, etc.

Based on the finite representation, an implementation of the proposed spatial network model is the next step. In particular, an implementation in a database context is expected. Additionally, an extension of SQL query language is to be designed and implemented to support querying of the spatial networks.

REFERENCES

Brinkhoff, T. (2002). A framework for generating network-based moving objects. In *Geoinformatica*, volume 6, pages 153–180.

Ding, Z., Yang, B., Güting, R., and Li, Y. (2015). Network-matched trajectory-based moving-object database: Models and applications. *IEEE Transactions on Intelligent Transportation Systems, Article in press.*

Dugundi, J. (1966). *Topology*. Allyn and Bacon.

Egenhofer, M. J. and Herring, J. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. In for Geographic Information, N. C. and Analysis, University of California, S.B.S.B. C., editors, *Tech. rep.90-12.*

Erwig, M., Güting, R. H., Schneider, M., and Vazirgiannis, M. (1998). Abstract and discrete modeling of spatio-temporal data types. In *6th ACM Symp. on Geographic Information Systems(ACM GIS)*, pages 131–136

Erwigm, M. and Güting, R. (1994). Explicit graphs in a functional model for spatial databases. *Knowledge and Data Engineering, IEEE Transactions on*, 6(5):787–804.

Gupta, S., Kopparty, S., and Ravishankar, C. (2004). Roads, codes, and spatiotemporal queries. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 115–124.

Güting, R. (1994). Graphdb: Modeling and querying graphs in databases. In *20th Int. Conf. on Very Large Databases*, pages 297–308.

Güting, R., Almeida, V., and Ding, Z. (2006). Modeling and querying moving objects in networks. In *The VLDB Journal*, volume 15, pages 165–190.

Jensen, C., Pedersen, T., Speicys, L., and Timko, I. (2003). Data modeling for mobile services in the real world. In *Int. Conf. on Advances in Spatial and Temporal Databases*, pages 1–9.

Jeung, H., Yiu, M., Zhou, X., and Jensen, C. (2010). Path prediction and predictive range querying in road network databases. In *The VLDB Journal*, volume 19, pages 585–602.

Krogh, B., Pelekis, N., Theodoridis, Y., and Torp, K. (2014). Path-based queries on trajectory data. In *22nd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*.

Liu, H. and Schneider, M. (2010). Detecting the topological development in a complex moving region. *Journal of Multimedia Processing and Technologies (JMPT)*, 1(3):160–180.

McKenney, M. and Schneider, M. (2007). Spatial partition graphs: A graph theoretic model of maps. In *Advances in Spatial and Temporal Databases*, pages 167–184. Springer.

Miller, H. and Shaw, S. (2001). *Geographic Information Systems for Transportation*. Oxford University Press.

Popa, I. S., Zeitouni, K., Oria, V., Barth, D., and Vial, S. (2011). Indexing in-network trajectory flows. In *The VLDB Journal*, volume 20, pages 643–669.

Qi, L. and Schneider, M. (2012). Monet: Modeling and querying moving objects in spatial networks. In *3rd ACM SIGSPATIAL Int. Workshop on GeoStreaming (IWGS)*.

Qi, L., Zhang, H., and Schneider, M. (2015). Design and representation of complex objects in database systems. In *23rd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*.

Scheider, S. and Kuhn, W. (2008). Road networks and their incomplete representation by network data models. In *GIScience '08: Proceedings of the 5th International Conference on Geographic Information Science*, pages 290–307.

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