MONET: Modeling and Querying Moving Objects in Spatial Networks

Lin Qi & Markus Schneider[*]
University of Florida
Department of Computer & Information Science & Engineering
Gainesville, Florida, USA
{lqi,mschneid}@cise.ufl.edu

ABSTRACT

Data about moving objects is being collected in many different application domains with the help of sensor networks, and GPS-enabled devices. In most cases, the moving objects are not free to move, they are usually restricted by some spatial constraints such as *Spatial Networks*. Spatial networks are ubiquitous and have been widely used in transportation, traffic planning, navigation as well as in Geographical Information System (GIS) applications. In most scenarios, moving objects such as vehicles move along predefined spatial networks like transportation networks. Unfortunately, the concepts for modeling and querying objects in unconstrained spaces like an outdoor space cannot be transferred to constrained spaces like a road network due to the different features of the environments in which the spatial objects move. Further, modern positioning devices as well as mobile and sensor technology have led to large volumes of moving objects in spatial networks. Therefore, we need a database-friendly data model to explicitly model spatial networks and, more importantly, describe relative movements in these networks. In this paper, we propose a new two-layered data model called *MONET* (Moving Objects in NETworks) model. The lower layer is a data model for spatial networks. This data model is the prerequisite for the upper model that represents moving objects in these networks. This layered model fits well to formulate relationships between moving objects and a network in queries. A query language, called *MONET QL* (MONET Query Language), allows a clear description of and access to moving objects in spatial networks and to provides high-level operations on them.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications – *data mining, spatial databases and GIS*

[*]This work was partially supported by the National Science Foundation under grant number NSF-IIS-0812194.

General Terms

Theory, Design

Keywords

Moving Objects, Spatial Networks, Modeling, Querying

1. INTRODUCTION

The field of *moving object databases* has received a lot of research interest in recent years. The advances in mobile communication and database technology have enabled innovative mobile applications monitoring moving objects. In most applications, the object movement is restricted by spatial constraints, like a vehicle travelling in a road network, or water flowing in a pipeline network. In these scenarios, objects cannot move freely in space, but are constrained by a spatial barrier. All these networks are characterized by an embedding of geometric objects (e.g., points, lines) in space and are denoted as *spatial networks*. Spatial networks or *spatially embedded graphs* are needed in Geographical Information Systems, cartography, planning public transport network, and for navigation assistance. They are an important spatial concept in the geo-sciences and have been widely discussed in the literature. The largely increasing amount of generated data about spatial networks can only be efficiently stored and analyzed in a database system. Therefore, database support for large spatial networks in order to represent, store, query, and manipulate them is important. Further, adding movement relative to a spatial network leads us to *moving objects in spatial networks*. That is, the motion of spatial objects (often point objects) is constrained by a static network. This leads to novel and interesting operations and predicates, thus new and, compared to the unconstrained case, different kinds of queries. The increasing interest and importance of moving objects in spatial networks has led to a large increase of generated spatial networks data and moving objects data research and modeling issues, which will be discussed in the related work section.

Real-time stream data acquisition through sensors and imagery devices has been widely used in many applications. Consequently, database support is essential to store the large volumes of such information and to utilize them in various GIS applications in an efficient way. As a result, providing a spatial data type for moving objects in spatial networks by means of an *ADT* (Abstract Data Type) which is integrated into a database system would allow users to query and manipulate moving objects in spatial networks in a database setting by high-level operations defined on them. In order to achieve such an abstract data type integration, providing a formally defined abstract model of spatial networks with moving objects should be the first issue and will in turn serve as the specification for upcoming potential implementations.

The first goal of this paper is to design a formal abstract data model for moving objects in spatial networks, namely MONET. The lower layer is a data model for spatial networks. This data model is the prerequisite for the upper model that represents moving objects in these networks. Unfortunately, the concepts for modeling and querying objects in unconstrained spaces like an outdoor space cannot be transferred to constrained spaces like a road network due to the different features of the environments in which the spatial objects move. For example, most of existing work considers Cartesian (typically, Euclidean) spaces, where the distance between two objects is determined solely by their relative position in space. However, in practice, the movement constrained by the pre-defined environment is clearly different from free space in which the network distance should be measured instead, i.e., the length of the shortest trajectory connecting two objects, rather than their pure Euclidean distance. Therefore this model is explicitly different from the model of moving objects in unconstrained environments. Besides, from a conceptual view, the upper layer model should be constructed based upon the network model, which necessitates the network model to be easily and friendly extended to any add-on movements. The lower layer model 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. In most cases, spatial networks can be modeled as different point sets with respect to different network components where all points with equal thematic properties belong to the same network component. A necessity of this network model is to fulfill the needs that the needs of applications. Various queries will be posted on the moving objects in spatial networks, and the network model should be designed to easily support numerous meaningful operations and predicates as well as friendly to potential extensions. The second goal is the design of MONET QL. This SQL-like query language is developed to allow a clear description of and access to moving objects in spatial networks and to provides high-level operations on them. It should take advantage of the defined operations and predicates smoothly. Some examples queries in applications scenarios will demonstrate the merits of such query language.

In this paper, we confine ourselves exclusively to road networks as field of study, which contains all existing and potential features of a spatial network, thus not loses generality. The rest of the paper is organized as follows: we summarize the related work in Section 2. Section 3 shows the application-side analysis which would stimulate our model needs. Section 4 develops the lower layer model of spatial networks, it also gives a number of operations performed on spatial network components. Section 5 describes the upper layer model on moving objects in spatial networks as well as corresponding operations. This model is constructed based upon the lower layer network model proposed in the previous section. A formally defined query language for moving objects in spatial networks, *MONET QL*, is presented in Section 6 with some example applications and queries. Finally, Section 7 concludes the paper together with some future improvements.

2. RELATED WORK

In this section, we classify the related literature of this work into three parts, the spatial networks model, the models for moving objects in free space and moving objects in spatial networks.

Spatial networks are used in Geographical Information Systems, cartography, planning public transport network, and for navigation assistance. They are an important concept in the geo-sciences and have been widely discussed in the literature [1, 2, 3]. A natural approach in [3] and [1] is to model spatial networks as planar graphs in order to capture their structure and connectivity. In this way, queries such as *the shortest path problem* may be solved by Dijkstra's algorithm by simply mapping to well-known graph problems. The model in [3] has been designed for embedding graphs in databases and not specifically for spatial graphs. The work in [1] models road networks by defining road components based on their properties and the potential actions they may perform. Transportation networks are an important example of spatial networks and received empirical research and analysis (Figure 1 shows the road network of North America from [4]). Besides, routes correspond to paths in a graph and to roads in a transportation network, they are essential concepts in a road network, as shown in [2].

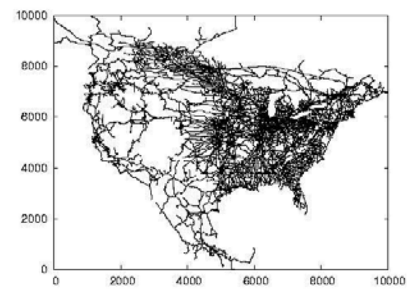


Figure 1: Road network of North America

Moving objects such as hurricanes, cars and animals have been studied intensively in the past decade in terms of moving object database [5, 6]. The MOST model [7] uses a particular dynamic location attribute which includes a polyline representing a path over the network plus some additional information like start time, start position, and speed. The fundamental idea is to introduce so-called dynamic attributes which change their values automatically with time. However, the underlying network is not modeled in any way.

Considering that spatial objects like cars, planes, trains, and people in buildings are restricted in possible motions since they move in spatially embedded networks like roads, highways, railways, lanes in factories for robots, buildings, or airplane routes. We speak of such scenario as moving objects in constrained environments [8]. As a conclusion, spatial networks should be taken into account in a data model and database query language for moving objects. In this way, it would be possible to describe movement relative to a network rather than the general 2D space. It would also enable easier formulation of queries and more efficient representations and indexing of moving objects for network-based database applications. Unfortunately, the existing network models may not be suitable for analysing and querying moving objects in them. In these models, a network can be explicitly stored in a database which enables the formulation of powerful queries and efficient execution. However these approaches cannot store and represent the thematic attributes of the network components. From another perspective, network alone cannot express relationships between moving objects and static objects in the network. Furthermore, the concepts for modeling and querying objects in unconstrained spaces like an outdoor space cannot be transferred to constrained spaces like a road network due to the different features of the environments in which the spatial objects move. The increasing interest and importance of moving objects in spatial networks has led to a large increase of generated spatial networks data and moving objects data research and modeling issues [7, 9, 10, 11]. However, related formal data models for moving objects in spatial networks are rare. The approaches in [12] look at data modeling issues for spatial networks regarding possible uses for location-based services. These interesting application studies emphasize the large complexity of real road networks that cannot be adequately modeled by simple directed graph models. The approach in [2] is so far the only one proposing data types for moving objects in networks. The lack of a formal abstract data model of moving objects in spatial networks is easy to see: no explicit relationship with the network, low efficiency of querying and processing, object-oriented storage problem in database, to name a few. In our proposed model MONET, we will come up with explicit definitions of moving objects and describe how they are embedded in spatial networks.

3. MOVING OBJECTS IN SPATIAL NETWORKS: MOTIVATION, APPLICATIONS, AND REQUIREMENTS

3.1 Overview

It is assumed here that a database consists of a set of *object classes* of different schemas. Each object class has an associated set of *objects*; each object has a number of *attributes* with values drawn from certain *domains* or *atomic data types*. For example, traditionally we have the following schema for moving objects class:

$$MovingObject\ (mid, mwidth, mlength, mheight, mweight, mtype)$$

where *mid* is the id of the moving object, this may refer to the licence of the vehicle when considering the road network environment, and *mwidth*, *mlength*, *mheight* and *mweight* refer to the width, length, height, and weight of the moving object while $mtype \in \{car, taxis, bus, truck, limousine, tractor, servicevehicle, unclassified\}$ indicates which type it belongs to. Besides, the schema for the road network can be designed in terms of *routes* ([2, 13], refer to the *lanes* of a road network, will be explained in detail in Section 5) as:

$$Route(rid, roadid, roadname, speed\_lmt, width\_lmt, height\_lmt, weight\_lmt, start, end, isDirected, rtype)$$

where *rid* refers to the id of the route, *roadid* and *roadname* give the id and name of the road that this route belongs to. *speed_lmt*, *width_lmt*, *height_lmt*, *weight_lmt* are the limits of this particular traffic route, for example a truck of height exceeds the height limit of the route is not allowed to travel along such road. *start*, *end* point out the starting location and the ending location for this route and *isDirected* is set to ± 1 if it is directed, set to 0 if it is dual directed. In addition, $rtype \in \{motorway, highway, interstate, local, residential, service, unclassified\}$ indicates which type the route belongs to.

However, such schemas are not enough. For example, if we want to answer query like "Where was taxi T at 2:00 pm yesterday?" we still need another table storing the location information of the moving object. This requires the join of tables and make the query processing complicated. From the modeling in Section 4 and Section 5, we would integrate those proposed abstract data types into traditional database context. By the corresponding modifications to schemas, it benefits us both from the conceptual level as well as the implementation level. These aforementioned attributes are only of thematic meanings, we now consider extensions to the basic model to capture time and space. Besides objects, attributes describing are of particular interest. Hence, in the next subsection, we would like to define collections of abstract data types, or in fact many-sorted algebras containing several related types and meaningful operations, for spatial values changing over time. Generally, there exists a validity interval for each object in the database since it would be thrown away at some later time. As a simplification, and to be able to cooperate with standard data models, this validity interval can be omitted here and we can just rely on time-dependent attribute values described later on.

3.2 Motivation

We assume the presence of two kinds of objects: moving and static. The movements of the moving objects are restricted to the road network including cars, taxis, trucks, etc which are capable of continuous movement. The static objects, all of which may be accessible via the road network, are the objects of interest ot the moving objects, for instance gas stations, airports, and supermarkets. Further, as general, we also assume that the moving objects are on-line and may communicate their locations information. This positioning functionality may be accomplished via the Global Positioning System, and the acceptance terminal would be able to update the information correspondingly in a database context. This setting, [14], allows the moving objects to use services that involve various location-based queries about the moving objects in the road network. For example, range queries and k-nearest neighbor queries are very important and meaningful queries which can be answered under such environment. The following is an example query: "Find me an up-to-date list of the three nearest gas stations within 5 miles." To process such queries, appropriate representations of road networks and moving and static objects are needed. As the moving objects are restricted to the road network, one natural and clear observation would be the measure of distance in the network. Obviously, network distance should be measured instead, i.e., the length of the shortest trajectory connecting two objects, rather than their pure Euclidean distance.

To design an abstract data model for representing such data, the most important property that attracts attention is the fidelity. The higher the degree of expressibility with more details that are captured from the real world, the more precision this model would be able to represent the real world. However, we would have to come to the efficiency issues when approaching the implementation level of this abstract model. Thus a more complex and voluminous model may be less efficient for querying while a simple, low-fidelity representation may be most amenable to efficient query handling. Often, one must be trade for the other. But here in abstract level, what we most care is the precision and the expressiveness that the model can reach, and later on in implementation we would try to optimize the query processing while maintain the complete and required fidelity extension. This rule is both for the lower-layer network model as well as the higher-layer moving objects model. Let's first look at some example queries to light up our minds what exactly are the requirements and necessities that this model should hold.

3.3 Applications and Requirements

In this subsection, we consider the application scenario of taxis moving along road networks. Our fundamental objective is to derive the colloquial requirements for network model as well as the moving objects model. Besides, a goal is to support mobile services that exploit location information from the moving taxis to deliver the desired functionality. To illustrate the contribution of our model in this context, we refer throughout the article to the following typical queries (H, C, and F stands for queries on historical information, current information, and future information respectively) based on the schemas designed in Subsection 3.1:

Query C1. Which available taxi is closest to location 2200 West University Avenue?
Query C2. What is the shortest path to the gas station G for taxi T?
Query C3. Is taxi T moving north?
Query C4. On which road is taxi T moving along right now?
Query C5. What is the current speed of taxi T?
Query C6. How long is the trip of taxi T since its last stop?
Query C7. Find all taxis which have travelled more than 40 miles from 1 hour ago till now.
Query C8. Return the taxis which have stayed in the same road for more than 1 hour.

In query C1 and C2, first of all the distance measure of two locations should be computed precisely using network distance operation *netDist*. This operation requires the network to provide the length information of roads and the road geometric information if needed. Second, a retrieval operation *getLocation* should be designed to get the location of a moving object at a certain time instance. And finally pick up the nearest taxi and return the result. Query C3 can be realized by an operation *getDirection* which takes the moving object as the parameter and return the moving direction. Query C4 returns the road that a taxi is moving on, this can be achieved by an operation *getRoad* that takes in the current location and return the road that this location belongs to. However, such a relationship should be kept within the network model and can be easily retrieved. What is more interesting is query C6, we may take into account of the historical location information to decide at which point the taxi stopped. Query C5, C7, and C8 is similar to C3, which needs a near-past location information of the certain moving object. Followed by is a list of queries on historical information:

Query H1. Where was taxi T at 2:00 pm yesterday?
Query H2. What was the first time that taxi T arrived the airport A in the last week?
Query H3. How long did taxi T park at the airport A?
Query H4. In the last month, which roads were visited most often?
Query H5. For the last week, how many times did taxi T visit gas station G?
Query H6. Find all taxis that visited airport A in the afternoon (2:00 pm through 6:00 pm) of last Friday.
Query H7. How many taxis passed gas station G yesterday?
Query H8. For taxi T, how many times did it change its moving direction in the past 30 minutes?
Query H9. Find all the taxis that have exceeded the speed limit by more than 30% of road R at 2:00 pm yesterday.
Query H10. Find cars that have violated the U-turn-forbidden rule at junction J in the last 2 hours.
Query H11. What was the average speed of taxi T on the interstate-75?

Query H1, H2, and H3 only consumes the information that has been stored in the database and then make a query. Query H4 is relative interesting, we have to first lift the taxis that have ever travelled in each roads and cumulate the times. This requires the network model to cooperate with the moving object model on the locations. In query H5, we would count the number of visits that taxi T hit gas station G. In this query, there should be a predicate indicating whether a taxi has arrived a gas station, which means a moving point trajectory contains a static network point. Similar to query H5, in query H6, we also need a predicate *inside* to check whether a sub-trajectory has traversed a road that the airport belongs to. Query H7 needs to count the number of taxis that have passed a network location in a given time period. Query H9 first retrieves the speed limit of road R then compares it with the actual velocity of taxis that have travelled in this road. Query H10 also requires the network to provide basic information of junctions, whether U-turns are permitted at these junctions. Query H8 and H11 are just retrieval queries, for which the model just needs to offer such operations. Besides, two queries based on future information are also presented here:

Query F1. Return all the taxis that will reach the junction of Highway 201 and I-10 in the next 10 minutes.
Query F2. Will it be able to reach the airport A in less than 10 minutes if taxi T keep the speed limit?

Query F1 will need to traverse all the taxis that are moving towards the junction of Highway 201 and Highway 10 to get their travelling velocities and corresponding distance to the junction at present. Then a predicate indicating whether a taxi can reach the junction in a time period will be applied and return all the taxis that hold such a predicate. Similar to query F1, in query F2, by retrieving the speed limit of the road that taxi T is moving on, and make a computation whether T can reach the airport with the shortest path under the speed limit.

4. A CONCEPTUAL LOWER-LAYERED MODEL OF SPATIAL NETWORKS

In this section we will introduce our model of spatial networks and their operations. Section 4.1 provides some intuitive descriptions of spatial networks. In Section 4.2 we give the formal description of our lower-layered model on spatial networks. We close this section by introducing the operations defined on spatial networks. As usual, we would again use road networks as our field of study.

4.1 What Are Spatial Networks

A spatial network is a network of spatial elements. More generally, the term spatial network has come to be used to describe any network in which the nodes are located in a space equipped with a metric. For most practical applications, the space is the two-dimensional space and the metric is the usual Euclidean distance. This definition implies in general that the probability of finding a link between two nodes will decrease with the distance. Transportation and mobility networks, internet, mobile phone networks, power grids, social and contact networks, neural networks, water pipelines are all examples where space is relevant and where topology alone does not contain all the information.

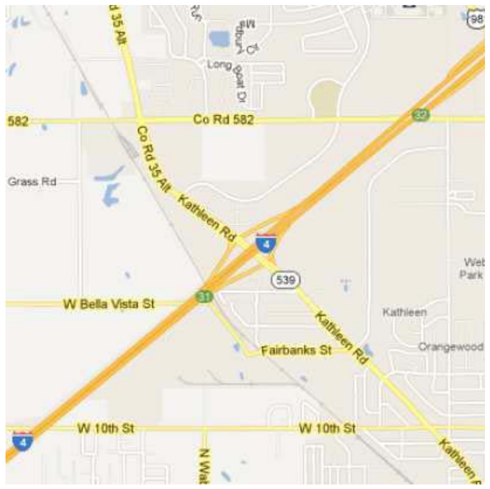
Figure 2(a) shows a map depicting a road network. And we simplify the representation of this road network as an abstract draft in Figure 2(b). We have extracted and highlighted the back bone roads from Figure 2(a) and draw them out in the corresponding area in Figure 2(b). Some interesting points draw our attention. *Junctions* are network locations where two or more routes intersect. For example, the point j_1 is a junction because at this point, the roads *W McNab Rd* and *SW 63rd Terrance* intersect. On the other hand, there exist locations where two or more routes interact but not actually accessible from each other. Point c_2 is a *crossover* at which the *Ronald Reagan Turnpike Highway* and *NW 62nd St* cross, i.e., cars cannot switch directly from one route to another at this point. Examples in real world may include that one road forms a bridge or a tunnel over the other. Readers may question that one location could be both of the two cases, and the answer is true: Junctions and crossovers are not mutually exclusive. One example could be that an interstate and a local street connects directly by the ramps, at the mean time, the interstate crosses the street with a vertical gap. r_1, r_2, ..., r_5 are *routes* of the road network, they're actually the constraints in this spatial network. Boundary points like p_1, p_2 form the limiting boundaries of the network.

4.2 Definitions and Operations of Spatial Networks

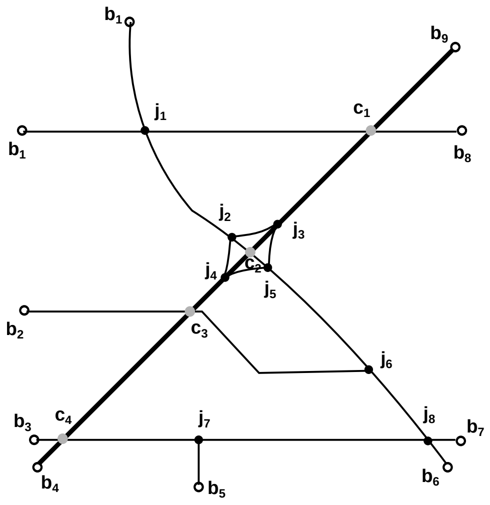
In this section, we provide data types *network*, *npoint*, *nline* and *nregion* to represent the network, a location lying in the network, a line within the network and a region within the network, respectively. By intuition, all points lie on a road network belong to some certain *Routes*. A route means a lane of a road connecting two different locations in the network. The value of an npoint can be a 2D coordinates of a location within the network, or be undefined if lying outside the network plane. It is usually referred to as static positions in the network, including normal point, junction, and crossovers. Type nline is used to represent the route, and nregion is for the representation of a static region in the network.

As we have introduced in Section 3, the schema for route is not enough to represent all the information. Therefore we can modify the schema of *Route* mentioned in Section 3 as

$$Route(rid, roadid, roadname, rprofile)$$



(a) Real road network



(b) Abstract road network

Figure 2: Example of a Spatial Network: A Road Network Glance

where *rprofile* is of type nline. We can then define all kinds of retrieval operations to get the speed limit, width limit, height limit, weight limit, starting point, ending point, isDirected and route type information, for example:

$$\textbf{speedlmt:}\ nline \times npoint \rightarrow real \tag{1}$$

$$\textbf{isDirected:}\ nline \times npoint \rightarrow int \tag{2}$$

$$\textbf{routeType:}\ nline \times npoint \rightarrow rtype \tag{3}$$

All of these are accessing route information operations, and they all return the information at a given location within that route. For example, *speedlmt* returns the speed limit of a route at a given location within that route. It will first check whether the given location is not on that route, if not, a NULL value will be returned directly without further calculation.

We can get routes, junctions, and crossovers of a network by the component retrieval operations

$$\textbf{routes: } \mathit{network} \rightarrow \mathit{nline} \quad (4)$$
$$\textbf{junctions,crossovers: } \mathit{network} \rightarrow \mathit{npoint} \quad (5)$$

An npoint contains a position on a given route, possibly on one of its sides. We should be able to access these components by:

$$\textbf{atRoute: } \mathit{npoint} \rightarrow \mathit{int} \quad (6)$$
$$\textbf{relativePos: } \mathit{npoint} \rightarrow \mathit{real} \quad (7)$$
$$\textbf{side: } \mathit{npoint} \rightarrow \mathit{int} \quad (8)$$

This overloaded operator *atRoute* could also take a network point and give back the route id it belongs to. *RelativePos* returns the relative position of the network point in the corresponding route, and *side* returns which side of the route the point resides, where *left*=-1, *right*=1, *none* =0. We also have the operation of computing the length of a given route or sub-route, i.e. the length between any two points within the same route, as:

$$\textbf{length: } \mathit{nline} \rightarrow \mathit{real} \quad (9)$$
$$\textbf{length: } \mathit{nline} \times \mathit{npoint} \times \mathit{npoint} \rightarrow \mathit{real} \quad (10)$$

This overloaded operator is far more useful when dealing with more complicated problems such as *shortest path*. To be clear, we list out a number operations that would be useful for processing queries in Table 1. For the predicates, the meaning should be obvious. In retrieval operations, *nlocation* returns a network point given a relative position $e \in [0, 1]$ and the corresponding route. For example, a value of $e = 0.5$ indicates the location is at the mid position of the route. In the set operations, we haven't taken *minus* operation into account. For the operator *npath*, it takes in two locations in the network and returns a path of type nline, which consists of a continuous intersected routes $r_i,\ r_{i+1}, ..., r_j$. One way to make this possible could be:

$$\begin{aligned}&\textbf{npath: } \mathit{nline} \times \mathit{real} \times \mathit{nline} \times \mathit{real} \rightarrow \mathit{nline}\\ &\quad (i)\ \ \forall k \in [i,\ j-1],\ r_k \in \mathit{nline} \in \mathit{routes}(\mathit{network})\\ &\quad (ii)\ \mathit{intersect}(r_k,\ r_{k+1}) = \mathit{true}\end{aligned}$$

5. MODELING MOVING OBJECTS IN SPATIAL NETWORKS

Abstract models allow us to make definitions in terms of infinite sets, without worrying about implementation issues like whether finite representations of these sets exist. This feature benefits us a lot, for example, we could view a moving point as a continuous curve in the 2D space, as an arbitrary mapping from an infinite time domain into an also infinite space domain. All the types that we get by application of the type constructor τ are functions over an infinite domain, hence each value is an infinite set.

5.1 Definitions of Moving Objects in Spatial Networks

This abstract view is the conceptual model that we are interested in. We consider two basic types *mpoint* and *mregion* here for representing the moving objects in road networks. The majority of the moving objects in networks can be represented by a *moving*-point whose size can be omitted at an abstract level but stored as attributes in the database sticking to this moving point. The other perspective of moving object is the *moving*-region, which reflects a region that is continuously moving and changing its shape. These two basic types here are enough to express the realities. Let us assume that purely spatial data types called *point* and *region* are given that describe a point and a region in the 2D-plane. A point here reflects a simple point ($\in \mathbb{R}^2$), and a simple moving object is represented by an mpoint. A region may have holes, and generally, we would use mregion to represent moving object like a traffic jam which consists an area of vehicles. Further, a type *time* describes the valid time dimension and is isomorphic to the real numbers ($\in \mathbb{R}$). Then we can view the types mpoint and mregion as mappings from time into space as:

$$\mathit{mpoint} = \mathit{time} \rightarrow \mathit{point}$$
$$\mathit{mregion} = \mathit{time} \rightarrow \mathit{region}$$

and we can use retrieval operator *mlocation* to retrieve the position of a mpoint and a center point of a mregion at a time instance:

$$\textbf{mlocation: } \mathit{mpoint} \times \mathit{time} \rightarrow \mathit{point}$$
$$\textbf{mlocation: } \mathit{mregion} \times \mathit{time} \rightarrow \mathit{point}$$

However, since the moving points are restricted by the constrained environment, e.g. road network, the complete definition for moving points in spatial networks are defined as:

$$\begin{aligned}\mathit{mpoint} =\ & \mathit{time} \rightarrow \mathit{point}\\ & (i)\ \ \forall p \in \mathit{point} \in \mathbb{R}^2\\ & (ii)\ \forall t \in \mathit{time},\ \exists e \in [0,1],\ r \in \mathit{routes}(\mathit{network})\\ & \qquad s.t.\ \mathit{nlocation}(r, e) = p\\ & (iii) \forall \epsilon > 0,\ \exists \delta > 0,\ \forall t_1, t_2 \in \mathit{time}:\\ & \qquad |t_1 - t_2| < \delta,\ s.t.\\ & \qquad\qquad \mathit{ndistance}(\mathit{mlocation}(\mathit{mpoint}, t_1),\\ & \qquad\qquad\qquad \mathit{mlocation}(\mathit{mpoint}, t_2)) < \epsilon\end{aligned}$$

Condition (i) ensures that the moving point moves in 2D plane, and condition (ii) restricts all the possible locations that a moving point can reach must belong to a certain route, i.e. belongs to the network. Condition (iii), often refers to as the *continuous restriction*, ensures that the movement of the moving point is continuous. Obviously, a moving point should move along a continuous curve, only continuity reflects the real movement. More generally, we can introduce a type constructor τ which transforms any given atomic data type α into a type $\tau(\alpha)$ with semantics as

$$\tau(\alpha) = \mathit{time} \rightarrow \alpha$$

Figure 3 shows a simple moving point that move either in free space or road network. We assume that space and time dimensions are continuous which are isomorphic to the real numbers. A value of type mregion is a set of volumes in the 2D space plus time (x, y, t). Given a plane $t = t_1$ of time domain, mregion will yield a region value, describing the shape and contour of the moving region at t_1. It is possible that this region can shrink and disappear, as it can grow and come into being, i.e. an empty region is also a proper region value. Figure 4 shows an example of moving regions in both free space and in road networks.

Table 1: Minimal set of operations that the network should support

Description	*Signature*		*Operators*
Predicates	$npoint \times npoint$	$\rightarrow bool$	$=, \neq$
	$npoint \times nline$	$\rightarrow bool$	**inside**
	$npoint \times nregion$	$\rightarrow bool$	**inside**
	$nline \times nline$	$\rightarrow bool$	**intersect**
Component operations	$network$	$\rightarrow nline$	**routes**
	$network$	$\rightarrow npoint$	**junctions, crossovers**
Accessing operations	$nline \times npoint$	$\rightarrow real$	**speedlmt, widthlmt**
			heightlmt, weightlmt, isDirected
	$nline \times npoint$	$\rightarrow rtype$	**routeType**
Retrieval operations	$npoint$	$\rightarrow int$	**atRoute**
	$npoint$	$\rightarrow real$	**relativePos**
	$nline \times real$	$\rightarrow npoint$	**nlocation**
	$nline \times npoint \times npoint$	$\rightarrow real$	**side**
Set operations	$npoint \times npoint$	$\rightarrow npoint$	**intersection**
	$npoint \times nline$	$\rightarrow npoint$	**intersection**
	$npoint \times nline$	$\rightarrow nline$	**union**
	$nline \times nline$	$\rightarrow nline$	**union,intersection**
Auxiliary operations	$npoint \times npoint$	$\rightarrow nline$	**npath**
	$nline \times real \times npoint$	$\rightarrow nline$	**npath**
	$nline \times real \times nline \times real$	$\rightarrow nline$	**npath**
	$npoint \times npoint$	$\rightarrow real$	**ndistance**
	$nline \times real \times npoint$	$\rightarrow real$	**ndistance**
	$nline \times real \times nline \times real$	$\rightarrow real$	**ndistance**

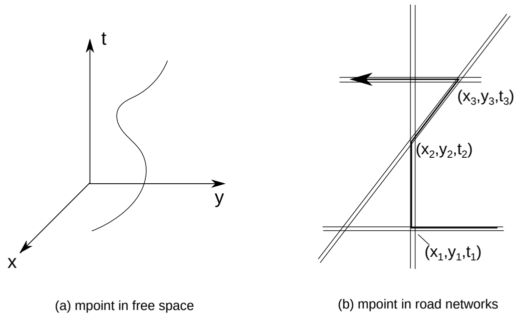


Figure 3: Simple moving points

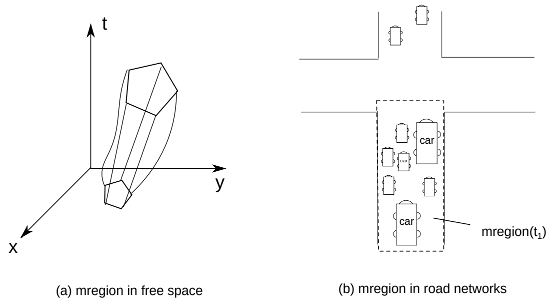


Figure 4: Simple moving regions

5.2 Operations on Moving Objects in Spatial Networks

Now it is time to describe a relatively comprehensive set of operations for these data types of moving objects in spatial networks. Generic operations for moving objects include, for example:

$$\textbf{atTime: } \tau(\alpha) \times time \rightarrow \alpha \tag{11}$$

$$\textbf{atPresent: } \tau(\alpha) \rightarrow \alpha \tag{12}$$

Operator *atTime* returns the value of moving object given a particular time instance. When referring to a simple moving point, this operator is equivalent to the operator *getLocation* mentioned in Section 3.3. *atPresent* is an operator that returns the value of moving object at present.

$$\textbf{start: } \tau(\alpha) \rightarrow time \tag{13}$$

$$\textbf{end: } \tau(\alpha) \rightarrow time \tag{14}$$

Start and *end* give the minimum and maximum of a moving value's time domain, respectively.

$$\textbf{lifetime: } \tau(\alpha) \rightarrow real \tag{15}$$

Operator *lifetime* will give the total length of time intervals that a moving object value is defined, namely the existing lifetime of the moving object.

$$\textbf{directionAt: } \tau(\alpha) \times time \rightarrow dcode \tag{16}$$

$$\textbf{directionPresent: } \tau(\alpha) \rightarrow dcode \tag{17}$$

It is quite important and useful to tell the direction of movement in road networks when posting queries like "Find all the taxis who are moving towards north along Main Street". The operator *directionAt* takes a certain time instance and returns the moving direction as the result, a type that we call *dcode*. This type is hard-coded, and roughly means a value in the thematic set $\{Anchor, North, South, ...\}$, and table 2 reflects the mappings, where anchor means that the moving object has stopped, and other values give the moving

Table 2: Direction Mapping Table (*DMT*)

dcode	0	1	2	3	4	5	6	7	8
Direction	Anchor	N	W	S	E	NW	NE	SW	SE

directions.

$$\textbf{speed: } \tau(\alpha) \rightarrow mreal \tag{18}$$

$$\textbf{speedAt: } \tau(\alpha) \times time \rightarrow real \tag{19}$$

$$\textbf{speedPresent: } \tau(\alpha) \rightarrow real \tag{20}$$

Speed returns the moving speed of a moving object at all times and hence returns a time changing real number, a type that we name as *mreal*:$mreal = \tau(real)$. And *speedAt*, *speedPresent* returns the moving speed at a given time instance or the present speed, respectively.

$$\textbf{atLocation: } \tau(\alpha) \times \alpha \rightarrow time \tag{21}$$

Compared to *atTime*, *atLocation* returns the exact time that the moving object is of a particular value. This operation is quite useful when processing query like "At what time did postman Jim visited cityhall yesterday?".

$$\textbf{atRoute: } \tau(\alpha) \times \alpha \times time \rightarrow int \tag{22}$$

This operator plays with the network together, as have been discussed in the previous section. *AtRoute* returns the route identifier that the moving object is moving on. This information is often requested by queries like "On what road that the taxi T is moving on? Does postman Jim still drive along University Avenue?"

Particularly, we may also have operations for spatial moving objects as:

$$\textbf{mdistance: } mpoint \times mpoint \rightarrow mreal \tag{23}$$

$$\textbf{mdistance: } mpoint \times mregion \rightarrow mreal \tag{24}$$

Operator *mdistance* computes the distance between the two moving points at all times and therefore returns a time changing real number, *mreal* as aforementioned. At the same time, we could overload this operator by taking parameters of mpoint and mregion to get the distance between the moving point and the moving region at all times.

Operations may also involve pure spatial or pure temporal types and other auxiliary types. Let *line* be a data type describing a curve in 2D plane which may consist of several disjoint pieces (which means a complex line); let *region* be a type for regions in the space which may consist of several disjoint faces with holes. Therefore we may have operations like

$$\textbf{trajectory: } mpoint \rightarrow line \tag{25}$$

$$\textbf{traversed: } mregion \rightarrow region \tag{26}$$

Here *trajectory* represents the projection of a moving point onto the plane. And *traversed* operator is the corresponding projection for moving regions, which gives the total area the moving region has ever covered. We may also need the following predicate and operations:

$$\textbf{inside: } point \times region \rightarrow bool \tag{27}$$

$$\textbf{length: } line \rightarrow real \tag{28}$$

$$\textbf{mlength: } mpoint \rightarrow real \tag{29}$$

where *inside* checks whether a point lies into a region. *Length* and *mlength* return the length of a line type, and particularly, mlength could take a moving point as an input and return the total length of the trajectory of the moving point throughout lifetime.

The presented data types can now be embedded into any DBMS data model as attribute data types, and the operations be used in queries. Therefore we can modify the schema of *MovingObject* mentioned in Section 3 as

$$MovingObject$$
$$(mid, mwidth, mlength, mheight, mweight, mtype, traj)$$

by adding an attribute *traj* of type *mpoint*. Further more, we could process queries based on this modified schema. This will be discussed in Section 6. To be clearer, at this end, we list out a number operations that would be useful for processing queries in Table 3. *Mnintersect* returns the result whether the moving point intersects with a given route in the network. Operator *mndistance* computes the distance between a network point and a moving point at a time instance. More interestingly, *mnpath* will return the shortest path from a moving point location at a time instance to another network point or even another moving point.

6. MONET QL: A QUERY LANGUAGE FOR MOVING OBJECTS IN SPATIAL NETWORKS

MONET QL extends the SQL by adding and taking advantage of the self-defined functions. Fortunately, we can write user-defined functions in PL/SQL or Java to provide functionality that is not available in SQL or SQL built-in functions. User-defined functions can appear in a SQL statement anywhere SQL functions can appear, that is, wherever an expression can occur.

6.1 Query Scenarios

hpabilities of a DBMS is to easily integrate new types and operations in the SQL standard interface. The query examples in this section are based on a relational schema with one table that contains information on vehicle trips as follows:

$$Taxis(id, width, length, height, weight, type, trajectory)$$
$$Route(rid, roadid, roadname, rprofile)$$
$$GasStations(gid, brand, rid, rloc, side, gprofile)$$
$$Airport(aid, rid, rloc, side, aprofile)$$

6.2 Example Queries

In this subsection, due to the page limitations, we will take seven of the queries proposed in Section 3 and expand them into details. *CurrentTime* below refers to the current time. And to be clear, we also provide the time interval

Table 3: Moving objects in spatial networks related operations

Description	*Signature*		*Operators*
Predicates	$mpoint \times mpoint$	$\rightarrow bool$	$=, \neq$
	$mpoint \times mregion$	$\rightarrow bool$	**inside**
	$mregion \times mregion$	$\rightarrow bool$	**intersect**
Retrieval operations	For $\alpha \in \{mpoint, mregion\}$		
	$moving(\alpha) \times time$	$\rightarrow \alpha$	**atTime**
	$moving(\alpha)$	$\rightarrow \alpha$	**atPresent**
	$moving(\alpha) \times time$	$\rightarrow point$	**mlocation**
	$moving(\alpha)$	$\rightarrow time$	**start, end**
	$moving(\alpha)$	$\rightarrow real$	**lifetime**
	$moving(\alpha) \times time$	$\rightarrow dcode$	**directionAt**
	$moving(\alpha)$	$\rightarrow mreal$	**speed**
	$moving(\alpha) \times time$	$\rightarrow real$	**speedAt**
	$moving(\alpha)$	$\rightarrow real$	**speedPresent**
	$moving(\alpha) \times \alpha$	$\rightarrow time$	**atLocation**
	$moving(\alpha) \times \alpha \times time$	$\rightarrow int$	**atRoute**
Auxiliary operations	$mpoint \times mpoint$	$\rightarrow mreal$	**mdistance**
	$mpoint \times mregion$	$\rightarrow mreal$	**mdistance**
	$mpoint \times mregion$	$\rightarrow mpoint$	**mintersect**
	$mpoint \times nline$	$\rightarrow bool$	**mnintersect**
	$mpoint$	$\rightarrow line$	**trajectory**
	$mregion$	$\rightarrow region$	**traversed**
Metric operations	$mpoint$	$\rightarrow real$	**mlength**
	$mpoint \times time \times npoint$	$\rightarrow real$	**mndistance**
	$mpoint \times time \times npoint$	$\rightarrow nline$	**mnpath**
	$mpoint \times time \times mpoint \times time$	$\rightarrow nline$	**mnpath**

constructing operations as auxiliary operations:

$$\textbf{year: } int \rightarrow time \quad (30)$$

$$\textbf{month: } int \times int \rightarrow time \quad (31)$$

$$\textbf{day: } int \times int \times int \rightarrow time \quad (32)$$

$$\textbf{hour: } int \times int \times int \times int \rightarrow time \quad (33)$$

$$\textbf{minute: } int \times int \times int \times int \times int \rightarrow time \quad (34)$$

$$\textbf{second: } int \times int \times int \times int \times int \times int \rightarrow time \quad (35)$$

For example, times *year*(2011), *day*(2011, 6, 29), or even *minute*(2011, 6, 29, 14, 30) can be constructed using such operations. Besides, *period* is introduced as a type of a duration of time, having the constructor

$$\textbf{period: } time \times time \rightarrow period \quad (36)$$

Query 1 (C2). What is the shortest path to the gas station G for taxi ASCC96?
SELECT npath(atPresent(trajectory),nLoction(gprofile))
FROM Taxis AS t, Route AS r, GasStations AS g
WHERE t.id = 'ASCC96' AND g.gid = 'G'

The operation **npath** will return the shortest path between two locations in the network.

Query 2 (C3). Is taxi ASCC96 moving north?
SELECT t.id
FROM Taxis AS t
WHERE directionAt(trajectory, currentTime) = 1

It is quite important and useful to tell the direction of movement in road networks. The operation *directionAt* takes a certain time instance and returns the moving direction as the result, a type that we call *dcode*, in which *dcode =1* indicates North direction (See Tabel 2).

Query 3 (C4). On which road is taxi ASCC96 moving along right now?
SELECT r.roadname
FROM Taxis AS t, Route AS r
WHERE t.id = 'ASCC96'
AND r.rid = **atRoute**(trajectory, currentTime)

The operation **atRoute** returns the route that the moving object is moving along given a time instance.

Query 4 (C7). Find all taxis which have travelled more than 40 miles from 1 hour ago till now.
LET onehour_ago = **time**(2012,6,30,17)
LET Original_loc = ELEMENT(
SELECT mlocation(trajectory, onehour_ago)
FROM Route, Taxis
WHERE atRoute(trajectory, onehour_ago)
SELECT t.id
FROM Taxis AS t, Route AS r
WHERE ndistance(Original_loc, mlocation(trajectory, currentTime))

Query 5 (C8). Return the taxis which have stayed in the same road for more than 1 hour.
LET onehour_ago = **time**(2012,6,30,17)
LET Original_Road = ELEMENT(
SELECT rid **FROM** Route, Taxis
WHERE atRoute(trajectory, onehour_ago)
SELECT t.id
FROM Taxis AS t
WHERE Original_Road =
atRoute(t.trajectory, currentTime)

Query 6 (H2). What was the first time that taxi ASCC96 arrived at airport A in the last week?
LET lastweek =
period(day(2012, 6, 24), day(2012, 6, 30));
SELECT atLocation(t.trajectory, nloc(a.aprofile)) AS t
FROM Taxis AS t, Route AS r, Airport AS a

WHERE t.id = 'ASCC96' AND a.id = 'A'
AND t <= ANY
(**SELECT** atLocation(t.trajectory, nloc(a.aprofile))
FROM Taxis AS t, Route AS r, Airport AS a
WHERE t.id = 'ASCC96' AND a.id = 'A')

In this way, the earliest (first time) that taxi ASCC96 arrived at airport A can be selected.

Query 7 (H6). Find all taxis that visited airport A in the afternoon (2:00 pm through 6:00 pm) of last Friday.
LET FriA =
period(hour(2012, 6, 29, 14), hour(2012, 6, 29, 18));
LET road = ELEMENT(
SELECT rid **FROM** Airport **WHERE** aid ='A')
SELECT t.id
FROM Taxis AS t
WHERE road
inside trajectory(**atperiods**(trajectory, FriA))

The **inside** operation used here has the signature $int \times nline \rightarrow bool$.

Query 8 (H9). Find all the taxis that have exceeded the speed limit by more than 30% of road R at 2:00 pm yesterday.
LET yesterday2pm = **time**(2012, 6, 29, 14);
SELECT t.id
FROM Taxis AS t, Route AS r
WHERE speedAt(t.trajectory, yesterday2pm) >
1.3 * **speedlmt**(r.rprofile, mloction(t.trajectory, yesterday2pm))

Query 9 (F2). Will it be able to reach the airport A in less than 10 minutes if taxi ASCC96 keep the speed limit?
SELECT **reachable**(**speedlmt**(r.rprofile,
mloction(t.trajectory, currentTime),
period(minute(currentTime),
minute(currentTime + 10 minutes))),
ndistance(mloction(t.trajectory, currentTime),
nlocation(a.aprofile))
FROM Taxis AS t, Route AS r, Airport AS a
WHERE t.id = 'ASCC96' AND a.id = 'A'

Operator **reachable** has the signature $real \times time \times real \rightarrow bool$, it will return whether a moving object can travel a distance given a speed limit and a time interval.

As a conclusion, in this section, we only give some ideas about query processing issues and strategies in this framework. Remaining works including detailed study of query processing, implementation of all operators, and optimization rules are left to future work.

7. CONCLUSIONS AND FUTURE WORK

Moving objects are ubiquitous in our life and a promising concept to adequately model and represent the changing space-time behaviour of geometric objects in spatio-temporal applications. In this paper, we propose a new two-layered data model MONET. The lower layer is a data model for spatial networks. This data model is the prerequisite for the upper model that represents moving objects in these networks. This layered model fits well to formulate relationships between moving objects and a network in queries. A query language, called MONET QL, allows a clear description of and access to moving objects in spatial networks and to provides high-level operations on them.

Future work includes a generic two-layered model design for all kinds of spatial networks, more formal descriptions and generality features will be discussed. In addition, we will also address implementation issues such as implementations of data types, design of algorithms for the operations and query processing and even optimization strategies in later work. As our model is database-friendly, it is extremely easy to register a new operator as all needs. Furthermore, extensions for, such as traffic jam discovery and prediction are an interesting research issue. The question is how to precisely represent traffic jam and what parameters, outputs such operators should hold. These are all need to be investigated.

8. REFERENCES

[1] S. Scheider and W. Kuhn. Road networks and their incomplete representation by network data models. In *5th Int. Conf. on Geographic Information Science*, pages 290–307, 2008.

[2] R. Güting, T. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *The VLDB Journal*, 15(2):165–190, 2006.

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

[4] Real datasets for spatial databases: Road networks and points of interest (http://www.cs.fsu.edu/ lifeifei/spatialdataset.htm).

[5] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.

[6] R. Güting, M.Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. In *ACM Trans. on Database Systems*, volume 25, pages 1–42, 2000.

[7] A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Int. Conf. on Data Engineering*, pages 422–432, 1997.

[8] M. Schneider. *Moving Objects in Databases and Gis: State-of-the-Art and Open Problems*, pages 169–188. Springer-Verlag, 2009.

[9] K. S. Hornsby and K. King. Modeling motion relations for moving objects on road networks. *GeoInformatica*, 12:477–495, 2008.

[10] Fuyu Liu, Kien A. Hua, and Kien A. Hua. Moving query monitoring in spatial network environments. In *MONET*, pages 234–254, 2012.

[11] M. Vazirgiannis and O. Wolfson. A spatiotemporal model and language for moving objects on road networks. In *Int. Symp. on Spatial and Temporal Databases*, volume 25, pages 20–35, 2001.

[12] M. Erwig, R. Güting, M. Schneider, and M. Vazirgiannis. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296, 1999.

[13] M. Erwig and R. Güting. Explicit graphs in a functional model for spatial databases. 6(5), 1994.

[14] L. Speicys, C.S. Jensen, and A. Kligys. Computational data modeling for network-constrained moving objects. In *Proc. of the 11th ACM Symp. on Advances in Geographic Information Systems*, 2003.