DOI: 10.1111/tgis.12220

RESEARCH ARTICLE

WILEY Transactions in GIS

An intersection-based trajectory-region movement study

Longgang Xiang[1] | Tao Wu[2] | Dick Ettema[3]

[1] The State Key Lab-LIESMARS, Wuhan University, Wuhan, China

[2] School of Geosciences and Info-Physics, Central South University, Changsha, China

[3] Faculty of Geosciences, Utrecht University, Utrecht, the Netherlands

Correspondence
Tao Wu, School of Geosciences and Info-Physics, Central South University, Changsha, China.
Email: taotaomails@gmail.com

Abstract

In order to better understand the movement of an object with respect to a region, we propose a formal model of the evolving spatial relationships that transition between local topologies with respect to a trajectory and a region as well as develop a querying mechanism to analyze movement patterns. We summarize 12 types of local topologies built on trajectory-region intersections, and derive their transition graph; then we capture and model evolving local topologies with two types of trajectory-region strings, a movement string and a stop-move string. The stop-move string encodes the stop information further during a trajectory than the movement string. Such a string-format expression of trajectory-region movement, although conceptually simple, carries unprecedented information for effectively interpreting how trajectories move with respect to regions. We also design the corresponding Finite State Automations for a movement string as well as a stop-move string, which are used not only to recognize the language of trajectory-region strings, but also to deal effectively with trajectory-region pattern queries. When annotated with the time information of stops and intersections, a trajectory-region movement snapshot and its evolution during a time interval can be inferred, and even the relationships among trajectories with respect to the same region can be explored.

KEYWORDS
intersection, movement string, stop-move string, trajectory-region movement, trajectory-region pattern

1 | INTRODUCTION

As location positioning and wireless communication technologies develop, especially approaches related to GPS, it is now common to equip GPS chipsets into various kinds of moving objects (e.g. persons, vehicles and animals). Consequently, an immense amount of positional data has been collected for various fields, including urban planning, traffic management and wildlife behavior monitoring, to name but a few. Generally speaking, such positional data is captured and maintained in the form of trajectories recording the evolving positions of moving objects in space during a certain time interval.

Since trajectories take place in geographical space, it is very natural for researchers to associate a trajectory with contextual geographical elements (Yan, Chakraborty, Parent, Spaccapietra, & Aberer, 2012), like starting from a hotel, going through a road, or crossing a park. Such associations will enrich trajectory data greatly, and therefore benefit further analysis. Usually, geographical elements in the real world, like buildings, roads and districts, can be any spatial type, but for simplicity, they are abstracted as three fundamental objects: points, lines and regions. As regions (e.g. parks, districts, no-fly zones, etc.) are general and universal elements in geographical space, this article focuses hereinafter on the movements of individual objects with respect to a single region, i.e. trajectory-region movement, aiming at describing and analyzing dynamic spatiotemporal relationships as well as semantics between trajectories and regions.

The work of this article is motivated by the research activities of two diverse fields: one is the querying/analyzing of trajectory movement patterns, and the other is topological relationships between geographical elements. The former (e.g. Giannotti, Nanni, Pineli, & Pedreschi, 2007; Vieira, Bakalov, & Tsotras, 2010) focuses on computation methods, usually adopting a simple model of spatial relationships, while the latter (e.g. Egenhofer & Herring, 1991; Egenhofer & Franzosa, 1991; Kurata & Egenhofer, 2007) emphasizes the exploration and application of spatial topologies, rarely considering the problem of storage and querying. In this article, we try to combine the strength of the two fields mentioned above: (1) first by identifying the qualitative features of trajectory-region movement and modeling them as topological invariants; and (2) then expressing qualitative trajectory-region movements with string-format structures, based on which patterns concerning the behaviors of moving objects with respect to a single region are queried and analyzed.

To summaries, the contributions of this article are threefold. First, we classified 12 types of local trajectory-region topologies by means of the 6-intersection model, and proposed two types of trajectory-region strings, i.e. the movement string and stop-move string, to model sophisticated trajectory-region movements. Second, we designed two Finite State Automations (FSAs) to recognize complex strings arising from possible transitions among local topologies as well as Stop/Move behaviors. Third, we reported on studies with trajectory-region movement analyses based on the FSAs mentioned above, which suggest that the proposed approach is: (1) capable of interpreting and querying the behaviors behind a trajectory-region movement; and (2) capable of analyzing the relationships between multiple trajectories with respect to the same region. Additionally, we also presented four typical queries based on the proposed model, which are then evaluated on analog data through an effective FSA-based inner structure.

The remainder of this article is organized as follows. Section 2 reviews related work and Section 3 introduces basic concepts about trajectories. Local topologies of trajectory-region intersections and their transition graph are identified and discussed in Section 4. In Section 5, two types of string-format expressions (trajectory-region strings) are presented to encode trajectory-region movement and the corresponding recognition FSAs are designed to recognize the language of the kinds of strings, based on the derived local topologies and their possible transitions. To facilitate the study of trajectory-region movement encoded as strings, trajectory-region patterns are designed by following the Backus-Naur Form in Section 6, which allows users to raise queries against the database of trajectory-region strings. After that, experiments are launched to check the feasibility of proposed models and methods on analog data in Section 7. Finally, Section 8 concludes this article and discusses some future problems.

2 | RELATED WORK

This article tries to develop a topology-oriented model for qualitatively describing trajectory-region movement and to carry out query/analysis of movement patterns based on the proposed model. Among various research conducted on trajectory data, more related work mainly comes from the following three aspects: (1) trajectory data modeling (e.g. Güting et al., 2000; Spaccapietra et al., 2008); (2) describing topological relationships (e.g. Egenhofer & Franzosa, 1991; Kurata & Egenhofer, 2007); and (3) spatio-temporal querying (e.g. Hadjieleftheriou, Kollios, Bakalov, & Tsotras, 2005; Vieira et al., 2010).

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

(a)

$$M^{+}(L,R)=\begin{bmatrix} L^{o}\cap R^{o} & L^{o}\cap\partial R & L^{o}\cap R^{-} \\ \partial_{s}L\cap R^{o} & \partial_{s}L\cap\partial R & \partial_{s}L\cap R^{-} \\ \partial_{e}L\cap R^{o} & \partial_{e}L\cap\partial R & \partial_{e}L\cap R^{-} \\ L^{-}\cap R^{o} & L^{-}\cap\partial R & L^{-}\cap R^{-} \end{bmatrix}$$

(b)

FIGURE 1 9-intersecion and 9^{+}-intersection models: (a) 9-intersecion model; and (b) 9^{+}-intersection model. (R^{o}–interior, ∂ R–boundary, R^{-}–exterior, ∂_{s}L–directed line's starting point, ∂_{e}L-directed line's ending point)

2.1 | Trajectory data modeling of trajectories

This article models trajectories from the aspect of topology, while a lot of work that address the problem of trajectory modeling, particularly in the field of database, launches studies from the aspect of geometry. The latter approaches, of course share certain research ideas with the former studies, can be divided into two distinctive categories: data centered and semantics oriented. Data centered models of trajectories (e.g. Güting et al., 2000; Wolfson, Xu, Chamberlain, & Jiang, 1998) extended existing data models to cope with 2D geometric data, representing a trajectory as a kind of moving shape, for example points, lines or regions. Following this method, several trajectory database management systems have been developed, such as Secondo (Güting, 2005) and Hermes (Pelekis, Theodoridis, Vosinakis, & Panayiotopoulos, 2006). However, semantics oriented models (e.g. Giannotti et al., 2007; Spaccapietra et al., 2008) regard trajectory streams as sequences consisting of interleaved semantic objects instead of focusing on geometric features, such as stops and moves in the Stop/Move model (Yan et al., 2012), with which various context information can be annotated. In the Stop/Move model, stop, in contradiction to move, imply no movement at all or a slow speed within a small area, which can be extracted by applying various criteria (e.g. Alvares et al., 2007; Buchin, Driemel, Kreveld, & Sacristan, 2011).

The methods mentioned above still cannot give answers to questions about topological relationships among trajectories of moving objects and other geographical elements during their trips although to a certain degree they integrate trajectory data with the relevant geographic information as well as with user-specified spatial and non-spatial features by combining different objects. As a consequence, one goal of this article is to develop a comprehensive approach for describing evolving local topologies between trajectories and geographic data, which is restricted to spatial regions, quite a broad class of spatial elements. Moreover, the developed approach should integrate the information of stops, for it is an important semantic unit during a trajectory.

2.2 | Topological relationships describing of trajectories

Since the projection of a trajectory in R^{2} is a line, approaches for modeling the movements of trajectories proposed in this article draw lessons from topological relationship studies (e.g. Munkres, 1966; Spanier, 1966). A variety of topological relationships (Egenhofer, 1989) between two geographical elements (i.e. points, lines and regions) can be distinguished by methods based on the point-set topology (Alexandroff, 1961) like the 4-intersection model (Egenhofer & Franzosa, 1991), the 9-intersection model (Egenhofer & Herring, 1991) and their extensive models. From all of these approaches, it is the 9-intersection, widely used and studied for modeling topological relations, which creates a matrix characterized by 3×3 types of intersections in terms of topological parts (interior, boundary and exterior) between two geographical elements, as shown in Figure 1a. Nineteen topological relationships can be distinguished for a line and a region (Egenhofer & Herring, 1991) and 43 relationships for a complex line and a complex region (Schneider & Behr, 2006) from 512 totally uncertain permutations of the 9-intersection matrix. The conceptual neighborhood graph for 19 line-region relationships is discussed by Egenhofer and Mark (1995).

By further differentiating the two end-points of a directed line, 9^{+}-intersection model is introduced as an extension of the 9-intersection model (Kurata & Egenhofer, 2007). It is, therefore, a 3×4 matrix (shown in Figure 1b), from which 26 relationships can be captured for a directed line and a region. However, owing to lack of appropriate representations of temporal and spatial features, the 9-intersection model, 9^{+}-intersection model and their extensions, as spatial configuration tools, cannot give the path to evolve spatial relationships between a trajectory and a region as time goes despite their performance on static topological description.

It should be pointed out that this article can to a certain degree be seen as a continuation and extension of Kurata's work (Kurata et al., 2007), in which the authors not only identified possible topologies and their conceptual neighborhood graph, but also indicated that one topology can be assigned to more than one spatial configuration, represented by a triple structure. As far as the model is concerned, ours is event-oriented, capturing events of intersections as well as stops, while Kurata's is topology-oriented, purely tracking the changes of topological locations. Accordingly, our model is more concise and semantic-carried. Taking the behavior of region-crossing as an example, the notation of Kurata's model is "EBIBE" while it is "eie" in our model. One can see that the former records four topological location changes, but our expression captures two consecutive intersection events, (i.e. "ei" and "ie", two of our 12 local topologies about intersection events). Moreover, we go one step further to carry out pattern analysis of trajectory-region movement. For example, we can retrieve trajectories that go along the border of a region at least twice. To the best our knowledge, such queries are seldom studied in this category.

2.3 | Querying of trajectories

A typical spatiotemporal query of trajectories demands information against spatiotemporal relationships between trajectories and spatial elements, i.e. points, lines and regions. For instance, range searching and nearest neighbor querying can be used to explore the relationships between a trajectory and a region for a conventional application, like retrieving all trajectories that stayed in Wuhan University at 10:00 a.m. To evaluate such queries, the majority of previous studies approximated trajectories using the concept of Minimum Bounding Regions (MBRs) (e.g. Pfoser, Jensen, & Theodoridis, 2000; Tao & Papadias, 2001), which were then indexed by hierarchical spatiotemporal indexing structures, like R-trees (Guttman, 1984), STR-trees (Pfoser et al., 2000) or MVR-trees (Hadjieleftheriou, Kollios, Tsotras, & Gunopulos, 2002).

In recent years, researchers have increasingly focused on pattern query, i.e. identifying all trajectories that follow user-defined visiting patterns in both space and time. Hadjieleftheriou et al. (2005) proposed specialized index structures and evaluation algorithms against pattern queries, and therefore achieved many more efficiencies than traditional methods. Mouza and Rigaux (2005) managed to employ pattern queries over a region-based representation of underlying space and introduced a query language to express pattern queries of movements. This work was further refined by Vieira et al. (2010), in which a flexible framework for processing pattern queries was proposed.

Among the above studies, it is the string-format expressions of the last two works that enlightened us and motivated us to design our own string model for describing the movement of a trajectory in a region-partitioned space. However, they paid more attention on the efficient computation of satisfied trajectories and the underlying model of spatial relationships they adopted is very simple and rough, ignoring the sophisticated relationships that existed between trajectories and regions. To be specific, only simple relationships, like "enter" and "leave", were considered in their work, which failed to describe the complex movement behavior of trajectories with respect to a region.

3 | BASIC CONCEPTS ABOUT TRAJECTORY

Before introducing the approaches we explore to reach the goals above, this section provides basic concepts from which we start our study. The trajectory is a spatial-temporal concept, including two different folds: space and time (Spaccapietra et al., 2008). From the perspective of space, the evolving position of traveling objects in R^2 can be a criterion to distinguish themselves from non-moving elements (e.g. buildings, mountains). And from the perspective of time, traveling from one place to another takes a finite amount of time, i.e. trajectories are confined by time intervals. Thus, the two angles are bridged by a mapping function from time to space. Here, we come to the formal definitions about trajectories, which are given as below:

DEFINITION 3.1. (Trajectory). A trajectory is the image of a continuous mapping g: $[t_s, t_e]$ -> R^2, where t_s is the starting time instant and t_e is the ending time instant. Accordingly, $g(t_s)$ denotes the starting point and $g(t_e)$ denotes the ending point.

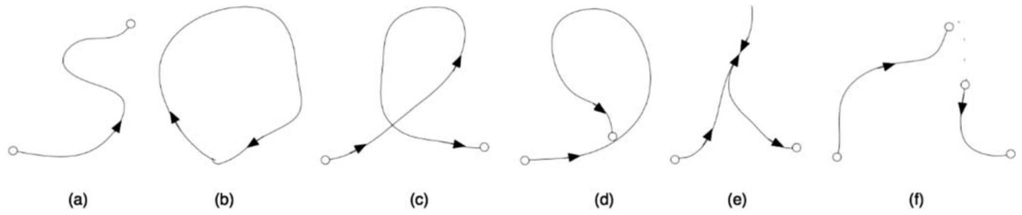


FIGURE 2 Examples of: (a) a simple trajectory line; (b) a closed trajectory line; (c) a self-intersected trajectory line; (d) a self-touched trajectory line; (e) a multi-branched trajectory line; and (f) two trajectory lines

The projection of a trajectory in R^2 (i.e. a line) is called a trajectory line throughout this article. However, trajectory lines cannot be categorized as simple lines (Figure 2a), because they may be closed (Figure 2b), self-intersected (Figure 2c), self-touched (Figure 2d) or even multi-branched (Figure 2e). Note that the upper part in Figure 2e is traversed twice: going upward and then doubling back. As a trajectory is a continuous movement of some object within a connected time interval, its trajectory line cannot have multiple disconnected components. The two disconnected lines in Figure 2f are considered as two trajectory lines, although belonging to the same trip.

It is well known that a moving object does not always change its position during a trip because of stopping at some places occasionally for certain activities, e.g. refueling at gas stations, waiting for traffic lights, etc. When a stop occurs, all time values within the stopping period will be mapped to a same position, i.e. the stop place. Therefore, a trajectory stop is defined as below:

DEFINITION 3.2. (Trajectory stop). Given a trajectory *T* with mapping *g*: $[t_s, t_e]$ -> R^2, a stop is a continuous subset of *T* with time interval $[t_p, t_q]$, such that 1) $t_s<t_p<t_q<t_e$, 2) $t_i, t_j \in [t_p, t_q], t_i < t_j$, $distance(g(t_i), g(t_j)) < d_\varepsilon$, and 3) $interval(t_q, t_p) > t_\varepsilon$. Here, d_ε is the threshold value of ceiling stay distance, while t_ε is the threshold value of the floor residence time.

As a reasonable stop, it should meet the lower bound of a minimum residence time, which accounts for the third condition included in the definition above. A stop, however, can take place at any meaningful location, including the starting and ending points. When a stop ends, the object will resume its movement until it stops at another place, or reaches the ending point. Consequently, a trajectory is composed of a series of interleaved stops and moves. This is what we called the stop/move model of trajectory, which was firstly introduced by Spaccapietra et al. (2008).

Figure 3 illustrates a typical trajectory of a salesman in one day from two aspects: the raw data model and the stop/move model. Obviously, it is very difficult for normal users to understand the behavior of a trajectory with the raw data model. However a trajectory can carry rich contextual information within the stop/move model, which makes its semantics apparent to users. Compared with the raw data model, the stop/move model abstracts a trajectory as semantic stop/move objects, enabling more powerful analyses on trajectory data.

4 | TRAJECTORY-REGION LOCAL TOPOLOGIES

Intersections between a trajectory and a region, i.e. trajectory-region intersections, characterize the movements of the trajectory with respect to the region and therefore provide significant spatial information for relevant research within

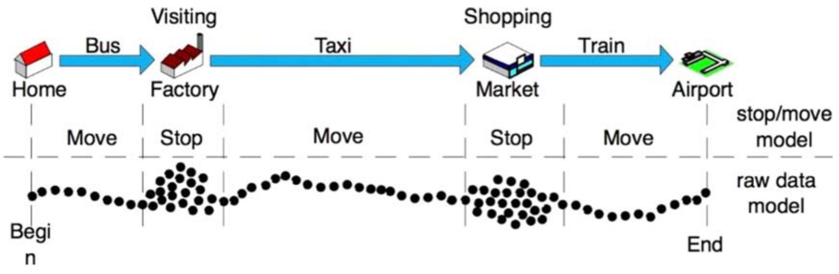


FIGURE 3 An example of stop/move trajectory

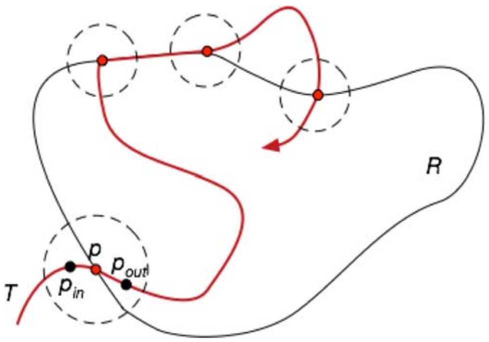


FIGURE 4 Trajectory-region intersections during a trip. The dotted circles denote the infinitesimal neighbourhood of different points

the realm of GIS. From this point of view, a trajectory-region movement can be captured and represented as a sequence of trajectory-region intersections.

4.1 | Local topology types of trajectory-region intersection

The region, in this article, is referred to as a single connected two-dimensional component in R^2, which means it does not have two or more disconnected interiors, but may have holes. A region consists of three topological parts in R^2, the *Boundary*, *Interior* and *Exterior*, which are pairwise disjoint and jointly exhaustive, i.e. $Boundary \cap Exterior = \emptyset$, $Boundary \cap Interior = \emptyset$, $Exterior \cap Interior = \emptyset$, and $Boundary \cup Exterior \cup Interior = R^2$. The *Boundary* of a region is composed of a closed line or a union of multiple disjoint closed lines; the *Interior* of a region is bounded by its boundary; the *Exterior* of a region is the complement of its *Boundary* and *Interior* with respect to R^2.

During a trip, a trajectory-region intersection occurs each time the object enters, leaves, touches, or crosses the boundary of a region. For instance, in the scenario that an object entered the boundary of a region, then moved along it and finally left the boundary, two trajectory-region intersections will be triggered. In other words, a trajectory-region intersection is always 0-dimensional, i.e. a point event. We define a trajectory-region intersection as follows:

DEFINITION 4.1. (Trajectory-region intersection). Given a trajectory T with mapping g: $[t_s, t_e]$ -> R^2 and a region R. Let $p=g(t_p)$ be a point of T that falls into the boundary of R and N be an infinitesimal neighborhood around p. Assume $p_{in}=g(t_{in})$ and $p_{out}=g(t_{out})$, $t_s<t_{in}<t_p<t_{out}<t_e$, are two points of T that fall into N. A trajectory-region intersection takes place around p if neither p_{in} and p_{out} simultaneously falls into the boundary of R.

It should be pointed out that trajectory-region intersections may occur at the starting (or ending) point of T, in which case, p_{in} (or p_{out}) will be empty while p_{out} (or p_{in}) falls into the boundary of R. One can see that a trajectory-region intersection is a topological invariant, which is termed as local topology throughout this article for it is defined locally, i.e. confined to an infinitesimal neighbourhood. Figure 4 shows four trajectory-region intersections occurring on the duration of the trajectory's trip.

The characteristics of local topologies are based upon the comparison of three topological parts of R and two points: p_{in} and p_{out}. Let *Ip(T, R)* be a local topology at a common point p between T and R. Then it is expressed in the matrix form, consisting of p_{in} and p_{out}, as well as topological parts of region R, called the 6-intersection, which is concisely represented as a 2×3-matrix:

$$I_p(T,R) = \begin{bmatrix} P_{in} \cap Boundary & P_{in} \cap Interior & P_{in} \cap Exterior \\ P_{out} \cap Boundary & P_{out} \cap Interior & P_{in} \cap Exterior \end{bmatrix} \quad (1)$$

For the 6-intersection, each set describes a unique local topology, and intersections with the same specifications will be considered as topological equivalence. Each element of the 6-intersection is characterized by a value empty (Ø) or non-empty (¬ Ø), enabling the matrix to describe $2^6 = 64$ patterns. Note that, if p_{in} (or p_{out}) is empty, the corresponding elements of course are characterized by a value empty (Ø). Among these 64 patterns, however, only 12 patterns make sense for the trajectory-region scenario in R^2. For simplification, an iconic representation is introduced to

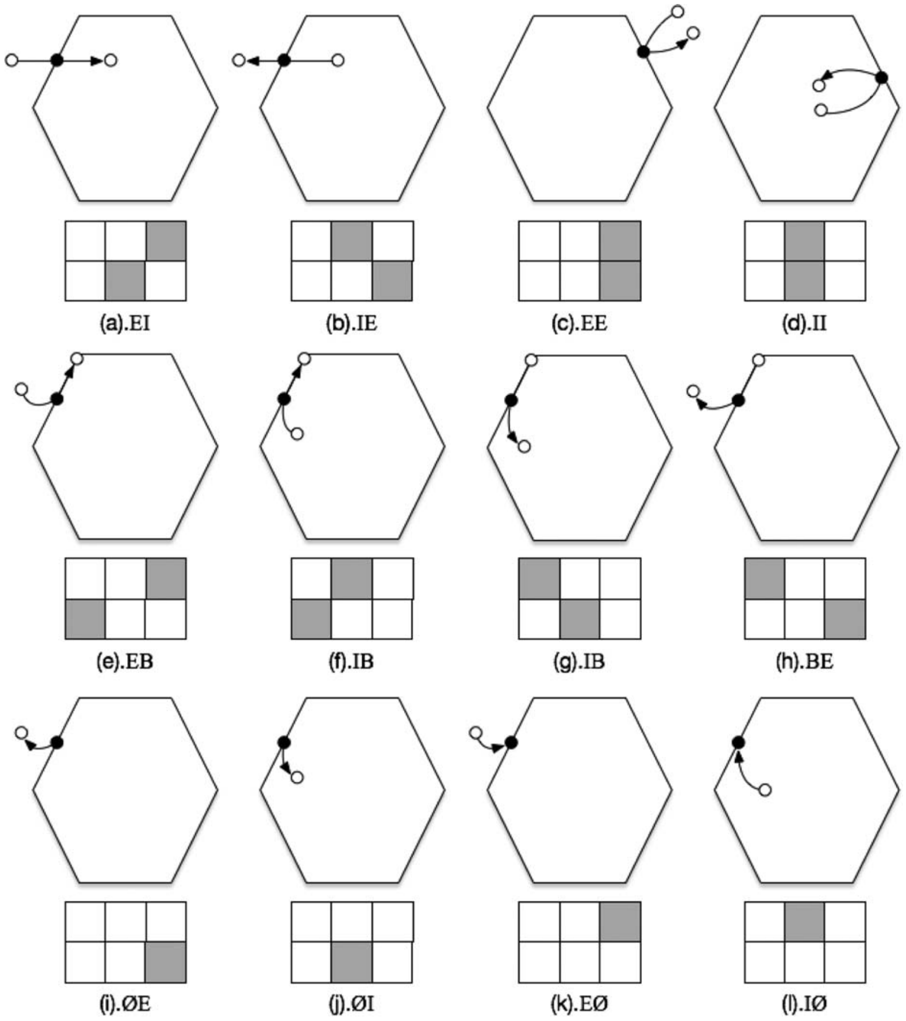


FIGURE 5 Twelve types of local topologies about local trajectory-region intersections. The thick points denote intersection points of different types

the 6-intersection matrices (Figure 5). In our iconic representation, each icon has 2 × 3 cells corresponding to the matrix's 2 × 3 elements. And each cell is marked out if the corresponding element is non-empty ($\neg$ Ø).

To depict the local topological detail of a trajectory-region intersection conveniently, this article also introduces an alternative notation by characters, which records the topological parts marked in the 6-intersection. I, B, E represent the topological parts of the region *R* (*Boundary*, *Interior* and *Exterior*), respectively. If any topological part is characterized by a value non-empty ($\neg$ Ø) in the 6-intersection, it is labeled as *B*, *E* or *I* accordingly. Otherwise, it is labeled as Ø. The type of a local topology is marked as a label concatenation of format *XY*, where *X* is the label of the topological part marked by the p_{in} and *Y* is the one marked by the p_{out}. To be noted, besides topological details, local topologies also hint at the basic semantics of corresponding trajectory-region intersections. For instance, the local topology marked as "EI", as seen in Figure 5a, denotes the p_{in} falling into the exterior as well as the p_{out} falling into the interior, which means the small fragment of the trajectory *T* within the neighborhood of the point *p* goes from the exterior to the interior.

4.2 | Transition graph of local topologies

We schematize the transition of the 12 types of local topologies by designing a Freksa point-polygon graph (Freksa, 1992). Each local topology is represented by a node in the graph. And an edge is directed from one node to another if

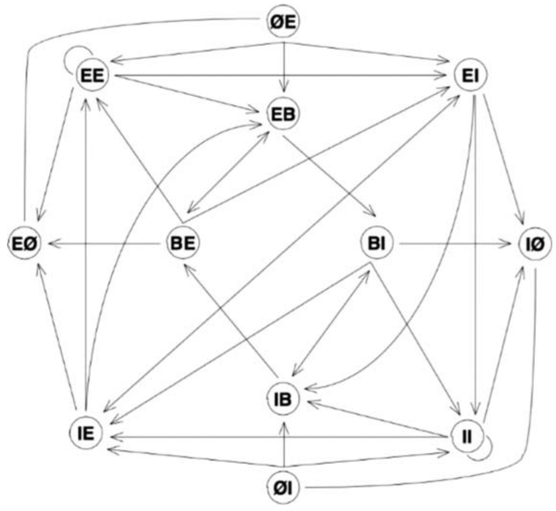


FIGURE 6 The transition graph for 12 local topologies

the local topology represented by the latter node might be derived from the former one following the motion of trajectory with respect to the region. For example, the node of *EI* can be directed from the node of *IE*, because an *EI* intersection will be triggered if the object continues to move to a cross boundary after an *IE* intersection. In turn, *IE* can also be directed from *EI*. However, this does not mean that the pointing of the edges is symmetric. For example, *EI* can be directed from *EE* but cannot direct to *EE*, because after an *EI* intersection, the object is located in the interior of a region and only those local topology types that start with 'I' (i.e. *II*, *IE*, *IB* and *IØ*) can be directly triggered.

We identified 36 transition relations among the 12 local topologies, from which a transition graph can be built (Figure 6). Obviously, this graph schematises trajectory-region intersections based on their prospective relevance over temporal logic. In this graph, three pairs of nodes, i.e. *EI-IE*, *EB-BE* and *BI-IB*, are symmetric, and two nodes, i.e. *EE* and *II*, are reflexive because they can be directed from themselves. *EØ* and *IØ*, as occur at the ending point, can direct to nothing; *EB* and *IB*, enter into the boundary and each have two nodes as possible next states; the remaining eight nodes each have four nodes to be possible nest states. Note that in the transition graph, any non-starting node (i.e. all nodes except *EØ* and *IØ*) can be reached from any non-ending node (i.e. all nodes except *ØE* and *ØI*) through a path composed of at most two edges.

5 | INTERSECTION-BASED MOVEMENT MODELING WITH RESPECT TO A REGION

Based on 12 types of local topologies and their transition graph, this section develops the concept of trajectory-region strings to qualitatively describe the movement of an object in terms of a region. It is realized as two types: a movement string and a stop-move string. The former encodes intersections in the chronological order while the latter further adds the information on the stops.

5.1 | Movement string

Each path, in the transition graph, indicates a trajectory-region movement, while different paths imply distinctive movement of behaviors. This promotes our focus of depicting trajectory-region intersection types to model trajectory-region movements. It is presented as an intersection-based string, called movement string, the formal definition of which is given as below:

DEFINITION 5.1. (Movement string). Given a trajectory *T* and a region *R*, the movement of *T* with respect to *R* is given through a string in forms of $y_s x_1 x_2 ... x_n y_e$, $n=0$ or $n \geq 2$, such that:

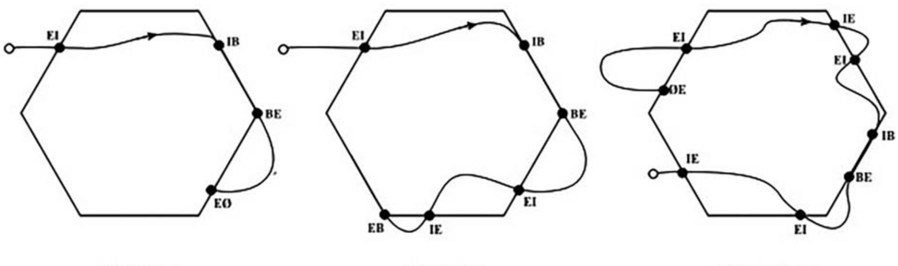


FIGURE 7 Three trajectory-region scenes and their movement strings. White dots denote end-points, black dots denotes intersections, and arrows denote movement directions [Correction added on 23 September 2016, after first online publication: Figure 7 image has been amended to match the caption]

- y_s, $y_s \in \{e, b, i\}$, and
- $x_1 = Ø$, *if* $y_s = b$, while $x_1 = y_s$ for others, and
- $x_n = Ø$, *if* $y_e = b$, while $x_n = y_e$ for others, and
- $x_i x_{i+1}$, $1 \le i < n$, is a local topology.

A movement string (denoted by *MS*) without y_s and y_e is called an intersection string (denoted by *IS*), which only states the information of trajectory-region local topology presented by two successive characters. In the simplest situation, it holds that n = 0 (non-intersection) and the topological movement between *T* and *R* is just described by two characters indicating topological parts: starting-from and ending-at. Though the permutations for a 2-character movement string are $3^2 = 9$, just three of them make sense, i.e. *ee* (staying outside), *bb* (moving along boundary) and *ii* (staying inside). For cases with intersections, the minimum length for its movement string will be not shorter than four. For example, a trajectory-region movement that starts from exterior, then crosses the boundary, and ends at interior will result in a movement string of *eeii*.

Figure 7 illustrates three trajectory-region scenes and their movement stings. Consider Figure 7a, it started from the exterior, then triggered four intersections, i.e. *EI*, *IB*, *BE* and *EØ*, and finally ended at the boundary, so its movement string is *eeibeØb* (accordingly, the intersection string is *eibeØ*). In turn, one can obtain rich information from a movement string without having to check the trajectory-region movement on a map, which will be discussed in detail in Section 6.

Figure 8 presents a non-deterministic finite state automation (NFA) that recognizes the language of movement strings, i.e. all reasonable character combinations over the alphabet {*e*, *b*, *i*, *Ø*}. It has one starting node (S_0), eight finishing nodes (F_1-F_8), and each edge represents a transition. Given a movement string, there exists a unique path from the starting node to one finishing node, and vice versa. The path always starts with the node S_0 (which means an empty

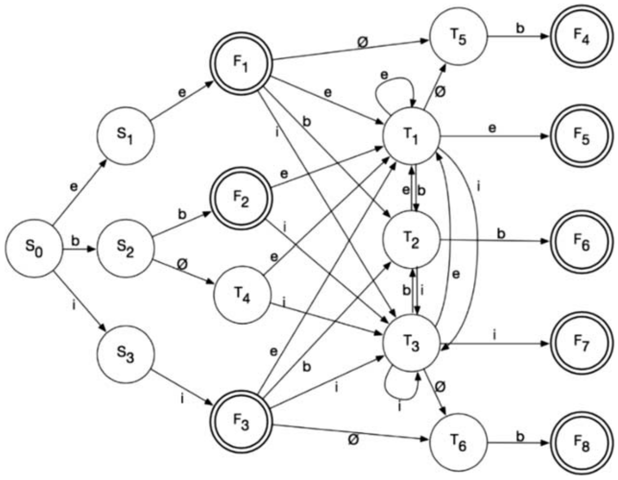


FIGURE 8 The NFA for the language of movement strings

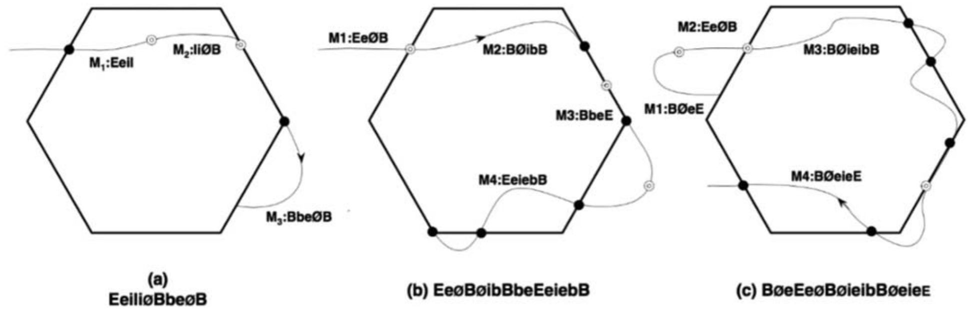


FIGURE 9 Three trajectory-region scenes and their stop-move strings. Double-circle points denotes inner-stops

state), and then transforms from one node to another in accord with the ordered sequence of characters within a movement string. For example, the movement string of Figure 7a is "*eeibeØb*" whose corresponding path is "$S_0 \gg S_1 \gg T_1 \gg T_7 \gg T_6 \gg T_5 \gg T_8 \gg F_7$".

It is worth pointing out that nodes from S_1 to S_3 can only reside in the second position of a path presenting the starting position, and nodes F_1 to F_3 indicate that the movement string might end up with its second character. Other nodes appear after the third place of a path through the NFA. For instance, when the character 'e' is inputted after the second character, the recognition may jump to node T_1 from other nodes to accept an intersection of type *EE*, *IE* or *BE*.

5.2 | Stop-move string

The moving object may occasionally stop at some place for a certain time to carry out some activities during its trajectory, so it is very necessary to extend the movement string through further integrating the information of stops, which can be accomplished by encoding the stop locations with regard to a region and inserting them into the movement string. Such an extended movement string is what we called a stop-move string, see below:

DEFINITION 5.2. (Stop-move string). Given a trajectory *T* and a region *R*, the comprehensive information of *T*'s movement with respect to *R* is given through a string in the form of $s_1 m_1 s_2 ... m_n s_{n+1}$, such that:

- $s_i \in \{E, B, I\}$, and
- m_i, in form of $x_1 x_2 ... x_k$, is the intersection string between s_i and s_{i+1}, and
- $x_1 \in \{b, \emptyset\}$, *if* $s_i = B$, while x_1 equals the lowercase of s_i for others, and
- $x_k \in \{b, \emptyset\}$, *if* $s_{i+1} = B$, while x_k equals the lowercase of s_{i+1} for others.

According to this definition, each capitalized letter in a stop-move string (*SS*) indicates a stop event and the substring between two consecutive stops denotes the intersection string of the corresponding movement. The two end-points (corresponding to the timestamps *ts* and *te*, respectively) of *T*'s trajectory line are considered as two special stops. And s_i denotes the i^{th} stop event during *T* and is assigned an uppercase letter that labels the topological part where the stop is located with respect to *R*; m_i denotes the i^{th} movement separated by stops and is formatted as the intersection string of the i^{th} movement.

It is necessary to note that a stop-move string can have a length of any size not less than two. That is, a length of 2 means that the object neither stopped nor triggered intersections during its trajectory, while a length of 3 means it did not trigger intersections but stopped for one time (i.e. *EEE*, *III* and *BBB*). Besides the two end-stops, a trajectory can have inner-stops of zero or multiple times. For a trajectory with *n* inner-stops, there exist (n+1) stop-separated movements, which is obviously a region-independent property.

Figure 9 illustrates three trajectory-region scenes and their stop-move stings. A stop-move string can be generated by first encoding each stop-separated movement and then concatenating them. As a stop-separated movement does not contain inner-stops, it can be encoded by first generating the movement string and then capitalizing the two end-characters. Take Figure 9b as an example to illustrate how to generate strop-move strings. It has three inner-stops, one

TABLE 1 Six situations for stops with respect to a region

Case No	Three-character encode	Situation
1	*eEe*,*eEE*, *EEE*, *EEe*	stop at exterior, e.g., the last stop in Figure 8b
2	*iIi*, *iII*, *III*, *IIi*	stop at interior, e.g., the first stop in Figure 8a
3	*bBb*, *bBB*, *BBB*, *BBb*	stop at boundary but not at an intersection point, e.g., the second stop in Figure 8b
4	*ØBØ*	stop at an intersection point of type *ei*, *ie*, *ee* or *ii*, e.g., the first stop in Figure 8b
5	*ØBb*	stop at an intersection point of type *ib* or *eb*, e.g., the second stop in Figure 8a
6	*bBØ*	stop at an intersection point of type *bi* or *be*, e.g., the last stop in Figure 8c

in the exterior and two at the boundary, resulting in four stop-separated movements, i.e. *EeØB*, *BØibB*, *BbeE* and *EeiebB*, so the stop-move string is *EeØBØibBbeEeiebB*.

One can obtain from Figure 9 that there exist totally six distinct situations for causing stops with respect to a region, which are summarized in Table 1. According to this table, stop-move strings can be easily transformed into the corresponding movement strings: firstly, convert the first and last letter to lowercase and then for each of three consecutive characters, if it conforms to any case listed in Table 1, keep only one lowercase letter (if it failed to detect lowercase letters, e.g. case 4, delete all three characters). For example, the transformation procedure for Figure 9c is: *BØeEeØBØieibBØeieE* -> *bØeEeØBØieibBØeiee* -> *bØeØBØieibBØeiee* -> *bØeieibBØeiØb* -> *bØeieibeiee*.

Note that cases 1, 2 and 3 in Table 1 each have four sub-cases. For any of the three cases, though four sub-cases share a common concept, they imply different movement semantics, which are illustrated in Figure 10. Take the stop of *EEe* (see the last stop in Figure 10a) as an example, it means that the trajectory has given rise to a stop (occurred in exterior) before this stop and will trigger an intersection (of type *EI* here, but *EE* and *EB* are also possible) after this stop.

By slightly modifying the FSA of movement string presented in Figure 8, an FSA that recognises the language of stop-move string over the alphabet {*e*, *b*, *i*, Ø, *E*, *B*, *I*} can be obtained, which is given in Figure 11. As the stop information is encoded with uppercase letters, the automation for stop-move strings is a deterministic finite state automation (DFA). In addition, the finishing nodes in the automation of stop-move strings act as intermediate nodes.

Take node F_1 as an example: when reaching the end of the input string, the recognition is finished; when inputted with character ‘e’, the recognition jumps to node T_1, which means an intersection of type *EE*, *EB*, *EI* or *EØ* will be triggered; when inputted with character ‘E’, the recognition still stays at node F_1 but has finished the recognition of one exterior stop. For example, the stop-move string of Figure 9b is “*EeØBØibBbeEeiebB*” whose corresponding path in the DFA is “$S_0 \gg S_1 \gg T_1 \gg T_8 \gg F_2 \gg T_3 \gg T_7 \gg T_6 \gg F_2 \gg T_2 \gg T_5 \gg F_1 \gg T_1 \gg T_7 \gg T_5 \gg T_6 \gg F_2$”.

5.3 | Trajectory-region string

As mentioned above, the *MS* (movement string) is a kind of encoding, based on intersections, for depicting the evolution of topological relationships between a trajectory and a region, while the *SS* (stop-move string) carries extra

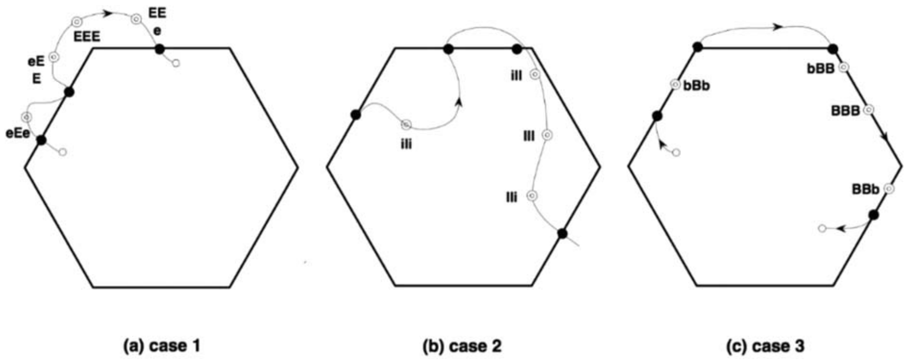


FIGURE 10 Sub-cases for the first three cases in Table 1

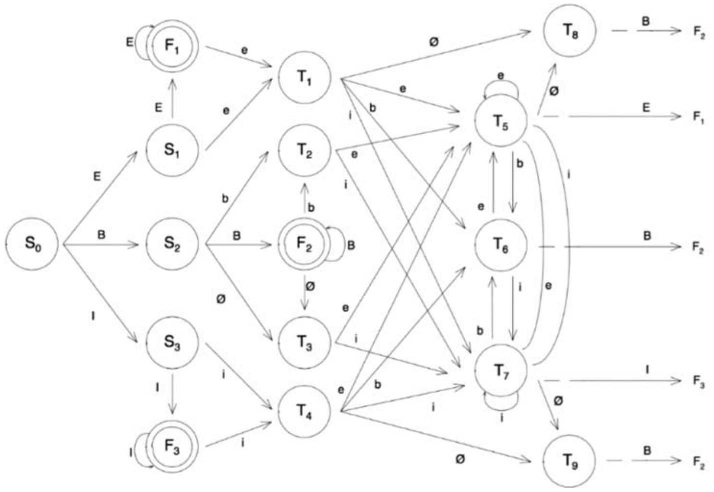


FIGURE 11 The DFA for the language of stop-move strings

information about stop behaviours during the trajectory. In more general cases, we need only recordings of *SS* by default if there is no confusion with specific requirements on semantics information like "stop" and "move". Thus, the above strings are also simply called trajectory-region strings.

Due to the unique structure of trajectory-region strings (including *MS* and *SS*), one can capture detailed information about movements of a trajectory related to a region by just parsing the string without resorting to more complicated techniques.

The intersectional information between a trajectory and a region can be derived by parsing its corresponding *MS* (or a deduced *MS* from a *SS*) under the following three properties:

1. Character Ø only appears at the second or penultimate position of a *MS* (i.e. the head and tail of an *IS*).
2. Intersections are expressed by two adjacent characters of an *IS*.
3. As to information about stops, there need to be further constraints and properties for *SS*.
   (a) Character Ø only appears next to an uppercase letter.
   (b) Stops are expressed by uppercase characters of a *SS* except the head and tail of it.

Besides acquiring information about stops, intersections and segments, we can parse a trajectory-region string and interpret it as stop-included movement behaviour with respect to a region. We can also verify trajectory-region strings under given characteristics of movements, such as starting from the exterior of the region, moving along the boundary of the region, etc.

Figure 12 presents two real GPS trajectories with respect to a campus (marked with light yellow). The red-dotted line shows a teacher's trip (drive to office, stay at office for work, then drive out of the campus), while the yellow-dotted line captures a student's trip (walk to classroom, study all morning, then walk out to take bus). The two trajectories are actually the manually processed results, in which noise was filtered out, points were rectified to their real positions and stops were labelled (marked as green arrows). Then the teacher's trajectory is encoded as "EeeEeilieE" (SS) or "eeeiee" (MS), and the student's as "IiIIlibBØeEE" (SS) or "iibee" (MS). Obviously, any one of the input trajectories should be at least processed through four stages away from encoding trajectory-region strings: noise-filtering, stop-extraction, map-matching and spatio-joining. These processes are very challenging because the trajectories often carry noise and may be sparsely and irregularly sampled, yet they are beyond the scope of this article and will be discussed in our future work.

Plenty of information can be easily found about the movements of the two trajectories related to the campus. The teacher's trip started from and ended at exterior, which consists of two stops (one exterior stop, one interior stop) and

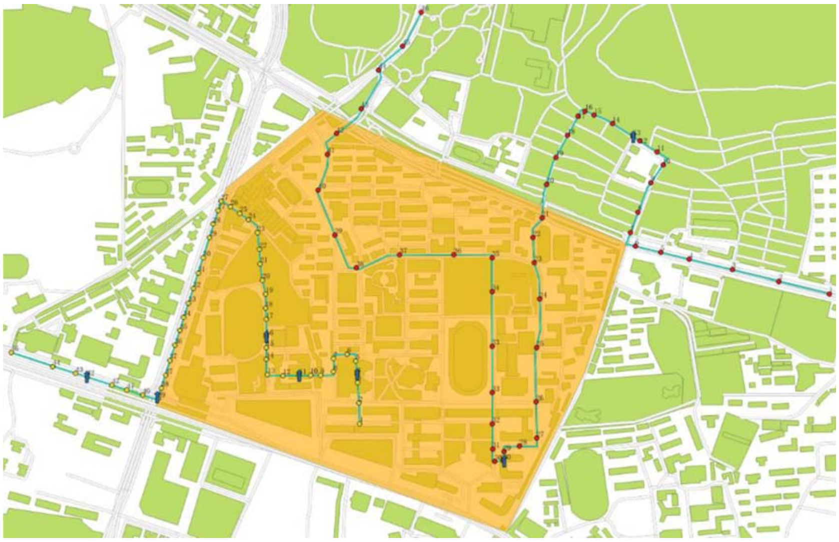


FIGURE 12 Two real trajectories with respect to a campus

three intersections (ee, ei, ie). And the student's trip started from interior and ended at exterior, covering five stops (three interior stops, one boundary stop, one exterior stop) and two intersections (ib, be). The semantics of these stops can be further inferred if annotated with address and time information. For example, the teacher's first stop is to have breakfast in a snack bar while her second stop is to work in an office building.

6 | TRAJECTORY-REGION MOVEMENT ANALYSIS

The goal of our work is not only to provide trajectory-region strings encoding movement of objects, but also a query language capable of analyzing hidden information within such time-dependent spatial data, including those querying and analyzing operations on describing the movement characteristic of moving objects. To achieve a smooth interplay between user operations and languages of trajectory-region strings recognized by the FSAs, a unified expression framework is needed. In order not to be bound to any particular SQL language, we employ standard BNF (Backus-Naur Form) for the framework of these languages with which most readers should be familiar.

For the remainder of this article, attention is restricted to features of trajectory movements expressed by the trajectory-region pattern which is a unified framework interpreting the appropriate spatial and temporal properties of trajectory-region strings by partial groups of characters, as well as advanced analyses by use of these features. All of these are executed in a more flexible way for which a generic representation for patterns based on BNF is proposed, allowing users to express pattern queries against trajectory-region strings. And this prompts us to develop a more effective way to evaluate BNF-expressed trajectory-region patterns based on the constrained FSAs. Without specification, the trajectory-region patterns discussed later are expressed via stop-move strings with appropriate DFA.

6.1 | Trajectory-region pattern

The trajectory-region pattern is a predefined generic framework capturing certain spatial and temporal properties of a moving object with respect to a region, which allows users to get customizable presentations of trajectory-region movements while modifying and adjusting the pattern to the requirements specific to the application at hand. Specifically, the framework provides a facility to retrieve trajectories that match certain user specified trajectory-region patterns delivering features on both space and time.

A trajectory-region pattern which is described in terms of constraints discriminates qualified nodes (nodes are specified and permitted) from disabled notes (nodes are specified but not permitted) in the automation. A trajectory-region string matches with a class of trajectory-region pattern if it satisfies all constraints of the pattern, meaning that all motion states of sub-section of it coincide with the pattern. We adopt a standard expression framework following BNF for trajectory-region patterns, which is then represented as follows:

$$<\mathit{Pattern}> ::= <\mathit{Simple\ Pattern}> \mid <\mathit{Composite\ Pattern}> \tag{2}$$

where the symbol "::="means the "*Pattern*" on the left "can be represented as" the expression of elements on the right, which are separated by the vertical bar, "|", linking two or more alternative elements on the left. Following this presentation, a trajectory-region pattern *Pattern* can either match a single feature of trajectory-region strings, with an *Temporal constraint* optionally, called *Simple Pattern*, or match a sequence with multiple patterns, which might be a *Simple Pattern* or even another a sequence, connected or limited by logical operators, called *Composite Pattern*. Technically, any complex patterns can be expressed as a *Composite Pattern* in the form of a combination of simpler and more comprehensible sub patterns with certain operators.

6.2 | Simple patterns

In the following, we introduce a critical foundation for trajectory-region patterns, the simple pattern. A pattern will be a *Simple Pattern* when it covers only a simple unit of motions, called *Basic Spatial Constraints* (*BSC*), as well as its corresponding temporal information optionally, called *Temporal constraint* (*TC*). The representation of a *Simple Pattern*, consisting of the spatial constraints and time constraints description, is

$$<\mathit{Simple\ Pattern}> ::= <\mathit{Basic\ Spatial\ Constraint}> <\mathit{Temporal\ Constraint}> \tag{3}$$

Here a *BSC* is a constraint on a single state of the automation, or further matches a sequence of states. And the *TC* in square brackets means that it is an optional nonterminal, where the temporal information could either be time instants or intervals.

6.2.1 | Basic spatial constraints

To allow flexible description of trajectory-region movements effectively, *Basic Spatial Constraints* that fixate on simple units of motion with regard to local topological relations of a trajectory and a region should be formalized first. These *Basic Spatial Constraints*, describing incidents during the trip, come along with an automation-based formalism, which includes: (1) constraints at endpoints; (2) constraints during intersections; and (3) constraints at stop/move. FSAs allow basic patterns to be specified by constraints expressed with states (nodes).

Constraints at endpoints, "Starting" and "Finishing", are a pair of special motions of a trajectory, which are used for limiting topological relations between a trajectory and a region at start and end points. "Starting" and "Finishing" are optional, since the head and tail states are not always desired.

- <Starting constraint>:: = "start_from" <*node*>. A starting constraint claims the initial state in which the automation must start its recognition from *node* $\in\{S_1, S_2, S_3\}$.
- <Finishing constraint>:: = "finish_at" <*node*>. A finishing constraint claims the final state in which the automation must finish its recognition at *node* $\in\{F_1, F_2, F_3\}$.

Examples: A simple pattern of trajectory starting from the exterior is formalized as "P = start_from S_1". By contrast, a pattern finishing at the interior is formalized as "P = finish_at F_1".

Since a trajectory can be, in a sense, decomposed into a sequence of stops separating moves, which can be further enriched by attached information like activities and locations, it is essential to identify these two kinds of behaviors as forms of *Basic Spatial Constraints*. "Moving"and"Stopping" are forms of basic constraints that focus on spatial position changes and stay fixed during a trajectory.

- <Moving constraint>:: = "move_about" <*node*>. A moving constraint claims the pass-on state in which the automation must have its recognition at *node* ∈{T_1, T_2, . . ., T_7}.
- <Stopping constraint>:: = "stop_at" <*node*>. A stopping constraint claims the transient-stay state in which the automation must have its recognition passing *node* ∈{F_1, F_2, F_3}.

Examples: A simple pattern of trajectory passing the exterior is formalized as "P = move_about T_1" or "P = move_about T_5". And a pattern stopping at the interior is formalized as "P = stop_at F_1".

Basic Spatial Constraints also cover sequences of constraints, the GoPass constraints, meaning that substrings of the *SS* string are in accord with sets of given combination paths. Thus, these patterns can be used to identify similarity on path between one trajectory-regions string and others, which extends and enhances basic patterns.

- <GoPass constraint>:: = "go_pass" <*path*>. A GoPass constraint claims a successive variation of motion states in which the automation passes its recognition upon a *path*, a sequence in the form of "<*path*>:: = <*node*> {<*alternative set*> <*node*>}", which consists of nodes from {S_1, S_2, S_3, F_1, F_2, F_3, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8} correspondingly. The "*alternative set*" matches a collection of states, <*alternative set*>:: = "("<*node*> {"," <*node*>} ")", which are accepted to appear 0 or more times between two assured states.

Examples: A simple pattern of trajectory which enters the boundary from the exterior and gets into the interior after walking along the boundary for a while is formalized as "P = go_pass $T_1T_6T_7$" or "P = go_pass $T_5T_6T_7$".

6.2.2 | Temporal constraints

Having temporal constraints enables the identification of more specific patterns, and thus makes it possible to capture objects' motions corresponding timing to certain characteristics. Considering properties of time, three kinds of temporal constraints and an operation are defined as follow.

- "Recurring constraint":: = <*BSC*> "recur" [=|>|<] <*n*>. A recurring constraint claims repetitive appearances of a kind of *Basic Spatial Constraints* in accord with repeated substrings of a trajectory-region string. Since the *BSC* has potential duplicates, the recurring constraint makes it possible to express its frequency range with additional parameters: ({=, >, <} *n)*. The additional parameter can take the common relational operators, equal to (=), greater than (>) and less than (<), allowing numeral conditions of repeats to be specified.

Examples: A simple pattern of trajectory stopping at the exterior more than two times is formalized as "P = stop_at F_1 recur > 2".

The constraints "atInstant"and "duringInterval" allows indications of the detailed temporal information of those basic constraints. Although in many cases, research focuses on the relative temporal order of basic patterns, specific time is definitely required to build a composite constraint.

- <AtInstant constraint>:: = <*BSC*> "at_instant" <*instant*>. An atInstant constraint claims an explicit parameter for a time instant of trajectories featuring a set of certain specific behaviors or states of motions, which includes additional parameters: *instant*.
- <DuringInterval constraint>:: = <*BSC*> "during_interval" <*interval*>. A duringInterval constraint claims a time length of motions that provide an insight into an object's behavior across certain time intervals that are in form of (<*instant*$_{\_S}$>, <*instant*$_{\_E}$>).

Examples: A simple pattern of trajectory passing the exterior at a given time *t* is formalized as "P = move_about T_1 at_instant *t*" or "P = move_about T_5 at_instant *t*". And the pattern is formalized as "P = move_about T_1 during_interval (t_s, t_e)" or "P = move_about T_5 during_interval (t_s, t_e)" when it occurs in the interval (t_s, t_e).

An operation for temporal information are "toTime" allowing extractions of the detailed temporal information of *Basic Spatial Constraints* for the reason that temporal constraints of patterns usually appear in forms of specific time information corresponding to certain *BSC*.

- toTime: to_Time (<*BSC*>) => *retime*. A "toTime" operation returns the temporal information of trajectories featuring a specific *Basic Spatial Constraint* mentioned above. The result of the toTime, *retime*, is a sequence of time interval in the form of (<*interval*$_{\_1}$>, <*interval*$_{\_2}$>...<*interval*$_{\_k}$>), where each <*interval*$_{\_i}$> is a pair of instants, i.e. <*interval*$_{\_i}$>:: = "("<*instant*$_{\_i\_S}$> "," <*instant*$_{\_i\_E}$> ")". Note that, if a *BSC* features an instantaneous behaviour or state of motions then the value of <*instant*$_{\_i\_S}$> equals the value of <*instant*$_{\_i\_E}$>.

6.3 | Composite patterns

The trajectory-region patterns presented so far are simple patterns of individual incidents during a trajectory. Difficulties come with the need to express complicated spatiotemporal patterns for practical analyses. *Pattern*, except for matching a *Simple Pattern*, most of the time presents a more complicated structure, multiple patterns connected or limited by logical operators, called *Composite Patterns*. Three logical operators,"NOT", "AND" and "OR", are employed to link two sub patterns within a *Composite Pattern*. A composite pattern is formalised as a sequence of composite patterns or simple patterns that are limited by temporal annotation patterns as often happens linked by appropriate logical operators:

<*Composite Pattern*> ::= ["NOT"] <*Pattern*> { ["AND" | "OR"] <*Pattern*>} (4)

where the curly braces denote that elements within them may appear zero or more times, the logical operator "NOT" is an operation on one *Pattern*, and "AND" and "OR" are both operations on two *Patterns*. To be noted, the element "<*Pattern*>" on the right expression refers to a *Simple Pattern* or a simpler sub *Composite Pattern*. In other words, sub-patterns in this description can be either a *Simple Pattern* or even relatively simple composite patterns.

6.4 | Trajectory-region patterns for query

Trajectory-region patterns make a big difference for query and retrieval applications when exploring trajectory-region movements. It is clear that trajectory-region patterns are closer to human thinking than trajectory-region strings, and strictly follow the principle accepted by the FSAs derived from trajectory-region strings. Thus, trajectory-region strings can be interpreted into any of the reasonable states, and support the specification of complex classes of trajectories patterns within our compact expression structure, which provides a convenient and easy to understand the methodology for query as well as further analysis that requires only basic programming skills.

Here we give several intuitive examples for illustrating queries based on trajectory-region patterns.

Q1. Give all trajectories that started from exterior, stopped at boundary for more than two times and ended at interior with respect to region *R*.

P = start_from S1 AND stop_at F2 recur>2 AND finish_at F3 (5)

If the trajectories of an object *o* match the pattern, then: "start_from S_1" means that *o* starts motion from the state S0 to S1 at the very earliest (starting from the exterior of *R*); "stop_at F_3 recur >2" denotes that *o* experiences the state F3 more than two times before it reaches its final state; "finish_at F_3" means that *o*'s final state is F3 (ends its trip at the interior of *R*).

Q2. Give all trajectories that crossed region *R* for more than two times.

P = go_pass T1(T3,T6,T8,F2)T7(T3,T4,T6,T7,T8,T9,F2,F3)T5 recur>2 (6)

When it comes to GoPass constraints, we can also use the three logical operators to detect or describe those complex motion patterns of trajectories. This example has no constraints on endpoints of trajectories but employs a

GoPass constraint with alternative sets to match any optional state of the automation, in which the trajectory may experience any other state while passing the path "$T_1T_7T_5$". To be noted, queries without any information about "stops", like the Q2, can also be expressed via the movement string and its NFA-automation, leading to a more simple and effective representation. For instance, Q2 can be simplified down to "P = go_pass T1(T6)T7(T6,T7)T5 recur>2".

It is possible to acquire metric details for trajectory-region strings and even carry out spatiotemporal relationship analyses among trajectories with respect to a same region as long as temporal constraints are considered. Let us consider the following example further:

Q3. Give trajectories that entered into region *R* before time t_i and then went out of *R* after t_j.

P = move_about T7 during_interval (ts,ti) AND move_about T5 during_interval (tj,te) NOT move_about T5|T6|T9 during_interval (ti,tj).

If a trajectory satisfies the pattern, it should have passed the state T_7 before t_i and hold states without T_5, T_5 and T_9 during time interval [t_i, t_j].

Specifically, three time constraints should be introduced against the automation of stop-move string, which are constraint "go_pass (T_1,T_2,T_3,T_5,T_6)T_7" that occurs before time t_i and constraint "go_pass (T_5,T_6,T_9)T_5" that does not occur during time interval [t_i, t_j]. As paths "(T_1,T_2,T_3,T_5,T_6)T_7" in Figure 10 correspond to intersections of three types, i.e. *EI*, *BI* and *ØI*, that lead to the entering of the region interior, the corresponding occurrence time can be easily obtained. Therefore for the first time-limited constraint, only if a qualified path (i.e. T_1T_7, T_2T_7, T_3T_7, T_5T_7 or T_6T_7) that occurred before time t_1 is traversed, it is satisfied. The second time-limited constraint can be evaluated in a similar way.

Moreover, one can search for all related trajectories based on spatiotemporal properties of a given trajectory-region movement. A typical example of such analysis is as below:

Q4. Give trajectories that stayed in the interior of region *R* at the time that trajectory *X* finally left *R*.

P1 = go_pass T5 (T1,T5,F1).

Let t = to_time(P1) from *X*.

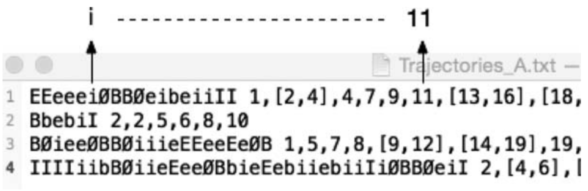
P2 = start_from S3 at_instant t OR finish_at F3 at_instant t OR stop_at F3 at_instant t OR move_about T4 at_instant t OR move_about T7 at_instant t.

If a trajectory satisfies the pattern, it should have an intersection-separated segment that falls into the interior of *R* and whose lifecycle covered the time instant of *X*'s last intersection of type *EE*, *IE*, *BE* or *ØE*.

Obviously, it is more convenient to evaluate Q4 against the automation of movement strings. Firstly, check the movement string of *X*-*R* with constraint "go_pass T5 (T1,T5,F1)", and return the time instant *t* of the last qualified path; then construct a time-limited constraint, i.e. "start_from S3 at_instant t OR finish_at F3 at_instant t OR stop_at F3 at_instant t OR move_about T7 at_instant t"; Finally apply the constraint on the automation to match. Note that within this time constraint, each qualified path actually specifies a state that is covered by the interior of *R*: S3 (starting), F3 (ending and stoping), as well as T4 and T7 (moving).

7 | EXPERIMENTS

For the purpose of validating the proposed model and methods, we execute sets of experiments using sampled analog data in this section, and in the future, we will consider building applications on real trajectory-region movements, in which trajectory-region strings should be first extracted and stored in files or databases. We first present the analog data for the experiments and then introduce the evaluation methodology. Finally, we analyze our test results, followed by discussion. All the experiments are implemented in Java JDK 1.8 on an Intel Core Quad CPU i7 2.30GHz machine with 8 GB of memory running Microsoft Windows 8.1.



```
EEeeeiØBBØeibeiiII 1,[2,4],4,7,9,11,[13,16],[18,
BbebiI 2,2,5,6,8,10
BØieeØBBØiiieEEeeEeØB 1,5,7,8,[9,12],[14,19],19,
IIIIiibBØiieEeeØBbieEebiiebiiIiØBBØeiI 2,[4,6],
```

FIGURE 13 An example of trajectory-region analog dataset

7.1 | Trajectory-region analog dataset

The trajectory-region analog dataset is a set of analog trajectory-region data generated by going through automations randomly. The dataset contains five different levels of aggregations covering from 10,000 to 500,000 records consisting of trajectory-region strings as well as temporal information of activities, respectively. For simplicity and intuition, the timestamps of characters inside the trajectory-region analog dataset are replaced as a set of random integers in ascending order. As shown in Figure 13, the temporal information is annexed to the trajectory-region string in the form of a timestamp sequence associated with corresponding characters (including uppercase and lowercase letters) except "Ø". Each timestamp corresponds to the starting time of a motion state described by a character apart from the one in uppercase. And its duration can be concluded by calculating with the timestamps of the next state and itself. For example, the sixth character of the first record, "i", associates with the timestamp "11", while its next character "Ø" lacks any timestamp.

7.2 | Evaluation methodology

The experiments were conducted to evaluate the model's competencies for querying target trajectories with respect to trajectory-region patterns under various constraints settings. Among them, the effectiveness and flexibility of our proposals are reflected by expressing and matching patterns of various complexity through the FSAs with various constraints instead of operating directly on trajectory-region strings. And the efficiency is measured by the runtime of querying, which assesses how efficiently the trajectory-region patterns approach querying targets from a huge analog dataset.

In this research, we start with a preprocessing step to facilitate fast retrieval, translating trajectory-region strings into a constrained-DFA based inner structure, which means not merely spatial features of trajectories matched with target patterns, but other non-spatial attributes like temporal information. This kind of pre-treatment will of course take some time, but it is a one-off conversion, and therefore benefits all queries on patterns afterwards. Obviously, for the trajectory-region movements, the string expression is elegant and more suitable for storage, but the DFA-based inner structure is more appropriate for human interpretation as well as machine evaluation. On the upper part of Figure 14, for example, an original trajectory is displayed in the form of a stop-move string, and then the lower part of Figure 14 shows the corresponding automation-based inner structure. This kind of representation permits us, for a specific stop-move string, to derive the specific starting and ending time of each state while calculating the timestamp of the moment as the relations between a trajectory and the region changes. Then it is easy to obtain the behaviors of a

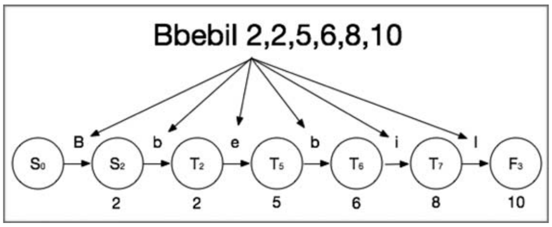


FIGURE 14 An example of a DFA-based inner structure

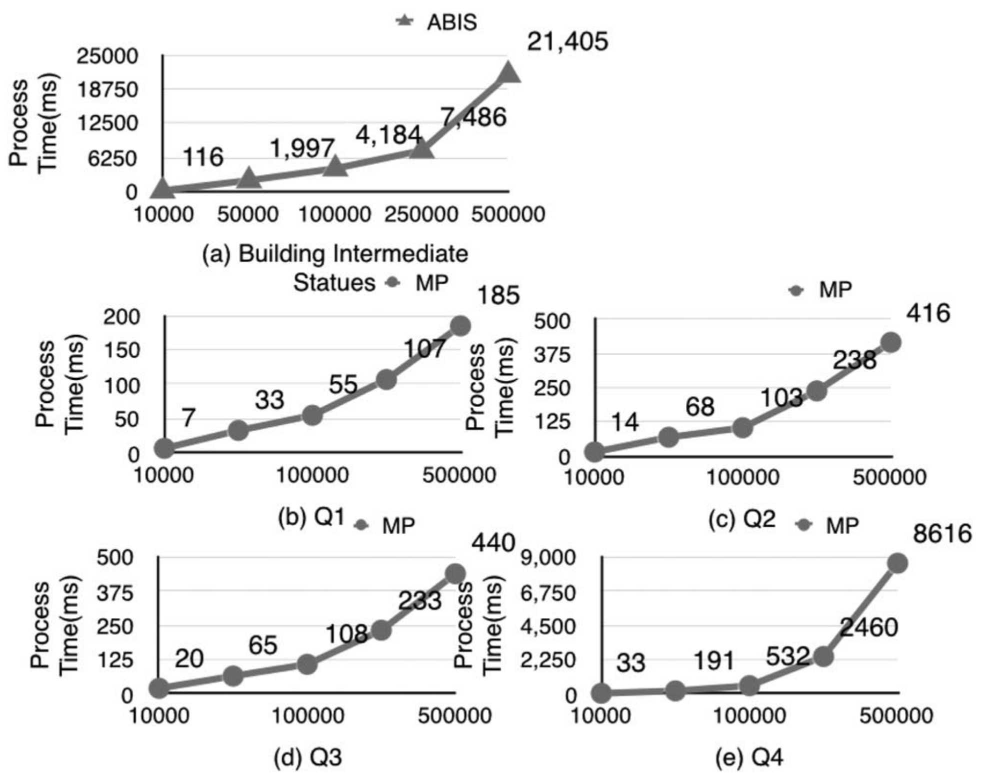


FIGURE 15 FSA-based queries for the performance tests

trajectory within certain given time constraints. In the remainder of this section, we will evaluate the cost of building the intermediate statues representation at first, and then test queries based on expressions of automations respectively.

7.3 | Performance tests

We analyze the performance of our approach with various magnitudes (five levels ranging from 10,000 to 500,000 trajectories). The results presented in Figure 15 have used trajectory sets with five levels of magnitude (10,000, 50,000, 100,000, 250,000 and 500,000 trajectories) to provide the search situation of the four examples mentioned in Section 6: Q1, Q2, Q3 and Q4. The runtime of queries is also calibrated accordingly. To measure the efficiency of our method further, the runtime of the pre-processing, translating trajectory-region strings into FSA-based inner structure (ABIS), is tested first, and the following are the runtimes of querying, matching patterns on the inner structures (MP).

Figure 15a shows the variation of time cost of building automation-based inner structures for five analog data sets with different magnitude from 10,000 to 500,000 recordings. And Figure 15b-e records the runtime of querying trajectories against four patterns respectively. It is easy to see that the runtime linearly increases with respect to the size of analog datasets. We observe that querying latencies are much lower than the runtime of translating trajectory-region strings into automation-based inner structures. That is because validating raw trajectory-region strings with FSA is resource intensive and time consuming. Fortunately, we will only take this process as a pre-treatment once ahead of focusing on executing query processes on FASs.

Additionally, as shown in Figures 15b-e, the runtime varies as the complexity of their pattern structures changes. For all of the four queries, Q4 is much more complicated than the other three. It involves two trajectory-region patterns as well as an operation of getting time information from specific trajectories. This translates to increased time consumption as trajectories increase. We plan to offset the effects of pattern complexity by employing index structures in subsequence research work.

8 | CONCLUSIONS

The increasing adoption of GPS chipset equipped devices is leading to the collection of large quantities of trajectory data in various sectors, and to the opportunity of discovering hidden knowledge behind raw point data. As a result, a huge amount of research work is involved in the exploring of trajectory data and its contextual information.

In this article, we investigated the problem of modeling and querying trajectory-region movements. Our proposal is based on a categorization of local topologies built on trajectory-region intersections, upon which the transition graphs of possible state transitions among local topologies are derived. We designed two types of trajectory-region strings, i.e. movement strings and stop-move strings, one to encode the sequence of trajectory-region intersections and the other to further integrate the stop information. With such string-format expressions, the movement of an object with respect to a region can be well interpreted without having to check them on a map. We also designed Finite State Automations to recognize the language of trajectory-region strings.

In order to facilitate the querying and analysis of interesting movement patterns hidden in trajectory-region strings retrieved from databases, we also designed trajectory-region patterns following the Backus-Naur Form, with which users can pose a wide range of queries to analyze trajectory-region movement, from simple queries like Q1 to advanced queries like Q4. Further, we developed a FSA-based method to effectively evaluate queries based on trajectory-region patterns, which was validated by experiments using analog data.

In a near future, besides continuing to explore trajectory-region movements based on trajectory-region strings, we will investigate three related research issues raised by this string-expressed model. The first one is to extend it to model the movement of one object against multiple regions, where the visiting order may be arbitrary, like firstly visiting region *A*, going next to region *B*, and then back to *A* again. With such an extension, we hope to support more general and advanced queries. The second point is to measure the similarities of a set of trajectories that have visited a same region. This will help to automatically identify trajectories with similar movement behaviors, and therefore can be used to classify trajectory-region movements. Finally, we are interested to further analyze those trajectories that never intersect with the boundary of a region but run close to the boundary from time to time, which will be useful for security monitoring systems to detect dangerous or abnormal movements. A feasible solution to this problem is to introduce a buffer zone to the region's boundary and then analyze non-intersecting trajectories with respect to the buffer.

ACKNOWLEDGMENT

This research was supported by the National Science Foundation of China (Grant Nos. 41471374 and 41001296).

REFERENCES

Alexandroff, P. (1961). *Elementary concepts of topology*. New York, NY: Dover Publications Inc.

Alvares, L. O., Bogorny, V., Kuijpers, B., Fernandes, J. A., Moelans, B., & Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. In *Proceedings of 15th Annual ACM International Symposium on Advances in Geographic Information Systems* (pp. 162–169). Seattle, WA.

Buchin, M., Driemel, A., Kreveld, M., & Sacristan, V. (2011). Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, *3*, 33–63.

Egenhofer, M. (1989). A formal definition of binary topological relationships. In *Third International Conference on Foundations of Data Organization and Algorithms* (pp. 457–472). Paris, France.

Egenhofer, M., & Franzosa, R. (1991). Point-set topological spatial relation. *International Journal of Geographical Information Systems*, *5*(2), 161−174.

Egenhofer, M., & Herring, J. (1991). Categorizing binary topological relationships between regions, lines and points in geographic databases. *Santa Barbara CA National Center for Geographic Information and Analysis Technical Report 01*, *94*, 1–28.

Egenhofer, M., & Mark, D. (1995). Modeling conceptual neighborhoods of topological line-region relations. *International Journal of Geographical Information Systems*, *9*(5), 555−565.

Freksa, C. (1992). Temporal resoning based on semi-intervals. *Artificial Intelligence*, *54*, 199–227.

Giannotti, F., Nanni, M., Pineli, F., & Pedreschi, D. (2007). Trajectory patten mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 330–339). San Jose, CA.

Güting, R. H. (2005). SECONDO: A database system for moving objects. *GeoInformatica*, *9*(1), 33–60.

Güting, R. H., Bohlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., & Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, *25*(1), 1–42.

Guttman, M. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data* (pp. 47–57). Boston, MA.

Hadjieleftheriou, M., Kollios, G., Bakalov, P., & Tsotras, V. J. (2005). Complex spatio-temporal pattern queries. In *Proceedings of the 31st International Conference on Very Large Data Bases* (pp. 877–888). Trondheim, Norway.

Hadjieleftheriou, M., Kollios, G., Tsotras, V. J., & Gunopulos, D. (2002). Efficient Indexing of Spatio-temporal objects. In *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 251–268). Prague, Czech Republic.

Kurata, Y., & Egenhofer, M. J. (2007). The 9^{+}-intersection for topological relations between a directed line and a region. In *First Workshop on Behavior Monitoring and Interpretation, in Conjunction with KI-2007* (pp. 62–76). Osnabrück, Germany.

Mouza, C. D., & Rigaux, P. (2005). Mobility patterns. *Geoinformatica*, *9*(4), 297–319.

Munkres, J. (1966). *Elementary differential topology*. Princeton, NJ: Princeton University Press.

Pelekis, N., Theodoridis, Y., Vosinakis, S., & Panayiotopoulos, T. (2006). HERMES - A framework for location-based data management. In *Proceedings of the Tenth International Conference on Advances in Database Technology* (pp. 1130–1134). Munich, Germany.

Pfoser, D., Jensen, C. S., & Theodoridis, Y. (2000). Novel approaches in query processing for moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases* (pp. 395–406). Cairo, Egypt.

Schneider, M., & Behr, T. (2006). Topological relationships between complex spatial objects. *ACM Transactions on Database Systems*, *31*(1), 39–81.

Spaccapietra, S., Parent, C., Damiani, M., Macedo, J., Porto, F., & Vangenot, C. (2008). A conceptual view on trajectories. *Data & Knowledge Engineering*, *65*(1), 126—146.

Spanier, E. (1966). *Algebraic topology*. New York, NY: McGraw-Hill.

Tao, Y., & Papadias, D. (2001). MV3R-Tree: A spatio-tempoal access method for timestamp and interval queries. In *Proceedings of the 27th International Conference on Very Large Data Bases* (pp. 431–440). Roma, Italy.

Vieira, M. R., Bakalov, P., & Tsotras, V. J. (2010). Querying trajectories using flexible patterns. In *Proceedings of the 13th International Conference on Extending Database Technology* (pp. 406–417). Lausanne, Switzerland.

Wolfson, O., Xu, B., Chamberlain, S., & Jiang, L. (1998). Moving objects databases: Issues and solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management* (pp. 111–122). Capri, Italy.

Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., & Aberer, K. (2012). Semantic trajectories: Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology*, *9*(4), 1–34.

How to cite this article: Xiang L, Wu T, Ettema D. An intersection-based trajectory-region movement study. *Transactions in GIS*. 2017;21:701–721. https://doi.org/10.1111/tgis.12220