Trajectory databases: data models, uncertainty and complete query languages

Bart Kuijpers and Walied Othman

Theoretical Computer Science Group
Hasselt University & Transnational University of Limburg, Belgium

Abstract. Moving objects produce trajectories. We describe a data model for trajectories and trajectory samples and an efficient way of modeling uncertainty via beads for trajectory samples. We study transformations for which important physical properties of trajectories, such as speed, are invariant. We also determine which transformations preserve beads. We give conceptually easy first-order complete query languages and computationally complete query languages for trajectory databases, which allow to talk directly about speed and beads. The queries expressible in these languages are invariant under speed- and bead-preserving transformations.

1 Introduction and summary

The research on spatial databases, which started in the 1980s from work in geographic information systems, was extended in the second half of the 1990s to deal with spatio-temporal data. One particular line of research in this field, started by Wolfson, concentrated on *moving object databases* (MODs) [4, 12], a field in which several data models and query languages have been proposed to deal with moving objects whose position is recorded at, not always regular, moments in time. Some of these models are geared towards handling uncertainty that may come from various sources (measurements of locations, interpolation, ...) and often ad-hoc query formalisms have been proposed [11]. For an overview of models and techniques for MODs, we refer to the book by Güting and Schneider [4].

In this paper, we focus on the trajectories that are produced by moving objects and on managing and querying them in a database. We therefore think it is more appropriate to talk about *trajectory databases*, rather than to refer to the moving objects that produce these trajectories. We give a data model for trajectory data, an efficient way of modeling uncertainty, we study transformations for which important physical properties of trajectories are invariant and we give first-order complete and computationally complete query languages for queries invariant under these transformations.

We propose two types of trajectory data, namely *trajectories*, which are curves in the plane (rationally parameterized by time) and *trajectory samples*, which are well-known in MODs, namely finite sequences of time-space points. A trajectory database contains a finite number of labeled trajectories or trajectory samples. There are various ways to reconstruct trajectories from trajectory

samples, of which linear interpolation is the most popular in the literature [4]. However, linear interpolation relies on the (rather unrealistic) assumption that between sample points, a moving object moves at constant minimal speed. It is more realistic to assume that moving objects have some physically determined speed bounds. Given such upper bounds, an uncertainty model has been proposed which constructs *beads* between two consecutive time-space points in a trajectory sample. Basic properties of this model were discussed a few years ago by Egenhofer et al. [1, 7] and Pfoser et al. [9], but beads were already known in the time-geography of Hägerstrand in the 1970s [6]. A bead is the intersection of two cones in the time-space space and all possible trajectories of the moving object between the two consecutive time-space points, given the speed bound, are located within the bead. Beads manage uncertainty more efficiently than other approaches based on cylinders [12] (by a factor of 3).

Speed is not only important in obtaining good uncertainty models, but also many relevant queries on trajectory data involve physical properties of trajectories of which speed is the most important. Geerts proposed a model which works explicitely with the equations of motion of the moving objects, rather than with samples of trajectories, and in which the velocity of a moving object is directly available and used [3]. If we are interested in querying about speed, it is important to know which transformations of the time-space space preserve the speed of a moving object. We characterize this group $\mathcal{V}$ of transformations as the combinations of affinities of time with orthogonal transformations of space composed with a spatial scaling (that uses the same scale factor as the temporal affinity) and translations. In [2], transformations that leave the velocity vector invariant were discussed, but starting from spatial transformation that are a function of time alone. Our result holds in general. We also show that the group $\mathcal{V}$ contains precisely the transformations that preserve beads. So, the queries that involve speed are invariant under transformations of $\mathcal{V}$, as are queries that speak about uncertainty in term of beads. Therefore, if we are interested in querying about speed and dealing with uncertainty via beads, it is advisable to use a query language that expresses queries invariant under transformations of $\mathcal{V}$. Beads have never before been considered in the context of query languages.

As a starting point to query trajectory (sample) databases, we take a two-sorted logic based on first-order logic extended with polynomial constraints in which we have trajectory label variables and real variables. This logic has been studied well in the context of constraint databases [8] and also allows the expression of speed and beads. We remark that the $\mathcal{V}$-invariant queries form an undecidable class, and we show that this fragment is captured by a three-sorted logic, with trajectory label variables, time-space point variables and speed variables, that uses two very simple predicates: $\mathsf{Before}(p, q)$ and $\mathsf{minSpeed}(p, q, v)$. For time-space points p and q, the former expresses that the time-component of p is smaller than that of q. The latter predicate expresses that the minimal constant speed to travel from p to q is v. This logic also allows polynomial constraints on speed variables. We show that using these two, conceptually intuitive, predicates, all the $\mathcal{V}$-invariant first-order queries can be expressed. This

language allows one to express all queries concerning speed on trajectory data and all queries concerning uncertainty in terms of beads on trajectory samples. In particular, a predicate $\mathsf{inBead}(r, p, q, v)$ can be defined in this logic, expressing that r is in the bead of p and q with maximal speed v.

We also show that a programming language, based on this three-sorted logic, in which relations can be created and which has a while-loop with first-order stop conditions, is sound and complete for the computable $\mathcal{V}$-invariant queries on trajectory (sample) databases. The proofs of these sound and completeness results are inspired by earlier work on complete languages for spatial [5] and spatio-temporal databases [2]. Compared to [2], the language we propose is far more user oriented since it is not based on geometric but speed-oriented predicates. We remark that the completeness and soundness results presented in this paper hold for arbitrary spatio-temporal data, but we present them for trajectory (sample) data. In any case, in all the presented languages it is expressible that an output relation is a trajectory (sample) relation.

This paper is organized as follows. In Section 2, we give definitions and results concerning trajectories and Section 3 deals with uncertainty via beads. Trajectory databases and queries are discussed in Section 4 and results on complete query languages are in Section 5.

2 Trajectories and trajectory samples

2.1 Definitions and basic properties

Let $\mathbf{R}$ denote the set of real numbers. We will restrict ourselves to the real plane $\mathbf{R}^2$ (although all definitions and results can be generalized to higher dimensions).

Definition 1. A *trajectory* T is the graph of a mapping $I \subseteq \mathbf{R} \to \mathbf{R}^2 : t \mapsto \alpha(t) = (\alpha_x(t), \alpha_y(t))$, i.e., $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$. The *image of the trajectory* T is the image of the mapping α that describes T. The set I is called the *time domain* of T. □

Often, in the literature, conditions are imposed on the nature of the mappings α_x and α_y. For instance, they may be assumed to be continuous, piecewise linear [4], differentiable, or C^∞ [11]. For reasons of finite representability, we assume that I is a (possibly unbounded) interval and that α_x and α_y are continuous semi-algebraic functions (i.e., they are given by a combination of polynomial inequalities in x and t and y and t respectively). For example, the set $\{(t, \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}) \mid 0 \leq t \leq 1\}$ describes a trajectory on a quarter of a circle. In this example, α_x is given by the formula $x(1+t^2) = 1 - t^2 \wedge 0 \leq t \leq 1$.

Definition 2. A *trajectory sample* is a list $\langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\rangle$, with $x_i, y_i, t_i \in \mathbf{R}$ for $i = 0, ..., N$ and $t_0 < t_1 < \cdots < t_N$. □

For the sake of finite representability, we may assume that the time-space points (t_i, x_i, y_i), have rational coordinates. This will be the case in practice, since these points are typically the result of observations.

A classical model to reconstruct a trajectory from a sample is the *linear-interpolation model* [4], where the unique trajectory, that contains the sample and that is obtained by assuming that the trajectory is run through at constant lowest speed between any two consecutive sample points, is constructed. For a sample $S = \langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N) \rangle$, the trajectory $LIT(S) := \bigcup_{i=0}^{N-1} \{(t, \frac{(t_{i+1}-t)x_i + (t-t_i)x_{i+1}}{t_{i+1}-t_i}, \frac{(t_{i+1}-t)y_i + (t-t_i)y_{i+1}}{t_{i+1}-t_i}) \mid t_i \leq t \leq t_{i+1})\}$ is called the *linear-interpolation trajectory* of S.

We now define the speed of a trajectory.

Definition 3. Let $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$ be a trajectory. If α_x and α_y are differentiable in $t_0 \in I$, then the *velocity vector of T in t_0* is defined as $(1, \frac{\mathrm{d}\alpha_x(t_0)}{\mathrm{d}t}, \frac{\mathrm{d}\alpha_y(t_0)}{\mathrm{d}t})$ and the length of the projection of this vector on the (x, y)-plane is called the *speed of T in t_0*. □

Let $S = \langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N) \rangle$ be a sample. Then for any t, with $t_i < t < t_{i+1}$, the velocity vector of $LIT(S)$ in t is $(1, \frac{x_{i+1}-x_i}{t_{i+1}-t_i}, \frac{y_{i+1}-y_i}{t_{i+1}-t_i})$ and the corresponding speed is the minimal speed at which this distance between (x_i, y_i) and (x_{i+1}, y_{i+1}) can be covered. At the moments $t_0, t_1, ..., t_N$ the velocity vector and speed of $LIT(S)$ may not be defined.

2.2 Transformations of trajectories

Now, we study transformations of trajectories under mappings $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t, x, y), f_x(t, x, y), f_y(t, x, y))$. We assume that f preserves the temporal order of events (for a technical definition we refer to [2]). It has been shown that this is equivalent to the assumption that f_t is a monotone increasing function of time alone, i.e., that $(t, x, y) \mapsto f_t(t)$ [2]. We further assume transformations to be bijective and differentiable. We remark that if f is as above and f_t is a monotone increasing function of t, then f maps trajectories to trajectories.

If $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ transforms a trajectory, then we can roughly say that $\mathrm{d}f = \begin{pmatrix} \frac{\partial f_t}{\partial t} & 0 & 0 \\ \frac{\partial f_x}{\partial t} & \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} \\ \frac{\partial f_y}{\partial t} & \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} \end{pmatrix}$, the *derived transformation of f*, transforms in each point of the trajectory the velocity vector.

Theorem 1. *A mapping $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t, x, y), f_x(t, x, y), f_y(t, x, y))$ preserves at all moments the speed of trajectories and preserves the order of events if and only if*

$$f\,(t, x, y) = a \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & a_{11} & a_{12} \\ 0 & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} b \\ b_1 \\ b_2 \end{pmatrix},$$

with $a, b, b_1, b_2 \in \mathbf{R}$, $a > 0$, and the matrix $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathbf{R}^{2\times 2}$ defining an orthogonal transformation (i.e., its inverse is its transposed). We denote the group of these transformations by $\mathcal{V}$.

Proof. Let $f : (t,x,y) \mapsto (f_t(t,x,y), f_x(t,x,y), f_y(t,x,y))$ be a transformation. If f preserves the order of events, then everywhere $\frac{\partial f_t}{\partial x} = 0$, $\frac{\partial f_t}{\partial y} = 0$ and $\frac{\partial f_t}{\partial t} > 0$ [2], which means that f_t is a reparameterization of time, i.e., $(t,x,y) \mapsto f_t(t)$.

Consider a trajectory $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$. The trajectory T will be transformed to a trajectory $f(T)$ given by $\beta : \mathbf{R} \to \mathbf{R} \times \mathbf{R}^2 : \tau \mapsto (\tau, \beta_x(\tau), \beta_y(\tau))$. Since f_t is a reparameterization of time, we can write $\tau = f_t(t)$ and $t = f_t^{-1}(\tau)$. The mapping f transforms $(t, \alpha_x(t), \alpha_y(t))$ into $f(T)$ which is $(f_t(t), f_x(t, \alpha_x(t), \alpha_y(t)), f_y(t, \alpha_x(t), \alpha_y(t)))$ and which can be written as $(\tau, f_x(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau)), \alpha_y(f_t^{-1}(\tau))), f_y(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau)), \alpha_y(f_t^{-1}(\tau))))$. This trajectory is given as β (depending on the parameter τ).

We assume that f preserves, at all moments in time, the speed of trajectories, which means that the length of $(1, \partial\alpha_x(t)/\partial t, \partial\alpha_y(t)/\partial t)$ equals that of $(1, \partial\beta_x(\tau)/\partial\tau, \partial\beta_y(\tau)/\partial\tau)$. Since $(f \circ \alpha)'(t)$ and $\frac{\partial\beta(\tau)}{\partial t}$ have to be equal, and since $(f \circ \alpha)'(t) = df_{\alpha(t)} \circ \alpha'(t)$ and $\frac{\partial\beta(\tau)}{\partial t} = \beta'(\tau) \cdot \frac{\partial\tau(t)}{\partial t} = \beta'(\tau) \cdot f_t'(t)$, we have $df_{\alpha(t)} \circ \alpha'(t) = \beta'(\tau) \cdot f_t'(t)$ which means $\left(\frac{1}{f_t'(t)} \cdot df_{\alpha(t)}\right) \circ \alpha'(t) = \beta'(\tau)$ and that $\frac{1}{f_t'(t)} \cdot df_{(t,x,y)}$ must be an isometry of $\mathbf{R} \times \mathbf{R}^2$ for each (t,x,y).

Let A be the matrix associated to the linear mapping $\frac{1}{f_t'(t)} \cdot df_{(t,x,y)}$. Since this linear transformation must be orthogonal, we have that $A \cdot A^T = A^T \cdot A = I$ and $\det(A) = \pm 1$. These conditions lead to the following equations. Firstly, $(\frac{\partial f_t}{\partial t}\frac{\partial f_x}{\partial t})/(f_t'(t))^2 = 0$, which means $\frac{\partial f_x}{\partial t} = 0$, because $\frac{\partial f_t}{\partial t} > 0$. Similarly, we have that $\frac{\partial f_y}{\partial t} = 0$. Secondly, $(\partial f_x/\partial x)^2 + (\partial f_x/\partial y)^2 = (f_t'(t))^2$. We remark that the right-hand side is time-dependent and the left-hand side isn't, and vice versa the left-hand side is dependent on only spatial coordinates and the right-hand side isn't, which means both sides must be constant. This implies that $f_t(t) = at + b$ where $a > 0$ since f_t is assumed to be an increasing function. The condition $(\frac{\partial f_x}{\partial x})^2 + (\frac{\partial f_x}{\partial y})^2 = a^2$ is known as a *differential equation of light rays* [10], and has the solution $f_x(x,y) = a_{11}x + a_{12}y + b_1$, where $a_{11}^2 + a_{12}^2 = a^2$ and where b_1 is arbitrary. Completely analogue, we obtain $f_y(x,y) = a_{21}x + a_{22}y + b_2$ where $a_{21}^2 + a_{22}^2 = a^2$ and where b_2 is arbitrary.

Thirdly, $\left(\frac{\partial f_x}{\partial x}\frac{\partial f_y}{\partial x} + \frac{\partial f_y}{\partial y}\frac{\partial f_x}{\partial y}\right) / (f_t'(t))^2 = 0$. And finally, $\det(A) = \pm 1$ gives $a_{11}a_{22} - a_{12}a_{21} = \pm 1$. If we write $a'_{ij} = \frac{a_{ij}}{a}$, then we all these equations lead to the following form of f: $f(t,x,y) =$

$$a \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & a'_{11} & a'_{12} \\ 0 & a'_{11} & a'_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} b \\ b_1 \\ b_2 \end{pmatrix}$$

where $a > 0$, and the matrix of the a'_{ij} determines an orthogonal transformation of the plane. It is also clear that transformations of the above form preserve at any moment the speed of trajectories. This completes the proof. □

Examples of speed-preserving transformations include the spatial translations and rotations, temporal translations and scalings of the time-space space.

3 Uncertainty via beads

In 1999, Pfoser et al. [9], and later Egenhofer et al. [1, 7], introduced the notion of *beads* in the moving object database literature to model uncertainty. Before Wolfson used *cylinders* to model uncertainty [4, 12]. However, cylinders give less precision (by a factor of 3, compared to beads). Let S be a sample $\langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\rangle$. Basically, the cylinder approach to managing uncertainty, depends on an uncertainty threshold value $\varepsilon > 0$ and gives a buffer of radius ε around $LIT(S)$. In the bead approach, for each pair $(t_i, x_i, y_i), (t_{i+1}, x_{i+1}, y_{i+1})$ in the sample S, their bead related does not depend on a uncertainty threshold value $\varepsilon > 0$, but rather on a maximal velocity value $v_{\max}$ of the moving object.

Definition 4. Given $(t_i, x_i, y_i), (t_{i+1}, x_{i+1}, y_{i+1})$, with $t_i < t_{i+1}$ and $v_{\max} > 0$, the *bead of* $(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$, denoted $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$, is the set $\{(t, x, y) \in \mathbf{R} \times \mathbf{R}^3 \mid (x - x_i)^2 + (y - y_i)^2 \leq (t - t_i)^2 v_{\max}^2 \wedge (x - x_{i+1})^2 + (y - y_{i+1})^2 \leq (t_{i+1} - t)^2 v_{\max}^2 \wedge t_i \leq t \leq t_{i+1}\}$. □

The bead in Figure 1 shows at each moment a *disk* or a *lens*.

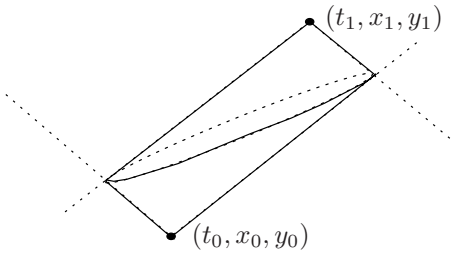

Fig. 1. An example of a bead $B(t_0, x_0, y_0, t_1, x_1, y_1, 1)$.

We remark that for a sample $S = \langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\rangle$ the set $\bigcup_{i=0}^{N-1} B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ is called the *bead chain* of S [1].

Suppose we transform a bead $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ by a function $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t), f_x(t, x, y), f_y(t, x, y))$, with f_t strictly monotone, as we have done earlier with trajectories. We ask which class of transformations map a bead to a bead. Also here we assume transformations to be bijective and differentiable.

Theorem 2. *Let* $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t), f_x(t, x, y), f_y(t, x, y))$ *be a transformation that preserves the order of events. Then for arbitrary time-space points* (t_i, x_i, y_i) *and* $(t_{i+1}, x_{i+1}, y_{i+1})$ *with* $t_i < t_{i+1}$ *and arbitrary* $v_{\max} > 0$, $f(B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max}))$ *is also a bead if and only if*

$$f\,(t, x, y) = \begin{pmatrix} a & 0 & 0 \\ 0 & ca_{11} & ca_{12} \\ 0 & ca_{21} & ca_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} b \\ b_1 \\ b_2 \end{pmatrix},$$

with $a, b, c, b_1, b_2 \in \mathbf{R}$*,* $a, c > 0$*, and the matrix of the* a_{ij} *defining an orthogonal transformation. Furthermore, if these conditions are satisfied, then* $f(B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})) = B(f(t_i, x_i, y_i), f(t_{i+1}, x_{i+1}, y_{i+1}), \frac{cv_{\max}}{a})$.

Proof. Let f be a transformation of $\mathbf{R} \times \mathbf{R}^2$ that preserves the order of events. Suppose that for any bead $B = B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$, $f(B)$ is a bead.

Let us first consider the special case, $v_{max} = \frac{d((x_i,y_i),(x_{i+1},y_{i+1}))}{(t_{i+1}-t_i)}$ (this means that the maximal speed is also the minimal speed). Then the bead B is the straight line segment between (t_i, x_i, y_i) and $(t_{i+1}, x_{i+1}, y_{i+1})$ in the (t, x, y)-space. This segment is not parallel to the (x, y)-plane (like all beads that are lines). Since B is one-dimensional and since $f(B)$ is assumed to be a bead and since $f(B)$ at any moment consists of one point also $f(B)$ must be a straight line segment not parallel to the (x, y)-plane in the (t, x, y)-space. We can conclude that f maps line segments not parallel to the (x, y)-plane to line segments not parallel to the (x, y)-plane.

Secondly, let us consider a bead B with $(x_i, y_i) = (x_{i+1}, y_{i+1})$ and $v_{\max} > 0$. This bead consists of a cone between t_i and $(t_i + t_{i+1})/2$ with top (t_i, x_i, y_i) and base the disk $D = \{((t_i+t_{i+1})/2, x, y) \mid (x-x_i)^2+(y-y_i)^2 \leq v_{\max}^2((t_{i+1}-t_i)/2)^2\}$ and a cone between $(t_i+t_{i+1})/2$ and t_{i+1} with top (t_{i+1}, x_i, y_i) and the same disk D as base. Consider the straight line segments emanating from the top (t_i, x_i, y_i) and ending in the central disk D. They are mapped to straight line segments in $f(B)$ (as we have argued before) that emanate from the top $f(t_i, x_i, y_i)$ of $f(B)$ and that end up in some figure $f(D)$ in the hyperplane $t = f_t((t_i + t_{i+1})/2)$. Since $f(B)$ is assumed to be a bead, the image of the bottom cone of B is again a cone, and the aforementioned figure $f(D)$ in the hyperplane $t = f_t((t_i + t_{i+1})/2)$ is also a closed disk. The same holds for the top cone of B. This half of B is mapped to a cone with top $f(t_{i+1}, x_{i+1}, y_{i+1})$ and base $f(D)$.

Therefore, $f(B)$ is the union of two cones, one with top $f(t_i, x_i, y_i)$, the other with top $f(t_{i+1}, x_i, y_i)$ and both with base $f(D)$. Since $f(B)$ is a bead that at no moment in time is a lens, it must itself be a bead with equally located tops. This means that $f_x(t_i, x_i, y_i) = f_x(t_{i+1}, x_i, y_i)$ and $f_y(t_i, x_i, y_i = f_y(t_{i+1}, x_i, y_i)$. In other words, the functions f_x and f_y are independent of t. This argument also shows that $f_t((t_i + t_{i+1})/2)$ is the middle of $f_t(t_i)$ and $f_t(t_{i+1})$. This means that for any t_i and t_{i+1}, $f_t((t_i + t_{i+1})/2) = \frac{1}{2}(f_t(t_i) + f_t(t_{i+1}))$. It is then easy to show that, $f_t(t) = at + b$ with $a > 0$.

So, we have shown that a bead-preserving transformation f is of the form $f(t, x, y) = (at + b, f_x(x, y), f_y(x, y))$. Now we determine f_x and f_y. If we restrict ourselves to a (x, y)-plane at some moment t between t_i and t_{i+1}, the bead $B = B(t_i, x_i, y_i, t_{i+1}, x_i, y_i, v_{\max})$ shows a disk. Since $f(B)$ is a bead again, it will also show a disk at $f_t(t)$. Since f_x and f_y are independent of t, they map disks on disks, hence distances between points are all scaled by a positive factor c by this transformation. To determine what f_x and f_y look like we can restrict ourselves to a mapping from $\mathbf{R}^2$ to $\mathbf{R}^2$, since f_x and f_y depend only on x and y. Consider the transformation $\tilde{f}(x, y) = (f_x(x, y), f_y(x, y))$, we know now that for all points $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{R}^2$, $\|\mathbf{x} - \mathbf{y}\| = \frac{1}{c}\|\tilde{f}(\mathbf{x}) - \tilde{f}(\mathbf{y})\|$. Now con-

sider $\hat{f} = \frac{1}{c}\tilde{f}$, this means $\|\mathbf{x} - \mathbf{y}\| = \|\hat{f}(\mathbf{x}) - \hat{f}(\mathbf{y})\|$ and thus $\hat{f}$ is an isometry. Just like before (cfr. speed preserving-transformations), we can conclude that $\tilde{f}(x, y) = (f_x(x, y), f_y(x, y))$ is a plane-similarity, i.e., composed of a linear plane isometry, a scaling and a translation.

We know that $(x' - x'_i)^2 + (y' - y'_i)^2 = c^2((x - x_i)^2 + (y - y_i)^2)$ and that $(t' - t'_i)^2 = a^2(t - t_i)^2$. That means that if B is a bead between the points (t_1, x_1, y_1) and (t_2, x_2, y_2) and speed v_{max}, then B' is a bead between the points (t'_1, x'_1, y'_1) and (t'_2, x'_2, y'_2) and speed $v'_{max} = \frac{c.v_{max}}{a}$. This has to hold for all beads, hence all v_{max} since degenerate beads must be transformed to degenerate beads. This concludes the proof since it is clear that all transformations of this form also map beads to beads. □

From this result it follows that if f maps a bead B with maximal speed $v_{\max}$ to a bead $f(B)$, the latter has maximal speed $\frac{cv_{\max}}{a}$. So, we get the following.

Corollary 1. *If $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ is a transformation that preserves the order of events, then f maps beads to beads with the same speed, if and only if, f preserves the speed of trajectories (i.e., f belongs to $\mathcal{V}$ defined in Theorem 1).* □

4 A model for trajectory databases and queries

4.1 Trajectory and trajectory sample databases and queries

We assume the existence of an infinite set $\mathsf{Labels} = \{a, b, ..., a_1, b_1, ..., a_2, b_2, ...\}$ of *trajectory labels*. We now define the notion of trajectory (sample) database.

Definition 5. A *trajectory relation* R is a finite set of tuples (a_i, T_i), $i = 1, ..., r$, where $a_i \in \mathsf{Labels}$ can appear only once and where T_i is a trajectory. Similarly, a *trajectory sample relation* R is a finite set of tuples $(a_i, t_{i,j}, x_{i,j}, y_{i,j})$, with $i = 1, ..., r$ and $j = 0, ..., N_i$, such that $a_i \in \mathsf{Labels}$ cannot appear twice in combination with the same t-value and such that $\langle (t_{i,0}, x_{i,0}, y_{i,0}), (t_{i,1}, x_{i,1}, y_{i,1}), ..., (t_{i,N_i}, x_{i,N_i}, y_{i,N_i})\rangle$ is a trajectory sample.

A *trajectory (sample) database* is a finite collection of trajectory (sample) relations. □

Without loss of generality, we will assume in the sequel that a database consists of one relation. In Section 2, we have discussed how we finitely represent trajectories and trajectory samples.

Now, we define the notion of a trajectory database query. We distinguish between trajectory database transformations and boolean trajectory queries.

Definition 6. A *(sample-)trajectory database transformation* is a partial computable function from (sample-)trajectory relations to (sample-)trajectory relations. A *boolean (sample-)trajectory database query* is a partial computable function from (sample-)trajectory relations to $\{0, 1\}$. □

When we say that a function is computable, this is with respect to some fixed encoding of the trajectory (sample) relations (e.g., rational polynomial functions represented in dense or sparse encoding of polynomials; or rational numbers represented as pairs of natural numbers in bit representation).

4.2 $\mathcal{V}$-equivalent trajectory databases and $\mathcal{V}$-invariant queries

Definition 7. Let R and S be trajectory (sample) databases. We say that R and S are $\mathcal{V}$-*equivalent*, if there is bijection $\mu : \mathsf{Labels} \to \mathsf{Labels}$ and a speed-preserving transformation $f \in \mathcal{V}$ such that $(\mu \times f)(R) = S$. □

In this paper, we are especially interested in transformations and queries that are invariant under elements of $\mathcal{V}$.

Definition 8. A trajectory (sample) database transformation Q is $\mathcal{V}$-*invariant* if for any trajectory (sample) databases S_1 and S_2 which are $\mathcal{V}$-equivalent by $\mu \times f$, also $(\mu \times f)(Q(S_1)) = Q(S_2)$ holds.

A boolean trajectory (sample) database query Q is $\mathcal{V}$-*invariant* if for any $\mathcal{V}$-equivalent trajectory (sample) databases R and S, $Q(R) = Q(S)$. □

5 Complete query languages for trajectory databases

5.1 First-order queries on trajectory (sample) databases

A first query language for trajectory (sample) databases we consider is the following extension of first-order logic over the real numbers, which we refer to as $\mathsf{FO}(+, \times, <, 0, 1, S)$.

Definition 9. The language $\mathsf{FO}(+, \times, <, 0, 1, S)$ is a two-sorted logic with *label variables* $a, b, c, ...$ (possibly with subscripts) that refer to trajectory labels and *real variables* $x, y, z, ...$ (possibly with subscripts) that refer to real numbers. The atomic formulas of $\mathsf{FO}(+, \times, <, 0, 1, S)$ are

- $P(x_1, ..., x_n) > 0$, where P is a polynomial with integer coefficients in the real variables $x_1, ..., x_n$;
- $a = b$; and
- $S(a, t, x, y)$ (S ia a 4-ary predicate).

The formulas of $\mathsf{FO}(+, \times, <, 0, 1, S)$ are built from the atomic formulas using the logical connectives $\wedge, \vee, \neg, ...$ and quantification over the two types of variables: $\exists x$, $\forall x$ and $\exists a$, $\forall a$. □

The label variables are assumed to range over the labels occurring in the input database and the real variables are assumed to range over $\mathbf{R}$. The formula $S(a, t, x, y)$ expresses that a tuple (a, t, x, y) belongs to the input database. The interpretation of the other formulas is standard. It is well-known that $\mathsf{FO}(+, \times, <, 0, 1, S)$-expressible queries can be evaluated effectively [8].

The $\mathsf{FO}(+, \times, <, 0, 1, S)$-sentence

$$\exists a \exists b (\neg(a = b) \wedge \forall t \forall x \forall y S(a, t, x, y) \leftrightarrow S(b, t, x, y)), \qquad (\dagger)$$

for example, expresses the boolean trajectory query that says that there are two identical trajectories in the input database with different labels.

The $\mathsf{FO}(+, \times, <, 0, 1, S)$-formula

$$S(a, t, x, y) \wedge t \geq 0 \qquad (*)$$

returns the subtrajectories of the input trajectories at positive time moments.

Boolean queries can be expressed by sentences in $\mathsf{FO}(+, \times, <, 0, 1, S)$ (for example, the sentence (†)). Trajectory transformations can be expressed by formulas $\varphi(a, t, x, y)$ in $\mathsf{FO}(+, \times, <, 0, 1, S)$ with four free variables (for example, the formula $(*)$). We remark that not every $\mathsf{FO}(+, \times, <, 0, 1, S)$-formula $\varphi(a, t, x, y)$ defines a trajectory relation on input a trajectory. However, it can be syntactically guaranteed that the output of such a query is a trajectory (sample), since this can be expressed in $\mathsf{FO}(+, \times, <, 0, 1, S)$. Indeed, it is expressible that a semi-algebraic set is a function and also that it is continuous. By combining a formula $\varphi(a, t, x, y)$ with a guard that expresses that the output of $\varphi(a, t, x, y)$ is a trajectory, we can determine a closed or safe fragment of $\mathsf{FO}(+, \times, <, 0, 1, S)$ for transforming trajectories.

Property 1. There is a $\mathsf{FO}(+, \times, <, 0, 1, S)$-formula that expresses that S is a trajectory (sample). □

5.2 A point-based first-order language for trajectory (sample) databases

In this section, we consider a first-order query language, $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$, for trajectory (sample) databases.

Definition 10. $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ is a three-sorted logic with *label variables* $a, b, c, ...$ (possibly with subscripts) that refer to labels of trajectories; *point variables* $p, q, r, ...$ (possibly with subscripts), that refer to time-space points (i.e., elements of $\mathbf{R} \times \mathbf{R}^2$); and *speed variables* $u, v, w, ...$ (possibly with subscripts), that refer to speed values (i.e., elements of $\mathbf{R}^+$).

The atomic formulas of $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ are

- $P(v_1, ..., v_n) > 0$, where P is a polynomial with integer coefficients in the velocity variables $v_1, ..., v_n$;
- equality for all types of variables; and
- $\tilde{S}(a, p)$ (here $\tilde{S}$ is a binary predicate);
- $\mathsf{Before}(p, q)$, $\mathsf{minSpeed}(p, q, v)$.

The formulas of $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ are built from the atomic formulas using the logical connectives $\wedge, \vee, \neg, ...$ and quantification over the three types of variables: $\exists a$, $\forall a$, $\exists p$, $\forall p$ and $\exists v$, $\forall v$. □

The label variables are assumed to range over the labels occurring in the input database, the point variables are assumed to range over the set of time-space points $\mathbf{R} \times \mathbf{R}^2$ and the velocity variables are assumed to range over the positive real numbers $\mathbf{R}^+$.

If p is a time-space point, then we denote its time-component by p_t and its spatial coordinates by p_x and p_y. The formula $S(a, p)$ expresses that a tuple (a, p_t, p_x, p_y) belongs to the input database. The atomic formula $\mathsf{Before}(p, q)$ expresses that $p_t \leq q_t$. The atomic formula $\mathsf{minSpeed}(p, q, v)$ expresses that $(p_x - q_x)^2 + (p_y - q_y)^2 = v^2(p_t - q_t)^2 \wedge (\neg q_t \leq p_t)$, in other words, that v is the minimal speed to go from the spatial projection of p to that of q in the time-interval that separates them.

For example, the $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-sentence

$$\exists a \exists b (\neg(a = b) \wedge \forall p \tilde{S}(a, p) \leftrightarrow \tilde{S}(b, p)) \qquad (\dagger')$$

equivalently expresses (†). To define equivalence of (queries expressible by) formulas in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ and $\mathsf{FO}(+, \times, <, 0, 1, S)$, we define the canonical mapping $can : (a, p) \mapsto (a, p_t, p_x, p_y)$. If $\tilde{A}$ is an instance of $\tilde{S}$, then $id \times can(\tilde{A})$ is an instance of S. We say that a formula $\tilde{\varphi}(a, p)$ in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ and a formula $\varphi(a, t, x, y)$ in $\mathsf{FO}(+, \times, <, 0, 1, S)$ express *equivalent* transformations if for any $\tilde{A}$, $id \times can(\{(a, p) \mid \tilde{A} \models \tilde{\varphi}(a, p)\}) = \{(a, t, x, y) \mid id \times can(A) \models \varphi(a, t, x, y)\}$. For boolean queries the definition is analogue.

For the formula $(*)$, there is no equivalent in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$. The reason for this is given by the following theorem in combination with the observation that the formula $(*)$ does not express a $\mathcal{V}$-invariant transformation.

Theorem 3. *A $\mathcal{V}$-invariant trajectory (sample) transformation or a boolean trajectory (sample) query is expressible in $\mathsf{FO}(+, \times, <, 0, 1, S)$ if and only if it is expressible in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$.*

Before giving the proof of Theorem 3, we introduce some more predicates on time-space points and speed values, which will come in handy later on:

- $\mathsf{inBead}(r, p, q, v)$ expresses that $r = (r_t, r_x, r_y)$ belongs to the bead $B(p_t, p_x, p_y, q_t, q_x, q_y, v)$ (assuming that $p_t \leq q_t$);
- $\mathsf{Between}^2(p, q, r)$ expresses that the three co-temporal points p, q and r are collinear and that q is between p and r;
- $\mathsf{Between}^{1+2}(p, q, r)$ expresses that the three points p, q and r are collinear and that q is between p and r;
- $\mathsf{EqDist}(p_1, q_1, p_2, q_2)$ expresses that the distance between the co-temporal points p_1 and q_1 is equal to the distance between the co-temporal points p_2 and q_2;
- $\mathsf{Perp}(p_1, q_1, p_2, q_2)$ expresses that the vectors $\overrightarrow{p_1 q_1}$ and $\overrightarrow{p_2 q_2}$ of the co-temporal points p_1, q_1, p_2 and q_2 are perpendicular.

Lemma 1. *The expressions $\mathsf{inBead}(r, p, q, v)$, $\mathsf{Between}^2(p, q, r)$, $\mathsf{Between}^{1+2}(p, q, r)$, $\mathsf{EqDist}(p_1, q_1, p_2, q_2)$, and $\mathsf{Perp}(p_1, q_1, p_2, q_2)$ can all be expressed in the logic $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed})$.* □

Proof. In the proof of Theorem 3, a key predicate to simulate addition and multiplication in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed})$ is $\mathsf{Between}^2$. Here, we only sketch how this predicate can be expressed. We omit the other expressions.

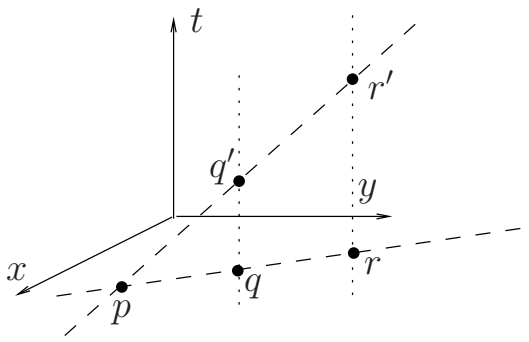

Fig. 2. The geometric construction of $\mathsf{Between}^2$.

First, we introduce predicates to denote co-spatiality and co-temporality. Equality of the spatial coordinates, $=_S (p, q)$, is expressed as $\exists v(\mathsf{minSpeed}(p, q, v) \wedge v = 0) \vee p = q$. Co-temporality of time-space points, $=_T (p, q)$, is expressed as $\mathsf{Before}(p, q) \wedge \mathsf{Before}(q, p)$. With the help of these predicates we can express $\mathsf{Between}^2(p, r, q)$ as

$$
\begin{aligned}
&=_{\mathsf{T}}(p, r) \wedge =_{\mathsf{T}}(r, q) \wedge \neg(p = r \vee r = q \vee p = q) \wedge \\
&\quad \exists r' \exists q' \exists v(=_{\mathsf{S}}(r, r') \wedge =_{\mathsf{S}}(q, q') \wedge \neg\mathsf{Before}(r', p) \wedge \neg\mathsf{Before}(q', r') \wedge \\
&\qquad \mathsf{minSpeed}(p, q', v) \wedge \mathsf{minSpeed}(p, r', v) \wedge \mathsf{minSpeed}(r', q', v)).
\end{aligned}
$$

The first line states that p, q and r should be co-temporal and distinct. Next we say that there exist points r' and q' with the same spatial coordinates as r and q respectively. The last line states that p, r' and q' are collinear and that r' is between p and q'. Therefore the projected points p, r and q are also collinear and r is between p and q. The above expression describes the geometric construction illustrated in Figure 2. □

For the purpose of the proof of Theorem 3, we need to give a more general definition of $\mathcal{V}$-invariance of $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formulas.

Definition 11. A $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formula $\varphi(a_1, ..., a_n, p_1, ..., p_m, v_1, ..., v_k)$ expresses a $\mathcal{V}$-invariant query Q if for any trajectory (sample) databases S_1 and S_2 for which there is a bijection $\mu : \mathsf{Labels} \to \mathsf{Labels}$ and a transformation $f \in \mathcal{V}$ such that $(\mu \times f)(S_1) = S_2$, also $(\mu^n \times f^m \times \mathrm{id}^k)(Q(S_1)) = Q(S_2)$. □

This definition corresponds to the definition for transformations and boolean queries (Definition 8), if we take $n = m = 1$, $k = 0$ and $n = m = k = 0$.

Proof (of Theorem 3). We have to prove soundness and completeness.
Soundness: Firstly, we show that every $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formula is equivalently expressible in $\mathsf{FO}(+, \times, <, 0, 1, S)$ and that every query expressible in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ is $\mathcal{V}$-invariant.

We assume prenex normal form for formulas, and translate the atomic formulas first. Logical connectives, and finally quantifiers, can then be added in a straightforward manner. A label variable is left unchanged. A point variable p is simulated by three real variables p_x, p_y and p_t and a speed variable v is simulated by a real variable v and when it appears it is accompanied with the restriction $v \geq 0$. An appearance of the trajectory predicate $\tilde{S}(a, p)$ is translated into $S(a, p_t, p_x, p_y)$. By switching to coordinate representations, the predicates $\mathsf{minSpeed}(p, q, v)$ and $\mathsf{Before}(p, q)$ are translated to $(p_x - q_x)^2 + (p_y - q_y)^2 = v^2 (p_t - q_t)^2 \wedge (\neg q_t \leq p_t)$ and $p_t \leq q_t$ respectively. Polynomial constraints on speed variables are literally translated (adding $v \geq 0$). Logical connectives, and finally quantifiers, can then be added in a straightforward manner ($\exists p$ is translated to $\exists p_t \exists p_x \exists p_y$).

Speed-preserving transformations preserve the order of events. That means the predicate Before is $\mathcal{V}$-invariant. The predicate $\mathsf{minSpeed}$ is also $\mathcal{V}$-invariant. If f belongs to $\mathcal{V}$, then we know from Theorem 1 that f is the composition of a scaling by a positive factor a and an orthogonal transformation and a translation. Suppose that $f(p_t, p_x, p_y) = (p'_t, p'_x, p'_y) = p'$ and $f(q_t, q_x, q_y) = (q'_t, q'_x, q'_y) = q'$. Then $(p'_x - q'_x)^2 + (p'_y - q'_y)^2 = v^2(p'_t - q'_t)^2 = a^2((p_x - q_x)^2 + (p_y - q_y)^2) = v^2 a^2 (p_t - q_t)^2$. So, $\mathsf{minSpeed}(p, q, v)$ holds if and only if $\mathsf{minSpeed}(p', q', v)$ holds.

The polynomial constraints on speed variables are by definition $\mathcal{V}$-invariant (see Definition 11). Now, it is easy to show, by induction on the syntactic structure of $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formulas that they are all $\mathcal{V}$-invariant.

Completeness: Now, we show that every $\mathcal{V}$-invariant trajectory query, expressible in $\mathsf{FO}(+, \times, <, 0, 1, S)$, can equivalently be expressed in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$. We will sketch the proof, as a rigorous proof easily becomes long and tedious. The general strategy that we outline is based on proof strategies introduced in [5] for spatial data and later developed for spatio-temporal data in [2]. Label variables are literally translated. The real variables are translated into point variables and we simulate addition and multiplication operations and order in a "computation plane". To do this we need a coordinate system for $\mathbf{R} \times \mathbf{R}^2$ that is the image of the standard coordinate system of $\mathbf{R} \times \mathbf{R}^2$ under some element of $\mathcal{V}$. Let (u_0, u_1, u_2, u_3) be such a coordinate system, meaning u_0, u_2 and u_3 are co-temporal, $\overrightarrow{u_0 u_1}$, $\overrightarrow{u_0 u_2}$ and $\overrightarrow{u_0 u_3}$ are perpendicular and have equal length and u_0 is a point Before u_1. All of this is expressible in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ with the predicates introduced in Lemma 1. The predicate $\mathsf{CoSys}(u_0, u_1, u_2, u_3)$ expresses that (u_0, u_1, u_2, u_3) is the image of the standard coordinate system under some speed-preserving transformation. Next all real variables are directly translated into point variables on the line $u_0 u_2$, the idea is to translate a real variable x to a point variable p^x with a cross ratio (u_0, u_2, p^x) equal to x. Using only $\mathsf{Between}^2$ we can express all addition and multiplication operations in the plane spanned by the co-temporal points u_0, u_2 and u_3.

At this point, we have in our translated formula too many free variables, since we translated variables, which represent coordinates, to point variables and added a coordinate system. We need to introduce new point variables and express that the translated coordinate point variables are coordinates for these new point

variables. Thus linking every triple of coordinate point variables on the line u_0u_2 with a single point variable. This can be done with a predicate $\mathsf{Coordinates}(u_0, u_1, u_2, u_3, t, x, y, u)$ which expresses that the cross ratios (u_0, u_2, t), (u_0, u_2, x) and (u_0, u_2, y) are the coordinates for the point variable u with respect to the coordinate system (u_0, u_1, u_2, u_3). This can be done using only the predicate $\mathsf{Between}^{1+2}$ as was shown in [5]. The relation S is translated in a similar straightforward manner. Finally we add existential quantifiers for all the coordinate point variables and for the points u_0, u_1, u_2 and u_3. □

As a corollary of Theorem 3 and Property 2, is the following.

Property 2. There is a $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formula that expresses that $\tilde{S}$ is a trajectory (-sample). □

5.3 Computationally complete query language for trajectory (sample) databases

In this section, we consider computationally complete query languages for trajectory (sample) databases. We start by extending the logic $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ with a sufficient supply of relation variables (of all arities), assignment statements and while-loops. Afterward, we will prove that this extended language is computationally sound and complete for $\mathcal{V}$-invariant computable queries on trajectory (sample) databases.

Definition 12. A program in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+while is a finite sequence of *assignment statements* and *while-loops*:

- An *assignment statement* is of the form

  $$\tilde{R} := \{(a_1, \ldots, a_k, p_1, \ldots, p_l, v_1, \ldots, v_m) \,|\varphi\,(a_1, \ldots, a_k, p_1, \ldots, p_l, v_1, \ldots, v_m)\};$$

  where $\tilde{R}$ is a relation variable of arity k in the label variables, arity l in the time-space point variables and arity m in the speed variables, and φ is a formula in the language $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ extended with the relation labels that were previously introduced in the program.
- A *while-loop*

  $$\textbf{while } \varphi \textbf{ do } P;$$

  contains a sentence φ in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ extended with previously introduced relation labels and a $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+while-program P (again extended with previously introduced relation labels).
- One relation variable is designated as an output relation $\tilde{R}_{out}$. The program ends once that particular relation variable has been assigned a value. □

The semantics of $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+while should be clear and is like that of $\mathsf{FO}(+, \times, <, 0, 1, S)$+while. A program defines a query on a trajectory (sample) database. Indeed, given an input relation, as soon as a value is assigned to the relation $\tilde{R}_{out}$, the program halts and returns an output; or the program

might loop forever on that input. Thus, a program defines a partial function from input to output relations. We remark that the output relation is computable from the input.

Once we have fixed a data model for trajectories or trajectory samples (see Section 2) and concrete data structures to implement the data model, we say that a partial function on trajectory (sample) databases is *computable*, if there exists a Turing machine that computes the function, given the particular data encoding and data structures (see [8] for details).

We omit the proof of the following result.

Theorem 4. $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+$\mathsf{while}$ *is sound and complete for the computable $\mathcal{V}$-invariant queries on trajectory (sample) databases.* □

Acknowledgments. This research has been partially funded by the European Union under the FP6-IST-FET programme, Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledege Discovery and Delivery, and by the Research Foundation Flanders (FWO-Vlaanderen), Research Project G.0344.05.

References

1. M. Egenhofer. Approximation of geospatial lifelines. In *SpadaGIS, Workshop on Spatial Data and Geographic Information Systsems*, 2003. Electr. proceedings, 4p.
2. F. Geerts, S. Haesevoets, and B. Kuijpers. A theory of spatio-temporal database queries. In *Database Programming Languages (DBPL'01)*, volume 2397 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002 (a full version will appear in ACM Transactions on Computational Logic).
3. Floris Geerts. Moving objects and their equations of motion. In *Constraint Databases (CDB'04)*, volume 3074 of *Lecture Notes in Computer Science*, pages 41–52. Springer, 2004.
4. R. Güting and M. Schneider. *Moving Object Databases*. Morgan Kaufmann, 2005.
5. M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometric query languages. *J. Comput. System Sci.*, 58(3):483–511, 1999.
6. T. Hägerstrand What about People in Regional Science? *Papers of the Regional Science Association* vol.24, 1970, pp.7-21.
7. K. Hornsby and M. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1–2):177–194, 2002.
8. J. Paredaens, G. Kuper, and L. Libkin, editors. *Constraint databases*. Springer-Verlag, 2000.
9. D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Advances in Spatial Databases (SSD'99)*, volume 1651 of *Lecture Notes in Computer Science*, pages 111–132, 1999.
10. A. D. Polyanin, V. F. Zaitsev, and A. Moussiaux. *Handbook of First Order Partial Differential Equations*. Taylor & Francis, 2002.
11. J. Su, H. Xu, and O. Ibarra. Moving objects: Logical relationships and queries. In *Advances in Spatial and Temporal Databases (SSTD'01)*, volume 2121 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2001.
12. O. Wolfson. Moving objects information management: The database challenge. In *Proceedings of the 5th Intl. Workshop NGITS*, pages 75–89. Springer, 2002.