

A Query Language for Moving Object Trajectories*

(Extended Abstract)

Hoda M. O. Mokhtar

Department of Computer Science
University of California at Santa Barbara
hmokhtar@cs.ucsb.edu

Jianwen Su

Department of Computer Science
University of California at Santa Barbara
su@cs.ucsb.edu

Abstract

Trajectory properties are spatio-temporal properties that describe the changes of spatial (topological) relationships of one moving object with respect to regions and trajectories of other moving objects. Trajectory properties can be viewed as continuous changes of an object's location resulting in a continuous change in the topological relationship between this object and other entities of interest. In this paper we develop a query language TQ for expressing trajectory properties. Our model and query language are based on the framework of constraint query languages. We present some preliminary complexity and expressive power results for the proposed language.

1 Introduction

Rapid technology advancement is revolutionizing the modern society in almost every possible ways. In particular, the ever shrinking computing devices and wireless communication devices are making it easier to collect, transmit, and process data. For example, wireless applications are now seen in everyday activities ranging from mobile phones applications to military and navigational applications. Today's market already has cellular phones (GSM), global positioning systems (GPS), traffic navigational systems, sensor networks, digital assistants (PDA); more innovations are on their way. Such an explosion of technology not only brings new problems but also presents serious challenges in areas such as data management.

Moving object databases (spatio-temporal databases) are one of those recent evolutions that emerged to fulfil some of the new urging requirements. Moving object databases integrate traditional spatial and temporal databases and studies managing and querying objects whose location and/or

shape change over time. Moving object databases appear in numerous applications including emergency services (E911), navigational and military services, flight management and tracking, m-commerce, and various location based services (LBS) as fleet management, vehicle tracking, mobile advertisements, etc. These advancements demand new techniques for managing and querying changing location information.

The aim of this paper is to study time varying topological properties for moving objects that we refer to as trajectory properties. In general, both moving object databases and topological properties have been of interest in a number of research work. For moving objects databases many research work was conducted to examine some problems including modeling and query languages [33, 38, 14, 12, 18, 35, 26, 7], handling large volume of location information through the use of efficient index structures [29, 21, 1, 32, 36, 5, 24], efficient data management, specifically, processing queries and handling updates [33, 34], query evaluation [20, 34, 3], and uncertainty management [28, 30, 37, 4, 25]. On the other hand topological predicates were studied in several works including [10, 11, 22, 27].

Nevertheless, designing efficient trajectory query languages continues to be an interesting problem. Prior approaches to query languages are mostly based on extending known languages or frameworks for spatial or spatio-temporal databases to allow expressing properties concerning trajectories (e.g., [33, 7, 18, 35, 15]). While general purpose spatial query languages could be used, the uniqueness of trajectories representing objects moving in space makes these general-purpose spatio-temporal languages unfit. For example, [26] exhibits difficulties for the general constraint query language to express some trajectory specific queries. In fact, querying trajectories seems to demand new techniques in the query languages. The recent effort in [13, 9] made a significant step towards understanding some of these issues. In [13] Erwig and Schneider elevate topological predicates to spatio-temporal predicates by aggregating a

*Support in part by NSF grant IIS-0101134.

temporal dimension to spatial predicates. They present a framework for expressing spatio-temporal predicates. The framework is based on aggregating time to elementary spatial predicates and sequential composition of predicates. In [9] du Mouza and Rigaux introduced the notion of mobility patterns which describe the motion pattern of a moving object. They model a moving object trajectory as a sequence of spatial zones and time spent at each zone. Their focus is on continuous evaluation of pattern matching queries and how to incrementally maintain the results.

Following those efforts we propose in this paper a constraint based query language (TQ) for reasoning about trajectory properties. We use the prevailing model of linear motions for moving object trajectories and allow a database to consist of a finite set of trajectories and a finite set of regions. We adopt constraint databases techniques [19, 23, 6, 16] to represent the trajectories and regions as “generalized relations”, i.e., boolean combination of linear constraints interpreted over the structure of real numbers. By basing our design on constraint databases, many techniques of constraint query languages and evaluations are immediately available to use.

Our language TQ consists of two components: a spatial component (called SQ) focussing on expressing spatial relationships at a time instant and a temporal component. SQ generalizes the classical CQL of [19] by allowing variables to represent regions and trajectories. This permits expressing spatial properties among trajectories and regions. SQ queries are then generalized to allow existential and universal quantification on a time interval, i.e., interval based spatial properties. The instant/interval properties are then put together in the temporal component as regular (formal) languages to model properties of trajectories. While such a method of “gluing” spatial properties along the time line is similar to [13, 9], it turns out that the use of constraint query languages in SQ significantly enhances the query language. We provide an expressive power comparison of our language with the languages in [13, 9] in the paper.

We study the expressive power of TQ and the complexity of evaluating queries in TQ. In this paper we establish the following technical results.

1. SQ (i.e. snapshot query language) has polynomial time data complexity and exponential space combined complexity (i.e. query expression and database complexity).
2. TQ queries can be effectively evaluated.
3. TQ is more expressive than the language of [13] and the variable free queries of [9].

The paper is organized as follows. Section 2 defines the model for the moving object trajectories. Section 3 introduces the language TQ for trajectories. Section 4 presents

the complexity results for TQ and a sublanguage of TQ (star-free). Section 5 studies expressive power of TQ. Section 6 concludes the paper.

2 A Data Model for Moving Objects

In this section we present necessary concepts for spatio-temporal objects to be used in the paper. Key notions include that of a “moving object” and its “trajectory.” Roughly speaking, moving objects are spatio-temporal objects whose location and/or extent change over time. In general, such spatio-temporal objects can be points or regions. Moving points are suitable to model planes, cars, buses, trains, people, etc. whose locations and movements are important. On the other hand, applications concerning oil spills, fires, forests, pollution, etc. are very dependent on both shapes and locations of such moving objects. In this paper, we assume moving points and static regions to focus on query languages for trajectories. With some proper modeling of regions that changing over time, the results may be generalized to include such regions.

Our data model is based on “linear constraints” that are logical formulas over the real closed field. Kanellakis, Kuper, and Revesz in their seminal paper [19] demonstrated that such logical formulas can represent spatial and spatio-temporal information in an abstract manner independent of the underlying storage mechanisms (cf [23]).

Specifically, our model is an extension of the constraint data model [19] to represent a finite set of regions¹ and moving object trajectories in the database. Some key techniques of the results concerning constraint formulas in this paper are also extended from the constraint query evaluation techniques [23]. More details will be discussed in Sections 3 and 4.

We now proceed with the technical presentation, starting with constraints. Let \mathbb{R}, \mathbb{N} be the set of real and natural numbers (respectively). Consider a first order language L for \mathbb{R} that includes equality and order predicates ($=, <, \leq, >, \geq$), a binary function for addition ($+$), and a unary *coefficient* function “ c .” for each real number $c \in \mathbb{R}$. Intuitively, the unary coefficient functions are used to represent real coefficients. For simplicity, we will denote “ $c \cdot (x)$ ” as “ cx ” for each $x \in \mathbb{R}$.

Let $n \in \mathbb{N}$ and $n > 0$. An *atomic linear constraint* over variables x_1, \dots, x_n is an expression of the following form:

$$(\sum_{i=1}^n c_i x_i) \theta c_0 \quad \text{or} \quad c_1 x_1 + c_2 x_2 + \dots + c_n x_n \theta c_0$$

where c_0, c_1, \dots, c_n are real numbers in \mathbb{R} and θ is a predicate in L . Constraints are interpreted over the real numbers in the natural manner, i.e., if $\varphi(x_1, \dots, x_n) =$

¹Regions in this paper are non-changing and spatial. Generalization to time-dependent regions can be done easily.

$(\sum_{i=1}^n c_i x_i) \theta c_0$ is an atomic constraint, and a_1, \dots, a_n are real numbers in \mathbb{R} , $\varphi(a_1, \dots, a_n)$ is true if $(\sum_{i=1}^n c_i a_i) \theta c_0$ is satisfied (interpreted over the real numbers).

A *linear constraint* over variables x_1, \dots, x_n is a boolean combination of atomic linear constraints over variables x_1, \dots, x_n . An advantage of (linear) constraints is their ability to *finitely* represent potentially *infinite* sets of points (i.e., regions).

Definition: Let $n > 0$ be a natural number. An *n-dimensional region* is a linear constraint in disjunctive normal form over n variables. Let Reg^n denote the set of all n -dimensional regions.

In the remainder of the paper, we will fix the dimensionality of the space to be n for some $n > 0$ and may simply use Reg instead of Reg^n .

We now consider moving objects and regions in the n -dimensional real space \mathbb{R}^n . Object movements are viewed as location changes over “time”. We model the time domain as a domain isomorphic to \mathbb{R} (with a dense total order, addition, and multiplication by a real number). As we shall see in the technical presentation, we will introduce some restrictions on time instants in both the data model and query languages. To make the presentation clear, we use \mathbb{T} to denote the densely ordered domain of *time instants*. We only allow one variable for the time domain, denoted as t .

A moving object trajectory basically defines the motion of an object. Different alternatives for modeling trajectories have been studied in [33, 38, 14, 12, 18, 35, 26]. In this paper we use the standard model for trajectory as a sequence of line segments in the n -dimensional space. Instead of keeping the segments’ endpoints to define a trajectory as in [14, 29], we use constraints to define a trajectory. Constraints allow us to focus on logical properties in querying trajectories. Technically, a trajectory is a sequence of linear motions defined as follows.

Let Flin be a set of expressions of the form “ $at + b$ ” where $a, b \in \mathbb{R}$. Each element in Flin represents a linear function from \mathbb{T} to \mathbb{R} .

Definition: An *n-dimensional motion* is an n -tuple $m \in \text{Flin}^n$. For each $1 \leq i \leq n$, m_i denotes the i -th value of m .

Intuitively, an n -dimensional motion m defines a linear function from \mathbb{T} to \mathbb{R}^n ; m_i represents the location change along the i -th dimension. For example, for a given time instant a and a motion m , $m(a)$ represents the point (location) in \mathbb{R}^n at time a , while $m_i(a)$ is the position on the i -th dimension.

Definition: A *trajectory* is a sequence $(z_0, m_0, z_1, m_1, \dots, z_k, m_k)$, where $k \in \mathbb{N}$ and for each $0 \leq i \leq k$,

- z_i is a time instant in \mathbb{T} such that for each $0 \leq i < k$, $z_i < z_{i+1}$, and

- m_i is a motion such that for $0 \leq i < k$,

$$m_i(z_{i+1}) = m_{i+1}(z_{i+1}).$$

Let Traj denote the set of all trajectories.

Intuitively, in a trajectory $(z_0, m_0, z_1, m_1, \dots, z_k, m_k)$, the motion m_i (for $1 \leq i < k$) defines the object location from time z_i to z_{i+1} , m_k defines the object location for all times after z_k , and the object does not exist before time z_0 . It is easy to see that a trajectory is a continuous piecewise linear function from \mathbb{T} to \mathbb{R}^n which always has a starting time but no ending time.

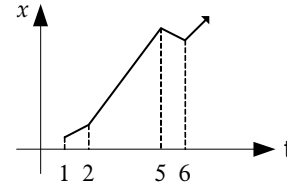


Figure 1. A 1-dimensional Trajectory

Example 2.1 Consider the 1-dimensional space and a moving object whose trajectory is defined as:

$$(1, (t + 4), 2, (2t + 2), 5, (-t + 17), 6, (2t - 1)).$$

Figure 1 shows the beginning part of the trajectory. The trajectory consists of a sequence of four motions h, p, q, r where $h = (t + 4)$ defined the object location for the time interval $[1, 2]$, $p = (2t + 2)$ for the time interval $[2, 5]$, $q = (-t + 17)$ for the time interval $[5, 6]$, and $r = 2t - 1$ for all times after 6. ■

We assume that \mathbf{U}_R and \mathbf{U}_T are two disjoint countably infinite set of names (identifiers) for regions and trajectories, respectively.

Definition: A (*moving object*) *database* (or *MOD*) is a quadruple $d = (N_R, N_T, f_R, f_T)$ where $N_R \subseteq \mathbf{U}_R$ and $N_T \subseteq \mathbf{U}_T$ are finite subsets, f_R is a mapping from N_R to Reg , and f_T is a mapping from N_T to Traj .

The data model (trajectories and regions) are similar to the ones in the literature (e.g., [18, 38, 35]), except that regions do not change over time. The trajectory model was originally defined in [26], it is different from the model used in [9] where a trajectory is a sequence of spatial zones and time values representing the time spent at each zone. In [13] the authors also discuss spatio-temporal properties but no model for trajectories is provided in their framework.

Finally, the model ignores the usual semantic modeling, i.e., separating the set of “cars” from that of “trucks”. This is to simplify the technical presentation and to allow us focussing on the querying aspect of trajectories.

3 A Query Language for MOD

In this section we propose a trajectory query language (TQ) for expressing spatio-temporal properties of moving object trajectories. TQ is based on constraint query languages and is powerful enough to express properties between objects and regions and between different objects.

Definition: A (*spatial*) *term* is an expression of one of the following forms:

- a where $a \in \mathbb{R}$,
- x where x is a spatial (i.e., real) variable,
- cx where $c \in \mathbb{R}$ is a coefficient function and x a spatial variable, and
- $s_1 + s_2$ where s_1 and s_2 are spatial terms.

We now define the central notion of a “spatial formula”. Intuitively, a spatial formula can express properties concerning positions with the time variable t as a parameter. Spatial formulas resemble formulas in constraint query languages of [19]. However, the primary difference is that in constraint query languages, only named regions can be used, while in our language, spatial formulas can reference named regions (i.e., region names), named trajectories, and also variables that represent regions and trajectories. This is natural since a moving object database may have arbitrary number of regions and trajectories.

Recall that n is the number of dimensions and t is the time variable.

Definition: The set of *spatial formulas* is defined recursively as follows.

- $s_1 \theta s_2$ is an *atomic* spatial formula if s_1 and s_2 are spatial terms and $\theta \in \{=, <, >, \leq, \geq\}$,
- $v(s_1, \dots, s_n)$ is an *atomic* spatial formula if v is a region variable or a region name in \mathbf{U}_R and s_1, \dots, s_n are spatial terms,
- $v(t, s_1, \dots, s_n)$ is an *atomic* spatial formula if v is a trajectory variable or a trajectory name in \mathbf{U}_T and s_1, \dots, s_n are spatial terms,
- $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, and $(\neg\varphi)$ are spatial formulas if φ and ψ are spatial formulas,
- $\exists x\varphi$, $\forall x\varphi$ are spatial formulas if x is a spatial variable and φ a spatial formula.

The semantics of spatial formulas is defined in the straightforward manner. Technically, a *valuation* is a mapping that maps

- each spatial variable to \mathbb{R} ,
- the time variable t to \mathbb{T} ,
- each region name and each region variable to a region (a generalized relation) in Reg , and
- each trajectory name and each trajectory variable to Traj .

A valuation can be naturally extended to all spatial terms. Then the *truth* value of a spatial formula φ under a valuation is identical to a constraint formula in a constraint query language with the following exception:

- We will represent the time domain \mathbb{T} as real numbers in \mathbb{R} , and consequently, each trajectory in Traj is treated as an $(n + 1)$ -ary generalized relation with the first dimension representing the time.

Spatial formulas are then used to construct “query formulas” defined below.

Definition: Let $d = (N_R, N_T, f_R, f_T)$ be a MOD. The set of *query formulas* over d is defined below.

- A spatial formula φ is a query formula if φ has no spatial variables occurring free and each region (respectively trajectory) name occurring in φ is in N_R (respectively N_T).
- $z_1 = z_2$ is a query formula if both z_1, z_2 are region variables/names, or trajectory variables/names.
- $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$, $(\neg\varphi)$, $(\exists z\varphi)$, $(\forall z\varphi)$ are query formulas if z is a region variable or trajectory variable and $\varphi, \varphi_1, \varphi_2$ are query formulas.

A *snapshot (trajectory) query* is a query formula with exactly two variables occurring free: the time variable t and a trajectory variable. Let SQ denote the set of all snapshot queries.

The semantics of the query language SQ is defined in the standard way based on the semantics of spatial formulas. Let $d = (N_R, N_T, f_R, f_T)$ be a moving object database and $\varphi(t, z)$ a snapshot query. A mapping is said to be an *assignment* with respect to d if it maps each variable to its domain, i.e., spatial variables to \mathbb{R} , t to \mathbb{T} , region variables to N_R , and trajectory variables to N_T . Indeed, let α be an assignment, we can derive a valuation by simply taking the composition $\alpha \circ (f_R \cup f_T)$. The semantics for quantifiers on region and trajectory variables is standard, here we take the “active domain” semantics for these variables, i.e., only region and trajectory names in the database are considered. Finally, we say that the database *satisfies* φ at time $c \in \mathbb{T}$ for some trajectory (name) $\tau \in N_T$, written as $(d, c) \models \varphi[\tau]$, if the query formula returns true.

Example 3.1 Let τ be the name of an object trajectory and r_1, r_2 be names of two regions in the 2 dimensional space. The property that τ is in the intersection of r_1, r_2 can be expressed as the following spatial formula (snapshot query).

$$\exists x \exists y \tau(t, x, y) \wedge r_1(x, y) \wedge r_2(x, y)$$

The property that no regions intersecting the rectangle of size $2a$ centered at τ can be expressed by the following snapshot query:

$$\exists x \exists y \tau(t, x, y) \wedge \forall r \forall x' \forall y' (r(x', y') \rightarrow \neg(x - a \leq x' \leq x + a \wedge y - a \leq y' \leq y + a)) \quad \blacksquare$$

Snapshot queries provide a basis for expressing trajectory properties. In [13, 9] trajectory properties are considered as spatial properties varying along time. Similar to their approaches, we will develop a “temporal” portion on top of the SQ language to express trajectory properties. Different from their approaches, our language allows explicit definition of transition time (from one spatial property to the next) through querying the database explicitly, and the use of SQ significantly enriches the expressiveness of spatial properties.

Trajectory properties maybe expressed as snapshot properties or interval properties. As in SQ, snapshot properties focus on the properties that a moving object trajectory would satisfy at a given time instant, whereas interval properties require a moving object trajectory to satisfy the property during a given interval.

Since trajectory properties depend on either the time instant or the time interval it is checked against, expressing time in a trajectory property is a key issue. In many applications, a query checks a reoccurring property that repeats itself at different time instants. Such queries require the language to support expressing both absolute and relative times. The following example illustrates the difference between queries that require absolute and relative time resp.

Example 3.2 Consider the following queries on trajectories:

Q_1 : Retrieve all delivery trucks that entered the Santa Barbara area at 2:00pm and stayed there until 5:00pm.

Q_2 : Retrieve all delivery trucks that entered the Santa Barbara area at 2:00pm and stayed there for 35 minutes.

The main difference between Q_1 and Q_2 is that the ending time for Q_1 is fixed. On the other hand, Q_2 requires the trajectory property of staying inside Santa Barbara to be checked until 35 minutes after the time when the object entered. Q_2 thus requires the language to express things like “entering time” +35. \blacksquare

In our design, we capture this requirement through (1) the use of “time expressions” to define time instants, (2) the use of a “relative time expressions” to refer to the difference between the current time and the previous time instant of interest.

To facilitate the development, we start with some necessary technical notions. Note that t is the time variable,

Definition: Let φ be a query formula with only t occurring free. Then, a *time expression* is one of the following:

- c , if $c \in \mathbb{T}$,
- $\min(\varphi)$, or
- $\max(\varphi)$.

A *relative time expression* is an expression “+ e ” where e is a time expression and “+” is a special symbol.

Intuitively, a (relative) time expression defines a time instant either explicitly or through querying the database. The functions \min, \max returns the smallest, respectively largest time instant that satisfies the query formula. In case the smallest/largest instant does not exist, the expression is not *well-formed*. Let e be a time expression and d a database. We denote by $e(d)$ the *result* of e under d . The following property shows that well-formedness can be checked if a database is given; undecidable if it is well-formed independent of databases.

Proposition 3.3 For a given time expression e and a MOD d , it can be decided where e is well-formed. However, it is undecidable if e is well-formed for all databases.

The positive result can be proved using an argument similar to the proof of Lemma 4.1 in Section 4 about complexity. The negative results is a direct consequence of undecidability results of constraint query properties [17].

Combining snapshot queries with (relative) time expressions we can define a language for expressing trajectory properties.

Definition: Let z be a designated trajectory variable. If e is a (relative) time expression and φ is a snapshot query in SQ, then the following are *startless atomic trajectory queries*:

- $\varphi(t, z)$ is a startless *snapshot* trajectories query,
- $\epsilon^\exists.\varphi(t, z)$ is a startless *existential-time* trajectory query, and
- $\epsilon^\forall.\varphi(t, z)$ is a startless *universal-time* trajectory query.

A *startless trajectory query* is a regular expression over the set of startless atomic trajectory queries, i.e., composed from startless atomic trajectory queries using concatenation ($q_1 q_2$), union ($q_1 + q_2$), and closure (q^*), where q, q_1, q_2 are

startless trajectory queries. If the regular expression does not use the closure operator, the startless trajectory query is called *star-free*.

A startless trajectory query q defines a formal language (i.e., set of words) over startless atomic trajectory queries. We denote the language as $\text{SEQ}(q)$. Each word expresses a sequence of atomic properties. There are three types of atomic properties in the language. A snapshot trajectory query examines the spatial properties at a time instant. An existential-time trajectory query checks if the spatial properties hold at some time instant during a time interval. Finally, a universal-time trajectory query insists that the spatial properties should be true for all time instants during the time interval. The word “startless” indicates that the start time is not given. The end of a time interval for an existential- or universal-time trajectory query is given by the time expression ϵ preceding the snapshot query formula.

Definition: An (*atomic, snapshot, existential-time, universal-time, star-free*) trajectory query is a pair (ϵ, q) , where ϵ is a time expression and q a startless (resp., atomic, snapshot, existential-time, universal-time, star-free) trajectory query. Let TQ denote the set of all trajectory queries.

We outline the semantics for TQ below.

Let $c_0 \in \mathbb{T}$ be a time instant, $q_1 \dots q_\ell$ be a sequence of atomic trajectory queries, and d a MOD database. We define a time instant sequence c_0, c_1, \dots, c_ℓ as follows: for each $1 \leq i \leq \ell$,

- if q_i is a snapshot query, $c_i = c_{i-1}$,
- otherwise, let e_i be the time expression in q_i . Let $c_i = c_{i-1} + e_i(d)$ if q_i has a relative time expression, otherwise let $c_i = e_i(d)$.

Let $\tau \in \mathbf{U}_T$ be a trajectory name. We say that the database d satisfies the atomic query sequence $q_1 \dots q_\ell$ for τ at c_0 , denoted as $(d, c_0) \models q_1 \dots q_\ell[\tau]$, if the sequence c_0, c_1, \dots, c_ℓ is monotonically increasing (not necessarily strict) and for each $1 \leq i \leq \ell$, the following are true:

1. If $q_i = \varphi(t, z)$ is a snapshot query, then $(d, c_i) \models \varphi[\tau]$,
2. If $q_i = \epsilon^\exists.\varphi(t, z)$ is an existential-time query, then $(d, c) \models \varphi[\tau]$ for some $c_{i-1} < c < c_i$, and
3. If $q_i = \epsilon^\forall.\varphi(t, z)$ is a universal-time query, then $(d, c) \models \varphi[\tau]$ for each $c_{i-1} < c < c_i$.

Let $q = (\epsilon, q_1 \dots q_\ell)$ be a trajectory query where each q_i ($1 \leq i \leq \ell$) is atomic, and $d = (N_R, N_T, f_R, f_T)$ a database. A trajectory (name) $\tau \in N_T$ is in the *answer* to q over d , if $(d, \epsilon(d)) \models q_1 \dots q_\ell[\tau]$. Denote by $q(d)$ the set of all trajectories in the the answer to q over d .

Let $q = (\epsilon, q')$ be a trajectory query and $d = (N_R, N_T, f_R, f_T)$ be a MOD database. The *answer* of q over d is the set $q(d) =$

$$\{\tau \mid \tau \in N_T, \text{ and } \exists q'' \in \text{SEQ}(q'), (d, \epsilon(d)) \models q''[\tau]\}$$

4 Complexity Results

In this section we present complexity results for the trajectory query language TQ. In the literature, data and combined complexity are used to measure the complexity of query languages. The main results in the section show that SQ and TQ have polynomial time data complexity and exponential space combined complexity.

The *complexity* of a query is the time/space needed to compute the answer. We consider two complexity measures. *Data complexity* of a query measures the complexity in terms of the database size, i.e., the query expression is considered fixed; *combined complexity* of a query measures the complexity in terms of both the database size and query expression size. We use PTIME, EXPSPACE to denote the polynomial time, (respectively) exponential space complexity classes. A query language has *complexity* C if every query in the language has complexity C .

Lemma 4.1 The snapshot query language SQ has PTIME data complexity and EXPSPACE combined complexity.

We note here that the main source of combined complexity is from the number of variables. Indeed, we can show that the space complexity is exponential in the number of variables in a query expression φ .

Proof: (Sketch) Technically, the result states that for each time instant $c \in \mathbb{T}$, each SQ query $\varphi(t, z)$, each database $d = (N_R, N_T, f_R, f_T)$, and each trajectory (name) $\tau \in N_T$, $(d, c) \models \varphi[\tau]$ can be decided in (1) polynomial time in the size of d , and (2) exponential space in the size of d and φ .

The main idea of the proof is as follows. We consider a fixed mapping γ that assigns a region name in N_R to each region variable and a trajectory name in N_T to each trajectory variable including τ . Let $Q[\gamma]$ denote the snapshot query obtained from Q by replacing each region/trajectory variable z by $\gamma(z)$. Note that $Q[\gamma]$ is a snapshot query without any region and trajectory variables, it does have, however, spatial variables. Mapping the time domain \mathbb{T} to real numbers \mathbb{R} and t to a spatial variable, we can now view $Q[\gamma]$ as a constraint query in the model of [19, 17]. It can be concluded that $Q[\gamma]$ can be evaluated in polynomial time in the size of the image of the composed mapping $\gamma \circ (f_R \cup f_T)$ which is a subset of d . From the results in [19], the result of evaluating $Q[\gamma]$ is a set of intervals for t that makes $\varphi[\gamma]$ true. In particular, the set can be computed effectively since $\varphi[\gamma]$ has only linear constraints.

Now let τ, z_1, \dots, z_ℓ be an enumeration of region and trajectory variables occurring in φ . We can consider all possible assignments from $\{\tau, z_1, \dots, z_\ell\}$ to $N_R \cup N_T$ that preserve the type. For each such assignment γ , we can repeat the above process. Since the total number of assignments is $(|N_R| + |N_T|)^{\ell+1}$, and ℓ depends on the query expression φ that is considered as fixed, the number of assignment is a polynomial in the size of d .

Finally the combined complexity is due to fact that the complexity of evaluating queries is double exponential time in the number of quantifiers [31]. ■

Theorem 4.2 Atomic trajectory queries have PTIME data and EXPSPACE combined complexity.

Proof: (Sketch) Clearly, snapshot queries have PTIME data and EXPSPACE combined complexity, by Lemma 4.1. From constraint query evaluation algorithms, we can see that the result of evaluating a snapshot query is a set of time instants (possibly infinite) represented in constraint form as a generalized relation. Therefore, checking the existential-time and universal-time property simply becomes checking the intersection and (respectively) containment of interval, which can be done efficiently. ■

We now consider general trajectory queries. The idea is to use the evaluation procedure for atomic queries as the basis and extend for “path queries” defined in regular expressions. Technically, we will construct from a trajectory query a “query graph” and then develop a generic iterative process for query evaluation. We show that the iterative process ends in finite number of steps. The number of steps, however, may depend on the properties of the constraints used in defining the regions and trajectories in the database.

Let $q=(\epsilon, q')$ be a trajectory query. Since q' is a regular expression, let G_q be a finite automaton that accepts the same language as q' (see [2]). Without loss of generality, we assume that G_q does not have *empty* moves. In fact, G_q can be viewed as a (*query*) *graph* (V, E, v_0, F, Q) where V is a finite set of nodes (states), $E \subseteq V \times V$ is a set of edges between nodes, $v_0 \in V$ and $F \subseteq V$ are the starting and final nodes respectively, and Q is a mapping from E to atomic trajectory queries in q' .

A path u_0, u_1, \dots, u_ℓ in a query graph G_q is called an *advancing edge* if (i) $\ell > 0$, and (ii) either $u_\ell \in F$ is a final node, or $Q(u_{\ell-1}, u_\ell)$ is not snapshot but $Q(u_{i-1}, u_i)$ is a snapshot for all $1 \leq i < \ell$.

Let $d = (N_R, N_T, f_R, f_T)$ be a database. We define an (infinite) relation $E \subseteq \mathbb{T}^2 \times V^2 \times N_T$ as follows:

- $(c_1, c_2, u_0, u_\ell, \tau) \in E$ if there is an advancing edge u_0, u_1, \dots, u_ℓ such that if $Q(u_{\ell-1}, u_\ell)$ is not snapshot, $(d, c_1) \models \bigwedge_{i=1}^{\ell-1} Q(u_{i-1}, u_i)[\tau]$, c_2 is the result of (relative) time expression in $Q(u_{i-1}, u_i)$, and τ satisfies

$Q(u_{\ell-1}, u_\ell)$ in d from time c_1 to c_2 , otherwise, $(d, c_1) \models \bigwedge_{i=1}^{\ell} Q(u_{i-1}, u_i)[\tau]$ and $c_1 = c_2$.

By Theorem 4.2, $(c_1, c_2, u_0, u_\ell, \tau) \in E$ can be decided in polynomial time in the size of d .

Finally, we define a sequence of relations $A_i \subseteq \mathbb{T} \times V \times N_T$ as follows:

- (i) $A_0 = \{c_0\} \times \{v_0\} \times N_T$ where c_0 is the value of ϵ , and
- (ii) $A_{i+1} = A_i \bowtie E$, where the join is defined as $(c_2, u_2, \tau) \in A_i \bowtie E$ if both $A_i(c_1, u_1, \tau)$ and $E(c_1, c_2, u_1, u_2, \tau)$ hold.

Proposition 4.3 Given the above context,

$$q(d) = \pi_3 \sigma_{2 \in F} \cup_{i \geq 0} A_i. \quad \blacksquare$$

The procedure outlined above can semi-compute the query answer, i.e., if a trajectory $\tau \in \pi_3 A_i$ for some i , then τ is in the answer. However, it is not clear when we can decide if τ is not in the answer. The following result implies that there is a time upper bound to decide if τ will never be in the answer.

Theorem 4.4 Given a trajectory query q , a MOD d , and a trajectory name in d , there exists $c \in \mathbb{T}$ such that for all $h, h' \in \mathbb{T}$ if $h > c$ and $h' > c$, $(d, h) \models q'[\tau]$ iff $(d, h') \models q'[\tau]$ where q' is an atomic query in q .

The proof of the result is rather involved and will be given in the full paper. The key idea is based on algebraic cell decomposition (e.g., [31, 8]). Specifically, for a given finite set of constraints, there is a partition of space into a finite number of “cells”, each of which preserves the constraints. This intuitively would indicate that after some point in time, the constraint (spatial) properties will not change.

Theorem 4.5 1. Let q be a trajectory query and d a MOD. It can be decided if a trajectory is in $q(d)$.

- 2. Star-free TQ has PTIME data and EXPSPACE combined complexity. ■

In the remainder of the section we introduce a restricted sublanguage of TQ, called “intermediate-variable free” trajectory queries. Roughly, these queries use neither region nor trajectory variables, and have only n quantifiers over the spatial variables for the n dimensions.

Definition: A query formula is *intermediate-variable free*, (or *int-var-free*), if it only has n quantifiers, one for each spatial variable for a dimension. A trajectory query in TQ (snapshot query in SQ) is *intermediate-variable free* if all query formulas in it are int-var-free.

Example 4.6 The query “was the United Flight UA80 inside the Los Angeles airport between 5pm and 5:10pm?” can be expressed by in int-var-free TQ as

$$(e_0, e_1. \forall x_1 \exists x_2 \text{UA80}(t, x_1, x_2) \wedge \text{LAX}(x_1, x_2)),$$

where e_0, e_1 are time expressions. ■

We argue that many queries can be expressed in this set. For technical results, we show that star-free trajectory queries in this subclass have PTIME combined complexity.

Theorem 4.7 Int-var-free SQ and star-free, int-var-free TQ have PTIME combined complexity. ■

5 Expressive Power

In this section we study the expressive power of trajectory languages developed in this paper. Our main goal is to compare our languages with the trajectory query languages in [13, 9]. Our results show that TQ is quite expressive, and even int-var-free TQ is sufficient to express many queries in the languages of [13, 9].

Since we will compare different languages that are defined over different data models, we need to formulate the technical basis for comparison. We outline the comparison framework informally below, the precise definitions are provided in the full paper.

In the “mobility patterns” language developed by du Mouza and Rigaux [9], a database consists of (1) a set of regions that form a partition of \mathbb{R}^2 , and (2) a set of objects each of which is a sequence of regions the object goes through (in that order) and the length of time it stays in each region. The time domain used in their model is discrete. For the sake of comparisons, we extend their time domain to a continuous one isomorphic to \mathbb{T} . Although the change may affect query evaluation, the semantics is clear. We use MP to denote the mobility patterns language with this extension.

Given a MOD d over \mathbb{R}^2 , we map d into the MP data model in the following sense: we construct a partition of \mathbb{R}^2 using the regions in d by considering all possible combinations among them. Given a trajectory z in d , we then construct an MP object o by listing the regions in the partition z goes through. The time lengths can also be computed easily. We denote the mapping MP.

A TQ query Q_1 and an MP query Q_2 are *equivalent* if for each MOD d , $\text{MP}(Q_1(d)) = Q_2(\text{MP}(d))$.

A language QL_2 is as *expressive* as another language QL_1 , denoted as $\text{QL}_1 \sqsubseteq \text{QL}_2$, if for every query in QL_1 there is an equivalent query in QL_2 . QL_1 and QL_2 are *equivalent*, $\text{QL}_1 \equiv \text{QL}_2$, if $\text{QL}_1 \sqsubseteq \text{QL}_2$ and $\text{QL}_2 \sqsubseteq \text{QL}_1$.

Roughly speaking, a mobility pattern is a sequence of regions that a moving object goes through. Based on a single “inside-a-region” predicate, a mobility pattern is a regular expression over the predicates. MP allows regions to be identified with either names or variables. When a variable is used, the variable is instantiated *prior* to matching the patterns. In other words, region variables are global to the mobility patterns. Clearly TQ has no such global region variables and having such global region variable would increase expressive power. Thus,

Theorem 5.1 Variable-free MP $\sqsubseteq \neq$ int-var-free TQ. ■

The reason for the converse is that more spatial properties can be expressed in TQ using SQ.

The main reason that MP is not contained in int-var-free TQ is that it can use (existentially quantified) region variables while describing pattern expressions. It appears that TQ can be extended to allow free region and even trajectory variables while expressing patterns. Although it is interesting to work out the detailed semantics (e.g., allowing arbitrary quantifications), we think the data and combined complexity will remain unchanged with respect to the PTIME and EXPSPACE complexity classes.

We now consider the “language” proposed by Erwig and Schneider [13], which we call MS. Technically, MS allows expressing temporal properties in the same spirit as TQ, but it does not have a coincide language corresponding to SQ to express spatial properties. For our comparison purposes, we will use (a sublanguage of) SQ as the underlying sublanguage for spatial properties. For example, the language MS+SQ uses SQ for spatial properties and MS for temporal trajectory properties. Note that by using MS+SQ, comparisons will use MOD databases.

There are two important differences. Firstly, time is implicit in the predicates through lifting them from pure topological predicates to spatio-temporal predicates. In contrast, TQ allows an SQ query to be evaluated to define the time. For example, one can express in TQ when two moving objects enter region r_1 , the third object must be outside of region r_2 . Such a flexibility of referencing time is not provided in MS. Secondly, MS does not allow the use of closure in expressing patterns. Clearly this puts a severe restriction on its expressiveness.

Thus our comparison is based on the setting that when (int-var-free) SQ is used as the underlying spatial language in MS. We have:

Theorem 5.2 (1) MS+SQ $\sqsubseteq \neq$ star-free TQ.

(2) MS+int-var-free SQ $\sqsubseteq \neq$ star-free int-var-free TQ. ■

6 Conclusions

In this paper we consider the problem of querying about spatial properties of moving objects. We presented a language TQ for expressing trajectory properties. We studied the complexity and expressive power of TQ. Our preliminary results show that TQ queries can be effectively evaluated, star-free TQ queries have PTIME data complexity, and intermediate variable free TQ queries have PTIME combined complexity.

Our work reported here makes only one step towards the design of query languages for trajectories. One immediate problem is to identify sublanguages of TQ that allow lower theoretical complexity bounds, efficient algorithms, and suitable data structures. It is unclear at this point how the known trajectory index techniques could be utilized in the query evaluation algorithms for trajectory query languages.

References

- [1] P. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *19th ACM Symposium on Principles of Database Systems*, pages 175–186, 2000.
- [2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, 1974.
- [3] R. Benetis, C. S. Jensen, G. Karcauskas, and S. Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *Sixth International Database Engineering and Applications Symposium*, Edmonton, Alberta, Canada, July 2002.
- [4] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *ACM International Conference on Management of Data SIGMOD*, 2003.
- [5] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *vldb*, Toronto Canada, 2004.
- [6] J. Chomicki and P. Revesz. Constraint-based interoperability of spatiotemporal databases. *GeoInformatica*, 3(3):211–243, 1999.
- [7] J. Chomicki and P. Z. Revesz. A geometric framework for specifying spatiotemporal objects. In *Proc. 6th Int. Workshop on Temporal Representation and Reasoning, (TIME '99)*, pages 41–46, Orlando, FL, May 1999.
- [8] G. E. Collins. Quantifier elimination for real closed fields by cylindrical decompositions. In *Proc. 2nd GI Conf. Automata Theory and Formal Languages*, volume 35 of *Lecture Notes in Computer Science*, pages 134–83. Springer-Verlag, 1975.
- [9] C. du Mouza and P. Rigaux. Mobility patterns. In *Proceedings of the Second Workshop on Spatio-Temporal Database Management (STDBM04)*, Toronto, Canada, August 2004.
- [10] M. J. Egenhofer. Reasoning about binary topological relations. In *Proc. Symp. on Large Spatial Databases*, 1991.
- [11] M. J. Egenhofer and R. Franzosa. Point-set topological spatial relations. *Int. Journal of Geo. Info. Systems*, 5(2):161–174, 1991.
- [12] M. Erwig, R. H. 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 M. Schneider. Spatio-temporal predicates. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):881–901, 2002.
- [14] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structures for moving objects databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2000.
- [15] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE system for complex spatial queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, June 1998.
- [16] S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-temporal data handling with constraints. In *Proc. ACM Symp. on Advances in Geographic Information Systems*, 1998.
- [17] S. Grumbach and J. Su. Finitely representable databases. *J. Comput. Syst. Sci.*, 55(2):273–298, Oct. 1997.
- [18] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Varigiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1), 2000.
- [19] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. *J. Comput. Syst. Sci.*, 51(1):26–52, 1995.
- [20] G. Kollios, D. Gunopulos, and V. J. Tsotras. Nearest neighbor queries in a mobile environment. In *Spatio-Temporal Database Management*, pages 119–134, 1999.
- [21] G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *Proc. ACM Symp. on Principles of Database Systems*, pages 261–272, 1999.
- [22] B. Kuijpers and J. V. den Bussche. On capturing first-order topological properties of planar spatial databases. In *Proc. Int. Conf. on Database Theory*, 1999.
- [23] G. Kuper, L. Libkin, and J. Paredarns, editors. *Constraint Databases*. Springer Verlag, 2000.
- [24] B. Lin, H. Mokhtar, R. Pelaez-Aguilera, and J. Su. Querying moving objects with uncertainty. In *IEEE semiannual Vehicular Technology Conference*, 2003.
- [25] H. Mokhtar and J. Su. Universal trajectory queries for moving object databases. In *mdm*, pages 133–144, Berkeley California USA, Jan. 2004.
- [26] H. Mokhtar, J. Su, and O. Ibarra. On moving object queries. In *Proc. ACM Symp. on Principles of Database Systems*, 2002.
- [27] C. H. Papadimitriou, D. Suciuc, and V. Vianu. Topological queries in spatial databases. In *Proc. ACM Symp. on Principles of Database Systems*, 1996.
- [28] D. Pfoser and C. Jensen. Capturing the uncertainty of moving-object representations. In R. H. Güting, D. Papadias, and F. H. Lochovsky, editors, *Advances in Spatial Databases, 6th International Symposium, SSD'99, Hong*

Kong, China, July 20-23, 1999, *Proceedings*, volume 1651 of LNCS, pages 111–132. Springer, 1999.

- [29] D. Pfoser, C. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In *Proc. Int. Conf. on Very Large Data Bases*, 2000.
- [30] D. Pfoser and N. Tryfona. Capturing fuzziness and uncertainty of spatiotemporal objects. In *ADBIS*, pages 112–126, 2001.
- [31] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13:255–352, 1992.
- [32] S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2000.
- [33] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proc. Int. Conf. on Data Engineering*, 1997.
- [34] Z. Song and N. Roussopoulos. k -nearest neighbor search for moving query point. In *Proc. Int. Sym. on Spatial and Temporal Databases*, pages 79–96, 2001.
- [35] J. Su, H. Xu, and O. Ibarra. Moving objects: Logical relationships and queries. In *Proc. Int. Sym. on Spatial and Temporal Databases*, pages 3–19, 2001.
- [36] Y. Tao and D. Papadias. MV3R-tree: A spatio-temporal access method for timestamp and interval queries. In *Proc. Int. Conf. on Very Large Data Bases*, 2001.
- [37] G. Trajcevski, O. Wolfson, F. Zhang, and S. Chamberlain. The geometry of uncertainty in moving objects databases. In *Advances in Database Technology - EDBT 2002, 8th Int. Conf. on Extending Database Technology*, pages 233–250. Springer, March 2002.
- [38] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: issues and solutions. In *Proc. Int. Conf. on Statistical and Scientific Database Management*, pages 111–122, 1998.