Trajectory databases: data models, uncertainty and complete query languages ⋆

Bart Kuijpers and Walied Othman *

Hasselt University and 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 of the ambient space 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 uncertainty in terms of beads. The queries expressible in these languages are invariant under speed- and bead-preserving transformations.

Key words: moving objects databases, spatio-temporal databases, constraint databases, trajectories, trajectory samples, beads, uncertainty

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) [9,21], 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

⋆ An extended abstract appeared in the proceedings of the 11th International Conference on Database Theory (ICDT'07) [14].

* Corresponding author: Walied Othman, Hasselt University, Department of Mathematics, Physics and Computer Science, 3590 Diepenbeek, Belgium; phone: +32 11 26 82 95; fax: +32 11 26 82 99; email: walied.othman@uhasselt.be.

uncertainty that may come from various sources (measurements of locations, interpolation, ...) and often ad-hoc query formalisms have been proposed [20]. For an overview of models and techniques in the field of moving object databases, we refer to the recent textbook by Güting and Schneider [9].

In this paper, we focus on the trajectories that are produced by moving objects and on managing and querying them in a database. Hence, we think it is more appropriate to talk about *trajectory databases*, rather than to refer to the moving objects that produce these trajectories. We can summarize our results as follows: 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. Firstly, we have *trajectories*, which are curves in the real plane $\mathbf{R}^2$ that are rationally parameterized by time ($\mathbf{R}$ denotes the set of real numbers). Secondly, we consider *trajectory samples*, which are well-known in MODs, and which are finite sequences of time-space points (i.e., finite sequences of elements of $\mathbf{R} \times \mathbf{R}^2$). A trajectory database contains a finite number of labeled trajectories or labeled trajectory samples. There are various ways to reconstruct trajectories from trajectory samples, of which linear interpolation between consecutive sample points is the most popular in the literature [9]. 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 in the geographic information system (GIS) community by Pfoser et al. [16], Egenhofer et al. [3,12] and Miller [**?**], but beads were already known in the time-geography of Hägerstrand in the 1970s [11].

A bead is the intersection of two cones (one pointing upward and one downward) 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. The chain of beads connecting consecutive trajectory sample points is called a *lifeline necklace* [3]. Figure 1 illustrates the concepts of bead and lifeline necklace. Beads manage uncertainty more efficiently than other approaches based on cylinders [21] (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

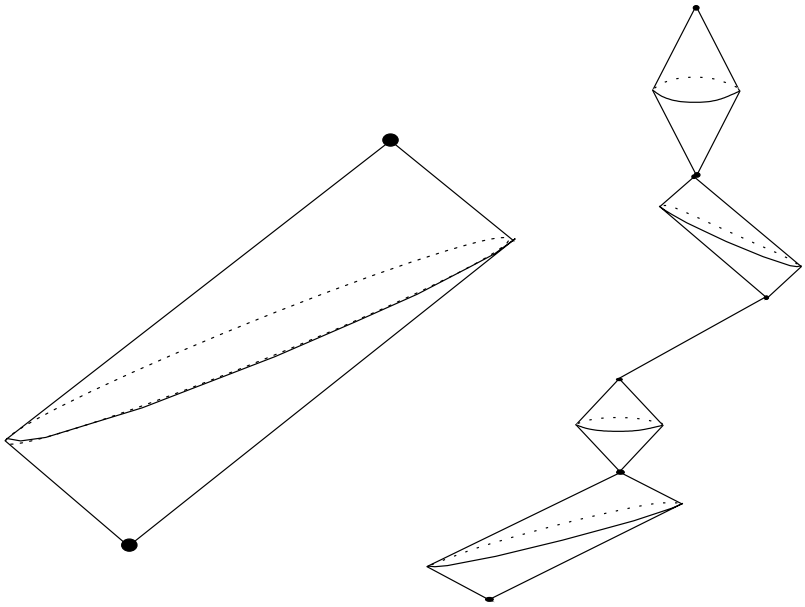

Fig. 1. An example of a bead (left) and a lifeline necklace (right).

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 [8]. 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 scalings (that uses the same scale factor as the temporal affinity) and translations. In [4], 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, for arbitrary smooth transformations of time-space space. 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 been studied before in the context of modelling uncertainty [3,12,**?**,16], but have not been considered in the context of query languages before.

As a starting point to query trajectory (sample) databases, we take a two-sorted logic based on first-order logic over the real numbers (i.e., the relational calculus extended with polynomial constraints) in which we have trajectory-label variables and real variables. First-order logic over the real numbers has been studied well in the context of constraint databases [15]. This logic is expressive enough to talk about speed and beads. We remark that the $\mathcal{V}$-invariant queries form an undecidable class, and we show that this fragment

of the above mentioned two-sorted logic 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 [10] and spatio-temporal databases [4]. Compared to [4], the language we propose is far more user oriented since it is not based on geometric but speed-related 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 for which the bead model is specifically designed. 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 the transformation group $\mathcal{V}$ and Section 3 deals with uncertainty via beads. Trajectory databases and queries are discussed in Section 4. The various query languages and completeness results are given in Section 5.

2 Trajectories and trajectory samples

2.1 Definitions and basic properties

Let $\mathbf{R}$ denote the set of real numbers. We restrict [1] ourselves to movement in the real plane $\mathbf{R}^2$. Time-space space will be denoted $\mathbf{R} \times \mathbf{R}^2$, where the first

[1] We remark that all definitions and results in this paper can be generalized to higher dimensions in a straightforward way. For ease of exposition, we restrict ourselves to two dimensions, which is the case relevant for geographic information system applications [3,12,**?**,16].

dimension represents time and the latter two represent space. Typically, we will use t as a variable that ranges over time points and x, y as variables that range over spatial coordinates. [2]

Definition 1 Let $I \subseteq \mathbf{R}$ be an interval. A *trajectory* T is the graph of a piecewise-smooth [3] (with respect to t) mapping

$$\alpha : I \subseteq \mathbf{R} \rightarrow \mathbf{R}^2 : t \mapsto \alpha(t) = (\alpha_x(t), \alpha_y(t)),$$

i.e., $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$. The set I is called the *time domain* of T. $\square$

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 piecewise linear [9], differentiable or even C^∞ [20]. For reasons of finite representability, we may, for instance, assume that I is a (possibly unbounded) interval in $\mathbf{R}$ with rational end points and that α_x and α_y are 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 the quarter of the unit circle located in the first quadrant. In this example, α_x is given by the formula $x(1+t^2) = 1 - t^2 \wedge 0 \leq t \leq 1$ and α_y is given by the formula $x(1+t^2) = 2t \wedge 0 \leq t \leq 1$.

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

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.

Definition 3 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 of a trajectory and let $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R} \times \mathbf{R}^2 \mid t \in I\}$ be a trajectory. We say that the trajectory T is *consistent* with the sample S, if $t_0, t_1, ..., t_N \in I$ and $\alpha_x(t_i) = x_i$, $\alpha_y(t_i) = y_i$ for $i = 0, ..., N$. $\square$

A classical model to reconstruct a trajectory from a sample is the *linear-interpolation model* [9], where the unique trajectory, that is consistent with 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 con-

[2] So, we have (t, x, y)-tuples in the space $\mathbf{R} \times \mathbf{R}^2$. Strictly speaking, we should write $\mathbf{R} \times \mathbf{R} \times \mathbf{R}$ or simply $\mathbf{R}^3$, but we prefer the notation $\mathbf{R} \times \mathbf{R}^2$ to stress the distinction between time and space.

[3] Smooth is here used in the terminology of differential geometry [**?**], meaning differentiable or C^1.

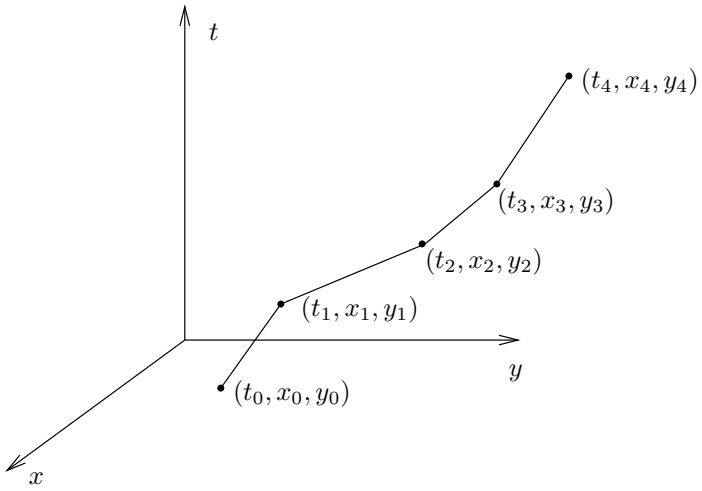

Fig. 2. The linear interpolation trajectory for the sample $\langle (t_0, x_0, y_0), (t_1 x_1, y_1), (t_2, x_2, y_2), (t_3, x_3, y_3), (t_4, x_4, y_4) \rangle$.

structed. 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. The functions describing the x- and y-coordinates are everywhere differentiable except may be at the moments $t_0, t_1, ..., t_N$.

Figure 2 gives an illustration of the linear-interpolation trajectory. Obviously, many samples may give rise to the same linear-interpolation trajectory.

2.2 Speed of a trajectory

Definition 4 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 $\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}/(t_{i+1} - t_i)$, which corresponds to 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.

2.3 Transformations of trajectories

Now, we study transformations of trajectories under bijective mappings

$$f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t, x, y), f_x(t, x, y), f_y(t, x, y)).$$

Since we are interested in transformations that preserve the speed of trajectories at all moments in time, we assume, in the spirit of differential geometry [**?**], that the transformations $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ are (globally) smooth. We will call globally smooth bijective mappings of $\mathbf{R} \times \mathbf{R}^2$ *transformations* for short.

We further assume that f preserves the uni-directional nature of time and the temporal order of events.

An *event* is a subset of $\mathbf{R} \times \mathbf{R}^2$. The projection of an event A on the time-axis is denoted by $\pi_t(A)$ and called the *time-domain* of A.

Let A and B be events. In the terminology of Allen's interval calculus [1,2], A and B are called *co-temporal* if $\pi_t(A) = \pi_t(B)$ (we denote this by $A =_t B$). Allen says A is *before* B if $t_A < t_B$ for all $t_A \in \pi_t(A)$ and all $t_B \in \pi_t(B)$ (we denote this by $A <_t B$).

Remark that $A \leq_t B := (A =_t B$ or $A <_t B)$ is a pre-order on events.

Definition 5 We say that a transformation $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ *preserves the order of events* if for all events A and B, $A =_t B$ implies $f(A) =_t f(B)$ and $A <_t B$ implies $f(A) <_t f(B)$. □

It is easy to show the following property (for a proof see [5]).

Property 1 *A transformation* $f = (f_t, f_x, f_y) : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t, x, y), f_x(t, x, y), f_y(t, x, y))$ *preserves the order of events if and only if* f_t *is a strictly monotone increasing bijection of* t *alone.* □

Assuming the above restrictions on f, from now on we shall therefore write $f_t : \mathbf{R} \to \mathbf{R} : t \mapsto f_t(t)$.

Property 2 *Let* T *be a trajectory. If* f *is as above and* f_t *is a monotone increasing function of* t*, then* $f(T)$ *is also a trajectory.*

Proof 1 (of Property 2). Let $T = \{(t, \alpha_x(t), \alpha_y(t)) \mid t \in I\}$ be a trajectory and $f = (f_t, f_x, f_y) : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ be a function with f_t a monotone

increasing function of t. First, we observe that from the fact that f is a bijective mapping, it follows that f_t must be a monotone continuous function with a continuous inverse (i.e., f_t is a homeomorphism of $\mathbf{R}$). Even more, since f is assumed to be differentiable, f_t is also differentiable and the inverse function theorem guarantees that also f_t^{-1} is differentiable. Therefore, f_t is a diffeomorphism of $\mathbf{R}$.

We have that $f(T) = \{(f_t(t), f_x(t, \alpha_x(t), \alpha_y(t)), f_x(t, \alpha_x(t), \alpha_y(t))) \mid t \in I\}$ and if we write $\tau = f_t(t)$, then we get $f(T) = \{(\tau, f_x(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau)), \alpha_y(f_t^{-1}(\tau))), f_y(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau)), \alpha_y(f_t^{-1}(\tau)))) \mid \tau \in f_t(I)\}$. Since f_t is a diffeomorphism of $\mathbf{R}$, $f_t(I)$ is also an interval in $\mathbf{R}$. It is clear that $f_x(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau)), \alpha_y(f_t^{-1}(\tau)))$ and $f_x(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau)), \alpha_y(f_t^{-1}(\tau)))$ are functions that are defined on this interval and that they are differentiable in all points except maybe in those $\tau_0 = f_t(t_0)$, with t_0 a moment in time where α_x or α_y is not differentiable. Therefore, $f(T)$ is a trajectory. $\square$

We remark that the restriction concerning finite representability of trajectories that we have given after Definition 1, might not be fulfilled for $f(T)$ for some f. For the moment we do not worry about this. For the relevant f, that we will identify in Theorem 1, this problem vanishes.

We remark that the fact that f_t is a monotone increasing function of t alone, can be expressed as by the conditions: $\frac{\partial f_t}{\partial x} = 0$, $\frac{\partial f_t}{\partial y} = 0$ and $\frac{\partial f_t}{\partial t} > 0$.

If $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ is as above, then the matrix

$$\mathrm{d}f = \begin{pmatrix} \frac{\partial f_t}{\partial t} & 0 & 0 \\ \frac{\partial f_x}{\partial t} & \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} \\ \frac{\partial f_y}{\partial t} & \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} \end{pmatrix}$$

is called the *total derivative of* f [**?**]. This is in each time-space point a linear transformation of $\mathbf{R} \times \mathbf{R}^2$ that, when applied to a trajectory, describes how the velocity vector is transformed.

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

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

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 the transformations of $\mathbf{R}\times\mathbf{R}^2$ identified in this theorem by $\mathcal{V}$.

Proof 2 (of Theorem 1). Let $f : (t, x, y) \mapsto (f_t(t, x, y), f_x(t, x, y), f_y(t, x, y))$ be a transformation as in the statement of the theorem. As remarked before, we have $\frac{\partial f_t}{\partial x} = 0$, $\frac{\partial f_t}{\partial y} = 0$ and $\frac{\partial f_t}{\partial t} > 0$, which means that f_t is a reparameterization of time and that f can be simplified to $f : (t, x, y) \longmapsto (f_t(t), f_x(t, x, y), f_y(t, x, y))$.

Consider a trajectory $T = \{(t, \alpha_x(t), \alpha_y(t)) \in \mathbf{R}\times\mathbf{R}^2 \mid t \in I\}$. For the purpose of this proof it suffices to consider trajectories for which $I = \mathbf{R}$ and for which α_x and α_y are everywhere differentiable. The trajectory T will be transformed to a trajectory $f(T)$ given by $\beta : \mathbf{R} \to \mathbf{R}\times\mathbf{R}^2 : \tau \mapsto (\tau, \beta_x(\tau), \beta_y(\tau))$, where $\tau = f_t(t)$ or $t = f_t^{-1}(\tau)$ and $\beta_x(\tau) = f_x(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau), \alpha_y(f_t^{-1}(\tau))$ and $\beta_y(\tau) = f_y(f_t^{-1}(\tau), \alpha_x(f_t^{-1}(\tau), \alpha_y(f_t^{-1}(\tau))$, as remarked in the proof of Property 2.

We assume that f preserves, at all moments in time, the speed of trajectories, which means that $\left\|\left(1, \frac{\partial\alpha_x(t)}{\partial t}, \frac{\partial\alpha_y(t)}{\partial t}\right)\right\| = \left\|\left(1, \frac{\partial\beta_x(\tau)}{\partial\tau}, \frac{\partial\beta_y(\tau)}{\partial\tau}\right)\right\|$.

Let $\bar{\alpha}$ be the mapping $t \mapsto (t, \alpha_x(t), \alpha_y(t))$ and $\bar{\beta}$ be the mapping $\tau \mapsto (\tau, \beta_x(\tau), \beta_y(\tau))$. Since $(f \circ \bar{\alpha})(t)$ is equal to $\bar{\beta}(\tau)$, we have that the derivative[4] $(f \circ \bar{\alpha})'(t)$ should be equal to $\frac{\partial\bar{\beta}(\tau)}{\partial t}$.

Since $(f \circ \bar{\alpha})'(t) = df_{\bar{\alpha}(t)} \circ \bar{\alpha}'(t)$ and $\frac{\partial\bar{\beta}(\tau)}{\partial t} = \bar{\beta}'(\tau) \cdot \frac{\partial\tau(t)}{\partial t} = \bar{\beta}'(\tau) \cdot f_t'(t)$, we have $df_{\bar{\alpha}(t)} \circ \bar{\alpha}'(t) = \bar{\beta}'(\tau) \cdot f_t'(t)$. Since f_t is strictly monotone $f_t'(t) \neq 0$ for all t and we can therefore write $\left(\frac{1}{f_t'(t)} \cdot df_{\bar{\alpha}(t)}\right) \circ \bar{\alpha}'(t) = \bar{\beta}'(\tau)$ and conclude 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 $\frac{1}{f_t'(t)} \cdot df_{(t,x,y)}$, i.e., A is

$$\frac{1}{f_t'(t)} \cdot \begin{pmatrix} \frac{\partial f_t}{\partial t} & 0 & 0 \\ \frac{\partial f_x}{\partial t} & \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} \\ \frac{\partial f_y}{\partial t} & \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} \end{pmatrix}.$$

Since this linear transformation must be orthogonal, we have that $A \cdot A^{\mathsf{T}} =$

[4] If f is a function of t alone, we write f' instead of $\frac{df}{dt}$.

$A^{\mathsf{T}} \cdot A = I$ and therefore $\det(A) = \pm 1$. 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, we have

$$\left(\frac{\partial f_x}{\partial x}\right)^2 + \left(\frac{\partial f_x}{\partial y}\right)^2 = (f_t'(t))^2.$$

We remark that the right-hand side is time-dependent and the left-hand side is not, and vice versa the left-hand side is dependent on only spatial coordinates and the right-hand side is not, 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* [17], 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:

$$f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto a \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & a'_{11} & a'_{12} \\ 0 & a'_{11} & a'_{22} \end{pmatrix} \begin{pmatrix} t \\ x \\ y \end{pmatrix} + \begin{pmatrix} b \\ b_1 \\ b_2 \end{pmatrix}$$

where $a > 0$, and $\begin{pmatrix} a'_{11} & a'_{12} \\ a'_{11} & a'_{22} \end{pmatrix}$ is 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.

The previous result generalizes a result from [4,5], where the same conclusion was derived, not starting from a general transformation $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$, but rather from time-dependent and space-independent affinities of $\mathbf{R} \times \mathbf{R}^2$.

3 Uncertainty via beads

In 1999, Pfoser *et al.* [16] introduced the notion of *beads* in the moving object database literature to model uncertainty. Beads were later studied by Egenhofer *et al.* [12,3] and Miller [**?**]. Before Wolfson used *cylinders* to model uncertainty [9,21]. Cylinders give less precision (by a factor of 3, compared to beads), however.

Let S be a sample $\langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N) \rangle$. More formally, 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 elementary geometry, we can define this set as $\{(t, x, y) \in \mathbf{R} \times \mathbf{R}^2 \mid t_0 \leq t \leq t_N \wedge \exists x' \exists y' (x', y') \in LIT(S) \wedge (x - x')^2 + (y - y')^2 \leq \varepsilon^2\}$.

In the bead approach, for each pair $(t_i, x_i, y_i), (t_{i+1}, x_{i+1}, y_{i+1})$ of consecutive sample points in S, their bead does not depend on a uncertainty threshold value $\varepsilon > 0$, but rather on a maximal speed value $v_{\max}$ of the moving object.

We now define the notion of bead technically.

Definition 6 Let $v_{\max} \in \mathbf{R}$ and $(t_i, x_i, y_i), (t_{i+1}, x_{i+1}, y_{i+1}) \in \mathbf{R} \times \mathbf{R}^2$, with $t_i < t_{i+1}$ and $v_{\max} > 0$ be given. 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 of points $(t, x, y) \in \mathbf{R} \times \mathbf{R}^2$ satisfying the following constraints:

$$\begin{cases} t_i \leq t \leq t_{i+1} \\ (x - x_i)^2 + (y - y_i)^2 \leq (t - t_i)^2 v_{\max}^2 \\ (x - x_{i+1})^2 + (y - y_{i+1})^2 \leq (t_{i+1} - t)^2 v_{\max}^2. \end{cases} \qquad \square$$

We call the set given by the constraints$t_i \leq t$ and $(x - x_i)^2 + (y - y_i)^2 \leq (t - t_i)^2 v_{\max}^2$ the *bottom cone* of the bead and the set given by the constraints $t \leq t_{i+1}$ and $(x - x_{i+1})^2 + (y - y_{i+1})^2 \leq (t_{i+1} - t)^2 v_{\max}^2$ the *top cone* of the bead. The axis of these cones is parallel to the t-axis. Clearly, the bead is the intersection of its bottom and top cone.

Figure 3 illustrates a bead with $v_{\max} = 1$. For this bead, the slope of the two cones is determined by the value of $v_{\max}$ and in this case it is 45°.

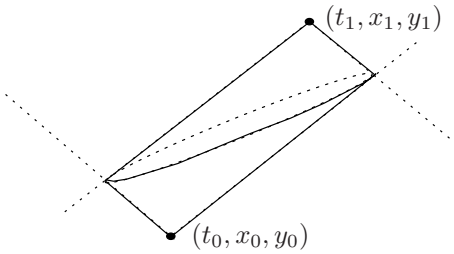

Fig. 3. An example of a bead $B(t_0, x_0, y_0, t_1, x_1, y_1, v_{\max})$ with $v_{\max} = 1$.

3.1 Basic properties of beads

We give some basic properties of beads. Let S be a sample $\langle (t_0, x_0, y_0), (t_1, x_1, y_1), ..., (t_N, x_N, y_N)\rangle$.

Definition 7 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* (or *lifeline necklaces* [3]) of S. □

∗B.vraagt :∗

[waarom staat die def. hier. moet nog meer over gezegd worden...]

The bead $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ is the intersection of two cones with slope determined by $v_{\max}$. At each moment t, with $t_i \leq t \leq t_{i+1}$, the intersection of the bead with the plane at moment t, parallel to the (x, y)-plane is a an intersection of two disks, which is a *disk* or a *lens*. This disk or lens is given by the constraints

$$\begin{cases} (x - x_i)^2 + (y - y_i)^2 \leq (t - t_i)^2 v_{\max}^2 \\ (x - x_{i+1})^2 + (y - y_{i+1})^2 \leq (t_{i+1} - t)^2 v_{\max}^2. \end{cases}$$

The case of a lens is illustrated in Figure 4.

More specifically, let $d_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$. At any moment t between t_i and $\frac{t_i+t_{i+1}}{2} - \frac{d_i}{2v_{\max}}$ the bead shows a disk with center (x_i, y_i) and radius $v_{\max}(t - t_i)$. At any moment between $\frac{t_i+t_{i+1}}{2} - \frac{d_i}{2v_{\max}}$ and $\frac{t_i+t_{i+1}}{2} + \frac{d_i}{2v_{\max}}$ the bead is a lens and between $\frac{t_i+t_{i+1}}{2} + \frac{d_i}{2v_{\max}}$ and t_{i+1} it shows a disk with center (x_{i+1}, y_{i+1}) and radius $v_{\max}(t_{i+1} - t_i)$.

There are obviously a number of special or degenerate cases that are discussed in the following property, which follows immediately from the above remarks.

Property 3 *Let $(t_i, x_i, y_i), (t_{i+1}, x_{i+1}, y_{i+1})$ be time-space points with $t_i < t_{i+1}$ and let $v_{\max} > 0$. Then we have*

(1) $d((x_i, y_i), (x_{i+1}, y_{i+1})) > v_{\max}(t_{i+1} - t_i)$ if and only if $B(t_i, x_i, y_i, t_{i+1}, x_{i+1},$

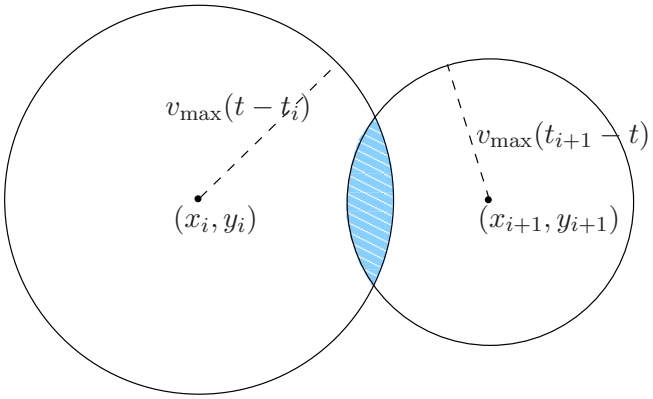

Fig. 4. An example of a lens in a bead $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ at moment t, with $t_1 \leq t \leq t_{i+1}$.

$y_{i+1}, v_{\max})$ *is empty;*

(2) *if* $v_{\max} = \frac{d((x_i,y_i),(x_{i+1},y_{i+1}))}{t_{i+1}-t_i}$, *then* $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ *is a line segment in the* (t, x, y)*-space that is not parallel to the* (x, y)*-plane;*

(3) *the bead* $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ *is at no moment between* t_i *and* t_{i+1} *a lens (i.e., it is always a disk) if and only if* $(x_i, y_i) = (x_{i+1}, y_{i+1})$ *and* $v_{\max} > 0$. *In this case,* $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ *is the union of two cones, one with top* (t_i, x_i, y_i)*, one with top* (t_{i+1}, x_i, y_i) *and both with the disk with center* (x_i, y_i) *and radius* $v_{\max}(\frac{t_{i+1}+t_i}{2})$ *at* $\frac{t_{i+1}+t_i}{2}$ *as base.* □

Case (3) of this property is illustrated by the bead on the top right in Figure 1.

The following two properties can be proven quite easily in an analytical way. We omit the proofs.

Property 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 projection of the bead* $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ *onto the* (x, y)*-plane is the area boredered by the ellipse with foci* (x_i, y_i) *and* (x_{i+1}, y_{i+1}) *and with long axis* $\frac{v_{\max}(t_{i+1}-t_i)}{2}$. *The equation of this ellipse is*

$$\frac{(2x - x_i - x_{i+1})^2}{v^2(t_{i+1} - t_i)^2} + \frac{(2y - y_i - y_{i+1})^2}{v^2(t_{i+1} - t_i)^2 - (x_i - x_{i+1})^2 - (y_i - y_{i+1})^2} = 1. \qquad \square$$

We denote the area boredered by the ellipse from the previous property by $\pi_{x,y}(B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max}))$.

Property 5 *Given* $(t_i, x_i, y_i), (t_{i+1}, x_{i+1}, y_{i+1})$, *with* $t_i < t_{i+1}$ *and* $v_{\max} > 0$, *then any trajectory from* (t_i, x_i, y_i) *to* $(t_{i+1}, x_{i+1}, y_{i+1})$ *for which the speed at any moment* $t_i \leq t \leq t_{i+1}$ *is less than* $v_{\max}$ *is located within* $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ *and the projection of such a trajectory on the* (x, y)*-plane is located within* $\pi_{x,y}(B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max}))$. *Furthermore, for any point* (t, x, y) *in* $B(t_i, x_i, y_i, t_{i+1}, x_{i+1}, y_{i+1}, v_{\max})$ *there exists a trajectory from*

(t_i, x_i, y_i) to $(t_{i+1}, x_{i+1}, y_{i+1})$ that passes through (t, x, y). $\square$

We remark that the trajectory $LIT(\langle (t_i, x_i, y_i), (t, x, y), (t_{i+1}, x_{i+1}, y_{i+1}\rangle)$ can be taken to prove the last part of the the previous property.

3.2 Transformations of beads

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 $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2 : (t, x, y) \mapsto (f_t(t), f_x(t, x, y), f_y(t, x, y))$, where we assume f to be a globally smooth bijection (called *transformation*, for short) and f_t to be strictly monotone, as we have done earlier in Section 2.3 for trajectories. We ask ourselves which class of transformations map a bead to a bead. Also here we assume transformations to be smooth and bijective. Before answering this we give the following technical lemma.

Lemma 1 *Let $f : \mathbf{R} \to \mathbf{R} : t \mapsto f(t)$ be a smooth function. If f is strictly monotone increasing and if for any $s, t \in \mathbf{R}$, we have that $2 \cdot f(\frac{s+t}{2}) = f(s) + f(t)$, then $f(t) = (f(1) - f(0)) \cdot t + f(0)$.* $\square$

Proof 3 (of Lemma 1) *Suppose f is a smooth function for which for any $s, t \in \mathbf{R}$, we have that $f(\frac{s+t}{2}) = \frac{1}{2}(f(s) + f(t))$. If we take the derivative on both sides to the variable t, we get*

$$\frac{\partial}{\partial t} f\left(\frac{s+t}{2}\right) = \frac{\partial}{\partial t} \frac{f(s) + f(t)}{2} \text{ or } f'\left(\frac{s+t}{2}\right) \cdot \frac{\partial}{\partial t}\left(\frac{t}{2}\right) = \frac{f'(t)}{2}$$

and thus $f'\left(\frac{s+t}{2}\right) = f'(t)$ for all t, which means $f'(t)$ is constant and $f(t) = at + b$. Since f is assumed to be strictly monotone increasing, we must have $a > 0$. Clearly, $f(0) = b$ and $f(1) = a + b = a + f(0)$. $\square$

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 is of the form*

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

with $a, b, c, b_1, b_2 \in \mathbf{R}$, $a, c > 0$, and the matrix $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \in \mathbf{R}^{2\times 2}$ 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 4 (of Theorem 2) Let $f : \mathbf{R} \times \mathbf{R}^2 \to \mathbf{R} \times \mathbf{R}^2$ be a transformation 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 again a bead.

Let us first consider the special case, $v_{max} = \frac{d((x_i,y_i),(x_{i+1},y_{i+1}))}{(t_{i+1}-t_i)}$ (i.e., 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 in the (t, x, y)-space. We can conclude that f maps any line segment not parallel to the (x, y)-plane to a line segment 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 $\frac{t_i+t_{i+1}}{2}$ with top (t_i, x_i, y_i) and base the disk $D = \{(\frac{t_i+t_{i+1}}{2}, x, y) \mid (x - x_i)^2 + (y - y_i)^2 \leq v_{\max}^2(\frac{t_{i+1}-t_i}{2})^2\}$ on the one hand and a cone between $\frac{t_i+t_{i+1}}{2}$ 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 some point of 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 $t = f_t\left(\frac{t_i+t_{i+1}}{2}\right)$. Since $f(B)$ is assumed to be a bead, the image of the bottom cone of B is again a cone, [waarom is dat zo?] *B.vraagt :* and the aforementioned figure $f(D)$ in the hyperplane $t = f_t\left(\frac{t_i+t_{i+1}}{2}\right)$ 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, by Property 3, 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 $f_t(\frac{t_i+t_{i+1}}{2})$ 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}, $f_t(\frac{t_i+t_{i+1}}{2}) = \frac{1}{2}(f_t(t_i) + f_t(t_{i+1}))$. By Lemma 1, $f_t(t) = at + b$ 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} $(t_i < t_{i+1})$, 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 again a bead, it will also show a disk at $f_t(t)$. Since f_x and f_y are independent of t, they map disks to 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 $\mathbf{R}^2$, $\|\mathbf{x}-\mathbf{y}\| = \frac{1}{c}\left\|\tilde{f}(\mathbf{x}) - \tilde{f}(\mathbf{y})\right\|$. Now consider $\hat{f} = \frac{1}{c}\tilde{f}$, this means $\|\mathbf{x}-\mathbf{y}\| = \left\|\hat{f}(\mathbf{x}) - \hat{f}(\mathbf{y})\right\|$ and thus $\hat{f}$ is an isometry. Just like before (cfr., the result on speed preserving-transformations), we can conclude that $\tilde{f}(x,y) = (f_x(x,y), f_y(x,y))$ is a similarity of the plane, i.e. composed of a linear isometry, a scaling and a translation of the planel.

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 $f(B) = B'$ is a bead between the points (t'_1, x'_1, y'_1) and (t'_2, x'_2, y'_2) and speed $v'_{max} = \frac{c.v_{max}}{a}$, because we know that $(x'-x'_i)^2 + (y'-y'_i)^2 = c^2\left((x-x_i)^2 + (y-y_i)^2\right)$ and that $(t'-t'_i)^2 = a^2(t-t_i)^2$.

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 results it follows that if f maps a bead B with maximal speed $v_{\max}$ to a bead $f(B)$, the latter has maximal speed $\frac{cv_{\max}}{a}$. Therefore, we can conclude 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 sample-trajectory 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 8 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 $\{R_1, R_2, ..., R_M\}$ of trajectory (sample) relations. $\square$

Without loss of generality, we will assume in the sequel that a database consists of one relation and when we refer to the database we will refer to its relation.

In Section 2, we have discussed how we finitely represent trajectories and trajectory samples.

The following is an example of a trajectory relation containing three trajectories.

label	geometry(x)	geometry(y)	Time domain
a	$\frac{1}{t^2+1}$	$\frac{t}{t^2+1}$	$[0,1]$
b	$\frac{1-t^2}{t^2+1}$	$\frac{2t}{t^2+1}$	$[0,1]$
c	1	2	$[1,2]$

The second trajectory, b, describes a movement on a segment of a circle. The third, c, is a stationary trajectory in the point $(1,2)$.

The following is an example of trajectory sample relation.

label	t	x	y
a	0	0	0
a	1	0	1
a	2	0	2
a	3	0	3
b	0	0	0
b	1	1	0
b	2	2	0
b	3	3	0
c	0	0	0
c	1	0	0
c	2	0	0
c	3	0	0

It contains samples of objects with labels a, b and c between time moments 0 and 3. The object a is moving over the y-axis at uniform speed, the object b is moving over the x-axis at uniform speed and the object c that remains stationary in the origin.

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

Definition 9 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 $\{0, 1\}$. □

We could consider other types of queries, but without loss of generality we can restrict ourselves to these. [is dat zo?????] *B.vraagt :*

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 10 Let R and S be trajectory (sample) databases. We say that R and S are $\mathcal{V}$-*equivalent*, if there is bijection μ : Labels $\rightarrow$ 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 11 A trajectory (sample) database transformation Q is $\mathcal{V}$-*invariant* if for any trajectory (sample) databases R and S which are $\mathcal{V}$-equivalent, i.e., for which there is a bijection μ : Labels $\rightarrow$ Labels and a transformation $f \in \mathcal{V}$ such that $(\mu \times f)(R) = S$, also $(\mu \times f)(Q(R)) = Q(S)$.

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

5 Complete query languages for trajectory databases

5.1 First-order queries on trajectory (sample) databases

A first query language for trajectory (sample) databases that 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 12 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 $\exists a$, $\forall a$. $\square$

For what concerns the semantics of queries, expressed by $\mathsf{FO}(+, \times, <, 0, 1, S)$-formulas, when applied to some input trajectory (sample) database, we observe the following. 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 trajectory (sample) database, i.e., to the trajectory (sample) relation. The interpretation of the other formulas is standard. The logic $\mathsf{FO}(+, \times, <, 0, 1, S)$ is a constraint database query language [15,18]. It is well-known that $\mathsf{FO}(+, \times, <, 0, 1, S)$-expressible queries can be evaluated effectively [15].

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.

As another example, the $\mathsf{FO}(+, \times, <, 0, 1, S)$-sentence

$$\exists a \exists t_1 \exists x_1 \exists y_1 \exists t_2 \exists x_2 \exists y_2 (S(a, t_1, x_1, y_1) \wedge S(a, t_2, x_2, y_2) \wedge \\ 0 < t_2 - t_1 \leq 10 \wedge (x_1 - x_2)^2 + (y_1 - y_2)^2 \geq 100^2)$$

expresses the boolean trajectory query that says that there is a trajectory that at some interval has an average speed higher than 10.

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

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

has free variables a, t, x and y and returns the subtrajectories of the input trajectories at positive time moments.

Sentences in $\mathsf{FO}(+, \times, <, 0, 1, S)$ (for example, the sentence $(\dagger)$) express Boolean queries.

Trajectory transformations can be expressed in $\mathsf{FO}(+, \times, <, 0, 1, S)$ by formulas $\varphi(a, t, x, y)$ 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. The formula

$$\exists t' \exists x' \exists y' S(a, t', x', y') \wedge t > 0 \wedge x > 0 \wedge y > 0$$

is an example of a formula that does not return a trajectory (sample).

However, it can be syntactically guaranteed that the output of such a query is a trajectory (sample), since the property of being a trajectory (sample) can be expressed in $\mathsf{FO}(+, \times, <, 0, 1, S)$.

Property 6 *There is an* $\mathsf{FO}(+, \times, <, 0, 1, S)$*-formula that expresses that a set* $\{(t, x, y) \mid \varphi(t, x, y)\}$*, with* $\varphi(t, x, y)$ *an* $\mathsf{FO}(+, \times, <, 0, 1, S)$ *formula, is a trajectory (sample).* □

Proof 5 (of Property 6) It is well known that a $\mathsf{FO}(+, \times, <, 0, 1, S)$-definable set $\{(t, x, y) \mid \varphi(t, x, y)\}$, is a semi-algebraic set [15]. It is expressible that a semi-algebraic set is a function of the form $t \mapsto (x(t), y(t))$ and also that it is piecewise smooth. Indeed, differentiability of a function in a point t_0 can be first-order expressed using the ε-δ-definition of differentiability (see for example [7,6]). Piecewise smoothness is then expressed by saying that the set of moments t where the function is not differentiable is finite. Finiteness of a semi-algebraic set can be expressed by saying that all the elements of the set are isolated points of the set [15]. Therefore, it is $\mathsf{FO}(+, \times, <, 0, 1, S)$-expressible that $\{(t, x, y) \mid \varphi(t, x, y)\}$ is a trajectory.

To express that a set $\{(t, x, y) \mid \varphi(t, x, y)\}$ is a trajectory sample, it suffices to say that this set is finite and that for each t-value, there is at most one accompanying (x, y)-value. □

By combining a formula $\varphi(a, t, x, y)$ with a guard that expresses that for every label a in the output of $\varphi(a, t, x, y)$, the corresponding (t, x, y) values

form a trajectory (sample), we can determine a *closed* or *safe* fragment of $\mathsf{FO}(+, \times, <, 0, 1, S)$ for transforming trajectories.

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 13 The language $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ 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 $\mathbf{R} \times \mathbf{R}^2$);
- *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 speed variables $v_1, ..., v_n$;
- $a = b$; and
- $\tilde{S}(a, p)$ (so, 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 $\exists v$, $\forall v$. $\square$

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 speed variables are assumed to range over the positive real numbers, i.e., over $\mathbf{R}^+$.

If p is a time-space point, then we denote its time-component by p_t and its spatial coordinates with respect to the standard coordinate system 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 (p_x, p_y) of p to the spatial projection (q_x, q_y) of q in the time-interval $[p_t, q_t]$

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 the languages $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ and $\mathsf{FO}(+, \times, <, 0, 1, S)$, we define the canonical mapping

$$can : p \mapsto (p_t, p_x, p_y).$$

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 $\tilde{A}$, the set $(id \times can)(\{(a, p) \mid \tilde{A} \models \tilde{\varphi}(a, p)\})$ is equal to the set $\{(a, t, x, y) \mid (id \times can)(A) \models \varphi(a, t, x, y)\}$. For boolean queries the definition is analogue.

For the formula $(*)$, there is no equivalent formula 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 $B(p_t, p_x, p_y, q_t, q_x, q_y, v)$, where $p = (p_t, p_x, p_y)$ and $q = (q_t, q_x, q_y)$ (assuming that $p_t \leq q_t$);
- $\mathsf{Between}^2(p, r, q)$ expresses that the three co-temporal points p, q and r are collinear and that r is strictly between p and q;
- $\mathsf{Between}^{1+2}(p, r, q)$ expresses that the three points p, q and r are collinear and that r is strictly between p and q;
- $\mathsf{EqDist}(p_1, q_1, p_2, q_2)$ expresses that the distance between the co-temporal points p_1 and q_1 is equal to the the distance between the co-temporal points p_2 and q_2;
- $\mathsf{Middle}(p, r, q)$ expresses that $\mathsf{Between}^2(p, r, q)$ and that r lies in the middle between p and q;
- $\mathsf{Perp}(p_1, q_1, p_2, q_2)$ 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.

We remark that a key predicate to simulate addition and multiplication in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed})$ is $\mathsf{Between}^2$ [10]. We now show that these predicates belong to $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed})$.

Lemma 2 *The expressions* $\mathsf{inBead}(r, p, q, v)$, $\mathsf{Between}^2(p, q, r)$, $\mathsf{Between}^{1+2}(p, q, r)$, $\mathsf{EqDist}(p_1, q_1, p_2, q_2)$, $\mathsf{Middle}(p, r, q)$ *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 6 (of Lemma 2) First, we introduce some abbreviations, namely predicates to denote co-spatiality and co-temporality:

- equality of the spatial coordinates, denoted $=_S$ (p, q), is expressed in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed})$ as

$$\exists v(\mathsf{minSpeed}(p, q, v) \wedge v = 0) \vee p = q;$$

- co-temporality of time-space points, denoted $=_T$ (p, q) is expressed in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed})$ as

$$\mathsf{Before}(p, q) \wedge \mathsf{Before}(q, p).$$

Now we turn to the predicates in the statement of the lemma.

- For what concerns the predicate $\mathsf{inBead}(r, p, q, v)$, we remark that the point r lies in the bead if and only if the line connecting p and r is steeper than the edge of the bottom cone and the same is true for the line connecting q and r and the top cone. This means that an object traveling along a trajectory that is linearly interpolated between p, r and q has speed less than v. Therefore, $\mathsf{inBead}(r, p, q, v)$ is expressed as

$$\exists v_1 \left(v_1 \leq v \wedge \mathsf{minSpeed}(p, r, v_1)\right) \wedge \exists v_2 \left(v_2 \leq v \wedge \mathsf{minSpeed}(r, q, v_2)\right);$$

- The perdicate $\mathsf{Between}^2(p, r, q)$ is expressed by the formula

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

  Indeed, the first line states that the three points p, q and r are co-temporal and distinct, and that the points r' and q' have the same spatial coordinates as r and q respectively. The second line states that p, r' and q' are collinear, and therefor p, r and q are as well. The last line simply states that r is between p and q in a temporal sense.

  This expression depicts a geometric situation as illustrated in Figure 5. If there exists a line, not parallel to the spatial plane, through one of the points, in this case p, and two parallel lines (the dotted lines in the figure) through the other points, r and q, that intersect this line, then the plane, defined

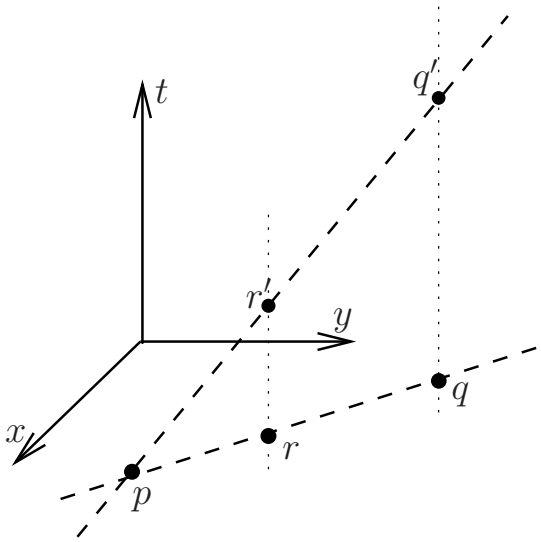

Fig. 5. The geometric construction of $\mathsf{Between}^2$.

by this line and the two parallel ones, cuts the spatial plane (containing p, r and q) in a line containing these three points p, r and q, and hence they are collinear.

To show that p, r' and q' are collinear, we assume for the sake of contradiction that they are not. That means $d(p,q') < d(p,r') + d(r',q')$ due to the triangle inequality. Thus, using Pythagoras' theorem, we have

$$\sqrt{d_S(p,q')^2 + (t_{q'} - t_p)^2} < \sqrt{d_S(p,r')^2 + (t_{r'} - t_p)^2} + \sqrt{d_S(r',q')^2 + (t_{q'} - t_{r'})^2}$$

where d_S denotes the spatial distance between the spatial projection of its arguments.

Due to the $\mathsf{minSpeed}$ relations, we have $d_S(p,q') = v(t_{q'} - t_p)$, $d_S(p,r') = v(t_{r'} - t_p)$ and $d_S(r',q') = v(t_{q'} - t_{r'})$. Substituting this in the inequality above yields to

$$\sqrt{(v^2+1)(t_{q'} - t_p)^2} < \sqrt{(v^2+1)(t_{r'} - t_p)^2} + \sqrt{(v^2+1)(t_{q'} - t_{r'})^2},$$

which holds if and only if

$$\sqrt{(v^2+1)}(t_{q'} - t_p) < \sqrt{(v^2+1)}(t_{r'} - t_p) + \sqrt{(v^2+1)}(t_{q'} - t_{r'}).$$

The latter condition holds if and only if $(t_{q'} - t_p) < (t_{r'} - t_p) + (t_{q'} - t_{r'})$ or equivalently $(t_{q'} - t_p) < (t_{q'} - t_p)$. Since this is impossible, p, r' and q' must be collinear.

- The predicate $\mathsf{Between}^{1+2}$ is a more general than $\mathsf{Between}^2$, but the expression for this predicate is quite similar. We have $\mathsf{Between}^{1+2}(p,r,q)$ if and only if

$$\mathsf{Between}^2(p,r,q) \vee (\neg(p = r \vee r = q \vee p = q) \wedge \exists v(\mathsf{minSpeed}(p,q,v) \wedge \mathsf{minSpeed}(p,r,v) \wedge \mathsf{minSpeed}(r,q,v) \wedge \neg\mathsf{Before}(r,p) \wedge \neg\mathsf{Before}(q,r))).$$

The fact that the same speed v can be used to travel from p to q; from p to r and from r to q, expresses that these three points must be collinear.

- We can write $\mathsf{EqDist}(p_1, q_1, p_2, q_2)$ as

$$\exists p_1' \exists q_1' \forall r_1 \forall r_2 \exists v (=_{\mathsf{S}}(p_1, p_1') \wedge =_{\mathsf{T}}(p_2, p_1') \wedge =_{\mathsf{S}}(q_1, q_1') \wedge =_{\mathsf{T}}(q_2, q_1') \wedge$$
$$=_{\mathsf{T}}(r_1, r_2) \wedge =_{\mathsf{S}}(r_1, q_1') \wedge =_{\mathsf{S}}(r_2, q_2) \wedge \neg(\mathsf{Before}(r_1, q_1') \vee \mathsf{Before}(r_2, q_2)) \wedge$$
$$\mathsf{minSpeed}(p_2, r_2, v) \wedge \mathsf{minSpeed}(p_1', r_1, v)).$$

  The first line states that we are projecting p_1 onto a point p_1' with the same spatial coordinates as p_1 and with the same time coordinate as p_2. The same holds for q_1 and q_2. The second line introduces any two co-temporal points r_1 and r_2 with the same spatial but greater time coordinates than q_1' and q_2, respectively. And finally the last line states that we can reach r_1 and r_2, which are spatially the same as q_1' and q_2, from p_1' and p_2 with the same speed and in the same time frame. Hence their distance must be equal.

- The expression $\mathsf{Middle}(p, r, q)$ is a little bit more specialized than $\mathsf{Between}^2(p, r, q)$ in the sense that r lies in the middle between p and q. We can express $\mathsf{Middle}(p, r, q)$ as

$$\mathsf{Between}^2(p, r, q) \wedge \forall r' \exists v (=_{\mathsf{S}}(r, r') \wedge \mathsf{Before}(r, r') \wedge$$
$$\neg =_{\mathsf{T}}(r, r') \wedge \mathsf{minSpeed}(p, r', v) \wedge \mathsf{minSpeed}(p, r', v)).$$

  This expresses that r can be reached from p and q with the same speed and in the same time-frame. This means the distance from p to r is equal to the distance from q to r.

- Finally, we have $\mathsf{Perp}(p_1, q_1, p_2, q_2)$. We can express $\mathsf{Perp}(p_1, q_1, p_2, q_2)$ as

$$\exists r \exists p_1' \exists q_1' ((\mathsf{Between}^2(p_1, q_1, r) \vee \mathsf{Between}^2(p_1, r, q_1) \vee \mathsf{Between}^2(r, p_1, q_1)) \wedge$$
$$\wedge (\mathsf{Between}^2(p_2, q_2, r) \vee \mathsf{Between}^2(p_2, r, q_2) \vee \mathsf{Between}^2(r, p_2, q_2)) \wedge$$
$$\mathsf{Middle}(p_1', r, p_1) \wedge \mathsf{Middle}(q_1', r, q_1) \wedge \mathsf{EqDist}(p_1, p_2, p_1', p_2) \wedge$$
$$\mathsf{EqDist}(q_1, p_2, q_1', p_2) \wedge \mathsf{EqDist}(p_1, q_2, p_1', q_2) \wedge \mathsf{EqDist}(q_1, q_2, q_1', q_2)).$$

  First, we state that r is the point of intersection of the straight lines going through p_1 and q_1 and through p_2 and q_2. Then we say that p_1' is a point on the line through the point p_1 and q_1 such that r is the middle of the segment bound by p_1 and p_1'. The point q_1' is defined similarly. Finally, we express that the line through p_2 and q_2 is the perpendicular bisector of the segment bound by p_1 and p_1' and the segment bound by q_1 and q_1'. Hence the vectors $\overrightarrow{p_1 q_1}$ and $\overrightarrow{p_2 q_2}$ are perpendicular. □

For the purpose of the proof of Theorem 3, we need to give a more general definition (than Definition 11), of $\mathcal{V}$-invariance of queries expressed by $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formulas.

Definition 14 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 R and S for which there is a bijection $\mu : \mathsf{Labels} \to \mathsf{Labels}$ and a transformation $f \in \mathcal{V}$ such that $(\mu \times f)(R) = S$, also $(\mu^n \times f^m \times \mathrm{id}^k)(Q(R)) = Q(S)$. $\square$

This definition corresponds to the definition for transformations and boolean queries (Definition 11), if we take $n = m = 1$, $k = 0$ and $n = m = k = 0$, respectively.

Now, we are ready for the proof of Theorem 3.

Proof 7 (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 $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-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_t, p_x and p_y that represent the real coordinates of p with respect to the standard coordinate system of $\mathbf{R} \times \mathbf{R}^2$. 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 $(p_x - q_x)^2 + (p_y - q_y)^2 = v^2 (p_t - q_t)^2$ 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. In particular, $\exists p$ is translated to $\exists p_t \exists p_x \exists p_y$.

Speed-preserving transformations preserve the order of events. That implies that the predicate Before is $\mathcal{V}$-invariant. The predicate $\mathsf{minSpeed}$ is also $\mathcal{V}$-invariant. To see this take any f that 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 $f(q_t, q_x, q_y) = (q'_t, q'_x, q'_y) = q'$. Now if $\mathsf{minSpeed}(p', q', v)$ holds, then $(p'_x - q'_x)^2 + (p'_y - q'_y)^2 = v^2(p'_t - q'_t)^2$. Because $(p'_x - q'_x)^2 + (p'_y - q'_y)^2 = a^2((p_x - q_x)^2 + (p_y - q_y)^2)$ and $v^2(p'_t - q'_t)^2 = v^2a^2(p_t - q_t)^2$, we have $a^2((p_x - q_x)^2 + (p_y - q_y)^2) = v^2a^2(p_t - q_t)^2$ or $(p_x - q_x)^2 + (p_y - q_y)^2 = v^2(p_t - q_t)^2$. In other words, $\mathsf{minSpeed}(p, q, v)$ 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 14). 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. Secondly, 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 [10] for spatial data and later developed for spatio-temporal data in [4]. The details for the current setting can be easily reconstructed using the proofs in [4,10] and the outline below.

Label variables are left unchanged in the translation. The real variables are translated into point variables and we simulate addition, multiplication and order on real values, on these point variables in a "computation plane". We explain this now in detail.

To simulate addition, multiplication and order, 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, $\overrightarrow{u_0u_1}$, $\overrightarrow{u_0u_2}$ and $\overrightarrow{u_0u_3}$ 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 2. Indeed, the formula

$$\begin{aligned}&\exists v(=_{\mathsf{T}}(u_0, u_2) \wedge =_{\mathsf{T}}(u_0, u_3) \wedge =_{\mathsf{S}}(u_0, u_1) \wedge \neg u_0 = u_2 \wedge \\ &\quad \mathsf{EqDist}(u_0, u_2, u_0, u_3) \wedge \mathsf{Perp}(u_0, u_2, u_0, u_3) \wedge v = 1 \wedge \mathsf{minSpeed}(u_2, u_1, v))\end{aligned}$$

expresses that (u_0, u_1, u_2, u_3) is such a coordinate system. When travelling from u_1 to u_2 with speed 1, the elapsed time equals the elapsed space. Therefore, the distance between u_0 and u_2 equals the distance between u_0 and u_1. Let $\mathsf{CoSys}(u_0, u_1, u_2, u_3)$ abbreviate this $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$-formula that expresses that (u_0, u_1, u_2, u_3) is the image of the standard coordinate system under some speed-preserving transformation.

As a next step in the translation, all real variables are directly translated into point variables on the line u_0u_2. The idea is to translate a real variable x to a point variable p^x, where the cross ratio (u_0, u_2, p^x) corresponds to the real value x.

It is obvious that the order relation can be simulated using $\mathsf{Between}^2$. But Tarski showed that we can construct point based predicates that simulate addition and multiplication using only $\mathsf{Between}^2$ [19] (see also [10]). Moreover, these simulations occur in the plane spanned by the co-temporal points u_0, u_2 and u_3 (hence the term *computation plane*).

At this point, we have in our translated formula too many free variables. First of all, there is the coordinate system we have chosen to represent the

time-space points in. Secondly, we have translated variables, which represent coordinates, to point variables. But we need to group triples of coordinates, all points that are on the line u_0u_2, in time-space points of which they are the coordinates. More precisely, we also introduce true time-space points. And we want to be able to express that three time-space points, located on the line through u_0 and u_2, represent the coordinates of a given time-space point. 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). The predicate $\mathsf{Coordinates}$ can be expressed using only the predicate $\mathsf{Between}^{1+2}$ as was shown in [10].

The relation S is translated in a similar straightforward manner: whenever $S(a, t, x, y)$ appears, we translate it by $\tilde{S}(a, p)$ and an expression

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 7, is the following.

Property 7 *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. For the sake of the better understanding of the proof of Theorem 5, we start with showing the computational completeness of the programming language $\mathsf{FO}(+, \times, <, 0, 1, S)$+$\mathsf{while}$, which, apart from the use of label variables, is described in Chapter 2 of [15]. We define this language here in the presence of label variables.

We assume that in the language $\mathsf{FO}(+, \times, <, 0, 1, S)$+$\mathsf{while}$, we have a sufficient supply of relation variables (of all arities). In this language we have assignment statements and while-loops.

More formally, $\mathsf{FO}(+, \times, <, 0, 1, S)$+$\mathsf{while}$ is defined as follows.

Definition 15 A *program* in $\mathsf{FO}(+, \times, <, 0, 1, S)$+$\mathsf{while}$ is a finite sequence of *assignment statements* and *while-loops*:

(1) An assignment statement is of the form

$$R := \{(a_1, ..., a_k, x_1, \ldots, x_l) \,|\varphi\,(a_1, ..., a_k, x_1, \ldots, x_l)\};$$

where R is a relation variable that is of arity k in the label variables and arity l in the real variables, and where φ is a formula in the logic $\mathsf{FO}(+, \times, <, 0, 1, S)$ extended with the relation names, previously introduced in the program.

(2) A *while-loop*

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

contains a sentence φ, in the logic $\mathsf{FO}(+, \times, <, 0, 1, S)$ extended with the relation names previously introduced in the program, and a program P.

(3) One relation variable is designated as an output relation R_{out}. The program stops and returns R_{out}, when that particular relation variable has been assigned a value.

The semantics of a $\mathsf{FO}(+, \times, <, 0, 1, S)$+while-program applied to a trajectory (sample) database is the step by step execution. The right-hand side of every assignment statement is computed by evaluating the $\mathsf{FO}(+, \times, <, 0, 1, S)$ +while-formula, extended with previously introduced relation names, on the input database. Then the result is assigned to the relation variable on the left-hand side.

The body P of a while-loop is executed as long as the sentence φ evaluates to *true*. If and when the program ends the value of R_{out} is considered the output of the program. □

First, we prove the following theorem, which is, if we assume that the label values can be encoded as (or are) natural numbers, a straightforward generalization of the case without labels (see Chapter 2 of [15]).

Theorem 4 *The language* $\mathsf{FO}(+, \times, <, 0, 1, S)$+while *is sound and complete for the computable trajectory (sample) queries (in particular, transformation or boolean query).*

Proof 8 (of Theorem 4) We assume that labels are natural numbers. Now we can consider that an input instance of $S\,(a, t, x, y)$ is given as a quantifier-free formula in the logic $\mathsf{FO}(+, \times, <, 0, 1)$. This formula will contain the symbols $(,\ )$, $\neg$, $\vee$, $+$, $\times$, 0, 1, a, t, x and y. Suppose these symbols are numbered ranging from 1 to 11. The quantifier-free formula will be a string a of symbols $\alpha_1\alpha_2 \ldots \alpha_m$. This string will be encoded as a natural number n as follows, $n = p_1^{s_{\alpha_1}} \cdots p_m^{s_{\alpha_m}}$ where p_i is the ith prime number, and s_{α_i} is the number between 1 and 11 which corresponds with the symbol α_i. This product is called the Gödel-number of a formula and is unique for every formula encoded in this way.

We now give the encoding algorithm in the language $\mathsf{FO}(+, \times, <, 0, 1, S)$+while which encodes the input relation S.

$m_S := 0$, $T := \emptyset$, $F := \emptyset$, Found :=False

while ¬Found **do**
$m_S := m_S + 1$
if m_S encodes a **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, a_1) \mid a_i \in \mathbf{R}\}$
else if m_S encodes t **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, a_2) \mid a_i \in \mathbf{R}\}$
else if m_S encodes x **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, a_3) \mid a_i \in \mathbf{R}\}$
else if m_S encodes y **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, a_4) \mid a_i \in \mathbf{R}\}$
else if m_S encodes 0 **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, 0) \mid a_i \in \mathbf{R}\}$
else if m_S encodes 1 **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, 1) \mid a_i \in \mathbf{R}\}$
else if m_S encodes $(s + t)$ **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, c + d) \mid T(enc(s), a_1, a_2, a_3, a_4, c) \wedge T(enc(t), a_1, a_2, a_3, a_4, d)\}$
else if m_S encodes $(s \times t)$ **then**
$T := T \cup \{(m_S, a_1, a_2, a_3, a_4, cd) \mid T(enc(s), a_1, a_2, a_3, a_4, c) \wedge T(enc(t), a_1, a_2, a_3, a_4, d)\}$
else if m_S encodes $(s \leq t)$ **then**
$F := F \cup \{(m_S, a_1, a_2, a_3, a_4) \mid (\exists c)(\exists d)(T(enc(s), a_1, a_2, a_3, a_4, c) \wedge T(enc(t), a_1, a_2, a_3, a_4, d) \wedge (c \leq d))\}$
else if m_S encodes $(\neg\varphi)$ **then**
$F := F \cup \{(m_S, a_1, a_2, a_3, a_4) \mid \neg F(enc(\varphi), a_1, a_2, a_3, a_4)\}$
else if m_S encodes $(\varphi \vee \psi)$ **then**
$F := F \cup \{(m_S, a_1, a_2, a_3, a_4) \mid F(enc(\varphi), a_1, a_2, a_3, a_4) \vee F(enc(\psi), a_1, a_2, a_3, a_4)\}$
end if
Found:= m_S encodes a formula and $\forall a_1 \forall a_2 \forall a_3 \forall a_4$ $(F(m_S, a_1, a_2, a_3, a_4) \iff S(a_1, a_2, a_3, a_4))$
end while
$R_{out} := \{(m_S)\}$

When the encoding program ends, T will contain all tuples $(m_S, r_1, r_2, r_3, r_4, f)$ where m_S is the encoding of a term in the variables a, t, x and y that outputs f on input (r_1, r_2, r_3, r_4).

And F contains all tuples $(m_S, r_1, r_2, r_3, r_4)$ where m_S is the encoding of a formula φ in the variables a, t, x and y where $\varphi(r_1, r_2, r_3, r_4)$ evaluates to *true*.

Note that the algorithm works because sub-formulas or sub-terms are evaluated before the formulas or terms in which they appear are evaluated (because, clearly, $enc(s) \leq enc(t)$ if s is a sub-formula or sub-term of t).

Now there exists, for each query Q a counter program M_Q, such that for each input database S on which Q is defined, and which is encoded by m_S, $M_Q(m_S)$

is the encoding (a natural number) of a quantifier-free $\mathsf{FO}(+, \times, <, 0, 1)$-formula representing $Q(S)$. If the query is a boolean query the formula will return either *true* or *false*, if the query is a trajectory transformation, then a formula in the variables a, t, x and y is returned. If Q is not defined on S then M_Q does not halt on input m_S. Furthermore, since we have full computational power over the natural numbers in $\mathsf{FO}(+, \times, <, 0, 1, S)$+while, M_Q can be simulated in $\mathsf{FO}(+, \times, <, 0, 1, S)$+while. This concludes the computation step.

Decoding can easily be done by slightly adapting the encoding program. This time the input is a number f, where f is the natural number $M_Q(m_S)$. The stop condition for the while-loop is replaced by $Found := (m = f)$ and the algorithm outputs a set $\{(r_1, r_2, r_3, r_4) \mid F(n, r_1, r_2, r_3, r_4)\}$ representing the output of the query.

$m := 0$, $T := \emptyset$, $F := \emptyset$, Found :=False
while $\neg$ Found **do**
  $m := m + 1$
  if m encodes a **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, a_1) \mid a_i \in \mathbf{R}\}$
  else if m encodes t **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, a_2) \mid a_i \in \mathbf{R}\}$
  else if m encodes x **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, a_3) \mid a_i \in \mathbf{R}\}$
  else if m encodes y **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, a_4) \mid a_i \in \mathbf{R}\}$
  else if m encodes 0 **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, 0) \mid a_i \in \mathbf{R}\}$
  else if m encodes 1 **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, 1) \mid a_i \in \mathbf{R}\}$
  else if m encodes $(s + t)$ **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, c + d) \mid T(enc(s), a_1, a_2, a_3, a_4, c) \wedge T(enc(t), a_1, a_2, a_3, a_4, d)\}$
  else if m encodes $(s \times t)$ **then**
    $T := T \cup \{(m, a_1, a_2, a_3, a_4, cd) \mid T(enc(s), a_1, a_2, a_3, a_4, c) \wedge T(enc(t), a_1, a_2, a_3, a_4, d)\}$
  else if m encodes $(s \leq t)$ **then**
    $F := F \cup \{(m, a_1, a_2, a_3, a_4) \mid (\exists c)(\exists d)(T(enc(s), a_1, a_2, a_3, a_4, c) \wedge T(enc(t), a_1, a_2, a_3, a_4, d) \wedge (c \leq d))\}$
  else if m encodes $(\neg\varphi)$ **then**
    $F := F \cup \{(m, a_1, a_2, a_3, a_4) \mid \neg F(enc(\varphi), a_1, a_2, a_3, a_4)\}$
  else if m encodes $(\varphi \vee \psi)$ **then**
    $F := F \cup \{(m, a_1, a_2, a_3, a_4) \mid F(enc(\varphi), a_1, a_2, a_3, a_4) \vee F(enc(\psi), a_1, a_2, a_3, a_4)\}$
  end if
  Found:= $m = f$
end while
$R_{out} := \{(r_1, r_2, r_3, r_4) \mid F(n, r_1, r_2, r_3, r_4)\}$

The query Q has now been effectively computed by the sequence

$$encode; compute; decode;$$

of programs. $\square$

Next, we extend 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 16 A program in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+while is a finite sequence of *assignment statements* and *while-loops*:

(1) An *assignment statement* is of the form

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

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.

(2) A *while-loop*

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

consists of a sentence φ in $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$ extended with previously introduced relation names and a $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+while-program P.

(3) 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. $\square$

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 [15] for details).

Theorem 5 $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+$\mathsf{while}$ *is sound and complete for the computable $\mathcal{V}$-invariant queries on trajectory (sample) databases.* □

Proof 9 (of Theorem 5) We need to prove that that every trajectory (sample) transformation or boolean trajectory (sample) query is expressible in $\mathsf{FO}(\mathsf{Before},\mathsf{minSpeed},\tilde{S})$+$\mathsf{while}$. To do this, we assume again that the label values are natural numbers. We consider again an instance $\tilde{S}(a, p)$ to be given as $S(a, t, x, y)$ as before, i.e., as a quantifier-free formula in $\mathsf{FO}(+, \times, <, 0, 1)$. We now present the algorithm to encode an input relation $S(a, t, x, y)$ in the language $\mathsf{FO}(\mathsf{Before}, \mathsf{minSpeed}, \tilde{S})$+$\mathsf{while}$.

To do this, we need the predicate $\mathbf{Plus}(e_0, e_2, e_3, a, b, c)$ that expresses that, relative to the computation plane (e_0, e_2, e_3) (see the proof of Theorem 3 for the notion of computation plane), c is the sum of a and b, where a, b and c are collinear with e_0e_2, thus simulating addition. The predicate $\mathbf{Times}(e_0, e_2, e_3, a, b, c)$ that expresses that, relative to the computation plane (e_0, e_2, e_3), c is the product of a and b, where a, b and c are collinear with e_0e_2, simulating multiplication. The predicate $\mathbf{Less}(e_0, e_2, e_3, a, b)$ expresses that, relative to the computation plane (e_0, e_2, e_3), a encodes a number that is smaller than the number encoded by b. All these predicates can be formulated using only the predicate $\mathsf{Between}^2$(see, e.g., [10]).

Again relation variables $\tilde{T}$ and $\tilde{F}$ are introduced. Terms are encoded in $\tilde{T}$ and formulas in $\tilde{F}$. The arity of $\tilde{T}$ is 10 and points in $\tilde{T}$ are of the form $(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, v)$, where m is a time-space point denoting a natural number that encodes a formula in the variables a, t, x and y. This formula (term), when evaluated in p_a, p_t, p_x and p_y, outputs v.

The arity of $\tilde{F}$ is 9 and points in $\tilde{F}$ are of the form $(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y)$, where m is a time-space point denoting a natural number that encodes a formula in the variables a, t, x and y. This formula, when evaluated in p_a, p_t, p_x and p_y, gives *true*.

In all the above tuples p_t, p_x and p_y are collinear with e_0 and e_2 and p_a is collinear with e_0 and e_1.

Aside from this, we also need to define the notion of $\mathcal{V}$-canonization and the $\mathcal{V}$-type of trajectory (sample) databases. As it has become clear from the previous proof, we can induce an order on the symbols used in $\mathsf{FO}(+, \times, <, 0, 1, S)$-formulas and thus induce an order on the formulas themselves using the numbers that encode them.

Definition 17 The *$\mathcal{V}$-canonization* of a trajectory (sample) database instance D (of S), denoted by $\mathbf{Canon}_{\mathcal{V}}(D)$, is the trajectory (sample) database D' that is $\mathcal{V}$-equivalent to D and is represented by a quantifier-free $\mathsf{FO}(+, \times, <, 0, 1)$-formula $\varphi_{Canon_{\mathcal{V}}(D)}$ that occurs first among all encodings of trajectory (sample)

databases that are $\mathcal{V}$-equivalent to D.

The $\mathcal{V}$-*type* of a trajectory (sample) database D, denoted by $Type_{\mathcal{V}}(D)$, is the set

$$Type_{\mathcal{V}}(D) = \{g \in \mathcal{V} \mid g(D) = \mathbf{Canon}_{\mathcal{V}}(D)\}.$$

□

We now present the encoding algorithm.

$m := 0$, $\tilde{T} := \emptyset$, $\tilde{F} := \emptyset$, Found:=False
while ¬Found **do**
  $m := m + 1$
  if m encodes a **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, p_a) \mid p_a \in e_0e_1 \text{ and } p_t, p_x, p_y \in e_0e_2\}$
  else if m encodes t **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, p_t) \mid p_a \in e_0e_1 \text{ and } p_t, p_x, p_y \in e_0e_2\}$
  else if m encodes x **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, p_x) \mid p_a \in e_0e_1 \text{ and } p_t, p_x, p_y \in e_0e_2\}$
  else if m encodes y **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, p_y) \mid p_a \in e_0e_1 \text{ and } p_t, p_x, p_y \in e_0e_2\}$
  else if m encodes 0 **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, e_0) \mid p_a \in e_0e_1 \text{ and } p_t, p_x, p_y \in e_0e_2\}$
  else if m encodes 1 **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, e_2) \mid p_a \in e_0e_1 \text{ and } p_t, p_x, p_y \in e_0e_2\}$
  else if m encodes $(s + t)$ **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, p_e) \mid \tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, p_t, p_x, p_y, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, p_t, p_x, p_y, p_d) \wedge \mathbf{Plus}(e_0, e_2, e_3, p_c, p_d, p_e)\}$
  else if m encodes $(s \times t)$ **then**
    $\tilde{T} := \tilde{T} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y, p_e) \mid \tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, p_t, p_x, p_y, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, p_t, p_x, p_y, p_d) \wedge \mathbf{Times}(e_0, e_2, e_3, p_c, p_d, p_e)\}$
  else if m encodes $(s \leq t)$ **then**
    $\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y) \mid (\exists p_c)(\exists p_d)(\tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, p_t, p_x, p_y, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, p_t, p_x, p_y, p_d) \wedge \mathbf{Less}(e_0, e_2, e_3, p_c, p_d)\}$
  else if m encodes $(s = t)$ **then**
    $\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y) \mid (\exists p_c)(\exists p_d)(\tilde{T}(e_0, e_1, e_2, e_3, enc(s), p_a, p_t, p_x, p_y, p_c) \wedge \tilde{T}(e_0, e_1, e_2, e_3, enc(t), p_a, p_t, p_x, p_y, p_d) \wedge (p_c = p_d)\}$
  else if m encodes $(\neg\varphi)$ **then**
    $\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y) \mid \neg\tilde{F}(e_0, e_1, e_2, e_3, enc(\varphi), p_a, p_t, p_x, p_y)\}$
  else if m encodes $(\varphi \vee \psi)$ **then**
    $\tilde{F} := \tilde{F} \cup \{(e_0, e_1, e_2, e_3, m, p_a, p_t, p_x, p_y) \mid \tilde{F}(e_0, e_1, e_2, e_3, enc(\varphi), p_a, p_t, p_x, p_y) \vee \tilde{F}(e_0, e_1, e_2, e_3, enc(\psi), p_a, p_t, p_x, p_y)\}$
  end if
  Found:= m encodes the formula $\mathbf{Canon}_{\mathcal{V}}(S)$
end while
$n_{Canon_{\mathcal{V}}(S)} := m$
$Type_{\mathcal{V}} := \{g \in \mathcal{V} \mid g(S) = \mathbf{Canon}_{\mathcal{V}}(S)\}$

As shown in [10],"m encodes the formula $\mathbf{Canon}_{\mathcal{V}}(S\,)$" can easily be checked since $\mathbf{Canon}_{\mathcal{V}}(S\,)$ can be computed because $\mathcal{V}$ is a semi-algebraic transformation group. The following formula in one free variable, the variable m which is the encoding of a formula, demonstrates how this can be done:

$$\begin{gathered}\forall u_0\forall u_1\forall u_2\forall u_3\exists a_{00}\exists a_{11}\exists a_{12}\exists a_{21}\exists a_{22}\exists b_0\exists b_1\exists b_2(\mathsf{CoSys}(u_0,u_1,u_2,u_3)\wedge\\ a_{00}>0\wedge a_{00}a_{00}=(a_{11}a_{22}-a_{12}a_{21})(a_{11}a_{22}-a_{12}a_{21})\wedge\\ \forall p_a\forall p_{t'}\forall p_{x'}\forall p_{y'}(\tilde{F}(u_0,u_1,u_2,u_3,m,p_a,p_{t'},p_{x'},p_{y'})\Leftrightarrow\\ \exists p\exists p_t\exists p_x\exists p_y(\tilde{S}(p_a,p)\wedge\mathsf{Coordinates}(u_0,u_1,u_2,u_3,p,p_t,p_x,p_y)\wedge\\ p_{t'}=a_{00}p_t+b_0\wedge p_{x'}=a_{11}p_x+a_{12}p_y+b_1\wedge p_{y'}=a_{21}p_x+a_{22}p_y+b_2))).\end{gathered}$$

We note that all variables are time-space points. For the sake of clarity note that the predicates, denoting that some time-space points play the role of a real number, have been omitted here. We have also used abbreviations for addition, multiplication and substraction. The above formula states that given a natural number m, then for all coordinate systems the following needs to be true: all points (p_a,p') for which the formula encoded by m gives true, are the image of a point (p_a,p) under a transformation in $\mathcal{V}$ and (p_a,p) is such that $\tilde{S}(p_a,p)$ is true. And vice versa. This transformation maps (t,x,y) to

$$\begin{pmatrix}a_{00}&0&0\\0&a_{11}&a_{12}\\0&a_{21}&a_{22}\end{pmatrix}\begin{pmatrix}t\\x\\y\end{pmatrix}+\begin{pmatrix}b_0\\b_1\\b_2\end{pmatrix},$$

where $a_{00},a_{11},a_{12},a_{21},a_{22},b_0,b_1,b_2$ are constrained as in the formula above, clearly making this transformation belong to $\mathcal{V}$.

Also mentioned in [10], $Type_{\mathcal{V}}$ can be computed by using a similar formula but without the quantification of the variables used to parameterize the transformation and coordinate system, i.e. without $\exists a_{00}\exists a_{11}\exists a_{12}\exists a_{21}\exists a_{22}\exists b_0\exists b_1\exists b_2$. This concludes the encoding step.

The computation step can be carried out as outlined in the proof of Theorem 4. Since the predicate $\mathsf{Between}^2$ implicitly belongs to $\mathsf{FO}(\mathsf{Before},\mathsf{minSpeed},\tilde{S})$, we can simulate all operations on natural numbers in that language extended with a while-loop, which means we can simulate a counter program. [dat moet toch "extended met while loop" zijn, nietwaar?] *VRAAG*

More specifically, there exists, for each query Q a counter program M_Q, such that for each database D on which Q is defined $M_Q(n_{Canon_{\mathcal{V}}(D\,)})$ is the encoding (a time-space point encoding a natural number) of a quantifier-free

formula representing $Q(\mathbf{Canon}_{\mathcal{V}}(D))$. If Q is not defined on D then M_Q does not halt. Furthermore, since we have full computational power over the natural numbers in FO(Before,minSpeed,$\tilde{S}$)+while, M_Q can be simulated in this programming language. This concludes the computation step.

Decoding requires a bit more work, since we wish to output $Q(D)$ and not $Q(\mathbf{Canon}_{\mathcal{V}}(D))$. Here is where we need $Type_{\mathcal{V}}(D)$ which was computed during the encoding step. Since Q is a $\mathcal{V}$-invariant query, we have, for all $g \in Type_{\mathcal{V}}(D)$ that $Q(\mathbf{Canon}_{\mathcal{V}}(D)) = Q(g(D)) = g(Q(D))$. The decoding can effectively be done by again slightly modifying the encoding algorithm. The stop condition for the while loop becomes $Found := m = M_Q(n_{Canon_{\mathcal{V}}(D)})$. On exiting the while loop we output the set $F(e_0, e_1, e_2, e_3, M_Q(n_{Canon_{\mathcal{V}}(D)}), a, p_t, p_x, p_y)$.

After exiting the while loop we execute one more statement, namely the computation of $Q(D)$. The latter is a set which contains all points (a, p), where a is a label and p a time-space point, for which there exists a transformation $g \in Type_{\mathcal{V}}(D)$ such that $g(p)$ has coordinates (encoded in time-space points on the line e_0e_2) $g(p)_t$, $g(p)_x$ and $g(p)_y$ for which $F(e_0, e_1, e_2, e_3, M_Q(n_{Canon_{\mathcal{V}}(D)}), a, g(p)_t, g(p)_x, g(p)_y)$ is *true*. This can effectively be written in a FO(Before, minSpeed,$\tilde{S}$)+while-expression as shown in [10] and sketched here:

$$\begin{aligned}
&\forall u_0 \forall u_1 \forall u_2 \forall u_3 \exists a_{00} \exists a_{11} \exists a_{12} \exists a_{21} \exists a_{22} \exists b_0 \exists b_1 \exists b_2 (\mathsf{CoSys}(u_0, u_1, u_2, u_3) \wedge \\
&\phi_{Type_{\mathcal{V}}(D)}(a_{00}, a_{11}, a_{12}, a_{21}, a_{22}, b_0, b_1, b_2) \wedge \\
&\forall p_{t'} \forall p_{x'} \forall p_{y'} (\tilde{F}(u_0, u_1, u_2, u_3, M_Q(n_{Canon_{\mathcal{V}}(D)}), p_a, p_{t'}, p_{x'}, p_{y'}) \\
&\wedge \exists p_t \exists p_x \exists p_y \mathsf{Coordinates}(u_0, u_1, u_2, u_3, p, p_t, p_x, p_y) \wedge \\
&p_{t'} = a_{00} p_t + b_0 \wedge p_{x'} = a_{11} p_x + a_{12} p_y + b_1 \wedge p_{y'} = a_{21} p_x + a_{22} p_y + b_2))
\end{aligned}$$

where $\phi_{Type_{\mathcal{V}}(D)}(a_{00}, a_{11}, a_{12}, a_{21}, a_{22}, b_0, b_1, b_2)$ is a formula that expresses that the transformation described above and parameterized by $(a_{00}, a_{11}, a_{12}, a_{21}, a_{22}, b_0, b_1, b_2)$ is part of $Type_{\mathcal{V}}(D)$.

The wanted FO(Before,minSpeed,$\tilde{S}$)+while program consists of successively executing

$$Encode; Compute; Decode;$$

This concludes the proof. □

6 Concluding remarks

We have given a data model for trajectory data and an efficient way of modeling uncertainty via beads. We have studied transformations for which impor-

tant physical properties of trajectories, such as speed and beads, are invariant. Finally, we have given first-order complete and computationally complete query languages for queries invariant under these transformations.

The results discussed in this paper concern movement in the unrestricted two-dimensional space. In particular beads are defined for free movement in $\mathbf{R}^2$. For practical purposes, this is unrealistic however. In applications of trajectory data, such as traffic management, movement is typically restricted to *road networks*. Currently, we are working on understanding properties of beads on road networks [13]. But the study of speed- and bead-preserving transformations on road networks and of query languages to express queries invariant under such transformations is still an unexplored field.

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] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[3] M. Egenhofer. Approximation of geopatial lifelines. In *SpadaGIS, Workshop on Spatial Data and Geographic Information Systsems*, 2003. Electr. proceedings, 4p.

[4] 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.

[5] F. Geerts, S. Haesevoets, and B. Kuijpers. A theory of spatio-temporal database queries. *ACM Transactions on Computational Logic*, 2008. To appear; see also http://arxiv.org/abs/cs.DB/0503012.

[6] F. Geerts and B. Kuijpers. Linear approximation of planar spatial databases using transitive-closure logic. In *Proceedings of the 19th ACM SIGACT-SIGART-SIGMOD Symposium on Principles of Database Systems (PODS'00)*, pages 126–135. ACM Press, 2000.

[7] F. Geerts, B. Kuijpers, and J. Van den Bussche. Linearization and completeness results for terminating transitive closure queries on spatial databases. *SIAM Journal on Computing*, 35(6):1386–1439, 2006.

[8] 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.

[9] R. Güting and M. Schneider. *Moving Object Databases*. Morgan Kaufmann, 2005.

[10] M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometric query languages. *Journal of Computer and System Sciences*, 58(3):483–511, 1999.

[11] T. Hägerstrand. What about people in regional science? *Papers of the Regional Science Association*, 24:7–21, 1970.

[12] K. Hornsby and M. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1–2):177–194, 2002.

[13] B. Kuijpers and W. Othman. Modeling uncertainty of moving objects on road networks via beads. 2007. Manuscript.

[14] B. Kuijpers and W. Othman. Trajectory databases: Data models, uncertainty and complete query languages. In *Proceedings of the 11th International Conference on Database Theory (ICDT'07)*, volume 4353 of *Lecture Notes in Computer Science*, pages 224–238, 2007.

[15] J. Paredaens, G. Kuper, and L. Libkin, editors. *Constraint databases*. Springer-Verlag, 2000.

[16] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Advances in Spatial Databases (SSD'99)*, volume ???? of *Lecture Notes in Computer Science*, pages 111–132, 1999.

[17] A. D. Polyanin, V. F. Zaitsev, and A. Moussiaux. *Handbook of First Order Partial Differential Equations*. Taylor & Francis, 2002.

[18] P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2002.

[19] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, 1983.

[20] 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.

[21] O. Wolfson. Moving objects information management: The database challenge. In *Proceedings of the 5th Intl. Workshop NGITS*, pages 75–89. Springer, 2002.