Modeling and Querying Spatial Networks in Databases

Virupaksha Kanjilal, Markus Schneider
University of Florida
USA
{vk4, mschneid}@cise.ufl.edu

ABSTRACT: *Infrastructures like transportation, power, and pipeline networks which are characterized by a spatial embedding are known as spatial networks. Spatial networks are ubiquitous in our everyday life and used in transport, navigation, and city planning as well as in Geographical Information Systems (GIS) and other geo-spatial applications. The largely increasing amount of generated data about spatial networks can only be efficiently stored and analyzed in a database system. However database support for large spatial networks in order to represent, store, query, and manipulate them is rare. This article aims to provide a conceptual, abstract, and formal model of spatial networks, called Spatial Network Algebra (SNA). It includes data types, operations, and predicates and serves as a specification for their later implementation in spatial database systems and GIS. Finally, we show how our spatial network concepts can be embedded into an SQL-like query language.*

Keywords: Spatial network, Spatial network algebra, Spatial network query language, Spatial database, GIS

Received: 31 May 2010, Revised 31 June 2010, Accepted 5 July 2010

1. Introduction

Spatial networks like road networks connecting cities, rail networks connecting railway stations, and pipeline networks supplying water to our houses are a ubiquitous spatial concept in our everyday life. They are *spatially embedded graphs* created by the interconnection of spatial elements such as spatial lines and spatial points. Their applications comprise geoscience disciplines like geography and cartography as well as related disciplines like Geographical Information Systems (GIS), spatial databases systems, location-based services, transportation, GPS-based navigation, traffic forecasting, and mobile computing, to name only a few.

The increasing interest in and importance of spatial networks has led to a large increase of generated spatial networks data. Consequently, 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. Spatial databases, which form the data storage foundation of geographical and GIS applications, only offer *spatial data types* [Schneider 1997 [20]] for single spatial objects like *points*, *lines*, and *regions* but are unable to adequately represent connectivity structures like spatial networks.

Therefore, current spatial systems that aim to represent spatial networks usually have a three layered architecture. A *data layer* that is typically implemented as a spatial database stores the basic components of a network in terms of points and line segments. A *middleware layer* loads the basic components of a network from a database into main memory and assembles them to a topological data structure outside of the database. Network operations are then executed on this data structure. *A visualization layer* at the top provides a graphical user interface to visualize a spatial network. This approach has two main drawbacks. First, in the data layer, it spreads the spatial components of a network like points and lines over various tables in

This work was partially supported by the National Science Foundation under grant number NSF-IIS-0812194.
Authors' addresses: Virupaksha Kanjilal and Markus Schneider, University of Florida, Department of Computer & Information Science & Engineering, PO Box 116120, Gainesville, FL 32611

the database. Hence, a spatial network is not visible and not accessible as a self-contained entity or object in the database. There is currently no concept of an explicit data type for spatial networks in a database system. Hence, spatial networks are not first class citizens in a spatial database. Second, in the middleware layer, this approach fails to take advantage of the many benefits provided by a database system like query formulation, query processing, concurrency control, recovery, and transaction management.

Consequently, providing a spatial data type for spatial networks by means of an abstract data type that is integrated into a database system would solve the aforementioned drawbacks of existing spatial systems and allow a user to query and manipulate spatial networks in a database setting by high-level operations defined on them. Spatial networks would then become first-class citizens in the database. In order to achieve such a data type integration, we need a formal model of spatial networks that will serve as the specification for any implementation. A major reason for the lack of a data type for spatial networks in databases is that it has not yet been formally defined due to the inherently complex nature of these networks.

The first goal of this article is the design of an abstract formal data model that is based on point set theory and point set topology and that offers a formal data type definition of spatial networks and of high-level operations on them. Spatial networks are modeled as particular point sets of the Euclidean plane. Each network point has thematic properties associated to it. The thematic properties distinguish the different components of a network, that is, all the points with equal thematic properties belong to the same network component. Interior and exterior points of a network can be distinguished based on the thematic properties. In particular, this approach supports two views. It enables us to consider attributes of single points (space based view) but also provides access to collections of points having equal attributes (object based view). The second goal is the design of a *spatial network query language* (SNQL). We show some queries in this SQL-like query language that takes advantage of the operations and predicates on a spatial network. The model presented in this article is part of a comprehensive algebra for *spatially embedded graphs*; it is called the *Spatial Network Algebra* (SNA).

Section 2 gives an overview of the various approaches to modeling spatial networks. Section 3 introduces the Spatial Network Algebra as our formal model of spatial networks. Section 4 defines a number of important geometric and metric operations in our spatial network model. In Section 5, we introduce an SQL-like query language that can be used to specify and query spatial networks. Finally, we draw some conclusions and sketch future work in Section 6.

2. Related work

Data modeling involves three levels of abstraction: the conceptual level, the logical level, and the physical level. In this section we give the background on earlier approaches to dealing with the conceptual modeling of spatial networks and incorporating them into databases. Data models for networks in GIS are built around two entities, namely nodes which are zero-dimensional entities and arcs which are one-dimensional entities. The planar embedding of the node-arc data model ensures topological consistency in the network. Since nodes and arcs correspond to the vertices and edges of a graph, it makes sense to view and model a spatial network as a directed graph $G = (V, E)$ where V is a set of nodes and $E \subseteq V \times V$ is a set of edges. Graph models are popular as they can capture the structure and connectivity of spatial networks. Queries like the *shortest path query* or the *maximum flow* query can be directly mapped to well known graph problems for which algorithms exist. For example, the shortest path problem may be solved by Dijkstra's algorithm, and the maximum flow problem can be solved by the Ford Fulkerson algorithm. The graph view of spatial networks has, for example, been used in [Brinkhoff 2002 [1]; Frentzos 2003 [6]; Gupta et al. 2004 [8]; Güting 1994 [10]; Jeung et al. 2010 [13]; Shekhar and Yoo 2003 [22]]. The authors in [Jensen et al. 2003 [12]] point out that the graph modeling of a spatial network is not appropriate as it does not present a realistic representation of the complexity of such a network in the real world. Graph models have been studied in the literature, for example, in [Cruz et al. 1987 [3]; Gyssens et al. 1994 [11]; Mannino and Shapiro 1990 [15]]. These approaches aim at separating the spatial and thematic data into different data models. A logical spatial data model encodes the nodes and arcs and maintains the geometry while the associated thematic information is stored in relational database tables. The connection between the records in the relational tables and the spatial objects is achieved by employing unique identifiers. This hybrid data management strategy has been developed to take advantage of a relational database management system in order to store and manipulate thematic attribute information [Longley and Rhind 2001 [14]]. This solution does not allow that relationships between a spatial object and the thematic data have their own attributes [Goodchild 1998 [7]]. Though effective, it has been shown that this solution is neither elegant nor robust [Miller and Shaw 2001 [17]].

Though graph modeling of spatial networks can capture their structure and connectivity, they are unable to represent the spatial embedding aspect. An alternate modeling approach is to have a graph representation in which each vertex is associated with a spatial embedding called *partially embedded graph* [Meschini et al. 2007 [16]; Scheider and May 2007 [18]]. The model

of a multimodal transport network has been developed in [Meschini et al. 2007 [16]]. It represents the network as a directed graph in which the vertices are associated with a spatial location characterized by geographic coordinates in the Euclidean plane. The geometries of the edges are not stored in either of the models but simply specified by a pair of (initial and final) vertices. The work in [Scheider and Kuhn 2008 [19]] models road networks as partially embedded graphs and defines road components based on their properties and the potential actions they may perform. The authors term this as an "affordance-based" theory of networks. These approaches are unable to distinguish between crossover points like a tunnel beneath a road or an overpass over a road and can lead to wrong results in shortest route calculations.

Routes correspond to roads or highways in real life and to paths in graphs. They are important conceptual entities as addresses are given relative to routes. Moreover, it is easy to relate network positions to routes. A number of approaches use routes as the base for defining a network [Güting et al. 2006 [9]; Erwig and Güting 1994 [4]]. A network is modeled as a set of routes and a set of junctions between routes. A route description consists of a route identifier, length information, a curve describing its geometry, and a route type. Junctions are described as positions connecting routes. Junction points are associated with "connectivity codes" which encode allowable movements at the junctions. This approach to modeling a network allows routes and junctions to be stored as records in a relational table, but a single network entity is not created; instead, the network is spread across numerous tables. Besides a lack of a network identity, this leads to serious performance problems since all data have to be loaded into the main memory first.

A combination of multiple representations of two-dimensional transport networks stored in *multi representational databases* (MRDB) has been provided in [Speičys and Jensen 2008 [23]; Ulugtekin et al. 2004 [24]]. The framework includes a semantically rich two-dimensional representation of a transportation network, a weighted multi-graph representation of a transportation network, and a mapping of instances of the former to instances of the latter. This mapping demonstrates how it is possible to represent complex transportation network by semantically simple graphs. The graph model and mapping are built to accurately capture the topologies of transportation networks and to also serve as a foundation for query processing. This model serves well for transport networks which have a fixed set of features like turns, roundabouts, freeways, etc. Different types of networks have a wide variety of properties and attributes which has to be represented. In our model, we assume that every point of the network is marked with its thematic attributes, and we aggregate all points with equal attributes to form network components. This enables the model to be truly generic as it can represent any type of spatial network.

3. A Conceptual Model of Spatial Networks

In this section we introduce our model of spatial networks. We begin by providing an intuitive description of spatial networks in Section 3.1. In Section 3.2 we give the formal definitions of spatial network components and the spatial network itself.

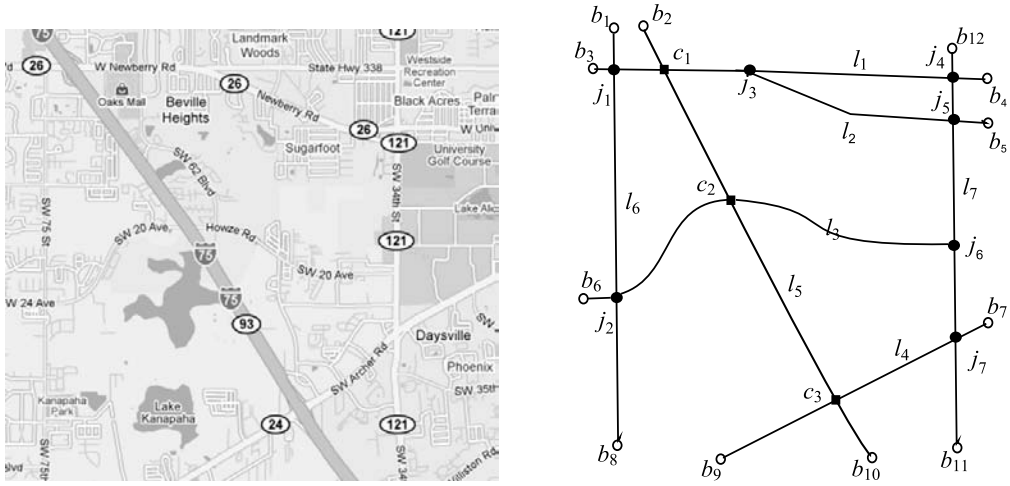
3.1 What are Spatial Networks?

Spatial networks area ubiquitous spatial concept. We use transportation networks like road networks for cars, buses, and taxis or railway networks for trains and metro every day. Water pipelines and power networks supply our houses with valuable resources. If we abstract from these particular kinds of networks, we can say that the primary purpose of spatial networks is to provide a spatially constrained and rather static environment for materials (in the broadest sense) to dynamically move or flow through them.

From a modeling perspective, a spatial network can be seen as a *spatially embedded* and *labeled* graph. This means that a spatial network has geometric and thematic aspects. Aspatially embedded graph is a graph whose vertices are mapped to points in space and whose edges are mapped to arcs which have end points that are images of two vertices [Fleming and Mellor 2006 [5]]. Geometric aspects comprise junctions, channels, boundary points, and cross over points that are embedded in space and characterized by precise locations. Thematic aspects specify the semantics (like names, functions, or properties) of geometric components and are attached to them in terms of annotations or *labels*. Figure 1a shows a map depicting a network of roads. Figure 1b is our simplified representation of the same road network. Some important roads from Figure 1a have been abstracted as curves which form the edges of the graph. There are various points of interest on the network which are analogous to nodes in a graph. We term these points as *interaction points*; they have been marked by filled and unfilled circles and rectangles in Figure 1b. Interaction points include junction points, crossover points, and boundary points. *Junction points* are network components which mark locations where two (or more) roads intersect. The point j_1 in Figure 1b is a junction point because at this point the roads *W Newberry Road* (shown as l_1 in Figure 1b) and *SW 75th street* (shown as l_6 in Figure 1b) intersect. Cars can switch from one road to another only at junction points. Junction points are marked with filled circles in Figure 1b. This is in contrast to *crossover points* where two roads interact but one road forms a bridge over the other. For example, points c_1, c_2, and c_3 in Figure 1b are crossover points because at these points, the *Interstate 75* (shown

as l_5 in Figure 1b) forms a bridge over *W Newberry Road, SW 20 Avenue,* and *SW Archer Road* (represented as l_1, l_3 and l_4 respectively in Figure1b). Cars cannot switch directly from one road to another at cross over points as the interacting roads are separated by a vertical gap. Crossover points are depicted as filled rectangles in Figure 1b. While other models ignore crossover points, our model considers them since operations like the shortest path operation lead to wrong results if cross-over points are ignored and treated as intersection points. Junction points and crossover points are not mutually exclusive; both may exist at the same point. As an example, three roads may interact at the same point, but only two of them actually intersect forming a junction, while the third one may form a bridge over it. Junction points and crossover points are identified in a 2D representation of a spatial graph by the degree of their nodes. They both have a node degree more than one. But it is not possible to distinguish between a junction point and a crossover point from the spatial graph alone due to the flat, two dimensional network representation. *Boundary points* are another kind of interaction points whose node degree is equal to one. The points b_1,...,b_{11} in Figure 1b are boundary points as they form the limiting points (extremities) of the network; they are drawn as unfilled circles in Figure 1b.

The interaction points in a spatial network are connected to each other by what we term as channels; they are the passages through which material is transported. A finite sequence of consecutively connected edges in a spatially embedded graph forms a single channel. In Figure 1a, the roads represent the channels. The pipes in a pipeline network and the tracks in a railway network are examples of channels. A channel has exactly two end points. An end point of a channel may be a junction point and thus link at least two different channels, or it forms an extremity of the network and is thus a boundary point. As an example, one of the end points of channel l_3 connects to channel l_7 at the junction point j_6. On the other hand, if an end point does not link at least two different channels, it represents an extremity of a network and is hence a boundary point. All the points marked by hollow circles in Figure 1b are boundary points of the network.



(a) (b)

Figure 1. An example of a spatial network (b) extracted from a map (a)

Channels and interaction points are not only specified by the geometries of their edges. In addition, they are defined by names attached to their geometries. For example, highway I75 cannot be identified in a network by its geometry alone. To identify it, we have to locate and gather all the points in the Euclidean plane which have the name "I75". In our spatial network model, we assign thematic attribute values to all points of a channel. These attribute values can be used to identify network components like channels and interaction points. We call these values *labels*, and each point in the Euclidean plane is mapped to a set of labels. Any label is a tuple of the form (*id_attr, theme_attr*). The *id_attr* part of a label is the *channel identifier* and uniquely represents a particular channel in a network. This can be a channel name (for example, I75) or a channel number (for example, pipe no. 6). All points in the Euclidean plane having the same channel identifier (*id_attr* part) in their labels are part of the same channel. The *id_attr* part is always at the first position of a label. The *theme_attr* part of a label models the thematic attributes of a point in the plane. A *theme_attr* value may have a simple type such as *integer* or *string*, or it may have a complex type whose values represent combinations of *n* values, for example. Examples of thematic attributes range from speed limits of a road to the capacity of an oil pipe. Hence, we can model, for example, that the same road has different speed limits in different sections of this road.

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 $\perp_{A\cup B}$ (that is, we take the coalesced sum). If no ambiguities can arise, we sometimes omit the type index and simply use $\perp$. In all network visualization tools, coloring and markings differentiate channels. This is similar to our assignment of labels to points in a spatial network.

3.2 Definition of Spatial Networks

As we have motivated in the previous subsection, a spatial network is a spatially embedded and labeled structure. We assume that each point in the Euclidean plane is associated with a semantically relevant thematic label. Different points can carry the same label. We call this many-to-one mapping between spatial points and labels *spatial mapping*. A correct assignment of labels to points in the Euclidean plane helps us identify channels, junctions, and crossovers, and also distinguish coexisting junctions and crossovers. Junction points and crossover points are formed by the interaction of two or more channels at a point in the plane; thus, these points are labeled by the combination of the labels of the interacting channels. Junction points express the connectivity of the network. This means, for example, that a car standing at a junction of a road is able to go to any of the roads connected to that particular junction. In order to maintain the connectivity information, a set of channel labels for a point indicates a junction formed by the channels belonging to the labels in the set. Figure 2 shows the road network from Figure 1 with an appropriate labeling. The road network shown has seven roads represented in Figure 1 by the channels with the labels from the range $l_1,...,l_7$. Thus, in this case, the road network is a spatial mapping of type A where $A = \{l_1, l_2, l_3, l_4, l_5, l_6, l_7\}$. The junction point j_1 in Figure 1 is formed by the intersection of the channels l_1 and l_6. We represent it by the set $\{l_1, l_6\}$ and put this set into a set for reasons we will see later; that is, we obtain the set $\{\{l_1, l_6\}\}$. Since both channels l_1 and l_6 are present at the point j_1, we collect their labels associated with this junction point. If such a set contains only one label, the corresponding point belongs only to a single channel without a junction involved. Hence, it is an interior, non-shared point or a boundary point of a channel.

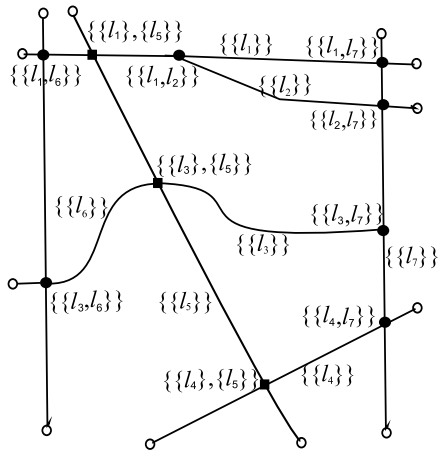
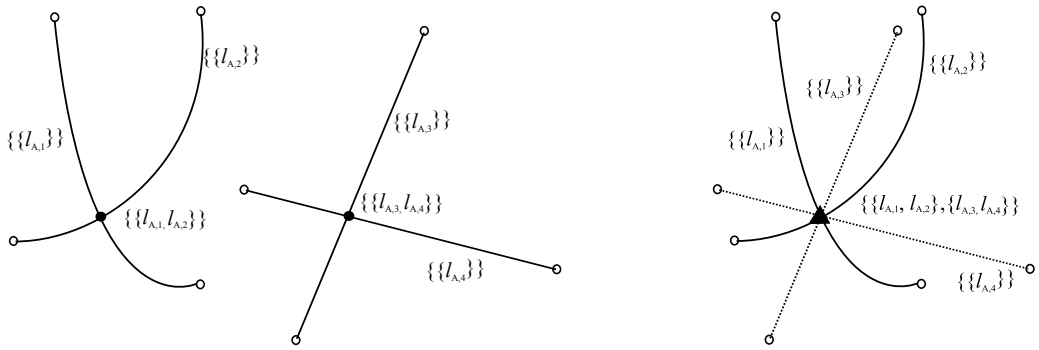


Figure 2. The spatial network of type *A* from Figure 1b with labels

A crossover point means that two or more channels interact but they do not geometrically join. We regard the explicit modeling of crossover points as important since operations like the shortest path operation lead to wrong results if crossover points are interpreted as junction points. A crossover point is modeled as a set of disjoint sets of labels. For example, the crossover point c_1 shown in Figure 1b is formed by the interaction of the channels l_1 and l_5. At this point, the interacting channels are represented as the singleton label sets $\{l_1\}$ and $\{l_5\}$. Hence, the label for the crossover point c_1 is represented as the set of these singleton sets, namely $\{\{l_1\}, \{l_5\}\}$ (Figure 2).

Sometimes, junctions and crossovers coexist. This means that some channels can form a bridge over a junction point (like a highway bridge passing an interstate) resulting in the creation of a crossover along with the junction at the same Euclidean point. It may also be the case that two or more junctions (represented by disconnected sets of channel labels) exist at the same point but are separated from each other by a vertical gap. In these two situations, a junction point and a crossover point appear together, and we call such points *dual interaction points*.

It is important to distinguish between junction points, crossover points, and dual interaction points in a spatial network because they have an impact on the computation of shortest paths and other network queries. As an example, we consider a spatial mapping of type A with $A = \{l_{A,1}, l_{A,2}, l_{A,3}, l_{A,4}\}$. We suppose that (i) the channels $l_{A,1}$ and $l_{A,2}$ form a junction represented as $\{\{l_{A,1}, l_{A,2}\}\}$, (ii) the channels $l_{A,3}$ and $l_{A,4}$ form another junction represented by $\{\{l_{A,3}, l_{A,4}\}\}$, and (iii) both junctions have the same location but form a crossover (Figure 3a). The two junctions thus represent a dual interaction point represented by $\{\{l_{A,1}, l_{A,2}\}, \{l_{A,3}, l_{A,4}\}\}$ as a set of sets of labels (Figure 3b).



(a) (b)

Figure 3. Two separate junctions (a) and the junctions co-existing ata singlepoint (b)

In summary, we see that a point of a channel that is neither a junction point nor a crossover point can be modeled as an element of a label set A; an example is $l_{A,3}$. The modeling of a junction requires a set of labels indicating the participating channels, that is, an element of 2^A, for example, $\{l_{A,1},l_{A,2}\}$. The modeling of a crossover point requires a set of sets of labels indicating the participating channels, that is, an element of 2^{2^A}, for example, $\{\{l_{A,1}\}, \{l_{A,2}\}\}$. The modeling of different junctions that form a crossover at the same location requires a set of sets of labels indicating the participating junctions, that is, an element of 2^{2^A}, for example, $\{\{l_{A,1},l_{A,3}\}, \{l_{A,2},l_{A,4}\}\}$.We see that points of the Euclidean plane can be either mapped to elements of A, 2^A, or 2^{2^A}. In order to obtain a unique mapping for all points of the Euclidean plane, we take the most general case and make a spatial mapping a function which is defined from R^2 to 2^{2^A}. Hence, we obtain the sets $\{\{l_{A,3}\}\}$, $\{\{l_{A,1},l_{A,2}\}\}$, $\{\{l_{A,1}\}, \{l_{A,2}\}\}$, and $\{\{l_{A,1},l_{A,3}\}, \{l_{A,2},l_{A,4}\}\}$ in the examples above. The set $\{\{\perp\}\}$ characterizes points having no label and makes the spatial mapping function a total function.

The definition of a spatial mapping is as follows: Let A be a label type. A spatial mapping of type A is a total mapping $\nu : \mathbb{R}^2 \rightarrow 2^{2^A}$. The set of all spatial mappings of type A is denoted by $[A]$, that is, $[A] = \mathbb{R}^2 \rightarrow 2^{2^A}$. When applied to a set X, the function is iteratively applied to all the elements of X, that is, $\nu(X) = \{\nu(p) | p \in X\}$. The concept of spatial mappings is too general for spatial networks. In other words, not every spatial mapping represents a spatial network. We have to impose certain restrictions on spatial mappings as described in the following. The idea is to infer and distinguish channels, junctions, and crossovers from the label information. Points which have a label other than $\{\{\perp\}\}$ belong to a channel. A crossover point is indicated by a set containing at least two sets of channel labels. To identify junction points, we look in to each set of labels. If any one of the sets has more than one channel label, it means that the channels which belong to those labels form a junction.

We now provide the definition for identifing channels, junction points, and crossover points in a spatial mapping. Let ν be a spatial mapping of type A. Then

(i) $L(\nu) = \{p \mid p \in \mathbb{R}^2 \wedge (p) \neq \{\{\perp\}\}\}$ (*channels*)

(ii) $J(\nu) = \{p \mid p \in \mathbb{R}^2 \wedge (\exists\, l \in (p): |l| > 1)\}$ (*junctions*)

(iii) $C(\nu) = \{p \mid p \in \mathbb{R}^2 \wedge |\nu(p)| > 1\}$ (*crossover*)

$L(\nu)$ contains all the points in the Euclidean plane which are part of a channel. Individual channels can be identified by the label information, as the channel identifier given by the *id_attr* part of the label uniquely identifies a channel. All the points of the same channel have the same channel identifier in their label. Thus, each channel may be distinguished by grouping the points in $L(\nu)$ by the *id_attr* part of the label. This necessitates a look into the labels and an extraction of parts from them. We can assume that an element of the label type A is a sequence of label attribute values and that each such value is of a

(possibly different) set A_i. That is, $A = \times_{i=1}^{k} A_i = A_1 \times ... \times A_k$. Let $I = \{1,...,k\}$, $S = \{j_1,...,j_n\}$, and $S \subseteq I$.To extract selected attribute values from a label, we define a projection operator Π as follows:

$$\Pi_S : \times_{i \in I} A_i \rightarrow \times_{j \in S} A_j$$

with $\Pi_S(a_1,...,a_k) = (a_{j_1},...,a_{j_n})$.

Next we define a function called *Id_Attr* to extract all the *id_attr* label parts (channel identifiers) actually present in a spatial mapping. It takes as argument a spatial mapping v of type A and computes the set of all *id_attr* values by using the projection operator . As the *id_attr* attribute is assumed to be always the first attribute a_1 in a label, we use $\Pi_{\{1\}}$ to extract its value. Consequently, A_1 is the label type of the *id_attr* attribute. Further, we generalize function applications from elements to sets of elements. Let $f : X \rightarrow Y$ be a function, and let $B \subseteq X$. Then we allow to use the notation $f(B)$ which is given as $f(B) = \{f(x) \mid x \in B\}$. This is here applied to a spatial mapping. The function *Id_Attr* is now defined for $v \in [A]$ as:

$$Id\_Attr(v) = \{\Pi_{\{1\}}(l) \mid s \in v(L(v)), e \in s, l \in e\}$$

Each channel has a unique *id_attr* value; thus, all points which belong to the same channel have the same *id_attr* value in their labels. The two functions Channel and Channels identify all channels in a spatial mapping. The function Channel gets an *id_attr* value as input and determines all points in the Euclidean plane that have this value in their labels and thus form a particular channel. The function Channels gets a spatial mapping as input and collects all channels by iterating over all of its *id_attr* values Points representing junction points or crossover points are part of interacting channels; hence, they appear in more than one channel.

Let v be a spatial mapping of type A, and let $l \in Id\_Attr(v)$. Then

(i) $Channel(v, l) = \{p \mid p \in L(v) \wedge \exists s \in v(p)\ \exists e \in s : \Pi_{\{1\}}(e) = l\}$

(ii) $Channels(v) = \bigcup_{l \in Id\_Attr(v)} Channel(v, l)$

For the definition of a spatial network, we have to consider its underlying line-shaped geometric structures. This requires the concept of a *simple line*. The set *sline* of all *simple lines* in the Euclidean plane is defined as:

$sline = \{L \subset \mathbb{R}^2 \mid$

(i) $\exists f : [0, 1] \rightarrow \mathbb{R}^2 : L = f([0, 1])$

(ii) f is a continuous mapping

(iii) $|f([0, 1])| > 1$

(iv) $\forall a, b \in [0, 1], a \neq b : f(a) \neq f(b)$

(v) $\forall a \in \{0, 1\} \forall b \in [0, 1]: f(a) \neq f(b)\}$

Conditions (i) and (ii) require the existence of a continuous function that generates the simple line. Condition (iii) avoids the anomaly that all elements of the unit interval are mapped to the same point. Condition (iv) states that a simple line is not allowed to be self-intersecting. Condition (v) requires that a simple line is not self-touching.

We are now able to define a *spatial network* of type A as a special spatial mapping of type A. *A spatial network* of type A is a spatial mapping v of type A such that

(i) $\forall L \in \text{Channels}(v): L \in sline$

(ii) $\forall p \in J(v): p \in L(v)$

(iii) $\forall p \in C(v): p \in L(v)$

(iv) $\forall p \in C(v)\ \forall s_1,s_2 \in v(p)\ \forall l_1 \in s_1\ \forall l_2 \in s_2 : \Pi_{\{1\}}(l_1)\ \forall = \Pi_{\{1\}}(l_2)$

Condition (iv) states that, in case of a crossover, the participating channels and/or junctions must be different since they cannot be physically present at more than one junction. This means that labels like $\{\{l_5\}, \{l_5\}\}$ or $\{\{l_1,l_5\}, \{l_3,l_5\}\}$ are invalid. The condition checks whether the *id_attr* values of the channels and /or junctions at each crossover point are disjoint.

We do not specify constraining topological relationships between different channels of a spatial network. This means that different channels may meet, partially overlap, or one channel may be contained in another channel. For example, in a road

network, many roads carry several names. This is, for instance, the case for U.S. Route 441 which is a spur route of U.S. Route 41.

A channel L with a describing function $f_L : [0, 1] \rightarrow \mathbb{R}^2$ has two *end points* $f_L(0)$ and $f_L(1)$. Dual interaction points, represented by $D(\nu)$, indicate the co-existence of junctions and crossovers. That is, $D(\nu) = J(\nu) \cap C(\nu)$. If $D(\nu) = \varnothing$ holds, the spatial network does not have dual interaction points. If additionally $C(\nu)\ \forall = \varnothing$ holds, this means that crossover points are passed by single channels.

The boundary points of a spatial network ν are those end points of the channels that are not shared by other channels. Let $Channels(\nu) = \{L_1,...,L_n\}$ that are described by functions $f_{L_1},...,f_{L_n}$. Let $E(\nu) = \bigcup_{i=1}^{n} \{f_{L_i}(0), f_{L_i}(1)\}$ be the set of end points of all channels of. The set $S(\nu) \subset E(\nu)$ of those points that are shared by more than one channel is given as

$$S(\nu) = \{ p \in E(\nu) \mid$$
$$card(\{ f_{l_i} \mid 1 \leq i \leq n \wedge f_{l_i}(0) = p\}) +$$
$$card(\{ f_{l_i} \mid 1 \leq i \leq n \wedge f_{l_i}(1) = p\}) \neq 1\}$$

Then the boundary points $B(\nu)$ are given as $B(\nu) = E(\nu) - S(\nu)$.

4. Operations On Spatial Networks

A number of interesting operations on spatial networks can be designed which assist the user in posing queries on spatial networks. We give the formal definitions of some of these operations and present their semantics. The definitions can later serve as a specification for the design of algorithms for the operations. The operations are divided into basic operations (Section 4.1), auxiliary operations (Section 4.2), retrieval operations (Section 4.3), and metric operations (Section 4.4).

4.1 Basic Operations on Spatial Networks

We introduce several basic operations on spatial networks that we deploy for the definition of the main operations on spatial networks in the subsequent subsections. These operations include a length operation (Section 4.1.1), a route calculation operation (Section 4.1.2), a geometry extraction operation (Section 4.1.3), and channel interaction operations (Section 4.1.4).

4.1.1 ***Length of a Channel***. The function *Length* is an important (overloaded) operator which calculates the length of a channel. It yields a real value as a result. The *Length* operator has the signature $[A] \rightarrow \mathbb{R}$. In its simplest form, this operator takes a channel as an input parameter. A function describing a channel must be integrable and bounded. This is always the case as the definition of channels requires that its describing function is continuous and bounded to the interval [0,1]. To calculate the length, we divide the entire channel into infinitesimally small chord approximations and integrate them. Let us consider a channel $L \in Channels(\nu)$ with a describing function f_L and the point set $f_L([0, 1])$. The Length operator is defined as

$$Length(L) = \int_{f_L(0)}^{f_L(1)} \sqrt{1 + (\partial f_L(x)/\partial x)^2}\partial x$$

This method may also be used to calculate the length between any two points in the same channel. Here we integrate from the first point in the channel to the second point in the channel. Consider again a channel L as described above, and the two points $p = f_L(a)$ and $q = f_L(b)$ with $a, b \in [0, 1]$. We define a modified Length operator with the signature Length: $[A] \times [0, 1] \times [0, 1] \rightarrow \mathbb{R}$ as

$$Length(L, a, b) = \int_{f_L(a)}^{f_L(b)} \sqrt{1 + (\partial f_L(x)/\partial x)^2}\partial x$$

Another variation of the *Length* operator is an extension of the first version with the same signature *Length*: $[A] \rightarrow \mathbb{R}$. But in this case, this operator takes a complete spatial network $\nu \in [A]$ as argument and sums up the lengths of all the channels in the spatial network. It is defined as

$$Length(\nu) = \sum_{L \in Channels(\nu)} Length(L)$$

4.1.2 ***Routes between Two Network Points***. A *route* is a course (way, path, connection) one can take in order to reach a second location from a first location. Given a spatial network, a route connects two points of the network through an alternating

sequence of channels and junctions of the same network. A route between the two cities Atlanta and Gainesville is an example. Finding routes is an important feature of spatial network applications as the locations of moving objects in a network are stored with respect to a particular route. There can be possibly a large number of routes between two points in a network. We consider routes to be spatial networks with certain constraints. As a route connects two points p and q, the route starts at p and ends at q; that is, p and q are the boundary points of the spatial network which represents a route. To prevent any discontinuity in the route, it can only have exactly two boundary points.

In our model, we define an operator called *Routes* which takes two points inside a spatial network and creates the set of all possible connections between them. Note that this is a conceptual consideration and not an algorithmic implementation strategy. The signature of the *Routes* operator is *Routes* : $[A] \times \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow 2^{[A]}$. This operator returns the set of all spatial sub networks representing routes between two selected points over a given spatial network. All points of each returned spatial network, that is, route, are a subset of the points of the original network. Figure 4 shows the resulting routes when the operation *Routes* (v, j_2 , j_6) is performed on the network depicted in Figure 2 to calculate the paths from j_2 to j_6. Given a spatial network $v \in [A]$ and two points $p, q \in \mathrm{L}(v)$, we define the *Routes* operator as

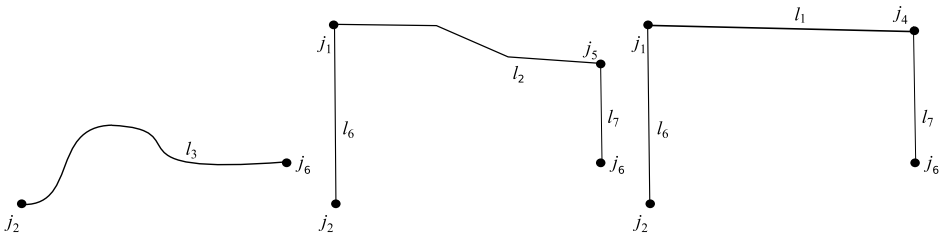


Figure 4. The routes returned by executing *Routes* (v, j_2 , j_6) on the spatial network shown in Figure 1b

Routes (v, p, q) = { v' | (i) $v' \in [A]$ is a spatial network

(ii) $L(v') \subseteq L(v)$

(iii) $\forall\, l \in v'(J(v')) : |l| = 2$

(iv) $p, q \in B(v')$

(v) $|B(v')| = 2$}

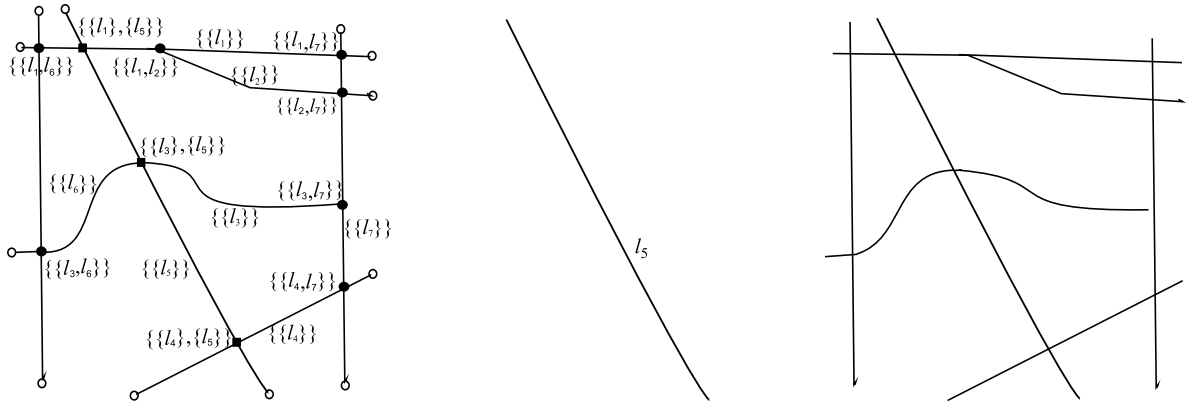
Condition (i) states that every route is a spatial network. Condition (ii) ensures that every route v' is a subnetwork of v. Condition (iii) requires that each junction of a route must have a degree of exactly two. Condition (iv) states that p and q are boundary points of the route v'. Condition (v) ensures that p and q are the *only* boundary points of v' .

4.1.3 ***Geometry of Spatial Networks***. The *getGeometry* operator extracts selected parts of the geometry of a spatial network by suppressing its thematic information and its connectivity information. It returns these geometric parts in the form of a spatial line object represented by the spatial data type *line* [Schneider 1997 [20]]. The *getGeometry* operator is overloaded and able to take a single channel or an entire network as an argument. The geometry of a channel is a simple (that is, continuous and non self-intersecting) line while the geometry of a network is an example of a complex line with possibly multiple components. The signature of the first version of the *getGeometry* operator is *getGeometry* : $[A] \times A_1 \rightarrow$ *line* in which A_1 is the first component type of the label type A and represents the type of the *id_attr* attribute. The signature of the second version is *getGeometry* : $[A] \rightarrow$ *line*. Figure 5b shows the result of running the operation *getGeometry* (v, l_5) on the road network $v \in [A]$ shown in Figure 5a. The result is a single line depicting the geometry of the highway named l_5. All junctions and crossover points are removed from it, leaving only its geometry. Similarly, if the operation *getGeometry* (v) is applied to the same network, the result is a complex line without any label information and any junction or crossover information as shown in Figure 5c. After obtaining the geometry of a spatial network or part of a spatial network as a line, it can be used to perform intersection operations as shown in Section 4.3.

Assuming a spatial network v and a channel identifier $l \in A_1$, we have the following definition for the two versions of *getGeometry*.

getGeometry(v, l) = *Channel*(l)

getGeometry (v) = $\{p \in R^2 \mid v(p) \neq \perp\}$



(a) (b) (c)

Figure 5. The result of executing *getGeometry*(v, l_5) (b) and *getGeometry*(v) (c) on the spatial network v shown in (a)

The first version uses the *Channel* operator described in Section 3.2 to return all the points belonging to the channel (identifier) l. The second definition collects all the points in the plane which do not have an exterior ($\perp$) label. These points form the geometry of the network as all points of the network have a non-exterior label.

4.1.4 *Points of Interaction*. Sometimes, given two channels, it is interesting to know whether they interact with each other, and if they do interact, we might want to classify the interaction as a crossover or a junction. The operations JunctionPoints and CrossoverPoints take two channel identifiers as arguments and return the set of points where they form junctions and crossovers respectively. Both operators return an empty point set if there are no such junction points or crossover points respectively. For example, the result of executing the operations JunctionPoints(v, l_7, l_2) on the network shown in Figure 1b returns the set $\{j_5\}$ which indicates that the point j_5 is the point where the channels with the labels (component id_attr) l_7 and l_2 interact to form a junction. The signatures of both operations are the same, namely JunctionPoints, CrossoverPoints: $[A] \times A_1 \times A_1 \rightarrow point$.

We now give the formal definition of both operations and assume a spatial network $v : [A]$ and two channel identifiers $l_1, l_2 \in A_1$ with $l_1 \neq l_2$.

$$JunctionPoints(v, l_1, l_2) = \{p \mid (\text{i}) \; p \in J(v)$$
$$(\text{ii}) \; \exists \, s \in v(p) \; \exists \, e_1, e_2 \in s : \Pi_{\{1\}}(e_1) = l_1 \wedge \Pi_{\{1\}}(e_2) = l_2\}$$

This operation returns a set of points (Condition (i)) whose label includes a single set containing both channel identifiers l_1 and l_2 (Condition (ii)). This indicates that the channels with the identifiers l_1 and l_2 forma junction at that particular point.

Similarly, the operation *CrossoverPoints* returns a set of points (Condition (i)) such that the label of that point contains multiple sets indicating a crossover point. Condition (ii) ensures that the channel identifiers l_1 and l_2 are elements of two different sets.

$$CrossoverPoints(v, l_1, l_2) = \{p \mid (\text{i}) \; p \in C(v)$$
$$(\text{ii}) \; \exists \, s_1, s_2 \in v(p), s_1 \neq s_2 \; \exists \, e_1 \in s_1, e_2 \in s_2 : \Pi_{\{1\}}(e_1) = l_1 \wedge \Pi_{\{1\}}(e_2) = l_2\}$$

These two operations are overloaded and also take only a single channel identifier $l \in A_1$ as an argument. In this case, the operations *JunctionPoints* and *CrossoverPoints* return all junction points and all crossover points of the channel identified by l. For example, in Figure 1b, the expression *JunctionPoints*(v, l_7) returns all the points on l_7 which represent the locations of junctions with other channels. The resulting set is $\{j_4, j_5, j_6, j_7\}$. Executing *CrossoverPoints*(v, l_5) yields the set $\{c_1, c_2, c_3\}$. The signature of both operators is *JunctionPoints, CrossoverPoints*: $[A] \times A_1 \rightarrow point$. The definition of this version of the operation *JunctionPoints* is as follows:

$$JunctionPoints(v, l) = \{p \mid (\text{i}) \; p \in J(v)$$
$$(\text{ii}) \; \exists \, s \in (p) \; \exists \, e \in s : \Pi_{\{1\}}(e) = l\}$$

Similarly, we define the corresponding version of the operation *CrossoverPoints*:

$CrossoverPoints(v, l) = \{p \mid$ (i) $p \in C(v)$

(ii) $\exists\, s \in (p)\ \exists\, e \in s : \Pi_{\{1\}}(e) = l\}$

4.2 Auxiliary Operations on Spatial Networks

In this section, we specify the three auxiliary operations *PartOfChannels*, *isDirectlyConnected*, and *DCN* on spatial networks. Each operation deploys the previous operation in this list for its definition.

The operation *PartOfChannels* takes any point on the network and returns the set of all channel identifiers to which the point belongs. For this purpose, it retrieves all channel labels associated with the point and extracts the channel identifiers from these labels. For a spatial network $v \in [A]$ and a network point $p \in L(v)$, this operation is defined as

$PartOfChannels(v, p) = \{\Pi_{\{1\}}(l) \mid e \in v\,(p), l \in e\}$

The Boolean predicate *isDirectlyConnected* returns true if two given points on a network are connected viaa single channel. That is, in order to go from one point to another point, there is no need to change channels. The predicate has the signature *isDirectlyConnected*: $[A] \times \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{B}$ and uses the operator *PartOfChannels* to find out if both points in question are parts of a common channel. For a spatial network $v \in [A]$ and two points $p, q \in L(v)$, this predicate is defined as

$isDirectlyConnected(v, p, q) = (PartOfChannels(v, p) \cap PathOfChannels(v, q) \neq \emptyset)$

The operation *DCN* applied to a spatial network and a reference junction point determines the set of junction points which are reachable from the reference junction point via a single channel, or in other words, that belong to the same channel. It checks all junction points of the spatial network and collects those for which the predicate *isDirectlyConnected* yields *true*. For a spatial network $v \in [A]$ and a junction point $p \in J(v)$, *DCN* is defined as

$DCN\,(v, p) = \{q \mid q \in J(v) \wedge isDirectlyConnected(v, p, q)\}$

4.3 Retrieval Operations on Spatial Networks

In this section we look at some high-level retrieval operations that return parts of spatial network. We deal with shortest route calculation (Section 4.3.1), network intersection (Section 4.3.2), and channel connection detection (Section 4.3.3).

4.3.1 *Shortest Route Calculation*. One of the classical queries in a spatial network is the shortest route (path) query. The task is to find a route between two points in a network which has the least distance among all routes between the two points. Shortest route queries are used to automatically find driving directions between physical locations, for example, between two cities. The *ShortestRoute* operator finds such a shortest route between two points $p, q \in L(v)$ in a network $v \in [A]$. The signature of this operator is *ShortestRoute*: $[A] \times \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow [A]$. Its definition is given as

$ShortestRoute(v, p, q) = \{sr \mid$ (i) $sr \in Routes(v, p, q)$

(ii) $\forall\, r \in Routes(v, p, q)$: $Length(r) \geq Length(sr)\}$

This operator checks all the routes between p and q and compares their length. It chooses the route with the smallest length as the shortest route. Since there could be several shortest paths, it returns all of them in a set. Note that this is a conceptual definition and not an algorithmic strategy to compute the shortest path.

4.3.2 *Network Intersection*. Consider the queries "Which parts of the national highway have been affected by the snowstorm X?". For this kind of query, we first need to obtain the extent of the snowstorm as a spatial region. If this region is geometrically intersected with the highway network, we obtain those parts of the network that have been affected by the snowstorm. Such a kind of query aiming at the intersection of a spatial network with a region can be useful in various situations. We provide the two operations *Window* and *Clipping* for this purpose.

The operation *Window* allows a user to retrieve those *complete* channels of a spatial network whose intersection with a given (region) window is not empty. Figure 6a shows a spatial network with a region r (colored in grey) overlaying it. Figure 6b demonstrates the effect of the operation $Window(v, r)$ on the spatial network v with respect to r. This operation completely preserves the channels l_3, l_5, and l_7 that intersect r. Their boundary points are also preserved but the junction points (like j_5 and j_7) and crossover points (like c_1 and c_3) with channels that are not part of the result are removed. Only for illustration purposes the query window in Figure 6 is a rectangle. But it can be any object of the spatial data type *region* [Schneider 1997 [20]], that is, in particular, it can be of any areal shape, have holes, and consist of multiple components.

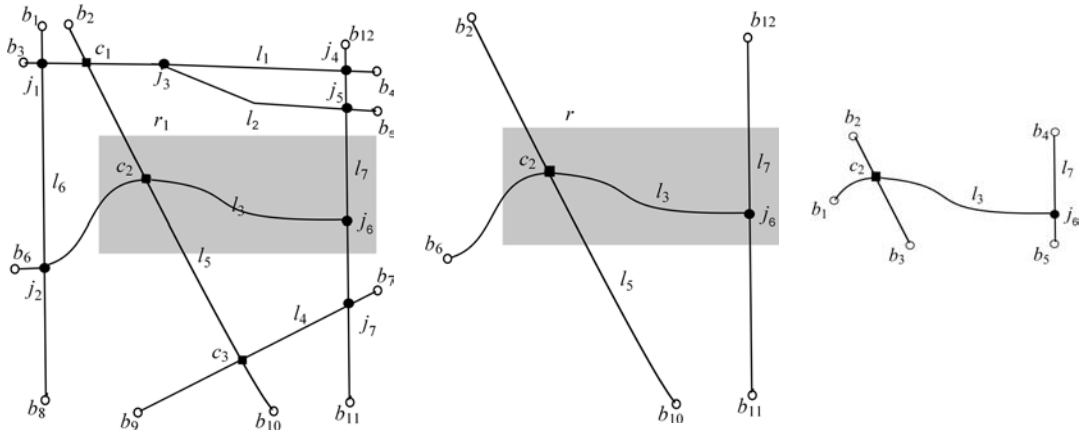


Figure 6. The original network and the region r_1. The result of the clip operation with the network and the region r_1 (a), and the result of the window operation with the network and the region r_1 (b)

The signature of the *Window* operation is *Window*: [*A*] × *region* → [*A*]. The definition of this operation makes use of the well known geometric set operation intersection (⊗) between spatial lines and regions. For a spatial network $v \in [A]$ and an arbitrary region object $r \in region$, the *Window* operation is defined as follows:

Window(v, r) = v'

such that the following conditions hold:

(i) $v' \in [A]$ is a spatial network

(ii) $L(v') \subseteq L(v)$

(iii) $\forall\ l \in Id\_Attr(v')$: $getGeometry(v, l) \otimes r \neq \emptyset$

(iv) $\forall\ l \in Id\_Attr(v) - Id\_Attr(v')$: $getGeometry(v, l) \otimes r = \emptyset$

Condition (i) states that the operation results in a spatial network of the same type. Condition (ii) requires that all points in the new network are elements of the original network. Condition (iii) ensures that all channels in the computed network geometrically intersect the given region r completely or partially. Condition (iv) guarantees that v' is the maximum spatial network that intersects r.

The operation *Clipping* does not only identify the channels that intersect a given rectangle but it also computes the intersection of the region with the network geometry, that is, with all intersecting channels. This operation may result in partial channels of the original network and thus gives rise to artifacts. For example, new boundary points are created wherever a part of the channel is clipped by the edge of the query region. But the interior parts of the network always remain intact. Figure 6c gives an example of the operation *Clipping*(v, r) and shows those parts of the network v in Figure 6a that geometrically intersect region r. The effect of this operation is that channels like l_1, l_2, l_4, and l_6 are removed, the original channels l_5 and l_7 are truncated, and new boundary points are created, for example, at the endpoints of the channels l_5 and l_7.

The signature of the *Clipping* operation is *Clipping*: [*A*] × *region* → [*A*]. For a spatial network $v \in [A]$ and a region object $r \in region$, the *Clipping* operation is defined as follows:

Clipping(v, r)= v'

such that the following conditions hold:

(i) $v' \in [A]$ is a spatial network

(ii) $L(v')$= $getGeometry(v) \otimes r$

(iii) $\forall\ p \in L(v')$: $v'(p) = (p)$

Condition (i) states that the operation results in a spatial network of the same type. Condition (ii) requires that all points of the resulting network are exactly those points of the original network that geometrically intersect region r. This means, in particular, that $L(v') \subseteq L(v)$ holds. Condition (iii) ensures that the labeling of the original network is preserved in the clipped new network.

4.3.3 *Channel Connection Detection*. A channel is connected to another channel only if the two channels share a junction. A particular channel may have a number of other channels connected to it by forming multiple common junction points. The operation *Connected_to* computes all channels that are connected to a particular channel. For example, a major river may have a number of tributaries. The operation *Connected_to* returns all tributaries which are connected to the river. In our example network shown in Figure 1b, the result of *Connected_to*(ν, l_3) is the sub network of that contains the channels with the identifiers l_6 and l_7. The channel l_6 is connected to channel l_3 at the junction point j_2. Similarly, the channel l_7 is connected to channel l_3 at the junction point j_6. However, the channel l_5 does not appear in the result even though it interacts with channel l_3. The reason is that the point of interaction is a crossover point and not a junction point. The operation has the signature *Connected_to*: $[A] \times A_1 \rightarrow [A]$, that is, it takes a spatial network $\nu \in [A]$ and a channel identifier $l \in A_1$ as arguments and returns the subnetwork of ν that contains all channels that are connected to l and thus share a junction point with l.

Connected_to(ν, l)= ν'

such that the following conditions hold:

(i) $\nu' \in [A]$ is a spatial network

(ii) $L(\nu') = \bigcup_{\substack{p \in JunctionPoints(\nu, l) \\ s \in \nu(p),\, e \in s - \{l\}, l' \in \Pi_{\{1\}}(e)}} getGeometry(\nu, l')$

(iii) $\forall\, p \in L(\nu') : \nu'(p) = \nu(p) - \{l\} - \{l' \in A \mid q \in JunctionPoints(\nu, l), S = \bigcup_{s \in \nu(q)} s, l' \notin S\}$

Condition (ii) requires that the resulting network only contains the points of channels that share a junction point with l. For this purpose, the operation first finds the junction points on l using the operation *JunctionPoints* (Section 4.1.4). Then it determines the channel identifiers associated with each junction point and retrieves the geometries for each çhannel identifier with the operation *getGeometry* (Section 4.1.3). Condition (iii) states that the labels of the new network ν' have to be inherited from the network ν. However, the labels of ν' have to be corrected in the sense that for each network point any of its labels that is not associated with a junction point on l has to be removed. In particular, label l has to be removed.

4.4 Metric Operations on Spatial Networks

An important class of operations on spatial networks are metric operations which compute metric properties and return a numerical result. Metrics like the degree of a junction or the network distance are examples of metric operations. An important class of metric operations called centrality measures like degree centrality and closeness centrality is very useful in network analysis. In subsequent discussions, the term *nodes* is used to denote points of interconnection in a spatial network which allow transfer opportunities. In our model, junction points are points which allows one to change or transfer from one channel to another; hence we express the junction points as nodes in our analysis. This section provides operations for various metrics, namely the *network distance* (Section 4.4.1), the *degree of a junction* (Section 4.4.2), the *characteristic path length* (Section 4.4.3), the *global effciency* (Section 4.4.4), the *degree centrality* (Section 4.4.5), and the *closeness centrality* (Section 4.4.6).

4.4.1 *Network Distance*. Network distance is the distance between any two points in the network via the network, that is, the minimum distance one has to travel in the network to reach the second point from the first. Hence, it is an extension of the function *Length* (Section 4.1.1) which calculates the distance between two points in the same channel. Queries like "What is the traveling distance from Gainesville to Atlanta?" may be answered based on the network distance. The definition of network distance makes use of the operation *ShortestRoute* (Section 4.3.1).Two points in a network may have multiple routes connecting them, but the most interesting route is the one which has the smallest total length and is thus called the *shortest route*. We now define the operation *NetworkDistance* whose signature is *NetworkDistance*: $[A] \times \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$. Given a spatial network $\nu \in [A]$ and two points $p, q \in L(\nu)$, we define this operation as

$$NetworkDistance(\nu, p, q) = Length(ShortestRoute(\nu, p, q))$$

4.4.2 *Degree of a Junction*. The degree of a junction or a node computes the total number of channels which intersect at the junction point. This measure indicate show well a particular junction is connected with the rest of the network and is also known as the connectivity of the node. The higher the degree is, the more important is the particular junction point in the network. A scale free graph like a road network follows the power law which states that new connections preferentially connect to nodes having higher degree.

In our model, any junction point or crossover point is formed by the interaction of a number of channels. A junction point belongs to all channels which intersect it. A point from any channel which is not a part of a junction or a crossover will only

belong to a single channel. Junction points and crossover points are labeled by the combination of all labels of channels which interact at the point. Hence, we only have to determine the cardinality of these labels. For this purpose, we define the operation *DegNode* whose signature is *DegNode*: $[A] \times \mathbb{R}^2 \rightarrow \mathbb{R}$. For a spatial network $\nu \in [A]$ and a junction $p \in L(\nu)$, the operation *DegNode* makes use of the operation *PartOfChannels* (Section 4.2) and computes the degree of p as

$$DegNode(\nu, p) = |PartOfChannels(\nu, p)|$$

4.4.3 *Characteristic Path Length*. The characteristic path length is understood as the impediment between all pairs of nodes with in a network. It may be expressed as the average distance (impediment) between all pairs of nodes in a network, and this measure is suited for a comparative analysis of public transport networks or for network assessment. A small characteristic path length indicates the presence of short-cut connections between the nodes in the network. This measure has vast significance in any type of spatial networks. For example, the work in [Chen et al. 2006 [2]] studies the impact of the characteristic path length on the structural vulnerability of power grids and concludes that with the decrease of the characteristic path length, the probability of mass load loss decreases dramatically.

The set of nodes considered here is the set of junctions in our model, and for a spatial network $\nu \in [A]$ it is given by $J(\nu)$. The operator *CPL*: $[A] \rightarrow \mathbb{R}$ computes the characteristic path length of a network by summing up the network distances between all pairs of distinct nodes in the network and by dividing the sum by the number of these pairs. It is defined as

$$CPL(\nu) = \frac{\sum_{p,q \in J(\nu), p<q} NetworkDistance(\nu, p, q)}{\frac{1}{2}|J(\nu)|(|J(\nu)| - 1)}$$

4.4.4 *Global Efficiency*. The ability of a spatial network such as a transportation network or a pipeline network to minimize spatial resistance or impediment to travel is indicated by its global efficiency. Global efficiency is useful for comparing centrality in networks before and after an alteration to its structure. It is calculated as the inverse average shortest route length between any two nodes (junctions) in a network. The operator *GlobalEffciency*: $[A] \rightarrow \mathbb{R}$ computes the global efficiency of a network ν of type $[A]$ and is defined as

$$GlobalEfficiency(\nu) = \frac{\sum_{p,q \in J(\nu), p<q} \frac{1}{NetworkDistance(\nu, p, q)}}{\frac{1}{2}|J(\nu)|(|J(\nu)| - 1)}$$

4.4.5 *Degree Centrality*. Degree centrality falls under a category of measures called centrality measures. These measures are indicators of individual nodes containing locally relevant information. Degree centrality of a node is defined as the ratio of nodes directly connected to it out of all the nodes in the network. It measures the average number of nodes encountered in order to access every other node in the network. In a public transportation network, it may be understood as the number of transfers required to reach a particular node. The definition of this operation takes the help of the predicate *is Directly Connected* and the operation *DCN* (both described in Section 4.2).

The degree centrality is computed by the operation *DegreeCentrality*: $[A] \times \mathbb{R}^2 \rightarrow \mathbb{R}$ which uses the operation *DCN* to find the number of directly connected nodes to a node $p \in J(\nu)$ in a spatial network $\nu \in [A]$ and divides this number by the total number of nodes in the network. It is defined as

$$DegreeCentrality(\nu, p) = \frac{|DCN(\nu, p)|}{|J(\nu)|}$$

4.4.6 *Closeness Centrality*. This centrality measure indicates the ease of movement between a node and the rest of the network, that is, how closely a particular node is situated to other nodes. Closeness centrality is measured by the impediment to travel, which in turn is measured by the network distance. It is defined as the inverse average distance between the node in question and all the other nodes in the network. This metric is computed by the operation *ClosenessCentrality*: $[A] \times \mathbb{R}^2 \rightarrow \mathbb{R}$ which sums up the shortest distance between a node $p \in J(\nu)$ to all the other junction nodes in the network $\nu \in$ [A] and averages it. The operation is defined as

$$ClosenessCentrality(\nu, p) = \frac{|J(\nu)|}{\sum_{q \in J(\nu), q \neq p} NetworkDistance(\nu, p, q)}$$

5. Spatial Network Query Language

In this section, we introduce an SQL-like query language for spatial networks. We call the language *Spatial Networks Query Language* (SNQL). SNQL is supposed to allow easy access to spatial networks and to provide a comfortable way to apply operations to them. A fundamental and challenging question is how spatial networks can be best modeled (and stored) in spatial databases. Our conceptual view is *not* that spatial network data are spread over a large number of database tables. This has the fundamental drawbacks that spatial networks are not recognizable as self-contained entities and that network operations cannot be directly applied to them. Instead, we advocate the concept of *abstract data types* in databases and apply it to spatial networks. This means that we consider a spatial network as a whole and as an object that is stored as an attribute value of a tuple. Information about a spatial network can only be obtained by high-level operations like those specified in this article.

Consider a national highway system represented as a collection of spatial networks. We assume a table *NationalHighway* (*sname* : *string*, *hwynet*: [*highwaynames*]) that stores the national highway (given by the attribute *hwynet*) for each state (given by its name *sname*). The term *highwaynames* denotes the label type for all highways, and the term [*highwaynames*] denotes the type of all spatial networks of type *highwaynames*. In other words, the brackets [...] serve as a type constructor that takes a label type and constructs a spatial network type from it. Our assumption is that label types are special string types. Hence, labels are enclosed by single quotes. The query is now that a traveler wants to drive from Gainesville to Miami. The main aim of the traveler is to reach Miami in the least amount of time. This means that he would like to travel on the route with the shortest length. The query may be formulated as

```
select ShortestRoute(hwynet, 'Gainesville', 'Miami') as sr
from NationalHighway
where sname = 'Florida'
```

Sometimes the shortest route may not necessarily be the least time taking route. There might be congestions and other causes of delay along this route. Hence, a traveler might be more interested in having a set of possible routes from Gainesville to Miami. He may then choose his preferred route based on various other considerations like speed limits and congestions. This may be formulated in a query. But the number of possible paths from Gainesville to Miami might be large. Thus the query should have a limit on the number of routes it will return. In this particular case, we restrict the network distance of the paths to not more than 500 miles. The query is as follows:

```
select Routes(hwynet, 'Gainesville', 'Miami') as sr
from NationalHighway
where sname = 'Florida' and Length(sr) < 500
```

In the database, we will find a tuple for each route found. Each route represents a spatial network.

In the next query, the select clause is used to project out a particular label attribute *speedlimit* from the network. For example, a query to find the average speed of the route from Gainesville to Tallahassee could be formulated as follows:

```
select average(hwynet, speedlimit) as avg_speed_limit
from (select ShortestRoute(hwynet, 'Gainesville', 'Tallahassee')
      from NationalHighway
      where sname = 'Florida')
```

This query first computes the shortest route between Gainesville and Tallahassee from the National Highway network in the inner query. This route is represented as a spatial network. Then a spatial network aggregation function named average is applied to this network that we have not defined in this article. It calculates the average of all values of the label attribute *speedlimit* with respect to the determined spatial network (here: route). The label attribute is assumed to belong to the theme_attr part (see Section 3.1) of the label type highwaynames of the network NationalHighway.

Another query could ask for the length of all highway roads in Orlando. We assume a table *Cities* (*cname* : *string*, *cloc* : *point*, *carea* : *region*) in which city locations are represented as point objects and city areas are represented as region objects.

```
select Length(Clipping(hwynet, carea)) as TotalLength
from NationalHighway, Cities
where sname = 'Florida' and cname = 'Orlando'
```

As pointed out earlier, spatial network analysis is important in urban planning. For example, assume that the City of Gainesville authorities notice that a part of the city road grid gets jammed during peak hours. They decide to widen any street to lessen the congestion. It would be most effective to widen that road which has the most potential of getting clogged. A fair

measure of this would be to count the number of roads leading up to the congested road. We formulate a query to determine this number of connected roads for the 13th Street in Gainesville, Florida.

select NumberOf(Id_Attr(Connected_to(GnvRoads, '13th Street'))) as no_of_connected_roads
from (select Clipping(hwynet, carea) as GnvRoads
from NationalHighway, Cities
where sname = 'Florida' **and** cname = 'Gainesville')

In a first step, the inner query calculates the highway network *GnvRoads* of the Gainesville area. In a second step, the query deploys the operation *Connected_to* to determine the subnetwork of channels that are connected to the 13th Street. In a third step, the operation *Id_Attr* determines all labels (channel identifiers) emerging in the subnetwork. Finally, the function NumberOf yields their number.

The query "Which parts of the national highway have been affected by snowstorms?" may be formulated by employing the operation *Window*. We assume a table Snowstorm that has an attribute *snowstorm_name* of type *string* and an attribute *snowstorm_area* which is of type *region* and represents the extent of a snow storm. The operation *Window* returns entire roads whichhave tobe closed because of a snowstorm.

select snowstorm_name, Window(hwynet, snowstorm_area) as affected_area
from NationalHighway, Snowstorms

Any public transport system has to evaluate the utility of proposed routes before putting them into service. Given two stops at the points j_1 and j_2 in Tampa, which a newly proposed route may service, decision makers will ask "Which of the two stops will be most accessible by the public?" since the higher the accessibility of a stop is, the higher is its importance. This may be answered by the *ClosenessCentrality* operator, and the query maybe formulated as

select ClosenessCentrality(TampaRoads, j1) > ClosenessCentrality(TampaRoads, j_2)
as j1_more_accessible
from (select Clipping(hwynet, carea) as TampaRoads
from NationalHighway, Cities
where sname = 'Florida' and cname = 'Tampa')

The operation *DegreeCentrality* may help the user answer queries of the form "How are cities connected by the national railway network?". The higher the degree centrality is, the lesser number of transfers are required to reach that particular node. We assume a table *NationalRailway* which keeps the railway networks all over the country. The name of a network is stored in an attribute *rwyname* and the network itself is stored in an attribute *rwynet*. We further assume that city locations are junction points in the railway networks. Then the query can be posed as

select cname, rwyname, DegreeCentrality(rwynet, cloc) as dc
from NationalRailway, cities
where getGeometry(rwynet) intersects carea

The *from* and *where* clause demonstrate a *partial spatial network* join in which a spatial network and a city area are combined and considered if both intersect each other. For the intersection test, we employ the topological predicate intersects on complex regions [Schneider and Behr 2006 [21]]. Only then the degree centrality is computed for a particular city and a particular railway network.

In a power network, the voltage in the wires drops with distance. In order to compare two power grids based on their usability, it is essential to look into their mean path length. The mean path length is given by the characteristic path length. Suppose we have a number of proposed power grids, we might ask "What is the lowest mean path length of a proposed power grid?". To answer this question, we assume that a table *ProposedPowerGrids* exists which contains a number of proposed power networks stored under the attribute *powergrid*. The query can be then formulated as

select min(CPL(powergrid)) as LowestMeanPathLength
from ProposedPowerGrids

6. Conclusions And Future work

This paper introduces a formal data model for generic spatial networks. We define a set of operators that (only) take one spatial network into account. The operations are classified into basic, retrieval, and metric operations and can be applied to a wide range of applications. This data model is expected to serve as a specification for a later implementation and integration in spatial (network) databases, Geographic Information Systems, transportation systems, and navigation systems. Assuming

a fictitious integration of the data model into a spatial database, the article demonstrates how an SQL-like query language named the Spatial Networks Query Language (SNQL) maybe used.

This work is a part of a larger effort called the *Spatial Networks Algebra* (SNA) which is supposed to become a generic model for a large range of spatial networks. It will have an even more comprehensive collection of operations and predicates defined on them. Especially binary operations on two spatial networks will be a focus of future work. Further, spatial networks are to be incorporated with spatial partitions and create a complete *Map Algebra*. The formal specification described in this paper takes an abstract approach. This work will be extended to create a discrete representation of the data model with the intention to implement it in database systems.

References

[1] Brinkhoff, T. A (2002). Framework for Generating Network-Based Moving Objects. *Geoinformatica* 6 (2) 153–180, June.

[2] Chen, X., Jiang, Q., Cao, Y. (2006). Impact of Characteristic Path Length on Cascading Failure of Power Grid. *In*: Int. Conf. on Power System Technology. p. 1–5.

[3] Cruz, I. F., Mendelzon, A. O., Wood, P. T. (1987). A Graphical Query Language Supporting Recursion, *In*: ACM SIGMOD Int. Conf. on Management of Data. p. 323–330.

[4] Erwig, M., Güting, R. H. Explicit Graphs in a Functional Model for Spatial Databases, *IEEE Transactions on Knowledge and Data Engineering* 6 (5) 787–804.

[5] Fleming, T., Mellor, B. (2006). An Introduction to Virtual Spatial Graph Theory. *In:* Int. Workshop on Knot Theory for Scientific Objects.

[6] Frentzos, E. (2003). Indexing Objects Moving on Fixed Networks, *In:* 8th Int. Symp. on Spatial and Temporal Databases. Springer, p. 289–305.

[7] Goodchild, M. F.(1998). Geographic Information Systems and DisaggregateTransportation Modeling. *Geographical Systems* 5 (1-2) 19–44, 1998.

[8] Gupta, S., Kopparty, S., Ravishankar, C. (2004). Roads, Codes, and Spatiotemporal Queries. *In*: 23rdACM SIGMOD SIGACT-SIGART Symp. on Principles of Database Systems. p. 115–124.

[9] Güting, H., de Almeida, T., Ding, Z. (2006). Modeling and Querying Moving Objects in Networks. The VLDB Journal 15 (2) 165–190.

[10] Güting, R. (1994). GraphDB: Modeling and Querying Graphs in Databases. In 20th Int. Conf. on Very Large Databases. p. 297–308.

[11] Gyssens, M., Paredaens, J., Van den Bussche, J., Van Gucht, D. A (1994). Graph-Oriented Object Database Model, *IEEE Transactions on Knowledge and Data Engineering* 6 (4) 417–424.

[12] Jensen, C. S., Pedersen, T. B., Speicys, L., Timko, I. (2003). Data Modeling for Mobile Services in the Real World. *In:* Int. Conf. on Advances in Spatial and Temporal Databases. p. 1–9.

[13] Jeung, H., Yiu, M. L., Zhou, X., Jensen, C. S. (2010). Path Prediction and Predictive Range Querying in Road Network Databases. *The VLDB Journal* 19 (4) 585–602.

[14] Longley, P. A., G. M. F. M. D. J., Rhind, D. W. (2001). Geographic Information Systems and Science.

[15] Mannino, M. V.,Shapiro, L. D.(1990). Extensions to Query Languages for Graph Traversal Problems, *IEEE Transactions on Knowledge and Data Engineering* 2 (3) 353–363.

[16] Meschini, L., Gentile, G., Papola, N. A (2007). Frequency Based Trans it Model for Dynamic Traffic Assignment to Multimodal Networks, *In:* 17th Int. Symp. on Transportation and Traffic Theory.

[17] Miller, H. J., Shaw, S.-L (2001). Geographic Information Systems for Transportation. Oxford University Press.

[18] Scheider, S. and May, M. A Method for Inductive Estimation of Public Transport Traffic using Spatial Network Characteristics. In *10th AGILE Int. Conf. on Geographic Information Science*, 2007.

[19] Scheider, S., Kuhn, W. (2008). Road Networks and Their Incomplete Representation by Network Data Models. *In*: 5th Int. Conf. on Geographic Information Science. p. 290–307.

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

[21] Schneider, M., Behr, T. (2006). Topological Relationships between Complex Spatial Objects, *ACM Trans. on Database Systems* 31 (1) 39–81.

[22] Shekhar, S., Yoo, J. S. (2003). Processing In-Route Nearest Neighbor Queries: A Comparison of Alternative Approaches. *In:* 11th ACM Int. Symp. on Advances in Geographic Information Systems. p. 9–16.

[23] Speičys, L., Jensen, C. S. (2008). Enabling Location-based Services–Multi-Graph Representation of Transportation Networks, *Geoinformatica* 12 (2) 219–253.

[24] Ulugtekin, N. N., Dogru, A. O., Thomson, R. C. (2004). Modelling Urban Road Networks Integrating Multiple Representations of Complex Road and Junction Structures. *In* 12th Int. Conf. on Geoinformatics -Geospatial Information Research: Bridging the Pacific and Atlantic.